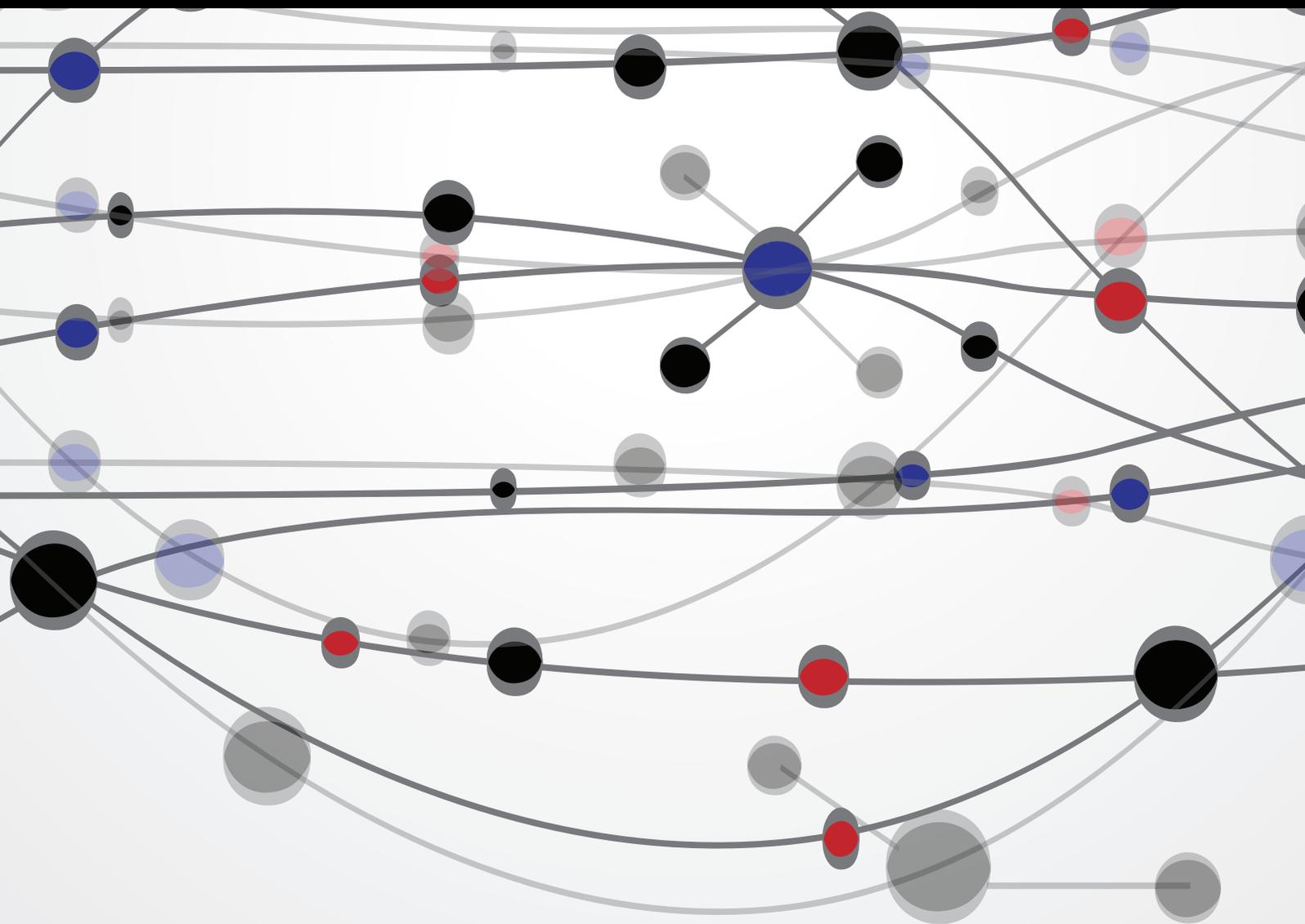


Recent Advances on Bioinspired Computation

Guest Editors: Zhihua Cui, Rajan Alex, Rajendra Akerkar,
and Xin-She Yang





Recent Advances on Bioinspired Computation

The Scientific World Journal

Recent Advances on Bioinspired Computation

Guest Editors: Zhihua Cui, Rajan Alex, Rajendra Akerkar,
and Xin-She Yang



Copyright © 2014 Hindawi Publishing Corporation. All rights reserved.

This is a special issue published in “The Scientific World Journal.” All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Contents

Recent Advances on Bioinspired Computation, Zhihua Cui, Rajan Alex, Rajendra Akerkar, and Xin-She Yang
Volume 2014, Article ID 934890, 3 pages

A High-Performance Genetic Algorithm: Using Traveling Salesman Problem as a Case, Chun-Wei Tsai, Shih-Pang Tseng, Ming-Chao Chiang, Chu-Sing Yang, and Tzung-Pei Hong
Volume 2014, Article ID 178621, 14 pages

Hybrid Algorithms for Fuzzy Reverse Supply Chain Network Design, Z. H. Che, Tzu-An Chiang, Y. C. Kuo, and Zhihua Cui
Volume 2014, Article ID 497109, 16 pages

Evolutionary Computation with Spatial Receding Horizon Control to Minimize Network Coding Resources, Xiao-Bing Hu and Mark S. Leeson
Volume 2014, Article ID 268152, 23 pages

Cooperative Quantum-Behaved Particle Swarm Optimization with Dynamic Varying Search Areas and Lévy Flight Disturbance, Desheng Li
Volume 2014, Article ID 370691, 11 pages

Decomposition-Based Multiobjective Evolutionary Algorithm for Community Detection in Dynamic Social Networks, Jingjing Ma, Jie Liu, Wenping Ma, Maoguo Gong, and Licheng Jiao
Volume 2014, Article ID 402345, 22 pages

A Prefiltered Cuckoo Search Algorithm with Geometric Operators for Solving Sudoku Problems, Ricardo Soto, Broderick Crawford, Cristian Galleguillos, Eric Monfroy, and Fernando Paredes
Volume 2014, Article ID 465359, 12 pages

Self-Adaptive MOEA Feature Selection for Classification of Bankruptcy Prediction Data, A. Gaspar-Cunha, G. Recio, L. Costa, and C. Estébanez
Volume 2014, Article ID 314728, 20 pages

A Hybrid Evolutionary Algorithm for Wheat Blending Problem, Xiang Li, Mohammad Reza Bonyadi, Zbigniew Michalewicz, and Luigi Barone
Volume 2014, Article ID 967254, 13 pages

Designing a Multistage Supply Chain in Cross-Stage Reverse Logistics Environments: Application of Particle Swarm Optimization Algorithms, Tzu-An Chiang, Z. H. Che, and Zhihua Cui
Volume 2014, Article ID 595902, 19 pages

Enhancing Artificial Bee Colony Algorithm with Self-Adaptive Searching Strategy and Artificial Immune Network Operators for Global Optimization, Tinggui Chen and Renbin Xiao
Volume 2014, Article ID 438260, 12 pages

Operational Optimization of Large-Scale Parallel-Unit SWRO Desalination Plant Using Differential Evolution Algorithm, Jian Wang, Xiaolong Wang, Aipeng Jiang, Shu Jiangzhou, and Ping Li
Volume 2014, Article ID 584068, 13 pages

An Adaptive Evolutionary Algorithm for Traveling Salesman Problem with Precedence Constraints, Jinmo Sung and Bongju Jeong
Volume 2014, Article ID 313767, 11 pages

Bioinspired Evolutionary Algorithm Based for Improving Network Coverage in Wireless Sensor Networks, Mohammadjavad Abbasi, Muhammad Shafie Bin Abd Latiff, and Hassan Chizari
Volume 2014, Article ID 839486, 8 pages

A Multipopulation Coevolutionary Strategy for Multiobjective Immune Algorithm, Jiao Shi, Maoguo Gong, Wenping Ma, and Licheng Jiao
Volume 2014, Article ID 539128, 23 pages

Heterogeneous Differential Evolution for Numerical Optimization, Hui Wang, Wenjun Wang, Zhihua Cui, Hui Sun, and Shahryar Rahnamayan
Volume 2014, Article ID 318063, 7 pages

Correction of Faulty Sensors in Phased Array Radars Using Symmetrical Sensor Failure Technique and Cultural Algorithm with Differential Evolution, S. U. Khan, I. M. Qureshi, F. Zaman, B. Shoaib, A. Naveed, and A. Basit
Volume 2014, Article ID 852539, 10 pages

A Novel Harmony Search Algorithm Based on Teaching-Learning Strategies for 0-1 Knapsack Problems, Shouheng Tuo, Longquan Yong, and Fang'an Deng
Volume 2014, Article ID 637412, 19 pages

A Game Theoretical Model for Location of Terror Response Facilities under Capacitated Resources, Lingpeng Meng, Qi Kang, Chuanfeng Han, Weisheng Xu, and Qidi Wu
Volume 2013, Article ID 742845, 10 pages

Naive Bayes-Guided Bat Algorithm for Feature Selection, Ahmed Majid Taha, Aida Mustapha, and Soong-Der Chen
Volume 2013, Article ID 325973, 9 pages

Hybrid Model Based on Genetic Algorithms and SVM Applied to Variable Selection within Fruit Juice Classification, C. Fernandez-Lozano, C. Canto, M. Gestal, J. M. Andrade-Garda, J. R. Rabuñal, J. Dorado, and A. Pazos
Volume 2013, Article ID 982438, 13 pages

A Modified Dynamic Evolving Neural-Fuzzy Approach to Modeling Customer Satisfaction for Affective Design, C. K. Kwong, K. Y. Fung, Huimin Jiang, K. Y. Chan, and Kin Wai Michael Siu
Volume 2013, Article ID 636948, 11 pages

From Nonlinear Optimization to Convex Optimization through Firefly Algorithm and Indirect Approach with Applications to CAD/CAM, Akemi Gálvez and Andrés Iglesias
Volume 2013, Article ID 283919, 10 pages

An Effective Hybrid Firefly Algorithm with Harmony Search for Global Numerical Optimization, Lihong Guo, Gai-Ge Wang, Heqi Wang, and Dinan Wang
Volume 2013, Article ID 125625, 9 pages

Noise-Assisted Concurrent Multipath Traffic Distribution in Ad Hoc Networks, Narun Asvarujanon, Kenji Leibnitz, Naoki Wakamiya, and Masayuki Murata
Volume 2013, Article ID 543718, 10 pages

Editorial

Recent Advances on Bioinspired Computation

Zhihua Cui,^{1,2} Rajan Alex,³ Rajendra Akerkar,⁴ and Xin-She Yang⁵

¹ *Complex System and Computational Intelligence Laboratory, Taiyuan University of Science and Technology, Taiyuan, Shanxi 030024, China*

² *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China*

³ *West Texas A&M University, Canyon, TX 79016, USA*

⁴ *Western Norway Research Institute, 6851 Sogndal, Norway*

⁵ *School of Science and Technology, Middlesex University, London NW4 4BT, UK*

Correspondence should be addressed to Zhihua Cui; cuizhihua@gmail.com

Received 24 February 2014; Accepted 24 February 2014; Published 6 May 2014

Copyright © 2014 Zhihua Cui et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Bioinspired computation (BIC) has become an emerging and popular part of modern computer science, artificial intelligence, computational intelligence, and evolutionary intelligence. BIC is usually based on biological systems that can be often self-organizing, adaptive, and tolerant of random defects. Most BIC algorithms are mostly swarm intelligence (SI) based, and they mimic the successful characteristics of biological and swarm systems. However, bioinspired computation can be very broad, including swarm intelligence [1–7], membrane [8] and memetic computing, DNA and molecular computing, neural computing, and others. Their applications are also very diverse, including computational intelligence, computational neuroscience, bioinformatics, natural language processing, machine learning, software engineering, scheduling and timetabling, data mining, algorithm theory, and many other areas [9–11].

Recent advances in this area are relatively extensive, and a special issue can serve as a good summary to provide a snapshot of the recent advances. The responses to our call for papers are overwhelming, and we received nearly two hundred submissions, and after rigorous peer review, only a small fraction of the submissions were accepted for this special issue. This special issue starts with the heterogeneous differential evolution by J. Wang et al., followed by the correction of faulty sensors in phased array radars by J. U. Khan et al. A new variant of harmony search has been proposed to solve binary knapsack problems; then a theoretical model for location of terror response facilities was studied by

L. P. Meng et al. In addition, a novel variant of bat algorithm based on naïve Bayes methods has been presented to carry out feature selection, followed by a hybrid model based on genetic algorithms and support vector machine for fruit juice classifications by C. Fernandez-Lozano et al., while a modified dynamic neural-fuzzy approach to model customer satisfaction was carried out by C. K. Kwong et al.

On the other hand, an improved firefly algorithm for nonlinear and convex optimization with applications to CAD/CAM has been proposed by A. Gálvez and A. Iglesias, and an effective hybrid by combining the firefly algorithm and harmony search was found to be efficient for global numerical optimization by L. H. Guo et al. Furthermore, noise-assisted multipath traffic distribution networks were studied by N. Asvarujanon et al. Other studies have focussed on the various improvements and diverse applications with detailed case studies. Interested readers are encouraged to read more about this special issue and the open access mode means that you can download papers and study them in great detail when appropriate.

Analyzing the current developments, we can expect to look forward to the future. From the bioinspired computation point of view, we think the following eight areas need more research.

- (i) Complex, real-world applications: more applications should focus on complex, real-world applications in various engineering and industrial designs. Such

applications can be highly nonlinear and multimodal, under stringent constraints.

- (ii) Computationally expensive methods: most applications nowadays are computationally expensive because the evaluations of designs, objectives, and computational tasks are becoming computationally extensive. Many design options have to be evaluated by finite element methods, large-scale finite volume methods, and boundary and extended element methods. For example, applications in aerospace and electromagnetic engineering can take hours up to weeks to evaluate computationally. Therefore, further developments in this area to speed up the computational methods are very important for practical applications.
- (iii) Data-intensive applications: as the data volumes are expanding dramatically, due to drive in information technology and social media, data-intensive applications become more important. Data mining techniques become more relevant, and bioinspired methods such as PSO, cuckoo search, and firefly algorithm become increasingly popular in such applications [1, 9, 12].
- (iv) Network and systems: current research activities have also focused on the modeling and simulation of complex networks and systems such as computer networks, electricity grids, energy networks, and biological systems. Other networks such as social networks have also received some attention. It can be expected that more research will be done in this area.
- (v) Biological applications: bioinspired computation has been applied to biological applications such as protein folding and biological systems [12]. Bioinspired computation has also found applications in bioinformatics and genetic engineering.
- (vi) Combinatorial problems: most combinatorial optimization problems such as the traveling salesman problem are NP-hard and thus are very challenging to solve. Bioinspired algorithms such as ant colony optimization and cuckoo search can be very powerful alternatives for dealing with such challenging problems.
- (vii) Large-scale problems: the current applications tend to have problem sizes with a few, up to a few dozen, design variables, while real-world applications tend to have thousands of design variables. Therefore, more applications are becoming increasingly large-scale, and it is not yet clear if the methods that work for small-scale problems can still work for large-scale problems. For example, protein folding problems can be very large-scale, and at the moment, nature-inspired methods such as simulated annealing and firefly algorithm have been found to be useful and efficient.
- (viii) Data mining and image processing: data mining tends to have data-intensive computation, while image processing can also be very computational extensive as

well. Important applications such as feature selection and classifications need efficient methods to solve, especially those large-scale problems.

As we have seen, bioinspired computation is a very active area, and many challenges still remain. It can be expected that more exciting research activities will be seen in the near future.

Acknowledgments

We, the editors, would like to thank all those who took part in this special issue, including the contributing authors and reviewers. This paper is supported by the National Natural Science Foundation of China under Grant no. 61003053 and the Program for the Top Young Academic Leaders of Higher Learning Institutions of Shanxi.

Zhihua Cui
Rajan Alex
Rajendra Akerkar
Xin-She Yang

References

- [1] X. S. Yang, Z. H. Cui, R. B. Xiao, A. H. Gandomi, and M. Karamanoglu, *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, Elsevier, Waltham, Mass, USA, 2013.
- [2] I. Fister, I. Fister Jr., X. S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, no. 1, pp. 34–46, 2013.
- [3] X. S. Yang and S. Deb, "Cuckoo search: recent advances and applications," *Neural Computing and Applications*, vol. 24, no. 1, pp. 169–174, 2014.
- [4] A. Adamatzky, "Slime mould computes planar shapes," *International Journal of Bio-Inspired Computation*, vol. 4, no. 3, pp. 149–154, 2012.
- [5] P. R. Srivastava, A. Varshney, P. Nama, and X. S. Yang, "Software test effort estimation: a model based on cuckoo search," *International Journal of Bio-Inspired Computation*, vol. 4, no. 5, pp. 278–285, 2012.
- [6] E. García-Gonzalo and J. L. Fernández-Martínez, "A brief historical review of particle swarm optimization (PSO)," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 3–16, 2012.
- [7] Y. Zhang, "Application swarming intelligence in Bioinformatics: survey," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 27–39, 2012.
- [8] K. Q. Jiang, B. S. Song, X. L. Shi, and T. Song, "An overview of membrane computing," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 17–26, 2012.
- [9] X. S. Yang and S. Deb, "Two-stage eagle strategy with differential evolution," *International Journal of Bio-Inspired Computation*, vol. 4, no. 1, pp. 1–5, 2012.
- [10] Z. H. Cui, S. J. Fan, J. C. Zeng, and Z. Z. Shi, "Artificial plan optimisation algorithm with three-period photosynthesis," *International Journal of Bio-Inspired Computation*, vol. 5, no. 2, pp. 133–139, 2013.

- [11] X. S. Yang, M. Karamanoglu, and X. S. He, "Multi-objective flower algorithm for optimization," *Procedia Computer Science*, vol. 18, no. 1, pp. 861–868, 2013.
- [12] B. Maher, A. A. Albrecht, M. Loomes, X. S. Yang, and K. Steinhofel, "A firefly-inspired method for protein structure prediction in lattice models," *Biomolecules*, vol. 4, no. 1, pp. 56–75, 2014.

Research Article

A High-Performance Genetic Algorithm: Using Traveling Salesman Problem as a Case

Chun-Wei Tsai,^{1,2} Shih-Pang Tseng,¹ Ming-Chao Chiang,¹
Chu-Sing Yang,³ and Tzung-Pei Hong^{1,4}

¹ Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan

² Department of Applied Informatics and Multimedia, Chia Nan University of Pharmacy & Science, Tainan 71710, Taiwan

³ Department of Electrical Engineering, National Cheng Kung University, Tainan 70101, Taiwan

⁴ Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 81148, Taiwan

Correspondence should be addressed to Ming-Chao Chiang; mcchiang@cse.nsysu.edu.tw

Received 11 November 2013; Accepted 30 December 2013; Published 5 May 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 Chun-Wei Tsai et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents a simple but efficient algorithm for reducing the computation time of genetic algorithm (GA) and its variants. The proposed algorithm is motivated by the observation that genes common to all the individuals of a GA have a high probability of surviving the evolution and ending up being part of the final solution; as such, they can be saved away to eliminate the redundant computations at the later generations of a GA. To evaluate the performance of the proposed algorithm, we use it not only to solve the traveling salesman problem but also to provide an extensive analysis on the impact it may have on the quality of the end result. Our experimental results indicate that the proposed algorithm can significantly reduce the computation time of GA and GA-based algorithms while limiting the degradation of the quality of the end result to a very small percentage compared to traditional GA.

1. Introduction

In the area of combinatorial optimization research [1], the traveling salesman problem (TSP) [2] has been widely used as a yardstick by which the performance of a new algorithm is evaluated, for TSP is NP-complete [3]. As such, any efficient solution to the TSP can be applied to solve many real world problems, such as transportation control [4], network management [5], and scheduling [6]. Assuming that $d(c_i, c_j)$ represents the distance between each pair of cities c_i and c_j , the TSP asks for a solution—that is, a permutation $\langle c_{\pi(1)}, c_{\pi(2)}, \dots, c_{\pi(n)} \rangle$ of the given n cities—that minimizes

$$D = \left(\sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) \right) + d(c_{\pi(n)}, c_{\pi(1)}). \quad (1)$$

In short, (1) gives the distance D of the tour that starts at city $c_{\pi(1)}$, visits each city in sequence, and then returns directly to $c_{\pi(1)}$ from the last city $c_{\pi(n)}$. Since the brute force method is impractical for the TSP except when the number

of cities is small, the research direction for the TSP has been using heuristic search methods [7–9] to find a near-optimal solution.

Since the 1950s, heuristic algorithms have been developed for finding an approximate solution to the TSP and other complex optimization problems in a reasonable time [10]. Among the most widely used heuristic algorithms are evolutionary algorithms, swarm intelligence, and many others [11–16]. These algorithms eventually have a strong impact on modern computer science research because they help researchers solve problems in a variety of domains for which solutions in their full generality cannot be found in a reasonable time, even with the world's fastest computers. For reasons such as being an inherently parallel algorithm, being global search heuristics, and being easy to implement, GA [17, 18] has nowadays become one of the most popular heuristic algorithms. Moreover, Holland's schema theorem [17], which says that "short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a GA" and Goldberg's building

block hypothesis [18], which says that “a GA seeks near-optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks” tell us that good subsolutions (or partial solutions) of a GA have a high probability of surviving the evolution and ending up being part of the final solution. This is further confirmed by Glover’s proximate optimality principle (POP) [19], which says that “good solutions at one level are likely to be found close to good solutions at an adjacent level,” or good solutions have similar structures. A crucial observation above is that good subsolutions of a GA (or simply GA) (since no confusion is possible, we will use GA to represent simple or traditional GA throughout this paper) will become more and more similar to each other during its evolution process. This, in turn, implies that many of the computations of good subsolutions at the later generations of a GA are essentially redundant. The question is how do we eliminate these redundant computations at the early generations of a GA so that the computation time can be significantly reduced while at the same time retaining or enhancing the quality of the end result.

To make the idea more concrete, a simple example is given in Figure 1 to demonstrate how it works. As Figure 1 shows, let us suppose that there are two chromosomes, C_1 and C_2 , each of which is composed of ℓ genes. Let us further suppose that $\ell = 4$, and each gene can take only two possible values, namely, 0 and 1. Now let us assume, at a certain point in the evolution process, that the value of C_1 is 0-0-1-0, and the value of C_2 is 1-1-1-0 where the hyphen is used to separate the genes. Then what would be the values of C_1 and C_2 in the later generations? There are two answers to this question, depending on how the mutation operator is treated. The first answer is that if we use one point crossover and disregard the mutation operator altogether, then we are guaranteed that the values of the third and fourth genes of C_1 and C_2 will remain intact in the evolution process of a GA and will thus show up as part of the final solution. In other words, if the third and fourth genes of C_1 and C_2 (i.e., genes common to C_1 and C_2) are saved away, the number of genes will be cut into half and the computation time required by the crossover and mutation operators and the evaluation of the fitness function will be reduced. The second answer is that if we take into account the mutation operator, then the values of the third and fourth genes of C_1 and C_2 would have a small chance of not being 1-0. The probability for the values of the third and fourth genes of C_1 and C_2 being changed is, however, very small because only the mutation operator is allowed to change their values, and for GA, the mutation rate has almost always been set to a very small value, say, 1 or 2 percent.

The remainder of the paper is organized as follows. Section 2 gives a brief introduction to the genetic algorithm and the approaches taken to enhance its performance. Section 3 provides a detailed description of the proposed algorithm and a simple example to demonstrate how the proposed algorithm works. Performance evaluation of the proposed algorithm is presented in Section 4. Analysis of the proposed algorithm is given in Section 5. Conclusion is drawn in Section 6.

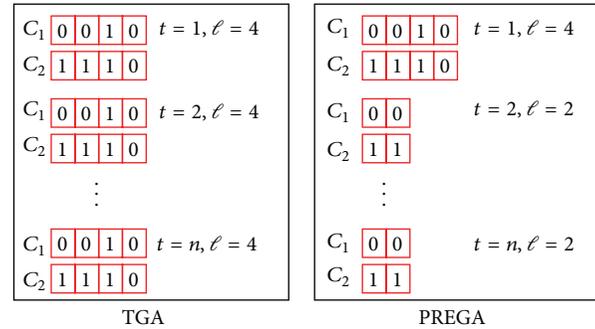


FIGURE 1: A simple example illustrating the difference between GA and PREGA. Note that genes common to chromosomes C_1 and C_2 are saved away by PREGA at generation $t = 1$ but not by TGA.

2. Related Work

As a particular class of evolutionary algorithms, it is well known that GA is a search technique aimed at finding true or approximate solutions to optimization problems. The operations used to emulate the evolution process of a GA are *selection*, *crossover*, and *mutation*. The simple or traditional GA [18] can be outlined as given in Algorithm 1. The selection operator takes the responsibility of guiding the search of GA toward the high quality or even optimal solution. The crossover operator plays the role of exchanging the information between the individuals in the population while the mutation operator is used to avoid GA from falling into local optima.

Researches on genetic algorithms focus not only on improving the quality of the end result but also on reducing the computation time of GA. Among them are parallel genetic algorithm, hybrid genetic algorithm, and radical modification of the evolutionary procedure or the design of GA.

- (1) Parallel genetic algorithm (PGA) [20, 21] is a very important technique for reducing the computation time of large problems, such as TSP [22]. The three distribution models [23] that have been proposed are master-slave model, fine-grained model (cellular model), and coarse-grained model (island model). However, in [24, 25], the authors indicate that the migration rate and strategy of the island model may affect its performance.
- (2) Hybrid genetic algorithm (HGA) [26] refers to the process of combining GA with other effective approaches for finding a better solution in terms of either the quality or the computation time. In general, the design of HGA may either integrate other heuristic algorithms [27] or combine local search methods [28] with GA. For instance, for an HGA that is a combination of GA and a local search method, GA is responsible for finding the global minima or pointing out the particular direction that may lead to a better solution while the local search method is used to find the local minima. For this reason, HGA will enhance the quality of the end result.

GA1. Randomly generate an initial population of chromosomes.
 GA2. Use the fitness function to select the fitter chromosomes.
 GA3. Apply the crossover and mutation operators in order.
 GA4. If a stopping criterion is satisfied, then stop and output the best chromosome.
 GA5. Go to step 2.

ALGORITHM 1: Outline of traditional genetic algorithm (GA).

In the research [29] on fast HGA (FHGA), Misevicius [29] points out that the design of FHGA should satisfy the following principles: (1) FHGA should arrive at the solutions quickly; (2) the populations should be compact to save the computation time; and (3) the diversity of the populations has to be maintained to avoid falling into the local minimum at early generations in the evolution process.

- (3) Another way to reduce the computation time is to radically change the evolutionary procedure or the design of GA. Michalski [30] presents a non-Darwinian-type evolution called learnable evolution model (LEM) that divides the whole population into two groups: high-performance group (H-group) and low-performance group (L-group). LEM first finds descriptions about why the H-group can obtain a better result and why the L-group may degrade the quality of the end result. Then, it uses the descriptions to generate chromosomes to replace those in L-group. Michalski also points out that LEM can speed up the number of evolutionary steps by a factor of two or more. Yet, when this kind of fast convergence methods [30, 31] of GA is used, it should be very careful about the convergence speed, or it may face the *premature convergence* problem. One possible solution to this problem is to use the fitness sharing [32] to avoid the diversity of the population being cut down too early.

The improvements that the abovementioned methods can achieve are limited intrinsically by the operators of GA. For example, the crossover and other genetic operators may disrupt the high quality subsolutions (building blocks, or BBs for short) that are found in the previous generations [33]. As a result, the convergence time of GA may increase [34]. Over the past two decades or so, various competent genetic algorithms (competent GAs) [33, 35–37] have been developed to tackle the linkage and scalability problem of GA. They can be broadly divided into two classes [36]. Also referred to as the perturbation technology, the first class is based on evolving the representation of solutions or adapting recombination operators among individual solutions. Among this class are the messy genetic algorithm, fast messy genetic algorithm (fmGA) [33, 38], and ordering messy GA (OmeGA) [36]. The fmGA differs from simple GA in several aspects. (1) Each gene of the fmGA is represented by its value and locus. (2) The fmGA uses variable-length chromosomes to represent the population. (3) The fmGA attempts to find

the building blocks by repeatedly performing selection of solutions and random deletion of genes [36]. (4) A so-called *competitive template* is required to fill up the missing genes of underspecified messy chromosomes so that the fitness values can be evaluated.

3. The Proposed Algorithm

In this section, we present a simple but efficient technique for eliminating the redundant computations of GA and GA-based algorithms based on the notion of pattern reduction. Algorithm 2 gives an outline of the pattern reduction enhanced genetic algorithm (PREGA). As Algorithm 2 shows, PREGA is built on the framework of GA; thus, it can be considered as an enhancement of GA with two operators—the common genes detection (CGD) and common genes compression (CGC) operators. If we disregard steps 3 and 4, PREGA given in Algorithm 2 will fall back into GA, as shown in Algorithm 1. The underlying idea of PREGA is to detect and compress genes common to all the chromosomes at the early generations of a GA to eliminate the redundant computations at the later iterations in the evolution process. In what follows, we will give a detailed description of the proposed algorithm.

3.1. Common Genes Detection (CGD). The common genes detection operator of PREGA is responsible for detecting genes that are common to all the individuals in the population and thus are unlikely to be changed at later generations of the GA. Nevertheless, for different problems, the representation of chromosomes may have to be modified or even redesigned. From a different point of view, the example given in Figure 1 can be considered as a special case in terms of the fact that all the genes encode only two possible values 0 and 1 and all the genes are uncorrelated. In some other situations, such as traveling salesman problem, however, the solution of each gene will certainly affect the other genes, and genes on the same position of all the chromosomes do not necessarily represent identical subsolutions. For the TSP, each chromosome can be used to encode a different tour, that is, a different permutation $\langle c_{\pi(1)}, c_{\pi(2)}, \dots, c_{\pi(n)} \rangle$ of the given cities. In other words, for all the chromosomes, $i \neq j$ —for all i and j , $1 \leq i \neq j \leq m$ —implies $C_i \neq C_j$. Alternatively, each chromosome can be used to encode edges (corresponding to roads connecting pairs of cities) connecting pairs of cities of a tour.

In this paper, we use binary encoding for finding edges common to all the chromosomes. First, let us suppose that $E(i, j)$ (here, we are assuming that node i , for all i , $1 \leq i \leq m$,

PREGA1. Randomly generate an initial population of chromosomes.
 PREGA2. Use the fitness function to select the fitter chromosomes.
 PREGA3. **Apply common genes detection (CGD) algorithm to find the common genes.**
 PREGA4. **Apply common genes compression (CGC) algorithm to reserve the common genes.**
 PREGA5. Apply the crossover and mutation operators in order.
 PREGA6. If a stopping criterion is satisfied, then stop and output the best chromosome.
 PREGA7. Go to step 2.

ALGORITHM 2: Outline of pattern reduction enhanced genetic algorithm (PREGA).

in the graph G representing the TSP, is labeled by city c_i . Thus, we will use i and c_i interchangeably) is the edge connecting the pair of cities c_i and c_j . Without loss of generality, let us further suppose that $i < j$. Otherwise, we can swap c_i and c_j or i and j , since insofar as this paper is concerned, only the symmetric TSP is considered. Then, all the $N = n(n - 1)/2$ edges $E(i, j)$, $1 \leq i < j \leq n$, can be assigned unique numbers in the range of 0 to $N - 1$, which can be computed as $(2n - i)(i - 1)/2 + (j - i - 1)$ (it follows from $(\sum_{k=n-i+1}^{n-1} k) + (j - i - 1)$ by simple algebraic manipulation).

To make the idea more concrete, (2) gives an example to show how all the $7(7 - 1)/2 = 21$ edges $E(i, j)$, $1 \leq i < j \leq 7$, are assigned unique numbers in the range of 0 to 20;

$$i \begin{pmatrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & - & 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & - & - & 6 & 7 & 8 & 9 & 10 \\ 3 & - & - & - & 11 & 12 & 13 & 14 \\ 4 & - & - & - & - & 15 & 16 & 17 \\ 5 & - & - & - & - & - & 18 & 19 \\ 6 & - & - & - & - & - & - & 20 \\ 7 & - & - & - & - & - & - & - \end{pmatrix}. \quad (2)$$

As the example shows, $E(1, 2)$ is assigned the number 0, $E(1, 3)$ the number 1, $E(1, 4)$ the number 2, and so on all the way up until $E(6, 7)$ is assigned the number 20. In other words, all the $n(n - 1)/2$ edges can be mapped to a one-dimensional array with exactly $n(n - 1)/2$ elements. This would save a little bit more than half of the space or more precisely $n(n + 1)/2$ entries. Now, to find edges common to all the chromosomes, we apply the common genes detection algorithm given in Algorithm 3.

Obviously, as Algorithm 3 shows, steps 1 and 3 take $O(n^2)$ time, and step 2 takes $O(mn) = O(n)$ time assuming that m is a constant. Thus, both the time and space complexities of the CGD algorithm are $O(n^2)$, as claimed. It is worth mentioning that the CGD algorithm described in Algorithm 3 can be made even more efficient if we keep track of in a stack or an array (of size no more than n) the edges common to all the chromosomes in step 2 when the *last* chromosome is being scanned; then step 3 can be eliminated altogether. If we go one step further, eventually, the CGD algorithm can be made much more efficient and scalable than as outlined in Algorithm 3 by using a more complicated data structure such as balanced trees (the basic operations of which—such as member, insert, and delete—take $O(\log n)$ time where

n is the number of nodes in the tree). Again, assuming that m is a constant and that a balanced tree is used, the time complexity of the CGD algorithm can be cut from $O(n^2)$ down to $O(mn \log mn) = O(n \log n)$ and the space complexity from $O(n^2)$ down to $O(mn) = O(n)$, as claimed. As the number of generations increases, the number of cities n will be quickly decreased. This implies that the CGD operator is in general much faster than specified by the above bounds, which will in turn enhance the performance of the CGC operator to be discussed next.

3.2. *Common Genes Compression (CGC).* The common genes compression operator of PREGA is responsible for compressing and removing the common genes detected by CGD. As outlined in Algorithm 4, the CGC algorithm will first compress the common genes detected by CGD—by choosing a representative for and saving away the information associated with all or each segment of the common genes depending on the applications—and then remove the common genes compressed so that later generations of the GA will only see the chosen representatives. A less number of genes are used to represent the common genes each of which represents a segment of the common genes. For instance, using TSP as an example and assuming that the common genes detected c_3 , c_4 , and c_5 form a segment of the path, then these genes—and the information associated with them such as the segment of the path they form as well as the length and direction of the segment—can be compressed, that is, represented by a single composite gene, say, c'_3 . Once this is done, GA will see only the gene c'_3 at later generations during its convergence process. In other words, each detected segment of the path can be represented by a single composite gene, which is independent of the number of cities of which each segment of the path is composed. Moreover, all the composite genes can be compressed again as the other “noncomposite” genes. It is worth mentioning that we have to take into consideration the relationships between subsolutions to see if they are dependent or independent before they are compressed. If they are independent, all the common genes can be compressed into a single gene. Otherwise, how they are compressed depends on the problem in question and the way the solutions are encoded.

3.3. *An Example.* In this section, we present a simple example to illustrate exactly how PREGA works for the TSP. As Figure 2 shows, the very first step of PREGA is exactly the

CGD1. Initialize the values of all the array elements to 0.
 CGD2. For each chromosome, we scan from left to right all the edges encoded within it, calculate the index for each edge scanned, and increment the value of the corresponding array element by one.
 CGD3. The result array is scanned from left to right looking for all the elements whose values are equal to m . Edges corresponding to indices to these array elements are common to all the chromosomes.

ALGORITHM 3: Outline of common genes detection algorithm.

CGC1. Compress the common genes detected by CGD—by choosing a representative for, and saving away the information associated with, each segment of the common genes.
 CGC2. Remove the common genes compressed in step 1 so that the later generations of the GA will only see the representatives chosen in step 1.

ALGORITHM 4: Outline of common genes compression algorithm.

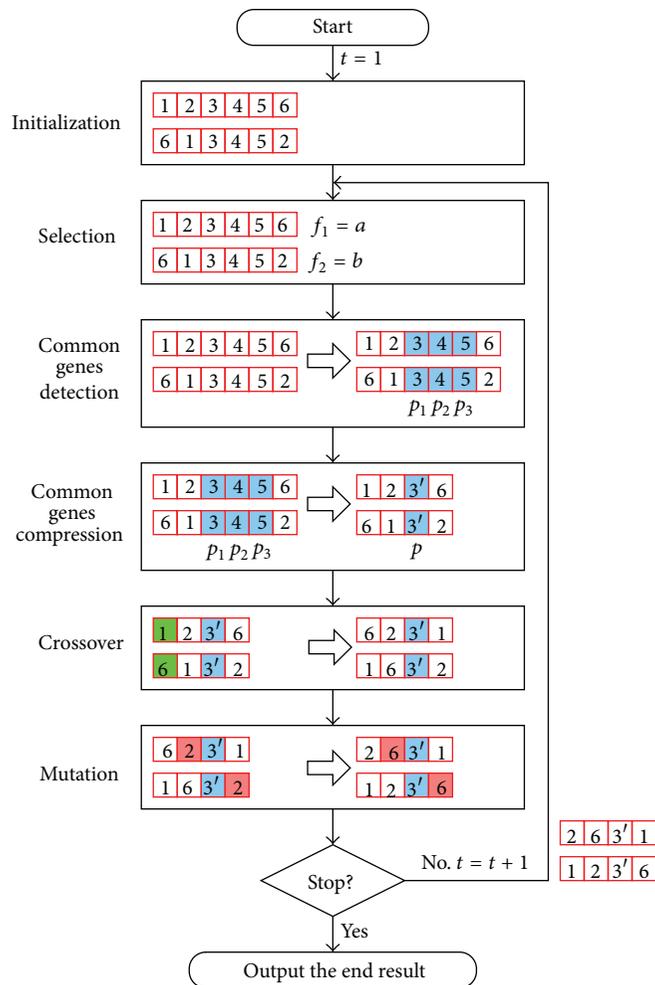


FIGURE 2: A simple example illustrating how PREGA works. See the text for more detailed explanation.

same as that of GA and is to randomly generate a population of chromosomes. For the purpose of illustration, a population of two chromosomes is generated in this case, and each gene is randomly assigned a distinct city number. Then, the selection operator is applied to select the “good” chromosomes in terms of the fitness value f_i of each chromosome. Then, the CGD and CGC operators, as described in Section 3, are applied for the detection and compression of the genes.

As Figure 2 shows, PREGA differs from GA by adding the CGD and CGC operators as described in Section 3 to eliminate the redundant computations encountered by GA. By doing this, the performance of GA can be significantly enhanced. The example given in Figure 2 shows that the common genes indicated by p_1 , p_2 , and p_3 are first detected by the CGD operator of PREGA and then compressed by the CGC operator of PREGA, which is denoted by p . In other words, after compression, we can choose either one of the three common genes 3, 4, and 5 as the representative to indicate the segment compressed. In this case, we choose 3. To avoid confusion, we use $3'$ instead of 3 in Figure 2. After that, the crossover and mutation operators as well as the evaluation of the fitness function will treat each compressed segment as a single pattern until the terminal condition is met. Note that if the genes detected are consecutive, they will be compressed into a single gene. Otherwise, they will be compressed into as few genes as needed; that is, they will be compressed segment by segment.

4. Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm by using it to solve the traveling salesman problem. The empirical analysis was conducted on an IBM X3400 machine with 2.0 GHz Xeon CPU and 8 GB of memory using CentOS 5.0 running Linux 2.6.18. All the programs are written in C++ and compiled using g++ (GNU C++ compiler). The benchmarks for the TSP are shown in Table 1. Unless stated otherwise, all the simulations are carried out for 30 runs, with the population size fixed at 80, the crossover probability at 0.5, the per-gene mutation probability at 0.01, the number of generations at 100, and the tournament size at 3 (i.e., 1 out of 3). For all the simulations, PR is started at the second generation.

To improve the quality of the end results of GA, PR, and other evolutionary algorithms, we use several useful technologies to solve the TSP. The nearest-neighbor method [39] is used in creating the initial solution for all the algorithms involved in the simulation. The 2-opt mutation operator [40] is employed as the local search method for fine-tuning the quality of the end results. Unless stated otherwise, all the simulations use HX as the crossover operator by default.

To simplify the discussion of the simulation results of TSP in Tables 2, 3, and 4, we will use the following conventions. Let TGA (traditional GA) [41], HeSEA (heterogeneous selection evolutionary algorithm) [42], SA (simulated annealing) [10], UMDA (univariate marginal distribution algorithm) [43], EHBSA (edge histogram based sampling) [44], ACS (ant colony system) [45], DPSO (discrete particle swarm

TABLE 1: Data sets for TSP.

Data set	Number of cities	Optimum
ch130	130	6,110
ch150	150	6,528
d198	198	15,780
a280	280	2,579
pcb442	442	50,778
d493	493	35,002
u574	574	36,905
u724	724	41,910
pr1002	1,002	259,045
u1060	1,060	224,094
d1291	1,291	50,801
u1432	1,432	152,970
d1655	1,655	62,128
u2152	2,152	64,253
pr2392	2,392	378,032
pcb3038	3,038	137,694
fnl4461	4,461	182,566
usa13509	13,509	19,982,889

optimization) [46], and PREGA denote algorithms involved in the simulation. Let $\beta \in \{D, T\}$ denote either the traveling distance ($\beta = D$) or the computation time ($\beta = T$). Let Δ_β denote the enhancement of β_ϕ with respect to β_ψ in percentage. Δ_β is defined as follows:

$$\Delta_\beta = \frac{\beta_\phi - \beta_\psi}{\beta_\psi} \times 100\%, \quad (3)$$

where β is either D or T for the TSP, and the subscripts ϕ and ψ are defined as follows.

- (i) For Table 2, $\phi = \text{PREGA}(x)$ implies $\psi = \text{TGA}(x)$, where x denotes the crossover operators [47–49] in use and is either partially matched crossover (PMX), order crossover (OX), heuristic crossover (HX), or edge-recombination crossover (ERX).
- (ii) For Tables 3 and 4, $\psi = \text{TGA}$, HeSEA, SA, UMDA, EHBSA, ACS, or DPSO, and $\phi = \text{PRE}\psi$. Note that in Table 3, to simplify the description, we use PRETGA to indicate PREGA.

Note that for $\beta \in \{D, T\}$, the more negative the value of Δ_β , the greater the enhancement.

4.1. Impact of Different Removal Strategies. To better understand the impact of the removal bound on the performance of PREGA, we tested several removal bounds—from 0% to 100% with an increment of 10%. 100% means that PREGA may reduce all the genes of chromosomes in the convergence process, whereas 0% means that no genes will be removed; and thus PREGA falls back to GA. More precisely, to simplify the implementation, what we have done is that, after step 2 but before step 3 as shown in Algorithm 2, we check to see if the removal bound is exceeded. If it is exceeded, then

TABLE 2: Simulation results of using different crossover operators.

Data set	PREGA(PMX)			PREGA(OX)			PREGA(HX)			PREGA(ERX)		
	Δ_D	c_v	Δ_T	Δ_D	c_v	Δ_T	Δ_D	c_v	Δ_T	Δ_D	c_v	Δ_T
ch130	0.92	(1.49%)	-80.00	0.85	(1.25%)	-81.82	1.56	(1.51%)	-87.50	1.95	(1.50%)	-85.71
ch150	1.67	(0.62%)	-84.62	1.53	(0.69%)	-85.71	1.43	(0.59%)	-90.00	1.31	(0.51%)	-88.24
d198	0.80	(1.05%)	-86.36	0.44	(1.46%)	-86.96	0.42	(0.90%)	-90.91	0.60	(0.81%)	-89.29
a280	0.39	(1.73%)	-86.36	0.09	(1.63%)	-91.11	-0.71	(1.68%)	-91.80	0.50	(1.37%)	-90.38
pcb442	1.18	(0.66%)	-90.18	0.55	(0.72%)	-92.98	-2.59	(0.69%)	-93.75	1.96	(0.99%)	-91.2
d493	2.74	(4.39%)	-81.76	-1.15	(1.04%)	-92.62	-2.67	(2.00%)	-92.55	1.16	(1.10%)	-90.18
u574	-1.05	(0.80%)	-91.17	-2.23	(0.59%)	-95.10	-5.51	(0.84%)	-94.66	0.95	(0.69%)	-93.38
u724	-0.78	(0.75%)	-87.41	-2.26	(0.64%)	-95.02	-5.11	(2.39%)	-94.58	1.22	(0.70%)	-93.16
pr1002	-1.48	(0.58%)	-91.94	-2.38	(0.65%)	-95.77	-5.08	(2.79%)	-94.84	0.81	(0.79%)	-94.79
u1060	-1.14	(0.99%)	-89.69	-2.87	(0.63%)	-95.52	-3.69	(4.31%)	-94.55	1.18	(0.98%)	-94.10
d1291	0.17	(0.74%)	-93.80	-0.10	(0.95%)	-96.43	-3.38	(0.86%)	-95.91	0.18	(1.04%)	-95.93
u1432	3.53	(4.36%)	-80.50	-3.68	(0.50%)	-95.19	-2.76	(4.20%)	-93.74	1.07	(1.00%)	-93.36
d1655	-0.85	(1.75%)	-91.79	-2.06	(1.10%)	-96.32	-3.39	(2.57%)	-95.47	0.98	(0.92%)	-95.36
u2152	-0.77	(2.67%)	-89.53	-2.05	(0.52%)	-96.38	-3.00	(2.41%)	-95.71	0.78	(0.52%)	-95.69
Average	0.38		-87.51	-1.09		-92.64	-2.46		-93.28	1.05		-92.20
Data set	PREGA(SBOX)			PREGA(SJOX)			PREGA(SB2OX)			PREGA(SJ2OX)		
	Δ_D	c_v	Δ_T	Δ_D	c_v	Δ_T	Δ_D	c_v	Δ_T	Δ_D	c_v	Δ_T
ch130	1.03	(1.51%)	-92.65	0.93	(1.54%)	-91.24	1.02	(1.45%)	-91.11	1.37	(1.71%)	-90.33
ch150	1.35	(0.58%)	-92.60	1.58	(0.56%)	-92.33	1.57	(0.55%)	-92.66	1.41	(0.56%)	-92.48
d198	0.55	(1.00%)	-92.36	0.85	(0.92%)	-91.81	0.67	(0.85%)	-90.78	0.86	(0.82%)	-90.17
a280	0.84	(1.25%)	-92.10	1.55	(1.58%)	-93.13	-0.11	(1.67%)	-91.58	0.68	(1.50%)	-91.98
pcb442	1.81	(0.94%)	-93.48	1.50	(0.80%)	-93.02	0.95	(0.89%)	-93.02	0.80	(1.03%)	-93.21
d493	1.07	(0.97%)	-92.63	0.85	(1.07%)	-92.98	0.16	(0.75%)	-91.61	0.63	(0.82%)	-90.77
u574	-0.13	(0.85%)	-92.52	0.06	(0.90%)	-93.07	-1.17	(0.64%)	-92.28	-1.31	(0.80%)	-92.45
u724	0.75	(0.79%)	-93.78	0.11	(0.57%)	-93.42	-0.85	(0.71%)	-93.70	-1.00	(0.73%)	-93.39
pr1002	-0.17	(0.50%)	-94.06	-0.26	(0.64%)	-93.64	-1.65	(0.69%)	-94.17	-1.76	(0.74%)	-93.85
u1060	-0.09	(0.63%)	-93.35	0.09	(0.72%)	-94.13	-1.39	(0.51%)	-93.11	-2.05	(0.93%)	-92.00
d1291	0.55	(1.08%)	-93.49	0.48	(0.92%)	-92.68	-0.22	(0.79%)	-92.78	-0.19	(0.82%)	-92.69
u1432	-0.20	(0.74%)	-93.41	-0.11	(0.73%)	-93.13	-2.08	(0.48%)	-93.48	-2.18	(0.77%)	-93.49
d1655	-0.22	(0.62%)	-94.08	-0.55	(0.65%)	-94.13	-1.60	(0.86%)	-94.29	-1.02	(0.91%)	-94.2
u2152	-0.18	(0.55%)	-94.78	-0.26	(0.57%)	-94.57	-0.38	(0.48%)	-94.67	-0.66	(0.52%)	-94.54
Average	0.50		-93.23	0.49		-93.09	-0.36		-92.80	-0.32		-92.54

T: time in seconds; c_v : coefficient of variation, which is defined to be $c_v = \sigma/\mu$, where μ is either D or Δ_D .

steps 3 and 4 will be bypassed. Otherwise, all the common genes detected at step 3 will be removed at step 4 even if it will exceed the removal bound. In other words, we may end up removing a few more genes than the removal bound says.

The experimental results showed that setting the removal bound to 0% (GA) or 100% is better than the others. Although setting the removal bound to 10%, 20%, and up to 90% can also reduce the computation time, setting the removal bound to 100% seems to give a good balance between the computation time and the quality of the end results. It shows that PREGA using 100% removal bound can obtain the best results compared to the other removal bound settings, that is, 10%, 20%, and up to 90%.

A very interesting result to be paid particular attention is that the end result of PREGA using 100% removal bound is better than the others. This result shows that the quality of

the PREGA is not linearly proportional to the removal bound. The main reason for this phenomenon is that the local search has to be split into two parts: one is for the common genes and the other is for the noncommon genes. This is required because the common genes have been compressed and thus cannot be mixed up with the noncommon genes. Otherwise, the common genes will become noncommon genes. This situation eventually affects the ability of the local search methods. In other words, with 100% and 0% removal bounds, the search ability of the local search methods is maximized because either all of the genes are either common or noncommon. In the case of 10%, 20%, and up to 90%, however, all the chromosomes are composed of two parts, thus limiting the local search methods to find better subsolutions in a smaller search space instead of the whole search space. This will degrade the quality of the end results, causing the quality of

TABLE 3: Simulation results of GA, HeSEA, LEM, and SA.

Data set	PREGA			PREHeSEA			PRELEM			PRESA		
	Δ_D	c_v	Δ_T									
a280 (c_v)	-0.71	(1.68%)	-91.84	0.22	(1.47%)	-94.79	1.17	(1.90%)	-95.92	-0.07	(1.48%)	-57.62
b_{30}	2,679.44			2,681.44			2,671.60			2,743.77		
u574 (c_v)	-5.51	(0.84%)	-94.76	-5.88	(0.65%)	-93.14	1.25	(0.83%)	-95.92	0.02	(1.17%)	-52.54
b_{30}	38,837.70			39,211.40			39,156.40			39,448.30		
u724 (c_v)	-5.12	(2.39%)	-94.61	-3.95	(0.77%)	-93.75	1.04	(0.89%)	-96.14	-0.27	(0.95%)	-52.68
b_{30}	44,399.10			44,347.70			44,388.70			45,044.90		
u1060 (c_v)	-3.69	(4.31%)	-94.55	-5.52	(2.28%)	-93.97	0.85	(0.77%)	-96.25	-0.29	(1.01%)	-49.33
b_{30}	241,758.00			238,428.00			237,896.00			238,928.00		
u1432 (c_v)	-2.76	(4.20%)	-93.76	-0.45	(3.80%)	-93.78	1.27	(0.80%)	-95.97	0.13	(0.75%)	-53.26
b_{30}	164,093.00			165,570.00			162,950.00			162,973.00		
pr2392 (c_v)	-3.32	(3.34%)	-95.06	-5.58	(2.43%)	-95.89	0.14	(0.46%)	-96.61	-0.25	(0.58%)	-53.20
b_{30}	405,470.00			405,612.00			407,597.00			419,796.00		
pcb3038 (c_v)	-0.55	(4.64%)	-94.72	-0.97	(4.22%)	-95.34	0.40	(0.66%)	-95.73	-0.05	(0.73%)	-59.18
b_{30}	148,567.00			148,293.00			148,353.00			152,449.00		
fnl4461 (c_v)	-4.09	(4.52%)	-95.32	-1.02	(3.06%)	-95.58	0.00	(0.51%)	-95.19	0.12	(0.36%)	-61.21
b_{30}	195,074.00			195,938.00			195,907.00			202,525.00		
usa13509 (c_v)	-0.77	(5.23%)	-92.18	6.09	(0.42%)	-95.47	-0.17	(0.28%)	-94.89	-1.41	(0.46%)	-85.03
b_{30}	21,500,000.00			23,700,000.00			21,900,000.00			22,600,000.00		
Average	-2.95		-94.09	-1.90		-94.63	0.66		-95.85	-0.23		-58.23

T: time in seconds; b_{30} : best solution in 30 runs; c_v : coefficient of variation as defined in Table 2.

the end results of PREGA to be not linearly proportional to the removal bound.

4.2. Impact of Different Kind of Crossover Operators. There are several different crossover operators [48, 49] for the TSP, such as PMX, OX, ERX, and HX. PMX is the most popular and simplest crossover operator, but it lacks searching direction. More recently, many researchers have focused their attention on finding and keeping the building blocks to enhance the performance of GA by either modifying or replacing the operators of GA. In [50], Ruiz et al. designed new crossover operators to identify and maintain the building blocks. In this paper, we use the PMX, OX, HX, and ERX operators to examine the search ability of PREGA when different crossover methods are used. In addition, we have also tested the crossover operators SBOX, SJOX, SB2OX, and SJ2OX [50] to better understand the performance of PREGA with other efficient crossover operators that are designed to avoid disrupting the building blocks on the convergence process. Note that, for the TSP in this paper, we use the *2-opt mutation* method for reversing two segments (the size of which must be the same) of a tour encoded in a chromosome. For each segment, the edges to the left and right of that segment (if we consider a chromosome as a ring, then the last gene will be

next to the first gene or vice versa, and thus there is always a gene to the left or right of a segment) will be replaced by two new edges.

As Table 2 shows, for the TSP, PREGA can effectively reduce the computation time from 80% up to 93.8% using PMX, from 81.82% up to 96.43% using OX, from 87.50% up to 95.91% using HX, from 85.71% up to 95.93% using ERX, from 92.10% up to 94.78% using SBOX, from 91.24% up to 94.57% using SJOX, from 90.78% up to 94.67% using SB2OX, and from 90.17% up to 94.54% using SJ2OX compared to those of traditional GA and GA-based algorithms alone. The simulation results further show that not only does PREGA preserve the accuracy rate of the end results, but also it can even give solutions that are better than those found by the traditional GA and GA-based algorithms alone.

The amount of time that can be reduced and the end results that can be improved depend, to a large extent, on the size of the problem. Our simulation results indicate that the larger the problem, the better the performance of the proposed algorithm. Table 2 also shows that PREGA can even improve the performance of most of the crossover operators, including the crossover operator as complex as HX. This can be easily justified by the following observation. The more complex the crossover operators, the more the computation

TABLE 4: Simulation results of UMDA, EHBSA, ACS, and DPSO.

Data set	PREUMDA			PREEHBSA			PREACS			PREDPDSO		
	Δ_D	c_v	Δ_T									
a280 (c_v)	6.73	(3.14%)	-66.00	1.59	(1.85%)	-69.82	-3.31	(1.24%)	-22.14	-5.13	(2.56%)	-75.00
b_{30}	2,722.10			2,687.78			2,625.38			2,701.27		
u574 (c_v)	-4.52	(0.71%)	-58.61	0.52	(0.80%)	-67.46	0.64	(0.86%)	-69.01	-7.85	(4.25%)	-84.39
b_{30}	39,013.00			39,392.90			38,299.70			39,031.40		
u724 (c_v)	-4.14	(0.67%)	-59.47	0.39	(0.78%)	-67.95	-0.85	(0.77%)	-78.31	-4.20	(4.19%)	-84.66
b_{30}	44,419.90			44,347.50			42,805.40			44,854.00		
u1060 (c_v)	-4.76	(0.70%)	-73.35	-0.09	(0.76%)	-74.11	1.46	(0.66%)	-80.67	-1.35	(3.24%)	-87.93
b_{30}	239,226.00			238,155.00			234,396.00			242,863.00		
u1432 (c_v)	-2.93	(0.65%)	-70.61	0.51	(0.67%)	-73.32	0.29	(0.66%)	-73.85	-0.11	(0.65%)	-76.42
b_{30}	163,520.00			163,504.00			161,204.00			183,637.00		
pr2392 (c_v)	-3.51	(0.46%)	-81.97	-0.90	(0.52%)	-75.59	2.51	(0.79%)	-89.26	-0.68	(0.42%)	-91.16
b_{30}	406,302.00			404,867.00			401,594.00			452,346.00		
pcb3038 (c_v)	-3.36	(0.33%)	-69.17	-0.86	(0.32%)	-76.67	4.58	(0.57%)	-91.07	-0.13	(0.55%)	-86.84
b_{30}	148,258.00			148,374.00			149,715.00			165,307.00		
fnl4461 (c_v)	-3.60	(0.22%)	-73.20	-1.46	(0.29%)	-79.02	2.09	(0.63%)	-92.30	-0.16	(0.38%)	-86.35
b_{30}	195,063.00			195,747.00			200,487.00			219,180.00		
usal3509 (c_v)	-4.14	(0.27%)	-70.52	-2.39	(0.26%)	-73.01	1.16	(0.34%)	-93.45	-0.49	(0.31%)	-88.24
b_{30}	21,479,200.00			21,531,300.00			22,584,600.00			24,222,100.00		
Average	-2.69		-69.21	0.3		-73.00	0.95		-76.67	-2.23		-84.55

T: time in seconds; b_{30} : best solution in 30 runs; c_v : coefficient of variation as defined in Table 2.

time required per gene. If the chromosome length or the number of genes can be reduced, it will in turn save the overall computation time. The results in Table 2 show that PREGA is robust even when combining with other efficient crossover operators (e.g., SBOX and SJOX) that use a different method to perform the crossover. Our experimental results also showed that if the original GA or GA-based algorithms do not give a solution that is close to the optimal, PREGA will help arrive at better solution. For example, for the benchmark u2152 using the HX crossover operator, the final result is 73,339.08, which is worse than those using the other crossover operators. PREGA(HX) can, however, save most of the computation time and even improve the quality of the end result by about 2.46%, compared to the others.

4.3. Comparison with Evolutionary-Based Algorithms.

Finally, for completeness, we compare the performance of traditional GA [41], HeSEA [42], LEM [30], SA [10], UMDA [43], EHBSA [44], ACS [45], and DPSO [46] by applying PR to all of them. Tables 3 and 4 show that not only can PR vastly reduce the computation time of these algorithms, especially for very large data sets, but it can also greatly reduce the computation time of evolutionary-based algorithms that each iteration of which takes a great deal of computation time. Note that the cunning length of EHBSA

is 1/3. The inertial weight ω of DPSO is 0.5, and the random numbers for determining the influence of personal best and global best rc_1 and rc_2 are, respectively, 0.3 and 0.7. For ACS, the settings are based on those specified in [45]. That is, the population size is 25; the importance of exploitation versus exploration q_0 is 0.9; the importance of pheromone β is 2.0; ρ is 0.1; and the number of generations is 320.

The results in Table 3 show that because HeSEA takes more computation time than GA per generation, the computation time saved for HeSEA is more than for GA. For instance, the simulation results of the largest benchmark usal3509 show that using GA, the computation time is reduced by a factor of 12.77, whereas using HeSEA, the computation time is reduced by a factor of 22.07. In addition, the results of SA and PRESA highlight a different concept of removing redundant patterns. Because SA is a single-solution-based iterative algorithm, the procedures CGD and CGC have to be modified accordingly. A very simple approach is to remove patterns that are not changed for, say, 1,000 iterations in succession. Furthermore, the simulations of SA and PRESA are carried out for 30 runs, with the initial temperature 1.0 and the change probability $P(\Delta E) = \exp(-\Delta E/k_b T)$, where T is the temperature and k_b is Boltzmann's constant [10]. The results of Table 3 show that the more the number of solutions (i.e., the larger the population

size of the population-based approach is) is used in an iteration, the better the end results is and the longer the computation time is. The results of Table 3 further show that the pattern reduction method can be applied to not only the population-based but also the single-solution-based algorithms where the former finds the common subsolutions to be removed by spatial distribution while the latter finds the common subsolutions to be removed by frequency.

The results in Table 4 show that not only can the proposed algorithm reduce a great deal of the computation time of other efficient evolutionary algorithms such as UMDA and EHBSA, but it can also reduce the computation time of swarm intelligence algorithms such as ACS and DPSO while limiting the loss of the quality of the end result. In other words, the results show that PR can cut down the computation time of evolutionary algorithms, which are themselves either faster or able to provide better results than GA. For instance, even though UMDA, EHBSA, and DPSO are faster than GA by about 44.45%, 29.94%, and 56.35%, respectively, for usa13509, PR can further reduce the computation time of UMDA from 58.61% up to 81.97%, the computation time of EHBSA from 67.46% up to 79.02%, and the computation time of DPSO from 75.00% up to 91.16%. The experimental results show that the proposed algorithm can be used to speed up the performance of all the abovementioned efficient algorithms.

5. Analysis of PREGA

5.1. Diversity Analysis. Two of the most important issues in using the pattern reduction method for enhancing the performance of GA or GA-based algorithms are how to ensure the pattern reduction method can effectively reduce the computation time and how to maintain the diversity of the population, that is, the quality of the end results. In this paper, we will discuss the impact of the pattern reduction method on the performance of GA or GA-based algorithms based on three different measures: (1) the average number of genes compressed, (2) the average quality of the end results, and (3) the average size of the search space. In other words, these measures provide an indication of the search ability and the speed of convergence of PREGA.

The search space or diversity of solutions can help us understand whether or not an algorithm is capable of avoiding falling into a local minimum at early generations in the evolution process. In this paper, we use the outdegree of cities as shown in Figure 3 to indicate the search ability of an algorithm. In other words, the higher the outdegree of a city, the higher the search ability. Now, by assuming that the cities next to each other are represented as an adjacency matrix as given in Figure 3(b), the average size of the search space at generation t , denoted by \bar{S}^t , is defined as

$$\bar{S}^t = \frac{1}{2n} \sum_{i=1}^n \sum_{j=1}^n e_{i,j}^t, \quad (4)$$

where n is the number of genes (cities) left in each chromosome and $e_{i,j}^t = 1$ if there exists an edge between cities i and j ; otherwise, $e_{i,j}^t = 0$. That is, (4) represents the average of the outgoing paths of all the cities currently encoded in all the chromosomes (i.e., not removed). For instance, as Figure 3 shows, sixteen edges exist in all the chromosomes encoding six cities of TSP. The average size of the search space can be computed as $(1/(2 \times 6)) \times 16 = 1.33$. This number can help us measure the diversity of the search space of a genetic algorithm at a particular generation.

Figure 4 compares the performance of GA and PREGA for solving the TSP using the simulation result of the benchmark pr1002 as an example. Figure 4(a) indicates that PREGA can find and remove more common edges than GA, and Figure 4(b) shows that PREGA can find better solutions than GA before generation 542. Figure 4(c) shows that PREGA can maintain more diversities than the others in the early generations during its evolution process. These results convey a very important message. That is, PREGA would find higher quality result with higher diversities (search space) at the early generations during the convergence process. At the later generations of PREGA, the diversity will become small because it is converging to a stable solution or the global optimum, but our simulation results show that even in this case, PREGA can still reduce most of the redundant computations.

For instance, as Figure 4(c) shows that at about generation 39, the curves of the average diversities of GA and PREGA cross over. That is, the search diversity of PREGA becomes smaller than that of GA at about generation 39, and the gap between these two methods is widened as the number of generations increases. It seems that the search ability of PREGA becomes worse than that of GA. But the result of Figure 4(b) shows that the search ability of PREGA does not eventually decrease between generations 39 and 542. More precisely, in terms of the distance, PREGA finds the solution 277,508.5 at generation 133, even though PREGA is unable to arrive at a better solution afterwards. GA, however, requires about 542 generations to arrive at the same solution 277,508.5 as PREGA. In addition, the final result found by GA is 277,079.43 at generation 914. Then, GA has a very small probability to find a better solution because the search diversity tends to be 1 at generation 916. Now, the most important question is if most of the genes are compressed by PREGA at generation 133 (Figure 4(a)) or later, then will it prevent PREGA from finding better solutions at later generations. According to our simulation results, if either the population size or the problem size is increased, then not all the genes will be removed at the early generations, so the problem will not exist, and PREGA will still outperform GA. Figure 4(b) also indicates that the quality of the final results using GA and PREGA differs by no more than 0.15% ($((277,508.5 - 277,079.43)/277,079.43) \times 100 = 0.15$).

Figure 4(d) gives another measure [42] that can help us understand the performance of GA. The number of genes that is optimal gives us a hint in understanding how fast

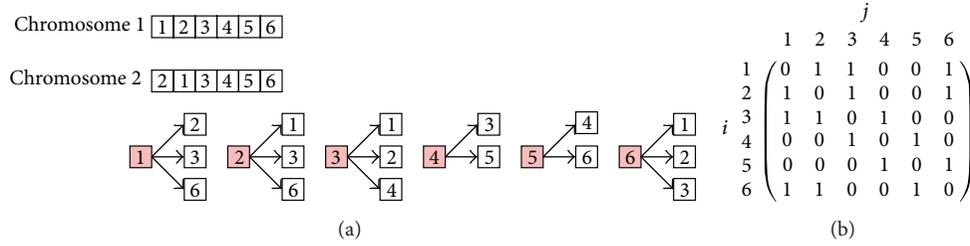
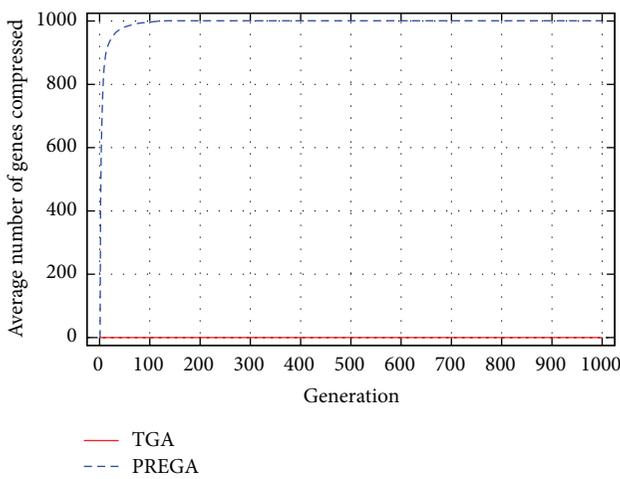
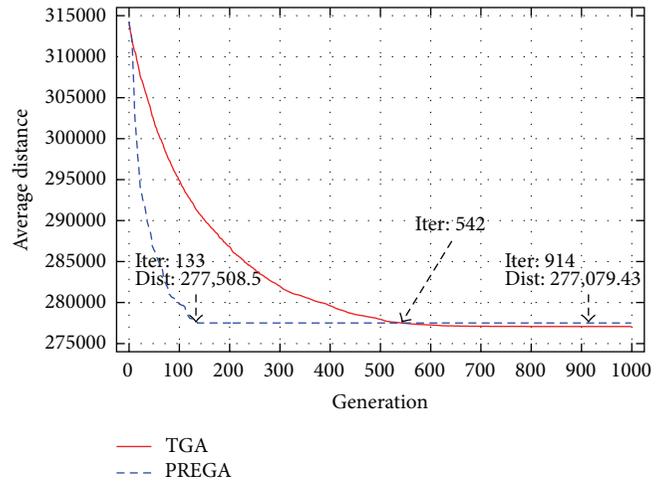


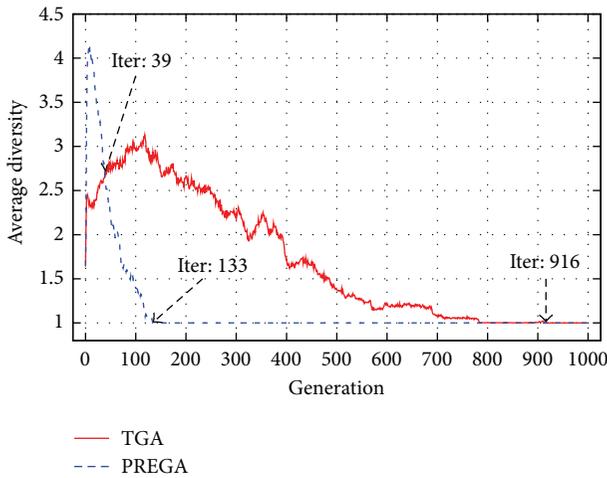
FIGURE 3: (a) Graphs showing tours encoded in two chromosomes and the outdegree of each city. For instance, the outdegree of city 1 is $d_1 = 3$. (b) Same information given in (a) represented as an adjacency matrix which makes it easier to understand how the average size of the search space is computed. The number 2 in the denominator in (4) indicates that the adjacency matrix is symmetric in this case.



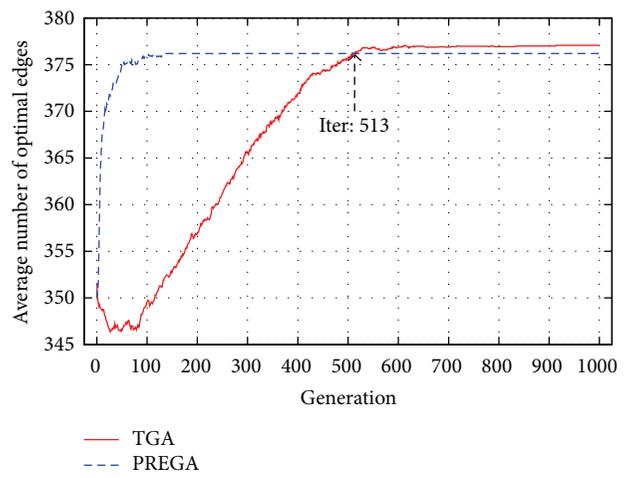
(a) Average number of genes compressed at generation t



(b) Average distance at generation t



(c) Average diversity at generation t



(d) Average number of optimal edges in the best chromosome at generation t

FIGURE 4: Example illustrating the performance of GA and PREGA for solving the benchmark pr1002.

the optimal solution can be reached by an algorithm. For instance, let us suppose that the path $\Psi = \{\Psi_1, \Psi_2, \dots, \Psi_n\}$, where Ψ_i is the optimal subsolution of an optimal solution for TSP. Now, by assuming that each chromosome is represented as a ring and letting $j = (i + 1) \bmod n$, the rate of edges that is

optimal in the best chromosome at generation t , denoted by O^t , is defined as

$$O^t = \sum_{i=1}^n o_{i,j}^t, \tag{5}$$

where n is the number of genes (cities), and $o_{i,j}^t = 1$ if there exists an edge that is the optimal subsolution between the pair of genes i and j ; otherwise, $o_{i,j}^t = 0$. Figure 4(d) shows the probability of edges that are optimal and may end up being in the final solution using GA and PREGA. As indicated in Figure 4(d), PREGA has higher probabilities to find the optimal subsolutions than GA. Also indicated in Figure 4(d) is that even though the average diversity of GA and PREGA crosses over at about generation 513, the final results of GA and PREGA are very similar. More precisely, the difference is about 0.86 optimal edges with a problem of size 1,002.

In summary, the down side of PREGA is that it may quickly converge to a suboptimal solution, but the up side is that the quality of the end result is very close to that of GA. For both GA and PREGA, the number of generations required for the diversity to converge to 1 is in general unpredictable. Using the benchmark pr1002 as an example, if the number of generations performed is 100, PREGA can not only reduce the computation time by about 94.84%, but it can also even enhance the quality of the end result by about 5.08%. However, the average diversities of GA and PREGA at generation 100 are both greater than 1, which indicates that if we let them run longer, they may be able to find a solution that is better than the current one. More precisely, as Figure 4(c) indicates, the average diversity of GA is about 1.4, and the average diversity of PREGA is about 3. Thus, to see what might happen to both GA and PREGA when the diversity approaches 1, the pr1002 benchmark is carried out again for 30 runs and 1,000 generations each run. In average, PREGA takes 0.86 s per run, and GA takes 102.54 s per run. PREGA reduces the computation time by about 99.16% $((0.86 - 102.54)/102.54) \times 100 = -99.16$) or by a factor of 119.23 compared to GA, and the quality of the end results is very close to each other. In other words, for a large problem, the number of generations required by GA to converge to even a suboptimal solution could be large and is unpredictable. On the other hand, PREGA can quickly provide a solution the quality of which is very close to that of GA even if the size of the problem is large.

5.2. Time Complexity of PREGA. The time complexity of genetic algorithm is a very important issue, and it has attracted much attention of many researches [51–53]. In [51], Ambati et al. used information exchange probability, reproduction time, and fitness computation time for estimating the time complexity of GA. According to the results of [51], Ambati et al. presented a GA-based algorithm for solving the TSP, the expected running time of which is $O(n \log n)$, where n is the number of cities. This is due to the fact that their simulations indicate that “good” solutions can be obtained by GA in $O(\log n)$ generations, even if the size of the TSP is large. In another research [53], Tseng and Yang showed that the time complexity of GA is $O(\ell mn^2)$ for data clustering problem, where ℓ is the number of generations, m the population size, and n the number of patterns.

In this paper, we assume that the time complexity of the traditional genetic algorithm is $(nm\ell)$, where n is the number of genes, m the number of chromosomes, and ℓ the number

of generations. This can be easily justified by the following analysis on the time complexity of the fitness function, selection, crossover, and mutation operators used by the traditional genetic algorithm as far as certain conditions are met. For instance, suppose that tournament selection is used as the selection operator, and its size is k (a constant that is far less than m). Let us further suppose that one point crossover with probability p_c and one point mutation with each gene having probability p_m which are mutated are used where p_c and p_m are less than 1. The selection operator takes km time at each iteration, because GA needs to randomly select k chromosomes from a set of m chromosomes to find the best one and performs this procedure m times. The one point crossover will exchange the information about $p_c mn$ time, and the mutation operator will take about $p_m mn$ time, and the fitness function takes mn time. The overall complexity of the traditional genetic algorithm is thus $O(nm\ell)$ (e.g., k , p_c , and p_m are parameters (constants) that you choose before a simulation is carried out, and all the simulation results given in Section 4 have $k = 3$ ($\ll m = 80$), $p_c = 0.5$ (< 1), and $p_m = 0.01$ ($\ll 1$)) where m and n are as defined above, ℓ is the number of generations required to converge, and the assumption that all the operators do not take more than n or m time holds. Otherwise, the time complexity could be $O(n^2 m\ell)$ or $O(nm^2 \ell)$. In the ideal case, the pattern reduction algorithm can reduce the time complexity of GA from $O(nm\ell)$ to $O(nm)$. This can be easily proved by letting Δ ($0 < \Delta < 1$) be a constant indicating the percentage of patterns retained at each iteration. In other words, $1 - \Delta$ is the percentage of genes removed at each iteration in all chromosomes. Then,

$$\begin{aligned} \sum_{i=0}^{\ell-1} \Delta^i nm &= nm \sum_{i=0}^{\ell-1} \Delta^i \\ &\leq nm \sum_{i=0}^{\infty} \Delta^i \\ &= nm \frac{1}{1 - \Delta} \\ &= O(nm). \end{aligned} \tag{6}$$

In summary, the time complexity of PREGA is bound from above by $O(nm\ell)$ and from below by $O(nm)$. In the best case, when the PREGA algorithm is started at the very first iteration and the removal bound is set to 100%, the time complexity will be $O(nm)$. In the worst case, if PREGA cannot detect any common genes to be removed, then PREGA will fall back to GA, and the time complexity will be $O(nm\ell)$. In other words, the time complexity of PREGA depends on (1) the iteration at which PREGA starts, (2) the number of patterns removed at each iteration, and (3) the removal bound, which is defined to be “up to $x\%$ of the genes detected can be removed,” though in practice, a little bit more than the removal bound of genes can be removed to simplify the implementation (more details can be found in Section 4.1). Our simulation results showed that PREGA can reduce the computation time of GA from about 80% to 95.32% when

the removal bound is set to 100% for complex data sets. The results further showed that if the number of generations of GA is set to an even larger value, we can reduce the time complexity of GA to approach that of the ideal case, that is, $O(nm)$.

6. Conclusion

This paper presents a novel technique for reducing the computation time of GA or GA-based algorithms based on the notion of pattern reduction. To evaluate the performance of the proposed algorithm, we use it to solve the traveling salesman problem, the benchmarks of which range in size from 130 to 13,509 cities. All our simulation results showed that the proposed algorithm can effectively cut down the computation time of GA and its variants, especially in cases where the data sets are large. Our simulation results further showed that the proposed algorithm can significantly reduce the computation time of the state-of-the-art heuristic algorithms we compared in the paper, such as ACO and PSO, even though these algorithms themselves are very efficient in solving the combinatorial optimization problems. In the future, our focus will be on enhancing the performance of the proposed algorithm and widening the domains of its application.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors thank the editors and the anonymous reviewers for their valuable comments and suggestions on the paper that greatly improved the quality of the paper. This work was supported in part by the National Science Council of Taiwan, ROC, under Contracts NSC102-2221-E-041-006, NSC102-2221-E-110-054, NSC99-2221-E-110-052, NSC102-2219-E-006-001, and NSC102-2221-E-390-017.

References

- [1] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Upper Saddle River, NJ, USA, 1982.
- [2] E. L. Lawer, J. K. Lenstra, A. H. R. Kan, and D. B. Shmoys, *The Traveling Salesman Problem*, John Wiley & Sons, New York, NY, USA, 1985.
- [3] A. Barvinok, S. P. Fekete, D. S. Johnson, A. Tamir, G. J. Woeginger, and R. Woodroffe, "The geometric maximum traveling salesman problem," *Journal of the ACM*, vol. 50, no. 5, pp. 641–664, 2003.
- [4] J. G. Rakke, M. Christiansen, K. Fagerholt, and G. Laporte, "The traveling salesman problem with draft limits," *Computers & Operations Research*, vol. 39, no. 9, pp. 2161–2167, 2012.
- [5] S. Yu, J. Kim, and J. Lee, "Lifetime improvement method using mobile sink for IoT service," in *Proceedings of the 10th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN '13)*, pp. 145–150, Barcelona, Spain, 2013.
- [6] A. Liefvooghe, J. Humeau, S. Mesmoudi, L. Jourdan, and E.-G. Talbi, "On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems," *Journal of Heuristics*, vol. 18, no. 2, pp. 317–352, 2012.
- [7] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.
- [8] A. Ouaarab, B. Ahiod, and X.-S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Computing and Applications*, 2013.
- [9] G. K. Jati and S. Suyanto, "Evolutionary discrete firefly algorithm for travelling salesman problem," in *Adaptive and Intelligent Systems*, vol. 6943 of *Lecture Notes in Computer Science*, pp. 393–403, Springer, Berlin, Germany, 2011.
- [10] J. W. Pepper, B. L. Golden, and E. A. Wasil, "Solving the traveling salesman problem with annealing-based heuristics: a computational study," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 32, no. 1, pp. 72–77, 2002.
- [11] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics*, Springer, New York, NY, USA, 2nd edition, 2010.
- [12] R. Martí, "Multi-start methods," in *Handbook of Metaheuristics*, F. W. Glover and G. A. Kochenberger, Eds., pp. 355–368, Kluwer Academic, Boston, Mass, USA, 1993.
- [13] M. P. Poland, C. D. Nugent, H. Wang, and L. M. Chen, "Genetic algorithm and pure random search for exosensor distribution optimisation," *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 359–372, 2012.
- [14] A. F. Sheta, P. Rausch, and A. S. Al-Afeef, "A monitoring and control framework for lost foam casting manufacturing processes using genetic programming," *International Journal of Bio-Inspired Computation*, vol. 4, no. 2, pp. 111–118, 2012.
- [15] J. Muñuzuri, P. C. Achedad, M. Rodríguez, and R. Grosso, "Use of a genetic algorithm for building efficient choice designs," *International Journal of Bio-Inspired Computation*, vol. 4, no. 1, pp. 27–32, 2012.
- [16] B. B. Pal, D. Chakraborti, P. Biswas, and A. Mukhopadhyay, "An application of genetic algorithm method for solving patrol manpower deployment problems through fuzzy goal programming in traffic management system: a case study," *International Journal of Bio-Inspired Computation*, vol. 4, no. 1, pp. 47–60, 2012.
- [17] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.
- [18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Boston, Mass, USA, 1989.
- [19] F. Glover and G. A. Kochenberger, Eds., *Handbook of Metaheuristics*, Springer, New York, NY, USA, 2003.
- [20] K. Wang and Z. Shen, "A GPU-based parallel genetic algorithm for generating daily activity plans," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1474–1480, 2012.
- [21] F. Defersha and M. Chen, "Mathematical model and parallel genetic algorithm for hybrid flexible flowshop lot streaming problem," *The International Journal of Advanced Manufacturing Technology*, vol. 62, no. 1–4, pp. 249–265, 2012.
- [22] L. Wang, A. A. Maciejewski, H. J. Siegel, V. P. Roychowdhury, and B. D. Eldridge, "A study of five parallel approaches to a genetic algorithm for the traveling salesman problem," *Intelligent Automation & Soft Computing*, vol. 11, no. 4, pp. 217–234, 2005.

- [23] E. Cantú-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles, Reseaux et Systems Repartis*, vol. 10, no. 2, pp. 141–171, 1998.
- [24] E. Cantú-Paz and D. E. Goldberg, "Efficient parallel genetic algorithms: theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 221–238, 2000.
- [25] E. Cantú-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic, Norwell, Mass, USA, 2000.
- [26] A. Sinha and D. E. Goldberg, "A survey of hybrid genetic and evolutionary algorithms," IlliGAL Report no. 2003004, University of Illinois at Urbana-Champaign, Urbana, Ill, USA, 2003.
- [27] S. W. Mahfoud and D. E. Goldberg, "Parallel recombinative simulated annealing: a genetic algorithm," *Parallel Computing*, vol. 21, no. 1, pp. 1–28, 1995.
- [28] J.-S. Lee, I.-S. Oh, and B.-R. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1424–1437, 2004.
- [29] A. Misevicius, "A fast hybrid genetic algorithm for the quadratic assignment problem," in *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference (GECCO '06)*, pp. 1257–1264, Seattle, Wash, USA, July 2006.
- [30] R. S. Michalski, "Learnable evolution model: evolutionary processes guided by machine learning," *Machine Learning*, vol. 38, no. 1-2, pp. 9–40, 2000.
- [31] C.-F. Tsai, C.-W. Tsai, and C.-P. Chen, "A novel algorithm for multimedia multicast routing in a large scale network," *Journal of Systems and Software*, vol. 72, no. 3, pp. 431–441, 2004.
- [32] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Application*, pp. 41–49, Cambridge, Mass, USA, 1987.
- [33] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik, "Rapid, accurate optimization of difficult problems using fast messy genetic algorithms," in *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 56–64, Morgan Kaufmann, Urbana, Ill, USA, 1993.
- [34] G. R. Harik and D. E. Goldberg, "Learning linkage," in *Proceedings of the 4th Workshop on Foundations of Genetic Algorithms*, pp. 247–262, San Diego, Calif, USA, 1996.
- [35] D. E. Goldberg, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Kluwer Academic, Norwell, Mass, USA, 2002.
- [36] D. Knjazew, *OmeGA: A Competent Genetic Algorithm for Solving Permutation and Scheduling Problems*, Kluwer Academic, Norwell, Mass, USA, 2002.
- [37] Y. Wang, B. Li, and T. Weise, "Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems," *Information Sciences*, vol. 180, no. 12, pp. 2405–2420, 2010.
- [38] D. E. Goldberg, K. Deb, and B. Korb, "Messy genetic algorithms: motivation, analysis, and first results," *Complex Systems*, vol. 3, no. 5, pp. 493–530, 1989.
- [39] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*, vol. 840, Springer, Berlin, Germany, 1994.
- [40] R. Yang, "Solving large travelling salesman problems with small populations," in *Proceedings of the 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, pp. 157–162, Glasgow, UK, 1997.
- [41] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, Germany, 1996.
- [42] H.-K. Tsai, J.-M. Yang, Y.-F. Tsai, and C.-Y. Kao, "An evolutionary algorithm for large traveling salesman problems," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 34, no. 4, pp. 1718–1729, 2004.
- [43] P. Larrañaga and J. A. Lozano, Eds., *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Genetic Algorithms and Evolutionary Computation, Springer, Norwell, Mass, USA, 2002.
- [44] S. Tsutsui, "Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram," in *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN '02)*, pp. 224–233, Granada, Spain, 2002.
- [45] M. Dorigo and T. Stützle, *Ant Colony Optimization*, The MIT Press, Cambridge, UK, 2004.
- [46] M. F. Tasgetiren, P. N. Suganthan, and Q.-K. Pan, "A discrete particle swarm optimization algorithm for the generalized traveling salesman problem," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 158–167, London, UK, July 2007.
- [47] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, pp. 154–159, Pittsburgh, Pa, USA, 1985.
- [48] L. Davis, "Applying adaptive algorithms to epistatic domains," in *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pp. 162–164, Los Angeles, Calif, USA, 1985.
- [49] D. Whitley, T. Starkweather, and D. Shaner, "The traveling salesman and sequence scheduling: quality solutions using genetic edge recombination," in *Handbook of Genetic Algorithms*, pp. 350–372, Van Nostrand Reinhold, New York, NY, USA, 1991.
- [50] R. Ruiz, C. Maroto, and J. Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problem," *Omega*, vol. 34, no. 5, pp. 461–476, 2006.
- [51] B. K. Ambati, J. Ambati, and M. M. Mokhtar, "Heuristic combinatorial optimization by simulated Darwinian evolution: a polynomial time algorithm for the traveling salesman problem," *Biological Cybernetics*, vol. 65, no. 1, pp. 31–35, 1991.
- [52] F. G. Lobo, D. E. Goldberg, and M. Pelikan, "Time complexity of genetic algorithms on exponentially scaled problems," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00)*, pp. 151–158, Las Vegas, Nev, USA, 2000.
- [53] L. Y. Tseng and S. B. Yang, "A genetic approach to the automatic clustering problem," *Pattern Recognition*, vol. 34, no. 2, pp. 415–424, 2001.

Research Article

Hybrid Algorithms for Fuzzy Reverse Supply Chain Network Design

Z. H. Che,¹ Tzu-An Chiang,² Y. C. Kuo,¹ and Zhihua Cui^{3,4}

¹ Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan

² Department of Business Administration, National Taipei College of Business, Taipei 10051, Taiwan

³ Complex System and Computational Intelligent Laboratory, Taiyuan University of Science and Technology, Taiyuan 030024, China

⁴ State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

Correspondence should be addressed to Z. H. Che; zhche@ntut.edu.tw

Received 11 October 2013; Accepted 23 December 2013; Published 24 April 2014

Academic Editors: J. Liu and K. Matsumoto

Copyright © 2014 Z. H. Che et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In consideration of capacity constraints, fuzzy defect ratio, and fuzzy transport loss ratio, this paper attempted to establish an optimized decision model for production planning and distribution of a multiphase, multiproduct reverse supply chain, which addresses defects returned to original manufacturers, and in addition, develops hybrid algorithms such as Particle Swarm Optimization-Genetic Algorithm (PSO-GA), Genetic Algorithm-Simulated Annealing (GA-SA), and Particle Swarm Optimization-Simulated Annealing (PSO-SA) for solving the optimized model. During a case study of a multi-phase, multi-product reverse supply chain network, this paper explained the suitability of the optimized decision model and the applicability of the algorithms. Finally, the hybrid algorithms showed excellent solving capability when compared with original GA and PSO methods.

1. Introduction

Many scholars have been devoted to the study of positive supply chains, for instance [1–5], however, the reverse supply chain has seldom been involved. Dowlatshahi [6] pointed out that reserve logistics, which is a new concept in logistics and supply chain management, offer efficient reverse logistics management that could improve the competitive power of the enterprises, especially when they face fierce competition with low-margin profits. Notwithstanding price and quality as important, influential marketing factors within a complex product path, business operations should focus on how to offer improved after-sale services to customer. Setting its target on household appliances and computers in 3C (Computers, Communications, Consumer Electronics).

Shih [7] discussed the reverse logistics system for end-of-life of products, viewing reverse logistics systems as a crucial element of future business operations; more efforts should be made in reutilization, rework, and recycling at end-of-life [8]; reverse logistics is a process of recycling and refabricating materials [9]; reverse logistics is helpful to protect environments and reduce waste of resources,

thus providing an opportunity for the reutilization of products. Ko and Evans [10] proposed a positive and a reverse supply chain network, based on a third party's logistics, and constructed a mixed-integer nonlinear planning model of a dynamic integrated distribution network, taking into account a two-echelon supply chain network and capacity constraints, underlining the significance of returning the defective products to the suppliers' partners. In addition, the defect ratios and transport loss ratios are fuzzy values in the real-world supply chain environments. Fuzzy defect ratios and fuzzy transport loss ratios, thus, are included in this reverse model for meeting actual conditions.

Thus, this paper focuses on the reverse supply chain and constructs an optimized decision model for the selection of supply chain partners as well as determination of production and distribution quantities. Multiechelon logistics issues could be regarded as a Knapsack of a multiple selection portfolio, and resource distribution as a NP-hard issue [11]; however, the reverse supply chain is more complex in this paper, with production defect ratios, transport loss ratios, and designated reprocessing are considered.

GAs have been widely used in solving real-world complex optimization problems [12–16] and Min et al. [17], Sha and Che [18], Tsai [19], Ko and Evans [10], Min et al. [20], Che and Chiang [21], and Che and Chiang [22] applied GA to solve the planning issues of a supply chain network. Moreover, PSOs also have been widely employed for solving optimization problems in different fields and some heuristic algorithms with similar concept of PSO are developed [23–30]. Yin and Wang [31], Yu and Fang [32], Chen et al. [33], Che and Cui [34], and Che [35] employed a PSO to optimize a resource distribution system with satisfactory results. Yet, no research efforts were made with respect to the planning of a reverse supply chain with GA and PSO. For this reason, these two heuristic algorithms were used to solve the reverse supply chain in this paper, along with other three hybrid algorithms: PSO-GA, PSO-SA, and GA-SA. The solving capability of these five algorithms was compared, and the optimum one was selected as a reference for decision-making.

2. Fuzzy Theory

Fuzzy theory, first proposed by Zaden in 1965 as an extension of a general set, is a numerical control methodology for imitating human thought and addressing the inaccuracy of all physical systems. According to Fuzzy Theory, the thought logic of human begins as fuzzy and is intended for judgment, even if conditions and data are uncertain, while modern computers feature bipolar logic, that is 0 or 1, different from the logic of humans. However, fuzzy logic theory can represent the degree of fuzzy concepts with values between 0 and 1, namely, “membership function.”

Karkowski [36] indicated that triangular is the most common membership function in solving possibilistic mathematical programming problems among the various types of membership function. Other related studies concerning the use of triangular fuzzy numbers for decision making problems are [37–40]. Hence, in this paper, the defect ratios and transport loss ratios are denoted by triangular fuzzy numbers. The triangular fuzzy numbers are not compulsory, if other types of fuzzy numbers are more applicable, they can be used.

The triangular fuzzy number can be represented as $f = (f_l, f_m, f_u)$, where f_l , f_m , and f_u are lower bound, mode, and upper bound values, respectively. Figure 1 shows the triangular possibility distribution of fuzzy number f .

In addition, it is essential for practical applications that a fuzzy number should be transformed to a numerical value. The transform process is called “defuzzification.” Associated ordinary number (AON) is a simple method in defuzzification and many researches have employed it directly and effectively [41–43]. In this paper, hence, the AON is used to find the crisp value for defect and transport loss ratios. The AON method can be expressed as

$$AON(f) = \frac{f_l + 2f_m + f_u}{4} \tag{1}$$

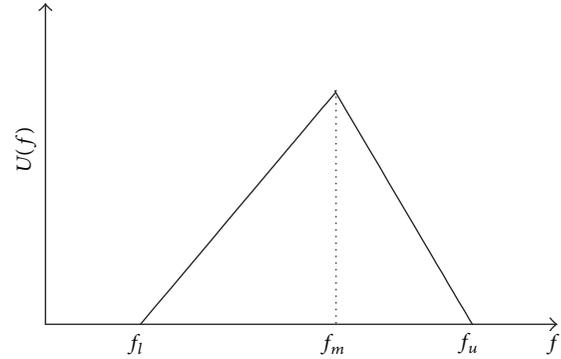


FIGURE 1: Membership functions of a triangular fuzzy number.

3. Problem Formulation and Solving

3.1. Description and Definition. This paper analyzed the reverse distribution activities of defective products in a complex supply chain network, and explored a method of feeding these products back to the manufacturing partners, with consideration of the capacity constraints of suppliers, and the demands of multi-phase, multi-product production and planning. Moreover, the partners shall be selected based on total objective functions (minimized production, transport, inspection, rework costs, and optimized production quality), in addition, the fuzzified production defect ratio and transport loss ratio are taken into account. Given the fact of numerous suppliers in a supply chain network, with different production characteristics and resource allocation capabilities covering their capacity and yield, it is crucial to select proper suppliers in a complex supply chain network.

Positive logistics are highlighted in a traditional supply chain, but unavoidable defects arising from production or transportation processes are overlooked. In practice, the defective products must be returned to the manufacturers. Relevant original data in this paper were subjected to T -treatment for data integration [44–46]. Equation (2) is a T -treatment formula (X : variable, \bar{X} : average, SD : standard deviation).

$$T = \frac{X - \bar{X}}{SD/10} + 50. \tag{2}$$

3.2. Mathematical Model for Multiphase and Multiproduct Reverse Supply Chain. The mathematical symbols of a reverse supply chain one listed in the Symbols of Mathematical Model section.

The mathematical model of a multi-phase plan of a reverse supply chain is detailed as follows:

$$\text{Minimize} \quad \left[\sum_{p=1}^P \sum_{i=1}^{I-1} \sum_{j=1}^{J_i} \sum_{k=1}^{K_{j+1}} \left(\text{PrC}_{((i,j))} \times X_{((i,j),(i+1,k))}^P \right) \right. \\ \left. + \sum_{p=1}^P \sum_{i=1}^I \sum_{j=1}^{J_i} \sum_{k=1}^{K_{j+1}} \left(\text{TrC}_{((i,j),(i+1,k))} \right) \right]$$

$$\begin{aligned} & \times \left(X_{((i,j),(i+1,k))}^P \right. \\ & \left. + RX_{((i+1,k),(i,j))}^P \right) \\ & + \sum_{p=1}^P \sum_{i=1}^I \sum_{j=1}^{J_i} \sum_{k=1}^{K_{i-1}} \left(\text{PrC}_{(i,j)} \times X_{((i-1,k),(i,j))}^P \right) \\ & + \sum_{p=1}^P \sum_{i=1}^I \sum_{j=1}^{J_i} \sum_{k=1}^{K_{i+1}} \left(\text{ChC}_{(i,j)} \times X_{((i,j),(i+1,k))}^P \right) \Big] \\ & \times \sum_{p=1}^P \sum_{i=1}^I \sum_{j=1}^{J_i} Q_{(i,j)}^P, \end{aligned} \tag{3}$$

$$\begin{aligned} \text{s.t } & \sum_{k=1}^{K_{i+1}} X_{((i,j),(i+1,k))}^P \\ = \text{INT } & \left\{ \sum_{k=1}^{K_{i+1}} X_{((i,j),(i+1,k))}^P \right. \\ & \left. \times \left(1 - {}^F\text{TR}_{((i,j),(i+1,k))} \right) + RX_{((i+1,k),(i,j))}^{P-1} \right\}, \\ & i = 1; \quad p = 1, 2, 3, \dots, P; \quad j = 1, 2, 3, \dots, J \end{aligned} \tag{4}$$

$$\begin{aligned} & \sum_{k=1}^{K_{i-1}} X_{((i-1,k),(i,j))}^P \\ = \text{INT } & \left\{ \left[\sum_{k=1}^{K_{i-1}} X_{((i-1,k),(i,j))}^P \times \sum_{k=1}^{K_{i+1}} \left(1 - {}^F\text{TR}_{((i,j),(i+1,k))} \right) \right] \right. \\ & \left. + \sum_{k=1}^{K_{i+1}} RX_{((i+1,k),(i,j))}^{P-1} - \sum_{k=1}^{K_{i-1}} RX_{((i,j),(i-1,k))}^P \right\}, \\ & i = 2, \dots, I - 1; \quad p = 1, 2, 3, \dots, P; \quad j = 1, 2, 3, \dots, J, \end{aligned} \tag{5}$$

$$\begin{aligned} & \sum_{k=1}^{K_{i-1}} X_{((i-1,k),(i,j))}^P \\ = \text{INT } & \left\{ \sum_{k=1}^{K_{i-1}} \left[X_{((i-1,k),(i,j))}^P \right. \right. \\ & \left. \times \left(1 - {}^F\text{TR}_{((i,j),(i-1,k))} \right) \right. \\ & \left. \left. - RX_{((i,j),(i-1,k))}^P \right] \right\}, \\ & i = I; \quad p = 1, 2, 3, \dots, P; \quad j = 1, 2, 3, \dots, J, \end{aligned} \tag{6}$$

$$\begin{aligned} & \sum_{k=1}^{K_{i-1}} RX_{((i,j),(i-1,k))}^P \\ = \text{INT } & \left\{ \left(\sum_{k=1}^{K_{i-1}} X_{((i-1,k),(i,j))}^P \right. \right. \\ & \left. \left. + \sum_{k=1}^{K_{i+1}} RX_{((i+1,k),(i,j))}^{P-1} \right) \times {}^F\text{DR}_{(i,j)} \right\}, \\ & i = 2, 3, \dots, I; \quad p = 2, 3, \dots, P; \quad j = 1, 2, 3, \dots, J, \end{aligned} \tag{7}$$

$$\begin{aligned} & X_{(i,j)} \\ = \text{INT } & \left\{ \sum_{k=1}^{K_{i+1}} X_{((i,j),(i+1,k))}^P \times \left(1 - {}^F\text{DR}_{(i,j)} \right) \right. \\ & \left. \times \left(1 - {}^F\text{TR}_{((i,j),(i+1,k))} \right) \right\}, \end{aligned} \tag{8}$$

$$\begin{aligned} & i = 1; \quad p = 1, 2, 3, \dots, P; \quad j = 1, 2, 3, \dots, J, \\ & X_{(i,j)} \\ = \text{INT } & \left\{ \sum_{k=1}^{K_{i-1}} X_{((i-1,k),(i,j))}^P \times \left(1 - {}^F\text{DR}_{(i,j)} \right) \right. \\ & \left. \times \left(1 - {}^F\text{TR}_{((i,j),(i+1,k))} \right) \right\}, \end{aligned} \tag{9}$$

$$\begin{aligned} & i = 2, 3, \dots, I - 1; \quad p = 1, 2, 3, \dots, P; \\ & j = 1, 2, 3, \dots, J, \end{aligned}$$

$$\begin{aligned} \text{Min CAP}_{(i,j)} & \leq \sum_{k=1}^{K_{i+1}} X_{((i,j),(i+1,k))}^P \leq \text{Max CAP}_{(i,j)}, \\ & i = 1, 2, \dots, I - 1; \quad p = 1, 2, 3, \dots, P; \quad j = 1, 2, 3, \dots, J, \end{aligned} \tag{10}$$

$$X_{((i,j),(i+1,k))}^P > 0 \quad \text{and } \in \text{integer } \forall i, j, k, p, \tag{11}$$

$$\begin{aligned} & RX_{((i,j),(i-1,k))}^P = 0 \\ & \text{for } p = 0; \quad i = 1, 2, \dots, I; \\ & j = 1, 2, 3, \dots, J; \quad k = 1, 2, 3, \dots, K. \end{aligned} \tag{12}$$

Equation (3) shows objective functions involved in minimizing production costs, transport costs, rework costs, check costs, and quality (quality level 1 is defined as optimum quality, and 10 as worst quality); (4), (5), and (6) represent quantity transported from the first-hierarchy partner to the second-hierarchy partner, and finally to the partner of last hierarchy, respectively, while the fuzzy transport loss ratio and rework quantity of returned defective products are also considered; (7) represents the rework quantity of defective products to be returned to the partner of previous hierarchy; (8) and (9) indicate that limited production quantities of partners from first to last hierarchy shall meet customer demands; (10) indicates that the limited transport quantity will not be bigger than the maximum capacity, nor less than the minimum capacity of the partner; (11) indicates that the limited transport quantity must be an integer bigger than zero; and (12) indicates that no defective product is produced in the early phase of multi-product, multi-phase production planning.

3.3. Proposed Algorithms. In this paper, GA, PSO, PSO-GA, GA-SA, and PSO-SA were applied to determine the best approximate solution, with minimum objective functions, in a reverse supply chain.

3.3.1. GA Solving Model

Step 1. Develop the structure of chromosome (Figure 2) and the production and transport quantities were real-coded. Eiben et al. [47] argued that real-coding could accelerate the algorithm convergence and improve consistency. In Figure 2, $P(1.1) \rightarrow P(2.1)$ represents the transport path from the first partner of the first hierarchy to the first partner of the second hierarchy, Gene value is an integral number representing the quantity of partner product transported on the distribution path.

Step 2. Generate a random initial population according to the final customer demands of the partners, restrict the population for meeting the demand constraint (4)~(12), and then obtain the Fitness value by substituting the chromosome into the objective formula (3).

Step 3. Use Roulette Wheel method to select the chromosomes. Calculate selection probability of every chromosome, of which the chromosomes of better fitness function have a higher probability.

Step 4. Randomly select two chromosomes from the population for crossover, and generate random crossover points, then exchange the genes of the chromosome, as shown in Figure 3; if the crossover rate is x_c , the number of chromosomes is y_c , the crossover quantity is $x_c \times y_c = n_c$; a better efficiency could be achieved if crossover rate is 0.75~0.95 [47].

Step 5. Randomly select a chromosome for mutation with its position; if the mutation rate is x_m , and the number of genes is y_m , the mutation number is $x_m \times y_m = n_m$, as shown in Figure 4; a higher mutation rate means a higher amplitude, a mutation rate < 0.1 is a typical value [48].

Step 6. When the chromosomes of best offspring are superior to those of worst population, this population will be replaced as a new one.

Step 7. Repeat Steps 2 to 6 until the stop conditions are satisfied.

3.3.2. PSO Solving Model

Step 1. Set the particle number, iteration number, maximum speed, learning factor, and inertia weight.

Step 2. Randomly generate the initial speed and position of every particle, with the range specified in the constraint (4)~(12).

Step 3. Substitute the particles of population into the objective function equation (3) to obtain the Pbest of each particle and the Gbest of the population.

Step 4. Update the speed and position of particles using Inertia Weight Method [49]; as shown in (13), when w ranges between 0.9 and 1.2, there is a higher opportunity to determine a global optimal solution [50] as follows:

$$v_i^n = wv_i^c + c_1 \times \text{rand}() \times (s_i' - s_i^c) + c_2 \times \text{rand}() \times (s_i'' - s_i^c), \quad (13)$$

$$s_i^n = s_i^c + v_i^n, \quad (14)$$

where v_i^c represents the speed when the particle position is changed, v_i^n is the new speed of particle i , s_i^c is the current position of particle, s_i' is the memory value of the individual best position of particle i , s_i'' is the memory value of the global best position, s_i^n is the new position of particle i , w is inertia weight, $\text{rand}()$ is a random variable between (0, 1), and c_1 and c_2 are learning factors.

Step 5. Substitute the updated particle position into the constraint (4)~(12); review if the updated position meets the requirements of constraints; if any particle exceeds the range, update again until all particles meet the requirements of constraint (4)~(12).

Step 6. Compare the particle's objective function value with updated particle position and its Pbest. If the objective function value is superior to the Pbest, the Pbest of this particle will be replaced by its objective function value with updated particle position.

Step 7. Compare the Pbest and Gbest. If the best value of particle is superior to Gbest, the Gbest will be replaced by the best value of the particle.

Step 8. Repeat Steps 2 to 7 until meeting the stop conditions or reaching the set iteration number.

3.3.3. PSO-GA Solving Model

Steps 1~3. Perform Steps 1~3 of PSO solving model.

Step 4. Perform Step 3 of GA solving model to select the particle positions of better fitness functions stored in the library. With this selection mechanism, there is a higher probability that the particle position of better fitness functions will become Gbest; another new Gbest could be obtained through recombination, and the particles could be evaluated and properly deleted, of which the worst individuals are replaced by optimum individuals, after sequencing of fitness functions.

Gene cell item	P(1.1) ↓	P(1.2) ↓	P(1.3) ↓	P(1.4) ↓	P(1.1) ↓	P(1.2) ↓	P(1.3) ↓	P(1.4) ↓	...	P(3.1) ↓	P(3.2) ↓	P(3.3) ↓	P(3.1) ↓	P(3.2) ↓	P(3.3) ↓
Gene value	43 ^a 24 ^b 49 ^c	46 6 0	37 9 3	205 191 14	89 3 59	14 2 0	21 56 10	188 38 55	...	76 27 21	285 128 95	0 0 90	55 128 21	241 43 130	164 33 105

a: product X c: product Z
b: product Y

FIGURE 2: Structure of chromosome.

Parent 1	Cut point							
Gene cell item	P(1.1) ↓	P(1.2) ↓	P(1.3) ↓	P(1.4) ↓	P(1.1) ↓	P(1.2) ↓	P(1.3) ↓	P(1.4) ↓
Gene value	11 ^a 25 ^b 66 ^c	150 74 81	68 79 14	23 41 39	57 76 22	41 15 52	76 37 90	20 101 38
Parent 2	P(1.1) ↓	P(1.2) ↓	P(1.3) ↓	P(1.4) ↓	P(1.1) ↓	P(1.2) ↓	P(1.3) ↓	P(1.4) ↓
Gene value	20 ^a 44 ^b 48 ^c	74 56 41	98 86 78	126 93 80	80 21 35	33 88 70	50 13 98	16 29 81
Offspring 1	One point crossover							
Gene cell item	P(1.1) ↓	P(1.2) ↓	P(1.3) ↓	P(1.4) ↓	P(1.1) ↓	P(1.2) ↓	P(1.3) ↓	P(1.4) ↓
Gene value	11 ^a 25 ^b 66 ^c	15 74 81	68 79 14	23 41 39	80 21 35	33 88 70	50 13 98	16 29 81
Offspring 2	P(1.1) ↓	P(1.2) ↓	P(1.3) ↓	P(1.4) ↓	P(1.1) ↓	P(1.2) ↓	P(1.3) ↓	P(1.4) ↓
Gene value	20 ^a 44 ^b 48 ^c	74 56 41	98 86 78	126 93 80	57 76 22	41 15 52	76 37 90	20 101 38

a: product X c: product Z
b: product Y

FIGURE 3: Single-point crossover.

Step 5. Obtain a new group of good populations using the selection mechanism, then perform Steps 1~3 of PSO solving model and obtain the new Pbest and Gbest.

Step 6. Perform Step 5 of GA solving model to randomly select mutation particles for single-point mutation and generate a new population.

Step 7. Perform Steps 4~7 of PSO solving model to obtain the new Pbest and Gbest of the new population.

Step 8. Repeat Steps 2 to 7 until stop conditions are met.

3.3.4. PSO-SA Solving Model

Steps 1~6. Perform Steps 1~6 of PSO solving model.

Step 7

Substep 7.1. Set start temperature (T_{start}), end temperature (T_{end}), cooling rate (α), and length of Markov Chain (M).

Substep 7.2. Disturb the updated particle under current temperature T , and generate a neighboring solution k .

Substep 7.3. Calculate the objective function, perform disturbance mechanism under current temperature to generate a neighboring solution k , and then substitute it into the objective function equation (5), and calculate its fitness function $f(X)$ and the difference of the fitness function $\Delta f(n) = f(X') - f(X)$. Obtain initial solution X , and through iteration of random disturbances neighboring reasonable solutions, obtain a new solution X' ; every particle runs across the length of Markov Chain M several times at start temperature T_{start} ; a fitness is obtained every time (energy function $\Delta f(n)$).

Substep 7.4. Judge if this neighboring solution is accepted through probability function, using probability function (15), as shown below, and then randomly generate random number r from 0 to 1 as follows:

$$P(n) = \begin{cases} 1, & \text{if } \Delta f(n) \leq 0, \\ \exp\left(\frac{-\Delta f(n)}{T}\right), & \text{if } \Delta f(n) > 0, \end{cases} \quad (15)$$

Old								
Gene cell item	$P(1.1)$ ↓ $P(2.1)$	$P(1.2)$ ↓ $P(2.1)$	$P(1.3)$ ↓ $P(2.1)$	$P(1.4)$ ↓ $P(2.1)$	$P(1.1)$ ↓ $P(2.2)$	$P(1.2)$ ↓ $P(2.2)$	$P(1.3)$ ↓ $P(2.2)$	$P(1.4)$ ↓ $P(2.2)$
Gene value	11/56/87	150/53/97	17/68/76	23/42/83	87/16/88	41/20/25	46/91/75	92/30/48
New								
	↓	Mutation operator			↓		↓	
Gene cell item	$P(1.1)$ ↓ $P(2.1)$	$P(1.2)$ ↓ $P(2.1)$	$P(1.3)$ ↓ $P(2.1)$	$P(1.4)$ ↓ $P(2.1)$	$P(1.1)$ ↓ $P(2.2)$	$P(1.2)$ ↓ $P(2.2)$	$P(1.3)$ ↓ $P(2.2)$	$P(1.4)$ ↓ $P(2.2)$
Gene value	20/79/64	150/53/97	17/68/76	23/42/83	57/37/75	41/20/25	28/74/39	92/30/48

FIGURE 4: Mutation.

Substep 7.5. Compare random number r with probability number $P(n)$; if $r \leq P(n)$, it will disturb the neighboring solution k ; replace the particle and its fitness function; if $r > P(n)$, it will not replace the particle. A new disturbance solution will generate if not accepted, until the termination of set search times.

Substep 7.6. Repeat Substeps 7.2–7.5 through Markov Chain (M) until M times, and then perform the cooling steps, indicating that a stable state is reached under this temperature.

Substep 7.7. Implement the cooling procedure at the set-cooling rate of α through a cooling mechanism $T = T \times \alpha$.

Substep 7.8. Judge if the cycling is finished through the set end temperature T_{end} , if $T \leq T_{end}$, perform the next step; if $T > T_{end}$, repeat Substeps 7.2 to 7.7 until $T \leq T_{end}$.

Step 8. Repeat Steps 2 to 7 until reaching the desired iteration number.

3.3.5. GA-SA Solving Model

Steps 1~5. Perform Steps 1 to 5 of GA solving model.

Step 6. Perform Substeps 7.1 to 7.8 of PSO-SA solving model.

Step 7. Generate new offspring through genetic evolution; if optimum fitness function of offspring is superior to that of the population will be replaced as a new one otherwise, maintain the chromosomes of original population for next-generation evolution.

Step 8. Repeat Steps 3 to 7 until the set stop conditions are met.

4. Illustrative Example and Results Analysis

4.1. Case Description. In a complex supply chain network, even a leading manufacturer cannot guarantee 100% yield during the production process, or prevent any defect during the transport process. However, the defect ratio and loss ratio are not fixed; thus, fuzzy defect ratio and fuzzy transport loss ratio are applied in this paper.

Based on a supply chain network of {4-4-3-3}, this paper simulated rework activity of returned defective products through a multi-product, multi-phase production plan. Assuming that the initial inventory is zero, the defective products arising from the production process of upstream manufacturers, and from the transport process, are returned to original manufacturers for rework; the suppliers of 1st–4th hierarchy have no fixed production defect ratio or transport loss ratio, meanwhile the multi-product, multi-phase production is planned into three phases, with defective products only returned during the second phase. Moreover, assuming that the production defect ratio and check costs of the first-hierarchy suppliers are not considered, only the defective products from the partners of 2nd to the 4th are returned for rework; in addition, assume that the rework process is the same as the production process, then the rework costs and production costs are the same. According to first-phase production planning, X products for final customer requirements amount to 400, 350, 450, and Y products amount to 250, 150, 200, and Z products amount to 100, 200, and 250. Figure 5 depicts the framework of a reverse supply chain and relevant parameters, including: production costs (PrC), transport costs (TrC), check costs (ChC), quality (Q), fuzzy defect ratio (FDR), fuzzy transport loss ratio (FTR), and maximum and minimum capacity (Max. CAP. and Min. CAP.). Since the production defect ratio and transport loss ratio are fuzzy sets with triangular fuzzy number, (1) is used for defuzzification. For partner $P(2.1)$, its fuzzy number of defect ratio is (0.6%, 2.2%, 3%), then its FDR is 2% according to (1). For the conciseness of this paper, the detail calculating processes of the production defect and transport loss ratios for all partners and transport paths are not presented, and their defuzzified values are shown in Figure 5.

4.2. Experimental Results and Analysis. PSO, GA, PSO-GA, GA-SA, and PSO-SA were applied for solving the distribution issues of a reverse supply chain network, the experimental design was conducted under the parameters of solving performance, of which every combination of parameters was implemented 20 times to determine an optimal combination, as listed in Table 1.

It is learnt from Table 1 that the optimal combination of PSO parameters is particle number 10, generation number 1000, maximum speed 1.25, inertia weight 2.15, and learning factor 2.05; the optimal combination of GA parameters is

TABLE 1: Combination of experimental parameters.

PSO	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Avg. fitness	24462289	24461153	24460103	24458116	24459087	24460841	24458923	24457531
	Avg. execution time (sec.)	6.731	6.896	12.468	12.391	12.842	12.546	21.391	21.016
	Avg. convergence time (sec.)	4.766	4.275	8.791	8.437	7.986	8.311	17.694	16.972
GA	Population size	5	5	5	5	10	10	10	10
	Generations	500	500	1000	1000	500	500	1000	1000
	Crossover rate	0.75	0.8	0.75	0.8	0.75	0.8	0.75	0.8
	Mutation rate	0.08	0.07	0.08	0.07	0.08	0.07	0.08	0.07
	Avg. fitness	24465387	24464085	24462752	24461924	24463297	24463159	244603191	24459450
	Avg. execution time (sec.)	19.373	18.859	67.693	68.047	47.375	47.734	225.836	217.512
	Avg. convergence time (sec.)	17.716	16.827	62.549	62.764	44.507	44.642	219.675	211.741
	PSO-GA	Population size	10	10	10	10	20	20	20
Generations		500	500	1000	1000	500	500	1000	1000
Max velocity		0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
Initial weight		1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
c_1, c_2		2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
Mutation rate		0.08	0.07	0.08	0.07	0.08	0.07	0.08	0.07
Avg. fitness		24790482	24692938	24662117	24579829	24507556	24453060	24697893	24662084
Avg. execution time (sec.)		6.758	6.579	11.864	12.714	12.898	12.685	23.257	22.934
GA-SA	Population size	5	5	5	5	10	10	10	10
	Generations	500	500	1000	1000	500	500	1000	1000
	Crossover rate	0.75	0.8	0.75	0.8	0.75	0.8	0.75	0.8
	Mutation rate	0.08	0.07	0.08	0.07	0.08	0.07	0.08	0.07
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
	Final temperature	1	1	5	5	1	1	5	5
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.25	2.15	1.25	2.15	1.25	2.15	1.25	2.15
	c_1, c_2	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
	Initial temperature	300	300	400	400	300	300	400	400
	Markov Chain Length	50	50	100	100	50	50	100	100
	Cooling rate	0.9	0.9	0.99	0.99	0.9	0.9	0.99	0.99
PSO-SA	Population size	10	10	10	10	20	20	20	20
	Generations	500	500	1000	1000	500	500	1000	1000
	Max velocity	0.95	1.25	0.95	1.25	0.95	1.25	0.95	1.25
	Initial weight	1.							

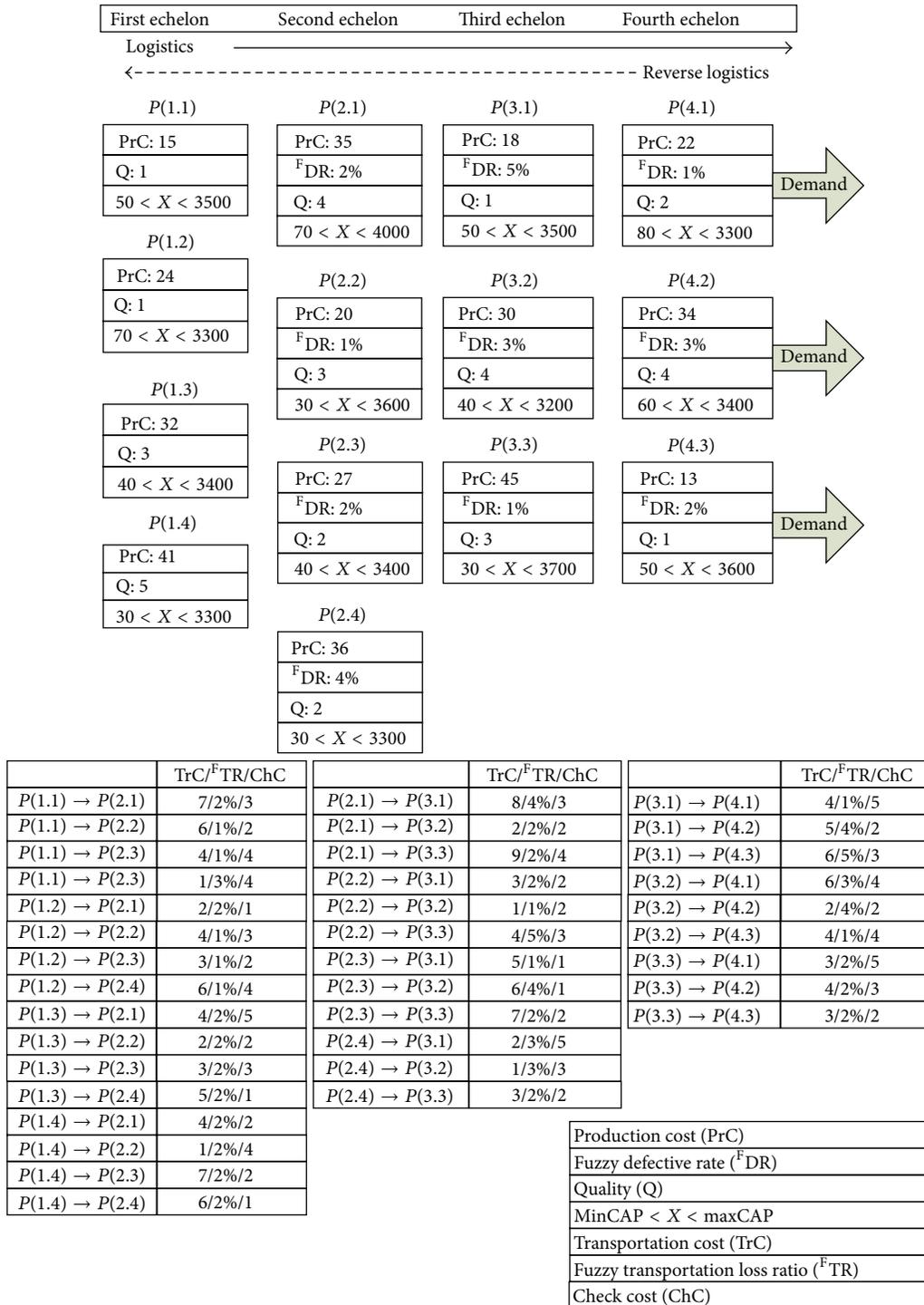


FIGURE 5: {4-4-3-3} reverse supply chain network.

population number 10, generation number 1000, crossover rate 0.8, and mutation rate 0.07; the optimal combination of PSO-GA parameters is particle number 20, generation number 500, maximum speed 1.25, inertia weight 2.15, learning factor 2.05, and mutation rate 0.07; the optimal combination of GA-SA parameters is: population number 10, generation number 500, crossover rate 0.8, mutation rate 0.07, start temperature 300, Markov Chain 50, cooling rate 0.9,

and end temperature 1; the optimal combination of PSO-SA parameters is: particle number 10, generation number 1000, maximum speed 1.25, inertia weight 2.15, learning factor 2.05, start temperature 400, Markov Chain 100, cooling rate 0.99, end temperature 5.

To compare the advantages and disadvantages of these algorithms, ANOVA was used to judge if convergence value, execution time, and convergence time differed considerably,

TABLE 2: ANOVA verification of fitness values.

$$H_0: \mu_{PSO}^{Fitness} = \mu_{GA}^{Fitness} = \mu_{PSO-GA}^{Fitness} = \mu_{GA-SA}^{Fitness} = \mu_{PSO-SA}^{Fitness}$$

$$H_1: \text{otherwise}$$

Algorithm	Numbers	Sum	Average	Variance		
PSO	30	734139577	24471319.23	1120394034		
GA	30	734406462	24480215.4	910941494		
PSO-GA	30	733498321	24449944.03	1372449396		
GA-SA	30	733929774	24464325.8	752499584		
PSO-SA	30	733666393	24455546.43	3467767352		
Source	SS	Freedom	MS	F	P value	Critical value
Intergroup	1.76E + 10	4	4389633912	2.87880644	0.02486	2.434065
Intragroup	2.21E + 11	145	1524810372			
Sum	2.39E + 11	149				

Result: reject H_0

TABLE 3: ANOVA verification of execution time.

$$H_0: \mu_{PSO}^{ET} = \mu_{GA}^{ET} = \mu_{PSO-GA}^{ET} = \mu_{GA-SA}^{ET} = \mu_{PSO-SA}^{ET}$$

$$H_1: \text{otherwise}$$

Algorithm	Numbers	Sum	Average	Variance		
PSO	30	372.022	12.40073	0.076341375		
GA	30	6524.661	217.4887	0.165604217		
PSO-GA	30	380.424	12.6808	0.076438303		
GA-SA	30	2004.847	66.82823	0.187643495		
PSO-SA	30	775.495	25.84983	0.192059868		
Source	SS	Freedom	MS	F	P value	Critical value
Intergroup	908155.7	4	227038.9	1626150.094	0.00000	2.434065
Intragroup	20.24453	145	0.139617			
Sum	908176	149				

Result: reject H_0

TABLE 4: ANOVA verification of convergence time.

$$H_0: \mu_{PSO}^{CT} = \mu_{GA}^{CT} = \mu_{PSO-GA}^{CT} = \mu_{GA-SA}^{CT} = \mu_{PSO-SA}^{CT}$$

$$H_1: \text{otherwise}$$

Algorithm	Numbers	Sum	Average	Variance		
PSO	30	246.605	8.220167	0.0913266		
GA	30	6375.408	212.5136	0.7256669		
PSO-GA	30	171.189	5.7063	0.014925		
GA-SA	30	1863.747	62.1249	0.4336888		
PSO-SA	30	326.895	10.8965	0.2385541		
Source	SS	Freedom	MS	F	P value	Critical value
Intergroup	939149	4	234787.2	780458.96	0.00000	2.434065
Intragroup	43.62068	145	0.300832			
Sum	1.23E + 09	89				

Result: reject H_0

TABLE 5: Multiple comparison on fitness value.

	Fitness μ_{PSO}	Fitness μ_{GA}	Fitness μ_{PSO-GA}	Fitness μ_{GA-SA}
Fitness μ_{GA}	(-, +)			
Fitness μ_{PSO-GA}	(+, +)	(+, +)		
Fitness μ_{GA-SA}	(+, +)	(+, +)	(-, +)	
Fitness μ_{PSO-SA}	(+, +)	(+, +)	(-, +)	(-, +)

TABLE 6: Multiple comparison on execution time.

	ET μ_{PSO}	ET μ_{GA}	ET μ_{PSO-GA}	ET μ_{GA-SA}
ET μ_{GA}	(-, -)			
ET μ_{PSO-GA}	(-, -)	(+, +)		
ET μ_{GA-SA}	(-, -)	(+, +)	(-, -)	
ET μ_{PSO-SA}	(-, -)	(+, +)	(-, -)	(+, +)

TABLE 7: Multiple comparison on convergence time.

	CT μ_{PSO}	CT μ_{GA}	CT μ_{PSO-GA}	CT μ_{GA-SA}
CT μ_{GA}	(-, -)			
CT μ_{PSO-GA}	(+, +)	(+, +)		
CT μ_{GA-SA}	(-, -)	(+, +)	(-, -)	
CT μ_{PSO-SA}	(-, -)	(+, +)	(-, -)	(+, +)

then the algorithms were compared using Scheffé’s multiple comparison method [51]; one-way ANOVA was used to check the difference of characteristics and variables [52]; Scheffé’s multiple comparison method was then used to check the differences of the various groups, and whether the differences reached a significant level. Tables 2, 3, and 4 list the comparative check data of the five algorithms, which were sourced from 30 occasions of independent calculations for the optimal combination of parameters.

In Tables 2–4, the results of ANOVA verification are shown that all H_0 are rejected (P value < $\alpha = 0.05$). Then, the fitness value, execution time, and convergence time of the algorithms were compared using Scheffé’s multiple comparison method, with Scheffé’s equation as follows:

$$\begin{aligned} \bar{x}_i - \bar{x}_j - \sqrt{(k-1) F_{\alpha(k-1)(n-k)}} \sqrt{\text{MSE} \left(\frac{1}{n_i} + \frac{1}{n_j} \right)}, \\ \bar{x}_i - \bar{x}_j + \sqrt{(k-1) F_{\alpha(k-1)(n-k)}} \sqrt{\text{MSE} \left(\frac{1}{n_i} + \frac{1}{n_j} \right)}, \end{aligned} \tag{16}$$

where \bar{x}_i, \bar{x}_j are the mean values of algorithms compared, k is group freedom, $F_{\alpha(k-1)(n-k)}$ is critical value, MSE is intergroup MS, n_i, n_j is number of samples.

Table 5 shows Fitness $\mu_{PSO-GA} = \text{Fitness } \mu_{GA-SA} = \text{Fitness } \mu_{PSO-SA} < \text{Fitness } \mu_{PSO} = \text{Fitness } \mu_{GA}$; that is, PSO-GA, GA-SA, and PSO-SA are all better than PSO and GA, and there are no clear differences in the fitness values of the three algorithms. The comparative result of system execution time is shown in Table 6, and

$\text{ET } \mu_{PSO} < \text{ET } \mu_{PSO-GA} < \text{ET } \mu_{PSO-SA} < \text{ET } \mu_{GA-SA} < \text{ET } \mu_{GA}$ is the PSO is superior to other solving models. The convergence times of the algorithms are shown in Table 7, and $\text{CT } \mu_{PSO-GA} < \text{CT } \mu_{PSO} < \text{CT } \mu_{PSO-SA} < \text{CT } \mu_{GA-SA} < \text{CT } \mu_{GA}$ PSO-GA < PSO < PSO-SA < GA-SA < GA; that is, PSO-GA, has faster convergence speed than PSO, PSO-SA, GA-SA, and GA. The results show that PSO-GA performs better in fitness value, execution time, and convergence time. Then, the PSO-GA is employed to make the distribution plan of three periods reverse supply chain and the results are shown in Tables 8, 9, and 10.

5. Conclusions

This paper focused on analyzing the issues of returning defective products to original manufacturers in a reverse supply chain system. Fuzzy defect and fuzzy transport loss ratios were considered, and an optimized mathematical model was developed. This model combined cost and quality with T-technology and described how to plan a reverse supply chain on the precondition of meeting customer demands and realizing the capacity of partners. To solve the problems efficiently, three hybrid algorithms were applied to this reverse model, including PSO-GA, PSO-SA, and GA-SA; then, the performances of these algorithms were compared. The experimental results show that if the fitness value, execution time, and convergence time are considered, PSO-GA has the minimal value, which means that PSO-GA has the qualities and capabilities for dealing with the production planning and distribution of a multi-phase, multi-product reverse supply chain.

TABLE 8: Distribution plan by PSO-GA (first period).

From	To	Echelon 2			Echelon 3			Echelon 4			
		P(2.1)	P(2.2)	P(2.3)	P(2.4)	P(3.1)	P(3.2)	P(3.3)	P(4.1)	P(4.2)	P(4.3)
Echelon 1	P(1.1)	10 ^a , 6 ^b , 49 ^c	73, 3, 6	42, 110, 59	24, 253, 5						
	P(1.2)	142, 47, 4	174, 26, 33	34, 18, 2	102, 10, 234						
	P(1.3)	157, 24, 57	256, 38, 1	7, 29, 56	22, 11, 55						
	P(1.4)	74, 39, 6	76, 17, 21	31, 8, 13	74, 14, 1						
Echelon 2	P(2.1)					177, 3, 83	116, 8, 0	82, 103, 25			
	P(2.2)					458, 2, 37	109, 14, 17	8, 67, 6			
	P(2.3)					53, 100, 60	41, 3, 3	17, 58, 65			
	P(2.4)					111, 236, 9	10, 29, 244	92, 11, 30			
Echelon 3	P(3.1)								170, 0, 0	100, 0, 174	91, 0, 255
	P(3.2)								95, 0, 0	47, 0, 0	123, 0, 0
	P(3.3)								139, 253, 101	213, 155, 32	245, 204, 0

^aProduct X; ^bproduct Y; ^cproduct Z.

TABLE 9: Distribution plan by PSO-GA (second period).

From	Echelon 1				Echelon 2				Echelon 3			Echelon 4				
	To	$P(1.1)$	$P(1.2)$	$P(1.3)$	$P(1.4)$	$P(2.1)$	$P(2.2)$	$P(2.3)$	$P(2.4)$	$P(3.1)$	$P(3.2)$	$P(3.3)$	$P(4.1)$	$P(4.2)$	$P(4.3)$	
Echelon 1	$P(1.1)$		100, 76, 12	69, 48, 80	61, 349, 47	18, 55, 0										
	$P(1.2)$		34, 19, 22	0, 27, 0	214, 50, 47	91, 10, 14										
	$P(1.3)$		95, 334, 95	131, 13, 51	8, 17, 40	28, 5, 126										
	$P(1.4)$		111, 11, 0	12, 15, 100	9, 91, 0	125, 2, 15										
Echelon 2	$P(2.1)$	0, 0, 0	3, 0, 0	3, 0, 0	1, 2, 2					130, 320, 67	189, 27, 59	10, 85, 0				
	$P(2.2)$	2, 0, 0	4, 0, 0	6, 0, 0	2, 2, 1					155, 37, 213	52, 38, 17	0, 29, 0				
	$P(2.3)$	1, 0, 0	1, 0, 0	0, 0, 0	1, 2, 2					82, 344, 12	123, 62, 108	81, 94, 10				
	$P(2.4)$	0, 0, 0	1, 0, 0	0, 0, 0	1, 8, 3					55, 0, 51	139, 7, 0	55, 63, 90				
Echelon 3	$P(3.1)$												41, 303, 0	134, 361, 0	117, 0, 0	
	$P(3.2)$												130, 0, 0	121, 0, 0	60, 396, 0	
	$P(3.3)$												226, 0, 151	143, 0, 304	64, 6, 150	
Echelon 4	$P(4.1)$															
	$P(4.2)$															
	$P(4.3)$															
Demand													400, 300, 150	400, 350, 300	250, 400, 150	

Symbols of Mathematical Model

n :	Number of products, $n = 1, 2, 3, \dots, N$
N :	Total number
i :	Hierarchy of supply chain, $i = 1, 2, 3, \dots, I$
I :	Total number of hierarchy of supply chain
p :	Production phase, $t = 1, 2, 3, \dots, T$
P :	Total production phase
J_i :	Serial number of partner of i th hierarchy
K_i :	Serial number of partner of i th hierarchy designated to original manufacturer
$ChC_{(i,j)}$:	Check cost of j th partner of i th hierarchy
$PrC_{(i,j)}$:	Production cost of j th partner of i th hierarchy
$TrC_{((i,j),(i+1,k))}$:	Transport cost from j th partner of i th hierarchy to original (designated) partner k of $i + 1$ hierarchy
$FTR_{((i,j),(i+1,k))}$:	Fuzzy transport loss ratio from j th partner of i th hierarchy to original (designated) partner of $i + 1$ hierarchy
$FDR_{(i,j)}$:	Fuzzy defect ratio of j th partner of i th hierarchy
$X_{(i,j)}^P$:	Production quantity of j th partner of i th hierarchy at phase p
$X_{((i,j),(i+1,k))}^P$:	Transport quantity from j th partner of i th hierarchy at phase p to original (designated) partner of $i + 1$ hierarchy
$RX_{((i+1,k),(i,j))}^{P-1}$:	Quantity of defective products transported from j th partner of i th hierarchy at phase $p - 1$ back to original (designated) partner of $i + 1$ hierarchy
$Max\ CAP_{(i,j)}$:	Maximum capacity of j th partner of i th hierarchy
$Min\ CAP_{(i,j)}$:	Minimum capacity of j th partner of i th hierarchy
$Q_{(i,j)}^P$:	Quality level of j th partner of i th hierarchy at phase p
$ChC_{(i,j)}$:	Check cost of j th partner of i th hierarchy
$PrC_{(i,j)}$:	Production cost of j th partner of i th hierarchy
$TrC_{((i,j),(i+1,k))}$:	Transport cost from j th partner of i th hierarchy to original (designated) partner k of $i + 1$ hierarchy
$SD_{(i,j)}^Q$:	Standard deviation of quality of j th partner of i th hierarchy
$SD_{(i,j)}^{ChC}$:	Standard deviation of check cost of j th partner of i th hierarchy
$SD_{(i,j)}^{PrC}$:	Standard deviation of production cost of j th partner of i th hierarchy
$SD_{((i,j),(i+1,k))}^{TrC}$:	Standard deviation of transport cost from j th partner of i th hierarchy to original (designated) partner k of $i + 1$ hierarchy
$INT\{\}$:	Integer.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank Mr. K. Hsiao for supporting writing programs and the National Science Council of Taiwan for their partial financial support (Grants nos. NSC 102-2410-H-027-009 and NSC 101-2410-H-027-006). The authors would also like to acknowledge the editors and anonymous reviewers for their helpful comments and suggestions, which greatly improved the presentation of this paper.

References

- [1] G. Desbarats, "The innovation supply chain," *Supply Chain Management*, vol. 4, no. 1, pp. 7–10, 1999.
- [2] S. M. Disney, M. M. Naim, and A. Potter, "Assessing the impact of e-business on supply chain dynamics," *International Journal of Production Economics*, vol. 89, no. 2, pp. 109–118, 2004.
- [3] T. M. Simatupang and R. Sridharan, "The collaboration index: a measure for supply chain collaboration," *International Journal of Physical Distribution and Logistics Management*, vol. 35, no. 1, pp. 44–62, 2005.
- [4] D. Y. Sha and Z. H. Che, "Virtual integration with a multi-criteria partner selection model for the multi-echelon manufacturing system," *International Journal of Advanced Manufacturing Technology*, vol. 25, no. 7-8, pp. 793–802, 2005.
- [5] M. Tuominen and M. Anttila, "Strategising for innovation and inter-firm collaboration: capability analysis in assessing competitive superiority," *International Journal of Technology Management*, vol. 33, no. 2-3, pp. 214–233, 2006.
- [6] S. Dowlatshahi, "Developing a theory of reverse logistics," *Interfaces*, vol. 30, no. 3, pp. 143–155, 2000.
- [7] L.-H. Shih, "Reverse logistics system planning for recycling electrical appliances and computers in Taiwan," *Resources, Conservation and Recycling*, vol. 32, no. 1, pp. 55–72, 2001.
- [8] A. Desai and A. Mital, "Evaluation of disassemblability to enable design for disassembly in mass production," *International Journal of Industrial Ergonomics*, vol. 32, no. 4, pp. 265–281, 2003.
- [9] I. Dobos, "Optimal production-inventory strategies for a HMMS-type reverse logistics system," *International Journal of Production Economics*, vol. 81-82, pp. 351–360, 2003.
- [10] H. J. Ko and G. W. Evans, "A genetic algorithm-based heuristic for the dynamic integrated forward/reverse logistics network for 3PLs," *Computers and Operations Research*, vol. 34, no. 2, pp. 346–366, 2007.
- [11] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York, NY, USA, 1997.
- [12] B. Stojanovic, M. Milivojevic, M. Ivanovic, N. Milivojevic, and D. Divac, "Adaptive system for dam behavior modeling based on linear regression and genetic algorithms," *Advances in Engineering Software*, vol. 65, pp. 182–190, 2013.
- [13] R. Belevičius, D. Jatulis, and D. Šešok, "Optimization of tall guyed masts using genetic algorithms," *Engineering Structures*, vol. 56, pp. 239–245, 2013.
- [14] D. S. Hong and H. S. Cho, "A genetic-algorithm-based approach to the generation of robotic assembly sequences," *Control Engineering Practice*, vol. 7, no. 2, pp. 151–159, 1999.
- [15] L. Özdamar, "A genetic algorithm approach to a general category project scheduling problem," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 29, no. 1, pp. 44–59, 1999.
- [16] M. F. Sebaaly and H. Fujimoto, "Genetic planner for assembly automation," in *Proceedings of the IEEE International Conference*

- on *Evolutionary Computation (ICEC'96)*, pp. 401–406, May 1996.
- [17] H. Min, C. S. Ko, and H. J. Ko, “The spatial and temporal consolidation of returned products in a closed-loop supply chain network,” *Computers and Industrial Engineering*, vol. 51, no. 2, pp. 309–320, 2006.
- [18] D. Y. Sha and Z. H. Che, “Supply chain network design: partner selection and production/distribution planning using a systematic model,” *Journal of the Operational Research Society*, vol. 57, no. 1, pp. 52–62, 2006.
- [19] C.-F. Tsai, “An intelligent adaptive system for the optimal variable selections of R&D and quality supply chains,” *Expert Systems with Applications*, vol. 31, no. 4, pp. 808–825, 2006.
- [20] H. Min, H. J. Ko, and C. S. Ko, “A genetic algorithm approach to developing the multi-echelon reverse logistics network for product returns,” *Omega*, vol. 34, no. 1, pp. 56–69, 2006.
- [21] Z. H. Che and C. J. Chiang, “A modified Pareto genetic algorithm for multi-objective build-to-order supply chain planning with product assembly,” *Advances in Engineering Software*, vol. 41, no. 7-8, pp. 1011–1022, 2010.
- [22] Z. H. Che and T. A. Chiang, “Designing a collaborative supply chain plan using the analytic hierarchy process and genetic algorithm with cycle time estimation,” *International Journal of Production Research*, vol. 50, no. 16, pp. 4426–4443, 2012.
- [23] Z. H. Che, “PSO-based back-propagation artificial neural network for product and mold cost estimation of plastic injection molding,” *Computers and Industrial Engineering*, vol. 58, no. 4, pp. 625–637, 2010.
- [24] E. García-Gonzalo and J. L. Fernández-Martínez, “A brief historical review of particle swarm optimization (PSO),” *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 3–16, 2012.
- [25] L. Ali and S. L. Sabat, “Particle swarm optimization based universal solver for global optimization,” *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 95–105, 2012.
- [26] R. K. Sahu, S. Panda, U. K. Rout, and P. Raul, “Application of gravitational search algorithm for load frequency control of multi area power system,” *Journal of Bioinformatics and Intelligent Control*, vol. 2, no. 3, pp. 200–210, 2013.
- [27] A. S. Reddy and K. Vaisakh, “Environmental constrained economic dispatch by modified shuffled frog leaping algorithm,” *Journal of Bioinformatics and Intelligent Control*, vol. 2, no. 3, pp. 216–222, 2013.
- [28] L. P. Xie, J. C. Zeng, and R. A. Formato, “Selection strategies for gravitational constant G in artificial physics optimisation based on analysis of convergence properties,” *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 380–391, 2012.
- [29] X. J. Cai, S. J. Fan, and Y. Tan, “Light responsive curve selection for photosynthesis operator of APOA,” *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 373–379, 2012.
- [30] K. Agarwal, M. Goyal, and P. R. Srivastava, “Code coverage using intelligent water drop (IWD),” *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 392–402, 2012.
- [31] P.-Y. Yin and J.-Y. Wang, “A particle swarm optimization approach to the nonlinear resource allocation problem,” *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 232–242, 2006.
- [32] J. X. Yu and X. Y. Fang, “Dynamic reactive power optimization based on particle swarm optimization algorithm,” *East China Electric Power*, vol. 34, no. 11, pp. 21–25, 2006.
- [33] Y. H. Chen, W. J. Wang, and C. H. Chiu, “Particle swarm optimization algorithm and its application to power system,” *Henan Science*, vol. 25, no. 1, pp. 29–32, 2007.
- [34] Z. H. Che and Z. Cui, “Unbalanced supply chain design using the analytic network process and a hybrid heuristic-based algorithm with balance modulating mechanism,” *International Journal of Bio-Inspired Computation*, vol. 3, no. 1, pp. 56–66, 2011.
- [35] Z. H. Che, “A particle swarm optimization algorithm for solving unbalanced supply chain planning problems,” *Applied Soft Computing Journal*, vol. 12, no. 4, pp. 1279–1287, 2012.
- [36] I. Karkowski, “Architectural synthesis with possibilistic programming,” in *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, vol. 1, pp. 14–22, 1995.
- [37] R. E. Giachetti and R. E. Young, “A parametric representation of fuzzy numbers and their arithmetic operators,” *Fuzzy Sets and Systems*, vol. 91, no. 2, pp. 185–202, 1997.
- [38] J. H. Moon and C. S. Kang, “Application of fuzzy decision making method to the evaluation of spent fuel storage options,” *Progress in Nuclear Energy*, vol. 39, no. 3-4, pp. 345–351, 2001.
- [39] J. K. Cochran and H.-N. Chen, “Fuzzy multi-criteria selection of object-oriented simulation software for production system analysis,” *Computers and Operations Research*, vol. 32, no. 1, pp. 153–168, 2005.
- [40] H. K. Chiou and G. H. Tzeng, “Fuzzy hierarchical evaluation with grey relation model of green engineering for industry,” *International of Fuzzy System*, vol. 3, pp. 466–475, 2001.
- [41] Y. Tsujimura, S. Chang, and M. Gen, “Effective method for large scale project planning with multiple fuzzy activity times,” in *Proceedings of the 2nd IEEE International Conference on Fuzzy Systems*, pp. 1009–1015, April 1993.
- [42] J. G. Greenhut, G. Norman, and C. Temponi, “Toward a fuzzy theory of oligopolistic competition,” in *Proceedings of the 3rd International Symposium on Uncertainty Modeling and Analysis and Annual Conference of the North American Fuzzy Information Processing Society (ISUMA-NAFIPS'95)*, pp. 286–291, September 1995.
- [43] S. Petrovic and C. Fayad, “A fuzzy shifting bottleneck hybridised with genetic algorithm for real-world job shop scheduling,” in *Proceedings of the Mini-EURO Conference, Managing Uncertainty in Decision Support Models*, C. H. Antunes and L. C. Dias, Eds., pp. 1–6, Coimbra, Portugal, 2004.
- [44] G. C. Huang and P. k. Wu, “The Development of a standardized mathematics. Achievement test for seventh graders and its related study: analyzed with IRT,” *Journal of Educational Research and Development*, vol. 2, pp. 109–142, 2006.
- [45] H. S. Wang and Z. H. Che, “An integrated model for supplier selection decisions in configuration changes,” *Expert Systems with Applications*, vol. 32, no. 4, pp. 1132–1140, 2007.
- [46] M. Y. Liou and C. M. Lu, “Application of the transtheoretical model to smoking behavior among freshmen,” *Journal of Health Education*, vol. 24, pp. 47–70, 2005.
- [47] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, “Parameter control in evolutionary algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.
- [48] W. D. Kelton, R. P. Sadowski, and D. A. Sadowski, *Simulation with ARENA*, McGraw-Hill, New York, NY, USA, 1998.
- [49] Y. Shi and R. C. Eberhart, “Comparison between genetic algorithms and particle swarm optimization,” in *Proceedings of the 7th International Conference on Evolutionary Programming VII*, vol. 1447, pp. 611–616, 1998.

- [50] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation (ICEC'98)*, pp. 69–73, May 1998.
- [51] H. Schee, "A method for judging all contrasts in the analysis of variance," *Biometrika*, vol. 40, no. 1-2, pp. 87–104, 1953.
- [52] C. C. Shen and C. Y. Hsieh, "A study on the relationship among attraction, tourist satisfaction and loyalty of religious tourism a case of Fo Guang Shan in Kaohsiung," *Tourism Management Research*, vol. 3, no. 1, pp. 79–95, 2003.

Research Article

Evolutionary Computation with Spatial Receding Horizon Control to Minimize Network Coding Resources

Xiao-Bing Hu^{1,2} and Mark S. Leeson²

¹ State Key Laboratory of Earth Surface Processes and Resource Ecology, Beijing Normal University, Beijing 100875, China

² School of Engineering, University of Warwick, Coventry CV4 7AL, UK

Correspondence should be addressed to Xiao-Bing Hu; dr_xiaobinghu@hotmail.co.uk

Received 7 November 2013; Accepted 30 December 2013; Published 14 April 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 X.-B. Hu and M. S. Leeson. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The minimization of network coding resources, such as coding nodes and links, is a challenging task, not only because it is a NP-hard problem, but also because the problem scale is huge; for example, networks in real world may have thousands or even millions of nodes and links. Genetic algorithms (GAs) have a good potential of resolving NP-hard problems like the network coding problem (NCP), but as a population-based algorithm, serious scalability and applicability problems are often confronted when GAs are applied to large- or huge-scale systems. Inspired by the temporal receding horizon control in control engineering, this paper proposes a novel spatial receding horizon control (SRHC) strategy as a network partitioning technology, and then designs an efficient GA to tackle the NCP. Traditional network partitioning methods can be viewed as a special case of the proposed SRHC, that is, one-step-wide SRHC, whilst the method in this paper is a generalized N -step-wide SRHC, which can make a better use of global information of network topologies. Besides the SRHC strategy, some useful designs are also reported in this paper. The advantages of the proposed SRHC and GA for the NCP are illustrated by extensive experiments, and they have a good potential of being extended to other large-scale complex problems.

1. Introduction

Network coding may significantly improve network performance in terms of network throughput [1, 2]. This advantage of network coding is demonstrated in Figure 1(a), where node 1 is the source, nodes 6 and 7 are sinks, and the capacity of every link is just 1 (in this paper, all links are of unit-capacity). If the nodes in the network only forward and replicate the data they receive, then it is easy to see that one sink can only receive 1 unit of data at one time, although the other sink may achieve a rate of 2. However, with network coding allowed, node 4 in Figure 1(a) may combine data from its two incoming links through the “+” operation, and then at both sinks, a rate of 2 can be achieved by using the “−” operation to decode data. Therefore, network coding increases the total rate of information flow through the same network from 3 to 4, which is obviously a significant improvement. Although network coding is usually allowed at all nodes in most relevant literature, an interesting observation is that a

given target rate can often be achieved by conducting network coding at only a relatively small proportion of the nodes [2]. For instance, in the network given by Figure 1(b), network coding at both node 4 and node 5 will make no difference in terms of the achieved rate at the sinks. In other words, network coding is not necessary in the network of Figure 1(b). Therefore, a question is raised: at which nodes does network coding need to be conducted, or how to make most of network capacity at a minimal cost in terms of network coding resources? To answer this question, a minimal set of nodes needs to be found for coding, which has been proved to be an NP-hard problem [3].

In this paper, the above problem of minimizing network coding resources is referred to as the network coding problem (NCP). To address this problem, researchers have already attempted many different methods such as minimal approaches [4, 5], linear programming methods [6], and genetic algorithms (GAs) [2, 7–10]. These methods were all reported to be effective to minimize network coding

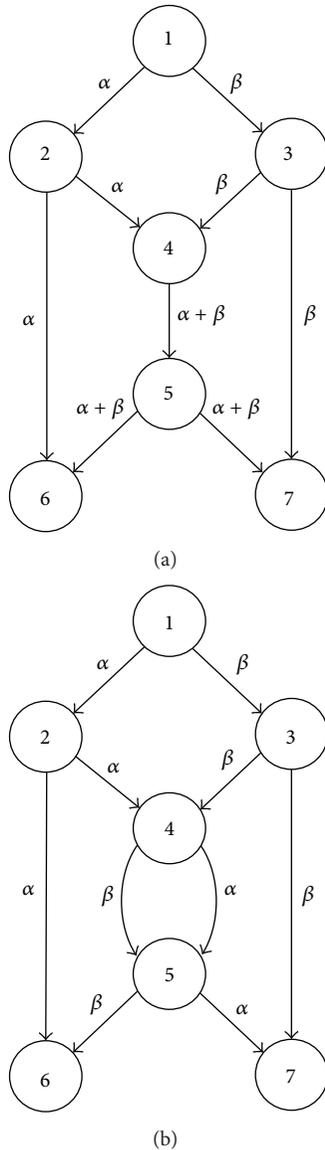


FIGURE 1: Basic idea of network coding.

resources. In particular, like in the applications to many other NP-hard problems, GAs as large-scale parallel stochastic searching and optimization algorithms have demonstrated good potential in resolving the NCP. However, the poor scalability of these reported methods largely hampers their applications in the large-scale NCP. For instance, the approaches in both [4, 5] determined the minimal set of nodes for coding by removing links in a greedy fashion. The optimal formulations of the linear programming method in [6] involve a number of variables and constraints that grows exponentially with number of sinks. As a family member of population-based algorithms, GAs are generally very expensive in terms of memory demand and computational time in the case of large-scale problems [11, 12]. To address the scalability problem, decentralized and distributed versions of algorithms often need to be developed, such as the GAs reported in [7, 9].

Before such decentralized and distributed algorithms can be applied, a problem partitioning method has to be employed in order to divide a large-scale network into some subgraphs of manageable size. This paper attempts to shed a bit of more light on how to design an effective scalable GA for the NCP.

In a conventional problem partitioning method (e.g., see [13, 14]), a large-scale problem is divided into some separate subproblems. Then, each subproblem is resolved in a rather isolated manner. After all subproblems have been resolved independently, their subsolutions are integrated together to form a complete solution to the original large-scale problem. However, even though optimal subsolutions to the subproblems can be found, the integrated complete solution to the original large-scale problem is often not optimal or even good. In other words, optimal subsolutions to the subproblems are often not optimal at all from a global point of view. A main cause of losing the global optimality is the independent/isolated way of resolving each subproblem. In this paper, inspired by the temporal receding horizon control (TRHC) strategy in the area of control engineering [15, 16], we propose a novel spatial receding horizon control (SRHC) strategy to partition a large-scale problem. In the SRHC problem partitioning method, a large-scale problem is divided into many subproblems, which compose a problem space; a spatial horizon is then defined which covers some subproblems each time and will recede in the problem space. The spatial horizon is composed of several spatial steps. Each time the spatial horizon recedes by a spatial step. All subproblems covered by a spatial horizon will be optimized as a whole, and only the subsolutions to the subproblems within the first step of the spatial horizon will be saved and fixed, whilst others will be discarded and then recalculated in the next spatial horizon. With the SRHC strategy, a subproblem will be optimized not in an independent/isolated manner, but by making use of its neighboring information in the problem space. Simply speaking, the conventional problem partitioning strategy can be viewed as a one-step-wide SRHC, whilst the new method proposed here is a generalized N -step-wide SRHC. Obviously, by optimizing a subproblem together with its neighboring sub-problems, it is likely to improve the quality of the associated subsolution in terms of global optimality. The solution quality may be further improved by integrating a GA into the SRHC method by setting up a solution pool for the subproblems in those decided spatial steps.

Besides the novel SRHC strategy, this paper also integrates some useful designs reported in [10]. There is a common assumption in many studies on the NCP: the target rate is always achievable if coding is allowed at all nodes. To avoid this unrealistic assumption, a more general objective function will be used, which aims not only to minimize network coding resources, but also to maximize the actually achieved rate at sinks. Regarding chromosome structure, rather than the widely used binary matrix, an integer-based permutation representation is adopted, which records relative signals on links and is therefore free of feasibility problems. The permutation representation also enables the derivation of exact information flow on links, which makes it possible to integrate many useful NCP-specific heuristic rules into the

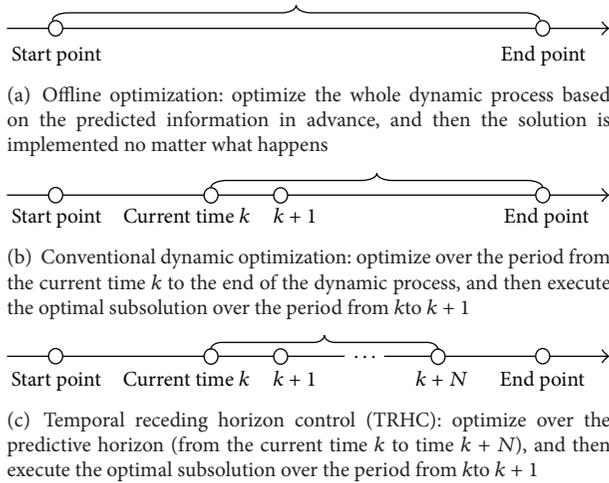


FIGURE 2: Illustration of temporal receding horizon control (TRHC).

algorithm, in order to significantly improve the overall quality of chromosomes. The remainder of this paper will give the details of the proposed SRHC based GA for the NCP.

2. Basic Idea of SRHC

2.1. Temporal Receding Horizon Control (TRHC) for Dynamic Problems. First of all, a brief review on the conventional receding horizon control (RHC) strategy in control engineering will be very useful. To distinguish from the method proposed in this paper, the conventional RHC in dynamic control problems is hereafter referred to as temporal receding horizon control (TRHC). TRHC, also known as model predictive control, has proved to be a highly effective online optimization strategy in the area of control engineering, and it exhibits many advantages against other control strategies [15, 16]. It is easy for TRHC to handle complex dynamic systems with various constraints. It also naturally exhibits promising robust performance against uncertainties since the online updated information can be sufficiently used to improve the decision. Simply speaking, TRHC is an N -step-ahead online optimization strategy to deal with dynamic problems. In this framework, decision is made by looking ahead for N steps in terms of a given cost/criterion, and the decision is only implemented by one step. Then the implementation result is checked, and a new decision is made by taking updated information into account and looking ahead for another N steps.

Figure 2 illustrates the basic idea of TRHC by comparing it with some other optimization strategies in an intuitive way. Apparently the offline optimization strategy, as shown in Figure 2(a), is not suitable for dynamic environments. The conventional dynamic optimization, as shown in Figure 2(b), is often criticized for its poor real-time properties and poor performance under disturbances and/or uncertainties in dynamic environments. As illustrated in Figure 2(c), thanks to the idea of temporal receding horizon, the TRHC strategy provides a possible solution to the problems confronted by

the conventional dynamic optimization strategy. A properly chosen temporal receding horizon can effectively filter out most unreliable information and reduce the scale of problem. The latter is especially important for complex systems and time-consuming algorithms to satisfy the time limit on the online optimization process. TRHC has now been widely accepted in the area of control engineering [15, 16]. Attention has also been paid to applications of TRHC to areas like management and operations research [17–19]. Particularly, the TRHC strategy has recently been reported to be successfully integrated into population-based algorithms to tackle various dynamic NP-hard optimization problems [20–22].

2.2. Spatial Receding Horizon Control (SRHC) for Static Problems. Inspired by the fact that the success of the TRHC strategy largely results from decomposing a complex dynamic process into a serial of temporally associated subprocesses, here we are thinking of how to extend the basic idea of TRHC in order to decompose a large-scale static problem into a serial of associated subproblems (please note that conventional partitioning methods decompose a static problem into a set of separated subproblems). Then in what terms could subproblems be associated in static environments? Basically, we need to create a problem-specific artificial space, project into the space all parts that compose a solution to the original static problem and then design a spatial horizon which recedes in the space. As the spatial horizon recedes out, the value/status of each part will be optimized along together with all other parts that are within the current horizon scope. Once the values/statuses of all parts are optimized, a final solution to the original static problem is determined. Now, one can see that subproblems will be spatially associated in the artificial space. Therefore, hereafter, we call our new strategy for decomposing static problems as spatial receding horizon control (SRHC).

After an artificial space is designed and all parts that compose a solution are projected into the space, it is crucial to design a spatial horizon receding process to decompose the original static problem into a serial of spatially associated subproblems. A basic spatial horizon receding process can be described as follows. Suppose a solution to a large-scale static problem is composed of M local parts. The SRHC strategy makes use of spatial structure (where positions indicate strength of influence between parts of a solution) to move from purely local, part-by-part, optimization to using information from the neighbouring, subglobal context. An optimization algorithm is applied M times to determine the M parts in a solution. Starting with a specified part, the algorithm calculates at each time step the N new parts (usually $N \ll M$), which are the most associated with the *decided* parts, (i.e., parts which have already been optimized in the previous iterations). Although the algorithm will optimize N parts each time, only the part that is the most associated with the decided parts will be added to the list of decided parts. The other $N - 1$ parts will be discarded to be recalculated in later iterations. The algorithm keeps running until all M parts have been optimized. This leads to a general N -step-wide static problem partitioning method, as illustrated in Figure 3.

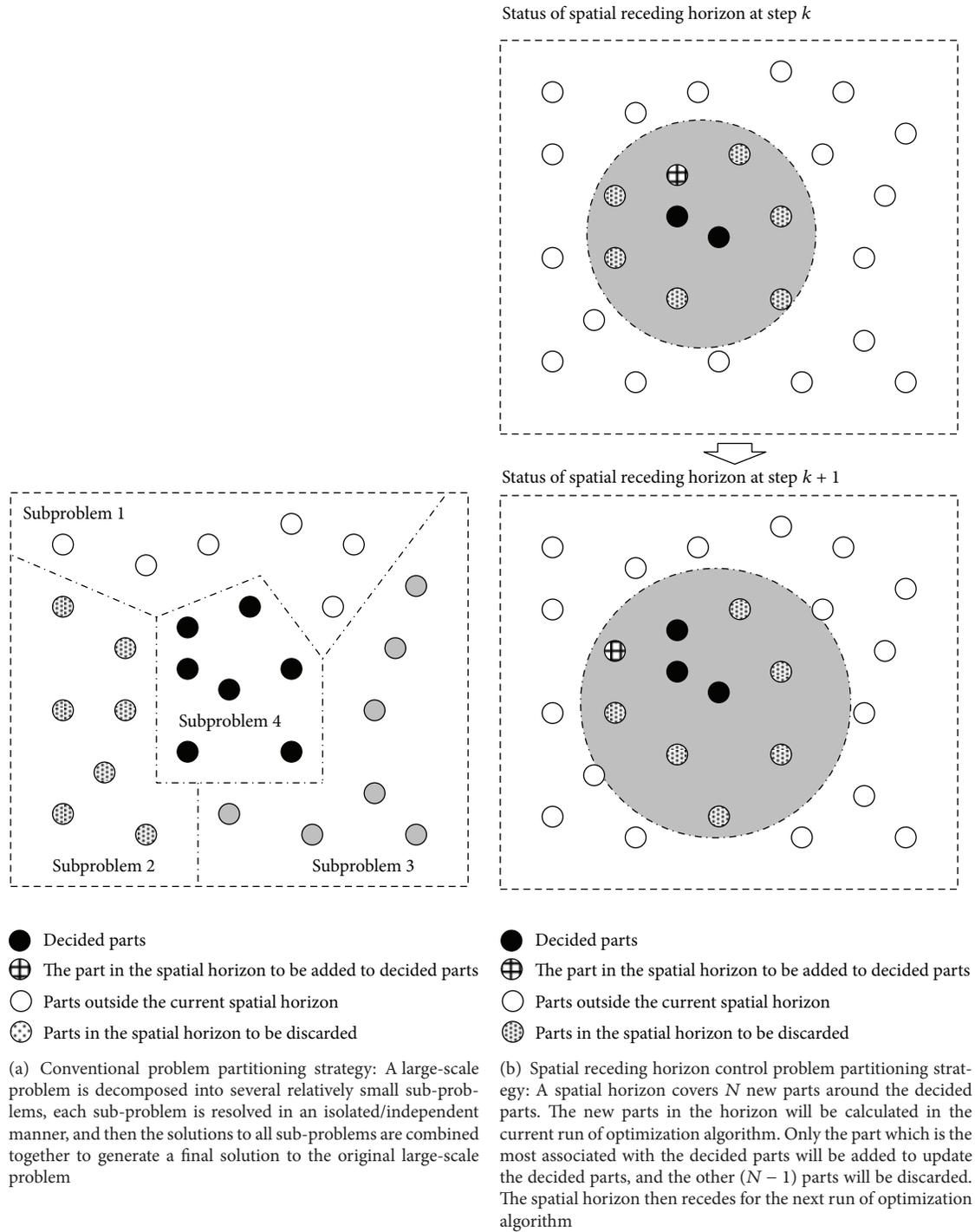


FIGURE 3: Illustration of Spatial Receding Horizon Control (SRHC).

Existing problem-partitioning methods may be considered as a one-step-wide SRHC strategy; that is, each part of a solution is determined in an isolated manner; for example, see [13]. In the generalized N -step-wide SRHC strategy, each part is calculated by referring to its most relevant surrounding parts. In other words, subglobal information is used in the determination of a local part. The extra information

considered by the N -step-wide SRHC strategy can improve the quality of each part and that of the global solution.

Apparently, the design of the artificial space and the spatial horizon receding process is a highly problem-specific task. In this paper, we will particularly discuss how to apply the SRHC strategy to decompose the NCP. After seeing those successful implementations of TRHC based evolutionary

algorithms in dynamic environments [20–22], we will also make an attempt to investigate whether integrating SRHC into GAs can deliver a powerful algorithm to resolve the static large-scale NCP.

2.3. SRHC and GA: A Perfect Match. Like the TRHC scheme having an online optimizer, the proposed SRHC also needs to run optimization repeatedly as the spatial horizon recedes step by step. General speaking, any optimization algorithm, deterministic or population-based, can be used by the SRHC strategy as long as it suits the concerned problem. However, in this study, we choose GA, because the SRHC strategy and population-based algorithms like GAs are a naturally perfect match to resolve large-scale static problems. On one side, population-based algorithms are very costly in terms of computational time and resources [23, 24]. Such computational costs often soar up exponentially as the problem scale increases. Therefore, an effective problem decomposing method like the proposed SRHC is crucial for a population-based algorithm to apply to large-scale problems. On the other side, like all other problem partitioning methods, losing global optimality or having shortsighted performance is still an issue the proposed SRHC has to address. If an algorithm, such as a deterministic algorithm, only outputs a single solution, then due to the receding horizon, the subsolutions for decided spatial steps will be uniquely determined and have no chance to change in future runs, as illustrated in Figure 4(a). The uniqueness of the subsolutions for decided spatial steps is a major cause of losing global optimality, because an optimal subsolution calculated within a spatial receding horizon may not be optimal or even good at all from a global point of view. If a population-based algorithm is employed, then the optimization within a spatial receding horizon will generate a population of solutions. Some top solutions in the population may usually have different subsolutions for decided spatial steps. A subsolution pool for decided spatial steps can then be set up according to such top solutions in the population. In the optimization of next spatial receding horizon, it needs not only to calculate those subsolutions covered by the new spatial receding horizon, but also to choose subsolutions from the pool for decided spatial steps. This is illustrated in Figure 4(b). It should be noted that the subsolution pool for decided spatial steps not only records independent subsolutions for each decided spatial step, but more importantly, also records the combination relationships between them as in the associated top solutions. Regarding the decided spatial steps in the new run of optimization, it actually only needs to choose a combination relationship saved in the pool. This can significantly reduce the search space for decided spatial steps. For instance, in Figure 4(b), the independent subsolutions saved in the pool may have at least 6 combinations for decided spatial steps, but the choice needs to be made between only 3 combinations as given by the previous run of optimization. At the same time, the global performance can be effectively improved, because some flexibility in the subsolutions for decided spatial steps is introduced by the pool referring to some top solutions of the previous spatial receding horizon. Therefore,

a population-based algorithm like GA can help to improve the global performance of the SRHC. Figure 5 summarizes the combination of the SRHC scheme with GA as a flowchart.

3. Modeling NCP Based on SRHC

3.1. Conventional Model of NCP. Suppose a network, denoted as $G\{V, E\}$ hereafter, where V and E are sets of vertices and edges, that is, nodes and unit-capacity links in this paper, respectively, has n_n nodes and n_l links. This paper considers only the one-source-multisink SNCP, so, all data originate from a certain node and need to go to some other nodes. For the sake of simplicity, but without losing generality, in this paper it is assumed that the source is always node 1 in the network. Let n_d be the number of signals originating from the source, n_s be the number of sinks in $G\{V, E\}$, and R_{target} be the target rate which is expected to be achieved at every sink. Basically, a network protocol and coding scheme define how each node in the network forwards, replicate, and/or encodes data. For instance, assuming that node i has $n_{\text{In}}(i)$ incoming links and $n_{\text{Out}}(i)$ outgoing links and the signal on the j th incoming link is $s_{\text{In}}(i, j)$, then a network protocol and coding scheme can be viewed as a mapping process to generate the signals on outgoing links; that is, $s_{\text{Out}}(i, j)$, $i = 1, \dots, n_n$, $j = 1, \dots, n_{\text{Out}}(i)$. Mathematically, a network protocol and coding scheme can be denoted as

$$M_{\text{NPCS}} : \{G(n_n, n_l), s_{\text{In}}(1, h)\} \longrightarrow s_{\text{Out}}(i, j) \quad (1)$$

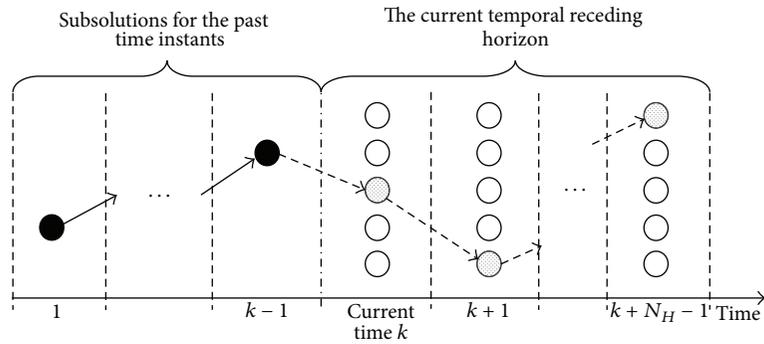
$$h = 1, \dots, n_d, \quad i = 1, \dots, n_n, \quad j = 1, \dots, n_{\text{Out}}(i).$$

The most widely used coding operation is linear network coding, which can be mathematically formulated as follows for an outgoing link:

$$s_{\text{Out}}(i, j) = \sum_{h=1}^{n_{\text{In}}(i)} w(i, j, h) s_{\text{In}}(i, h), \quad (2)$$

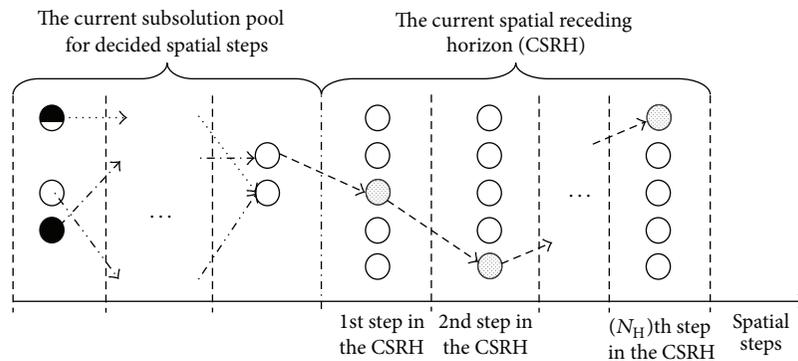
$$i = 1, \dots, n_n, \quad j = 1, \dots, n_{\text{Out}}(i),$$

where $w(i, j, h)$, $h = 1, \dots, n_{\text{In}}(i)$ are weights determining how to combine the $n_{\text{In}}(i)$ incoming signals of node i to generate a signal for the j th outgoing link of node i . In theory, $w(i, j, h)$ may be continuous, but as proved by [25, 26], sufficient finite discrete values for $w(i, j, h)$ can guarantee that the maximum possible throughput is achieved. Therefore, in this paper, $w(i, j, h)$ will choose its value from a finite set Θ_W . Assuming that Θ_W has $N_W \geq 2$ discrete values, then the field size for network coding is N_W in this study. A linear coding scheme is actually defined by a set of $w(i, j, h)$, in other words, all that is required in order to design a linear coding scheme is the appropriate choice of $w(i, j, h)$. Apparently, a network protocol and coding scheme is actually determined by the set of $w(i, j, h)$. For a given network protocol and coding scheme, suppose the numbers of coding nodes and links are N_{CN} and N_{CL} , respectively, and the actually achieved rate at sink i is $R(i)$.



- Unique combination relationship determined by the subsolutions for the past time instants
- > Undecided combination relationship within the current temporal receding horizon
- A candidate subsolution for a time instant (different time instant may have different candidate subsolutions to choose from)
- A decided subsolution for a past time instant, which is unchangeable in any future run of optimization
- ◐ A subsolution newly calculated for a time instant within the current temporal receding horizon

(a) In temporal receding horizon control, no matter what kind of method, deterministic method or population-based algorithm, is used as the online optimizer, the sub-solutions for the past time instants will have been uniquely decided and executed, and the sub-solutions for the future time instants will be optimized only based on the unchangeable consequence of the past sub-solutions. Therefore, short-sighted behaviours are common in temporal receding horizon control, because the unchangeable past sub-solutions might not be optimal or even good in terms of the performance over the entire time scope. A combination of spatial receding horizon control and deterministic method has the same solution pattern (except the time axis is replaced by a spatial axis)



- ⋯→ Fixed combination relationships given by candidates in the current subsolution pool for decided spatial steps
- > Undecided combination relationship within the CSRH, which needs to be calculated in the current run of optimization
- A candidate subsolution for a spatial step (different spatial step may have different candidate subsolutions to choose from)
- A subsolution for a decided spatial step, which may change depending on which candidate in the current pool for decided spatial steps is chosen
- ◐ A subsolution newly calculated for a spatial step within the CSRH

(b) In spatial receding horizon control combined with population-based algorithm, the sub-solutions for decided spatial steps are not fixed or executed. Based on some top different solutions (e.g., the best 10 different solutions) calculated in the last run of optimization, a sub-solution pool is set up for decided spatial steps. Different candidates in the pool may have different sub-solutions for a same decided spatial step. Therefore, in the current run of optimization, besides calculating the sub-solutions for the spatial steps within the current spatial receding horizon (CSRH), it also needs, for the sake of optimality, to choose a candidate from the pool for those decided spatial steps

FIGURE 4: Patterns of sub-solutions in TRHC and SRHC.

With the above preparation, the NCP in this paper is formulated as the following maximization problem:

$$\max_{M_{NPCS}} f_1 = \max_{w(i,j,h)} f_1, \quad (3)$$

$$i = 1, \dots, n_n, \quad j = 1, \dots, n_{Out}(i), \quad h = 1, \dots, n_{In}(i),$$

where

$$f_1 = \begin{cases} \alpha_1 \min(R(i)) + \alpha_2 \text{ave}(R(i)) \\ + \frac{\alpha_3}{(N_{CL} + 1)} \\ + \frac{\alpha_4}{(N_{CN} + 1)}, & \min(R(i)) < R_{Target}, \\ \alpha_1 \min(R(i)) + \alpha_2 \text{ave}(R(i)) \\ + \frac{\alpha_5}{(N_{CL} + 1)} \\ + \frac{\alpha_6}{(N_{CN} + 1)}, & \min(R(i)) \geq R_{Target}, \end{cases} \quad (4)$$

$$i = 1, \dots, n_s$$

$\alpha_k, k = 1, \dots, 6$, are weights, and

$$\begin{aligned} \min(\alpha_1, \alpha_2) &> \max(\alpha_3, \alpha_4), \\ \min(\alpha_5, \alpha_6) &\gg \max(\alpha_1, \alpha_2), \end{aligned} \quad (5)$$

subject to $G(n_n, n_l)$. Clearly, this maximization problem aims to find a network protocol and coding scheme to maximize f_1 defined by (4) and (5). From the above objective function, one can see that the NCP will firstly try to maximize the overall actually achieved rate, and once the target rate is achieved, the focus of the optimization will switch to minimizing the network coding resources. The term “ $\min(R(i))$ ” and term “ $\text{ave}(R(i))$ ” in (4) can be used to assess the actually achieved rate. Basically, a larger term value for “ $\text{ave}(R(i))$ ” is desirable. $R(i)$ should be optimized as evenly as possible; that is, increasing the rate at some sinks by largely sacrificing the rate at other sinks should be avoided. This can be reflected by the term value for “ $\min(R(i))$ ”; that is, the larger the value is, the more evenly $R(i)$ is optimized. At the same time, as reflected by the term “ $1/(N_{CL} + 1)$ ” and the term “ $1/(N_{CN} + 1)$,” the network coding resources should be minimized, particularly when the target rate can be achieved, that is, when $\min(R(i)) \geq R_{Target}$.

3.2. SRHC Based Model for NCP. To design an SRHC based model for the NCP, firstly we need to create an artificial space, then to project all network nodes into the space, then to design a spatial horizon receding process, and at last to reformulate the maximization problem given by (3) to (5) in order to make it fit in the SRHC framework.

Usually, a network where coding needs to be performed already defines its own space (real or virtual) and may have its

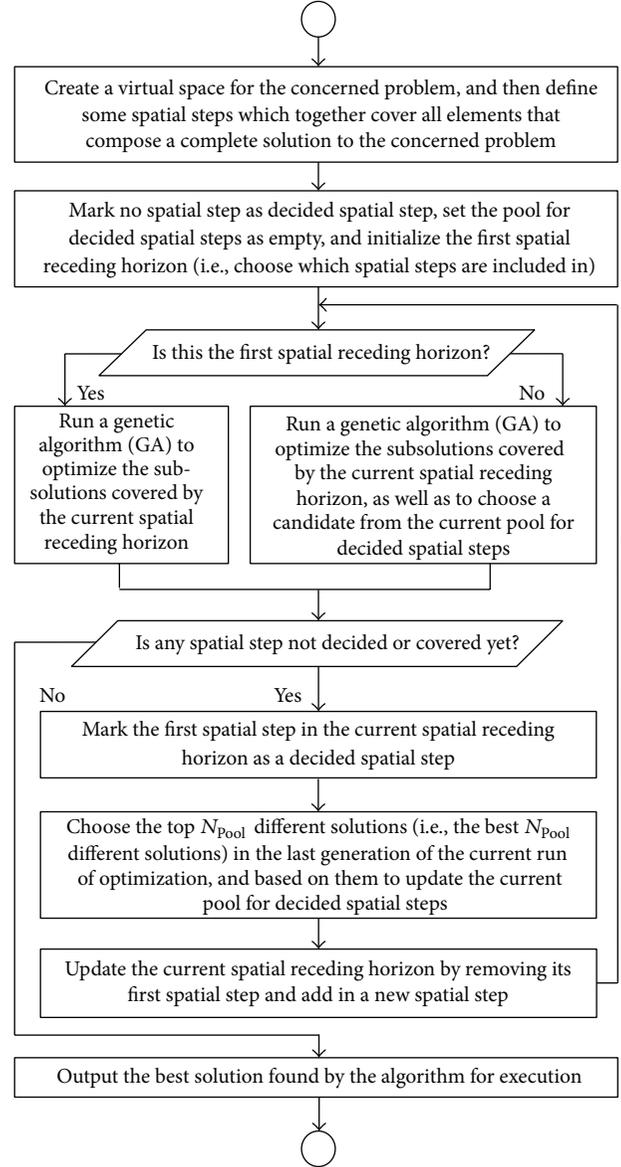


FIGURE 5: Flowchart of SRHC with GA as online optimizer.

nodes distributed in the space in a rather random manner, but such a space and the distribution of nodes are of little use to the design of artificial space and the projection of nodes in the SRHC model for the NCP. In the SRHC model, we simply use a purely imaginary two-dimensional space and then project network nodes into the space according to the connections between nodes. The projecting procedure is described as follows.

Step 1. Let $M_{LN}(i)$ be the set that records all nodes in the i th node layer, and $M_{LL}(i)$ records all links in the i th link layer. Start from the source, that is, node 1. Set node 1 as the only node in $M_{LN}(1)$, and set the end nodes of all outgoing links of node 1 as the nodes in $M_{LN}(2)$. Then set the current layer $l_C = 2$. Let Ω_N be the set of all nodes that are not included in $M_{LN}(1)$ and $M_{LN}(2)$.

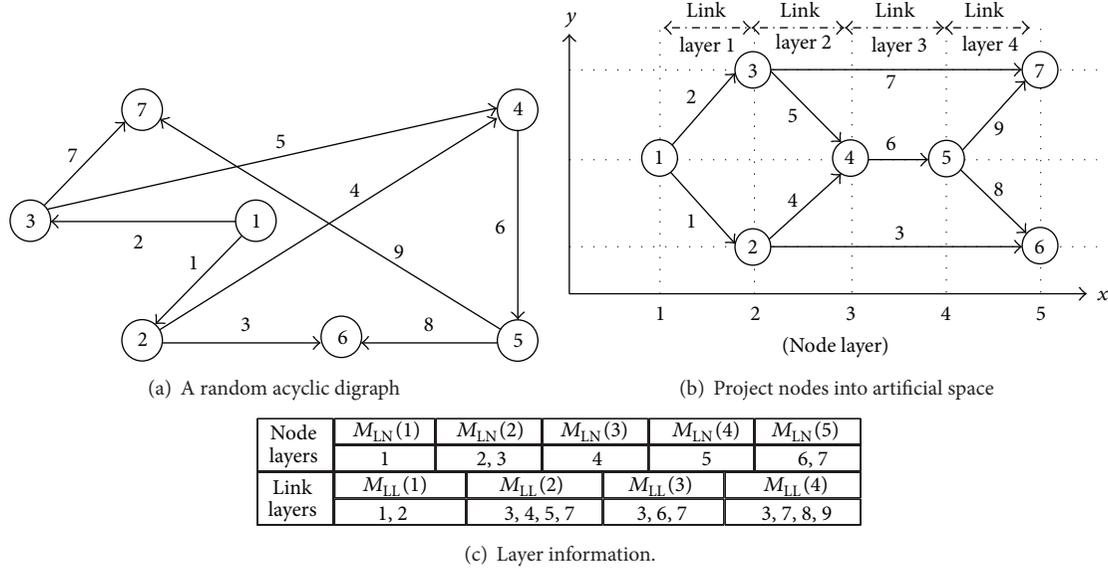


FIGURE 6: An illustration of node projecting procedure.

Step 2. While $\Omega_N \neq \emptyset$, do

Substep 2.1. Put all end nodes of all outgoing links of the nodes in $M_{LN}(l_C)$ as the nodes in $M_{LN}(l_C + 1)$.

Substep 2.2. If a node in $M_{LN}(l_C + 1)$ is already included in $M_{LN}(i)$, $i = 1, \dots, l_C$, then remove this node from $M_{LN}(i)$, and add it to Ω_N .

Substep 2.3. Remove all nodes of $M_{LN}(l_C + 1)$ from Ω_N . Let $l_C = l_C + 1$.

Step 3. Create a two-dimensional space, where the x axis is the node layer number, and the y axis has no specific meaning. Then project all nodes into the space according to M_{LN} . For instance, suppose a node belongs to $M_{LN}(i)$. Then the x coordinate of this node is i . The y coordinate of this node can be random, but for distinguishing purposes, the nodes in the same layer should be assigned with different values of y .

Step 4. For a link, suppose its starting node is within $M_{LN}(i)$ and its end node within $M_{LN}(j)$, $j > i$. Then add this link to $M_{LL}(i), \dots, M_{LL}(j - 1)$.

Figure 6 gives a simple illustration about the above node projecting procedure. The information of node layers and link layers is crucial not only to define the spatial horizon, but also to design the spatial horizon receding process for the NCP. As illustrated in Figure 7, the spatial horizon for the NCP is defined based on link layers. In each iteration of optimization, the spatial horizon covers some successive link layers; for example, in the case of Figure 7, the spatial horizon spans over two successive link layers. In an iteration of optimization, only those links that are covered by the current spatial horizon will be optimized. The spatial horizon

recedes for one link layer each time along the x axis in the artificial space. In the new iteration of optimization, those links that have been optimized in the previous iteration of optimization and get out of the current spatial horizon due to the horizon receding process will be fixed as decided links, whilst those links that have been optimized in the previous iteration of optimization but are still within the current spatial horizon will be optimized again along with the links that are newly covered by the spatial horizon. This process continues until all links become decided links. The details of the spatial horizon receding process are given as follows.

Step 1. Set up N_H , the length of the spatial horizon. Let Ω_S be the set of sinks, $\Omega_{DL} = \emptyset$ the set of decided links, and $\Omega_{UL} = \{M_{LL}(1), \dots, M_{LL}(N_{LL})\}$ the undecided links, where N_{LL} is the number of total link layers. Let $k = 1$.

Step 2. While $\Omega_{UL} \neq \emptyset$, do

Substep 2.1. Set the current spatial horizon as $\Omega_H(k) = \{M_{LL}(k), \dots, M_{LL}(k + N_H - 1)\}$. It should be noted that $M_{LL}(i) = \emptyset$ for $i > N_{LL}$.

Substep 2.2. Let $W(k)$ be the weights for the links in $\Omega_H(k)$, $M_{EN}(k + N_H - 1)$ the set of all end nodes of the links in $M_{LL}(k + N_H - 1)$, and $N_{CL}(k)$ and $N_{CN}(k)$ are the current numbers of coding links and coding nodes, respectively. Then calculate the following maximization problem:

$$\max_{W(k)} f_2(k), \tag{6}$$

subject to the signals on Ω_{DL} , where $f_2(k)$ is a new objective function defined as follows:

$$f_2(k) = \beta_1 f_{NT}(M_{EN}(k + N_H - 1)) + \beta_2 f_{MR}(M_{EN}(k + N_H - 1))$$

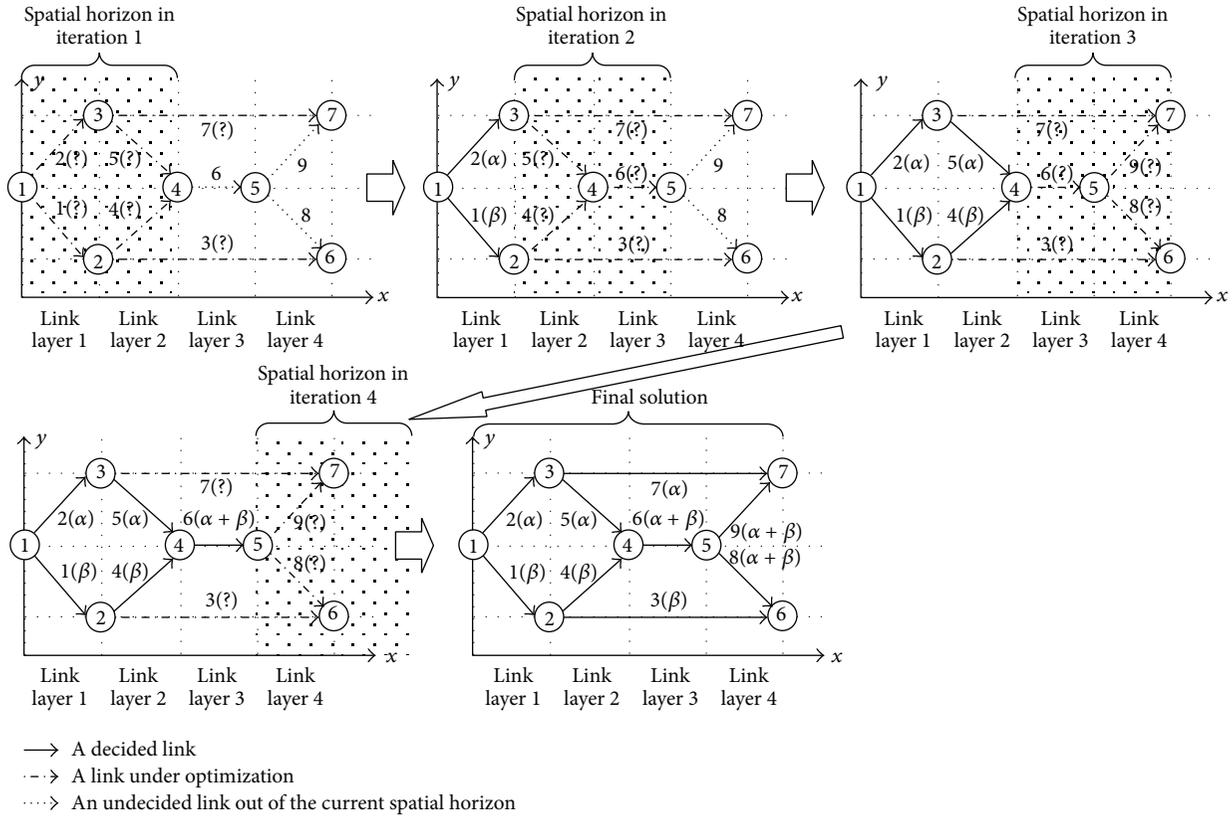


FIGURE 7: An illustration of spatial horizon receding process.

$$\begin{aligned}
 & + \beta_3 f_{SD}(M_{EN}(k + N_H - 1)) + \frac{\beta_4}{(N_{CL}(k) + 1)} \\
 & + \frac{\beta_5}{(N_{CN}(k) + 1)} + \beta_6 f_{TP}(k, \Omega_S),
 \end{aligned} \tag{7}$$

which will be explained later.

Substep 2.3. Remove $M_{LL}(k)$ from Ω_{UL} to Ω_{DL} ; that is, $\Omega_{UL} = \Omega_{UL} - M_{LL}(k)$ and $\Omega_{DL} = \Omega_{DL} + M_{LL}(k)$. Let $k = k + 1$.

In the above spatial horizon receding process, a new maximization problem as defined by (6) and (7) needs to be resolved during each iteration of optimization. In the new objective function $f_2(k)$, the term $f_{NT}(M_{EN}(k + N_H - 1))$ is a function that calculates the network throughput at all nodes in $M_{EN}(k + N_H - 1)$, the term $f_{MR}(M_{EN}(k + N_H - 1))$ is a function that calculates the minimal rate over the nodes in $M_{EN}(k + N_H - 1)$, the term $f_{SD}(M_{EN}(k + N_H - 1))$ is a function that assesses how much the signals on $M_{EN}(k + N_H - 1)$ are diversified, the term $f_{TP}(k, \Omega_S)$ is a terminal penalty which assesses the impact of the current stage solution on the network throughput at the sinks, and β_1 to β_6 are weights to combine different terms. Apparently $f_2(k)$ is quite different from the objective function of the conventional NCP model, that is, f_1 as defined in (4), mainly because of two new terms: $f_{SD}(M_{EN}(k + N_H - 1))$ and $f_{TP}(k, \Omega_S)$. The reason for introducing $f_{SD}(M_{EN}(k + N_H - 1))$

is illustrated in Figure 8, where one can see that, assuming all other terms in $f_2(k)$ are the same, Figure 8(a) is better than Figure 8(b) because the signals on $M_{EN}(k + N_H - 1)$ are better diversified, which means the downstream nodes will have more choices. The introduction of $f_{TP}(k, \Omega_S)$ is in line with the common practice of TRHC in the area of control engineering, which aims to minimize shortsighted behaviors, such as getting trapped in local optima and generating unstable/unconverged solutions, due to the fact that not all information is covered by the receding horizon. Before running the SRHC model, we need to count the number of downstream sinks for every node in the network. Then we can roughly assess the final impact of the signals received by a node. Basically, a node with more downstream sinks deserves a higher priority to receive more signals, just like Figure 9 illustrates.

The absolute value of different term in J_2 may vary in quite different range; for example, f_{NT} may be over 1000 whilst f_{MR} may be smaller than 10. This means the absolute values of different terms in J_2 are usually incomparable, and then they cannot be directly combined by J_2 . Therefore we need to unify the terms of J_2 ; in other words, we have to use the relative value of each term, as defined by the following:

$$\begin{aligned}
 f_3(k) = & \beta_1 \bar{f}_{NT}(M_{EN}(k + N_H - 1)) \\
 & + \beta_2 \bar{f}_{MR}(M_{EN}(k + N_H - 1))
 \end{aligned}$$

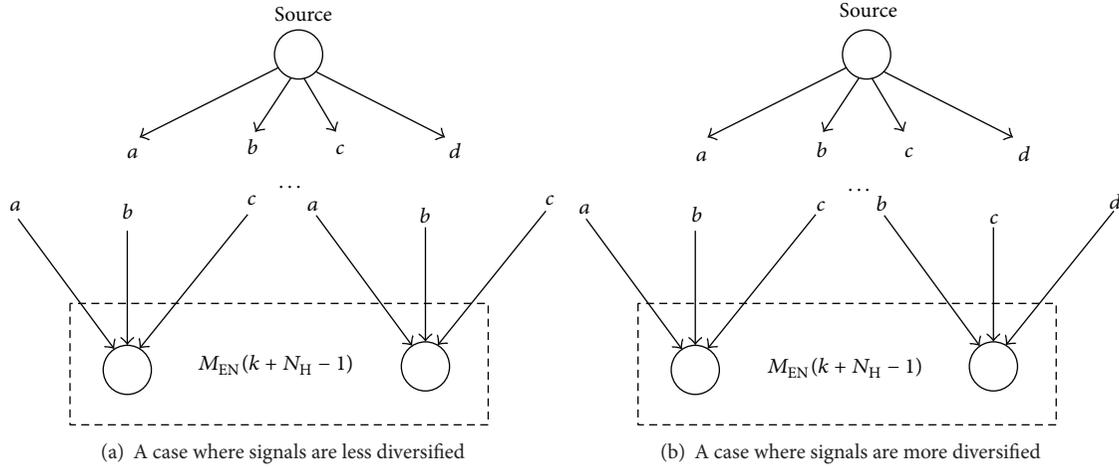


FIGURE 8: An illustration of signal diversification.

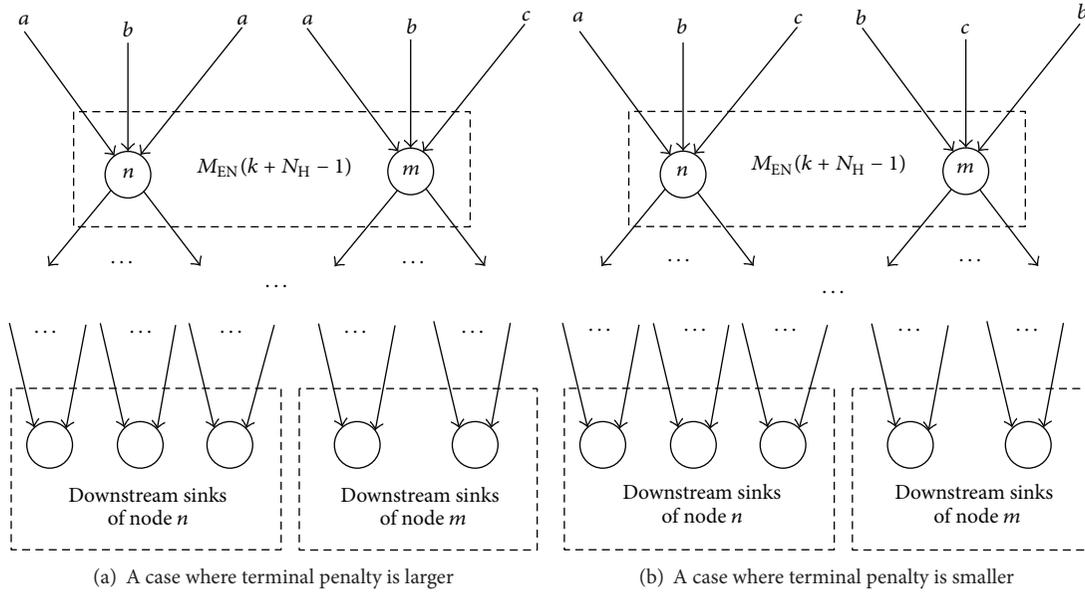


FIGURE 9: An illustration of terminal penalty.

$$+ \beta_3 \bar{f}_{SD}(M_{EN}(k + N_H - 1)) + \frac{\beta_4}{(N_{CL}(k) + 1)} \quad (8)$$

$$+ \frac{\beta_5}{(N_{CN}(k) + 1)} + \beta_6 \bar{f}_{TP}(k, \Omega_S),$$

$$\bar{f}_{NT}(M_{EN}(k + N_H - 1)) = \frac{f_{NT}(M_{EN}(k + N_H - 1))}{F_{NT}(M_{EN}(k + N_H - 1))}, \quad (9)$$

$$\bar{f}_{MR}(M_{EN}(k + N_H - 1)) = \frac{f_{MR}(M_{EN}(k + N_H - 1))}{F_{MR}(M_{EN}(k + N_H - 1))}, \quad (10)$$

$$\bar{f}_{SD}(M_{EN}(k + N_H - 1)) = \frac{f_{SD}(M_{EN}(k + N_H - 1))}{F_{SD}(M_{EN}(k + N_H - 1))}, \quad (11)$$

$$\bar{f}_{TP}(k, \Omega_S) = \frac{f_{TP}(k, \Omega_S)}{F_{TP}(k, \Omega_S)}, \quad (12)$$

where F is a function to calculate the value of a term in an ideal condition; for example, F_{NT} and F_{MR} assume that every node in $M_{EN}(k + N_H - 1)$ receives as many different signals as its incoming links or all signals sent from the source, whichever is larger, F_{SD} assumes that $M_{EN}(k + N_H - 1)$ receives as many different signals as the links in $M_{LL}(k + N_H - 1)$ or all signals sent from the source, whichever is larger, and F_{TP} assumes all signals received by a node in $M_{EN}(k + N_H - 1)$ can be sent to its every downstream sink. Apparently, the value of a unified term, that is, \bar{f} , is within $[0 \ 1]$, and this makes f_3 more reasonable and easier to tune. The design details of \bar{f} and F may vary, and due to limited space, here we skip them.

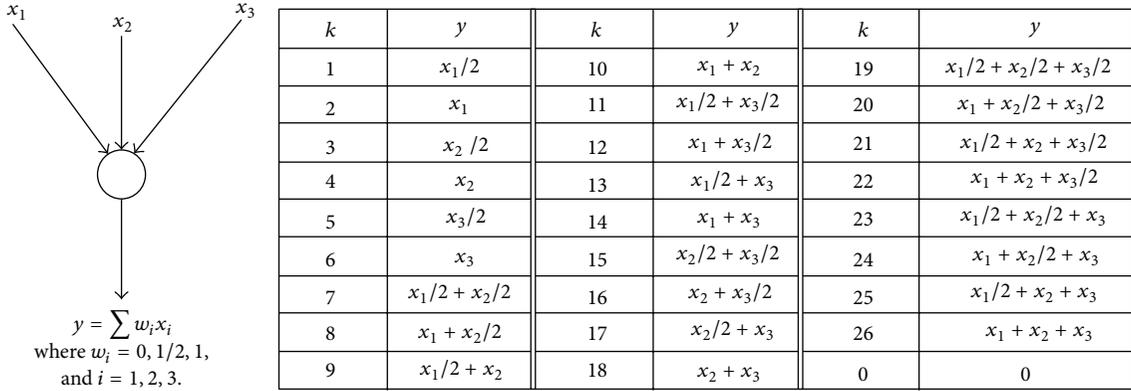


FIGURE 10: An illustration of definition of signal combinations.

4. SRHC Based GA for NCP

The design of GAs usually includes choosing an appropriate chromosome structure, developing effective evolutionary operators, introducing useful problem-specific heuristic rules, and adjusting algorithm-related parameters. This section will explain the first three aspects, and the last aspect will be discussed in the experiment section. Here we will firstly spend three subsections to describe some useful GA-related designs reported in [10]. Then some SRHC-related modifications to the GA designs will be discussed in order to properly integrate the GA into the SRHC method for the NCP.

4.1. Chromosome Structure. The chromosome structures of the GAs in [2, 7, 8] are based on the use of a binary matrix to record the active states of links, and such structures make it easy to apply graph theoretic methods to ascertain whether the target rate is or is not achievable by a given chromosome. A chromosome in [2, 7, 8] does not have full information concerning a specific network protocol and coding scheme and may associate with different specific network protocols and coding schemes. Although this means to some extent random linear coding can be employed, it may be difficult to determine the exact information flow on links. The lack of exact information flow on links will make it difficult not only to calculate the actually achieved rate at sinks, but also to integrate useful NCP-specific heuristic rules into the algorithms. Therefore, in this paper we construct chromosomes based on a permutation representation, in order to record exact information flow on links.

A permutation representation is often used when GAs are being applied to combinatorial problems (actually, the NCP is a combinatorial problem), because it can usually construct chromosomes straightforwardly based on their physical meanings. However, such a representation is often confronted by feasibility problems; that is, a chromosome may become infeasible in terms of its physical meaning during evolutionary operations. Sometimes some evolutionary operators have to be modified significantly or even discarded

in order to resolve such problems. For the NCP, a straightforward permutation representation is to use the absolute information flow on links to construct chromosomes, but this representation will cause serious feasibility problems during evolutionary operations, because the set of feasible signals from which a link can choose cannot be predetermined and varies over time according to the signals on other links. This means that any change in the signal on a link caused by evolutionary operations could make the unchanged signals on some other links infeasible.

Fortunately, the permutation representation in [10] is free of feasibility problems without sacrificing any of the merits of permutation representations. Instead of the absolute information flow on links, a chromosome in [10] records the relative information flow, that is, an integer k , whose meaning is a certain predefined combination of signals on incoming links of a node. For instance, Figure 10 shows an illustration of how to predefine k . In Figure 10, a table is set up to define all possible signal combinations at a node with three incoming links, and the field size is $N_W = 3$. A different number of incoming links require a different predefined table for k , as illustrated in Figure 11.

Let $head(i)$ denote the serial number of the starting node of link i . It is assumed that the source has as many incoming links as there are signals to be sent, and each signal is associated with one and only one of such assumed links. Let gene i , that is, $g(i)$, be associated with link i . Then $g(i) = k$, $k = 0, 1, \dots, N_W^{n_{in}(head(i))}$, where $n_{in}(head(i))$ is the number of incoming links from node $head(i)$. In other words, for an outgoing link, for example, link i , the number of possible combinations (including no coding) is $N_W^{n_{in}(head(i))}$. The exact combination that a value of k stands for needs be predefined. Hereafter, the value of $g(i)$ is called the *state* of link i . Then the set of possible *states* for link i is

$$\Theta_S(i) = \{0, 1, \dots, N_W^{n_{in}(head(i))}\}. \quad (13)$$

Therefore, the size of the solution space of the GA is

$$n_{SP} = \prod_{i=1}^{n_l} N_W^{n_{in}(head(i))}. \quad (14)$$

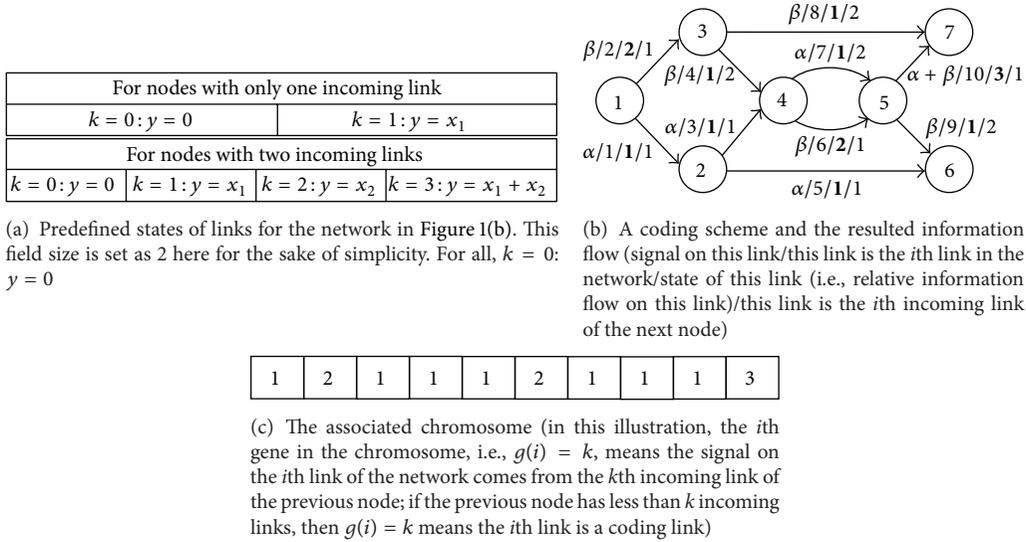


FIGURE 11: Chromosome structure based on relative information flow on links.

Unlike the absolute information flow on the links, $\Theta_S(i)$ only depends on the network topology and the number of signals that are to be sent, which are both fixed during a GA run. Therefore, as long as $g(i)$ remains within $\Theta_S(i)$ during the evolutionary operation, there will be no feasibility problem. As will be discussed in the following subsection, this condition is very easily fulfilled. On the other hand, the absolute information flow on links can be derived in a straightforward way from a chromosome of the new GA. The simple illustration in Figure 11 indicates how to use relative information flow on links to construct a chromosome.

The chromosome described above is a vector with a size of n_l . One may also use a $n_l \times \max(n_{in}(i))$ matrix to record, for each link, the weights applied to its incoming links. Such a matrix representation will need no predefined tables. In this study, we choose the vector representation because (i) it has a lower memory demand, particularly in the case of large-scale networks, and (ii) it is more efficient in terms of algorithm execution (the matrix representation requires to generate $n_{in}(head(i))$ random numbers to determine the relative information flow on link i , whilst the vector representation needs only one random number). However, for networks where a node may have many incoming links, the predefined tables for the vector representation will become enormously huge if the field size is also large. For instance, assuming $\max(n_{in}(i)) = 10$ and $N_W = 10$, then the largest predefined table will have 10^{10} entries for y . In this case, we can transform the network into an equivalent network which has a relatively small $\max(n_{in}(i))$. Actually, we can always transform a network into a new one with $\max(n_{in}(i)) = 2$, as illustrated in Figure 12, and then even if $N_W = 100$, the largest predefined table only needs $100^2 = 10^4$ entries for y . The transformed network will have more links than the original network, which means, according to (14), that the entire search space will increase, which will particularly become of concern in the case of large-scale networks if no problem partitioning method is used. Fortunately, with the

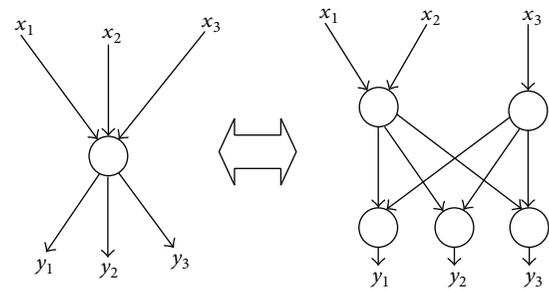


FIGURE 12: Transform a network into a new network with $\max(n_{in}(i)) = 2$.

proposed SRHC method to decompose large-scale networks, the search space during a spatial receding horizon can easily be restricted to a manageable size, no matter how large the original network scale is.

It should be noted that the search space given by (13) and (14) is much larger than those in previous studies. For instance, the search space size for link i is $2^{n_{in}(head(i))}$ in [2], and even down to $n_{in}(head(i)) + 2$ in [7, 8]. Fortunately, this disadvantage can be compensated by introducing some useful problem-specific heuristic rules based on exact information flow on links. Such heuristic rules are difficult to apply to previous chromosome structures such as in the case of [7, 8], because exact information flow on links can hardly be derived there. For instance, the modified binary representation in [7] is even “at the price of losing the information on the partially active link states that may serve as intermediate steps toward an uncoded transmission state.” Differently, the new GA reported in this study can derive the exact information flow on links associated with a chromosome and therefore can take advantage of many useful NCP-specific heuristic rules. As a result, the GA reported here may still find theoretically optimal solutions, despite the huge search space.

4.2. Evolutionary Operators. The mutation operator in this paper is designed as follows. A chromosome is chosen for mutation with probability p_m . Then a gene associated with a potential coding link needs to be chosen randomly. Suppose the i th gene, that is, $g(i)$, is chosen, whose associated link is the i th link in the network. Then the set of possible states for link i is given by $\Theta_S(i)$ as defined in (13). Mutation will randomly choose a value from the set $\Theta_S(i) - \{g(i)\}$ and then reset $g(i)$ to the new value. Since $\Theta_S(i)$ only depends on the network topology, the above mutation operation is free of feasibility problems.

This paper adopts uniform crossover, which is highly efficient in not only identifying, inheriting, and protecting common genes, but also in terms of recombining noncommon genes [17, 27]. Simply speaking, in uniform crossover, each gene of an offspring chromosome inherits the associated gene from its two parent chromosomes with a 50% chance. Thanks to the permutation representation, the i th genes of all chromosomes share the same set of possible states for link i , and, therefore, uniform crossover will cause no feasibility problems. Regarding the choice of two parent chromosomes, any chromosome in an old generation may be chosen as the first parent chromosome at a fixed probability of p_c , and then a different chromosome may be chosen as the second parent chromosome at a probability proportional to its fitness. In this way, every chromosome stands the same chance to become the first parent, while a fitter chromosome stands a better chance to cross over with most other chromosomes.

4.3. Heuristic Rules. It is well known that heuristic rules, particularly problem-specific rules, often play an important role in successful applications of GAs. What kinds of rules to introduce and how to integrate them into algorithms effectively are challenging tasks and usually need to be taken into account in the GA design stage. The permutation representation discussed in Section 4.1 makes it very easy to integrate the following NCP-specific rules.

Rule 1. All evolutionary operations only apply to potential coding nodes and links.

Rule 2. When initializing the first generation, a certain proportion of chromosomes will allow coding on all potential coding nodes, and for a potential coding node which has multiple outgoing links, choose at least one link randomly as a coding link. This rule can help to find a solution to achieve the target rate, if it is achievable, at all sinks.

Rule 3. Furthermore, in the initialization of the first generation, another proportion of chromosomes will allow no coding at all. This rule can help to explore the possibility of maximizing the rate actually achieved at the minimum cost of resources.

Rule 4. In either initialization or evolutionary operations, the states of incoming links of a potential coding node should be determined in such a way that the node will receive as many different signals as possible. In other words, the signals to a potential coding node should be diversified as much

as possible. This rule will allow as many choices as possible for network protocols and coding schemes and therefore can help to diversify a generation. It should be noted that it is the proposed permutation representation that makes it possible to integrate this rule into the algorithm, because the information flow on links associated with a chromosome can be easily checked out to see whether the signals to potential coding nodes are effectively diversified.

Rule 5. For a potential coding node with multiple outgoing links, there should be a high probability that the outgoing links have different states. This rule also takes advantage of the proposed permutation representation and can help to diversify a generation.

It should be noted that Rules 4 and 5 cannot be used by the methods reported in [2, 7–9], because the application of Rules 4 and 5 demands the availability of the exact information flow on links, which is guaranteed by the chromosome structure adopted in this paper. As will be revealed by the simulation results, Rules 4 and 5 can significantly improve the quality of chromosomes.

4.4. SRHC Related Modifications in GA. In order to properly integrate the above GA-related designs into the SRHC for the NCP, two modifications are necessary. One is related to the chromosome structure, and the other to the mutation operation. Hereafter for distinguishing purposes, a GA designed according to the above three subsections is referred to as GlobalGA, because no problem partitioning method is used, whilst a GA with the SRHC-related modifications discussed in this subsection is called SRHCGA. The SRHCGA can also be interpreted as the combination of the SRHC with GA, or an SRHC method with GA as optimizer.

As discussed in Section 2, in the SRHCGA, the optimization within a spatial receding horizon not only needs to calculate the subsolutions to the subproblems covered by the spatial receding horizon, but also has to choose the subsolutions for the subproblems in decided spatial steps. In the case of applying the SRHCGA to the NCP, besides assigning relative signals to the undecided links covered by the current spatial receding horizon, the optimization within the horizon will also choose a combination of relative signals for all decided links, and all candidate combinations are saved in a pool which is updated as the spatial horizon recedes. To be able to make such a choice for decided links, we need an additional special gene in the chromosome structure to record which candidate combination in the pool is chosen. This can be easily done by modifying the chromosome structure in Section 4.1 as follows. Suppose there are N_{SP} candidate combinations in the current pool for decided links and N_{ULSRHC} undecided links in the current spatial receding horizon. Then the modified chromosome structure has $(N_{ULSRHC} + 1)$ genes in total. The first gene records which candidate combination in the pool is chosen for decided links, that is, $g(1) = m$, $m \in \{1, \dots, N_{SP}\}$, means that the m th candidate combination saved in the pool has been chosen to set up the relative signals on decided links. The following N_{ULSRHC} genes, that is, $g(i)$, $i = 1, \dots, N_{ULSRHC}$, record

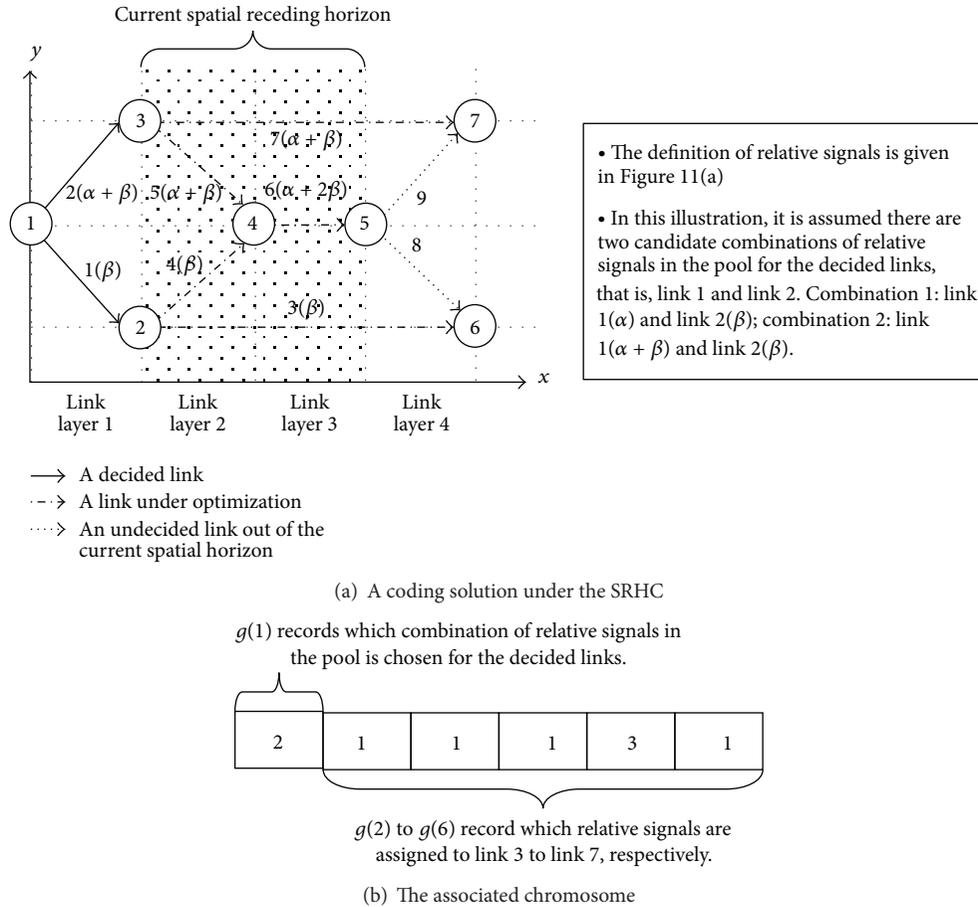


FIGURE 13: An illustration of the modified chromosome structure for SRHC.

the relative signals assigned to the undecided links covered by the spatial horizon, and their definition is exactly the same as described in Section 4.1. All candidate combinations in the pool for decided links are predetermined by the optimization of the previous spatial receding horizon and then are fixed during the optimization of the current spatial horizon. Therefore, the new $g(1)$ will cause no feasibility problem, just like other genes which record relative signals. Figure 13 gives an illustration of the modified chromosome structure for SRHC.

The second SRHC-related modification is made to the mutation operation given in Section 4.2. Actually, the modification is minor: the search space for the new $g(1)$ is not given by (13), and $g(1)$ always mutates within $\{1, \dots, N_{SP}\} - g(1)$.

5. Experimental Results

In this section, we will firstly test whether the GlobalGA, which employs no problem partitioning method, is effective to resolve the NCP. Then, we will investigate the performance of the proposed SRHC scheme by comparing the SRHCGA with the GlobalGA. There are two sets of networks, see Table 1 for details, which are taken from [2] for comparative purposes. The networks in Set I are actually generated by the algorithm in [28], which constructs connected acyclic

directed graphs uniformly at random. There are two networks used for simulations in Set I: one network, denoted as Case I-1, has 20 nodes, 80 links, 12 sinks, and rate 4, and the other network, denoted as Case I-2, has 40 nodes, 120 links, 12 sinks, and rate 3. The networks in Set II are constructed by cascading a number of copies of network (b) in Figure 1 such that the source of each subsequent copy of network (b) in Figure 1 is replaced with an earlier copy's sink. Set I has 4 networks, which uses fixed-depth binary trees containing 3, 7, 15, and 31 copies of network (b) in Figure 1, respectively. These 4 networks in Set II are referred to as Case II-1 to Case II-4 in this section. These 4 networks have a maximum multicast rate of 2, which is achievable without coding; that is, the optimal solutions have no coding links.

5.1. Tests on GlobalGA. Here the GlobalGA developed in this paper will be tested by comparing it with the GA reported in [2] (denoted as GA[2]) and two minimal approaches reported in [4, 5] (denoted as Minimal 1 and Minimal 2, resp.). As discussed in Section 4, the permutation representation makes it very easy to integrate many problem-specific heuristic rules, that is, Rule 1 to Rule 5 given in Section 4.3, which are expected to improve the performance of the GlobalGA. In order to examine whether such heuristic rules really work,

TABLE 1: Networks used in different test cases.

	Copy the network Figure 1(b) or generated by [28]	Nodes	Links	Sinks	Target rate
Case I-1	Generated by [28]	20	80	12	4
Case I-2	Generated by [28]	40	120	12	3
Case II-1	3 copies of Figure 1(b)	19	30	4	2
Case II-2	7 copies of Figure 1(b)	43	70	8	2
Case II-3	15 copies of Figure 1(b)	91	150	16	2
Case II-4	31 copies of Figure 1(b)	187	300	32	2

TABLE 2: Comparative results with existing methods (number of coding links).

	Case I-1		Case I-2		Case II-1		Case II-2		Case II-3		Case II-4	
	Best	Ave.	Best	Ave.	Best	Ave.	Best	Ave.	Best	Ave.	Best	Ave.
Minimal 1	0	1.35	0	1.85	3	3.00	7	7.00	15	15.00	31	31.00
Minimal 2	0	1.85	0	1.90	0	2.15	2	4.70	7	11.60	28	52.80
GA [2]	0	1.20	0	1.05	0	0.65	0	2.15	3	5.35	12	17.20
GlobalGA1	0	1.20	0	0.80	0	0.00	0	0.00	0	0.80	0	6.30
GlobalGA2	0	1.15	0	0.70	0	0.00	0	0.00	0	0.30	0	5.00
GlobalGA3	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00

three versions of the GlobalGA are used in the experiments: the first version, denoted as GlobalGA1, only employs Rules 1 and 2, the second version, denoted as GlobalGA2, uses one more rule, that is, Rule 3, than GlobalGA1, and the third version, denoted as GlobalGA3, adopts Rule 1 to Rule 5. The reason why GlobalGA1 is included is because it employs exactly the same heuristic rules as used in GA[2]; therefore, any difference in performance between GlobalGA1 and GA[2] should mainly result from the basic designs, for example, chromosome structures and associated operations, used in [2] and those used in this paper. The reason for including GlobalGA2 is because there is no difficulty in applying Rule 3 to GA[2]. Since Rule 3 can improve the performance of GlobalGA2, one can expect that Rule 3, once applied, might also benefit GA[2]. As mentioned in Section 4, it is because of the permutation representation that the integration of Rules 4 and 5 becomes possible; therefore, GlobalGA3 will reveal the extent to which the GlobalGA reported in this paper is advantageous.

To make a fair comparison, GlobalGA1 to GlobalGA3 have the same population size (150) and upper bound (300) for the number of generations for evolution as GA[2] does. Then 20 random runs of each algorithm are conducted, and the average results are listed in Tables 2 and 3 reveal more details about the performance of GlobalGA1 to GlobalGA3. For the sake of simplicity, in most parts of the simulation $w(i, j, h) = 0$ or 1 in (2), that is, the field size $N_W = 2$, unless specified otherwise. From these results one can make the following observations.

- (i) Table 2 shows that, in the cases of Set I, that is, Case I-1 and Case I-2, all the methods perform similarly. In more precise terms, the GlobalGAs reported in this paper, that is, GlobalGA1, GlobalGA2, and GlobalGA3, return slightly lower average numbers of coding links than the existing methods. However, since all methods can find the optimal (i.e., no coding

required), or almost optimal solutions to both the cases in Set I, we cannot claim that our algorithm has a significant advantage compared with existing algorithms. Analysis of the network topologies in Set I suggests that these networks have too many links; for one network, $n_l = 4n_n$, and for the other, $n_l = 3n_n$. In the Graph Drawing Community, graphs (i.e., networks) having $n_l = 4n_n$ links are actually considered to be dense [28]. In such a network with dense links, it is easy to achieve a relatively small target rate without network coding. Compared with Set I, all the networks in Set II have $n_l < 2n_n$. Therefore, although the target rates in Set II are smaller than those in Set I, it is probably more difficult to find a no-coding solution to achieve the smaller target rates in Set II. Actually, in the Set II cases, that is, Case II-1 to Case II-4, the results of a comparison of these methods show significant differences, which may suggest that the networks in Set II are more suitable for testing different methods. Therefore, hereafter, we will only focus on analyzing the results of Case II-1 to Case II-4.

- (ii) Table 2 also shows that, in Case II-1 to Case II-4, GlobalGA1, GlobalGA2, and GlobalGA3 clearly outperform the existing algorithms, that is, Minimal 1, Minimal 2, and GA[2]. Unlike the existing algorithms, which can hardly find the theoretically optimal solutions, particularly in complicated cases such as Case II-3 and Case II-4, all three new GlobalGAs are capable of finding the theoretically optimal solutions in all 4 cases of Set II.
- (iii) GlobalGA1 adopts exactly the same heuristic rules as GA[2] does, but the performance of GlobalGA1 is clearly much better than that of GA[2], particularly in Case II-3 and Case II-4. This may suggest that the

TABLE 3: Details of the results of the new GlobalGAs.

(Average results of 20 runs)	Case I-1	Case I-2	Case II-1	Case II-2	Case II-3	Case II-4
Final max fitness						
GlobalGA1	149.65	190.00	240.00	230.00	181.67	40.21
GlobalGA2	171.54	195.66	240.00	240.00	195.64	59.09
GlobalGA3	280.00	260.00	240.00	240.00	240.00	240.00
How many generations to achieve final max fitness						
GlobalGA1	245.50	239.60	2.35	9.40	242.40	300.00
GlobalGA2	210.75	221.00	1.05	5.80	171.90	300.00
GlobalGA3	64.40	39.10	1.00	2.20	11.45	56.70
Average minimal coding links						
GlobalGA1	1.20	0.80	0.00	0.00	0.80	6.30
GlobalGA2	1.15	0.70	0.00	0.00	0.30	5.00
GlobalGA3	0.00	0.00	0.00	0.00	0.00	0.00
Maximum minimal coding links						
GlobalGA1	4	3	0	1	3	22
GlobalGA2	3	1	0	0	2	12
GlobalGA3	0	0	0	0	0	0
Minimum actually achieved rate at sinks						
GlobalGA1	3	3	2	2	1	1
GlobalGA2	3	3	2	2	2	1
GlobalGA3	4	3	2	2	2	2

designs of the GlobalGA1 here, for example, the new NCP model, the new chromosome structure, and the associated operations, are more suitable for the NCP than the GA designs in [2].

- (iv) On average, GlobalGA2 achieves a better performance than GlobalGA1 does. Since GlobalGA2 has one more heuristic rule, that is, Rule 3, than GlobalGA1 has, it is reasonable to assume that the improvement in performance of GlobalGA2 is mainly due to Rule 3. As Rule 3 can also apply to GA[2], one may assume that the performance of GA[2] would also be improved if it employed Rule 3.
- (v) It should be noted that, according to the fitness function given by (4) to (5) with $\alpha_1 = \alpha_2 = 10$, $\alpha_3 = 1$, $\alpha_4 = 0$, $\alpha_5 = 200$, and $\alpha_6 = 0$, the theoretical maximum fitness is 240 for Case II-1 to Case II-4. Table 3 shows that GlobalGA3 always achieves this maximum fitness within 300 generations of evolution. From this table, one can see that GlobalGA3 converges much more quickly than GlobalGA1 and GlobalGA2, and it finds much better solutions than GlobalGA1 and GlobalGA2. Actually, GlobalGA3 always finds the theoretical optimal solutions. Since the only difference between GlobalGA3 and GlobalGA2 is the integration of Rules 4 and 5 into GlobalGA3, it is reasonable to conclude that it is the impact of these two additional rules that plays a significant role in improving the performance of the algorithm. It should be noted that these two rules, that is, Rules 4 and 5, are not designed only for the particular networks used in the experiments but developed without

reference to any specific network topology, making them generally applicable regardless of topology.

- (vi) The theoretical optimal solutions in all cases require no coding, whilst Rule 3 initializes some chromosomes without coding. Therefore, could Rule 3 accidentally introduce such theoretical optimal solutions into the gene pool right from the start, and then bias the GlobalGA2 and GlobalGA3 results? It should be pointed out that the no-coding solutions are not equal to the optimal solutions without coding. Actually, most no-coding solutions cannot achieve the theoretical maximum throughput. In other words, although the gene pool already includes some no-coding solutions due to Rule 3, it is very unlikely that such no-coding solutions can be guaranteed to be theoretical optimal solutions, and therefore they still need to evolve. For instance, Table 3 clearly shows that, on average, even GlobalGA3 needs to evolve tens of generations to find the theoretical optimal solutions. This implies that, most of the time, Rule 3 cannot introduce any theoretical optimal solution at all.

The above experimental results show that GlobalGA3 is the best algorithm, largely because of the introduction of Rules 4 and 5. Here we will further investigate the roles played by Rules 4 and 5. To save space, the experimental results reported here are all based on only one case, that is, Case II-4, which is the hardest case. In all previous experiments, when the focus was to improve a chromosome, GlobalGA3 applies Rules 4 and 5 no more than once. In other words, GlobalGA3 uses Rules 4 and 5 to modify no more than one

TABLE 4: Computational efficiencies of different GlobalGAs based on Case II-4.

(Ave. results of 20 exp.)	GlobalGA1	GlobalGA2	GlobalGA3 with a N_{R4R5} of									
			1	2	3	4	5	6	7	8	9	10
Final max fitness	40.21	59.09	240.00	240.00	240.00	240.00	240.00	240.00	240.00	240.00	240.00	240.00
Number of coding links	6.30	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Generations to converge	300.00	300.00	56.70	28.40	15.60	11.10	4.30	3.90	3.40	3.00	3.00	2.70
Computational time of one generation (sec.)	1.39	1.38	2.92	3.42	4.16	4.85	5.78	6.48	7.09	7.28	7.82	8.72
Total computational time (sec.)	415.70	414.57	168.31	96.78	60.87	49.91	24.91	25.36	24.18	22.00	23.45	23.83

gene of a chromosome. In the following experiments, we will allow GlobalGA3 to apply Rules 4 and 5 to modify up to N_{R4R5} genes of a chromosome, where $N_{R4R5} = 1, \dots, 10$. All other algorithm-related parameters remain the same as in previous experiments. The results are given in Table 4. From Table 4, the following observations can be made.

- (i) GlobalGA3 can always find the theoretical optimal solutions, while GlobalGA1 and GlobalGA2 often struggle to do so. This proves that Rules 4 and 5 are the cause of the advantages.
- (ii) A GlobalGA3 with a larger N_{R4R5} needs fewer generations to converge to the optimal solutions. It is reasonable to suggest that Rules 4 and 5 play a crucial role in improving the performance of GlobalGA3: applying Rules 4 and 5 for more times will lead to better performance.
- (iii) However, applying Rules 4 and 5 causes additional computational burden; therefore, the computational time consumed by a generation of GlobalGA3 is larger than those of GlobalGA1 and GlobalGA2, and such computational time goes up as N_{R4R5} increases.
- (iv) Fortunately, when we combine the computational time consumed by a generation and the generations needed to converge to the optimal solutions, it becomes clear that the total computational time consumed by GlobalGA3 to find the optimal solutions is actually smaller than those of GlobalGA1 and GlobalGA2.
- (v) Considering the influence of N_{R4R5} on the total computational time of GlobalGA3, a balance should be made to set up N_{R4R5} , because the least total computational time occurs neither with a small N_{R4R5} , nor with a large N_{R4R5} , but with a medium N_{R4R5} . In the case of Case II-4, the best value for N_{R4R5} is 8, which results in GlobalGA3 being able to find the optimal solutions at the fastest speed.

Hence it may be concluded that GlobalGA3 outperforms GlobalGA1 and GlobalGA2 in terms of not only solution quality, but also in terms of computational efficiency. This

shows that the introduction of Rules 4 and 5 is very advantageous and hence justifies the use of the permutation representation.

As is well known, a large enough field size plays a crucial role in achieving the maximum possible throughput. Equation (13) shows that, in the case of our new GAs, the search space size for a single outgoing link will grow exponentially with the field size. Therefore, the focus of the following experiments is to explore and examine the influence of field size on the performance of our new GAs. Five field sizes, that is, $N_W = 2, 4, 6, 8,$ and 10 , are used in GlobalGA1, GlobalGA2, and GlobalGA3. Here N_{R4R5} is set as 8 for GlobalGA3, as Table 4 shows it gives the best performance. Based on those networks in Set II of Table 1, some key average results are given in Table 5, from which, the following observations can be made.

- (i) The field size has a significant influence on the performance of GlobalGA1 and GlobalGA2. In the case of Case II-1, the simplest network of all, GlobalGA1 and GlobalGA2 with different field size can always find the optimal solutions, but it takes more time when a larger N_W is adopted. In the case of Case II-2 and Case II-3, GlobalGA1 and GlobalGA2 may still find the optimal solutions when N_W is small, but the solution quality reduces quickly as N_W increases. In Case II-4, the most complex network of all, both algorithms struggle and usually can only find feasible solutions, regardless of the value of N_W .
- (ii) In all test cases, in terms of either solution quality or computational time, GlobalGA3 has a very robust performance against the change of N_W . Actually, for a given network, GlobalGA3 can always find the optimal solution with similar computational time, no matter what value N_W has.
- (iii) In summary, one can see that the field size has significant influence on GlobalGA1 and GlobalGA2, which have relatively poor local-searching capability, but, thanks to Rules 4 and 5, no obvious influence on GlobalGA3 is observed. In other words, GlobalGA3 can perform satisfactorily well for different field sizes.

TABLE 5: The influence of field size on the performance of new GAs.

(Ave. results of 20 exp.)	N_W				
	2	4	6	8	10
GlobalGA1					
Case II-1					
Final max fitness	240.00	240.00	240.00	240.00	240.00
Total computational time (sec.)	0.48	0.81	1.06	3.44	5.56
Case II-2					
Final max fitness	240.00	240.00	123.33	115.68	90.61
Total computational time (sec.)	2.40	59.26	88.65	93.46	89.27
Case II-3					
Final max fitness	181.67	54.09	50.05	49.72	52.18
Total computational time (sec.)	158.85	200.61	201.12	200.25	200.42
Case II-4					
Final max fitness	40.21	45.33	42.95	39.76	41.03
Total computational time (sec.)	415.70	420.58	405.17	411.54	425.67
GlobalGA2					
Case II-1					
Final max fitness	240.00	240.00	240.00	240.00	240.00
Total computational time (sec.)	0.47	0.47	0.50	0.48	0.47
Case II-2					
Final max fitness	240.00	240.00	240.00	240.00	240.00
Total computational time (sec.)	2.76	6.88	7.85	6.55	8.43
Case II-3					
Final max fitness	195.64	142.86	45.96	47.45	46.11
Total computational time (sec.)	127.24	201.60	199.53	200.97	204.34
Case II-4					
Final max fitness	59.09	40.16	39.50	41.88	42.02
Total computational time (sec.)	414.57	429.06	418.63	424.41	419.24
GlobalGA3					
Case II-1					
Final max fitness	240.00	240.00	240.00	240.00	240.00
Total computational time (sec.)	2.12	2.05	1.97	2.16	2.01
Case II-2					
Final max fitness	240.00	240.00	240.00	240.00	240.00
Total computational time (sec.)	3.37	4.15	3.68	3.44	3.51
Case II-3					
Final max fitness	240.00	240.00	240.00	240.00	240.00
Total computational time (sec.)	5.36	5.72	6.44	5.51	7.70
Case II-4					
Final max fitness	240.00	240.00	240.00	240.00	240.00
Total computational time (sec.)	22.00	19.24	21.27	24.84	19.13

Based on the test cases for GlobalGA3 in Table 5, where N_W makes no difference in the performance of GlobalGA3, one may ask: how important is field size for exact network coding? In fact, the importance of field size is mainly appreciated in random network coding, because a larger field size means a higher probability of achieving the target rate when random coding is used. However, even for a small field size, say $N_W = 2$, there could still exist a coding scheme to achieve the target rate. For instance, in a rectangular grid network where the source sends out two signals using the random coding scheme, the probability that a node located at grid position (x, y) relative to the source can decode both signals is at least $(1 - 1/N_W)^{2(x+y-2)}$ [26]. This implies that, for all $N_W \geq 2$, a rate of 2 is in theory always achievable at any node in a finite grid network. Unfortunately, the probability is so small under a small field size; say $N_W = 2$, that random coding can hardly determine a correct coding scheme. By employing a powerful method of searching, such as GlobalGA3 proposed in this paper, exact network coding may still stand a good chance of finding a correct coding scheme, even when $N_W = 2$. In other words, the field size might not be as important to exact network coding as is it to random network coding. This is definitely an issue worth further investigation in future research.

5.2. *Tests on SRHCGA.* In this subsection, we will study the proposed SRHC strategy. We will firstly test the general performance of the SRHCGA. Then we will investigate the influence of some SRHC-related parameters. Like in the tests on the GlobalGA, here again we have three versions of the SRHCGA: SRHCGA1 employing Rules 1 and 2, SRHCGA2 having Rule 1 to Rule 3, and SRHCGA3 including Rule 1 to Rule 5 (Rules 4 and 5 are only applied no more than once; that is, $N_{R4R5} = 1$). In the general tests, for all three SRHCGAs, the number of steps in a spatial horizon, N_H , changes from 1 to 4, and the spatial step length N_{SL} (i.e., how many link layers is covered by a spatial step) also changes from 1 to 4. Therefore, there are 16 (N_H, N_{SL}) pairs that are tested, and the total number of link layers covered by a spatial horizon varies from 1 to 16. The three SRHCGAs in the general tests share the same other SRHC-related parameters which are properly set up and fixed. For each pair of N_H and N_{SL} , 20 tests are conducted for each SRHCGA. Some important average results and the associated values for N_H and N_{SL} are summarized in Table 6, and the relationships between N_H , N_{SL} , and the associated average fitness are plotted in Figure 14. The vertical axis in Figure 14 is the fitness axis, the first horizontal axis (the left horizontal axis) is the N_H axis, and the second horizontal axis (the right horizontal axis) is N_{SL} axis. It should be noted that, according to (8), that is, the unified objective function for the SRHC strategy, the potential maximum fitness is 240 when $\beta_1 = 20, \beta_2 = 5, \beta_3 = 5, \beta_4 = 200, \beta_5 = 0$, and $\beta_6 = 10$.

From Table 6 and Figure 14, one may have the following observations.

- (i) From the typical (N_H, N_{SL}) pairs given in Table 6, one can see that small N_H and N_{SL} can deliver the best results in all test cases, which means it is not necessary to resolve the NCP as a whole, an appropriate problem

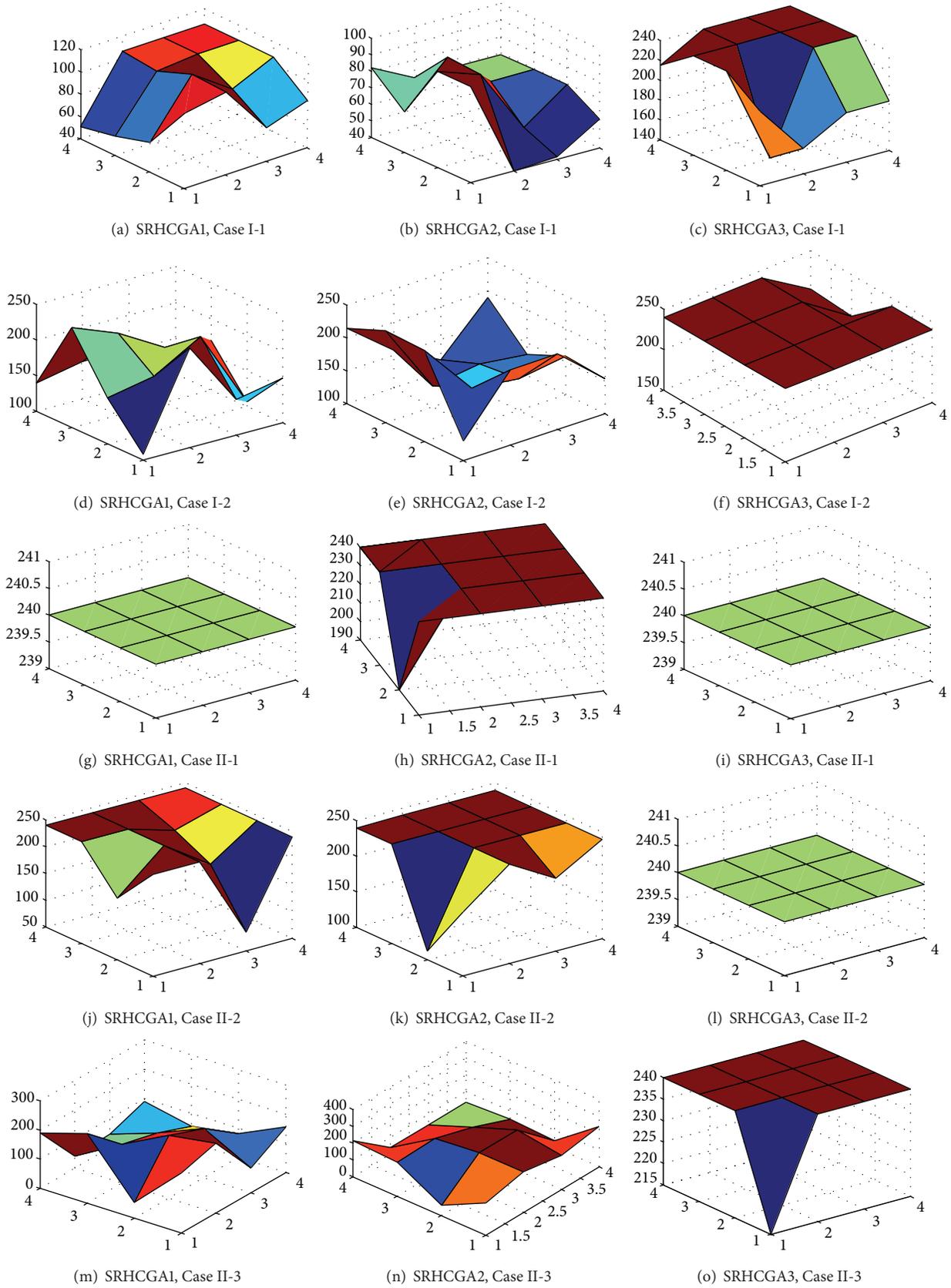


FIGURE 14: Continued.

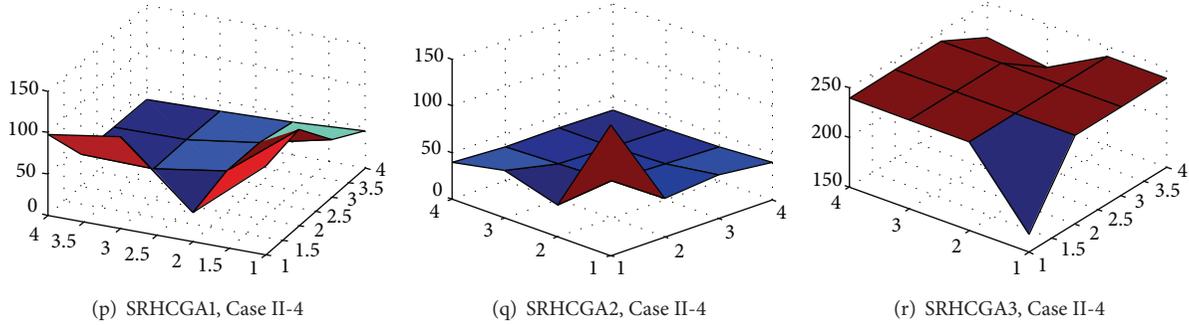


FIGURE 14: Influence of N_{SL} and N_H on fitness in general tests on SRHCGAs.

TABLE 6: Summarized results of general tests on SRHCGAs.

(Best average results of all combinations of N_{SL} and N_H)	Case I-1	Case I-2	Case II-1	Case II-2	Case II-3	Case II-4
Average minimal coding links						
SRHCGA1	0.75	0.00	0.00	0.00	0.00	1.50
SRHCGA2	0.75	0.00	0.00	0.00	0.00	0
SRHCGA3	0.00	0.00	0.00	0.00	0.00	0.00
Total computational time (sec.)						
SRHCGA1	35.61	11.94	0.00	4.72	12.89	64.59
SRHCGA2	28.25	9.83	0.00	3.35	11.17	58.93
SRHCGA3	32.86	6.04	0.00	3.82	8.43	27.38
Minimum actually achieved rate at sinks						
SRHCGA1	3.5	3	2	2	2	1.75
SRHCGA2	3.75	3	2	2	2	1.50
SRHCGA3	4	3	2	2	2	2
A typical (N_H, N_{SL}) pair that gives the best average results						
SRHCGA1	(2,2)	(3,1)	(2,2)	(2,2)	(3,1)	(1,2)
SRHCGA2	(2,1)	(3,1)	(2,2)	(2,2)	(2,2)	(1,1)
SRHCGA3	(2,1)	(2,1)	(2,2)	(2,2)	(2,2)	(2,2)
How many (N_H, N_{SL}) pairs that give the best average results						
SRHCGA1	1	2	16	12	2	1
SRHCGA2	2	2	16	13	4	1
SRHCGA3	10	14	16	16	16	13

partitioning method, such as the proposed SRHC strategy, can eventually find at least as good complete solutions to the NCP as the “resolve it as a whole” strategy can.

- (ii) Actually, when comparing the average minimal coding links and the minimal actually achieved rate at sinks in Tables 3 and 6, one can see that in Case I-1, Case I-2, Case II-3, and Case II-4, which have relatively larger network scale, the performances of SRHCGA1 and SRHCGA2 with small N_H and N_{SL} are obviously better than those of GlobalGA1 and GlobalGA2. As discussed in the previous tests on GlobalGAs, GlobalGA1 and GlobalGA2 have relatively poorer searching capability because they do not use Rules 4 and 5. When such an algorithm is applied to resolve a relatively larger NCP as a whole, it is difficult to find good solutions. However, when such an algorithm with poor searching capability

is integrated with the proposed SRHC strategy, its performance can be improved.

- (iii) Regarding the average minimal coding links and the minimal actually achieved rate at sinks, SRHCGA3 does not make difference when compared with GlobalGA3. This is because, thanks to Rules 4 and 5, GlobalGA3 is so powerful that it can almost always find the optimal solutions. Therefore, there is no room for SRHCGA3 to improve.
- (iv) Based on the above three bullet points, one may conclude that, for a given algorithm, (i) if the “resolve it as a whole” strategy can find the best solutions, then the SRHC strategy can also do it, and (ii) if the “resolve it as a whole” strategy struggles in finding the best solutions, the SRHC strategy may still find the best solutions, or at least find some better solutions.

- (v) When comparing the total computational times in Case II-4 consumed by SRHCGA1 to SRHCGA3 in Case II-4 and those by GlobalGA1 to GlobalGA3 with $N_{R4R5} = 1$ (see Table 4), one can see clearly that SRHCGA1 to SRHCGA3 are much more time-efficient than GlobalGA1 to GlobalGA3. This is understandable. The computational time of GAs usually soars up exponentially as the problem scale increases. The computational time consumed by an SRHCGA within a spatial horizon is therefore exponentially less than that by a GlobalGA (because the subproblem within a spatial horizon has a smaller problem scale). Even though an SRHCGA needs to experience a number of spatial horizons in order to get a complete solution, the total computational time just increases linearly and therefore is still less than that of a GlobalGA. Now it is clear that, for the NCP, the SRHC strategy is advantageous against the “resolve it as a whole” strategy in terms of both solution quality and computational time.
- (vi) Figure 14 reveals more details regarding how the SRHCGAs perform with different (N_H, N_{SL}) pairs. Basically, the largest fitness is achieved or can be achieved under a small (N_H, N_{SL}) pair in all tests cases. In particular, when an algorithm with poor searching capability, such as SRHCGA1 and SRHCGA2, is applied to a large-scale NCP, the largest fitness is always achieved under small (N_H, N_{SL}) pairs, and larger (N_H, N_{SL}) pairs usually have very small fitness (see Figures 14(a), 14(b), 14(d), 14(e), 14(m), 14(n), and 14(p)). These are in line with the observations made based on Table 6 and therefore further prove that “resolve it as a whole” strategy is not necessary and sometimes even disadvantageous.
- (vii) From Figure 14, one may notice that the smallest (N_H, N_{SL}) pair, that is, $N_H = 1$ and $N_{SL} = 1$, usually does not give the largest fitness. Even for SRHCGA3, the best algorithm of all, $N_H = 1$ and $N_{SL} = 1$ may lead to nonoptimal solutions; for example, see Figures 14(c), 14(o), and 14(r). One may also notice in Figure 14 that even SRHCGA3 fails to achieve the largest fitness sometimes when N_H and N_{SL} are both large; for example, see Figures 14(f) and 14(r). This may imply that it is crucial to find a suitable length for spatial receding horizon. A too small spatial horizon may lead to shortsighted performance, whilst a too large spatial horizon will likely make the SRHCGA strategy similar to the “resolve it as a whole” strategy. Therefore, a balance should be made when setting up the (N_H, N_{SL}) pair. According to Table 6 and Figure 14, it seems that (2,1), (2,2), and (3,1) are good (N_H, N_{SL}) pairs for the NCP. One may ask: given a really large network that has thousands of link layers, will these three (N_H, N_{SL}) pairs, which now appear very small, still be able to deliver good rather than shortsighted performance? This question may be partially answered in the following tests on

the importance of the terminal penalty term in the objective function of SRHCGA.

It is well known in the area of control engineering that the TRHC scheme may become unstable if no terminal penalty is included in the objective function. Simply speaking, a terminal penalty term is used to estimate the impact of the decisions made within the current temporal horizon on the future system behavior. Similarly, the terminal penalty term introduced for the SRHC strategy in Section 3.2 is used to estimate the impact of the decisions made within the current spatial horizon on those undecided links that are beyond the current spatial horizon. Table 7 compares the performances of SRHCGAs before and after the terminal penalty term is removed from the objective function. Only the typical (N_H, N_{SL}) pairs given in Table 6, which have delivered the best average results in the associated test cases, are used to conduct the new experiments associated with Table 7. To save space, only the results of average best fitness are given in Table 7. One can see clearly from Table 7 that, once the terminal penal is removed from the objective function, the performances of all SRHCGAs degrade dramatically in almost all test cases. Even SRHCGA3 often fails to find optimal solutions, even in the simplest test case, that is, Case II-1. This clearly verifies the importance of terminal penalty for the SRHC strategy in the NCP. The reason for the crucial role of terminal penalty was already explained in Section 3.2 (see Figure 9). From the importance of terminal penalty along with Rules 4 and 5, one may see a nature of the NCP: if the signals received by a link layer are better organized and diversified, then it is more likely that a node of the following link layer can receive more different signals. In other words, focusing on organizing and diversifying local information flow (to some extent the SRHC strategy does this job) may also lead to high-quality coding solutions for the entire network. This nature of the NCP may somehow explain why small (N_H, N_{SL}) pairs can always get the best results in all test cases in Table 6 and Figure 14. According to this nature, some small (N_H, N_{SL}) pairs, such as those revealed in Table 6 and Figure 14, might still be able to give satisfactory performance even in a network with thousands of link layers. This is because, as long as the signals on each link layer are well organized, the sink layer will receive a reasonably large number of diversified signals.

As discussed in Section 2.3, the SRHC strategy and population-based algorithms like GAs are a perfect match. One reason for this perfect match is that a GA will output a population of solutions, some of which can be then used to set up a pool for decided spatial steps, in order to avoid generating bad complete solutions caused by the uniqueness of subsolutions for decided spatial steps. In the experiments given as follows, we will test whether or not such a pool for decided spatial steps is useful. In all previous SRHCGA related experiments, the pool size was set as 10% of a GA population. Here another three pool sizes are used: 1 candidate combination only, 5% of a GA population, and 20% of a GA population. The results on average best fitness are listed in Table 8, from which one can see clearly that, in general, a smaller pool size will lead to a poorer performance

TABLE 7: The importance of terminal penalty in objective function for SRHCGAs.

(Based on the typical (N_H, N_{SL}) pairs given in Table 6)	Case I-1	Case I-2	Case II-1	Case II-2	Case II-3	Case II-4
SRHCGA1						
With terminal penalty	115.10	240.00	240.00	240.00	240.00	116.94
Without terminal penalty	69.47	165.77	220.35	203.71	104.68	32.07
SRHCGA2						
With terminal penalty	98.17	240.00	240.00	240.00	240.00	140.07
Without terminal penalty	47.29	144.75	228.60	185.95	123.66	40.51
SRHCGA3						
With terminal penalty	240.00	240.00	240.00	240.00	240.00	240.00
Without terminal penalty	159.55	172.81	231.92	222.90	195.85	147.62

TABLE 8: The influence of pool size on performance of SRHCGAs.

(Based on the typical (N_H, N_{SL}) pairs given in Table 6)		Pool size	Case I-1	Case I-2	Case II-1	Case II-2	Case II-3	Case II-4
SRHCGA1	1		40.53	116.38	217.44	179.36	122.97	43.47
	5% a GA population		110.47	214.61	240.00	240.00	221.85	122.73
	10% a GA population		115.10	240.00	240.00	240.00	240.00	116.94
	20% a GA population		101.26	225.36	240.00	233.85	217.81	98.48
SRHCGA2	1		49.38	100.94	225.63	194.95	151.38	51.55
	5% a GA population		76.86	230.44	240.00	236.04	213.25	118.87
	10% a GA population		98.17	240.00	240.00	240.00	240.00	140.07
	20% a GA population		108.47	226.75	240.00	240.00	240.00	133.79
SRHCGA3	1		167.53	182.44	232.35	235.11	194.69	151.36
	5% a GA population		233.85	240.00	240.00	240.00	240.00	229.74
	10% a GA population		240.00	240.00	240.00	240.00	240.00	240.00
	20% a GA population		240.00	240.00	240.00	240.00	240.00	240.00

for all SRHCGAs. As analyzed in Section 2.3, a smaller pool size means less flexibility in changing the subsolutions for decided links, so, an algorithm is more likely to be trapped to locally good solutions. If the pool only has one candidate, then it makes no difference in terms of flexibility for decided links when the SRHC strategy uses either a population-based algorithm or a deterministic algorithm, and the resulting performance is very poor (even SRHCGA3 often fails to find optimal solutions, even in the simplest test case, i.e., Case II-1). Table 8 also reveals that a too large pool size is not necessary, as it will not improve the performance further or significantly. Actually, a too large pool size may unnecessarily increase the complexity of overall search space, and consequently the performance of an algorithm with poor searching capability will degrade (see SRHCGA1 and SRHCGA2 in Table 8, e.g.). In the NCP experiments, a pool size that is 10% of a GA population seems able to give reasonably good performance for all SRHCGAs. Now one may conclude that a subsolution pool for decided spatial steps plays a crucial role for the SRHC strategy to achieve good performance. Therefore, a deterministic algorithm, which only outputs a single solution, is not suitable for conducting the optimization within a spatial receding horizon. Instead,

only a population-based algorithm like GA can take the full advantage of the SRHC strategy.

6. Conclusions

This paper attempts to develop an effective genetic algorithm (GA) for the network coding problem (NCP), where network coding resources such as coding nodes and links need to be minimized. The contributions of this reported work include the following. (i) A new mathematical formulation of the NCP is developed, which aims not only to minimize network coding resources, but also to maximize the actually achieved rate at sinks. (ii) A novel permutation representation instead of widely used binary matrix is proposed, which records relative signals on links and is therefore free of feasibility problems, and which also enables the derivation of exact information flow on links and consequently makes it possible to integrate many useful problem-specific knowledge into the algorithm. (iii) Some new NCP-specific heuristic rules are reported, which can significantly improve the overall quality of chromosomes. (iv) A novel spatial receding horizon control (SRHC) strategy is invented as problem partitioning

method, which is very effective to decompose large-scale networks and is also suitable for population-based algorithms, such as GAs, and therefore makes the proposed SRHC based GA have a good scalability for the NCP. The effectiveness of these new developments is illustrated by extensive experiments. It is worth investigation to generalize the SRHC scheme, in order to develop a general problem partitioning methodology of combining SRHC with population-based algorithms to apply to various large-scale problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this Paper.

Acknowledgments

This work was supported in part by the UK EPSRC Grant EP/F033591/1 and the Seventh Framework Programme (FP7) of the European Union under Grant PEOF-GA-2011-299725.

References

- [1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.
- [2] M. Kim, C. W. Ahn, M. Medard, and M. Effros, "On minimizing network coding resources: an evolutionary approach," in *Proceedings of the Workshop on Network Coding, Theory, and Applications (NetCod '06)*, Boston, Mass, USA, April 2006.
- [3] M. B. Richey and R. G. Parker, "On multiple steiner subgraph problems," *Networks*, vol. 16, no. 4, pp. 423–438, 1986.
- [4] C. Fragouli and R. G. Parker, "Information flow decomposition for network coding," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 829–848, 2006.
- [5] M. Langberg, A. Sprintson, and J. Bruck, "The encoding complexity of network coding," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2386–2397, 2006.
- [6] K. Bhattad, N. Ratnakar, R. Koetter, and K. R. Narayanan, "Minimal network coding for multicast," in *Proceedings of the International Symposium on Information Theory (ISIT '05)*, pp. 1730–1734, Adelaide, Australia, September 2005.
- [7] M. Kim, M. Médard, V. Aggarwal et al., "Evolutionary approaches to minimizing network coding resources," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 1991–1999, Anchorage, Alaska, USA, May 2007.
- [8] M. Kim, V. Aggarwal, U. M. O'Reilly, M. Medard, and W. Kim, "Genetic representation for evolutionary minimization of network coding resources," in *Proceedings of the 4th European Workshop on the Application of Nature-Inspired Techniques to Telecommunication Networks and Other Connected Systems (EvoCOMNET '07)*, Valencia, Spain, April 2007.
- [9] M. Kim, V. Aggarwal, U.-M. O'Reilly, and M. Medard, "A doubly distributed genetic algorithm for network coding," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 1272–1279, London, UK, July 2007.
- [10] X.-B. Hu, M. S. Leeson, and E. L. Hines, "An effective genetic algorithm for network coding," *Computers and Operations Research*, vol. 39, no. 5, pp. 952–963, 2012.
- [11] D. Thierens, "Scalability problems of simple genetic algorithms," *Evolutionary computation*, vol. 7, no. 4, pp. 331–352, 1999.
- [12] E. Cantú-Paz and D. E. Goldberg, "On the scalability of parallel genetic algorithms," *Evolutionary computation*, vol. 7, no. 4, pp. 429–449, 1999.
- [13] G. Colombo and S. M. Allen, "Problem decomposition for minimum interference frequency assignment," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 3492–3499, Singapore, September 2007.
- [14] S. Tsutsui, A. Ghosh, and Y. Fujimoto, "Forking genetic algorithms: GAs with search space division schemes," *Evolutionary Computation*, vol. 5, no. 1, pp. 61–80, 1997.
- [15] D. W. Clarke, *Advances in Model-Based Predictive Control*, Oxford University Press, 1994.
- [16] J. M. Maciejowski, *Predictive Control with Constraints*, Personal Education Limited, Marlow, UK, 2002.
- [17] X.-B. Hu and E. A. di Paolo, "A ripple-spreading genetic algorithm for the aircraft sequencing problem," *Evolutionary Computation*, vol. 19, no. 1, pp. 77–106, 2011.
- [18] S. Chand, V. N. Hsu, and S. Sethi, "Forecast, solution, and rolling horizons in operations management problems: a classified bibliography," *Manufacturing and Service Operations Management*, vol. 4, no. 1, pp. 25–43, 2002.
- [19] B. De Schutter and T. Van Den Boom, "Model predictive control for max-plus-linear discrete event systems," *Automatica*, vol. 37, no. 7, pp. 1049–1056, 2001.
- [20] X.-B. Hu and W.-H. Chen, "Genetic algorithm based on receding horizon control for arrival sequencing and scheduling," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 5, pp. 633–642, 2005.
- [21] X.-B. Hu, W.-H. Chen, and E. Di Paolo, "Multi-airport capacity management: genetic algorithm with receding horizon," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 254–263, 2007.
- [22] Z.-H. Zhan, J. Zhang, Y. Li et al., "An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 399–412, 2010.
- [23] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.
- [24] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, Berlin, Germany, 2003.
- [25] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proceedings of the 41st Annual Allerton Conference on Communication, Control and Computing*, Monticello, Va, USA, 2003.
- [26] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, Yokohama, Japan, July 2003.
- [27] G. Sywerda, "Uniform crossover in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 2–9, San Francisco, Calif, USA.
- [28] G. Melançon and F. Philippe, "Generating connected acyclic digraphs uniformly at random," *Information Processing Letters*, vol. 90, no. 4, pp. 209–213, 2004.

Research Article

Cooperative Quantum-Behaved Particle Swarm Optimization with Dynamic Varying Search Areas and Lévy Flight Disturbance

Desheng Li

Anhui Science and Technology University, Fengyang, Anhui 233100, China

Correspondence should be addressed to Desheng Li; ldsyy2006@126.com

Received 7 October 2013; Accepted 2 January 2014; Published 3 March 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 Desheng Li. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper proposes a novel variant of cooperative quantum-behaved particle swarm optimization (CQPSO) algorithm with two mechanisms to reduce the search space and avoid the stagnation, called CQPSO-DVSA-LFD. One mechanism is called Dynamic Varying Search Area (DVSA), which takes charge of limiting the ranges of particles' activity into a reduced area. On the other hand, in order to escape the local optima, Lévy flights are used to generate the stochastic disturbance in the movement of particles. To test the performance of CQPSO-DVSA-LFD, numerical experiments are conducted to compare the proposed algorithm with different variants of PSO. According to the experimental results, the proposed method performs better than other variants of PSO on both benchmark test functions and the combinatorial optimization issue, that is, the job-shop scheduling problem.

1. Introduction

PSO, originally introduced by Kennedy and Eberhart [1], has become one of the most important swarm intelligence-based algorithms. The unique information diffusion and interaction mechanism of PSO enable it to solve many problems with good performance at low computational cost. Hence, in the past decades, PSO is imported to solve the problems of numerical optimization [2], combinatorial optimization [3], controller optimization [4, 5], and software design optimization [6].

Among the applications, function optimization has been often chosen to check the performance of them, because benchmark functions are not only well described in literature such as their properties, locations, and values of the optimal solutions, but also have many different versions that can rove different capabilities of optimizer [7, 8]. However, according to the state of art in the relevant research [9] and in spite of its superior performance, PSO is even not a global optimization algorithm. To overcome this defect, some randomizing techniques are employed into the design of PSO, such as chaos [10] and quantum behavior [11, 12], to accelerate the global convergence of the algorithms. In literatures [11, 12], Sun et al. proposed a quantum-behaved PSO (QPSO) algorithm, which can be guaranteed theoretically to find optimal solution in

search space. The experimental results on some widely used benchmark functions show that the QPSO works better than standard PSO and should be a promising algorithm.

Like all other intelligence algorithms [8], escaping from the local optimum and preventing premature convergences are two inevitable difficulties in implementation. Especially, as dimensionality increases, these kinds of problems become more complex and the possibility for finding global optimum sharply decreases. Nevertheless, some applications really need to probe the global optimal solutions rather than local ones, such as function optimization and clusters structure optimization. Hence, the main motivation of this research is to find a solution to make the PSO adapt to the multidimension and difficult problems, for example, NP hard ones.

This paper proposes a novel particle swarm algorithm with two different methods to reduce the search space. One is called Dynamic Varying Search Area (DVSA), which takes charge of limiting the ranges of particles' activity, and the other is cooperative strategy, which divides the candidate solution vector into small subswarms. Moreover, in order to escape the local optima, Lévy flights are used to generate the stochastic disturbance in the movement of particles.

The remainder of this article is organized as follows. Section 2 provides a brief review on the related algorithms. Section 3 proposes the CQPSO-DVSA-LFD and gives the

main flow of the algorithm. Sections 4 and 5 illustrate the inner mechanisms, Dynamic Varying Search Area (DVSA), and Lévy Flights Disturbance (LFD), respectively. Section 6 describes the experimental framework and then presents and discusses the numerical results from the trails. Finally, Section 7 offers our conclusions and future work.

2. Review on Related PSO Algorithms

2.1. PSO. PSO algorithm was first introduced by Kennedy and Eberhart [1] as a simulation of the flock's behavior but quickly evolved into one of the most powerful optimization algorithms in the computational intelligence field. The algorithm consists of a population of particles that are flown through an n -dimensional search space. The position of each particle represents a potential solution to the optimization problem and is used in determining the fitness (or performance) of a particle. In each generation of iteration, particle in swarm can be updated by the values of the best solution found by it and the one found by the whole swarm by far according to the following equation:

$$\begin{aligned} v_{id}^{t+1} &= \omega \times v_{id}^t + c_1 \times r_1 \times (P_{id}^{\text{best}} - P_{id}^t) \\ &+ c_2 \times r_2 \times (P_{gd}^{\text{best}} - P_{id}^t), \\ P_{id}^{t+1} &= P_{id}^t + v_{id}^{t+1}, \end{aligned} \quad (1)$$

where v_{id}^t denotes the particle's velocity in t generation and P_{id}^t , P_{id}^{best} , and P_{gd}^{best} are the particle's position, personal historical best position, and swarm's global best position in t generation, respectively.

2.2. CPSO. In practice, most variants of standard PSO suffer from the curse of dimensionality, which may directly reduce to the performance deterioration. Therefore, it also causes the failure of finding the global optimum of a highdimensional problem. One effective way is to decompose the large search space into smaller swarms in lower dimensional vector space, that is, to employ cooperative strategy. Based on the rationale of Cooperative Coevolutionary Genetic Algorithm (CCGA) in [13], van den Bergh and Engelbrecht introduced the Cooperative PSO that employs a kind of cooperative behavior to significantly improve the performance of the original algorithm [9]. Compared to basic single swarm PSO, both robustness and precision are improved and guaranteed. The key idea of CPSO is to divide all the n -dimension vectors into k subswarms. So the front n/k swarms are $[n/k]$ -dimensional and the $k - (n/k)$ swarms behind have $[n/k]$ -dimensional vectors. In each pass of iteration, the solution is updated based on k subswarms rather than the original one. When the particles in one subswarm complete a search along some component, their latest best position will be combined with other subswarms to generate a whole solution.

The function b shown in (2) performs exactly like this: it takes the best particle from each of the other subswarms and concatenates them, splicing in the current particle from the current subswarm j in the appropriate position. According

to this function, the composition of P_{id}^{best} , P_{gd}^{best} , and P_{cgd}^{best} can be calculated based on (3)–(5). Due to the employment of this component, the particles in each subswarm therefore update their global best position by (6)–(7), which is the result associated with minimal fitness value of their local best positions and global best positions of electoral swarm:

$$\begin{aligned} b(u, Z) &= (S_1 \cdot P_{gd}^{\text{best}}, \dots, S_{u-1} \cdot P_{gd}^{\text{best}}, \\ &Z, S_u \cdot P_{gd}^{\text{best}}, \dots, S_k \cdot P_{gd}^{\text{best}}), \quad 1 \leq u \leq k, \end{aligned} \quad (2)$$

$$\begin{aligned} &b(u, S_u \cdot P_{id}^{\text{best}}) \\ &= \text{argmin fitness}(b(u, S_u \cdot P_{id}^{\text{best}}), b(u, S_u \cdot P_{id}^t)), \quad (3) \\ &1 \leq id \leq s, \quad 1 \leq u \leq k, \end{aligned}$$

$$\begin{aligned} &b(u, S_u \cdot P_{gd}^{\text{best}}) = \text{argmin fitness}(b(u, S_u \cdot P_{id}^{\text{best}}), \\ &1 \leq id \leq s, \quad 1 \leq u \leq k, \end{aligned} \quad (4)$$

$$\begin{aligned} &b(u, S_u \cdot P_{cgd}^{\text{best}}) \\ &= \text{argmin fitness}(b(u, S_u \cdot P_{id}^{\text{best}}), b(u, S_u \cdot P_{cgd}^{\text{best}})), \\ &1 \leq id \leq s, \quad 1 \leq u \leq k, \end{aligned} \quad (5)$$

$$\begin{aligned} v_{id}^{t+1} &= \omega \times v_{id}^t + c_1 \times r_1 \times (P_{id}^{\text{best}} - P_{id}^t) + c_2 \times r_2 \\ &\times (P_{gd}^{\text{best}} - P_{id}^t) + c_2 \times r_2 \times (P_{cgd}^{\text{best}} - P_{id}^t), \end{aligned} \quad (6)$$

$$P_{id}^{t+1} = P_{id}^t + v_{id}^{t+1}. \quad (7)$$

2.3. QPSO. In literature [14], Liu et al. proposed a Quantum Particle Swarm Optimization (QPSO), which discards the velocity vector of original PSO and consequently changes the updating strategy of particles' position to make the search more simple and efficient. QPSO is the integration of quantum computing and PSO. The QPSO is based on the representation of the quantum state vector. It applies the probabilistic amplitude representation of quantum to the coding of particles, which makes one particle represent the superposition of many states and uses quantum rotation gates to realize the update operation.

The iterative equation of QPSO is very different from that of PSO. Besides, unlike PSO, QPSO needs no velocity vectors for particles and also has fewer parameters to adjust, making it easier to implement. This can be seen clearly in Figure 1.

The updated strategy of particles' position of QPSO is as follows:

$$P_{id}^{t+1} = \varphi \times P_{id}^{\text{best}} + (1 - \varphi) \times P_{gd}^{\text{best}} \pm \beta \times |C_d - P_{id}^t| \times \ln\left(\frac{1}{u}\right), \quad (8)$$

where N is population of particles; D is dimension of problem; P_{id}^t is position of particle; P_{id}^{best} is local best position; P_{gd}^{best} is global best position; $C_d = (1/N) \sum_{i=1}^N P_{id}$, $d = 1, \dots, D$, $C = C_1, \dots$; C_d is mean local best positions.

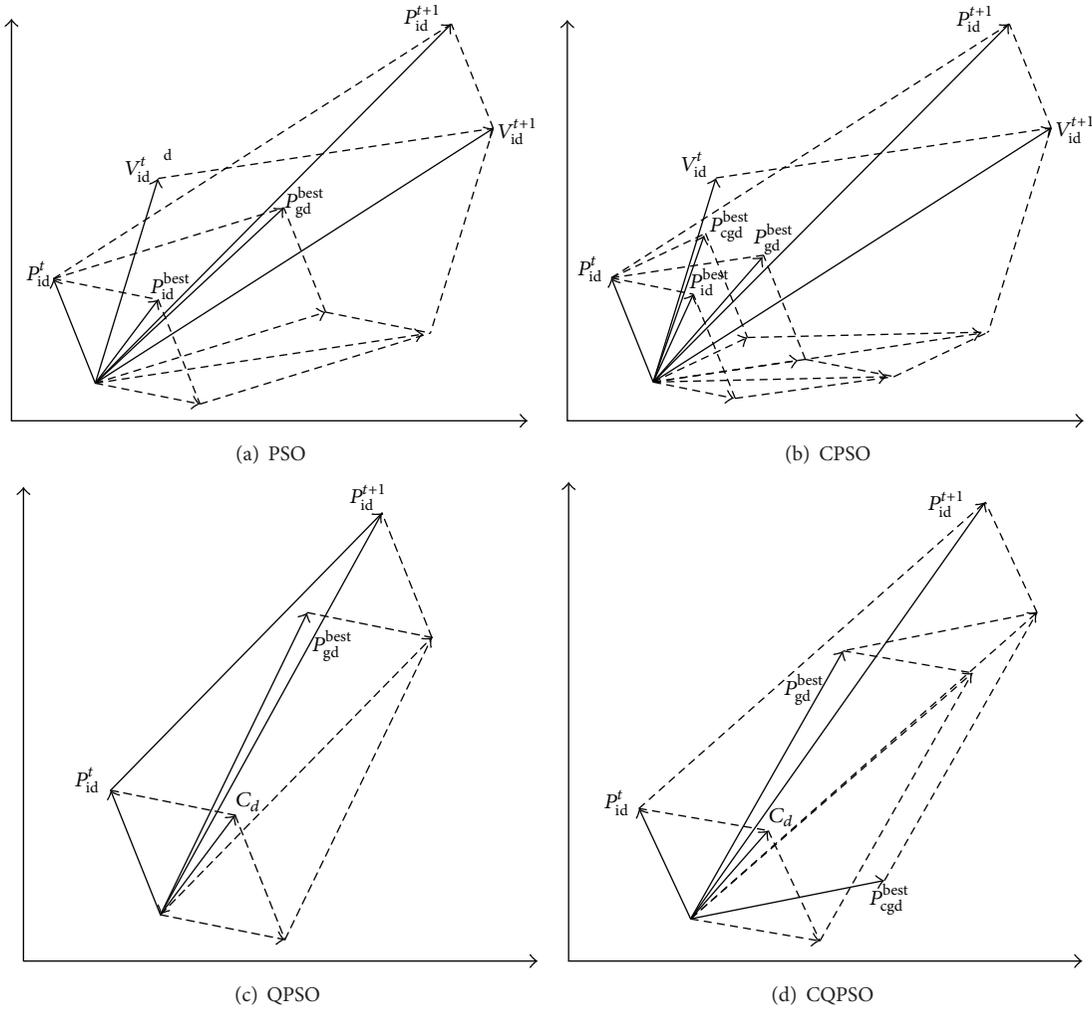


FIGURE 1: Particle movement principle of PSO, CPSO, QPSO, and CQPSO.

2.4. CQPSO. In CQPSO algorithm proposed in [15], an electoral swarm is generated by the voting of primitive subswarms and also participates in evolution of swarm, whose candidate particles come from primitive subswarms with variable votes. In reverse, the number of selected particles could also impact the voting of the primitive subswarms, such as the total number of candidates and quota of selected ones. The selected candidates could share their components with best segments of position, which are then composed into a new particle position to participate in the combining of positions. Like the treatment in our previous work, a new component of particle's position is also imported, that is, P_{cgd}^{best} , denoting the electoral best position composed by the dimensions of elected candidates.

The updating strategy of particles' position of CQPSO is as follows:

$$P_{id}^{t+1} = \varphi \times P_{id}^{best} + \psi \times P_{gd}^{best} + (1 - \varphi - \psi) \times P_{cgd}^{best} \pm \beta \times |C_d - P_{id}^t| \times \ln\left(\frac{1}{u}\right). \quad (9)$$

3. Proposed Algorithm: CQPSO-DVSA-LFD

In this paper a novel particle swarm algorithm with two different methods is proposed to reduce the search space. One is called Dynamic Varying Search Area (DVSA), which takes charge of limiting the ranges of particles' activity and the other is cooperative strategy, which divides the candidate solution vector into small subswarms. The illustration of DVSA will be introduced in Section 4.

On the other hand, theoretically, CQPSO algorithm could solve any problem by QPSO. However, due to the possibility of trapping in a case that all subswarm could not find the optimum, CQPSO could also reach the current minimum. To avoid this kind of stagnation, we employ a stochastic disturbance method, that is, Lévy flights disturbance, to generate a random movement of the stagnant subswarms. The details of Lévy flights disturbance will be introduced in Section 5.

Differently with other similar methods, we use the output parameters of Lévy flights to intervene the position change directly, which can be seen in (10) as follow, where $\text{Angle}_{L\acute{e}vy}$

```

Initiation;
Label 1: Generation primitive sub-swarms;
ForEach sub-swarm-i In sub-swarms Do
    Calculate the fitness value;
    If (run==first-time)
    Then Update the personal and global optimal position as in QPSO;
    Else Update the personal and global optimal position with;
    Calculate the best particles;
    Check the condition of DVSA, if not satisfied, and then go to the second step.
    Calculate the reduced search area.
End ForEach
Calculate the fitness value;
ForEach dimension-i In D Do
    Update the personal and global optimal position;
    Update the particles based on quantum behavior with the Lévy disturbance;
End ForEach
Calculate the electoral best position;
Test whether satisfy the condition of termination;
If (Meet terminal condition) Then ends
Else repeat from Label 1;
End If
End.

```

ALGORITHM 1: Pseudocode of CQPSO-DVSA-LFD.

and $\text{Step}_{\text{Lévy}}$ are the output parameters of Lévy flights which are randomly generated, while $\varepsilon_1, \varepsilon_2$, and ε_3 are the parametric empirical coefficient:

$$\begin{aligned}
 P_{\text{id}}^{t+1} &= \varphi \times P_{\text{id}}^{\text{best}} + \psi \times (P_{\text{gd}}^{\text{best}} + \varepsilon_1 \times \text{Angle}_{\text{Lévy}}) \\
 &+ (1 - \varphi - \psi) \times (P_{\text{cgd}}^{\text{best}} + \varepsilon_2 \times \text{Angle}_{\text{Lévy}}) \quad (10) \\
 &\pm \beta \times \left| (C_d + \varepsilon_3 \times \text{Step}_{\text{Lévy}}) - P_{\text{id}}^t \right| \times \ln\left(\frac{1}{u}\right).
 \end{aligned}$$

Based on the above introduction, we can now present the proposed CQPSO-DVSA-LFD algorithm in the following steps in Algorithm 1.

4. Dynamic Varying Search Area (DVSA)

4.1. Rationale of Dynamic Varying Search Area (DVSA). As we know, complexity of optimization problem does not only rely heavily on the objective/constraint function but also relates to its search area. Simply speaking, subjected to the same objective/constraint function, the larger the search area is, the harder it can find the solution [16]. Based on this idea, to change the search area dynamically, or say it reduces, is necessary to accelerate the processing of algorithm. On the other hand, when the search area reduced, the populations of subswarms are unnecessary as big as previous ones.

Given an optimization function:

$$\min f(x), \quad x = (x_1, x_2, \dots, x_{N_d})^T \in S \subseteq R^{N_d}, \quad (11)$$

where $S = [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_{N_d}, b_{N_d}]$, the basic rationale of Dynamic Varying Search Area (DVSA) could be illustrated as the following description. Firstly, assume

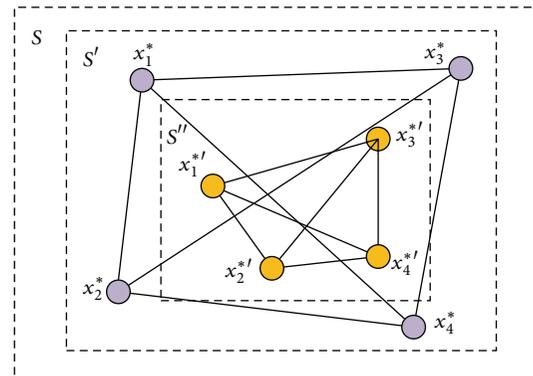


FIGURE 2: A case of DVSA.

that N_p cooperative subswarms probe is in the search space. When the minimal distances between optimal individuals of each subswarm reached a threshold, according to the maximum likelihood estimation, the hypothesis that the real optimal solution is in the area surrounded by these particles was established. Then reduce the previous search area S to S' , generate a new swarm with same subswarms on S' , and decrease the populations meanwhile. Finally, repeat the above procedures until satisfying the end condition.

Consider the vector x before the r th reduce, where the i th component x_i ranges over $[a_i^{r-1}, b_i^{r-1}]$. Then x could be expressed as $x^{r-1} \in [a_i^{r-1}, b_i^{r-1}]$. Figure 2 exemplifies the case that four cooperative subswarms reduce their search area. First, they probe the solution in S and get the best particles x_1^*, x_2^*, x_3^* , and x_4^* included in S' . So the search area becomes S' . The next time of reduction to S'' is the same procedure.

4.2. *Condition of DVSA.* In this part, we will give the condition when the DVSA occurs. Suppose there exist N_p subswarms, the best particles set found is written as

$$\begin{aligned} x_b^{r-1} &= \{x_b^{r-1,1}, x_b^{r-1,2}, \dots, x_b^{r-1,N_p}\}, \\ x_b^{r-1,p} &= (x_{b1}^{r-1,p}, x_{b2}^{r-1,p}, \dots, x_{bN_d}^{r-1,p}), \\ p &= 1, 2, \dots, N_p. \end{aligned} \tag{12}$$

Now, let us consider the distance among them:

$$\begin{aligned} D(x_b^{r-1,i}, x_b^{r-1,j}) &= \|x_b^{r-1,i}, x_b^{r-1,j}\|_2, \\ D^{r-1} &= \max_{x_b^{r-1,i}, x_b^{r-1,j} \in x_b^{r-1}} D(x_b^{r-1,i}, x_b^{r-1,j}), \end{aligned} \tag{13}$$

where $\|\cdot\|_2$ is the 2-norm on corresponding search area.

When D^{r-1} reached a small threshold, according to the maximum likelihood estimation, the hypothesis that the real optimal solution is in the area surrounded by these particles was established. So the latter search can be performed around these particles.

In light of this, we can give the condition of DVSA as shown below

$$D^{r-1} < \lambda \cdot \|a^{r-1} - b^{r-1}\|_2, \quad \lambda \in (0, 1/N_p]. \tag{14}$$

In other words, if the above equation is satisfied, then change the search area of the next generation of subswarms until the DVSA occurs again. λ can be a fixed number, but more often, it is a parameter that can be changed adaptively according to the results of evolution.

Let us consider the search area after reduction. Note that after the r th reduce, x_i ranges over $[a_i^r, b_i^r]$. Then the upper/lower bounds are defined by the following equation:

$$\begin{aligned} a_i^r &= \min \{x_{bi}^{r-1,p}\} - \xi \cdot (b_i^{r-1} - a_i^{r-1}), \\ b_i^r &= \min \{x_{bi}^{r-1,p}\} + \xi \cdot (b_i^{r-1} - a_i^{r-1}), \end{aligned} \quad \xi \in (0, 1]. \tag{15}$$

To guarantee that the new search area is not larger than the previous area, the above equation should be modified as follows:

$$\begin{aligned} a_i^r &= \begin{cases} a_i^{r-1}, & a_i^r < a_i^{r-1}, \\ a_i^r, & \text{otherwise,} \end{cases} \\ b_i^r &= \begin{cases} b_i^{r-1}, & b_i^r < b_i^{r-1}, \\ b_i^r, & \text{otherwise.} \end{cases} \end{aligned} \tag{16}$$

4.3. *Policy of Population Scale Adjustment.* The computational complexity also relies heavily on the scale of the population of the swarm/subswarm. In general, the more take time about particle evaluation is, the more computation takes place. Hence, under the permission of optimization performance, it is necessary to cut down the population of subswarms.

In this article, we will follow a traditional method called search granularity. Take the particle after the r th reduce for instance, whose i th component x_i ranges over $[a_i^r, b_i^r]$. The distance of this interval can be written as (17) which reflects the refined effort of search. If the distance among the solutions is small, we can say that search granularity is small and vice versa. From the real experience, the bigger the swarm is, the less distance exists among the particles, which also lessen the search granularity:

$$d_i^r = b_i^r - a_i^r. \tag{17}$$

Furthermore, if it is asked that the search granularity on $[a_i^r, b_i^r]$ should be $1/N_{ik}$, the population scale of subswarm can be determined by the below equation:

$$N_i^r = \left\lfloor \prod_{k=1}^{N_d} N_{ik} \cdot d_i^r \right\rfloor, \tag{18}$$

where $\lfloor \cdot \rfloor$ is the floor function. When the search area decreases, the population of the related subswarm also becomes small.

4.4. *Theoretical Analysis.* In this subsection, an analysis of the convergence of CQPSO-DVSA-LFD is provided. We discuss it from two perspectives, that is, search area and population of swarms.

Firstly, we analyze the variance of interval measure caused by two neighboring reduces. According to the policy of DVSA, it can be described as follows according to (19), (20):

$$b_i^r - a_i^r \leq b_i^{r-1} - a_i^{r-1}. \tag{19}$$

Without loss of generality, let

$$b_i^r - a_i^r = k_i^r \cdot (b_i^{r-1} - a_i^{r-1}), \quad k_i^r \in (0, 1], \tag{20}$$

then

$$\begin{aligned} b_i^r - a_i^r &= k_i^r \cdot (b_i^{r-1} - a_i^{r-1}) \\ &= k_i^r \cdot k_i^{r-1} (b_i^{r-2} - a_i^{r-2}) = \dots = K_i^r \cdot (b_i - a_i), \end{aligned} \tag{21}$$

$$K_i^r = \prod_{j=1}^r k_j^r \leq \min \{k_i^1, k_i^2, \dots, k_i^r\}.$$

From formula (21), we can see that when search area varies, the reduced area becomes the k_i^r times of origin area. So when several generations of this procedure happen, the final area could be heavily reduced with the considerable promotion of efficiency.

Secondly, in consideration of swarm populations, we can get the result from

$$\begin{aligned} N_j^r &= \left\lfloor \prod_{i=1}^{N_d} N_{ji} \cdot d_i^r \right\rfloor = \left\lfloor \prod_{j=1}^{N_d} N_{ji} \cdot (b_i^r - a_i^r) \right\rfloor \\ &= \left\lfloor \prod_{j=1}^{N_d} N_{ji} \cdot K_i^r (b_i^r - a_i^r) \right\rfloor < \left\lfloor \prod_{j=1}^{N_d} N_{ji} \cdot (b_i - a_i) \right\rfloor. \end{aligned} \tag{22}$$

The above inference shows that as the search area decreases, the related populations of swarms can also be cut down with a certain rate.

5. Lévy Flights Disturbance

The technique of random disturbance is often imported to improve the performance of PSO or QPSO. When QPSO was proposed, the Gaussian and Cauchy probability distribution disturbances have been used to avoid premature convergence. In [17], the random sequences in QPSO were generated using the absolute value of the Gaussian probability distribution with zero mean and unit variance. Based on the characteristic of QPSO, the variables of the global best and mean best positions are mutated with Cauchy distribution, and an adaptive QPSO version was proposed in [14].

In this paper, another random method, Lévy flights, is employed to do this work. Lévy flights, named after the French mathematician Paul Pierre Lévy, are Markov processes. After a large number of steps, the distance from the origin of the random walk tends to a stable distribution. Lévy flights, which can be characterized by an inverse square distribution of step length, may optimize the random search process when targets are scarce and at scarcity of resources. In contrast, Brownian motion is usually suitable for the case when there is a need to locate abundant prey or targets. These traits of two random walks inspired us to improve our swarm intelligence optimization, where Lévy flights can improve the ability of "exploration" while Brownian motion benefits the "exploitation."

Mathematically, Lévy flights are a kind of random walk whose step lengths meet a heavy-tailed Lévy alpha-stable distribution, often in terms of a power-law formula, $L(s) \sim |s|^{-1-\beta}$, where $0 < \beta \leq 2$ is an index. A typical version of Lévy distribution can be defined as [18]

$$L(s, \gamma, \mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp\left[-\frac{\gamma}{2(s-\mu)}\right] \frac{1}{(s-\mu)^{3/2}}, & 0 < \mu < s < \infty; \\ 0, & s \leq 0. \end{cases} \quad (23)$$

At the change of β , this can evolve into one of Lévy distributions, normal distributions; and Cauchy distributions.

Taking the 2D-Lévy flights for instance, the steps follow a Lévy distribution as in Figure 3(b), while the directions of its movements meet a uniform distribution as in Figure 3(a). As shown in Figure 3(c), an instance of the trajectory of 500 steps of random walks is obeying a Lévy distribution. Note that the Lévy flights are often more efficient in exploring unknown and large-scale search space than Brownian walks. One reason for this argument is that the variance of Lévy flights $\delta^2(t) \sim t^{3-\beta}$, $1 \leq \beta \leq 2$ increases faster than that of Brownian random walks, that is, $\delta^2(t) \sim t$. Also, compared to Gaussian distribution, Lévy distribution is advantageous since the probability of returning to a previously visited site is smaller than that for a Gaussian distribution, irrespective of the value of μ chosen.

From the update strategy of CQPSO-DVSA-LFD, we can draw a conclusion that all particles in CQPSO-DVSA-LFD will converge to a common point, leaving the diversity of the population extremely low and particles stagnated without further search before the iterations are over. To overcome the problem, we exert a disturbance generated by Lévy flights on the mean best position, global best position, and electoral best position when the swarm is evolving as shown in the following equation (24). To the local attractor, the hop steps in Lévy flights promise the random traversal in the search space. However, to the global and electoral best locations, they only need a slightly disturbance; that is, the angles meet a uniform distribution, to exploit the particles nearby

$$\begin{aligned} C'_d &= C_d + \varepsilon_3 \times \text{Step}_{\text{Lévy}}, \\ P_{\text{gd}}^{\text{best}'} &= P_{\text{gd}}^{\text{best}} + \varepsilon_1 \times \text{Angle}_{\text{Lévy}}, \\ P_{\text{cgd}}^{\text{best}'} &= P_{\text{cgd}}^{\text{best}} + \varepsilon_2 \times \text{Angle}_{\text{Lévy}}, \end{aligned} \quad (24)$$

where ε_1 , ε_2 , and ε_3 are a prespecified parameter, $\text{Step}_{\text{Lévy}}$ is a number in a sequence by Lévy flights, angle is the angles of directions in Lévy flights.

6. Experimental Studies

6.1. Experiments on Continuous Optimization Benchmarks. To study the search behavior and its performance of CQPSO-DVSA-LFD with other versions of PSO, such as plain PSO, CPSO, and CQPSO, some typical benchmark functions of continuous optimization are selected as the examples [19, 20].

Rastrigin's function is frequently used as a test function to test the performance of optimization algorithms. Based on Sphere function, it uses cosine function to generate lots of local optimal points. It is a complex multimodal function, and optimization falls into the local optimum easily. Griewank function is a spin, inseparable, variable-dimension, multi-mode function as shown in Figure 4. At the increase of its dimension, the scope of local optimum gets narrower so that searching global optimum becomes easy relatively. Therefore, for Griewank function, it is harder to get solution in low dimension than in high dimension. Michalewicz function is a multimodal function with parameter m which changes the steepness of valleys. The Lévy number 8 function has one global minimum and, approximately, 125 local minima.

Computational results of variants of PSO used in the paper are qualitatively ranked in Table 1. From it, we can clearly get that the proposed CQPSO-DVSA-LFD algorithm performed greatly better than the plain PSO and QPSO. Also, compared to the basic Cooperative PSO (CPSO), the convergence property has been enhanced by the proposed techniques in the paper.

In Figure 5, the black cycles denote the distribution of particles of 2-d Griewank function in QPSO under DVSA and LFD, while the red ones express that of CQPSO-DVSA-LFD with only two cooperative subswarms. It can be clearly seen that in CQPSO-DVSA-LFD, the search area in each generation of iteration is reduced dynamically into the

TABLE 1: Results of functions optimization in benchmark.

Functions	PSO			QPSO			CQPSO			CQPSO-DVSA-LFD		
	Minimum	Maximum	Average	Minimum	Maximum	Average	Minimum	Maximum	Average	Minimum	Maximum	Average
Rastrigin's	6.12E-06	9.00E+00	4.67E+00	3.01E+00	8.90E+01	4.91E+01	2.50E+01	1.25E+02	7.52E+01	2.50E+01	2.50E+01	2.50E+01
Griewank	9.87E-03	1.50E-01	5.04E-02	9.12E-06	8.68E-02	3.86E-04	2.58E-06	4.95E-02	1.56E-02	0	8.04E-10	6.83E-11
Michalewicz	-9.284715	-7.846484	-8.387755	-9.375576	-8.195449	-9.006959	-9.613477	-8.394369	-9.237160	-9.660151	-9.660151	-9.660151
Lévy	0.6663	4.6048	2.3496	0.1019	11.5187	2.2827	3.27E-05	2.94E-04	1.48E-04	2.01E-05	6.37E-04	2.38E-04

The bold values denote the stable approximate optimal solution compared to other algorithms.

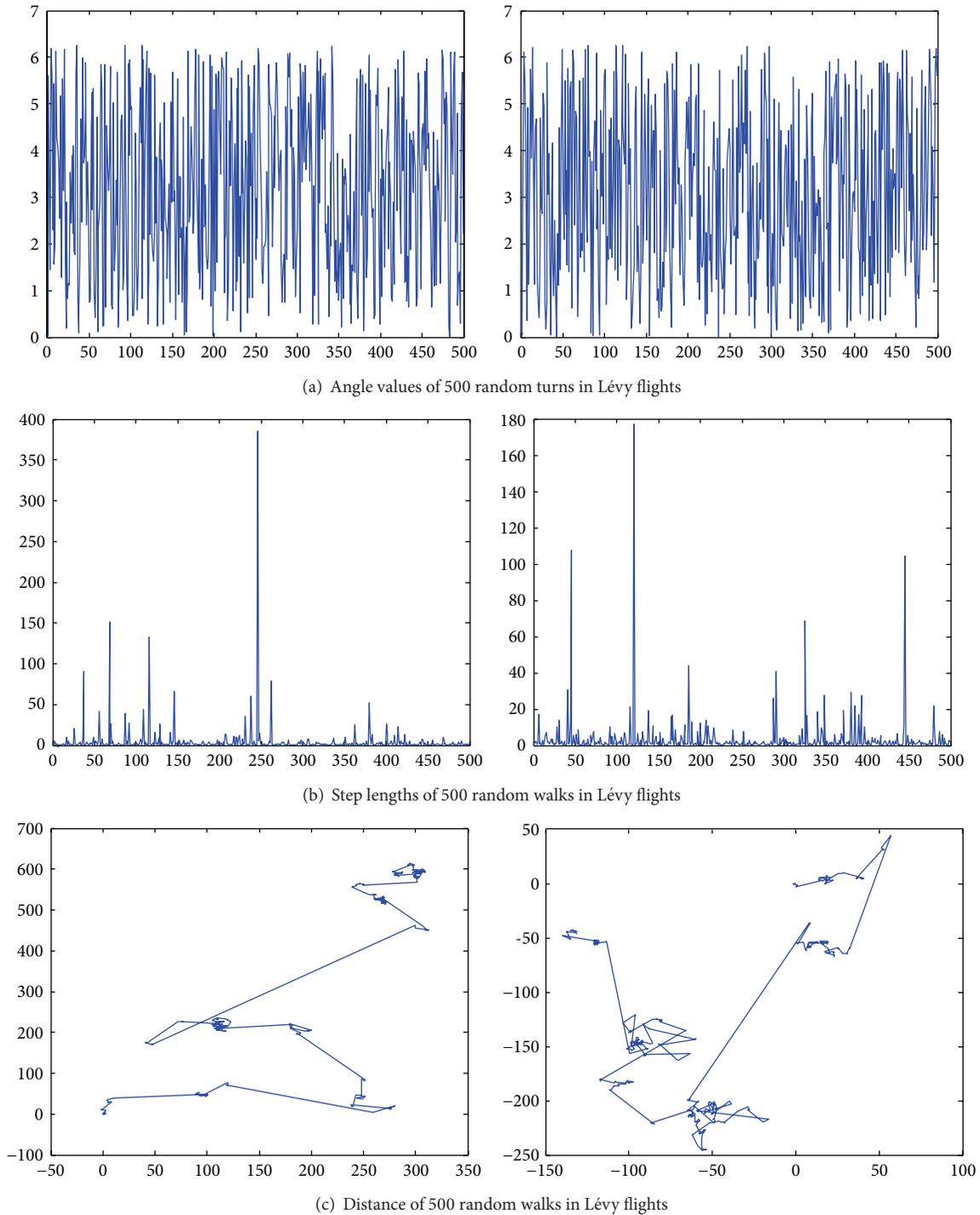


FIGURE 3: 2D Lévy flights in 500 steps.

potential rectangles along two red lines on horizontal/vertical directions. In addition, we can also find that the populations of the latter generations has been reduced obviously, which means the lower computational complexity meanwhile.

Moreover, the convergence ability is also investigated in our experiment. Figure 3 illustrates the typical convergence of PSO, CPSO, QPSO, CQPSO, and CQPSO-DVSA-LFD

on the benchmark Michalewicz function. From the figure, it can be seen that the varying curves of objective values using the CQPSO-DVSA-LFD descend much faster than that when using plain PSO and QPSO. In addition, the fitness values descent to lower level by using CQPSO-DVSA-LFD than CPSO due to the different mechanisms of simulated annealing and DVSA.

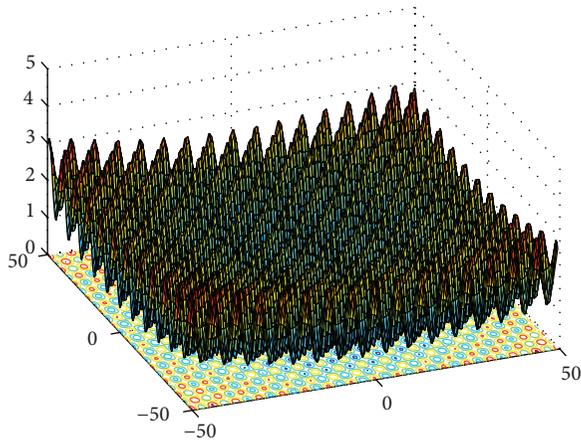


FIGURE 4: Surface landscape of Griewank function.

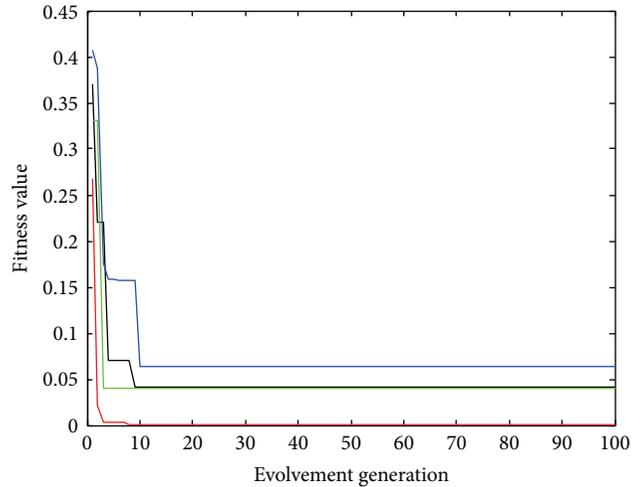


FIGURE 6: Evolution curves of Griewank function.

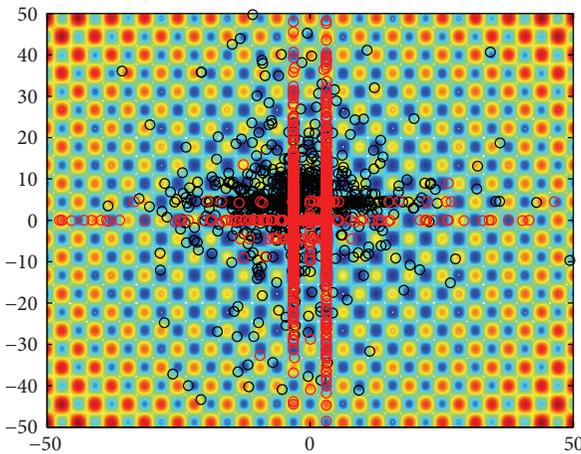


FIGURE 5: Particles in QPSO and CQPSO-DVSA-LFD.

From Figure 6, the results of the experiments indicated that the proposed CQPSO-DVSA-LFD can lead to more efficiency and stability than plain PSO, QPSO, CPSO, and CQPSO.

6.2. Experiments on Combinatorial Optimization Problem.

For the combinatorial optimization problem, we choose job-shop scheduling problem (JSSP) to test the performance of our algorithm. Job-shop scheduling problem (JSSP) is well known that it is in the family of NP-hard and NP-complete and has proven to be an impossible task for human schedulers. In the job-shop scheduling problem, finite jobs are to be processed by fix-numbered machines. Each job consists of a predetermined sequence of task operations, each of which needs to be processed without preemption for a given period of time on a given machine. But, tasks of the same job could not be processed concurrently and each job must be on each machine exactly once. Moreover, each operation cannot be commenced until the processing is completed, if the precedent operation is still being processed. A schedule is an assignment of operations to time slots on a machine. The makespan is the maximum completion time of

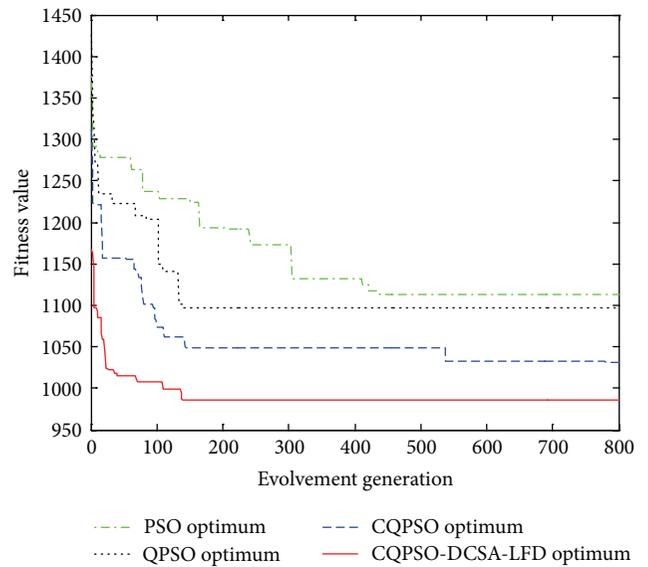


FIGURE 7: Evolution curves of JSSP.

the jobs and the objective of the JSSP is to find a schedule that minimizes the makespan.

The experiments on JSSP are performed on 6 * 6, 10 * 10, and 20 * 5 instances, respectively, and its convergence efficiency and GANT are shown in Figures 7 and 8. We can see that the convergence rate of CQPSO-DVSA-LFD is clearly faster than other PSO algorithms from our simulation solution.

7. Conclusions and Future Work

In this paper, we proposed a CQPSO-DVSA-LFD algorithm which is a combination of QPSO, cooperative mechanisms which are used to find better particles in shorter time, and

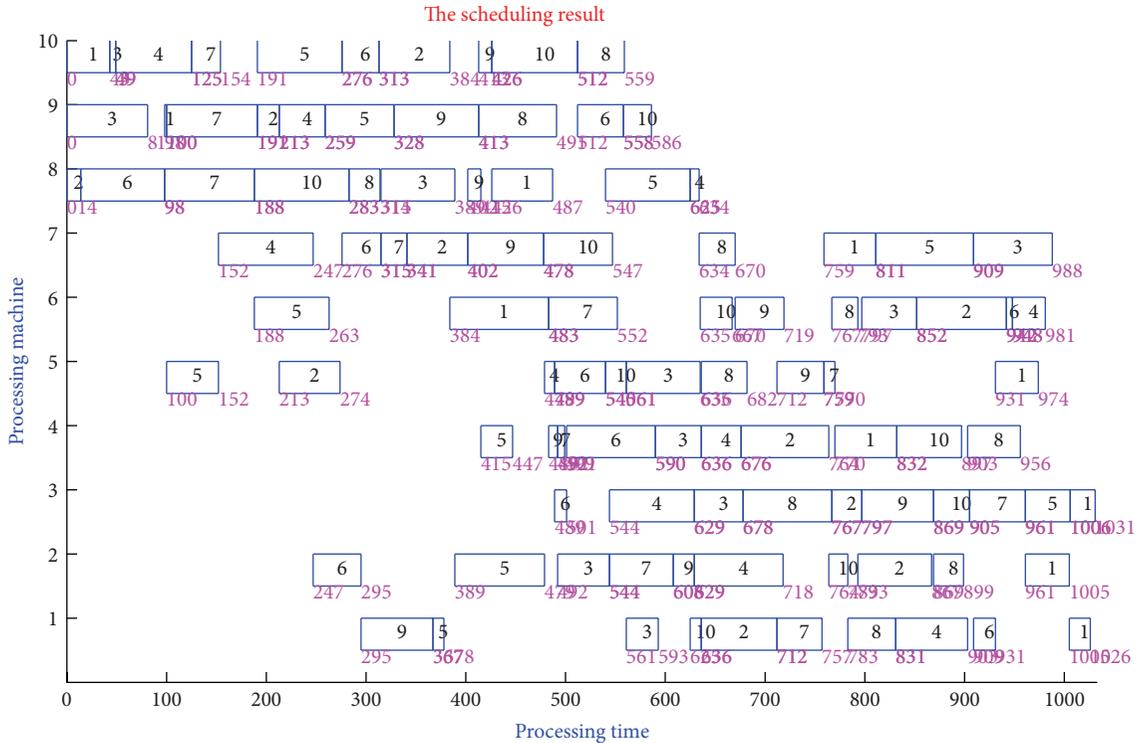


FIGURE 8: GANTT graph of 6 * 6 JSSP generated by CQPSO-DVSA-LFD.

two ways to reduce the search space. One is called Dynamic Varying Search Area (DVSA), which takes charge of limiting the ranges of particles' activity; the other is cooperative strategy, which divides the candidate solution vector into small subswarms. Moreover, to help escape from local optima, a disturbance generated by Lévy flights is embedded as a hybrid strategy. Computational results and comparisons on both continuous optimization benchmarks and JSSP problem show that it outperforms other related algorithms.

Future research may include a further investigation of the algorithm to solve other problems. It is also worthwhile to tune Lévy flights disturbance approaches and analyse the performance about the improvement by this random disturbance. The parameter setting optimally and adjustment of algorithm termination conditions are also one of our focuses.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The research was supported by the Natural Science Foundation of Anhui Province (no. 1308085QF103), Natural Science Foundation of Educational Government of Anhui Province (no. KJ2013B073), Science, and Technology Plan Project of Chuzhou City (no. 201236), Talent Introduction Special Fund of Anhui Science and Technology University (no.

ZRC2011304). The author sincerely thanks the anonymous reviewers for their constructive comments and helpful suggestions.

References

- [1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, December 1995.
- [2] Z. Cui and X. Cai, "Integral particle swarm optimization with dispersed accelerator information," *Fundamenta Informaticae*, vol. 95, no. 4, pp. 427–447, 2009.
- [3] I. Budinská, T. Kasanický, and J. Zelenka, "Production planning and scheduling by means of artificial immune systems and particle swarm optimisation algorithms," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 237–248, 2012.
- [4] Z. Cui, X. Cai, J. Zeng, and Y. Yin, "PID-controlled particle swarm optimization," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 16, no. 6, pp. 585–609, 2010.
- [5] C. Priya and P. Lakshmi, "Particle swarm optimisation applied to real time control of spherical tank system," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 206–216, 2012.
- [6] H. Derrar, M. Ahmed-Nacer, and O. Boussaid, "Particle swarm optimisation for data warehouse logical design," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 249–257, 2012.
- [7] A. Lazinica, *Particle Swarm Optimization*, IN-TECH, 2009.
- [8] X. S. Yang, Z. H. Cui, R. B. Xiao, A. H. Gandomi, and M. Karamanoglu, *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, Elsevier, Waltham, Mass, USA, 2013.

- [9] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.
- [10] A. H. Gandomi, G. J. Yun, X. S. Yang, and S. Talatahari, "Chaos-enhanced accelerated particle swarm optimization," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 2, pp. 327–340, 2013.
- [11] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, pp. 325–331, June 2004.
- [12] J. Sun, W. Xu, and B. Feng, "A global search strategy of Quantum-behaved Particle Swarm Optimization," in *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*, pp. 111–116, December 2004.
- [13] N. Keerativuttitumrong, N. Chaiyaratana, and V. Varavithya, "Multi-objective co-operative co-evolutionary genetic algorithm," in *Parallel Problem Solving from Nature-PPSN VII*, pp. 288–297, Springer, Berlin, Germany, 2002.
- [14] J. Liu, J. Sun, and W. B. Xu, "Design FIR digital filters using quantum-behaved particle swarm optimization," in *Advances in Natural Computation*, vol. 4222 of *Lecture Notes in Computer Science*, pp. 637–640, 2006.
- [15] D. Li and N. Deng, "An electoral quantum-behaved PSO with simulated annealing and Gaussian disturbance for permutation flow shop scheduling," *Journal of Information & Computational Science*, vol. 9, 2012.
- [16] A. El Dor, M. Clerc, and P. Siarry, "A multi-swarm PSO using charged particles in a partitioned search space for continuous optimization," *Computational Optimization and Applications*, vol. 11, pp. 1–25, 2011.
- [17] L. S. Coelho, "Novel Gaussian quantum-behaved particle swarm optimiser applied to electromagnetic design," *IET Science, Measurement and Technology*, vol. 1, no. 5, pp. 290–294, 2007.
- [18] X. S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, John Wiley & Sons, 2010.
- [19] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.
- [20] A. R. Hedar, "Test functions for unconstrained global optimization," http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm.

Research Article

Decomposition-Based Multiobjective Evolutionary Algorithm for Community Detection in Dynamic Social Networks

Jingjing Ma, Jie Liu, Wenping Ma, Maoguo Gong, and Licheng Jiao

Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, Xidian University, Xi'an 710071, China

Correspondence should be addressed to Maoguo Gong; gong@ieee.org

Received 21 October 2013; Accepted 22 December 2013; Published 2 March 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 Jingjing Ma et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Community structure is one of the most important properties in social networks. In dynamic networks, there are two conflicting criteria that need to be considered. One is the snapshot quality, which evaluates the quality of the community partitions at the current time step. The other is the temporal cost, which evaluates the difference between communities at different time steps. In this paper, we propose a decomposition-based multiobjective community detection algorithm to simultaneously optimize these two objectives to reveal community structure and its evolution in dynamic networks. It employs the framework of multiobjective evolutionary algorithm based on decomposition to simultaneously optimize the modularity and normalized mutual information, which quantitatively measure the quality of the community partitions and temporal cost, respectively. A local search strategy dealing with the problem-specific knowledge is incorporated to improve the effectiveness of the new algorithm. Experiments on computer-generated and real-world networks demonstrate that the proposed algorithm can not only find community structure and capture community evolution more accurately, but also be steadier than the two compared algorithms.

1. Introduction

Many real-world complex systems can be represented as complex networks. Networks could be modeled as graphs, where nodes (or vertices) represent the objects and edges (or links) represent the interactions among these objects. The area of complex networks has attracted many researchers from different fields such as physics, mathematics, biology, and sociology. Besides a number of distinctive properties such as the small-world effect, the right-skewed degree distributions, and network transitivity that many networks seem to share, community structure is another important property in complex networks [1]. Qualitatively, a community is defined as a subset of the graph nodes which densely connect with each other and sparsely connect with the rest of the networks [2, 3].

In recent years, dynamic networks have become an increasing interest due to their great potential in capturing natural and social phenomena over time [4], such as the analysis of the evolution of research communities within and across academic disciplines and the analysis of mobile

subscriber networks [5]. In the previous study, a two-step approach is widely adopted. Firstly, a static analysis is applied to the snapshots of the network at different time steps, and then community evolution is introduced afterward to interpret the change of communities over time [6]. However, data from real-world networks are ambiguous and subject to noise. Under such scenarios, if an algorithm extracts community structure for each time step independently, it often results in community structures with a high temporal variation [7].

Some recent studies have attempted to unify the processes of community extraction and evolution by using certain heuristics, such as regularizing temporal smoothness. This idea comes from a new kind of clustering concept called evolutionary clustering which has been proposed to capture the evolutionary process of clusters in temporal data [8]. This framework assumes that the structure of clusters significantly changing in a very short time is less desirable, and so it tries to smooth out each cluster over time.

Evolutionary clustering could be regarded as evolutionary multiobjective optimization (EMO). The optimization

problems with only one objective are called single-objective optimization problems, and those with more than one objective are called multiobjective optimization problems (MOPs). The main purpose of EMO is to deal with multiobjective optimization problems by evolutionary computation. It has become a hot topic in the area of evolutionary computation. By simultaneously optimizing two or more than two objectives, multiobjective optimization evolutionary algorithm (MOEA) can acquire a set of solutions considering the influence of all the objective functions. Each of those solutions cannot be said to be better than the others and corresponds to a tradeoff between those different objectives.

Community detection in dynamic networks is a problem which can naturally be formulated with two contradictory objectives and consequently be solved by an MOEA. Nevertheless, how to make the best use of MOEA to detect community structures in dynamic networks has not been fully investigated. Motivated by these, a decomposition-based MOEA for community detection in dynamic social networks (DYN-DMLS) is proposed. DYN-DMLS employs the framework of MOEA/D [9] to simultaneously optimize the modularity [2] and normalized mutual information [10], which quantitatively measures the quality of the community partitions and temporal cost, respectively. The problem-specific knowledge is incorporated in genetic operators and local search to improve the effectiveness and efficiency of our method. The uniqueness of DYN-DMLS lies in the following three aspects.

(a) It is the first time to apply the framework of MOEA/D to detect community structure of dynamic networks. MOEA/D is applied as the framework of the proposed algorithm. It optimizes N scalar subproblems simultaneously instead of a single one. It has been proved to be effective in solving MOPs by a lot of literature [9, 11–13] (<http://cswww.essex.ac.uk/staff/qzhang/webofmoead.htm>). Each subproblem is a single-objective optimization problem. To describe the advantage, we take DYN-MOGA [14] as the comparison. DYN-MOGA is significant because it may be the first trial that adopts MOEA to detect dynamic networks. Properly speaking, its optimization adopts the genetic algorithm with NSGA-II [15] dealing with multiple solutions. During the selection of solutions, to escape getting trapped into the local optimization, NSGA-II introduces the crowding distance to enhance the sparse part of the solution set, which relies on the assumption that the sparse helps to jump out of the local optimization. This is partly applicative and does not always give an expected result. MOEA/D decomposes the original problem into several subones. The scope is wide at the very beginning. When detecting dynamic networks, each calculation would involve different networks. That means that the preference to either objective may be dynamic. MOEA/D is more flexible and capable at this point. It provides an extensive possibility to approach the ideal solution before selection, rather than reforming the solution set as NSGA-II. The experimental results in Section 5 show that DYN-DMLS outperforms DYN-MOGA obviously.

(b) Problem-specific genetic operators and a local search operator are designed for community detection in dynamic

networks. Problem-specific genetic operators make use of neighborhood information to enhance the performance of crossover and mutation. The neighborhood information comes from the topology of network. Gene mutates among the neighboring alleles resulting in the fact that each offspring is a meaningful code. Then, uniform crossover would be surely safe in exchanging genes between two meaningful parents. The genetic operators avoid unnecessary search burden significantly. Moreover, label propagation [16] working as the local search operator could find a better solution effectively and efficiently. Label propagation also utilizes the topological information. Not only the connectivity but also the strength in numbers is under consideration. The quantity of neighbors with the same community ID plays a key role in adjusting the current individual's clustering. The principle meets the definition of community, in which members are closed to each other. Label propagation could produce a plenty of solutions like this in a very short time for each subproblem. Then, single-objective optimization could quickly find the best. In general, label propagation as local search strategy is effective for making use of the topology of network, while it is efficient for its perfect cooperation with MOEA/D.

(c) NMI and modularity work as objective functions perfectly, which is proven by our experiments. As mentioned in [8], evolutionary clustering decomposes the dynamic optimization problem into two objectives, snapshot quality and history cost. Snapshot quality is well studied on static networks, which is measured by famous modularity here. The most important is how to measure history cost. Based on the assumption of time smoothing (dramatic shifts between networks of two consecutive time steps are undesirable), a computable distance between two kinds of clustering is needed. Normalized mutual information (NMI), a well-known entropy measure in information theory, just works here. A number of researches on static networks have regarded NMI as the evaluation metric to measure the accuracy of results compared with ground truth. Naturally, the calculated objects of NMI could be replaced by the results of two consecutive time steps. Therefore, the dynamic of networks could be described by NMI step by step. Some discussions on the effect of NMI as time smoothing are given in the experiment part. It is clear that time smoothing presented by NMI performs significantly. In addition, the time symmetry of DYN-DMLS is also discussed. We made an analysis from the view of MOEA and provided a simple experiment to show the time symmetry which is not so good.

Experiments on computer-generated and real-world networks show the performance of our algorithm. Compared to the state-of-the-art algorithms, our algorithm has the ability to discover the community structure and its evolution more accurately.

The remainder of this paper is organized as follows: Section 2 reviews several state-of-the-art MOEAs and introduces related work of community detection in dynamic networks. Section 3 describes the proposed algorithm in detail sequentially. Section 4 presents the experimental study. Finally, concluding remarks are given in Section 5.

2. Related Work

MOEA is the base of our work. It is so basic that a brief introduction is enough. There are many famous MOEAs that have been proposed in recent years. For instance, NSGA-II [15], SPEA2 [17], MOEA/D [9], MOPSO [18], and so forth are the state-of-the-art approaches. NSGA-II uses a nondominated sorting and crowding distance to generate the nondominated solution set. SPEA2 is an improved elitist multiobjective evolutionary algorithm that employs an enhanced fitness assignment strategy compared to its predecessor SPEA. In SPEA2, new techniques for archive truncation and density-based selection are proposed. MOPOS is an extension of PSO to handle multiobjective problems. In our precious work [19–23], MOEAs have been successfully applied to handle community detection problems. MOEA/D will be introduced in Section 4.

Dynamic network is the topic mainly discussed in this paper. Dynamic networks could be analyzed in many kinds of aspects, for example, the tracing of communities, the prediction, and the evolution. In [24], a multiple objective evolutionary algorithm has been proposed by us on dynamic networks. In [25], there is a summary. Next we will introduce some significant related events in this field.

Existing methods for analyzing communities and their temporal evolution in dynamic networks can be divided into two classes. For the first class, communities and their evolutions are studied separately (usually community structures are independently extracted at each time step and then in retrospect). For the second class, communities and their evolutions are studied in a unified framework where the temporal smoothness is incorporated into analyzing communities, in order to make community structure more appropriate.

The first method to detect dynamic network structures is proposed by Hopcroft et al. [26]. They identify natural communities in each snapshot by hierarchical clustering based on similarity and match them among different snapshots. Their approach can trace the lifetime of communities, but hierarchical clustering hardly works well. In [4], Palla et al. give a thorough study on real dynamic social networks. Statistically, they conclude some principles correlated to the community size and the community life time; for example, larger communities may last longer. Their conclusions are realistic but highly rely on the static community detection algorithm CPM. The above two are typical two-stage approaches, which calculate the static networks first and then infer the correlations among them. The deviation of structure and time leads to unexpected fluctuation. Therefore, the result has its limits.

To overcome the fluctuation of two-stage approaches, the framework of evolutionary clustering was introduced by Chakrabarti et al. [8]. It is a unified framework to obtain clusters from snapshot quality and history cost simultaneously. The expected result has a high snapshot quality (it should fit the current network) and a low history cost (it should be similar to the previous one). It becomes the basic principle soon which many researchers apply and modify. Our DYN-DMLS is also in this category.

An evolutionary spectral clustering approach, proposed by Chi et al. [27], first employs the framework of temporal smoothness to cluster real blog data. They use graph cut as the metric for measuring community structure and community evolution. This method integrates temporal smoothness in the overall spectral clustering to obtain more stable and consistent clustering results, which are less sensitive to short-term noises and adaptive to long-term cluster drifts. Tang et al. [28] use a joint matrix factorization method to discover the community evolution. These methods try to maximize cluster accuracy, with respect to incoming data of the current time step, and minimize clustering drift from one time step to the successive one. In order to optimize both of these two competing objectives, it is necessary to control the degree of the user's preference towards either the snapshot cost or the temporal cost. Thus, Folino and Pizzuti proposed a multiobjective genetic algorithm to discover communities in dynamic networks by employing genetic algorithm [14]. Facetnet [29] proposed by Lin et al. relies on formulating the problem in terms of nonnegative matrix factorization. As the monotonic decrease of cost function, it promises to converge to an optimal solution with a low time complexity. But Facetnet needs a fixed community quantity, while Kim and Han's method [30] could handle the situation with variable community quantities. The concept of nanocommunity is applied to describe the dynamic of networks. The evolving, forming, and dissolving happen at the level of particles. Moreover, with cost embedded technique smoothing the network data, density-based clustering method dividing the network, and some deposition mapping local clusters, the evolution of the network would be revealed. But particle and density-based method is sensitive to the parameter used in the clustering step.

DYN-MOGA [14] proposed by Folino and Pizzuti can detect arbitrary quantities of network communities and is independent of parameters. It is an extension of multiobjective evolutionary algorithm. The two objectives naturally correspond to the snapshot quality and history cost. The optimization is a searching process for the individuals with a relatively high snapshot quality and a relatively low history cost. More recently, Lancichinetti and Fortunato propose consensus clustering [31]. It has a good performance on both the static and dynamic networks. Consensus clustering could extract stable results from a series of partitions. Therefore, it could be combined with any existing method in a self-consistent way, enhancing considerably both the stability and the accuracy of the resulting partitions [31]. For improving all kinds of classic static clustering algorithms, it is like standing on the shoulders of giants and being able to see further. It is also suitable to handle dynamic networks. The stable result generated by consensus clustering from several consecutive time steps could evaluate the network structure of a certain time window. They showed the great performance of consensus clustering on monitoring the evolution of community structure in real temporal networks.

Our DYN-DMLS has a close relationship with DYN-MOGA. It is certain that DYN-MOGA is involved in the comparison. While consensus clustering is the latest algorithm

representing the current level of this field, we also take it into the comparison.

There is a large amount of researches focusing on other properties of dynamic network besides community structure. Ahmed and Karypis tried to mine the evolution of conserved relational states from dynamic networks in their new paper [32]. Their work aims at finding the evolution path of detected relational states called evolving induced relational state (EIRS). It is similar to a set of communities with temporal relations. The entities involved may change slightly at every time step, but the main body remains stable. The changing states may lead to a better understanding of network dynamics. Kunegis et al. have another point of view on dynamic networks. In [33], they make efforts to validate that, in the spectral evolution model, the growth of large networks can be described by a change of the spectrum while the corresponding eigenvectors remain constant. Then, based on this experimental fact, they proposed two link prediction algorithms aiming at predicting where new edges will appear in a growing network. In [34], the three-way data and its evolution were paid much attention. In [35], mining temporal network models were focused on and the models are the base of the dynamic analysis. In [36], a model was presented, which can describe the collective blogging behavior on popular incidental topics.

3. Proposed Algorithm

A dynamic network (DN) can be modeled as a sequence of graph $G_t(V_t, E_t)$; that is, $DN = \{G_1, G_2, \dots, G_t, \dots\}$, where G_t is the graph representing a snapshot network at time step t , V_t is a set of objects in G_t , called nodes or vertices, and E_t is a set of links each of which connects two objects of V_t . A community in a network is a group of vertices having a high density of edges within them and a lower density of edges between groups. Let CR_t denote the set of community partitions for G_t , let C_t^i denote a community composing the community partition CR_t : that is, $CR_t = \{C_t^1, C_t^2, \dots, C_t^i, \dots, C_t^k\}$, and let k denote the number of communities in CR_t .

Assuming G_t has n nodes, the adjacency matrix W_t ($W_t \in R_+^{n \times n}$) is used to represent the link between nodes in G_t , where $w_{i,j}$ represents the element at the i th row and j th column of W_t . If there is an edge from node i to node j , $w_{i,j} = 1$; otherwise $w_{i,j} = 0$.

3.1. Framework of MOEA/D-Based Dynamic Community Detection. As mentioned above, community detection in dynamic networks is a problem which can naturally be formulated with two contradictory objectives. One objective is the community quality at the current time. The other objective is the temporal cost, which measures the distance between two community structures at consecutive time steps. In this paper, the framework of MOEA/D [9] is applied to optimize the two conflicting objectives in dynamic networks.

MOEA/D maintains a population $X = \{x_1, \dots, x_N\}$ throughout the optimization process. At each generation, the population is evolved in the following steps. First,

a unique solution $x \in X$ is assigned to each subproblem, which is called its representative. Then, N subpopulations are constructed, each for a subproblem. Second, for the i th subproblem, two parents are selected such that one is i th subproblem and the other is from the whole population. In such a way, a very wide range of child solutions could be generated due to the dissimilarity among these parent solutions. Therefore, the exploration ability of the search could be enhanced. Then crossover, mutation, and local search are applied to the parents to generate an offspring y_i to update the current solutions to its neighboring subproblems. By repeating this procedure for all subproblems, a new population $Y = \{y_1, \dots, y_N\}$ is generated. Finally, an external population (EP), which is used to store nondominated solutions found during the search, is maintained. The framework of the MOEA/D-based community detection algorithm for dynamic networks is given in Algorithm 1.

3.2. Initialization. To start MOEA/D, the decomposition of the original problem is needed. MOEA/D initially decomposes the MOP into N single-objective subproblems. The subpopulation of a subproblem associated with a weight vector λ^i is composed of the representatives of the T subproblems whose associated weight vectors are the T closest (in terms of Euclidean distance) weight vectors to λ^i , where T is the size of subpopulation. As stated in [9], the optimal solution of the i th subproblem should be close to that of the j th subproblem if λ^i is close to λ^j . Thus, one new solution, generated for each subproblem, is employed to update that of the other subproblem in its subpopulation. Here, we employ the Tchebycheff approach [37] as a decompositional technique. The j th single-objective optimization subproblem is defined as follows:

$$\begin{aligned} \text{minimize} \quad & g^j(x | \lambda^j, z^*) = \max_{1 \leq i \leq 2} \{\lambda_i^j | f_i(x) - z_i^*\} \\ \text{subject to} \quad & x \in \Omega, \end{aligned} \quad (1)$$

where $z^* = (z_1^*, z_2^*)^T$ is the reference point, that is, $z_i^* = \max\{f_i(x) | x \in \Omega\}$, for each $i = 1, 2$, $\lambda^j = (\lambda_1^j, \lambda_2^j)^T$ is the weight coefficients of subproblem, and $\sum_{1 \leq i \leq 2} \lambda_i^j = 1$ and $\lambda_i = (i - 1)/(N - 1)$. To obtain the evenly distributed Pareto-optimal solutions, it is important to choose proper weights. In our approach, we adopt uniformly distributed weight vectors. In the aggregation approaches, such as MOEA/D, the λ^i is mainly used for decomposing an MOP into single-objective subproblems by adding different weights to the objectives. This is the initialization to the objective problem.

In order to solve each subproblem, the network at the first time step should provide a kind of community structure as the initialization to time smoothing. Because there is no history information at the first time step, the network can be clustered without time smoothing. Therefore, in Step 1 of Algorithm 1, it is a single optimization on the community quality objective. The memetic community detection algorithm (Meme-Net) by ourselves [38] is adopted. Meme-Net optimizes modularity to obtain a satisfying structure. All the subproblems share this structure as their base for the

Input:

- (i) The target dynamic network $DN = \{G_1, G_2, \dots, G_1, \dots\}$.
- (ii) T : the number of time steps.
- (iii) N : number of sub problems.
- (iv) S : neighborhood size.
- (v) Uniform spread of weight vectors $(\lambda^1, 1 - \lambda^1), \dots, (\lambda^N, 1 - \lambda^N)$.
- (vi) The maximum number of generations, gen_{max} .

Output: Sequence of network partitions $\{CR_1, CR_2, \dots, CR_T\}$.

Step 1. Generate an initial clustering $CR_1 = \{C_1^1, \dots, C_1^k\}$ of the network G_1 by optimizing only the community quality objective.

Step 2. For $t = 2$ to T

- (1) **Set** $EP = \emptyset, gen = 0, IP_{gen} = \emptyset$.
- (2) **Initialization:**
 - (a) Generate an initial population $IP_0 = \{x_1, \dots, x_N\}$ randomly. For each individual x_i , each gene g_j randomly takes one of its adjacent nodes of node j .
 - (b) Compute the Euclidean distance between each pair of weight vectors. Then, get the neighborhood $B(i) = \{i_1, \dots, i_s\}$ for i th sub problem, where $\lambda^{i_1}, \lambda^{i_2}, \dots, \lambda^{i_s}$ are the S closest weight vectors to λ^i (including λ^i itself).
 - (c) Initialize the reference point $z^* = (z_1, z_2)$, where z_i is the best value found so far for objective f_i .
- (3) **Search for new solutions:**
 - (a) Assign each sub problem a unique representative $x_i \in X$.
 - (b) Construct a subpopulation $P_i = \{x_{i_1}, \dots, x_{i_r}, \dots, x_{i_s}\}$ for i th sub problem, where $x_{i_r} = x_i$.
 - (c) For each sub problem $i = 1$ to N do
 - Randomly select a solution x_j from the whole population, and two parents consist of x_i and x_j .
 - Apply the crossover and mutation operators to x_i and x_j to generate y_i for P_i .
 - Local search operator is employed in some subpopulations, which can be seen in Section 3.4 in details.
 - Update P_i, z^*, IP and EP with y_i .
- (4) **Stopping Criteria:** If $gen \geq gen_{max}$, then go to (5). Otherwise, $gen = gen + 1$, go to (3).
- (5) Choose an optimal individual from the EP for the user as shown in Section 3.5. Decode the optimal individual to generate the partition $CR_t = (C_t^1, \dots, C_t^k)$ of the graph G_t in k connected components; Return the solution $CR_t = (C_t^1, \dots, C_t^k); t = t + 1$

Step 3. Output the sequence of network partitions $(CR_1, CR_1, \dots, CR_T)$.

ALGORITHM 1: Framework of the MOEA/D-based community detection algorithm.

second time step. In the coming time steps, each subproblem would develop by itself. Note that MOEA/D has turned the original problems into several sub-ones. Here, each individual in the population would correspond to one subproblem. Assume that the original problem is decomposed into N single-objective optimization subproblems. A population of initial solutions with size N would be generated. Meme-Net completes the initialization to the object of study.

3.3. Representation. Following our previous work in [38], in this paper, we use locus-based adjacency representation (LAR) proposed in [39] and employed by [40] for multi-objective clustering. In this graph-based representation, an individual g in the population consists of n genes, in which each gene corresponds to a node in the network and n denotes the total number of nodes in this network. And each gene i can take an arbitrary allele value j in the range $\{1, 2, \dots, n\}$, which means a link between node i and j existing in the corresponding graph G . This also means nodes i and

j might be in the same community in the network. The main advantage of this representation is that the number k of clusters is automatically determined by the number of components contained in an individual and the decoding step. In addition, the decoding process can be done in a linear time, which illustrates that this encoding schema is very effective for community detection. The LAR and the corresponding encoded genotype are shown in Figure 1.

The locus-based adjacency encoding scheme has several major advantages for our task. Firstly, it is unnecessary to fix the number of communities in advance, as it is automatically determined in the decoding step, which is an important feature to address the real-world networks with no prior knowledge. Secondly, some standard crossover operators such as uniform, one-point, or two-point crossover can be employed in this representation, which effortlessly implements merging and splitting operations of communities on individuals and also maintains the remainder of the partitioning. Finally, the genetic representation contains all

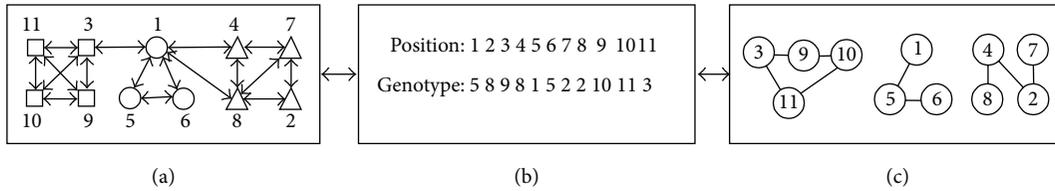


FIGURE 1: Illustration of the LAR. (a) A network modeled as a graph; (b) the LAR of one possible genotype; (c) the community structure of the genotype.

possibilities of connected subgraphs, which guarantees that a better community structure can be obtained by maximizing the modularity.

3.4. Objective Functions. As mentioned above, under the framework of temporal smoothness, we need two objective functions to quantitatively measure the quality of the communities and temporal cost, respectively. A quantitative definition, network modularity, proposed by Girvan and Newman [1], is one of the most popular quality functions to assess the goodness of the partitioning. We use modularity to measure how well the cluster structure represents the current network partition. Another quantitative definition, normalized mutual information (NMI), is a similarity measure proved to be reliable by Danon et al. [10]. We use NMI to estimate the similarity between the current community structure and the previous one.

The modularity criterion is based on the intuitive idea that a random graph does not exhibit cluster structure, while possibly there is cluster structures that is revealed by the comparison between the actual density of edges in a subgraph and the density which one would expect to have in the subgraph if the vertices of the graph were attached regardless of community structure [25]. This expected edge density depends on the chosen null model, that is, a copy of the original graph keeping some of its structural properties but without community structure [25]. Modularity can then be written as follows:

$$Q = \sum_{c=1}^k \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right], \quad (2)$$

where k is the number of clusters, l_c is the number of links inside cluster c , m is the total number of links in the network, and d_c is the total degrees of the vertices in cluster c . The first term of summand in (2) is the fraction of edges inside cluster c , and the second term represents the expected fraction of edges that would be in the random graph with the same community divisions. If the number of within-community edges is no better than that of the random, we will get $Q = 0$, while the value $Q = 1$, which is the maximum, indicates a strong community structure [1]. The higher the modularity Q is, the better the partition obtained is.

The second objective is NMI, which is a well-known entropy measure to evaluate how similar the community structure CR_t is with the previous clustering CR_{t-1} [10]. Given two partitions A and B of a network, let C be the confusion matrix whose element C_{ij} is the number of nodes of

community i of the partition A that are also in the community j of the partition B . The normalized mutual information $NMI(A, B)$ is defined as

$$NMI(A, B) = \frac{-2 \sum_{i=1}^{C_A} \sum_{j=1}^{C_B} C_{ij} \log(C_{ij}N/C_i C_j)}{\sum_{i=1}^{C_A} C_i \log(C_i/N) + \sum_{j=1}^{C_B} C_j \log(C_j/N)}, \quad (3)$$

where C_A and C_B denote the number of clusters in the partitioning A and B , respectively. C_i is the sum of the elements of C in row i , C_j is the sum of the elements of C in column j , and N is the number of nodes. If A is identical to B , $NMI(A, B) = 1$. If A and B are completely different, $NMI(A, B) = 0$. Otherwise, $NMI(A, B) \in (0, 1)$. In order to make the current communities and the previous communities as similar as possible, we need to maximize NMI.

3.5. Problem-Specific Operators

3.5.1. Uniform Crossover. In order to maintain the effective connections of the nodes in the child individual, uniform crossover is employed as the crossover operator in our method. Unlike one-point and two-point crossover, the uniform crossover enables the parent chromosomes to contribute to the gene level rather than the segment level and can generate any combination of alleles from the two parents [40]. Due to the biased initialization, if a gene i contains a value j , then the edge (i, j) exists, and each individual in the population is safe. Given two safe parents, individuals A and B , uniform crossover is performed on them to get the child individuals C and D . An example of crossover can be seen in Figure 2.

3.5.2. Mutation. For the mutation, we adopt the neighbor-based mutation [41], in which the possible values of an allele are restricted to the neighbors of gene i . The neighbor-based mutation guarantees that, in a mutated child, each vertex is linked only with one of its neighbors. This can avoid the useless exploration of the search space. In the mutation operator, we randomly select some genes and assign other randomly selected adjacent nodes to them which effectively guarantees the generation of a safe mutated child.

3.5.3. Local Search. According to many researches [16, 42, 43], the network structure itself can provide some crucial information about communities. The information is local and

Input:
 D : The subpopulation before local search
 S_D : Size of D
 K : Number of neighbors

Output:
 D' : The new subpopulation after local search

Step 1. Set $i = 1, D' = \emptyset$.
Step 2. If $i > S_D$, the algorithm terminates. Export D' as the new population.
 Otherwise, select the i th individual in D , set $k = 1$.
Step 3. If $k > K$, the search procedure stops for the i th individual,
 go to *Step 7*. Otherwise, go to *Step 4*;
Step 4. Assume the j th gene need to do local search, attain all the neighbors of node j ,
 find the label of community which most neighborhood nodes belong to.
 And then select one from these nodes to replace the j th gene
 by the corresponding value.
Step 5. Calculate the value of objective function of the new individual according to the
 corresponding single-objective sub problem. If its value is greater than
 that before local search, replace the current individual by the new one, go to *Step 7*,
 otherwise, go to *Step 6*.
Step 6. $k = k + 1$, go to *Step 3*.
Step 7. Add the current individual to $D', i = i + 1$, go to *Step 2*

ALGORITHM 2: Local search procedure.

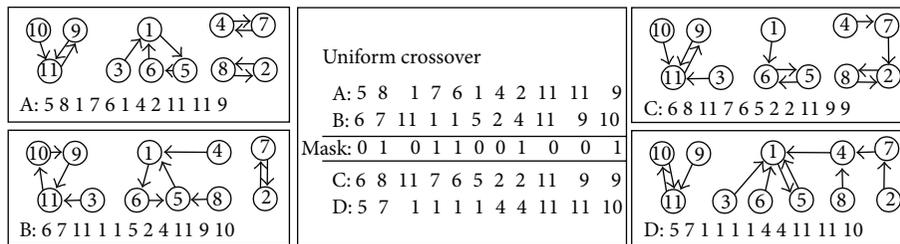


FIGURE 2: Illustration of uniform crossover. A uniform crossover on parents A and B is employed to generate the two children C and D.

the process is efficient. In [16], Raghavan et al. proposed label propagation method to detect large network communities efficiently. It is constructed on the direct analysis of the network structure without any objective function. A unique label is initially assigned to each node. Then the iteration begins with each node turning to have the same label with its most neighbors. It would not stop until no label change is going to happen, which means the community structure is stable. The advantage includes the efficiency and the independence of objective functions or similarity metrics. The disadvantage concerned most is the fluctuation of the result. Though it may not always converge to the same result, the idea of utilizing the neighborhood is meaningful. Therefore, we considered taking it as our local search strategy.

Why could label propagation be the local search strategy here? Firstly, it is localized and quick by making use of the neighborhood information. Based on a given partition, the interaction between members' labels will provide a membership adjustment which may lead to some improvement. Label propagation imitates the process of communication in the real world. It is consistent with the definition of community. Secondly, though label propagation has a low stability, it

still could contribute to the optimization process. As local search is a trail for better solutions, failure is acceptable. The effect of label propagation would be indicated within the iterations because it gets more chance to show. Thirdly, because MOEA/D works well in the global search, the input of the local search is nearly of high quality. It will enhance the stability of label propagation in some degree. The flow of local search is shown in Algorithm 2.

According to the feature of the MOEA/D, each subproblem is a single-objective optimization problem. Therefore, a better solution can be obtained by a local search procedure in optimizing corresponding single-objective problem.

In order to use the prior knowledge about relations between nodes, the local search strategy is based on the neighbor nodes. There is an obvious intuition that a node tends to be in the same community with most of its neighbors. In other words, if most of a node's neighbors are in the i th community, the node will be in the i th community with a high probability. Therefore, we should find the labels of all the neighbors of the node and record the nodes with the label whose number is the biggest among the neighbors. Then we randomly select one from these recorded nodes to replace

the original node. It will not result in merging or splitting communities when moving this node from one community to the other one.

3.6. Solution Selection. MOEA/D decomposes the original problem into several sub-ones. Each subproblem is a single-objective optimization and provides one solution at the end of each time step. Subproblems exchange information within their neighborhood. Though it is a single-objective optimization, dominant relationship is still implicated. The solution which is nondominated with both its own subpopulation and neighborhood would be reserved. These solutions form the nondominated solution set. The decomposition of the original problem supports the diversity of MOEA, while the dominant relationship with subpopulations and neighborhood pushes the solution set moving to Pareto front. The front is supposed to contain all the nondominated solutions theoretically. But in real world it is hard to realize. As the optimization of modularity and NMI are nondeterministic polynomial, we cannot identify whether the generated solutions by MOEA/D within limited generations are the optimal solution or not. MOEA/D makes efforts to approach Pareto front.

In this paper, modularity density [44] is used to select the best tradeoff solution from the dominant solution set. Modularity density is a quantitative measure for evaluating the partition of a network into communities based on the concept of average modularity degree, which has confirmed its effectiveness. Let an undirected graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = e$ edges. The adjacent matrix of the graph is A . Given a partition $\Omega = \{V_1, \dots, V_M\}$ of the graph, where V_i is the vertex set of subgraph G_i for $i = 1, \dots, M$. The modularity density (also called D values) is defined as

$$D = \sum_{i=1}^M \frac{L(V_i, V_i) - L(V_i, \bar{V}_i)}{|V_i|}, \quad (4)$$

where $L(V_i, V_i) = \sum_{m \in V_i, n \in V_i} A_{mn}$ means the internal degrees of the subgraph G_i ; $L(V_i, \bar{V}_i) = \sum_{m \in V_i, n \in \bar{V}_i} A_{mn}$, which means the external degrees of the subgraph G_i ; each summand means the ratio between the difference of the internal and external degrees of the subgraph G_i and the size of the subgraph. The larger the value of D becomes, the more accurate the partition is. Thus the solution we select as the best is the one with maximum modularity density D from the nondominated solution set.

4. Experimental Study

In this section, we evaluate the effectiveness of the proposed decomposition-based multiobjective evolutionary algorithm with local search for community detection in dynamic networks (termed as DYN-DMLS for short) on two synthetic networks and three real-world networks. The compared algorithms include DYN-MOGA [14] which is the only existing dynamic multiobjective community detection algorithm, DYN-DMLS without the local search strategy (termed as

DYN-DMEA), and consensus clustering [31] which is a different dynamic approach from MOEA. Consensus clustering could work with any static community detection algorithms. And many classic algorithms [45, 46] have been tested.

As to the performance metric, in the case that we have the ground truth for each time step, we directly adopt a similarity measure, normalized mutual information (NMI) [10, 47], to estimate the similarity between the true partitions and the detected ones. Note that the performance metric NMI is different from the objective function NMI. The former one calculates the difference between the current result and the ground truth, while the latter one calculates the difference between two consecutive time steps. The definition of NMI is introduced in detail in Section 4.4. In addition, the box plot [48] is used to illustrate the statistical distribution of the values of NMI for all compared algorithms based on 30 independent runs on most datasets. The box plot uses the median, the approximate quartiles, and the lowest and highest data points to convey the level, spread, and symmetry of the distribution of data values. In a notched box plot the notches represent a robust estimate of the uncertainty about the medians for box-to-box comparison. Symbol “+” denotes outliers.

The experiments are performed on an Intel Core2 Duo CPU machine with 1.98 GHz and 1.99 GB RAM. The parameter settings are as follows. The population size N is 100 and the number of generations is 300. The crossover operator and mutation operator are the same in the three algorithms, where crossover rate = 0.8 and mutation rate = 0.2. In MOEA/D, the neighborhood size is set to be 15. In the following experiments, the reported data are the statistical results based on 30 independent runs on each dataset.

Lancichinetti and Fortunato's consensus clustering needs a static algorithm as the base. In this study, we choose consensus clustering with label propagation method (LPM) [16] and order statistics local optimization method (OSLOM) [49] for comparison. The thought of LPM is applied in our local search strategy, while consensus clustering with OSLOM has been tested in [31], which works well in finding time evolution of clusters in real-world network. The two are more comparable than the rest.

Note that it is hard to determine which ground truth each consensus result corresponds to. This is due to the fact that several time steps are calculated together to produce a result representing the general state of the network structure during the time window. To make the comparison clear, we prepared a strategy for the consensus clustering determining the ground truth of its results. The strategy is to make the consensus algorithm running in a similar way to DYN-DMLS. In DYN-DMLS, the result of each time step is determined mainly by two aspects. One is the current network and the other is the previous one. Similarly, for consensus clustering algorithm we took $r = 2$ (which means every two consecutive time steps produce a result). And the ground truth of the result between t and $t + 1$ would be assigned to the $(t + 1)$ th time step. Then, it is easy to determine the ground truth to calculate the evaluation metric with. Different strategies may lead to different results. To emphasize $r = 2$, the two methods would be marked as consensus-LPM-2 and

consensus-OSLOM-2. In addition, the value of consensus run is set to 5, which means that the algorithm would collect 5 partitions for each time step and find the consensus among them. The threshold for pruning the consensus matrix is set to 0.5 as default.

4.1. Experiments on Synthetic Datasets. In order to evaluate the ability of our approach to successfully detect the community structures for dynamic networks, we use benchmark datasets. Benchmark networks take z as the parameter to control each node's connections with other communities. The higher the value of z becomes, the more confusing the network structures are. When z is higher than 8, usually it is considered as that there is not any community in the network. In our experiment, the benchmark parameter z varies from 5 to 8 and each z corresponds to a set of networks with 10 time steps.

Two kinds of benchmarks are involved. The first is the GN benchmark [1], which has a fixed quantity of community size. Each network has 128 nodes and 4 communities. Each community has 32 nodes. The dynamic is just the member exchanging between the original four communities. Between every two consecutive time steps each community has 10% nodes involved in the dynamic. The benchmark is marked as SYN-FIX.

The second is modified by Kim and Han [30] from the SYN-FIX, marked as SYN-VAR. It consists of a set of networks which contain the forming and dissolving of communities and the attaching and detaching of nodes. It also contains 10 timestamps and each one has 256 nodes. Its structure is changing in a more complex way. At the first time step the network has four communities. Then, at each time step, every one of the four original communities would update 16 nodes (16 nodes leave and another 16 join in). The number of communities is also changing. From start to end, the community quantity is changing as the sequence (4, 5, 6, 7, 8, 8, 7, 6, 5, 4). The creation of a new community is the combination of four sets of 8 nodes. Each set separately comes from one of the four original communities. Each new community contains 32 nodes and lasts for 5 timestamps before its nodes return to their original position. We run our algorithm on this dynamic network to show the ability of capturing the splitting and merging of communities.

The involved comparison algorithms include DYN-DMEA which is the version of DYN-DMLS without local search, DYN-MOGA, consensus clustering with OSLOM (consensus-OSLOM-2), and consensus clustering with LPM (consensus-LPM-2). As consensus clustering [31] is an excellent method for detecting complex network structures and it is a totally different framework from the evolutionary algorithm, it is worth to make a comparison between the proposed algorithm and the consensus clustering algorithm.

4.1.1. Results on SYN-FIX. In this study, we generate the datasets under four different levels by setting $z = 5, 6, 7, 8$, where the community structures gradually change from clear to fuzzy. The first time step is ignored here for a better view of the dynamic process. Each of the comparison algorithms

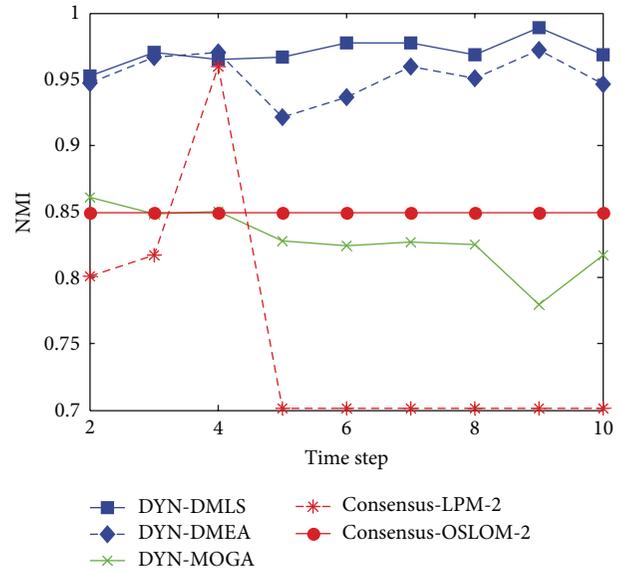


FIGURE 3: NMI results of the five algorithms on the SYN-FIX dataset with $z = 5$.

needs a static run at the first time step. Latter figures are all displayed like this.

Figure 3 shows the statistical average NMI values of time steps from 2 to 10 with $z = 5$. Among the three MOEAs, our DYN-DMLS performs best. Its results keep stable at a high level. Without local search, DYN-DMEA's results get lower slightly. Both of them are better than DYN-MOGA. However, the two based on consensus clustering have a totally different tendency from each other. Consensus-LPM-2 gets a great fluctuation, while consensus-OSLOM-2 stays absolutely stable. The reason may be that consensus clustering relies much on the static algorithm chosen. Their results are a little low. But actually each single run of LPM or OSLOM on a single time step nearly gets the same clustering with ground truth. The combination of two consecutive time steps may cause a shift leading the final result to some middle state. Here the correspondence of consensus results with ground truth is kind of rough.

Figure 4 shows the box plots of the value of NMI with respect to the ground truth at each time step over 30 independent runs with $z = 5$. Figures 4(a) and 4(b) clearly show that almost all of the values of NMI obtained by DYN-DMLS and DYN-DMEA are equal to 1, and DYN-DMLS is more stable than the DYN-DMEA. However, the values obtained by DYN-MOGA are not stable enough compared to the other two algorithms, as is shown in Figure 4(c). Therefore, DYN-DMLS is the most stable in the three algorithms. For consensus clustering, the result is not satisfying. The curve of consensus-OSLOM-2 is a straight line showing that the single run on each time step has the same result. Actually, OSLOM could detect static structures just the same as ground truth, but the consensus results turn to have a fixed deviation from ground truth. Consensus-LPM-2 performs worse.

Figure 5 shows the statistical average value of NMI when $z = 6$. It is similar to that when $z = 5$. The average value

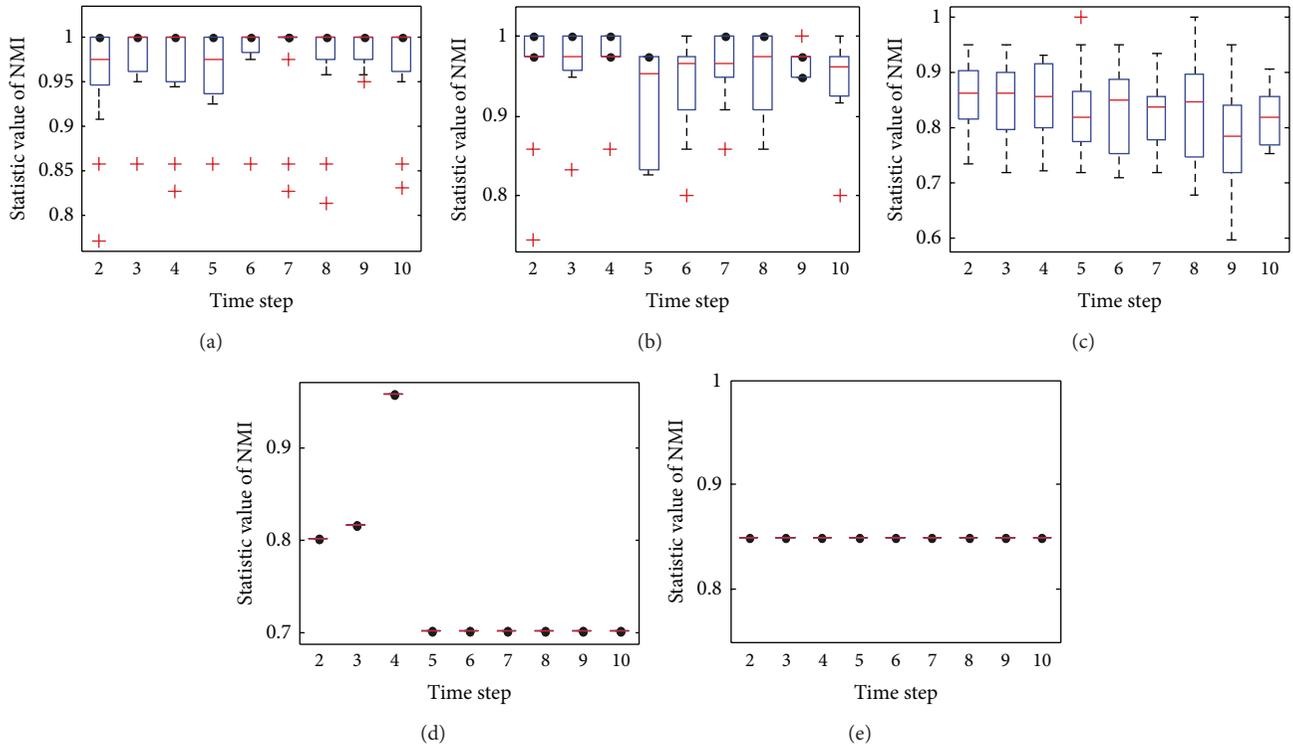


FIGURE 4: The box plots to illustrate the distribution of NMI at each time step when $z = 5$. (a) The box plot for DYN-DMLS; (b) the box plot for DYN-DMEA; (c) the box plot for DYN-MOGA; (d) the box plot for consensus with LPM; (e) the box plot for consensus with OSLOM.

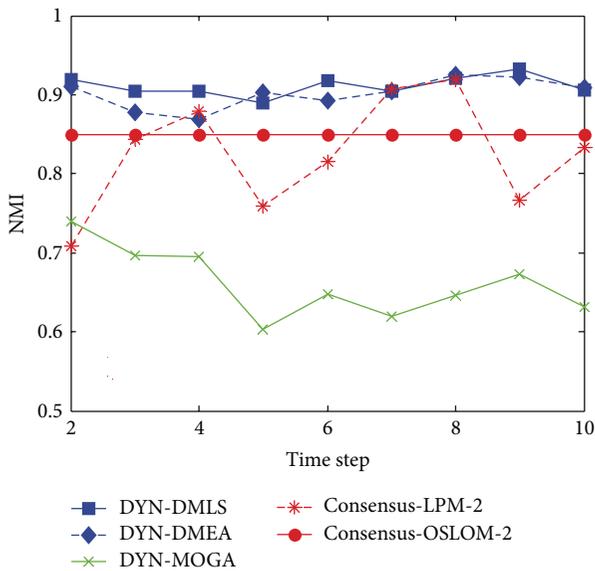


FIGURE 5: NMI results of the five algorithms on the SYN-FIX dataset with $z = 6$.

of NMI obtained by DYN-DMLS and DYN-DMEA is close to 1, while DYN-DMLS outperforms DYN-DMEA. However, it is obvious that DYN-MOGA cannot find a more accurate community structure than the first two algorithms. Results of all three MOEAs are lower than those when $z = 5$.

Consensus-LPM-2 is still of high variation and consensus-OSLOM-2 is also of high stability. Their results are lower than DYN-DMLS.

Figure 6 shows the box plots to illustrate the distribution of the value of NMI, when $z = 6$. It can be seen clearly from Figure 6 that all the average values of NMI obtained by DYN-DMLS are above 0.9 over 30 independent runs while those obtained by DYN-DMEA range around 0.9 and those obtained by DYN-MOGA range from 0.6 to 0.75. Consensus clustering shows its stability again. The repeated run seems to have no meaning for it. Consensus clustering with OSLOM is better than that with LPM.

Figure 7 shows the statistical average value of NMI with $z = 7$. The network is so confused that the result turns lower and lower. It can be seen clearly that the average value of NMI obtained by DYN-DMLS is around 0.9. Without local search, DYN-DMLS just reaches 0.7. However, results obtained by DYN-MOGA are less than 0.7. Therefore, DYN-DMLS can find the most accurate community partition in the three algorithms. For consensus clustering, the degree of results lowering with the raising of z is little. Consensus-OSLOM-2 narrows the gap numerically, while consensus-LPM-2 still varies heavily due to the fact that LPM produces different structures in comparison with OSLOM.

Figure 8 shows the box plots to illustrate the distribution of the value of NMI, when $z = 7$, where the community structures become fuzzy and evolve relatively unstable over time. It can be seen clearly from Figure 8 that the majority of NMI values obtained by DYN-DMLS are above 0.8, and those

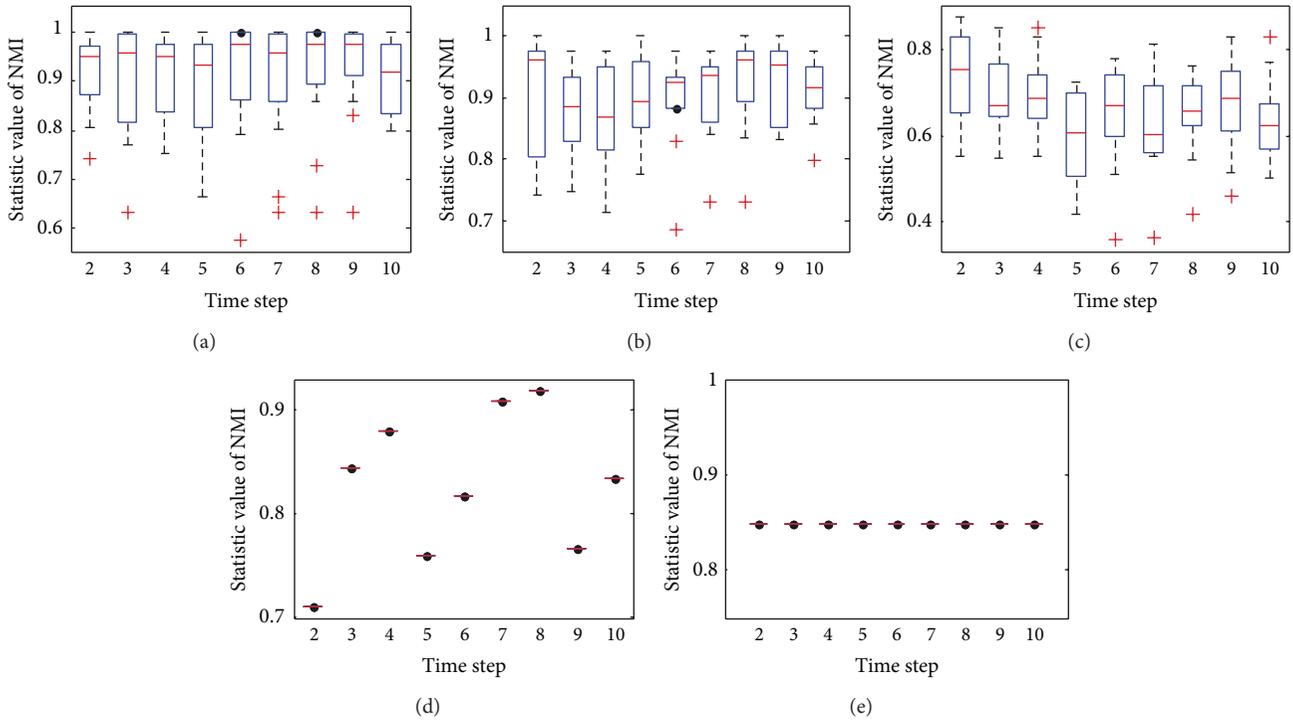


FIGURE 6: The box plots to illustrate the distribution of NMI at each time step when $z = 6$. (a) The box plot for DYN-DMLS; (b) the box plot for DYN-DMEA; (c) the box plot for DYN-MOGA; (d) the box plot for consensus with LPM; (e) the box plot for consensus with OSLOM.

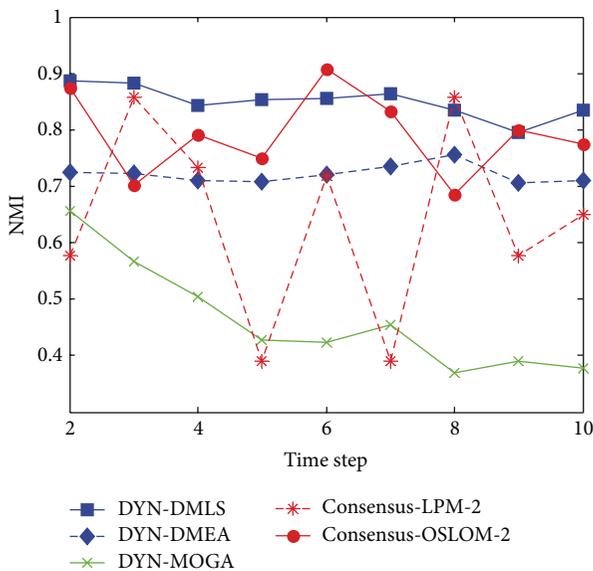


FIGURE 7: NMI results of the five algorithms on the SYN-FIX dataset with $z = 7$.

obtained by DYN-DMEA range below 0.8 at the majority of time steps, while those obtained by DYN-MOGA range below 0.7 all the time. Therefore, it is obvious that the stability of DYN-DMLS is the best, DYN-DMEA is the second, and DYN-MOGA is the worst in the three algorithms. Consensus with OSLOM shows some fluctuation between time stamps but still converges as expected. Its result range is relatively

high. Consensus with LPM varies heavily as usual and its range is wider and lower than that of consensus with OSLOM.

Figure 9 shows the statistical average value of NMI, when $z = 8$. Now the networks are a complete mess. In this situation, though DYN-DMLS hardly detects the community structure effectively, it still improves the result over time. It is the same tendency with DYN-DMEA, while DYN-MOGA just stays at a low level without any vitality. Therefore, the community structure found by DYN-DMLS is the best in the three algorithms. Unfortunately, consensus clustering could not provide any result. Actually the reason is not clear. It may be relevant to the network. On SYN-VAR when $z = 8$, consensus clustering works well.

Figure 10 shows the box plots to illustrate the distribution of the value of NMI when $z = 8$. Consensus clustering is not included for having no results. Three MOEAs are all of high variation ranging from bottom to top. It can be seen clearly the result of DYN-MOGA is stable but not good enough and that of DYN-DMEA could hardly reach more than 0.8. Therefore, DYN-DMLS is still the best in the three algorithms.

4.1.2. Results on SYN-VAR. In this study, we generate the SYN-VAR datasets under four different levels by setting $z = 5, 6, 7, 8$, where the community structures gradually change from clear to fuzzy. As introduced before, SYN-VAR contains the merging and splitting of communities. The dynamic process is more complex than SYN-FIX. Next the performance of the five algorithms on SYN-VAR would be checked.

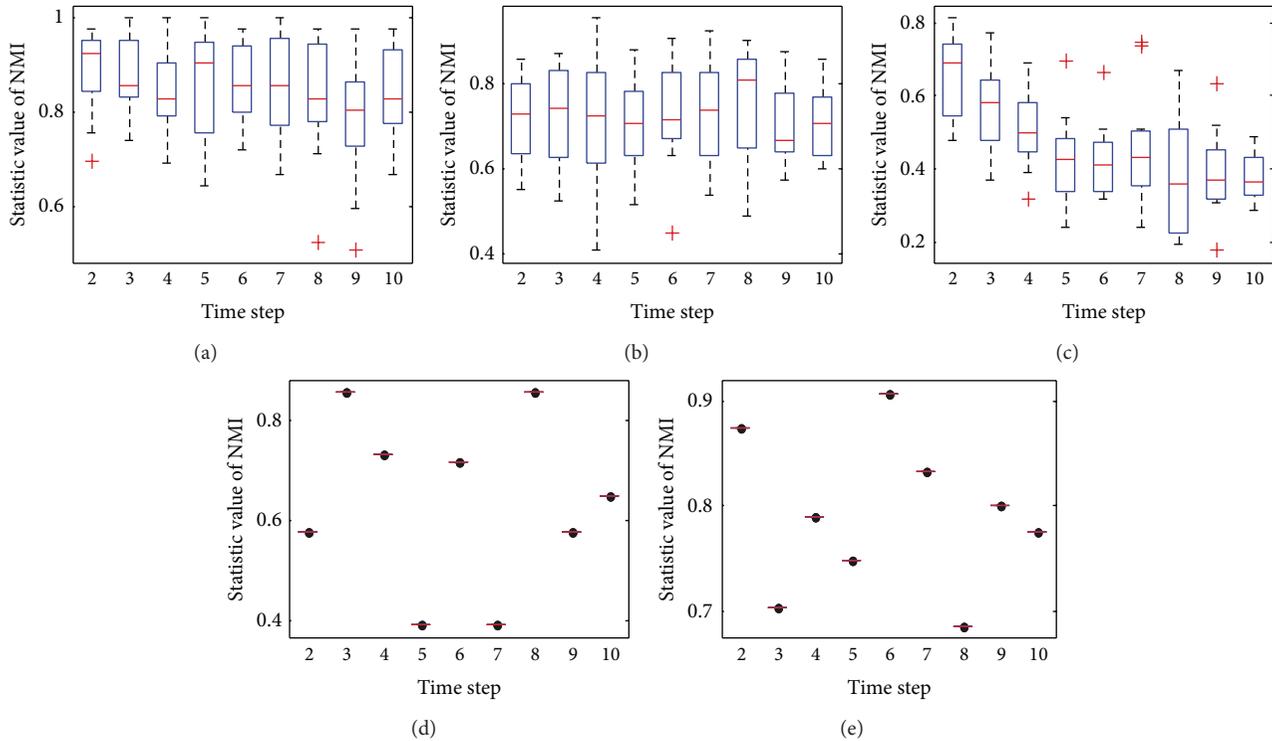


FIGURE 8: The box plots to illustrate the distribution of NMI at each time step when $z = 7$. (a) The box plot for DYN-DMLS; (b) the box plot for DYN-DMEA; (c) the box plot for DYN-MOGA; (d) the box plot for consensus with LPM; (e) the box plot for consensus with OSLOM.

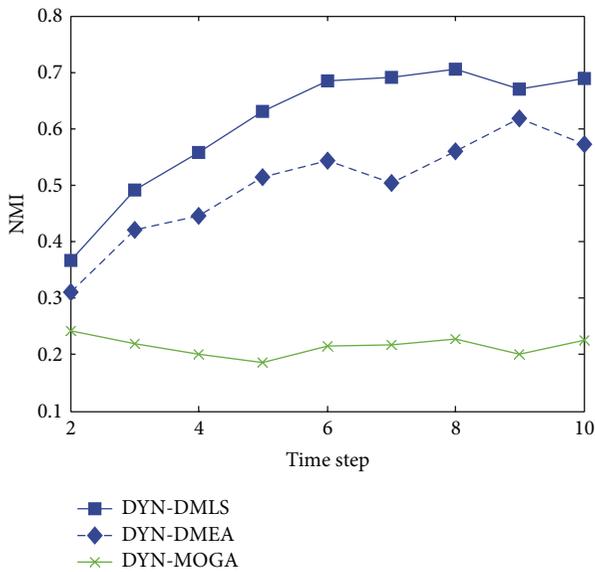


FIGURE 9: NMI results of the three algorithms on SYN-FIX dataset with $z = 8$.

Figure 11 shows the statistical average value of NMI, when $z = 5$. DYN-DMLS and DYN-DMEA perform similarly well and a tiny promotion brought by local search could be observed. The curve of DYN-MOGA is much lower than the above two. The curve obtained by consensus clustering is low at most time at most time. Though the single run

of LPM or OSLOM on the single time step network could result in the same with ground truth, the combination of the accurate structure could hardly satisfy neither of the involved time steps as the two structures differ much from each other. Consensus clustering may produce meaningful results, but it is not easy to prove by comparing with ground truth. However, MOEAs could do better under this evaluation system. Relatively speaking, DYN-DMLS could catch the merging and splitting of communities. More exactly, merging is traced more close than splitting as the second half of time steps, during which communities are merging, results better.

Figure 12 shows the box plots to illustrate the distribution of the value of NMI on SYN-VAR when $z = 5$. The network with $z = 5$ is relatively simple but the stability does not seem well for such a simple network. As the dynamic becomes more complex than that in SYN-FIX, greater fluctuation is obtained by all the five algorithms. Consensus clustering always gets the same result because the static result of each time step keeps invariant in every repeat.

In SYN-VAR, every two consecutive time steps are less similar. It affects MOEAs more than consensus clustering. The merging and splitting of communities would lower the value of objective function NMI directly. When the objective is low, the number of probable solutions would increase. Generally speaking, it leads to a larger search space. Then, great fluctuation appears in the box plot. As to consensus algorithm, the deviation of two consecutive time steps would affect neither the separated static run at each time step nor the consensus process. But generally consensus clustering could not obtain a satisfying result.

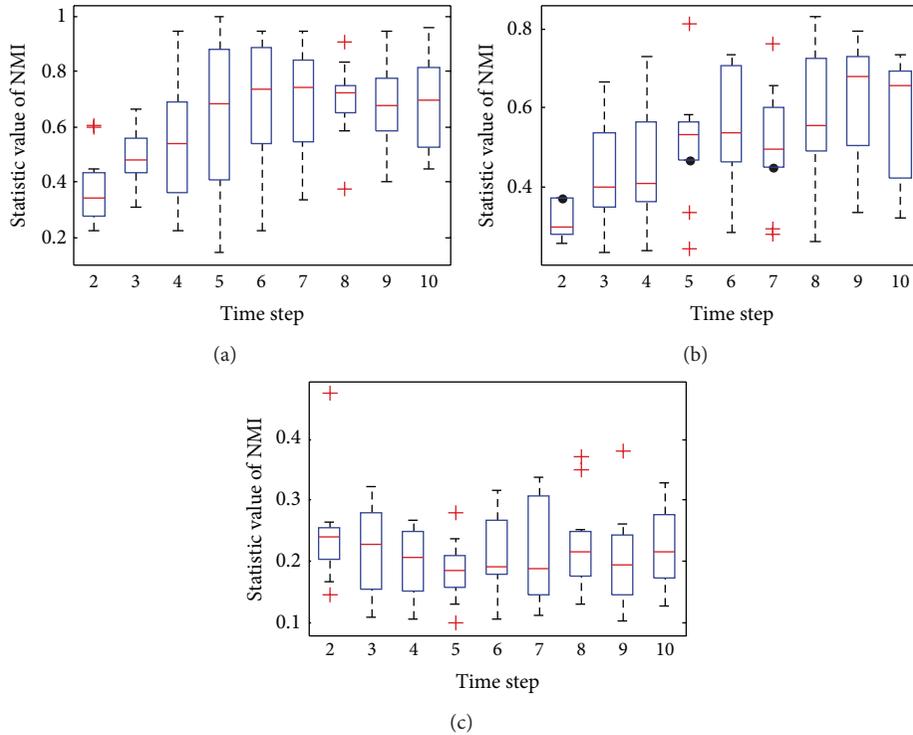


FIGURE 10: The box plot to illustrate the distribution of NMI at each time step when $z = 8$. (a) The box plot for DYN-DMLS; (b) the box plot for DYN-DMEA; (c) the box plot for DYN-MOGA.

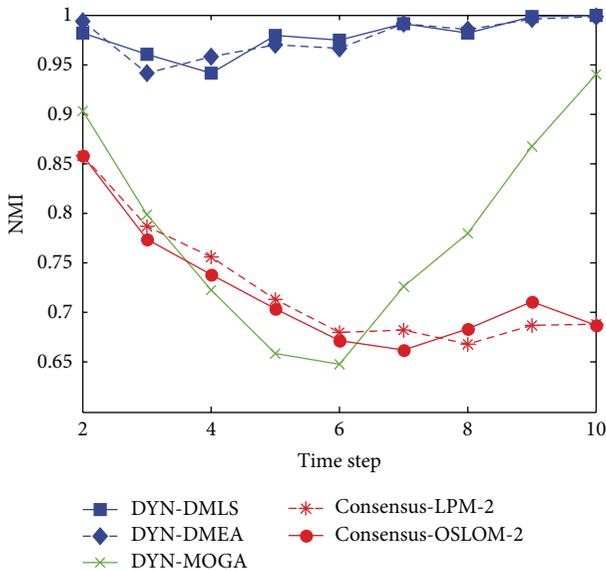


FIGURE 11: NMI results of the five algorithms on the SYN-VAR dataset with $z = 5$.

Figure 13 shows the statistical average value of NMI when $z = 6$. Figure 14 shows results with $z = 7$ and 8 . These curves reflect the same tendency. DYN-DMLS here outperforms other algorithms obviously. Even when $z = 8$, a high NMI value close to 1 is obtained. The others fail to do so. Consensus clustering can work on SYN-VAR too. When $z = 7$ or 8

networks are complex, it still can provide a satisfying result, better than DYN-MOGA but worse than DYN-DMLS and DYN-DMEA. Figure 15 shows the box plots of DYN-DMLS and consensus-OSLOM-2 to illustrate the distribution of the value of NMI on SYN-VAR when $z = 6, 7, 8$. Other algorithms are ignored here. The two are enough to lead to the conclusion.

Overall considering the experimental results on synthetic networks, we can conclude the following.

- (a) On most of the tested synthetic networks, DYN-DMLS performs best. Without local search process, DYN-DMEA always results in being a little lower than DYN-DMLS. This proves that the local search is effective. However, DYN-MOGA performs poor. These indicate the rationality of our method.
- (b) Consensus clustering is good at finding the consensus part from a set of structures. Its result may be meaningful. But to our evaluation metric, its performance is not so good. Consensus clustering is of strong convergence. As we can see from the box plot, repeats bring about the same result. Because of the strong convergence and the stable results from static algorithms, the results may be rough and have no chance to be improved. The most important question is that which structure should be regarded as the corresponding ground truth for comparison. The evaluation metric of consensus clustering on dynamic network is tough.

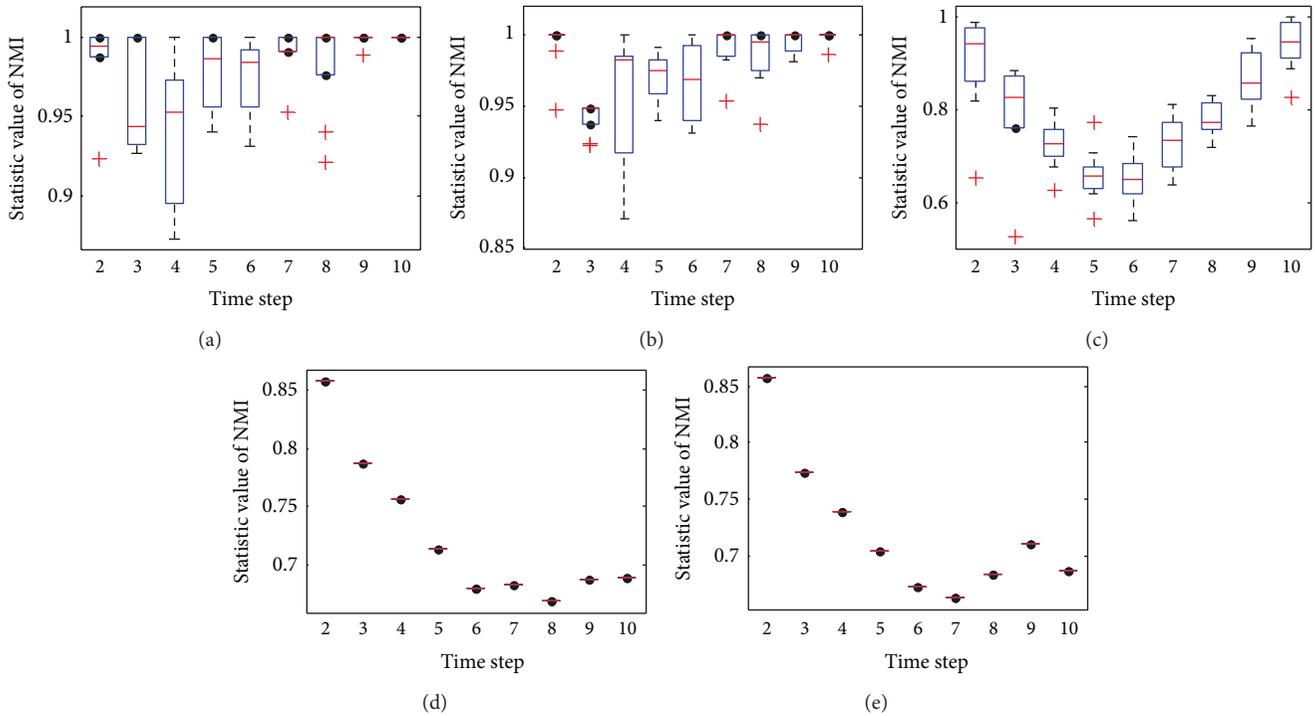


FIGURE 12: The box plots to illustrate the distribution of NMI at each time step on SYN-VAR when $z = 5$. (a) The box plot for DYN-DMLS; (b) the box plot for DYN-DMEA; (c) the box plot for DYN-MOGA; (d) the box plot for consensus with LPM; (e) the box plot for consensus with OSLOM.

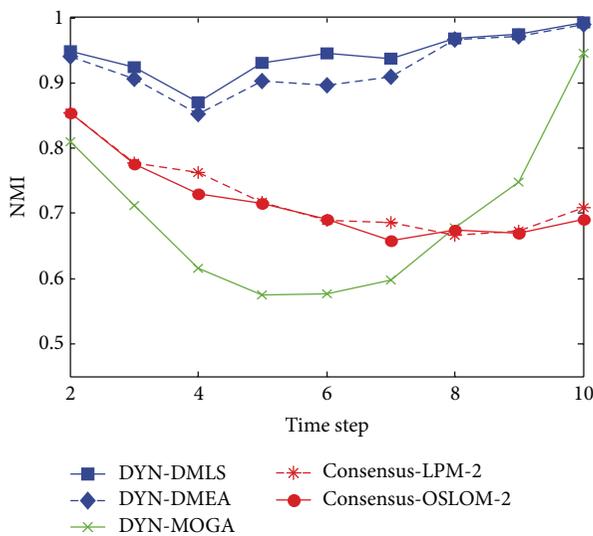


FIGURE 13: NMI results of the five algorithms on the SYN-VAR dataset with $z = 6$.

4.2. Experiments on Real-Life Datasets. In this section, we present experimental studies on three real-life datasets: the football network dataset (<http://www.jhowell.net/cf/scores/scoresindex.htm>), the VAST dataset (<http://www.cs.umd.edu/hcil/VASTchallenge08>), and the DBLP Coauthorship Dataset (<http://www.informatik.uni-trier.de/~ley/db/>).

4.2.1. Results on Football Network Dataset. The football network dataset is the National Collegiate Athletic Association

(NCAA) Football Division 1-A Schedule, which has been used by Newman and Girvan [2]. The NCAA divides 116 schools into eleven conferences and games are more frequent between members of the same conference; in addition, there are four independent schools: Army, Brigham Young, Navy, and Notre Dame, which are grouped into one conference, but they have no more games with members of the same conference than that of the other conference. Nodes in the graph represent teams and edges represent the regular season games between the two teams. In our study, we select the years 2005–2009 to evaluate the performance of our algorithm, each year as a snapshot graph. Therefore, we investigate the football match network over the 5 time steps and there are 12 conferences and 120 teams at each time step. Because there is a priori knowledge about the ground truth of the community structure in the football network, we still employ NMI to evaluate the performance of our algorithm.

Note that the football dataset is dynamic. Though the ground truth is invariant, the network is always changing. The edges between nodes are changing. At each timestamp, it is not prior knowledge that the division is the same. It could be considered as a kind of strong time smoothing information. If the previous result is accurate enough, time smoothing would contribute more to getting a satisfying result for the current. This is not against the dynamic assumption but a probable situation.

Figure 17 shows the statistical average value of NMI with respect to the ground truth over time. It can be clearly seen that both DYN-DMLS and DYN-DMEA algorithms have much better performance than DYN-MOGA, while

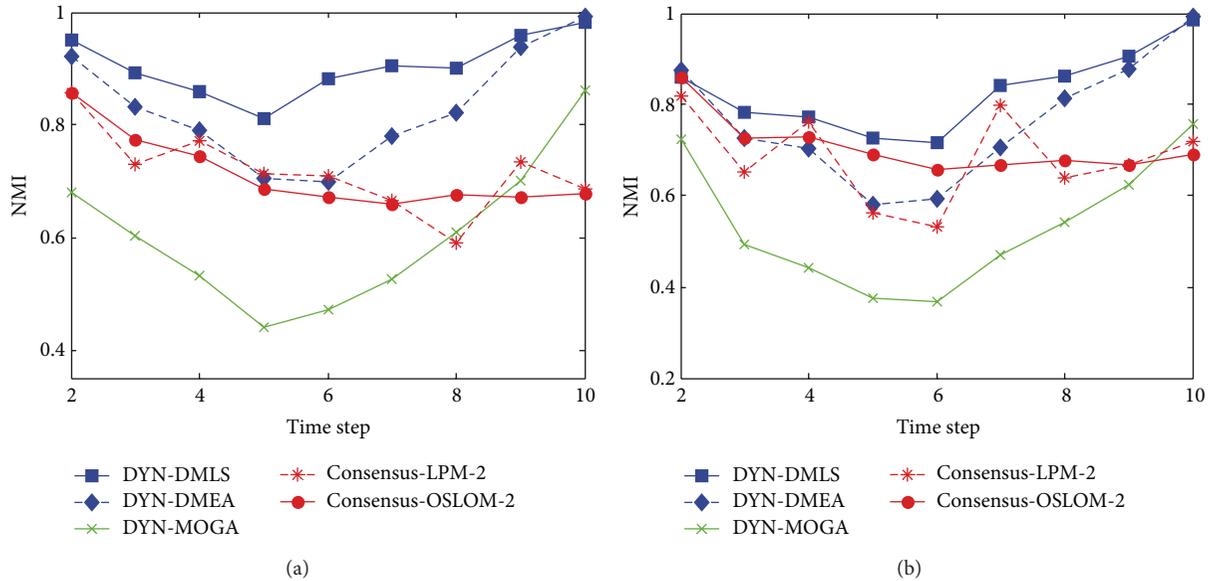


FIGURE 14: NMI results of the five algorithms on the SYN-VAR dataset with $z = 7$ and 8 .

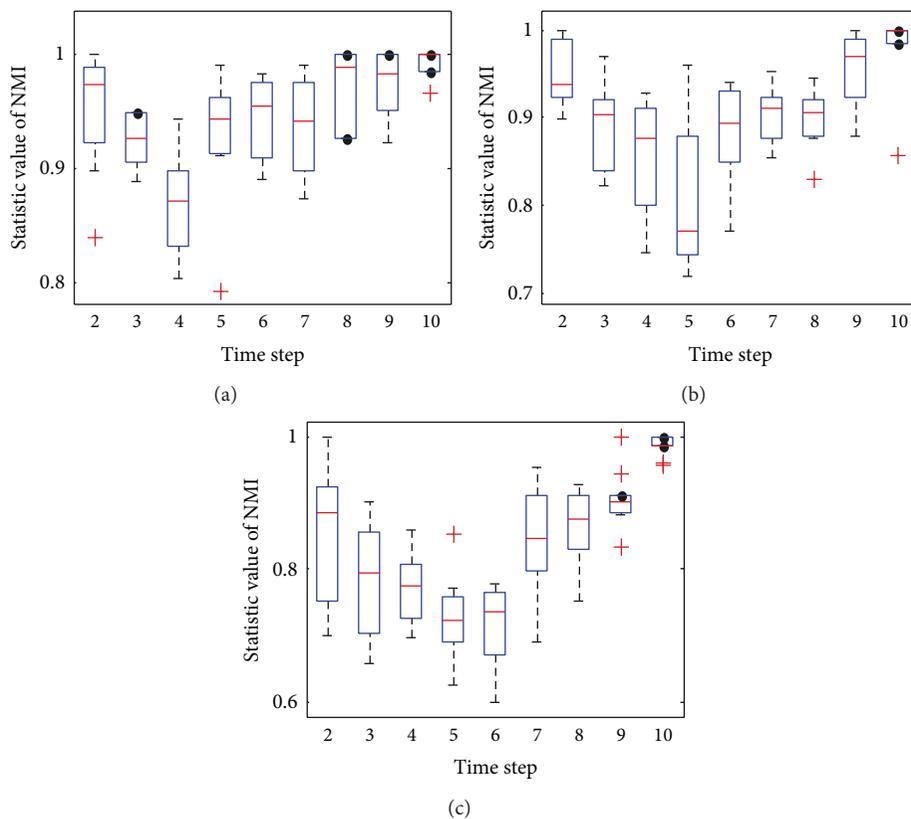


FIGURE 15: The box plots to illustrate the NMI distribution of DYN-DMLS at each time step on SYN-VAR when $z = 6, 7, 8$.

DYN-DMLS outperforms the DYN-DMEA. The average value of NMI obtained by DYN-DMLS is around 0.9 at each time step, which demonstrates that DYN-DMLS can find community structure accurately at each time step. However, consensus clustering is shining here. Both consensus-LPM-2

and consensus-OSLOM-2 could do better than DYN-DMLS. This is due to the fact that the football network has an invariant ground truth which gives it an environment similar to the static. So consensus clustering could produce a better result.

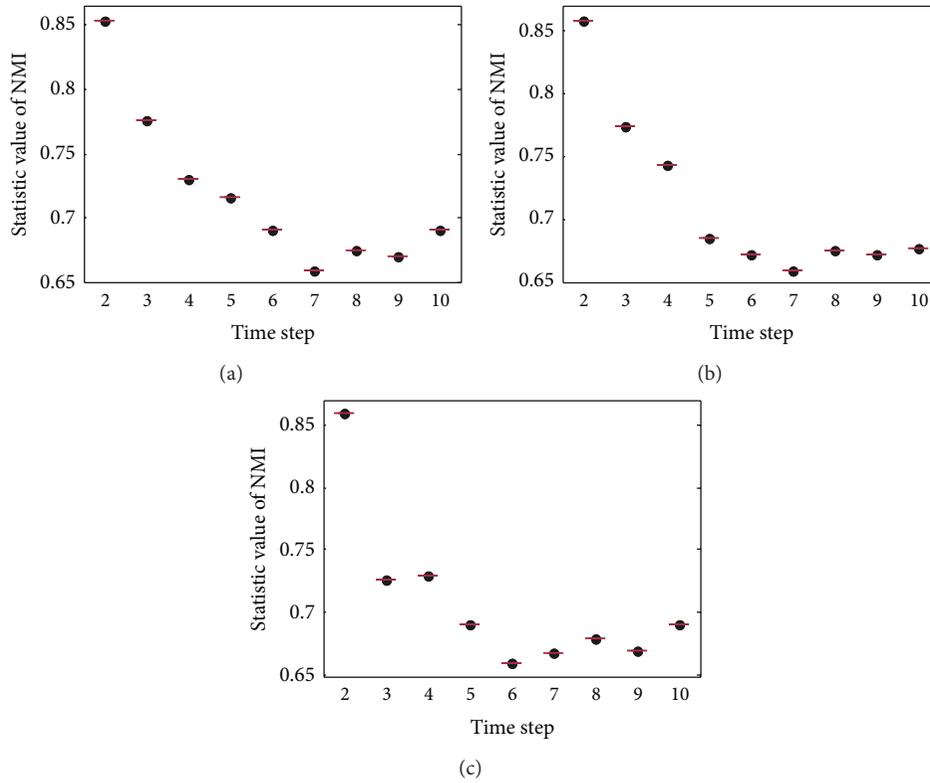


FIGURE 16: The box plots to illustrate the NMI distribution of consensus-OSLOM-2 at each time step on SYN-VAR when $z = 6, 7, 8$.

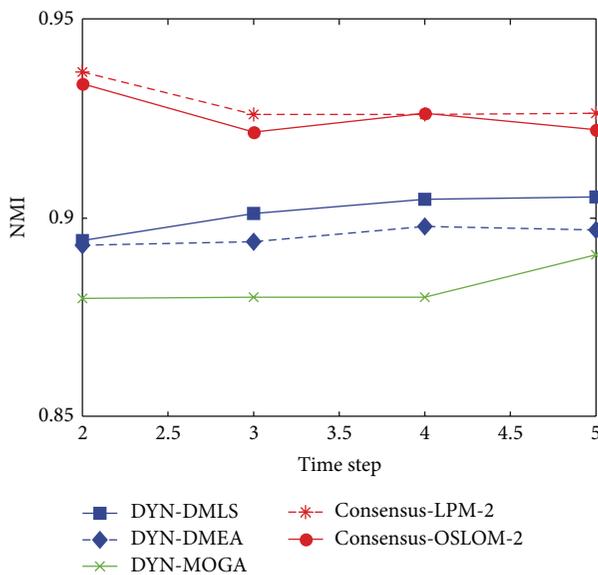


FIGURE 17: NMI results of the football dataset.

In order to analyze visually, the communities found by our algorithm DYN-DMLS on the football network for the year 2009 are shown in Figure 18. The figure is obtained by using Pajek software [50]. The nodes with the same color denote that they belong to the same communities (i.e., conference). Particularly, we use 12 distinct RGB colors to

label 12 true communities, which can be seen in Figure 16 in detail.

As what can be seen from Figure 18, DYN-DMLS can find 11 different communities. Almost all teams can be classified into true communities that they really belong to, which is an impossible mission for the other two algorithms. It can be clearly seen that only eight teams are mistakenly divided to the conferences Big 12, MAC, MWC, Pac 10, and WAC, respectively, which are shown in different colors in these 11 communities, where the teams of four independent schools, Army, Brigham Young, Navy, and Notre Dame, are included. The teams of the four independent schools should be in the same community according to the true communities partition, but they are divided into the other communities due to the fact that they have more frequent games with the teams in the other communities than between them. In a word, our algorithm can get better performance than the other two algorithms.

4.2.2. Results on VAST Dataset. The VAST Dataset is a challenge task from IEEE VAST 2008, whose primary task is to characterize the Catalno/Vidro social network based on the cell phone call data provided and to characterize the temporal changes in the social structure over the 10-day period.

This dataset consists of information about 9834 calls between 400 cellphones over a 10-day period in June 2006 in the Isla Del Sueño. It includes records with the following

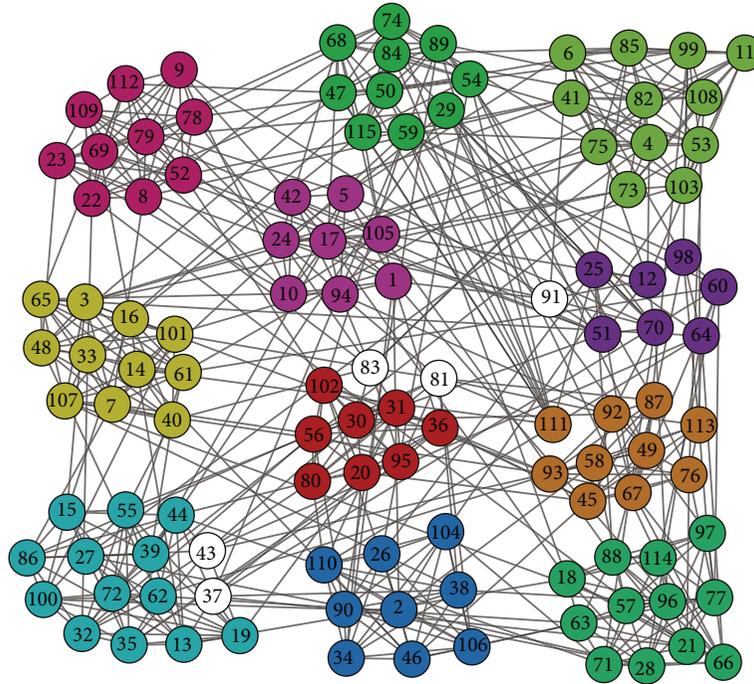


FIGURE 18: The communities found by DYN-DMLS on the football network for the year 2009.

fields: identifier for caller, identifier for receiver, time, duration, and call origination cell tower. In order to detect the communication patterns, we construct call graphs based on the call records. In order to evaluate our algorithm better, we convert the input social network and the corresponding dynamic graph into 5 snapshot graphs, where the graphs in every two days are aggregated into one snapshot graph and therefore we have 5 snapshot graphs over 10 days.

Note that the dataset records phone chains in 10 days. The phone call is a kind of temporary connection. It is in a short time window that the community structure would be confusing. Two members in the same community may make phone calls every day, while it is also common that there are just a few calls in the 10 days. One might not be able to catch the relationship at any time. To handle this kind of hidden information, time smoothing could work. As NMI is a kind of statistical information, little missing information would not affect the result too much. Therefore, time smoothing makes sense macroscopically.

Due to no a priori knowledge about ground truth of the cellphone network, the result has been figured out. Here, we only discover the community structure in the network to evaluate the performance of our algorithm, rather than performing the contest task which is the goal of the Mini Challenge 3.

As a challenge task from IEEE VAST 2008, this dataset has been analyzed by many researchers. It has been confirmed that the structure of the cellphone network changes drastically from the 7th day to the 8th day [51, 52]; that is, a significant variation happened at the high-level leaders during this period. Due to the 4th snapshot graph integrating the two graphs on the 7th and the 8th days, we display

the main structure of the cellphone network at time step 3 and at time step 5 to analyze the changes before and after the event. It can be seen from Figure 19(a) that node 200 is the main leader, which contacts with nodes 1, 2, 3, and 5 during this time step, while these nodes are also group leaders in the Catalano hierarchy at time step 3. However, it can be clearly seen from Figure 19(b) that these nodes 300, 306, 309, 360, and 397 emerge as a new hierarchy at the time step 4. The community structure discovered by DYN-DMLS is consistent with the analysis that has been made.

4.2.3. Results on DBLP Coauthorship Dataset. DBLP Coauthorship dataset is obtained from DBLP database, which has been described in [53]. In order to evaluate the performance of our algorithm, we choose a little part of DBLP data from the DBLP Coauthorship dataset to compose a connected graph. We select some authors from the DBLP database to form a small-scale DBLP Coauthorship dataset, which mainly focuses on the data mining areas. This dataset contains the coauthorship information among these papers over six years (2005–2010). By selecting authors who publish papers every year during the period of six years at these conferences including in DBLP database, 70 authors are chosen from the database to build connected snapshot graphs, where the nodes represent authors and edges represent coauthorship between two authors (nodes). The labels of these nodes corresponding to these authors are displayed in Table 1. For analyzing community evolutions, we aggregate data in every two years into one time step and therefore we have 3 time steps in total (corresponding to 3 snapshot graphs) for the dynamic network. In addition, we are not judging the quality or quantity of papers by an author. Instead, the importance of

TABLE 1: The author corresponding to the nodes in snapshot graphs.

1 "P. S. Yu"	2 "C. C. Aggarwal"	3 "M. S. Chen"	4 "W. Fan"	5 "B. Gedik"	6 "J. Han"
7 "B. Liu"	8 "L. Liu"	9 "J. Pei"	10 "H.X. Wang"	11 "K. Wang"	12 "K. L. Wu"
13 "Y. Xu"	14 "X. F. Yan"	15 "Z. F Zhang"	16 "J.Y. Wang"	17 "C. C. Chen"	18 "H. L. Chen"
19 "M. C. Chen"	20 "W. T. Chen"	21 "Y. H. Chu"	22 "K.T. Chuang"	23 "J. M. Ho"	24 "J.H. Hsiao"
25 "C. M. Hsu"	26 "J. W. Huang"	27 "H. P. Hung"	28 "K. H. Liu"	29 "W. G. Teng"	30 "C. Y. Tseng"
31 "M. Y. Yeh"	32 "K. Zhang"	33 "L. Liu"	34 "D. Cai"	35 "C. Chen"	36 "H. Gonzalez"
37 "X. F. He"	38 "S. K. Kim"	39 "X. L. Li"	40 "H.Y. Liu"	41 "Q. Z. Mei"	42 "Z. Shao"
43 "D. Xin"	44 "X. X. Yin"	45 "C. X. Zhai"	46 "F.D. Zhu"	47 "N. J."	48 "X. I. Li"
49 "J. Caverlee"	50 "K. K. Chen"	51 "A. Iyengar"	52 "C. Pu"	53 "A. Singh"	54 "M. Srivatsa"
55 "J. Yin"	56 "A. W. C. Fu"	57 "D. X. Jiang"	58 "X.M. Lin"	59 "Y. F. Tao"	60 "R. C. W. Wong"
61 "X. K. Xiao"	62 "X. m. Lin"	63 "X. F Meng"	64 "C. Zaniolo"	65 "B. C. M. Fung"	66 "E. P. Lim"
67 "H. W. Lauw"	68 "J.N. K. Liu"	69 "W. M. Ma"	70 "R. She"		

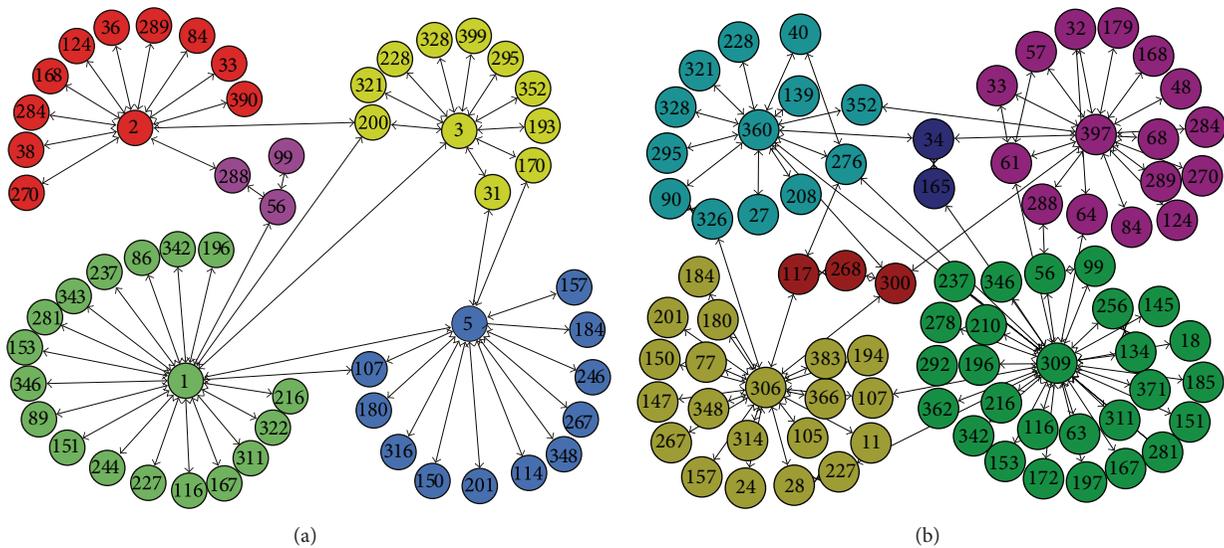


FIGURE 19: The main community structure of VAST found by DYN-DMLS at time step 3 and at time step 5. (a) The main community structure at time step 3; (b) the main community structure at time step 5.

a node in a community is determined by its contribution to the community structure.

In this experiment, we apply our DYN-DMLS algorithm to analyze this dynamic network. Firstly, we detect the communities in snapshot graph at first time step (2005-2006) without smooth evolution by employing the memetic community detection algorithm (Meme-Net) [38] and the partition can be seen in Figure 20(a).

Then, DYN-DMLS algorithms are employed to detect the communities on the snapshot graphs over the other two time steps. Using the solution of our algorithm, we analyze how some individual authors' community membership changes over time. Figure 20 shows the community partition on snapshot graph at each time step.

From Figure 20, we have the following observations. First, these partitions of the snapshot network at each time step can reflect the community structure well. Second, these partitions can reflect the temporal evolution well. For example, the nodes 6 and 13 are clustered into C5 at the first time step

but are clustered into the C1 at the second and third time step; similarly, the nodes 11, 17, 62, and 63 also leave their original community and join the C1. In addition, the nodes 65, 66, and 57 are clustered into the C4 at the third time step, but they belong to the C6 at the first and the second time step. Therefore, the nodes divided into C1 and C4 over time reflect the temporal evolution well. The temporal evolution of the community structure is aroused by varying connection between these nodes; that is, the coauthorships among individual authors evolve over time. From the analyzing of these figures, we can find that the partition obtained by our algorithm corresponds with the ground truth of coauthorship among authors in these bibliographies in DBLP database.

4.3. *The Effect of NMI as Time Smoothing.* In our approach based on evolutionary clustering, NMI between the consecutive time steps is used to represent time smoothing. Time smoothing assumes that two consecutive time steps have structure connection with each other. Their community

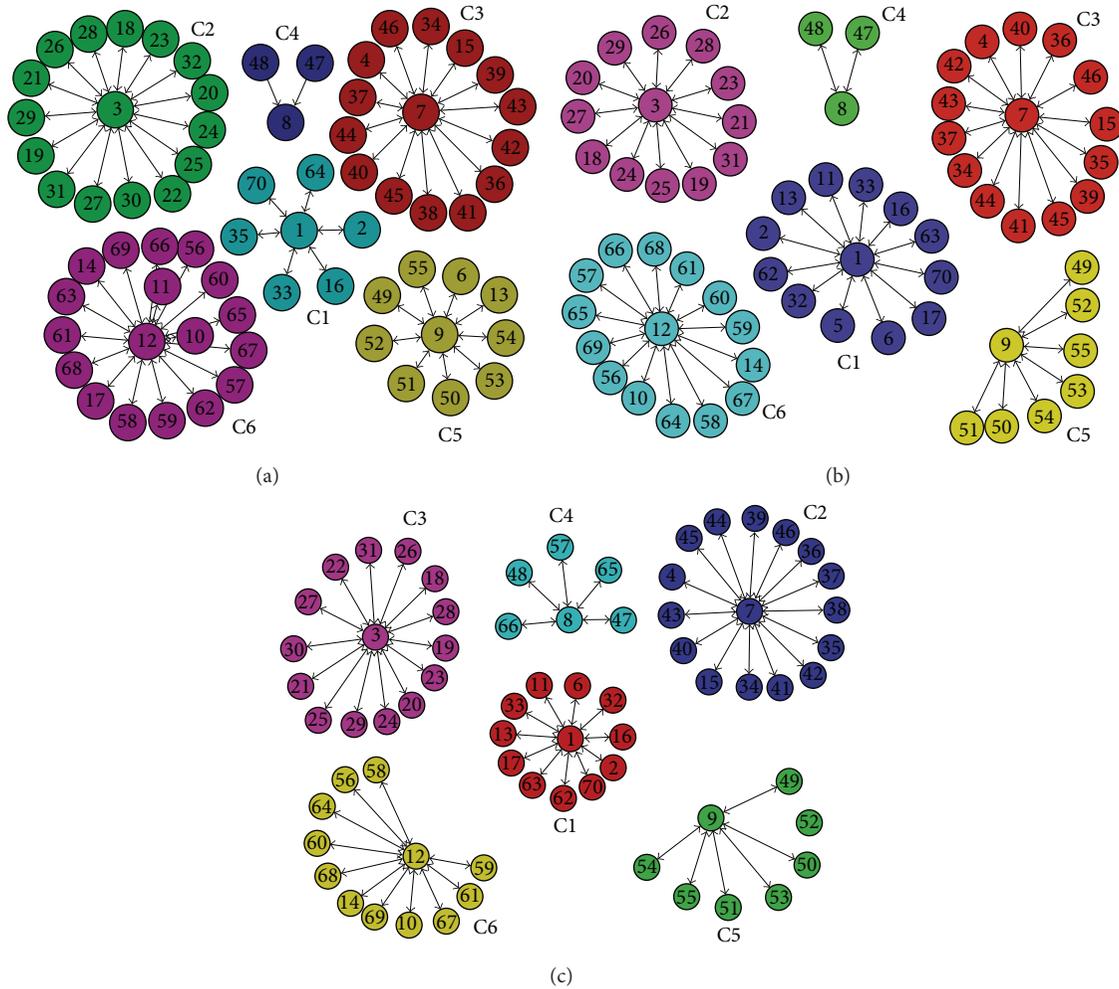


FIGURE 20: (a) The community partition on snapshot graph at first timestamp (2005-2006) without smooth evolution; (b) the community partition on snapshot graph at second timestamp (2007-2008) with smooth evolution; (c) the community partition on snapshot graph at third timestamp (2009-2010) with smooth evolution.

structures may not be the same but at least have a relatively higher similarity than that of two totally different ones. In evolutionary clustering, this corresponds to history cost (modularity corresponds to snapshot quality). NMI plays the role to measure the similarity. The higher the value of NMI is, the more similar the two consecutive time steps are. Therefore, its history cost is small. In the multiobjective optimization process, the solution which has high NMI but low modularity would be reserved in the population and it has a better chance to evolve into a proper solution than the one which has low NMI and modularity. Therefore, NMI can work as expected.

The experiment compares results of the two approaches on the benchmark network with $z = 8$. One is based on a single optimization of modularity which just calculates snapshot quality at every time step, and the other is our DYN-DMLS algorithm which simultaneously optimizes NMI and modularity (NMI represents time smoothing, while modularity represents snapshot quality).

When $z = 8$, networks are hard to cluster. In Figure 21, it is obvious that, in the complex situation ($z = 8$), single

optimization of modularity could hardly improve the result. With extra consideration on time smoothing, higher NMI (NMI here is evaluation metric which is different from objective function as time smoothing) results are obtained. It also shows that the proposed method brings about a significant improvement by simultaneously optimizing snapshot quality (modularity) and history cost (NMI). Therefore, it proves the effectiveness of NMI as time smoothing.

4.4. Time Symmetry of DYN-DMLS. Though the result of the first time step has an influence on the subsequent process, the influence is rather limited and diminishing. When calculating objective function NMI, the result is directly determined by the two consecutive time steps, the current one and the previous one. The t th time step may directly affect the $(t + 1)$ th but can hardly affect the $(t + 2)$ th. To each single time step except the first one, it is always the previous one that determines its time smoothing. It is hard to quantify the diminishing influence. Besides, there is still a weighted parameter to balance between history cost and snapshot quality. Time smoothing as history cost could hardly contribute more to

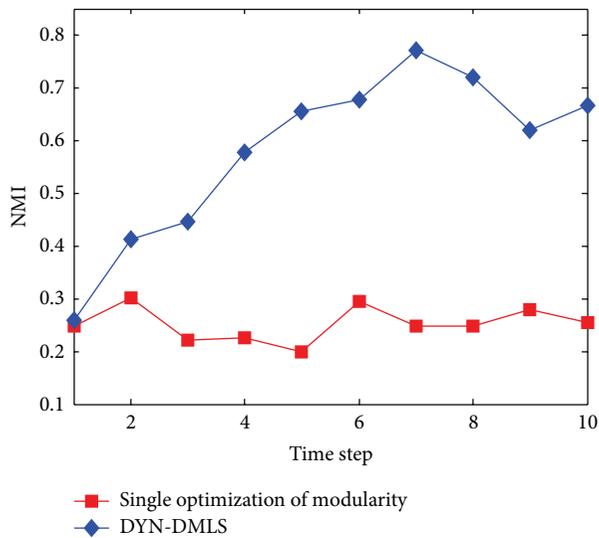


FIGURE 21: NMI results on the SYN-FIX with $z = 8$. Blue line represents the result of DYN-DMLS which simultaneously optimizes modularity and time smoothing. Red line represents the result of a single optimization of modularity at each time step.

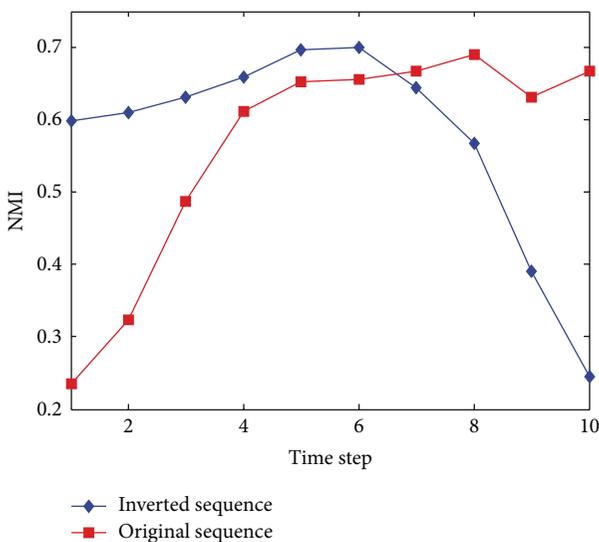


FIGURE 22: NMI results of the two sequences on the SYN-FIX with $z = 8$. Red line represents the result of a common and original sequence from 1 to 10, while blue line represents the result of an inverted sequence from 10 to 1. Method applied is DYN-DMLS.

the result. Therefore, the first time step has not been given an excessive importance.

From another point of view, multiobjective optimization is a global statistical search process. Each step may generate different results in different runs. So time symmetric may be a property of the dynamic network but in multiobjective optimization it is hard to maintain.

To support the above description, we test the dynamic network in an inverted time sequence and compare its result with the common one. Strictly speaking, it is not symmetric.

The result is shown in Figure 22. It is the improvement to the result that makes it look symmetric. The figure also tells that the first time step has not been given an excessive importance.

5. Concluding Remarks

The detection of communities and analysis of the community evolution in dynamic networks with temporal smoothness is a new challenging research problem with broad applications. In this paper, the two cost functions, community quality function and temporal cost function, are optimized simultaneously by the decomposition-based multiobjective evolutionary algorithm with a local search. The methods can provide the solution representing the best tradeoff between the accuracy of the communities structures obtained and the similarity between one time step and the previous one, without fixing a weight parameter in advance. In addition, a local search operator is incorporated into our method according to the problem-specific knowledge, which has a better ability to search the solution, especially when the community structure changes more dramatically over time. Experiments on SYN-VAR benchmark demonstrate that the proposed algorithm has a better accuracy in extracting community and capturing community evolution than the classic DYN-MOGA and consensus clustering algorithm. In our future work, we will expand our algorithm to be suitable for processing the large-scale networks in real life. Some better local search strategies should be studied to incorporate into our method to improve the performance further.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 61273317, 61202176, and 61203303), the National Top Youth Talents Program of China, the Specialized Research Fund for the Doctoral Program of Higher Education (Grant no. 20130203110011), and the Fundamental Research Fund for the Central Universities (Grant nos. K50510020001 and K5051202053).

References

- [1] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.
- [2] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, Article ID 026113, 2004.
- [3] B. Yang, W. K. Cheung, and J. Liu, "Community mining from signed social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 10, pp. 1333–1348, 2007.
- [4] G. Palla, A.-L. Barabási, and T. Vicsek, "Quantifying social group evolution," *Nature*, vol. 446, no. 7136, pp. 664–667, 2007.

- [5] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *Proceedings of the International Conference on Advances in Social Network Analysis and Mining (ASONAM '10)*, pp. 176–183, Odense, Denmark, August 2010.
- [6] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin, "Detecting communities and their evolutions in dynamic social networks—a Bayesian approach," *Machine Learning*, vol. 82, no. 2, pp. 157–189, 2011.
- [7] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Analyzing communities and their evolutions in dynamic social networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 2, article 8, 2009.
- [8] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pp. 554–560, Philadelphia, Pa, USA, August 2006.
- [9] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.
- [10] L. Danon, A. Daz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics*, vol. 2005, Article ID P09008, 2005.
- [11] A. Konstantinidis and K. Yang, "Multi-objective energy-efficient dense deployment in Wireless Sensor Networks using a hybrid problem-specific MOEA/D," *Applied Soft Computing Journal*, vol. 11, no. 6, pp. 4117–4134, 2011.
- [12] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/ D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.
- [13] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2010.
- [14] F. Folino and C. Pizzuti, "A multiobjective and evolutionary clustering method for dynamic networks," in *Proceedings of the International Conference on Advances in Social Network Analysis and Mining (ASONAM '10)*, pp. 256–263, Odense, Denmark, August 2010.
- [15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [16] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, Article ID 036106, 11 pages, 2007.
- [17] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proceedings of the Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pp. 95–100, 2001.
- [18] C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.
- [19] M. Gong, Q. Cai, X. Chen, and L. Ma, "Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 82–97, 2014.
- [20] M. Gong, X. Chen, L. Ma, Q. Zhang, and L. Jiao, "Identification of multi-resolution network structures with multi-objective immune algorithm," *Applied Soft Computing*, vol. 13, no. 4, pp. 1705–1717, 2013.
- [21] L. Ma, M. Gong, Q. Cai, and L. Jiao, "Enhancing community integrity of networks against multilevel targeted attacks," *Physical Review E*, vol. 88, no. 2, Article ID 022810, 2013.
- [22] M. Gong, L. Jiao, H. Du, and L. Bo, "Multiobjective immune algorithm with nondominated neighbor-based selection," *Evolutionary Computation*, vol. 16, no. 2, pp. 225–255, 2008.
- [23] M. Gong, L. Ma, Q. Zhang, and L. Jiao, "Community detection in networks by using multiobjective evolutionary algorithm with decomposition," *Physica A*, vol. 391, no. 15, pp. 4050–4060, 2012.
- [24] M. Gong, L. Zhang, L. Ma, and L. Jiao, "Community detection in dynamic social networks based on multiobjective immune algorithm," *Journal of Computer Science and Technology*, vol. 27, no. 3, pp. 455–467, 2012.
- [25] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.
- [26] J. Hopcroft, O. Khan, B. Kulis, and B. Selman, "Tracking evolving communities in large linked networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 1, pp. 5249–5253, 2004.
- [27] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*, pp. 153–162, August 2007.
- [28] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Community evolution in dynamic multi-mode networks," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pp. 677–685, August 2008.
- [29] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "FacetNet: a framework for analyzing communities and their evolutions in dynamic networks," in *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*, pp. 685–694, Beijing, China, April 2008.
- [30] M. S. Kim and J. Han, "A particle-and-density based evolutionary clustering method for dynamic networks," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 622–633, 2009.
- [31] A. Lancichinetti and S. Fortunato, "Consensus clustering in complex networks," *Scientific Reports*, vol. 2, article 336, 2012.
- [32] R. Ahmed and G. Karypis, "Algorithms for mining the evolution of conserved relational states in dynamic networks," *Knowledge and Information Systems*, vol. 33, no. 3, pp. 603–630, 2012.
- [33] J. Kunegis, D. Fay, and C. Bauckhage, "Spectral evolution in dynamic networks," *Knowledge and Information Systems*, vol. 37, no. 1, pp. 1–36, 2013.
- [34] D. Patnaik, S. Laxman, and N. Ramakrishnan, "Discovering excitatory relationships using dynamic Bayesian networks," *Knowledge and Information Systems*, vol. 29, no. 2, pp. 273–303, 2011.
- [35] W. Peng and T. Li, "Temporal relation co-clustering on directional social network and author-topic evolution," *Knowledge and Information Systems*, vol. 26, no. 3, pp. 467–486, 2011.
- [36] L. Zhao, X. Guan, and R. Yuan, "Modeling collective blogging dynamics of popular incidental topics," *Knowledge and Information Systems*, vol. 31, no. 2, pp. 371–387, 2012.
- [37] K. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer, Norwell, Mass, USA, 1999.

- [38] M. G. Gong, B. Fu, L. C. Jiao, and H. F. Du, "A memetic algorithm for community detection in networks," *Physical Review E*, vol. 84, no. 5, Article ID 056101, 2011.
- [39] Y. J. Park and M. S. Song, "A genetic algorithm for clustering problems," in *Proceedings of the 3rd Annual Conference on Genetic Programming*, pp. 568–575, 1998.
- [40] J. Handl and J. Knowles, "An evolutionary approach to multiobjective clustering," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 56–76, 2007.
- [41] C. Pizzuti, "GA-Net: a genetic algorithm for community detection in social networks," in *Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN '08)*, pp. 1081–1090, 2008.
- [42] M. J. Barber and J. W. Clark, "Detecting network communities by propagating labels under constraints," *Physical Review E*, vol. 80, no. 2, Article ID 026129, 11 pages, 2009.
- [43] X. Liu and T. Murata, "Advanced modularity-specialized label propagation algorithm for detecting communities in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 389, no. 7, pp. 1493–1500, 2010.
- [44] Z. Li, S. Zhang, R. S. Wang, X. S. Zhang, and L. Chen, "Quantitative function for community detection," *Physical Review E*, vol. 77, no. 3, Article ID 036109, 9 pages, 2008.
- [45] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics*, vol. 2008, no. 10, Article ID P10008, 2008.
- [46] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 4, pp. 1118–1123, 2008.
- [47] J. Chen and Y. Saad, "Dense subgraph extraction with application to community detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 7, pp. 1216–1230, 2010.
- [48] R. McGill, J. Tukey, and W. Larsen, "Variations of Boxplots," *The American Statistician*, vol. 32, pp. 12–16, 1978.
- [49] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *PLoS ONE*, vol. 6, no. 4, Article ID e18961, 2011.
- [50] W. D. Nooy, A. Mrvar, and V. Batagelj, *Exploratory Social Network Analysis with Pajek*, Cambridge University Press, New York, NY, USA, 2005.
- [51] Q. Ye, T. Zhu, D. Hu, B. Wu, N. Du, and B. Wang, "Cell phone mini challenge award: social network accuracy—exploring temporal communication in mobile call graphs," in *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST '08)*, pp. 207–208, October 2008.
- [52] S. Q. Yang, B. Wu, and B. Wang, "Tracking the evolution in social network: methods and results," in *Complex Sciences*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 693–706, 2009.
- [53] S. Asur, S. Parthasarathy, and D. Ucar, "An event-based framework for characterizing the evolutionary behavior of interaction graphs," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 4, article 16, 2009.

Research Article

A Prefiltered Cuckoo Search Algorithm with Geometric Operators for Solving Sudoku Problems

**Ricardo Soto,^{1,2} Broderick Crawford,^{1,3} Cristian Galleguillos,¹
Eric Monfroy,⁴ and Fernando Paredes⁵**

¹ Pontificia Universidad Católica de Valparaíso, 2362807 Valparaíso, Chile

² Universidad Autónoma de Chile, 7500138 Santiago, Chile

³ Universidad Finis Terrae, 7501015 Santiago, Chile

⁴ CNRS, LINA, University of Nantes, 44322 Nantes, France

⁵ Escuela de Ingeniería Industrial, Universidad Diego Portales, 8370109 Santiago, Chile

Correspondence should be addressed to Ricardo Soto; ricardo.soto@ucv.cl

Received 11 November 2013; Accepted 30 December 2013; Published 23 February 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 Ricardo Soto et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The Sudoku is a famous logic-placement game, originally popularized in Japan and today widely employed as pastime and as testbed for search algorithms. The classic Sudoku consists in filling a 9×9 grid, divided into nine 3×3 regions, so that each column, row, and region contains different digits from 1 to 9. This game is known to be NP-complete, with existing various complete and incomplete search algorithms able to solve different instances of it. In this paper, we present a new cuckoo search algorithm for solving Sudoku puzzles combining prefiltering phases and geometric operations. The geometric operators allow one to correctly move toward promising regions of the combinatorial space, while the prefiltering phases are able to previously delete from domains the values that do not conduct to any feasible solution. This integration leads to a more efficient domain filtering and as a consequence to a faster solving process. We illustrate encouraging experimental results where our approach noticeably competes with the best approximate methods reported in the literature.

1. Introduction

The Sudoku is a logic-based placement puzzle, widely present as pastime game in newspapers and magazines. It was initially popularized in Japan during the eighties, but today it is a worldwide popular game and a useful benchmark for testing artificial intelligence solving techniques. A Sudoku puzzle consists in filling a board of 9×9 subdivided into subgrids of size 3×3 so that each row, column, and subgrid contains different digits from 1 to 9. A Sudoku problem includes prefilled cells, namely, the “givens,” which cannot be changed or moved (see Figure 1). Certainly, the amount of givens has limited or no impact on the difficulty of the problem. The difficulty is mostly dependent on the positioning of the givens along the puzzle. A useful difficulty classification including easy, medium, and hard Sudokus has been proposed by Mantere and Koljonen [1].

The literature reports various approaches to solve Sudoku puzzles. For instance, exact methods such as constraint programming [2–4] and boolean satisfiability [5] are well-known candidates for the efficient handling of such kind of puzzles. In the context of approximate methods, genetic programming [1] and metaheuristics in general [6–10] have illustrated promising results. Additional, but less traditional, Sudoku solving techniques have been proposed as well, such as Sinkhorn balancing [11], rewriting rules [12], and entropy minimization [13].

In this paper, we focus on approximate methods. We propose a new algorithm for the efficient solving of Sudoku instances based on cuckoo search, geometric operators, and prefiltering phases. The cuckoo search is a relatively modern nature-inspired metaheuristic [14–18] to which we introduce geometric operators in order to correctly move to promising regions of a discrete space of solutions. This combination

	Subgrid			Column		
			2			5
	1			7	5	
	4				9	
		4	9			7
Row	8		1		3	
		3	6			2
	2				8	
		8		9	2	
			7			8

FIGURE 1: Sudoku puzzle instance.

is additionally enhanced with a domain reducer component based on local consistencies. The idea is to previously delete from the search space the values that do not conduct to any feasible solution. This integration straightforwardly alleviates the work of the metaheuristic leading to a faster solving process. We illustrate encouraging experimental results where our approach noticeably competes with the best approximate methods reported in the literature.

This paper is organized as follows. In Section 2, we describe the previous work. Section 3 presents the classic cuckoo search algorithm. The geometric operators and the prefiltering phase employed are illustrated and exemplified in Sections 4 and 5, respectively. The resulting new cuckoo search algorithm is presented in Section 6, followed by the corresponding experimental results. Finally, in Section 8, we conclude and give some directions for future work.

2. Related Work

Sudoku puzzles have been solved with various techniques during the last decades. For instance, complete methods such as Boolean satisfiability and constraint satisfaction can clearly be used to solve Sudokus [3–5, 19]. In this paper, we focus on incomplete search methods, specially on solving hard instances of such a puzzle. Within this scenario, different solutions have been suggested, mainly based on metaheuristics. For instance, Lewis [7] models the puzzle as an optimization problem where the number of incorrectly placed digits on the board must be minimized. The model is solved by using simulated annealing, but the approach is mostly focused on producing valid Sudokus than on the performance of the resolution. In [10], an ant colony algorithm is proposed, where the problem is modeled in an opposite form: maximizing the number of correctly filled cells. The best result completes only 76 out of 81 cells of the puzzle. In [8], a particle swarm optimizer (PSO) for solving Sudokus is presented, but the goal of authors was rather to validate the use of geometric operators in PSO for complex combinatorial spaces.

In [6], a hill-climbing algorithm for Sudokus is reported. The approach succeeds in solving easy Sudoku instances,

failing for medium and hard ones. In the same work, a genetic algorithm (GA) outperforms the hill-climber previously presented. Such a GA is tuned with geometric operators, in particular Hamming space crossovers and swap space crossovers, reporting solutions for a hard Sudoku. In Mantere and Koljonen [1], another GA is proposed that succeeds for easy and medium instances, but it only reaches the optimum in 2 out of 30 tries for a hard Sudoku. A cultural algorithm is proposed by the same authors [9], but it is generally outperformed by the GA previously reported. In Soto et al. [20], a tabu search is tuned with a prefiltered phase, being capable of solving 30 out of 30 tries for a hard Sudoku.

3. Cuckoo Search

Cuckoo search is a nature-inspired metaheuristic, based on the principle of the brood parasitism of some cuckoo species. This kind of bird has an aggressive reproduction strategy, which is based on the use of foreign nests for incubation. Cuckoos proceed by laying their eggs in nests from other bird species, removing the other bird eggs to increase incubation probability. Eventually, cuckoo eggs may be discovered by the host bird, which might act in two ways: taking off the alien eggs or simply abandoning its nest and building a new one elsewhere. In practice, an egg represents a solution and cuckoo eggs represent potentially better solutions than the current ones in nests. In the simplest form each nest has only one egg.

The cuckoo search procedure for minimization is described in Algorithm 1. The process begins by randomly generating an initial population of host nests. Next, the algorithm iterates until a given stop criterion is reached, which is commonly a maximum number of iterations. At line 3, a new solution is created, normally by employing a random walk via Lévy flights. Equation (1) describes such a random walk, where x_i^{t+1} is the new solution, t corresponds to the iteration number, and the product \otimes means entrywise multiplications. The α parameter is the step size, where $\alpha > 0$, and determines how far the process can go for a fixed number of iterations. Then, at line 4, an Egg_j is randomly chosen to be then compared with the previous one in order to keep the egg exhibiting the best cost. Finally, the worse nests are abandoned depending on the probability p_a and new solutions are built:

$$x_i^{t+1} = x_i^t + \alpha \otimes \text{Lévy}(\lambda), \tag{1}$$

$$\text{Lévy} \sim u = t^{-\lambda}, \quad (1 < \lambda \leq 3).$$

4. Geometric Operators

The cuckoo search has been originally designed for continuous domains, while Sudokus own discrete values. Then, a discretization phase for the CS algorithm is mandatory to correctly explore the potential solutions. The discretization phase applied here has been inspired from the work reported in [8], where a particle swarm optimization algorithm is adapted to solve discrete domains. The idea relies on the use of geometric-based operators able to correctly move to

```

Input:  $Nest_{size}, \alpha, \lambda$ 
Output:  $Egg_{best}$ 
(1)  $Nests \leftarrow \text{GenerateInitialPopulation}(Nest_{size})$ 
(2) While  $\neg \text{StopCondition}$  do
(3)    $Egg_i \leftarrow \text{GetCuckooByLevyFlight}(Nests, \alpha, \lambda)$ 
(4)    $Egg_j \leftarrow \text{ChooseRandomlyFrom}(Nests)$ 
(5)   If  $\text{cost}(Egg_i) \leq \text{cost}(Egg_j)$ 
(6)      $Egg_j \leftarrow Egg_i$ 
(7)   End If
(8)    $Egg_{best} \leftarrow \text{FindCurrentBest}(Nests)$ 
(9)    $Nests \leftarrow \text{AbandonWorseNests}(p_a, Nests)$ 
(10)   $Nests \leftarrow \text{BuildNewSolutions}(Egg_{best})$ 
(11) End While
    
```

ALGORITHM 1: Cuckoo search.

```

Input:  $Parent_1[], Parent_2[]$ 
Output:  $Child[]$ 
(1)  $Child \leftarrow \text{SelectRandomSegmentFrom}(Parent_1)$ 
(2)  $List_{val} \leftarrow \text{SelectNotCopiedToChild}(Parent_2, Child)$ 
(3) For Each  $val \in List_{val}$ 
(6)    $lval \leftarrow val$ 
(4)    $V \leftarrow Parent_1[\text{IndexOf}(val, Parent_2)]$ 
(5)   If  $\text{IndexOf}(V, Parent_2) \in \text{SegmentOf}(Child)$ 
(6)      $val \leftarrow V$ 
(6)     Go To (4)
(7)   Else
(8)      $Child[\text{IndexOf}(V, Parent_2)] \leftarrow lval$ 
(9)   End If
(10) End For Each
(11)  $\text{CopyRemaining}(Child, Parent_2)$ 
    
```

ALGORITHM 2: PMX crossover.

promising regions of a discrete search space. In particular, for this work, we employ the partially matched crossover, the geometric crossover, and the feasible geometric mutation, which are described in the following.

4.1. *Partially Matched Crossover.* The partially matched crossover (PMX) basically works with two parents, creating two crossover points that are selected at random and then PMX proceeds by position swap. The PMX process is described in Algorithm 2.

At the beginning, a segment from $Parent_1$ is randomly selected and copied to the child. Then, looking in the same segment positions in $Parent_2$, each value not copied to the child is stored in a list. Then, for each value in this list, the V value is located in $Parent_1$ in the position given by the index of val in $Parent_2$. Then, an if-else conditional operates as follows: if the index of V is present in the original segment, V becomes the new val and the process goes to line 4; otherwise, $lval$ is inserted into the child in the position given by the index of V in $Parent_2$. Finally, the remaining positions from $Parent_2$ are copied to the child. The PMX operator applied to the Sudoku can be seen in Example 1.

Example 1 (PMX crossover). Let two rows located in the same position from different solutions of a Sudoku instance be the parents. The corresponding child produced by the PMX crossover is constructed as follows.

- (1) A random segment of consecutive digits from $Parent_1$ is copied to the child. Assuming that 1 corresponds to the index of the first position, the segment has size 5, from index 4 to 8:

$$\begin{array}{l}
 Parent_1: 8 \ 4 \ 7 \ 3 \ 6 \ 2 \ 5 \ 1 \ 9 \\
 Parent_2: 9 \ 1 \ 2 \ \underline{3 \ 4 \ 5 \ 6 \ 7} \ 8 \\
 Child: \ - \ - \ - \ \underline{3 \ 6 \ 2 \ 5 \ 1} \ -
 \end{array} \tag{2}$$

- (2) “4” is the first value in the observed segment of $Parent_2$ that is not present in the child. Then, $val = 4$, and the index of val in $Parent_2$ is “5.” Hence, the V value corresponds to “6.” Next, the index of V in $Parent_2$ is “7.” This index exists in the observed segment, so the process comes back to line 4 using “6” as val :

$$\begin{array}{l}
 Parent_1: 8 \ 4 \ 7 \ \underline{3 \ 6 \ 2 \ 5 \ 1} \ 9 \\
 Parent_2: 9 \ 1 \ 2 \ \underline{3 \ 4 \ 5 \ 6 \ 7} \ 8 \\
 Child: \ - \ - \ - \ \underline{3 \ 6 \ 2 \ 5 \ 1} \ -
 \end{array} \tag{3}$$

```

Input:  $Parent_1[], Parent_2[], Parent_3[]$ 
Output:  $Child[]$ 
(1)  $Mask \leftarrow InitializeMask()$ 
(1) For Each  $i \in \{1, \dots, Mask_{length}\}$ 
(2)   If  $Mask[i] = 1$ 
(3)      $Child[i] \leftarrow Parent_1[i]$ 
(4)      $Parent_2 \leftarrow Swap(Parent_1[i], Parent_2, i)$ 
(5)      $Parent_3 \leftarrow Swap(Parent_1[i], Parent_3, i)$ 
(6)   Else If  $Mask[i] = 2$ 
(7)      $Child[i] \leftarrow Parent_2[i]$ 
(8)      $Parent_1 \leftarrow Swap(Parent_2[i], Parent_1, i)$ 
(9)      $Parent_3 \leftarrow Swap(Parent_2[i], Parent_3, i)$ 
(10)  Else
(11)    $Child[i] \leftarrow Parent_3[i]$ 
(12)    $Parent_1 \leftarrow Swap(Parent_3[i], Parent_1, i)$ 
(13)    $Parent_2 \leftarrow Swap(Parent_3[i], Parent_2, i)$ 
(14)  End If
(15) End For Each

```

ALGORITHM 3: Multiparental sorting crossover.

- (3) Now, using “6” as *val*, the new *V* is “5.” Then, the index of “5” in $Parent_2$ also appears within the segment. So, the process comes back again to line 4 using “5” as *val*:

$$\begin{array}{r}
 Parent_1: 8 \ 4 \ 7 \ 3 \ 6 \ 2 \ 5 \ 1 \ 9 \\
 Parent_2: 9 \ 1 \ 2 \ \underline{3 \ 4 \ 5 \ 6 \ 7 \ 8} \\
 Child: \ - \ - \ - \ 3 \ 6 \ 2 \ 5 \ 1 \ -
 \end{array} \quad (4)$$

- (4) Then, the *V* value is “2,” and its index in $Parent_2$ does not appear within the segment. Hence, we obtain a position in the child for the value “4” from step 2:

$$\begin{array}{r}
 Parent_1: 8 \ 4 \ 7 \ 3 \ 6 \ 2 \ 5 \ 1 \ 9 \\
 Parent_2: 9 \ 1 \ 2 \ \underline{3 \ 4 \ 5 \ 6 \ 7 \ 8} \\
 Child: \ - \ - \ 4 \ 3 \ 6 \ 2 \ 5 \ 1 \ -
 \end{array} \quad (5)$$

- (5) “7” is the next value from $Parent_2$ in the segment that is not already included in the child. Then, “1” is the *V* value, whose index does not appear within the segment as well. Hence, a position for the value “7” is obtained in the child:

$$\begin{array}{r}
 Parent_1: 8 \ 4 \ 7 \ 3 \ 6 \ 2 \ 5 \ 1 \ 9 \\
 Parent_2: 9 \ 1 \ 2 \ \underline{3 \ 4 \ 5 \ 6 \ 7 \ 8} \\
 Child: \ - \ 7 \ 4 \ 3 \ 6 \ 2 \ 5 \ 1 \ -
 \end{array} \quad (6)$$

- (6) Now, everything else from $Parent_2$ is copied down to the child:

$$\begin{array}{r}
 Parent_1: 8 \ 4 \ 7 \ 3 \ 6 \ 2 \ 5 \ 1 \ 9 \\
 Parent_2: 9 \ 1 \ 2 \ \underline{3 \ 4 \ 5 \ 6 \ 7 \ 8} \\
 Child: \ 9 \ 7 \ 4 \ 3 \ 6 \ 2 \ 6 \ 1 \ 8
 \end{array} \quad (7)$$

4.2. *Multiparental Sorting Crossover.* This operator may employ multiple parents; our approach based on [8] uses three, where each one represents a row of a potential solution:

one from the best solution of all generations, one from the best solution of the current generation, and one from the current solution. Each parent is associated with a weight (*w*) according to (8) in order to control the relevance of each solution in the generation of the new one. The influence of parents given by the weights is reflected in a mask used in the process.

The multiparental crossover is described in Algorithm 3. The three parents and the mask are the input of the procedure, and the child resulting from the crossover is the output. A for each loop is used to scan the mask, where every entry indicates which parent the other two parents need to be equal to for that specific position. The replacement depends on the value of the mask and it is performed by swapping the corresponding values as indicated in the conditionals stated at lines 2, 6, and 10. The swapping process is described in Algorithm 4:

$$w_1 + w_2 + w_3 = 1, \text{ such that } w_i > 0 \quad \forall i \in \{1, 2, 3\}. \quad (8)$$

Example 2 (multiparental sorting crossover). Let $Parent_1$ be a row from the best solution of all generations, $Parent_2$ a row from the best solution of the current generation, and $Parent_3$ a row from the current solution. We employ 0.55 as the weight for $Parent_1$, 0.33 for $Parent_2$, and 0.12 for $Parent_3$. Those weights represent the percentage of appearances of the given parent within the mask. For instance, $Parent_1$ appears five times in the mask, $Parent_2$ three times, and $Parent_3$ once. The corresponding child produced by the multiparental sorting crossover is constructed as follows.

- (1) A mask of parent length is randomly generated according to the parent weights:

$$\begin{array}{r}
 Mask: \ 3 \ 1 \ 2 \ 1 \ 1 \ 1 \ 1 \ 2 \ 2 \\
 Parent_1: 8 \ 4 \ 7 \ 3 \ 6 \ 2 \ 5 \ 1 \ 9 \\
 Parent_2: 9 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \\
 Parent_3: 4 \ 7 \ 9 \ 3 \ 6 \ 2 \ 5 \ 1 \ 8
 \end{array} \quad (9)$$

```

Input: value, Parent[], pos
Output: Parent[]
(1) If Parent[pos] ≠ value
(2)   For Each i ∈ {1, ..., Parentlength}
(3)     If Parent[i] = value
(4)       aux ← Parent[i]
(5)       Parent[i] ← Parent[pos]
(6)       Parent[pos] ← aux
(7)     End If
(8)   End For Each
(9) End If
    
```

ALGORITHM 4: Swap.

```

Input: initSol[[]], sol[[]]
Output: sol[[]]
(1) rows ← ChooseRowsRandomly(initSol)
(2) For Each row ∈ rows
(3)   pos1, pos2 ← ChooseTwoEmptyCells(row)
(4)   aux ← sol[row][pos1]
(5)   sol[row][pos1] ← sol[row][pos2]
(6)   sol[row][pos2] ← aux
(7) End For Each
    
```

ALGORITHM 5: Feasible geometric mutation.

(2) The first value of the mask corresponds to parent “3”, and then the first value of $Parent_1$ and $Parent_2$ needs to be equal to the first value of $Parent_3$, which is “4.” To this end, in $Parent_1$ and $Parent_2$, the first cell is swapped with the cell that holds the value “4”:

$$\begin{array}{r}
 \text{Mask: } 3 \ 1 \ 2 \ 1 \ 1 \ 1 \ 2 \ 2 \ 3 \\
 \text{Parent}_1: \underline{4} \ 8 \ 7 \ 3 \ 6 \ 2 \ 5 \ 1 \ 9 \\
 \text{Parent}_2: \underline{4} \ 1 \ 2 \ 3 \ 9 \ 5 \ 6 \ 7 \ 8 \\
 \text{Parent}_3: 4 \ 7 \ 9 \ 3 \ 6 \ 2 \ 5 \ 1 \ 8 \\
 \hline
 \text{Child} \ 4
 \end{array} \tag{10}$$

(3) Next, the second value from the mask is “1.” The swapping process is analogous:

$$\begin{array}{r}
 \text{Mask: } 3 \ 1 \ 2 \ 1 \ 1 \ 1 \ 2 \ 2 \ 3 \\
 \text{Parent}_1: 4 \ 8 \ 7 \ 3 \ 6 \ 2 \ 5 \ 1 \ 9 \\
 \text{Parent}_2: 4 \ 8 \ 2 \ 3 \ 9 \ 5 \ 6 \ 7 \ \underline{1} \\
 \text{Parent}_3: 4 \ 8 \ 9 \ 3 \ 6 \ 2 \ 5 \ 1 \ \underline{7} \\
 \hline
 \text{Child} \ 4 \ 8
 \end{array} \tag{11}$$

(4) Following the same procedure, the last step is shown below obtaining 4 8 2 3 6 7 9 5 1 as the new child:

$$\begin{array}{r}
 \text{Mask: } 3 \ 1 \ 2 \ 1 \ 1 \ 1 \ 2 \ 2 \ 3 \\
 \text{Parent}_1: 4 \ 8 \ 2 \ 3 \ 6 \ 7 \ 9 \ 5 \ 1 \\
 \text{Parent}_2: 4 \ 8 \ 2 \ 3 \ 6 \ 7 \ 9 \ 5 \ 1 \\
 \text{Parent}_3: 4 \ 8 \ 2 \ 3 \ 6 \ 7 \ 9 \ 5 \ 1 \\
 \hline
 \text{Child} \ 4 \ 8 \ 2 \ 3 \ 6 \ 7 \ 9 \ 5 \ 1
 \end{array} \tag{12}$$

4.3. Feasible Geometric Mutation. This is a simple operator used to maintain diversity in the solutions. It swaps two non-fixed elements in a row guaranteeing that mutation is applied only over the cells with no given value. The procedure is described in Algorithm 5.

Example 3 (feasible geometric mutation). Let us consider a given Sudoku row and a solution candidate row, as shown below:

$$\begin{array}{l}
 \text{Sudoku problem row: } \quad - \ - \ - \ 3 \ \ - \ \ - \ \ - \ \ - \ \ - \\
 \text{Solution candidate row: } 9 \ 7 \ 4 \ 3 \ 6 \ 2 \ 5 \ 1 \ 8
 \end{array} \tag{13}$$

The mutation is allowed in any cell except for cell 4, which owns the value 3 as given for the Sudoku instance. Examples of allowed and forbidden mutations are depicted below:

$$\begin{array}{l}
 \text{allowed mutation: } 9 \ 6 \ 4 \ 3 \ 7 \ 2 \ 5 \ 1 \ 8 \\
 \text{forbidden mutation: } 9 \ 7 \ 4 \ 4 \ 6 \ 2 \ 5 \ 1 \ 8
 \end{array} \tag{14}$$

5. Prefiltering Phase

In the presence of unfeasible solutions, the cuckoo procedure is responsible for detecting and discarding them in order to conduct the search to feasible regions of the space of solutions. The goal of the prefiltering phase is to alleviate the work of the cuckoo algorithm by previously eliminating those unfeasible values. This is possible by representing the Sudoku as a constraint network [4] and then applying efficient filtering techniques from the constraint satisfaction domain. In this context, arc-consistency [2] is a widely employed

local consistency for filtering algorithms. Arc-consistency was initially defined for binary constraint [21, 22]. We employ here the more general filtering for nonarbitrary constraints named generalized arc-consistency (GAC). The idea is to enforce a local consistency to the problem in a process called constraint propagation. Before detailing this process, let us introduce some necessary definitions [23].

Definition 4 (constraint). A constraint c is a relation defined on a sequence of variables $X(c) = (x_{i_1}, \dots, x_{i_{|X(c)|}})$, called the scheme of c ; c is the subset of $\mathbb{Z}^{|X(c)|}$ that contains the combinations of tuples $\tau \in \mathbb{Z}^{|X(c)|}$ that satisfy c . $|X(c)|$ is called the arity of c . A constraint c with scheme $X(c) = (x_1, \dots, x_k)$ is also noted as $c(x_1, \dots, x_k)$.

Definition 5 (constraint network). A constraint network also known as constraint satisfaction problem (CSP) is defined by a triple $N = \langle X, D, C \rangle$, where

- (i) X is a finite sequence of integer variables $X = (x_1, \dots, x_n)$;
- (ii) D is the corresponding set of domains for X ; that is, $D = D(x_1) \times \dots \times D(x_n)$, where $D(x_i) \subset \mathbb{Z}$ is the finite set of values that variable x_i can take;
- (iii) C is a set of constraints $C = \{c_1, \dots, c_e\}$, where variables in $X(c_i)$ are in X .

Example 6 (the Sudoku as a constraint network). Let $\langle X, D, C \rangle$ be the constraint network, which is composed of the following.

- (i) $X = (x_{1,1}, \dots, x_{n,m})$ is the sequence of variables, and $x_{i,j} \in X$ identifies the cell placed in the i th row and j th column of the Sudoku matrix, for $i = 1, \dots, n$ and $j = 1, \dots, m$.
- (ii) D is the corresponding set of domains, where $D(x_{i,j}) \in D$ is the domain of the variable $x_{i,j}$.
- (iii) C is the set of constraints defined as follows.
 - (a) To ensure that values are different in rows and columns, $x_{k,i} \neq x_{k,j} \wedge x_{i,k} \neq x_{j,k}$, for all $(k \in [1, 9], i \in [1, 9], j \in [i + 1, 9])$.
 - (b) To ensure that values are different in subgrid: $x_{(k1-1)*3+k2,(j1-1)*3+j2} \neq x_{(k1-1)*3+k3,(j1-1)*3+j3}$, for all $(k1, j1, k2, j2, k3, j3 \in [1, 3] \mid k2 \neq k3 \wedge j2 \neq j3)$.

Definition 7 (projection). A projection of c on Y is denoted as $\pi_{Y(c)}$, which defines the relation with scheme Y that contains the tuples that can be extended to a tuple on $X(c)$ satisfying c .

Definition 8 ((generalized) arc-consistency). Given a network $N = \langle X, D, C \rangle$, a constraint $c \in C$, and a variable $x_i \in X(c)$,

- (i) a value $v_i \in D(x_i)$ is consistent with $c \in D$ if and only if there exists a valid tuple τ satisfying c such

that $v_i = \tau[\{x_i\}]$; such a tuple is called a support for (x_i, v_i) on c ;

- (ii) the domain D is (generalized) arc-consistent on c for x_i if and only if all the values in $D(x_i)$ are consistent with c in D ; that is, $D(x_i) \subseteq \pi_{x_i}(c \cap \pi_{X(c)}(D))$;
- (iii) the network N is (generalized) arc-consistent if and only if D is (generalized) arc-consistent for all variables in X on all constraints in C .

The filtering process is achieved by enforcing the arc-consistency on the problem. This can be carried out by using Algorithms 6 and 7. The idea is to revise the arcs (the constraint relation between variables) by removing the values from $D(x_i)$ that lead to inconsistencies with respect to a given constraint. Such a revision process is done by Algorithm 6, which takes each value $v_i \in D(x_i)$ (line 2) and analyses the space $\tau \in c \cap \pi_{X(c)}(D)$, searching for a support on constraint c (line 3). If support does not exist, the value v_i is eliminated from $D(x_i)$. Finally, the procedure informs if the domain has been modified by returning the corresponding Boolean value (line 8).

The role of Algorithm 7 is to guarantee that every domain is arc-consistent. This is done by iteratively revising the arcs by performing calls to Algorithm 6. At the beginning, a list called Q is filled with pairs (x_i, c) such that $x_i \in X(c)$. Pairs for which $D(x_i)$ is not ensured to be arc-consistent are kept in order to avoid useless calls to Algorithm 7. This is a main advantage of AC3 with respect to its predecessor Algorithms AC1 and AC2. Then, at line 2, a while statement controls the calls to `Revise3`. If a true value is received from `Revise3`, $D(x_i)$ is verified and if no value remains within the domain, the algorithm returns false. If there still exist values in $D(x_i)$, normally, a value for another variable x_j has lost its support on c . Hence, the list Q must be refilled with all pairs (x_j, c) . The process finishes and returns true when all remaining values within domains are arc-consistent with respect to all constraints.

Example 9 (enforcing AC3 on a Sudoku puzzle). Let us exemplify the work of the prefiltering phase by enforcing the AC3 on three constraints of a Sudoku instance. We begin by enforcing the AC3 on a given subgrid (enclosed with dashed lines in Figure 2) of the Sudoku puzzle. The subgrid has three variables with no value assigned ($x_{4,9}$, $x_{5,8}$, and $x_{6,9}$). Then, enforcing AC3 via the GAC3 algorithm with respect to the subgrid constraint (second constraint from Example 6) leads to the elimination of six values from $D(x_{4,9})$, $D(x_{5,8})$, and $D(x_{6,9})$. Those values have no support on the verified constraint; that is, they have been already taken for another cell on the same subgrid. Thus, the original domains for the three variables are reduced to $\{5, 6, 8\}$.

In Figure 3, the AC3 is enforced to a row of the puzzle. This row has four variables ($x_{5,2}$, $x_{5,4}$, $x_{5,6}$, and $x_{5,8}$) with no assigned value. The GAC3 algorithm filters from domains five values with no support from $D(x_{5,2})$, $D(x_{5,4})$, and $D(x_{5,6})$. The variable $x_{5,8}$ has been refiltered, remaining only two

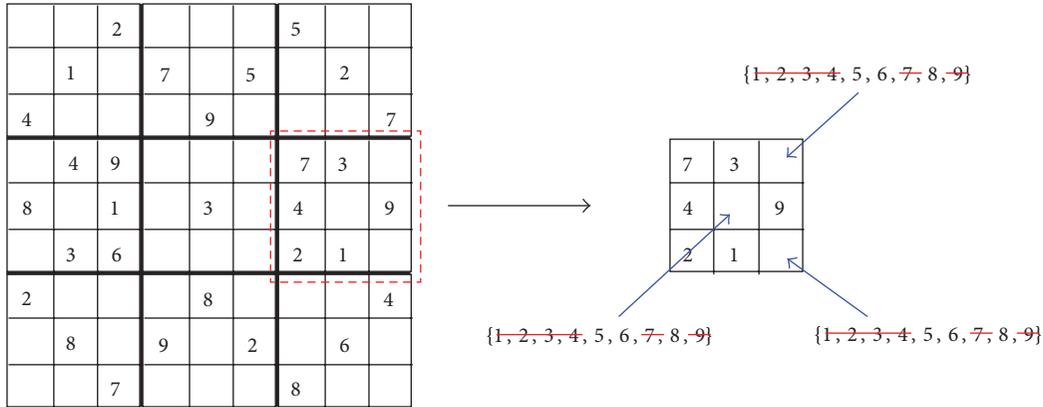


FIGURE 2: Enforcing AC3 to a subgrid constraint.

```

Input:  $x_i, c$ 
Output: CHANGE
(1) CHANGE  $\leftarrow$  false
(2) Foreach  $v_i \in D(x_i)$  do
(3)   If  $\nexists \tau \in c \cap \pi_{X(c)}(D)$  with  $\tau[x_i] = v_i$  do
(4)     remove  $v_i$  from  $D(x_i)$ 
(5)     CHANGE  $\leftarrow$  true
(6)   End If
(7) End Foreach
(8) Return CHANGE
    
```

ALGORITHM 6: Revise3.

```

Input:  $X, D, C$ 
Output: Boolean
(1)  $Q \leftarrow \{ (x_i, c) \mid c \in C, x_i \in X(c) \}$ 
(2) While  $Q \neq \emptyset$  do
(3)   select and remove  $(x_i, c)$  from  $Q$ 
(4)   If Revise3( $x_i, c$ ) then
(5)     If  $D(x_i) = \emptyset$  then
(6)       Return false
(7)     Else
(8)        $Q \leftarrow Q \cup \{ (x_j, c') \mid c' \in C \wedge c' \neq c \wedge x_i, x_j \in X(c') \wedge j \neq i \}$ 
(9)     End If
(10)  End If
(11) End While
(12) Return true
    
```

ALGORITHM 7: AC3/GAC3.

possible values. Finally, in Figure 4, four values are filtered from four variables, remaining only one possible value for variable $x_{5,8}$.

6. The Prefiltered Cuckoo Search via Geometric Operators

The cuckoo search proposed here combines geometric operators with prefiltering phases. The goal is to enhance the performance of the cuckoo search as well as to allow it

to correctly explore a discrete search space. Algorithm 5 illustrates the new hybrid algorithm. Now the input set is quite larger. It considers the size of the nest, the constraint network $\langle X, D, C \rangle$ representing the Sudoku problem, and two parameters that define probabilities for regulating the usage of geometric operators. As output, the procedure returns the best egg reached by the algorithm. The pre-filtering phase via the AC3 algorithm is triggered at the beginning. The AC3 algorithm receives as input the constraint network $\langle X, D, C \rangle$ of the Sudoku and reduces if possible the

```

Input:  $Nest_{size}, X, D, C, P_{pmx-multi}, P_{mutate}$ 
Output:  $Egg_{best}$ 
(1)  $D \leftarrow AC3(X, D, C)$ 
(2)  $Nests \leftarrow GenerateInitialPopulation(Nest_{size}, D)$ 
(3) While  $\neg StopCondition$  do
(4)    $Egg_i \leftarrow GetRandomlyFrom(Nests)$ 
(5)    $Egg_i \leftarrow GeometricOperators(P_{pmx-multi}, P_{mutate})$ 
(6)    $Egg_j \leftarrow ChooseRandomlyFrom(Nests)$ 
(7)   If  $cost(Egg_i) \leq cost(Egg_j)$ 
(8)      $Egg_j \leftarrow Egg_i$ 
(9)   End If
(10)   $Egg_{best} \leftarrow FindCurrentBest(Nests)$ 
(11)   $Nests \leftarrow AbandonWorseNests(p_a, Nests)$ 
(12)   $Nests \leftarrow BuildNewSolutions(Egg_{best}, D)$ 
(13) End While

```

ALGORITHM 8: Prefiltered discrete cuckoo search.

```

Input:  $Egg, Egg_{best}, Egg_{lastbest}, P_{pmx-multi}, P_{mutate}$ 
Output:  $Egg$ 
(1) Foreach  $Egg \in Nests$ 
(2)   If  $(Egg \neq Egg_{best})$ 
(3)     If  $(rand() < P_{pmx-multi})$ 
(4)        $Egg \leftarrow PMXCrossover(Egg, Egg_{best})$ 
(5)     Else
(6)        $Egg = MultiParentalSortingCrossover(Egg, Egg_{best}, Egg_{lastbest})$ 
(7)     End If
(8)     If  $(rand() < P_{mutate})$ 
(9)        $Egg \leftarrow FeasibleGeometricMutation(D, Egg)$ 
(10)    End If
(11)  End If
(12) End Foreach

```

ALGORITHM 9: Geometric operators.

set of domains D by deleting the unfeasible values. Then, an initial population of nests is generated but, unlike the classic cuckoo, the generation is bounded to the reduced set of domains D . Between lines 3 and 13, a while loop manages the iteration process until the stop condition is reached, which for the current implementation corresponds to a maximum number of iterations. At line 4, an Egg_i is randomly chosen from nests to which the geometric operators are then applied. The usage of geometric operators is illustrated in Algorithm 9, where they apply only whether the evaluated egg is not the best one. The PMX and multi-parent sorting crossovers act depending on a random value and on the $P_{pmx-multi}$ probability. Analogously, the mutation operates using the P_{mutate} probability, and D is used as input of the mutation to validate that only feasible mutations are carried out. At line 6, an Egg_j is randomly chosen to be then compared with the previous one in order to keep the one exhibiting the best cost. The cost of a solution corresponds to the sum of wrong values in subgrids, columns, and rows (see Figure 5). At line 11, the worse nests are abandoned depending on probability p_a . Finally, new solutions are built, but again, the set of filtered domains D is considered in order to avoid

unfeasible solutions. Algorithm 8 illustrates the prefiltered discrete cuckoo search.

7. Experimental Results

Different experiments have been performed in order to validate our approach. The Sudoku benchmarks used have been taken from [24], which are organized in three difficulty levels: easy, medium, and hard. All tested Sudokus have a unique solution. The algorithms have been implemented in Octave 3.6.3, and the experiments have been performed on a 2.0 GHz Intel Core2 Duo T5870 with 1 Gb RAM running Fedora 17. The configuration of the proposed cuckoo search is the following, which corresponds to the best one achieved after a tuning phase: $P_a = 0.25$, $P_{pmx-multi} = 0.7$, $P_{mutate} = 0.9$, and $Nest_{size} = 10$.

Table 1 illustrates the results of solving 9 problems, 3 from each difficulty level, by using the proposed cuckoo search algorithm considering 10000 iterations. From left to right, the table states the number of tries performed, the number of tries solved, the minimum solving time reached, the average

TABLE 1: Solving Sudokus with the prefiltered discrete cuckoo search considering 10000 iterations.

	Tries	Solved	Min. solving time (sec)	\bar{x} solving time (sec)	Max. solving time (sec)	SD time (sec)
Easy a	50	50	1.302	1.310	1.329	0.004
Easy b	50	50	1.005	1.007	1.029	0.003
Easy c	50	50	1.019	1.021	1.036	0.003
Medium a	50	50	2.501	384.341	1171.312	291.336
Medium b	84	50	38.32	729.673	1932.691	516.867
Medium c	67	50	26.807	800.471	1923.442	561.877
Hard a	97	50	385.652	2059.690	3777.340	984.921
Hard b	62	50	49.608	1484.692	6810.882	1473.731
Hard c	71	50	9.497	771.211	3561.042	825.516

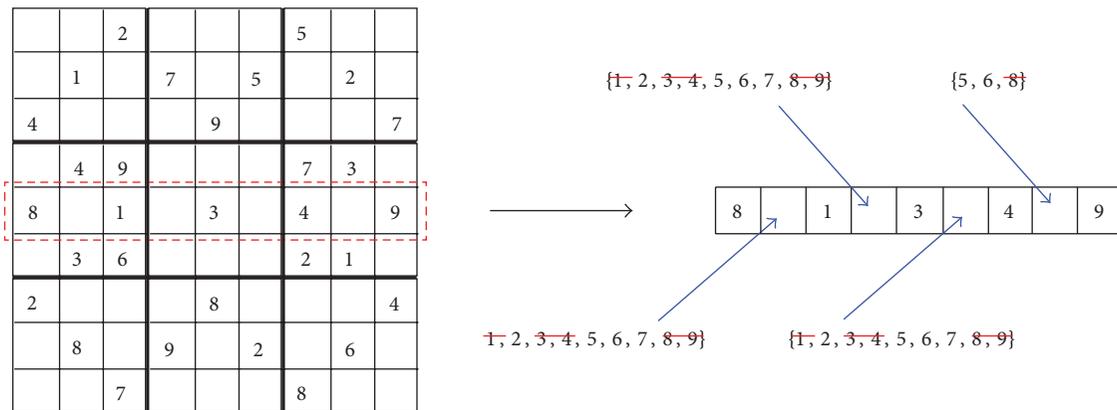


FIGURE 3: Enforcing AC3 to a row constraint.

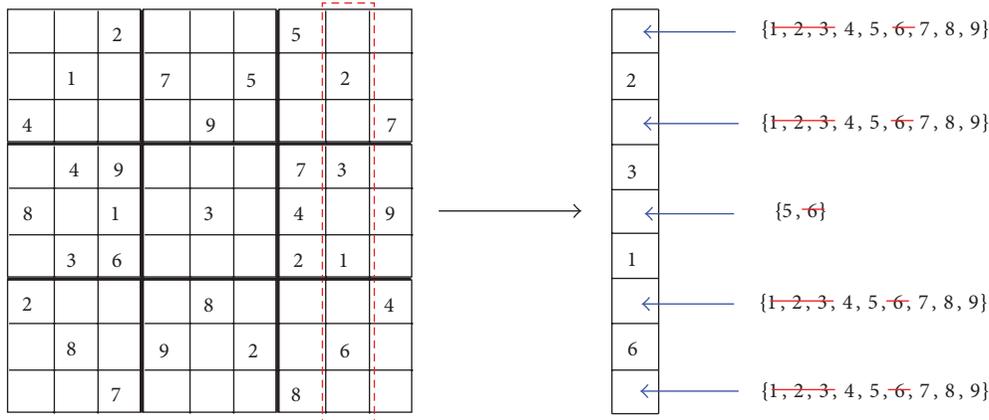


FIGURE 4: Enforcing AC3 to a column constraint.

solving time, the maximum solving time, and the standard deviation (SD). Regarding the easy Sudokus, the proposed cuckoo search is able to rapidly solve them reaching a 100% of success (50 out of 50 tries solved). Then, increasing one difficulty level, the percentage of success shortly diminishes, keeping the 100% of success for the “medium a” benchmark. Finally, for hard Sudokus, the runtime naturally gets bigger (see Figure 6); however, the performance is reasonable,

reaching a 51% (50 out of 97 tries solved) of success for “hard a,” 80% (50 out of 62 tries solved) of success for “hard b,” and a 70% (50 out of 71 tries solved) of success for “hard c.”

In Table 2, the performance of the proposed cuckoo search is compared with the best-performing incomplete methods reported in the literature: a genetic algorithm (GA) [1] and a hybrid tabu search (hybrid TS) [20]. We contrast the number of problems solved from a total of 30 tries

TABLE 2: Comparing the prefiltered discrete cuckoo search with the best-performing incomplete methods for Sudokus considering 100000 and unlimited iterations.

Problem	Prefiltered discrete CS		Hybrid TS		GA	
	Unlimited iterations	100000 iterations	Unlimited iterations	100000 iterations	Unlimited iterations	100000 iterations
Easy a	30	30	30	30	30	29
Easy b	30	30	30	30	30	30
Easy c	30	30	30	30	30	30
Medium a	30	30	30	30	30	10
Medium b	30	30	30	30	—	—
Medium c	30	30	30	30	—	—
Hard a	30	30	30	30	30	2
Hard b	30	30	30	30	—	—
Hard c	30	30	30	30	—	—

9	7	2	8	6	3	5	4	1
6	1	8	7	4	5	9	2	3
4	5	5	2	9	1	6	8	7
5	4	9	1	2	8	7	3	6
8	2	1	6	3	7	4	5	9
3	3	6	4	5	9	2	1	8
2	9	5	3	8	6	1	7	4
1	8	4	9	7	2	3	6	5
3	6	7	5	1	4	8	9	2

Cost 6

9	7	2	8	6	3	5	4	1
6	1	8	7	4	5	9	2	3
4	5	3	2	9	1	6	8	7
5	4	9	1	2	8	7	3	6
8	2	1	6	3	7	4	5	9
7	3	6	4	5	9	2	1	8
2	9	5	3	8	6	1	7	4
1	8	4	9	7	2	3	6	5
3	6	7	5	1	4	8	9	2

Cost 0

FIGURE 5: Solution cost of a Sudoku puzzle.

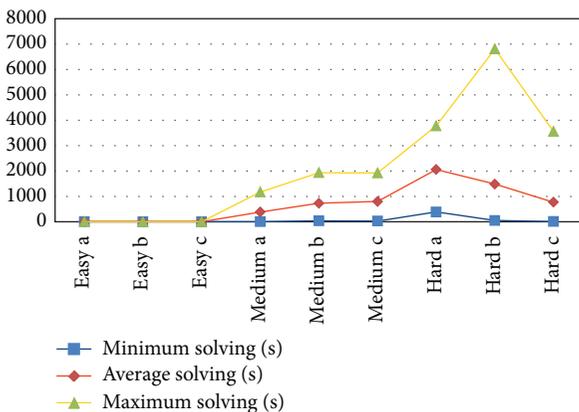


FIGURE 6: Comparing solving times for Sudoku.

taking into account both unlimited iterations and 100000 iterations. The results depict that the three techniques are able to easily succeed for the first Sudoku level. In the presence of medium Sudokus, the GA begins to decrease its performance, being able to solve a medium Sudoku with a 33% of success

(10 out of 30 tries solved). The cuckoo search and the hybrid TS keep their 100% of success. Finally, observing hard Sudokus, the performance of the cuckoo search and the hybrid TS is considerably better than GA, which solves only 2 out of 30 tries, while our proposal as well as the hybrid TS is able to reach a 100% of success.

8. Conclusions and Future Work

In this paper, we have presented a prefiltered cuckoo search algorithm tuned with geometric operators. The geometric operators allow one to drive the search to promising regions of a discrete space of solutions, while the prefiltering phase attempts to previously delete from domains the values that do not conduct to any feasible solution. In this way, the work of the metaheuristic is alleviated leading to a faster solving process. We have performed a set of experiments in order to compare our approach with the best-performing approximate methods reported in the literature. We have considered Sudokus from different difficulty levels: easy, medium, and hard. In the presence of easy Sudokus, the proposed cuckoo search is able to reach a 100% of success.

When solving medium difficulty Sudokus, the cuckoo search keeps its 100% of success, while the best GA reported is able only to solve 10 out of 30 tries. Finally, regarding hard Sudokus, the cuckoo search noticeably competes against the best incomplete method reported for Sudokus, both reaching a 100% of success considering 100000 iterations.

We visualize different directions for future work; perhaps the clearest one is the introduction of prefiltering phases to additional metaheuristics such as particle swarm optimization, ant, or bee colony algorithms to solve Sudokus or any combinatorial problem. Evaluating the behaviour of geometric operators in other swarm-based metaheuristics is another interesting work to develop. Finally, the use of autonomous search [25–28] for the self-tuning of a metaheuristic interacting with prefiltering phases will be also an appealing research direction to pursue.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Ricardo Soto is supported by Grant CONICYT/FONDECYT/INICIACION/11130459, Broderick Crawford is supported by Grant CONICYT/FONDECYT/REGULAR/1140897, and Fernando Paredes is supported by Grant CONICYT/FONDECYT/REGULAR/1130455.

References

- [1] T. Mantere and J. Koljonen, "Solving, rating and generating sudoku puzzles with GA," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 1382–1389, IEEE Computer Society, Singapore, September 2007.
- [2] F. Rossi, P. van Beek, and T. Walsh, *Handbook of Constraint Programming*, Elsevier, 2006.
- [3] T. K. Moon and J. H. Gunther, "Multiple constraint satisfaction by belief propagation: an example using sudoku," in *Proceedings of the IEEE Mountain Workshop on Adaptive and Learning Systems*, pp. 122–126, Logan, Utah, USA, July 2006.
- [4] H. Simonis, "Sudoku as a constraint problem," in *Proceedings of the 4th International Workshop on Modelling and Reformulating Constraint Satisfaction Problems*, pp. 13–27, Barcelona, Spain, 2005.
- [5] I. Lynce and J. Ouaknine, "Sudoku as a SAT problem," in *Proceedings of the International Symposium on Artificial Intelligence and Mathematics (ISAIA '06)*, Fort Lauderdale, Fla, USA, 2006.
- [6] A. Moraglio, J. Togelius, and S. Lucas, "Product geometric crossover for the sudoku puzzle," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 470–476, IEEE Computer Society, Vancouver, Canada, July 2006.
- [7] R. Lewis, "Metaheuristics can solve sudoku puzzles," *Journal of Heuristics*, vol. 13, no. 4, pp. 387–401, 2007.
- [8] A. Moraglio and J. Togelius, "Geometric particle swarm optimization for the sudoku puzzle," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 118–125, ACM Press, July 2007.
- [9] T. Mantere and J. Koljonen, "Solving and analyzing sudokus with cultural algorithms," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 4053–4060, IEEE Computer Society, Hong Kong, June 2008.
- [10] M. Asif and R. Baig, "Solving NP-complete problem using ACO algorithm," in *Proceedings of the International Conference on Emerging Technologies (ICET '09)*, pp. 13–16, IEEE Computer Society, Islamabad, Pakistan, October 2009.
- [11] T. K. Moon, J. H. Gunther, and J. J. Kupin, "Sinkhorn solves sudoku," *IEEE Transactions on Information Theory*, vol. 55, no. 4, pp. 1741–1746, 2009.
- [12] G. Santos-García and M. Palomino, "Solving sudoku puzzles with rewriting rules," *Electronic Notes in Theoretical Computer Science*, vol. 176, no. 4, pp. 79–93, 2007.
- [13] J. Gunther and T. K. Moon, "Entropy minimization for solving sudoku," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 508–513, 2012.
- [14] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, IEEE, Coimbatore, India, December 2009.
- [15] X.-S. Yang and S. Deb, "Multiobjective cuckoo search for design optimization," *Computers & Operations Research*, vol. 40, no. 6, pp. 1616–1624, 2013.
- [16] P. R. Srivastava, A. Varshney, P. Nama, and X.-S. Yang, "Software test effort estimation: a model based on cuckoo search," *International Journal of Bio-Inspired Computation*, vol. 4, no. 5, pp. 278–285, 2012.
- [17] M. K. Marichelvam, "An improved hybrid cuckoo search (ihcs) metaheuristics algorithm for permutation flow shop scheduling problems," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 200–205, 2012.
- [18] A. Gherboudj, A. Layeb, and S. Chikhi, "Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 229–236, 2012.
- [19] B. Crawford, M. Aranda, C. Castro, and E. Monfroy, "Using constraint programming to solve sudoku puzzles," in *Proceedings of the 3rd International Conference on Convergence and Hybrid Information Technology (ICCHIT '08)*, pp. 926–931, IEEE Computer Society, Busan, Republic of Korea, November 2008.
- [20] R. Soto, B. Crawford, C. Galleguillos, E. Monfroy, and F. Paredes, "A hybrid ac3-tabu search algorithm for solving sudoku puzzles," *Expert Systems with Applications*, vol. 40, no. 15, pp. 5817–5821, 2013.
- [21] A. K. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, no. 1, pp. 99–118, 1977.
- [22] A. Mackworth, "On reading sketch maps," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI '77)*, pp. 598–606, 1977.
- [23] C. Bessière, *Handbook of Constraint Programming*, Elsevier, 2006.
- [24] T. Mantere and J. Koljonen, "Sudoku research page," 2008, <http://lipas.uwasa.fi/~timan/sudoku/>.
- [25] B. Crawford, R. Soto, E. Monfroy, W. Palma, C. Castro, and F. Paredes, "Parameter tuning of a choice-function based hyperheuristic using Particle Swarm Optimization," *Expert Systems with Applications*, vol. 40, no. 5, pp. 1690–1695, 2013.
- [26] E. Monfroy, C. Castro, B. Crawford, R. Soto, F. Paredes, and C. Figueroa, "A reactive and hybrid constraint solver," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 25, no. 1, pp. 1–22, 2013.

- [27] R. Soto, B. Crawford, E. Monfroy, and V. Bustos, "Using autonomous search for generating good enumeration strategy blends in constraint programming," in *Proceedings of the 12th International Conference on Computational Science and Its Applications (ICCSA '12)*, vol. 7335 of *Lecture Notes in Computer Science*, pp. 607–617, Springer, Berlin, Germany, 2012.
- [28] B. Crawford, R. Soto, C. Castro, and E. Monfroy, "Extensible cp-based autonomous search," in *Proceedings of the HCI International*, vol. 173 of *Communications in Computer and Information Science*, pp. 561–565, Springer, Berlin, Germany, 2011.

Research Article

Self-Adaptive MOEA Feature Selection for Classification of Bankruptcy Prediction Data

A. Gaspar-Cunha,¹ G. Recio,² L. Costa,³ and C. Estébanez²

¹ *Institute of Polymers and Composites-I3N, University of Minho, Guimarães, Portugal*

² *Department of Computer Science, Universidad Carlos III de Madrid, Leganes, Madrid, Spain*

³ *Department of Production and Systems Engineering, University of Minho, Braga, Portugal*

Correspondence should be addressed to G. Recio; grecio@inf.uc3m.es

Received 1 October 2013; Accepted 25 December 2013; Published 23 February 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 A. Gaspar-Cunha et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Bankruptcy prediction is a vast area of finance and accounting whose importance lies in the relevance for creditors and investors in evaluating the likelihood of getting into bankrupt. As companies become complex, they develop sophisticated schemes to hide their real situation. In turn, making an estimation of the credit risks associated with counterparts or predicting bankruptcy becomes harder. Evolutionary algorithms have shown to be an excellent tool to deal with complex problems in finances and economics where a large number of irrelevant features are involved. This paper provides a methodology for feature selection in classification of bankruptcy data sets using an evolutionary multiobjective approach that simultaneously minimise the number of features and maximise the classifier quality measure (e.g., accuracy). The proposed methodology makes use of self-adaptation by applying the feature selection algorithm while simultaneously optimising the parameters of the classifier used. The methodology was applied to four different sets of data. The obtained results showed the utility of using the self-adaptation of the classifier.

1. Introduction

Bankruptcy prediction has become an important economic phenomenon [1, 2]. The high individual, economical, and social costs arising from bankruptcies have motivated further effort in understanding the problem and finding better prediction methods. In finances, bankruptcy prediction is an important topic of research as it provides a way of identifying business failure, that is, situations in which a firm or particular cannot pay lenders, preferred stock shareholders, suppliers, and so forth. An organisation which is unable to meet its scheduled payments when estimations of future cash show that the current financial situation will not change in the near future is said to undergo into financial distress. Signs of financial distress are evident long before bankruptcy occurs. Research in bankruptcy prediction started in [3] where a univariate discriminant model was used. This was followed by studies using traditional statistical methods which include correlation, regression, logistic models, and factor analysis [4, 5]. More recently, an overview of the classic statistical

divided them into four types: univariate analysis, risk index models, multivariate discriminant analysis, and conditional probability models [6].

Modern bankruptcy prediction models combine both statistical analysis and artificial intelligence techniques improving then the decision support tools and decision making [7–9]. In this manner, back propagation artificial neural networks have been applied to bankruptcy prediction [10] whose results revealed better accuracy than predictions made using some other techniques (recursive partitioning, *k*-nearest neighbours, C4.5, etc.). Consequently, research has focused on the combination of artificial neural networks with other soft computing tools such as fuzzy sets, genetic programming, ant colony optimisation, or particle swarm optimisation [11–14].

Support vector machines (SVMs) have been largely used for classification and pattern recognition applications. SVMs are a family of generalised linear classifiers widely used for classification of financial data. In particular, several studies have been published on the application of SVMs to

the problem of bankruptcy prediction [15–18]. A survey on support vector machines applied to the problem of bankruptcy prediction can be found in [19]. It is worth mentioning that support vector machines require solving a quadratic programming problem which is time consuming when considering large dimensional problems and also that it requires the optimisation of algorithm parameters which may affect its performance. The aim behind this research is to overcome the above limitations which will be accomplished by using feature selection and self-adaptation of the classification algorithm parameters.

Feature selection can be described as one of the initial stages of a classification process by which the complexity of the problem is reduced by elimination of irrelevant features [20]. Feature selection must be approached with the minor loss of information of the original set after the noisy or irrelevant features are removed; that is, the elimination of irrelevant features should not reduce the overall classification accuracy. Being X the original set of n features for a given classification task, the continuous feature selection problem consists in assigning weights w_i to each feature $x_i \in X$ in such a way that the order corresponding to its theoretical relevance is preserved. In a similar way, the binary feature selection problem refers to the assignment of binary weights that leads to a reduced subset $X' \subseteq X$ of m features (with $m < n$). In the general case, all features take part in the learning process, each one with a particular contribution. In binary feature selection, only a subset of the features is considered in the learning process for which all of them contribute in the same manner. For the purpose of this work, binary feature selection will be used. In [21], the problem of binary feature selection was formally defined, which, for the general case, consists in finding a compromise between minimising the number of features in X' and maximising an evaluation measure over the subset $J(X')$. Notice that an optimal subset of features is not necessarily unique which has motivated further research into this field. Also, there are many potential benefits of feature selection [22], that is, facilitating data visualisation and understanding, reducing the measurement and storage requirements, reducing training and using times, and so forth. Traditional feature selection methods used in bankruptcy prediction consist on applying statistical methods, such as t -test, correlation matrix, stepwise regression, principal component analysis, or factor analysis to examine their prediction performance [23]. The application of artificial intelligence techniques, such as evolutionary computation, to the problem of feature selection is now emerging in order to enhance the effectiveness of traditional methods [20].

The general case for feature selection fits into a multi-objective optimisation approach where the aim is to simultaneously optimise two or more conflicting objectives. In addition, identifying a set of solutions representing the best possible trade-offs among objectives of the problem instead of a single solution might be of interest in many cases. Within this context, evolutionary algorithms constitute a preferred choice as they simultaneously deal with a set of solutions, referred to as population, which allows several different solutions to be generated in a single run. Several evolutionary

multi-objective approaches (MOEAs) have been applied to finances and economics. The most popular application of MOEAs in the literature deals with the portfolio optimisation problem [24–26], although MOEAs have also been successfully applied to stock ranking [27], risk-return analysis [28, 29], and economic modelling [30, 31]. In a sense, this work constitutes a study on the consequences of simultaneously optimised two or three objective functions over real-world benchmark problems.

Another issue that will be considered in this work is the self-adaptation of the classifier algorithm parameters. Self-adaptation aims at finding suitable adjustment of the algorithm parameters efficiently [32]. In general, the definition of self-adaptation in evolutionary algorithms refers to the adjustment of control parameters that are related to evolutionary routines [33], that is, mutation or crossover rates, population size, and selection strategy. In this work, the scope of this definition will be modified and the aim will be the automatic adjustment of the classification process parameters, which in the present case include the training method, the training fraction, and the specific SVM parameters (e.g., kernel and regularisation parameters). Some other recent works that might be of interest for the reader are [34–38].

The aim of this work is to further investigate into the feature selection problem in bankruptcy prediction using a multi-objective approach, including self-adaptation of the classification algorithm parameters. This work is expected to contribute by introducing a novel multi-objective methodology for feature selection which provides a solution to the problem of bankruptcy prediction compromising both the minimisation of the number of features selected and the maximisation/minimisation of a quality measure of the classifier, for example, accuracy or error. Also, this paper will help to create a better understanding of the application of SVMs to real-world data. The proposed methodology will be validated using bankruptcy prediction datasets found in the literature.

The remaining of this paper is organised as follows. The proposed methodology will be described in detail in Section 2, for which the corresponding expertise areas of classification, using SVMs, feature selection in classification and multi-objective evolutionary optimisation will be introduced. Section 3 describes the datasets used during the experimental part of this research. Discussion on the performance of the algorithm will follow in Section 4. The paper finalises in Section 5 by pointing out the main contributions, limitations and further extensions to this work.

2. Multiobjective Feature Selection

2.1. Feature Selection. As stated above, the feature selection problem consists in finding the minimum number of features that are necessary to evaluate correctly a set of data. Considering X as the original set of features with a cardinality $|X| = n$, the following definition applies [39].

Definition 1 (feature selection). Let $J(X')$ be an evaluation measure to be optimised (considering here a maximisation

		Actual value	
		P	N
Prediction outcome	P'	TP	FP
	N'	FN	TN

FIGURE 1: Confusion matrix.

problem, without loss of generality) defined as $J : X' \subseteq X \rightarrow \mathbb{R}$. The selection of a feature subset can be seen under three considerations.

- (i) Set $|X'| = m < n$. Find $X' \subseteq X$, such that $J(X')$ is maximum.
- (ii) Set a value J_0 , that is, the minimum J that is going to be tolerated. Find the $X' \subseteq X$ with smaller $|X'|$, such that $J(X') \geq J_0$.
- (iii) Find a compromise among minimising $|X'|$ and maximising $J(X')$ (general case).

In the present work, a wrapper approach was used [40]. Usually, the existing data is divided in two sets, the training and the test data. For that purpose, the existence of (i) a representative set of data, capable of allowing the identification of the relations between the features and the classification of such data, (ii) an algorithm able to classify the data accurately (classification algorithm), (iii) and an optimisation algorithm able to find the best set (or the minimum) of features that classify the data with the best accuracy and/or the minimum error is necessary.

Figure 1 illustrates the well-known confusion matrix, for a situation with two classes. TP (true positives) are the positive instances correctly classified, TN (true negatives) are the negative instances correctly classified, FP (false positives) are the positive instances incorrectly classified, and FN (false negatives) are the negative instances incorrectly classified. Based on this taxonomy, different measures can be defined to quantify the accuracy and the error achieved by the classifier as follows:

$$\begin{aligned}
 \text{Acc} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \\
 R &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\
 P &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\
 e_I &= \frac{\text{FP}}{\text{FP} + \text{TN}} \\
 e_{II} &= \frac{\text{FN}}{\text{TP} + \text{FN}} \\
 F_m &= \frac{2 \cdot P \cdot R}{P + R},
 \end{aligned}
 \tag{1}$$

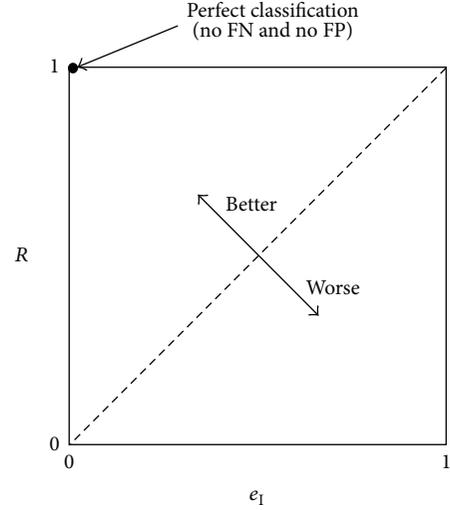


FIGURE 2: ROC curve.

where Acc is the accuracy, R is the recall or sensitivity, P is the precision, e_I and e_{II} are the classification errors of types I and II, respectively, and F_m is the harmonic mean of the sensitivity (R) and precision (P). After the above formalism, the problem consists in maximising Acc, R , P , and F_m and minimising the errors. There are other type of classification measures that can be also applied. However, the problem to be addressed is the simultaneous optimisation of some of these measures. For example, in bankruptcy prediction, the maximisation of the profits, but, simultaneously, the minimisation of losses is desired. In the present situation the profits can be quantified by recall (R), since it is a direct measure of the positives correctly classified (TP), that is, the companies that test *well* and are *healthy*, and the losses can be quantified by the error of type I (e_I), a measure of the positives incorrectly classified (FP), that is, the companies that test *well* but actually are in *bankruptcy*. The trade-off between R and e_I is known as the Receiver Operating Characteristics (ROC) curve [41, 42]. Figure 2 illustrates this concept. The ideal point is identified by “1” and means a perfect classification.

The above example illustrates the importance of optimising more than one objective simultaneously. In fact, in the case of feature selection, the first objective to be optimised (minimised) is the number of features that are necessary to get an accurate classification, which can be taken into account by maximising Acc or F_m , for example, but also by obtaining the best trade-off between R and e_I , as illustrated in Figure 2. In the first case, there are two objectives to be satisfied simultaneously, while, in the second, three objectives are considered. Therefore, the use of a multi-objective optimisation algorithm together with an accurate classifier is of primordial importance.

2.2. *Support Vector Machines.* There are available in the literature a large number of algorithms/methods for classification of data. For example, the WEKA software offers a great number of different methods ready to be used in

```

Create a random initial population (internal);
Create an empty external population;
while Not Stopping Condition do
  Evaluate internal population;
  Compute ranking of individuals using clustering;
  Compute fitness of the individuals using a ranking function;
  Copy the best individuals to the external population;
  if External population becomes full then
    Apply the clustering to this population;
    Copy the best individuals to the internal population;
  end if
  Select the individuals for reproduction;
  Crossover;
  Mutation;
  Archive best solutions;
end while

```

ALGORITHM 1: Reduced Pareto set genetic algorithm (RPSGA).

an straightforward way [43]. A good survey about the best classification algorithms can be found in [44].

The method adopted here is the Support vector machines (SVMs). SVMs are a set of supervised learning methods based on the use of a kernel, which can be applied to classification and regression [45]. In the SVM, a hyperplane or set of hyperplanes are constructed in a high-dimensional space. The initial step consists in transforming the data points, through the use of nonlinear mapping, into the high-dimensional space. In this case, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class. Thus, the larger this margin, the smaller the generalisation error of the classifier. SVMs can be seen as an extension to nonlinear models of the generalised portrait algorithm developed in [46]. For the purpose of this work, the SVM package from LIBSVM was used [47].

The SVMs depend on the definition of some important parameters. First, it is necessary to select the the type of kernel. In the present work, the Radial Basis Function (RBF) kernel was adopted due to its efficiency. Then, it becomes necessary to select the SVM type which depends on its usage, that is, if it is used for classification or regression. Since this work deals with classification, the μ -SVC and C -SVC methods were selected. Both, the kernel and the type of SVM, depend on the value defined for some parameters that must be carefully set, the kernel parameter (γ) and the regularisation parameter that depends on the type of SVM chosen (μ and C). Finally, some other parameters were studied including the training method and the training fraction. Two different training methods were tested, the holdout method, where a fraction (training fraction) of the instances are used to train the SVM and the remaining are used for testing and the k -fold method, that consists in dividing the set of instances in k subsets. Then $k - 1$ subsets are used to train the SVM and the remaining set is used for validation. The process is repeated k times, accounting for all subsets used for validation, and the accuracy is obtained as the average of the k training/testing steps [47].

Due to the large number of parameters that must be set before applying the optimisation algorithm, it makes sense to apply the feature selection algorithm and the optimisation of these parameters simultaneously. This is what is done in the present work. Therefore, the following parameters were optimised simultaneously with the process of feature selection: training method (holdout, H; or 10-fold, $K(10)$, validation), training fraction (TF), kernel (γ), and regularisation parameters (μ or C). More details about the implementation of this strategy are given in the next subsection.

2.3. Multiobjective Evolutionary Algorithm. In order to deal with multiple objectives multiobjective optimisation algorithms (MOOA) must accomplish two basic functions simultaneously: (i) they need to guide the population towards the optimal Pareto set. This can be done by using a fitness assignment operator that takes into account the non-dominance concept. (ii) The nondominated set must be maintained as diverse as possible; that is, the solutions must be well distributed along the entire optimal Pareto front. Additionally, it is also necessary to maintain an archive of the best solutions found during the various generations in order to prevent some nondominated solutions from being lost. Therefore, generally in MOEAs, it is only necessary to replace the selection phase of a traditional EA by a routine able to deal with multiple objectives [48, 49].

In this work, the MOEA adopted is the reduced Pareto set genetic algorithm (RPSGA) [50]. However, any other multi-objective algorithm can be used for the same purpose. The main steps of this algorithm are described below (Algorithm 1). The algorithm starts by the random creation of an internal population of size N and an empty external population of size $2N$. Then, at each generation (i.e., while the stopping criteria are not met), the following operations are performed: (i) the internal population is evaluated using the SVM routine; (ii) a clustering technique is applied to reduce the number of solutions on the efficient frontier and to calculate the ranking of the individuals of the internal population;

(iii) the fitness of the individuals is calculated using a ranking function; (iv) a fixed number of the best individuals is copied to the external population; (v) if the external population is not totally full, the genetic operators of selection, crossover, and mutation are applied to the internal population to generate a new population; (vi) when the external population becomes full, the clustering technique is applied to sort the individuals of the external population, and a predefined number of the best individuals is incorporated in the internal population by replacing lowest fitness individuals.

Detailed information about this algorithm can be found in [50, 51]. The influence of some important parameters of the algorithm, such as size of internal and external populations, number of individuals copied to the external population in each generation and from the external to the internal population, and the limits of the indifference of the clustering technique, had already been studied and the best values have been suggested [50].

2.4. Methodology for Feature Selection. The linkage between the problem to solve (the selection of features), the SVM, and the MOEA is done as follows. During the generation of the initial population, the chromosome (generated randomly) is constituted by a binary string identifying if the corresponding feature is present (value equal to 1) or not (value equal to 0) and the values of the classification algorithm/process (TE, H or $K(10)$, γ and μ , or C), which are used for self-adaptation. These chromosome values are then passed to the SVM during the evaluation of the population. The SVM returns the achieved values of accuracy and errors (I) obtained with the selected features and parameter values that are present in the chromosome of each individual.

The RPSGA algorithm was adapted to deal with the above feature selection problem. With respect to the classifier parameters, two approaches were considered. Initially, a pure feature selection problem was analysed where these parameters were not allowed to vary after being set up at the beginning of the algorithm. In a second approach, these parameters were included in the chromosome as variables to be optimised. The latter approach has the advantage of obtaining in a single run the best features and, simultaneously, fine tuning the classifier parameters (self-adaptation). Each candidate solution generated by the RPSGA will be externally evaluated by the SVM whose result will be returned to the RPSGA to be used as fitness in the genetic routine. New solutions will be generated based on the performance of the previous generation. As usual, the fittest solutions have more possibilities of survival.

3. Datasets

In the present study, the four datasets presented below will be used to validate the proposed methodology. Note that the DIANE data consists of two datasets from different years.

3.1. Industrial French Companies Data. In the present work, two samples, from the years 2005 and 2006, respectively, obtained from the DIANE database were selected. The original database comprised financial ratios of about 60 000

TABLE 1: Set of features considered for the industrial French companies.

Feature	Designation
F1	Number of employees
F2	Capital employed/fixed assets
F3	Financial debt/capital employed
F4	Depreciation of tangible assets
F5	Working capital/current assets
F6	Current ratio
F7	Liquidity ratio
F8	Stock turnover days
F9	Collection period
F10	Credit Period
F11	Turnover per employee (thousands euros)
F12	Interest/turnover
F13	Debt period days
F14	Financial debt/equity
F15	Financial debt/cashflow
F16	Cashflow/turnover
F17	Working capital/turnover (days)
F18	Net current assets/turnover (days)
F19	Working capital needs/turnover
F20	Export
F21	Value added per employee
F22	Total assets/turnover
F23	Operating profit margin
F24	Net profit margin
F25	Added value margin
F26	Part of employees
F27	Return on capital employed
F28	Return on total assets
F29	EBIT margin
F30	EBITDA margin

industrial French companies with at least 10 employees. The dataset includes information about 30 financial ratios of companies covering a wide range of industrial sectors (see Table 1). Since the original database contained many instances with missing values, especially, concerning defaults companies, the default cases were sorted by the number of missing values and only samples with less than 10 missing values were selected. A final set of 600 default examples was obtained. In order to create a balanced dataset, 600 random nondefault examples were selected and added to the dataset, thus resulting in a set of 1200 examples. Similar preprocessing of this dataset can be found in [31, 52, 53].

3.2. German Credit Data. The German Credit database was created at the University of Hamburg and is publicly accessible at the UCI Machine Learning Repository [54]. It consists of 1000 instances of credit applications which are described by the 20 attributes shown in Table 2. Examples of previous usage of the German Credit dataset can be found in [55, 56].

TABLE 2: Set of features considered for the German Credit.

Feature Designation	
F1	Status of existing checking account
F2	Duration in months
F3	Savings account/bonds
F4	Purpose
F5	Credit amount
F6	Savings account/bonds
F7	Present employment since
F8	Instalment rate in percentage of disposable income
F9	Personal status and sex
F10	Other debtors/guarantors
F11	Present residence since
F12	Property
F13	Age in years
F14	Other instalment plans
F15	Housing
F16	Number of existing credits at this bank
F17	Job
F18	Number of people being liable to provide maintenance for
F19	Telephone
F20	Foreign worker

There are two versions of the German dataset available, the original German Credit dataset which consists of numerical and nominal attributes and its numeric version produced at the Strathclyde University. As the method proposed in this paper only accepts numerical attributes, the numeric version of the data will be used.

3.3. Australian Credit Data. The Australian Credit database originates from [57] and concerns data from 690 credit card applications. The data are publicly available in the UCI Machine Learning Repository [54]. Each instance consists of 14 attributes and one of two possible classes (all attribute names and values were changed to meaningless symbols to protect the confidentiality of the data). The class distribution is similar for both, 44.5% versus 55.5%. Examples of previous usage of this dataset can be found in [58].

3.4. Data Normalisation. In general, a large amount of data is available and often these data are inconsistent and redundant being necessary considerable manipulation to make it useful for problems like credit risk analysis. It becomes important to identify the ratios or ranges of data that are relevant to the problem. Restricting the data to the relevant ranges represents an advantage to reduce the complexity of the problem.

Due to the large diversity of data concerning the type of data (e.g., real or integer values, numeric or categorical) and the range of variation of the values for each feature, some

TABLE 3: Set of computational experiments (N1 is the maximum number of features in the initial population; X means experiment not done).

Exp	SVM type	Objectives	N1 F-G-A
1	C-SVC14	NF + Acc	30-20-14
2	C-SVC01	NF + Acc	30-20-14
3	C-SVC02	NF + F_m	30-20-14
4	C-SVC03	NF + e_1	30-20-14
5	C-SVC04	NF + e_{11}	30-20-14
6	C-SVC05	NF + P	30-20-14
7	C-SVC06	NF + R	30-20-14
8	C-SVC07	NF + $R + e_1$	30-20-14
9	C-SVC08	NF + $R + e_1$	5-5-5
10	C-SVC09	NF + $R + e_1$	15-15-10
11	C-SVC10	NF + $R + e_1$	25-X-X
12	μ -SVC11	NF + Acc	30-20-14
13	μ -SVC12	NF + F_m	30-20-14
14	μ -SVC13	NF + $R + e_1$	30-20-14

normalisation of the data becomes necessary. Therefore, the data was transformed as follows:

(1) logarithmic transformation:

$$x'_{ij} = \begin{cases} \log(x_{ij} + 1) & x_{ij} \geq 0 \\ -\log(-x_{ij} + 1) & x_{ij} < 0, \end{cases} \quad (2)$$

(2) centering and standardizing the data:

$$x''_{ij} = \frac{x'_{ij} - \overline{\text{AVG}}(x'_j)}{\text{STD}(x'_j)}, \quad (3)$$

(3) normalisation of the data in the interval $[-1, 1]$:

$$y_{ij} = 2 \frac{x''_{ij} - \text{Min}(x''_{ij})}{\text{Max}(x''_{ij}) - \text{Min}(x''_{ij})} - 1, \quad (4)$$

where i represents the instance, j stands for feature, x_{ij} is the original data in a matrix form (which is transformed successively in x'_{ij} and x''_{ij}), $\overline{\text{AVG}}(x'_j)$ and $\text{STD}(x'_j)$ are the average and the standard deviation of all instances for feature j , respectively, and y_{ij} is the final value used by the classifier. The data used by the classifier is restricted to the interval $[-1, 1]$ as recommended in [44].

4. Results and Discussion

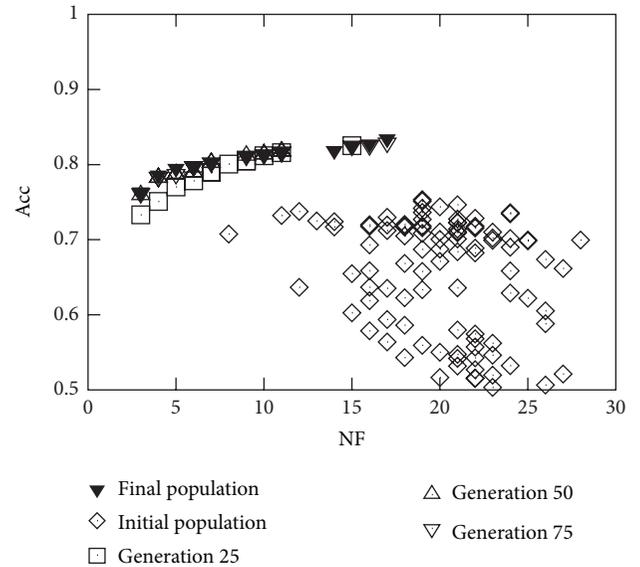
4.1. Computational Experiments. Table 3 presents the set of experiments carried out to test the proposed approach. Due to the stochastic nature of the evolutionary algorithm, 10 runs (a, b, \dots, j) for each experiment were performed, using

TABLE 4: Optimal solutions for *Run-a* of Experiment 2 (Diane 2005 database).

NF	Features	Acc	TM	TF	γ	C
3	1, 14, 24	76.3%	H	50.4%	0.501	10.4
4	1, 14, 16, 24	78.6%	H	50.6%	0.403	39.6
5	1, 8, 14, 21, 24	79.5%	H	52.0%	0.280	211.0
6	1, 8, 14, 15, 16, 24	79.8%	H	52.2%	0.297	173.9
7	1, 8, 12, 14, 15, 16, 24	80.3%	H	54.4%	0.543	112.8
9	1, 8, 12, 14, 21, 23, 24, 25, 27	81.2%	H	52.2%	0.134	86.9
10	1, 8, 12, 14, 15, 16, 21, 23, 24, 25	81.4%	H	53.6%	0.343	114.6
11	1, 8, 12, 14, 15, 16, 21, 23, 24, 25, 27	81.7%	H	52.2%	0.164	59.7
14	1, 5, 7, 8, 9, 12, 14, 15, 16, 18, 21, 23, 24, 25	81.9%	H	52.1%	0.539	23.9
15	1, 2, 5, 7, 8, 9, 12, 14, 15, 16, 18, 21, 23, 24, 25	82.5%	H	52.3%	0.384	26.8
16	1, 2, 5, 6, 7, 8, 9, 12, 14, 15, 16, 18, 21, 23, 24, 27	82.8%	H	52.2%	0.405	24.5
17	1, 2, 5, 6, 7, 8, 9, 12, 14, 15, 16, 18, 21, 23, 24, 25, 27	83.5%	H	52.1%	0.354	24.8

different seed values (as required by the random number generator). In the case of Experiment 1, the C-SVC method was used with the following fixed parameters: holdout (H) validation as training method, TM, training fraction, TF, equal to 0.7, kernel parameter, γ , equal to 0.1, and the regularisation parameter, C, equal to 10. In the remaining experiments, these parameters were allowed to range in the following intervals: $\gamma \in [0.005, 10]$, $C \in [1, 1000]$, $\nu \in [0.01, 0.5]$, $TM \in [H \text{ or } K(10)]$, and $TF \in [0.5, 0.8]$. In Table 3, $N1$ represents the maximum number of features allowed in the initial generation, that is, if $N1$ is equal to 5 means that in the initial generation the individuals of the population have at the most 5 features. In consecutive generations, the number of selected features was allowed to grow until the maximum of features for each database is reached: for French industrial companies, subscript F, $N_{\max} = 20$; for German Credit data, subscript G, $N_{\max} = 20$; and for Australian Credit data, subscript A, $N_{\max} = 14$. Besides, Figures 1 and 2 should not become a problem (with respect to the dataset dimension) for standard SVMs experimentation; this work tries to demonstrate that feature selection is useful for the application of SVMs over datasets of high dimension.

The aim of Experiment 1 is to compare the performance of the feature selection method proposed when the classifier parameters are fixed to that of the same method when the parameters are allowed to vary. This will be done by comparing Experiments 1, 2, and 12. Experiments 2 to 7 are thought to illustrate the influence of the method when different classification measures are applied. In the case of Experiments 8 to 11, the aim is to study the influence of the maximum number of features of the initial population (NF) in the evolution of ROC curves (i.e., R versus e_r). Finally, Experiments 12, 13, and 14 were intended to show the influence of the SVM method used. In all runs, the following RPSGA parameters were used (see [50] for more details): the main and elitist population sizes were 100 and 200 individuals, respectively; fitness proportional selection was adopted; crossover rate of 0.8 and mutation probability of 0.05 were used; the number of ranks was set to 30 and the limit of indifference of the clustering technique was set to 0.01, whereas the number of generations was set to 100 for all runs.

FIGURE 3: *Run-a* of Experiment 2 (initial population and nondominated solutions of the final population).

4.2. *Analysis of a Standard Experiment.* This section is aimed at showing the type of results that can be obtained using the proposed methodology. For that purpose, Figure 3 shows the entire initial population and the nondominated solutions corresponding to generations 25, 50, 75, and 100 for *Run-a* of Experiment 2. This graph presents the trade-off between Acc (to be maximised) and NF (to be minimised). It can be easily observed that the algorithm is able to evolve the population significantly, from the initial population (randomly generated), located predominantly at the bottom right corner, towards the top left corner. It is also noticeable that only 50 generations are needed to reach a reasonable approximation of the Pareto front. The use of 100 generations was only used to guarantee the convergence of the algorithm.

Table 4 shows the obtained results corresponding to the decision variable domain for the above run after 100 generations. The accuracy is ranged between 76.3% and

TABLE 5: Optimal solutions for Experiment 2 (Diane 2005 database).

NF	Run	Features	Acc	TM	TF	γ	C
2	f	7, 21	75.6%	H	73.9%	0.066	373.1
3	f	7, 21, 23	80.7%	H	74.5%	0.127	668.5
4	f	7, 21, 22, 29	80.8%	H	74.3%	0.0102	855.8
5	f	7, 13, 21, 23, 29	81.2%	H	74.4%	0.0104	844.4
6	e	6, 12, 13, 19, 21, F29	81.8%	H	75.3%	0.373	41.5
8	f	7, 8, 13, 18, 21, 23, 27, 28	83.0%	H	75.5%	0.195	754.0
9	f	7, 8, 13, 18, 21, 22, 23, 27, 28	84.2%	H	75.1%	0.179	901.7
10	f	7, 8, 10, 13, 18, 21, 22, 23, 27, 28	85.8%	H	75.3%	0.156	866.4

83.5%, when considering a minimum number of 3 features and a maximum of 17, respectively. In all cases, the holdout (H) cross validation training method was selected and the training fraction lies around 52% and γ is ranged between 0.13 and 0.55, whereas C fluctuate between 10 and 211. This indicates that decision variables (TM, TF, γ , and C) converge for a small interval when compared to the initial range where they are allowed to vary.

However, the target consists in finding better solutions than those obtained over a single run. Figure 4 shows the optimal Pareto curves of the 10 runs that were performed for Experiment 2. It can be seen that there is one of these runs that dominates the others, *Run-f*, except when $NF = 6$, where the best solution is obtained for *Run-e*. Table 5 shows the decision variable values of the corresponding Pareto front, for which Acc is ranged between 75.6% and 85.8%, the obtained TM is hold out for all cases, and the TF lies around 75%. On the other hand, the SVM parameters have a large variation which indicates that γ and C play an important role in acquiring best accuracies. Similar conclusions can be drawn when analysing the results obtained using the remaining datasets.

4.3. Analysis and Comparison of Results. Figures 5 and 6 represent the nondominated solutions of the 10 different runs carried out in Experiments 2 to 7 using to the French industrial companies in 2005 dataset. These plots allow to assess the efficiency of the proposed optimisation methodology when dealing with all the objective function measures presented in Section 2. As expected, and since the common objective used in these experiments is the minimisation of NF, the solutions evolve nicely towards the region where the true Pareto front is supposed to be; that is, when simultaneously maximising a second objective (e.g., Acc, F_m , R, and P) the solutions evolve towards the top left corner, while when simultaneously minimising a second objective (e.g., e_I and e_{II}), the solutions evolve towards the bottom left corner.

Further analysis of Figures 5 and 6 helps to identify the ranges that can be accomplished when using the different objective functions (for the French datasets): Acc and $F_m \in [70\%, 85\%]$, $P \in [80\%, 100\%]$, $R \in [60\%, 95\%]$, $e_I \in [0\%, 20\%]$, and $e_{II} \in [5\%, 35\%]$. However, when considering the best values in a particular run, the following values were found: Acc = 85.8%, $F_m = 85.0\%$, $e_I = 2.3\%$, $e_{II} = 7.1\%$, $P = 97.9\%$, and $R = 92.9\%$, corresponding to NF equal to 10,

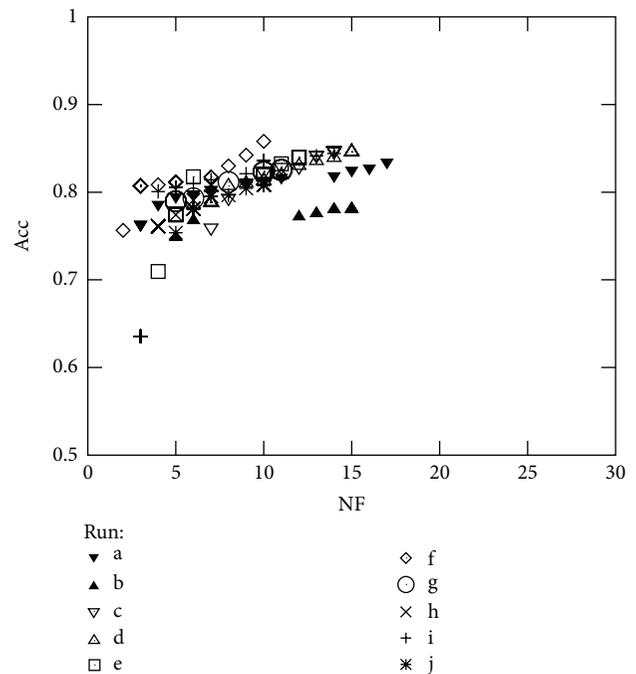
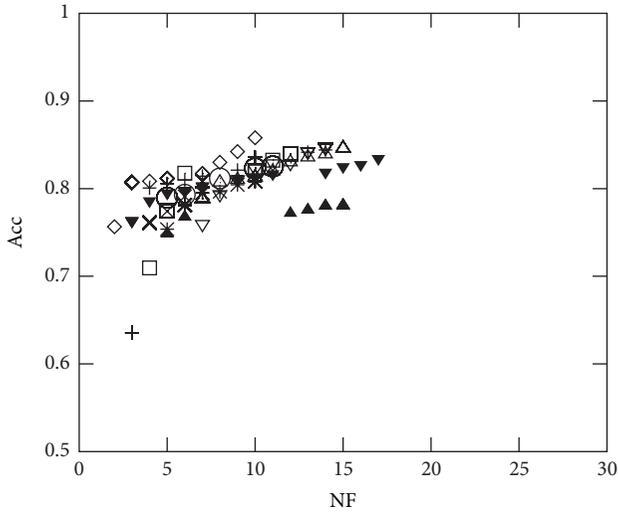


FIGURE 4: All runs of Experiment 2 (nondominated solutions of the final population).

11, 5, 13, 3, and 13, respectively. Considering a given number of features, for example, $NF = 10$, the following best values are found: Acc = 85.8%, $F_m = 84.4\%$, $e_I = 3.0\%$, $e_{II} = 9.8\%$, $P = 96.1\%$, and $R = 90.2\%$. On the other hand, when considering all ten runs of each experiment, the variation range for each objective function can be graphically observed. Such a variation enforces the use of several runs with different seed values in order to select the best set of features as well as the best classifier parameters. Since the final accuracy will depend certainly on the combination of the right features, the methodology adopted cannot be based on selecting the features that appear more frequently in the 10 runs performed for each experiment [59].

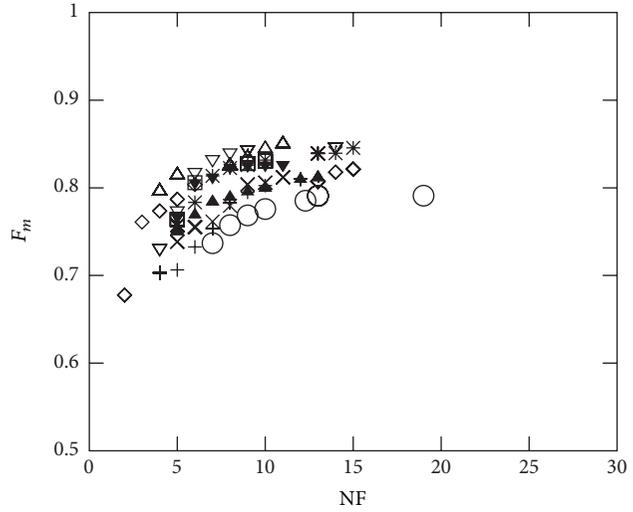
The above reasoning was used to select the best solution of the front when comparing the results from Experiments 1, 2, and 12 over all datasets studied. Note that Experiments 1, 2, and 12 consist on simultaneously optimising NF and Acc



Run:

- ▼ a
- ▲ b
- ▽ c
- △ d
- e
- ◇ f
- g
- × h
- + i
- * j

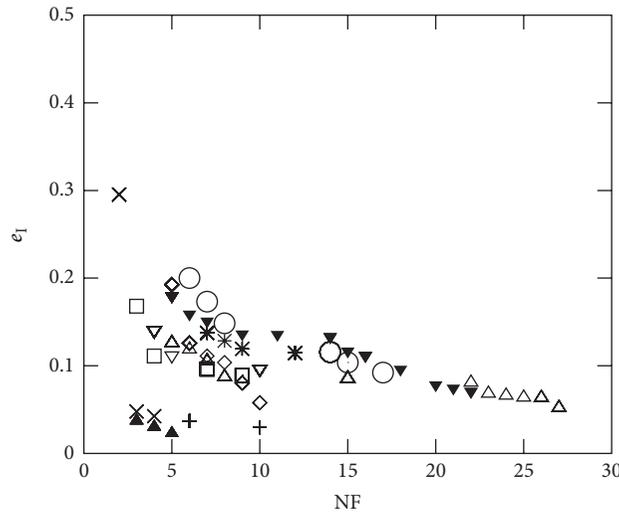
(a) Experiment 2



Run:

- ▼ a
- ▲ b
- ▽ c
- △ d
- e
- ◇ f
- g
- × h
- + i
- * j

(b) Experiment 3



Run:

- ▼ a
- ▲ b
- ▽ c
- △ d
- e
- ◇ f
- g
- × h
- + i
- * j

(c) Experiment 4

FIGURE 5: Optimal Pareto fronts for Diane 2005 data (10 runs).

(see Figures 7, 8, 9, and 10). Furthermore, the above analysis allowed to create Table 6 which summarises the solutions found for three different cases: solutions with best accuracy (Best) and best solutions using only 5 ($NF \leq 5$) and 10 ($NF \leq 10$) features, respectively.

As expected and in general, the results of Table 6 show that the best accuracy is obtained when the classifier parameters are also optimised (Experiments 2 and 12). Concerning the use of the C-SVC or the μ -SVC kernels, no definitive conclusion can be drawn, since the C-SVC kernel yielded

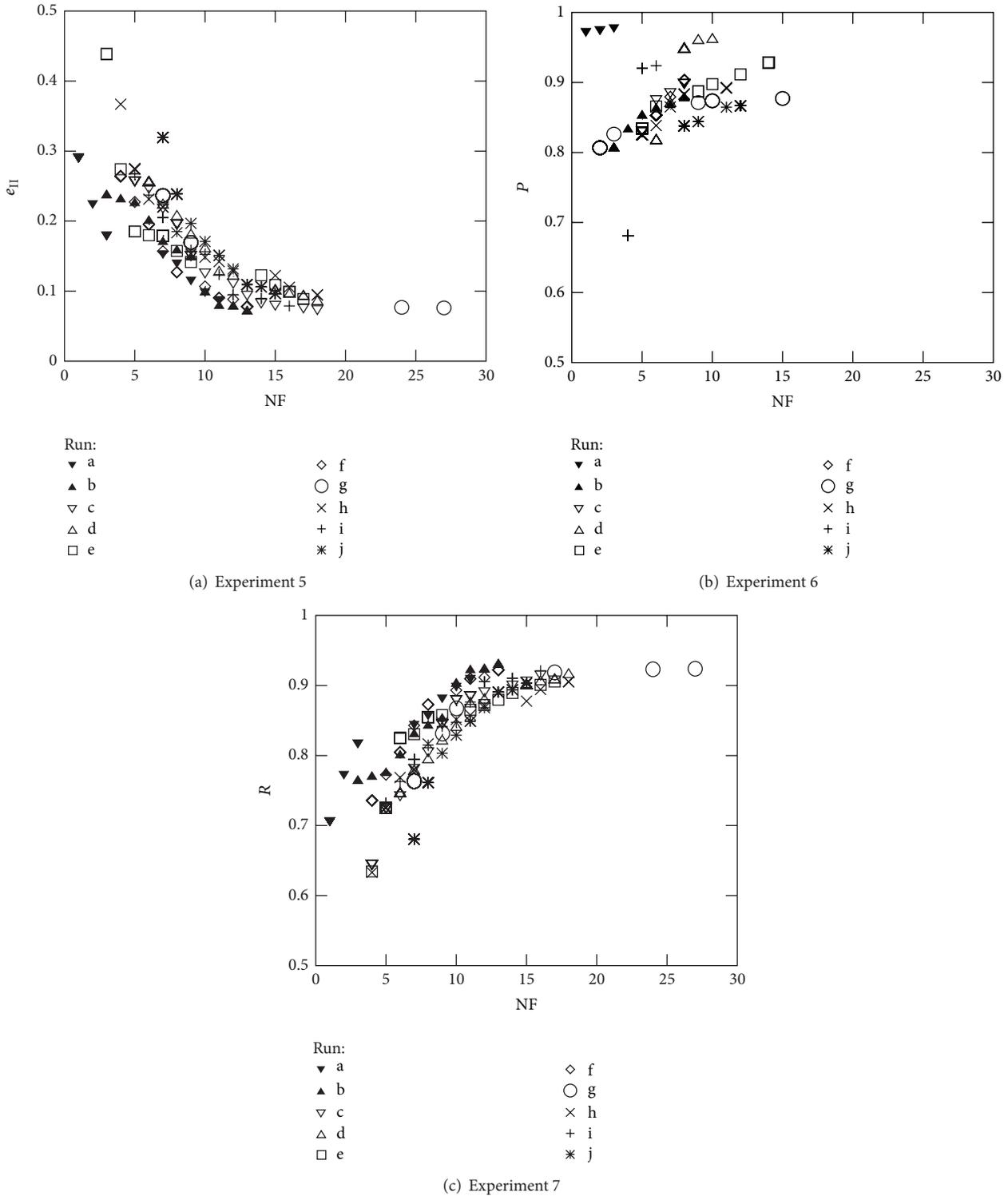


FIGURE 6: Optimal Pareto fronts for Diene 2005 data (10 runs).

the best result for Diene05, whereas the μ -SVC kernel yielded the best result the the Australian data and for some other cases the best result depends on the number of features (Diene06 and German data). With respect to the runs where the “best” results were obtained for each of the three

conditions that were analysed, again there is some variability; in some cases, the results were taken from the same run but in most of the cases they were not. Again, this fact was expected after the analysis made in the previous section. In all cases, the holdout validation method is selected, TF ranges between

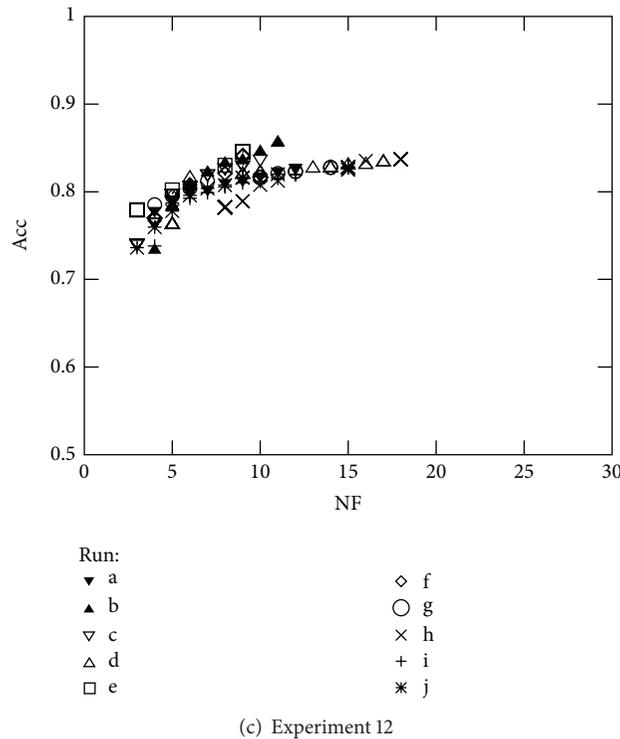
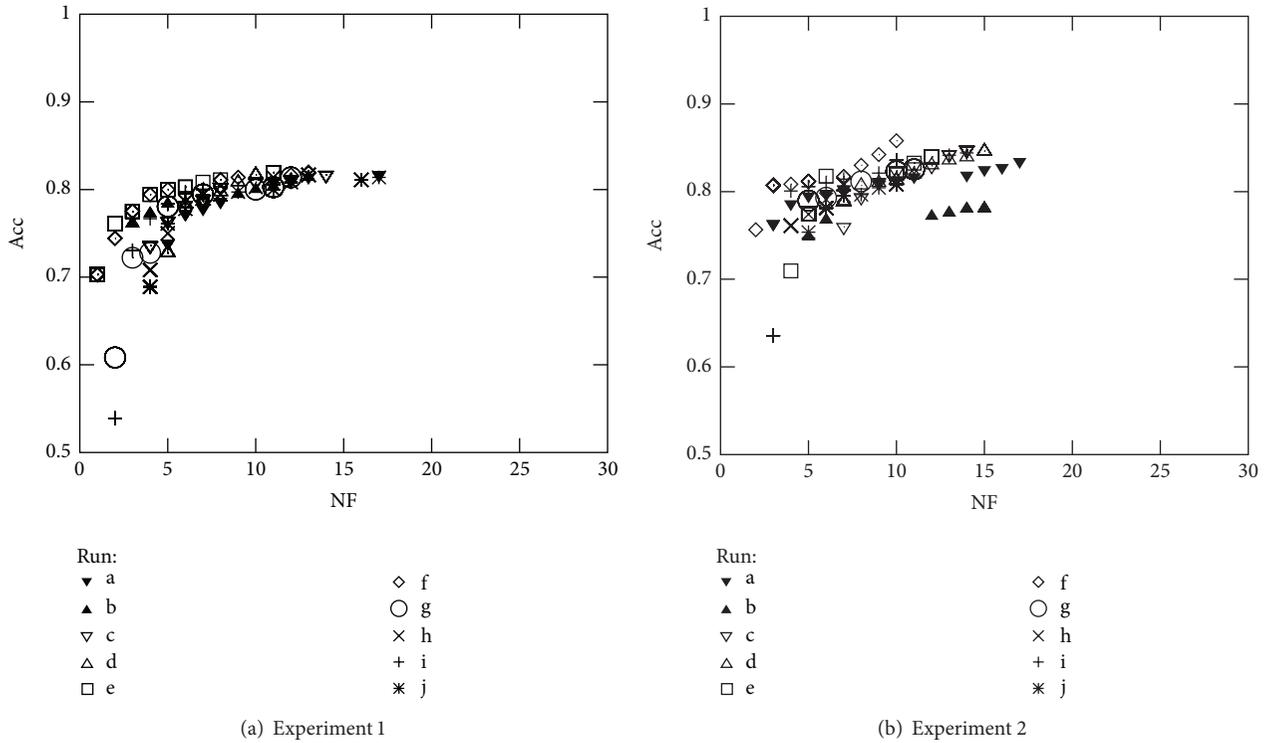


FIGURE 7: Optimal Pareto fronts for Diane 2005 data (10 runs).

70% and 80% in most cases (except in the case of the German database), and the kernel and regularisation parameters have a high variability to maximise the accuracy. This was also expected after the analysis of the previous section.

The analysis or results show that the desired accuracy can be achieved using several combinations of features. Results

coming from the same run tend to select the same features (this fact was also observed in the results presented in Tables 4 and 5). An interesting finding came from Experiment 2 over Diane05 database; it was observed that when the number of features was reduced to 5 at the most ($NF \leq 5$), four out of five of the features selected were identical to one of the features

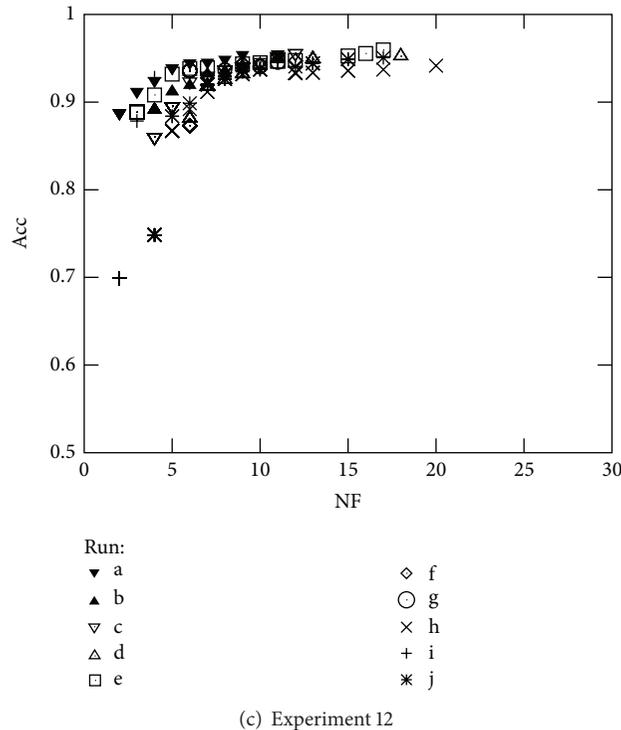
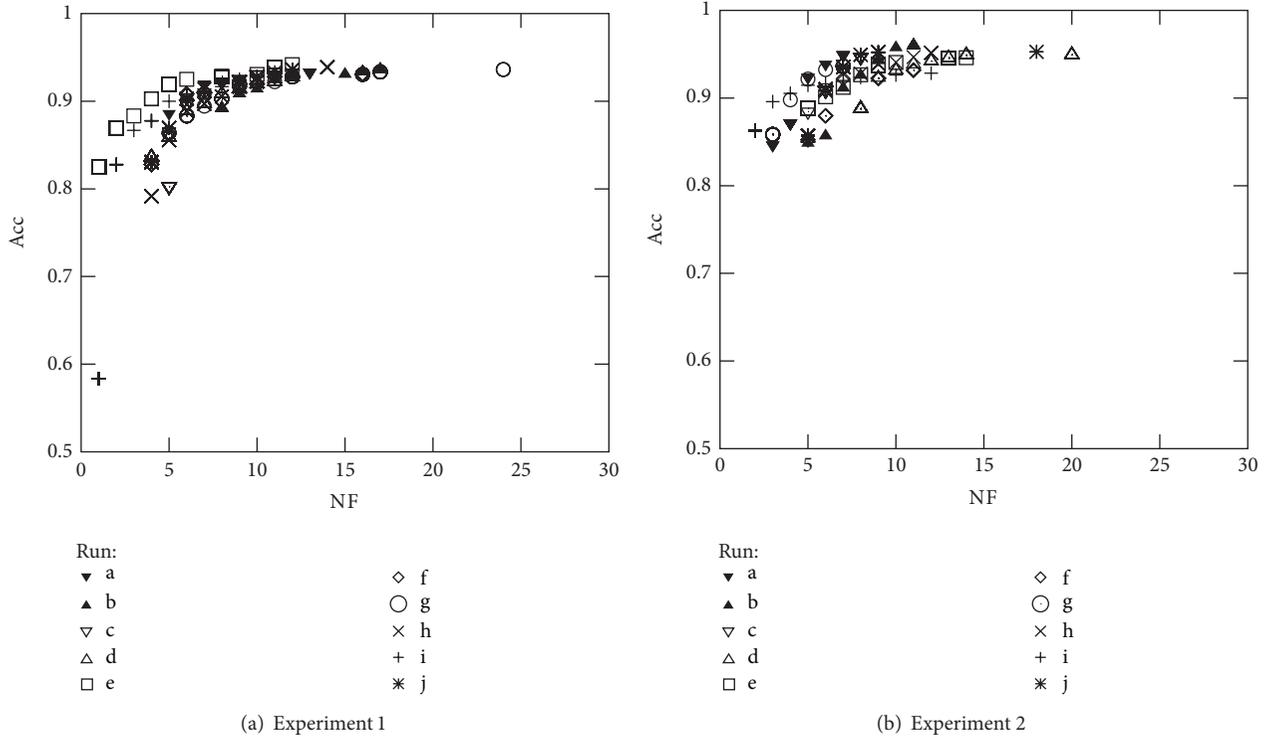


FIGURE 8: Optimal Pareto fronts for Diane 2006 data (10 runs).

that were selected for the best solution condition (features 7, 13, 21, and 23), but the last feature selected when using this constraint was not included in the best solution (feature 29). Many valuable information can be obtained from Table 6. As an example, if the problem consists on obtaining the best accuracy using five features at the most ($NF \leq 5$), the features

identified in bold should be selected to be used in future classifications together with their corresponding parameter for each dataset considered.

Figures 11 and 12 show the best results achieved in Experiments 8, 9, 10, 11, and 14. Note that these experiments consist in optimising three different objectives (R , e_1 , and NF)

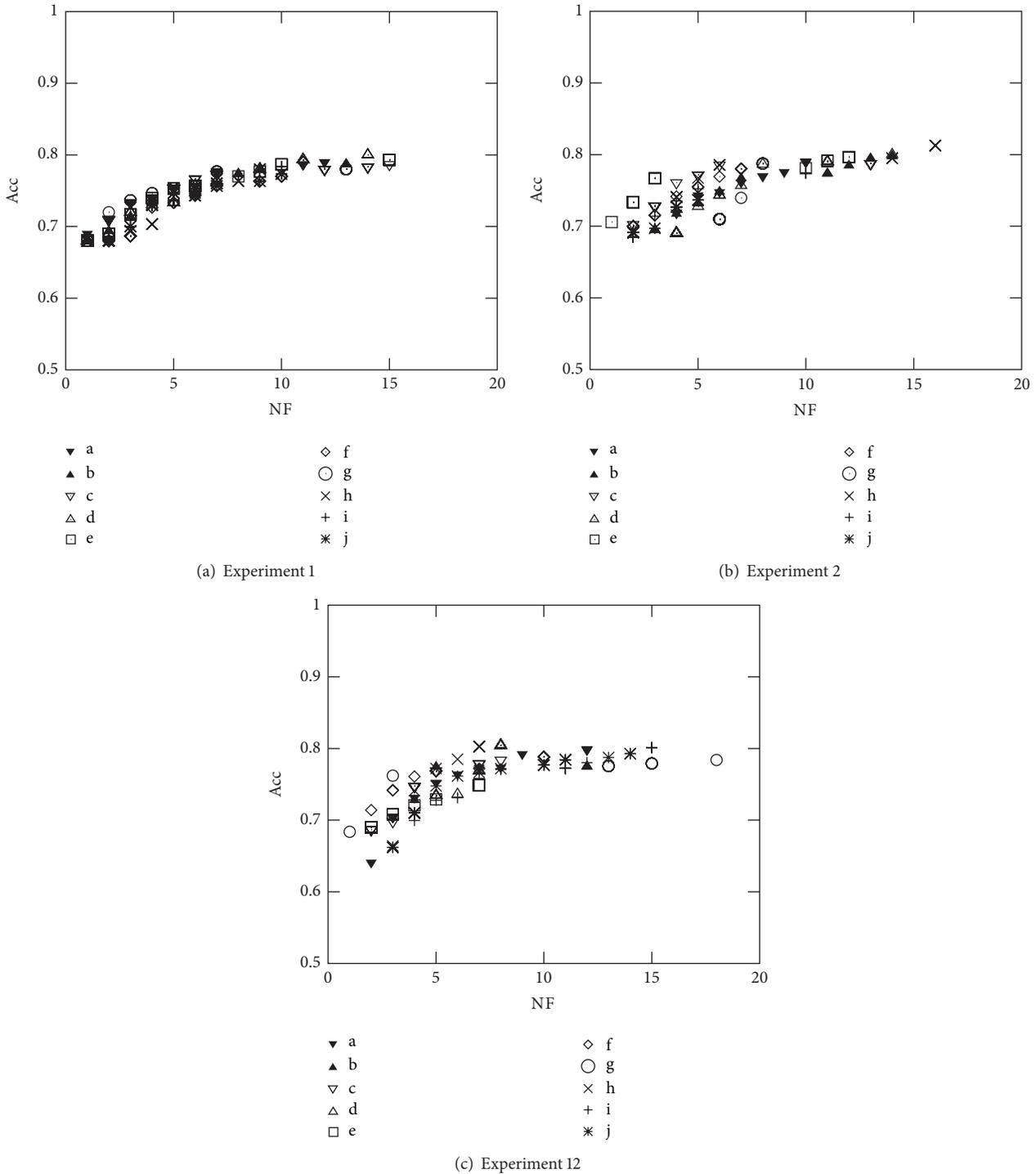
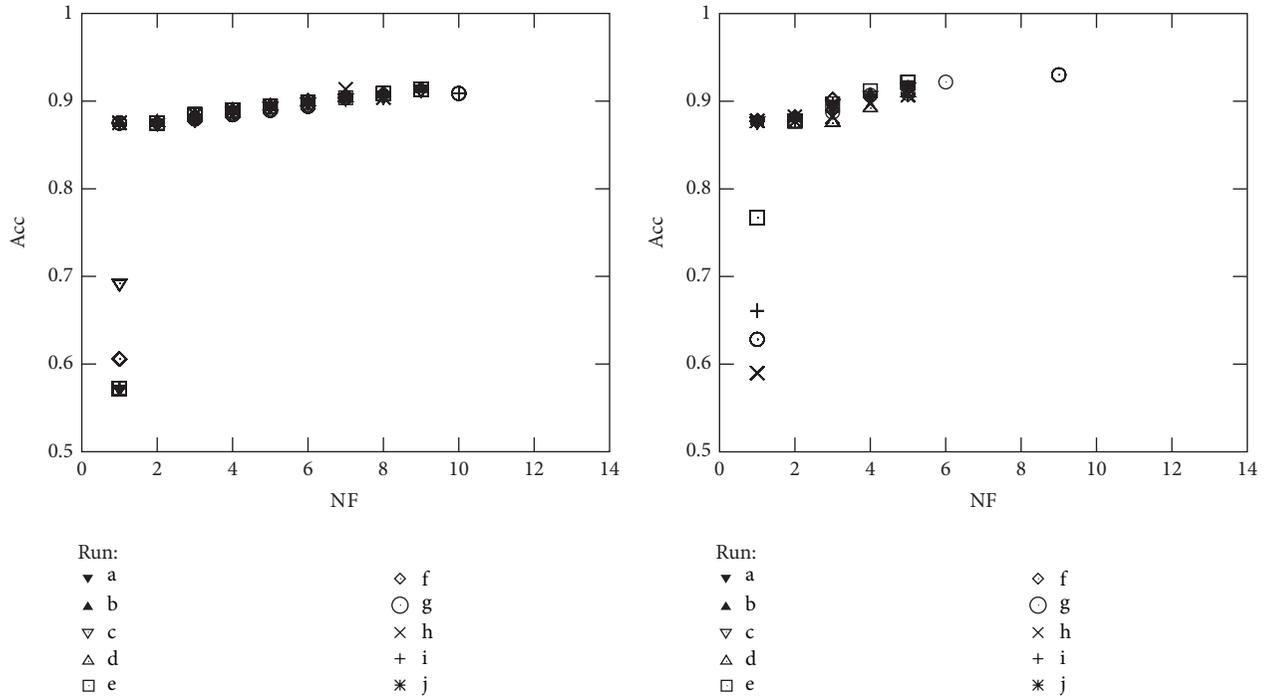


FIGURE 9: Optimal Pareto fronts for German Credit data (10 runs).

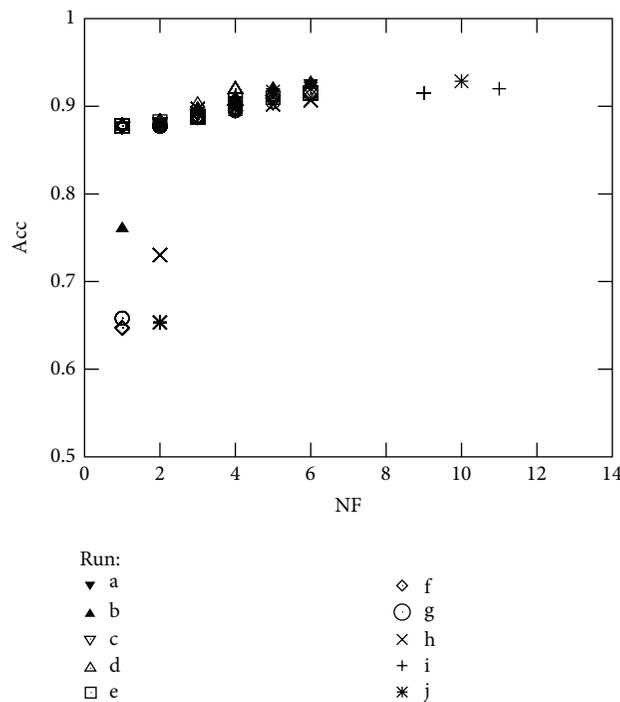
and were aimed to obtain the results that best fit in a ROC curve; that is, $R = f(e_1)$. Besides the optimisation that was carried out considering all three objectives, only nondominated solution with respect to objectives R and e_1 are presented (best of 10 runs for each experiment). Table 7 shows a summary of results from the above experiments for all databases using two different conditions ($e_1 \leq 10\%$

and $NF \leq 5$). The area under ROC was computed at first for all cases and then best results were presented for each condition. Identical conclusions, to that of the beginning of this section, Section 4.3, can be made here concerning the algorithm parameters, that is, best kernel (which depend on the database), best validation method, training fraction, and kernel and regularisation parameters. Similarly, there exist



(a) Experiment 1

(b) Experiment 2



(c) Experiment 12

FIGURE 10: Optimal Pareto fronts for Australian Credit data (10 runs).

various combinations of features that allow the obtention of the best R and e_1 values. As before, the best solutions using five features at the most, $NF \leq 5$, can be selected for each database. Such features are identified in bold in Table 7 and can be used in future classification together with their corresponding classifier parameters.

In [60], clustering feature selection methods were used to identify the most relevant features on several datasets. The Australian Credit dataset was used to test three versions of a clustering based algorithm with different optimisation strategies. The structure of clusters, found by the optimisation version of the algorithm proposed in the above paper,

TABLE 6: Results summary for Experiments 1, 2, and 12 and for all databases.

Data set	Experiment	Condition	Run	NF	Acc	Features	TM	TF	γ	C/μ
Diane05	1	Best	e	11	81.94%	7, 8, 10, 12, 14, 18, 21, 22, 24, 27, 30	H	70.00%	0.10	10.00
		NF \leq 10	d	10	81.67%	3, 8, 10, 14, 15, 19, 21, 22, 24, 30	H	70.00%	0.10	10.00
		NF \leq 5	e	5	80.00%	7, 8, 18, 21, 30	H	70.00%	0.10	10.00
	2	Best	f	10	85.81%	7, 8, 10, 13, 18, 21, 22, 23, 27, 28	H	75.32%	0.16	866.44
		NF \leq 10	f	10	85.81%	7, 8, 10, 13, 18, 21, 22, 23, 27, 28	H	75.32%	0.16	866.44
		NF \leq 5	f	5	81.17%	7, 13, 21, 23, 29	H	74.38%	0.01	844.37
	12	Best	b	11	85.57%	7, 10, 12, 13, 15, 16, 18, 22, 23, 24, 27	H	75.16%	0.77	0.46
		NF \leq 10	e	9	84.56%	5, 6, 10, 12, 13, 19, 21, 22, 29	H	75.21%	0.59	0.47
		NF \leq 5	e	5	80.21%	1, 10, 12, 13, 29	H	75.94%	1.90	0.49
Diane06	1	Best	e	12	94.17%	1, 8, 9, 10, 11, 12, 14, 18, 20, 21, 24, 26	H	70.00%	0.10	10.00
		NF \leq 10	e	10	93.06%	1, 8, 9, 10, 11, 14, 20, 21, 24, 26	H	70.00%	0.10	10.00
		NF \leq 5	e	5	91.94%	1, 9, 11, 14, 24	H	70.00%	0.10	10.00
	2	Best	b	11	95.99%	1, 10, 11, 12, 15, 19, 21, 24, 25, 27, 29	H	73.04%	0.13	697.55
		NF \leq 10	b	10	95.73%	1, 10, 11, 12, 15, 19, 21, 24, 25, 27	H	72.63%	0.21	705.63
		NF \leq 5	a	5	92.38%	11, 14, 19, 24, 28	H	74.86%	0.17	281.47
	12	Best	e	17	95.95%	1, 2, 8, 10, 11, 12, 13, 14, 19, 20, 21, 22, 23, 24, 25, 29, 30	H	75.29%	2.45	0.18
		NF \leq 10	a	9	95.43%	1, 10, 11, 12, 14, 15, 19, 21, 24	H	72.70%	3.45	0.27
		NF \leq 5	a	5	93.92%	10, 11, 14, 19, 28	H	75.33%	2.08	0.34
German	1	Best	d	14	80.00%	1, 2, 3, 5, 6, 10, 12, 14, 15, 16, 17, 18, 19, 20	H	70.00%	0.10	10.00
		NF \leq 10	e	10	78.67%	1, 2, 3, 5, 6, 7, 10, 12, 18, 19	H	70.00%	0.10	10.00
		NF \leq 5	a	5	75.33%	1, 3, 5, 8, 19	H	70.00%	0.10	10.00
	2	Best	h	16	81.25%	1, 2, 3, 5, 6, 7, 38310, 11, 12, 14, 15, 17, 18, 19, 20	H	68.04%	0.01	534.75
		NF \leq 10	a	10	78.99%	1, 2, 3, 4, 6, 8, 11, 12, 15, 19	H	58.63%	0.05	62.26
		NF \leq 5	c	5	77.16%	1, 3, 5, 7, 12	H	58.45%	0.17	72.52
	12	Best	h	7	80.29%	1, 2, 3, 5, 8, 11, 14	H	58.40%	0.14	0.45
		NF \leq 10	h	7	80.29%	1, 2, 3, 5, 8, 11, 14	H	58.40%	0.14	0.45
		NF \leq 5	b	5	77.43%	1, 5, 12, 13, 14	H	77.48%	1.41	0.45
Australian	1	Best	h	7	91.35%	1, 6, 8, 10, 12, 13, 14	H	70.00%	0.10	10.00
		NF \leq 5	h	5	89.42%	6, 8, 9, 10, 14	H	70.00%	0.10	10.00
	2	Best	g	9	93.00%	2, 4, 5, 6, 8, 10, 11, 13, 14	H	70.94%	0.02	180.23
		NF \leq 5	e	5	92.16%	2, 4, 6, 8, 9	H	70.48%	0.58	321.06
	12	Best	j	10	92.86%	2, 3, 4, 5, 6, 8, 10, 11, 12, 14	H	77.61%	2.06	0.34
		NF \leq 5	b	5	92.00%	3, 4, 5, 8, 10	H	71.02%	2.65	0.34

indicates the subset of three relevant features for classification, features 8, 9, and 14. Almost all solutions obtained in the experiments carried out in this paper (see Tables 6 and 7) using the Australian data include features 8 and 9. Feature 14 is also present in most of the solutions obtained in this work.

Prediction of financial distress of companies using Diane dataset, was previously analysed using several machine learning approaches [61]. Support vector machines achieved the highest accuracy considering five features selected by SVM attribute evaluation method. The five features selected for predicting failures during 2007 using historical data from 2006, 2005, and 2004 were 1, 4, 11, 16, and 28 (which differs

from the solution obtained here). However, it should be noted that these results were obtained using historical data and, therefore, they are not comparable to the results obtained in this work.

An approach to solving classification problems by combining feature selection and neural networks was proposed in [62]. A feature selection algorithm based on the notion of entropy from the information theory was applied to the German Credit dataset yielding the selection of the following seven features: 1, 2, 3, 5, 6, 15, and 20. Authors found that the predictive accuracy was marginally larger with the exclusion of the 13 redundant features. Most of the solutions obtained

TABLE 7: Results summary for Experiments 8, 9, 10, 11, and 14 and for all databases.

Dataset	Experiment	ROC area	Condition	Run	NF	R	e_1	Features	TM	TF	γ	C/μ	
Diane05	8	0.872	$e_1 \leq 10\%$	c	12	77.1%	9.4%	8, 10, 12, 13, 15, 16, 18, 22, 23, 24, 25, 27	H	77.6%	0.62	108.73	
			$NF \leq 5$	i	4	64.6%	6.4%	15, 18, 27, 28	H	75.2%	0.03	336.98	
	9	0.859	$e_1 \leq 10\%$	b	10	72.0%	8.8%	7, 8, 10, 12, 13, 19, 22, 23, 25, 29	H	76.3%	0.26	701.29	
			$NF \leq 5$	c	5	63.0%	7.2%	13, 14, 18, 27, 28	H	76.0%	0.05	29.51	
	10	0.876	$e_1 \leq 10\%$	d	7	72.0%	8.8%	8, 10, 11, 16, 22, 24, 27	H	76.2%	0.41	19.29	
			$NF \leq 5$	c	5	64.5%	7.3%	8, 18, 22, 28, 29	H	78.5%	0.22	107.89	
	11	0.877	$e_1 \leq 10\%$	a	18	77.0%	8.6%	5, 6, 7, 8, 10, 11, 12, 13, 16, 18, 19, 21, 22, 23, 24, 26, 28, 30	H	77.6%	0.35	25.35	
			$NF \leq 5$	h	4	60.4%	5.8%	3, 10, 14, 28	H	75.7%	0.21	6.48	
	14	0.867	$e_1 \leq 10\%$	f	13	76.0%	10.1%	5, 7, 12, 13, 16, 19, 21, 22, 23, 24, 25, 27, 28	H	76.0%	0.94	0.49	
			$NF \leq 5$	h	6	50.3%	3.7%	6, 14, 21, 24, 28, 29	H	76.1%	0.01	0.03	
	Diane06	8	0.981	$e_1 \leq 10\%$	e	9	95.3%	8.0%	4, 11, 12, 13, 19, 21, 22, 25, 29	H	76.1%	1.03	157.00
				$NF \leq 5$	b	3	68.1%	0.0%	1, 15, 28	H	78.1%	6.27	60.84
		9	0.985	$e_1 \leq 10\%$	b	7	96.8%	8.0%	5, 7, 10, 11, 21, 24, 25	H	75.5%	1.15	371.95
				$NF \leq 5$	b	5	92.9%	2.1%	7, 11, 21, 25, 28	H	75.4%	1.37	240.07
10		0.982	$e_1 \leq 10\%$	i	18	96.5%	8.3%	1, 3, 4, 6, 8, 11, 12, 13, 15, 16, 17, 19, 21, 22, 23, 24, 25, 30	H	77.1%	3.50	18.67	
			$NF \leq 5$	c	7	94.2%	8.0%	1, 4, 11, 21, 25, 27, 29	H	75.5%	0.32	57.04	
11		0.982	$e_1 \leq 10\%$	d	18	95.0%	9.4%	1, 3, 5, 8, 10, 11, 12, 13, 15, 16, 17, 19, 21, 22, 26, 27, 28, 30	H	75.6%	2.39	17.30	
			$NF \leq 5$	e	10	94.2%	7.3%	1, 3, 4, 7, 11, 14, 16, 19, 22, 28	H	75.5%	7.43	880.32	
14		0.981	$e_1 \leq 10\%$	h	11	96.2%	7.3%	1, 4, 7, 11, 13, 14, 16, 18, 20, 22, 28	H	75.6%	5.77	0.34	
			$NF \leq 5$	a	8	93.4%	3.6%	1, 11, 14, 19, 22, 24, 25, 30	H	75.8%	6.01	0.49	
German		8	0.765	$e_1 \leq 10\%$	h	11	57.5%	9.4%	1, 2, 3, 5, 8, 11, 12, 14, 16, 19, 20	H	76.8%	0.02	525.81
				$NF \leq 5$	h	5	41.5%	4.9%	1, 3, 5, 8, 14	H	58.2%	0.08	669.49
		9	0.757	$e_1 \leq 10\%$	c	7	50.8%	9.4%	1, 2, 3, 5, 7, 12, 13	H	59.6%	0.27	34.09
				$NF \leq 5$	h	4	41.8%	6.5%	1, 3, 5, 8	H	75.3%	0.15	136.06
	10	0.749	$e_1 \leq 10\%$	e	10	52.2%	9.6%	1, 2, 3, 5, 10, 14, 15, 16, 19, 20	H	72.1%	0.03	773.25	
			$NF \leq 5$	g	5	50.5%	8.2%	1, 2, 3, 5, 12	H	71.2%	0.17	669.11	
	14	0.766	$e_1 \leq 10\%$	c	4	32.6%	7.5%	1, 2, 5, 7	H	59.1%	0.50	0.47	
			$NF \leq 5$	c	4	32.6%	7.5%	1, 2, 5, 7	H	59.1%	0.50	0.47	
	Australian	8	0.964	$e_1 \leq 10\%$	b	6	97.1%	9.6%	3, 4, 5, 8, 12, 14	H	76.7%	0.53	473.81
				$NF \leq 5$	d	5	82.7%	3.5%	4, 8, 9, 13	H	71.9%	9.48	75.61
9		0.957	$e_1 \leq 10\%$	g	6	95.3%	9.6%	2, 3, 4, 5, 8, 10	H	71.0%	1.12	8.05	
			$NF \leq 5$	a	5	90.9%	8.0%	4, 5, 8, 11, 13	H	77.6%	8.50	106.32	
10		0.960	$e_1 \leq 10\%$	j	5	92.9%	8.8%	6, 8, 9, 13, 14	H	71.5%	1.13	839.90	
			$NF \leq 5$	j	5	92.9%	8.8%	6, 8, 9, 13, 14	H	71.5%	1.13	839.90	
14		0.967	$e_1 \leq 10\%$	e	5	93.2%	9.5%	5, 6, 8, 9, 12	H	70.5%	1.30	0.29	
			$NF \leq 5$	e	5	93.2%	9.5%	5, 6, 8, 9, 12	H	70.5%	1.30	0.29	

by the approach presented in this work include some of these features (in particular, features 1, 2, 3, and 5). However, it should be noted that the experimental set-up of the two studies is rather different and, therefore, conclusions must be drawn carefully.

5. Conclusion

With the current global economic situation where several countries are getting through economic recession, bankruptcy prediction is acquiring importance as a financial

topic of research. When the financial data to be analysed becomes large, the need for feature selection arises as a tool used to reduce both computational times and number of computations by getting rid of irrelevant features. Feature selection also gives a method to evaluate the importance of each feature within the studied dataset.

This work aimed at investigating the feature selection problem in bankruptcy prediction using a multi-objective approach which includes self-adaptation of the classification algorithm parameters. For that purpose, a new methodology has been proposed and its performance has been

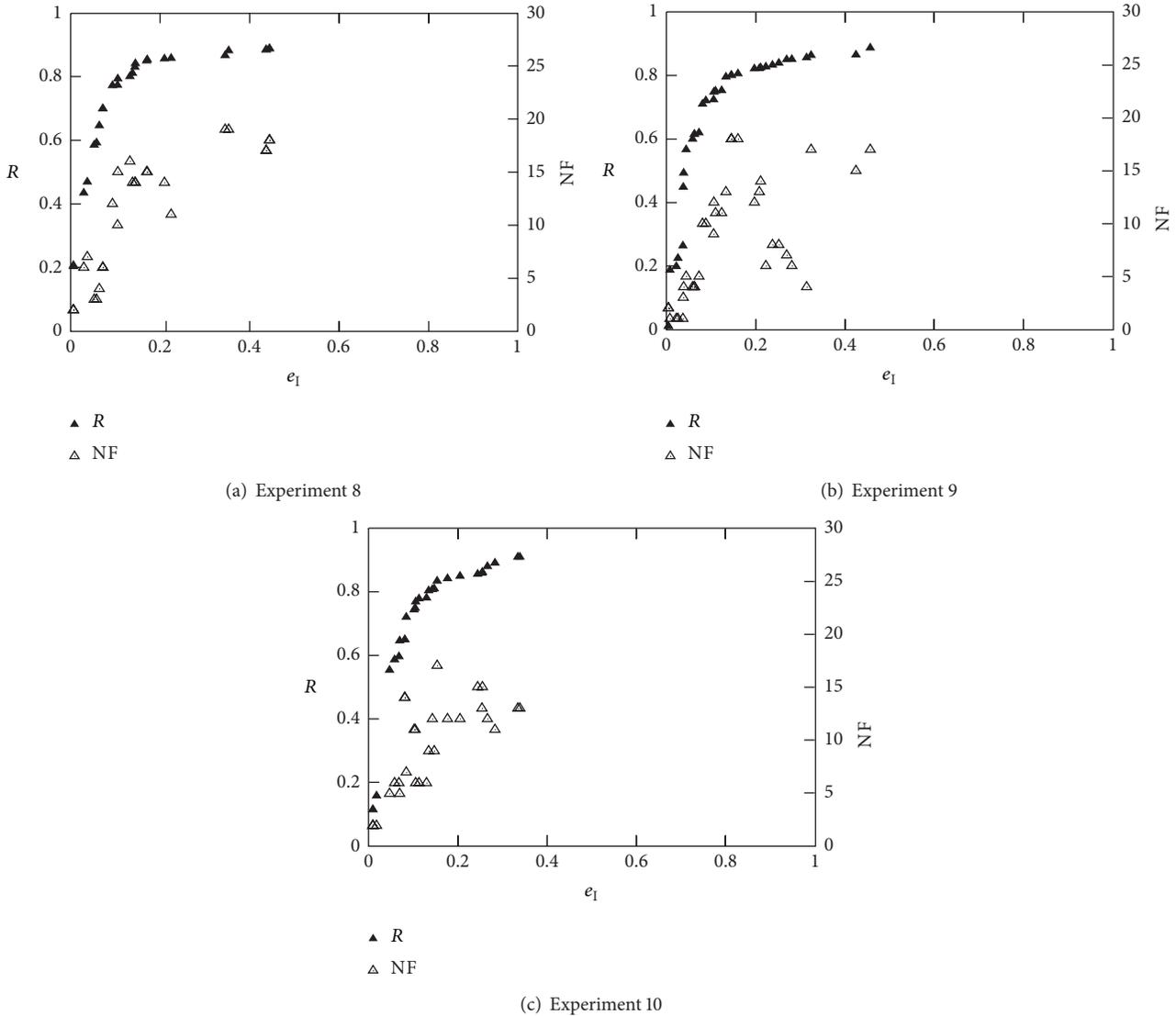


FIGURE 11: ROC curves for Diane 05 data (best of 10 runs).

evaluated using real-world benchmark problem datasets for bankruptcy prediction. A large set of experiments using different objective functions, such as accuracies, error, and sensitivity measures, have been performed which provides a better understanding on the application of SVMs to real-world data. The performance of the proposed method was also studied using two- and three-objective approaches.

Results have shown that the method performs well using the benchmark datasets studied. Large accuracies have been obtained using a significantly reduced subset of features. Consequently, the more the considered features, the larger the accuracies. Also, being a multi-objective technique, instead of a single solution, a set of nondominated solutions is provided which may help the decision maker to evaluate the trade-off in making a sacrifice in one of the objective functions towards obtaining a gain in some others. The inability for the

classifier to handle nominal features within the data turned out to be the main limitation of the proposed method. This limitation was inherent to the classifier used by the method; it was overcome by converting nominal attributes of the data to numerical.

A possible extension to this work could be made by taking advantage of the multi-objective nature of the set of solutions and analysing in detail the trade-off between them, thus helping decision makers to choose the preferred solution for their needs. The proposed method could also be extended to work with many objectives as real-world situations actually do.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

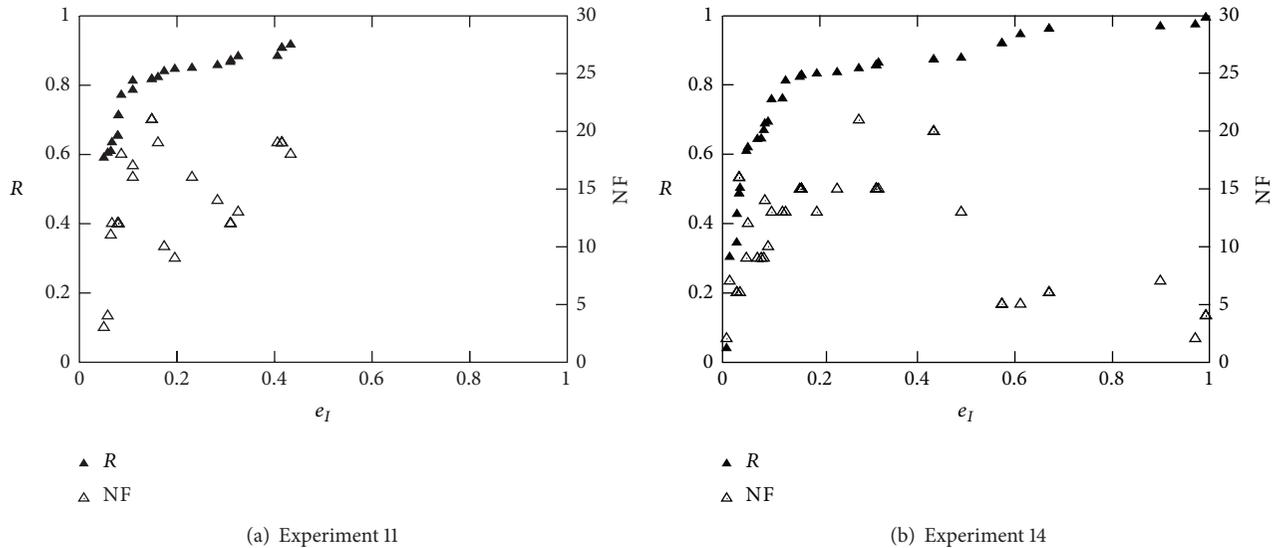


FIGURE 12: ROC curves for Diane 05 data (best of 10 runs).

Acknowledgments

This work was partially supported by the Portuguese Foundation for Science and Technology under Grant PEST-C/CTM/LA0025/2011 (Strategic Project-LA 25-2011-2012) and by the Spanish Ministerio de Ciencia e Innovación, under the project “Gestión de movilidad eficiente y sostenible, MOVES” with Grant Reference TIN2011-28336.

References

- [1] E. L. Altman, *Corporate Financial Distress and Bankruptcy*, John Wiley & Son, 1993.
- [2] C. Pate, *Bankruptcies: Recent Level and Implications for 2002*, chapter 11, Pricewaterhouse Coopers, 2002.
- [3] W. Beaver, “Alternative accounting measures as predictors of failure,” *The Accounting Review*, vol. 1, no. 1, pp. 113–122, 1968.
- [4] D. Martin, “Early warning of bank failure, a logit regression approach,” *Journal of Banking and Finance*, vol. 1, no. 3, pp. 249–276, 1977.
- [5] J. A. Ohlson, “Financial ratios and the probabilistic prediction of bankruptcy,” *Journal of Accounting Research*, vol. 18, pp. 109–131, 1980.
- [6] S. Balcaen and H. Ooghe, “35 years of studies on business failure: an overview of the classic statistical methodologies and their related problems,” *The British Accounting Review*, vol. 38, no. 1, pp. 63–93, 2006.
- [7] P. P. M. Pompe and A. J. Feelders, “Using machine learning, neural networks, and statistics to predict corporate bankruptcy,” *Microcomputers in Civil Engineering*, vol. 12, no. 4, pp. 267–276, 1997.
- [8] G. Zhang, M. Y. Hu, B. E. Patuwo, and D. C. Indro, “Artificial neural networks in bankruptcy prediction: general framework and cross-validation analysis,” *European Journal of Operational Research*, vol. 116, no. 1, pp. 16–32, 1999.
- [9] K.-S. Shin, T. S. Lee, and H.-J. Kim, “An application of support vector machines in bankruptcy prediction model,” *Expert Systems with Applications*, vol. 28, no. 1, pp. 127–135, 2005.
- [10] K. Y. Tam, “Neural network models and the prediction of bank bankruptcy,” *Omega*, vol. 19, no. 5, pp. 429–445, 1991.
- [11] W. L. Tung, C. Quek, and P. Cheng, “GenSo-EWS: a novel neural-fuzzy based early warning system for predicting bank failures,” *Neural Networks*, vol. 17, no. 4, pp. 567–587, 2004.
- [12] A. Tsakonas, G. Dounias, M. Doumpos, and C. Zopounidis, “Bankruptcy prediction with neural logic networks by means of grammar-guided genetic programming,” *Expert Systems with Applications*, vol. 30, no. 3, pp. 449–461, 2006.
- [13] S. Wang, L. Wu, Y. Zhang, and Z. Zhou, “Ant colony algorithm used for bankruptcy prediction,” in *Proceedings of the 2nd International Symposium on Information Science and Engineering (ISISE '09)*, pp. 137–139, December 2009.
- [14] L. Rui, “A particle swarm optimized Fuzzy Neural Network for bankruptcy prediction,” in *Proceedings of the International Conference on Future Information Technology and Management Engineering (FITME '10)*, pp. 557–560, October 2010.
- [15] K.-S. Shin, T. S. Lee, and H.-J. Kim, “An application of support vector machines in bankruptcy prediction model,” *Expert Systems with Applications*, vol. 28, no. 1, pp. 127–135, 2005.
- [16] J. H. Min and Y.-C. Lee, “Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters,” *Expert Systems with Applications*, vol. 28, no. 4, pp. 603–614, 2005.
- [17] A. Fan and M. Palaniswami, “Selecting bankruptcy predictors using a support vector machine approach,” in *Proceedings of the International Joint Conference on Neural Networks*, vol. 6, pp. 354–359, July 2000.
- [18] B. Baesens, S. Viaene, T. van Gestel et al., “Bankruptcy prediction with least squares support vector machine classifiers,” in *Proceedings of the IEEE International Conference on Computational Intelligence for Financial Engineering*, pp. 1–8, September 2000.
- [19] J. Jayanthi, J. K. Suresh, and J. Vaishnavi, “Bankruptcy prediction using svm and hybrid svm survey,” *International Journal of Computer Applications*, vol. 34, no. 7, pp. 39–45, 2011.
- [20] M. J. Martín-Bautista and M. A. Vila, “A survey of genetic feature selection in mining issues,” in *Proceedings of the Congress*

- on *Evolutionary Computation (CEC '99)*, vol. 2, pp. 306–313, 1999.
- [21] M. Kudo and J. Sklansky, “A comparative evaluation of medium and largescale feature selectors for ptttern classifiers,” in *Proceedings of the 1st International Workshop on Statistical Techniques in Pattern Recognition*, pp. 91–96, 1997.
- [22] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.
- [23] C.-F. Tsai, “Feature selection in bankruptcy prediction,” *Knowledge-Based Systems*, vol. 22, no. 2, pp. 120–127, 2009.
- [24] F. Duran, C. Cotta, and A. Fernandez, “On the use of sharpe’s index in evolutionary portfolio optimization under Markowitz’s model,” in *Proceedings of the Conention on Adaptive and Emergent Behaviour and Complex Systems*, 2009.
- [25] P. Skolpadungket, K. Dahal, and N. Harnpornchai, “Portfolio optimization using multi-objective genetic algorithms,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 516–523, September 2007.
- [26] L. Dioşan, “A multi-objective evolutionary approach to the portfolio optimization problem,” in *Proceeding of the International Conference on Computational Intelligence for Modelling, Control ad Automation*, vol. 2, pp. 183–187, November 2005.
- [27] S. Mullei and P. Beling, “Hybrid evolutionary algorithms for a multiobjective financial problem,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, pp. 3925–3930, October 1998.
- [28] F. Schlottmann and D. Seese, “Hybrid multi-objective evolutionary computation of constrained downside risk-return efficeient sets for credit portfolio,” in *Proceedings of the 8th International Conference of the Society for Computational Economics, Computing in Economics and Finances*, 2002.
- [29] A. Mukerjee, R. Biswas, K. Deb, and A. Mathur, “Multi-objective evolutionary algorithms for the risk-return trade-off in bankloan management,” *International Transactions in Operational Research*, vol. 9, no. 5, pp. 583–597, 2002.
- [30] S. Mardle, S. Pascoe, and M. Tamiz, “An investigation of genetic algorithms forthe optimization of multiobjective fisheries bioeconomic models,” *International Transactions in Operational Research*, vol. 7, no. 1, pp. 33–49, 2000.
- [31] A. Gaspar-Cunha, F. Mendes, J. A. Duarte, A. Vieira, B. Ribeiro, and J. A. Neves, “Multi-objective evolutionary algorithms for feature selection: application in bankruptcy prediction,” in *Proceedings of the 8th International Conference on Simulated Evolution and Learning*, pp. 319–328, 2010.
- [32] S. Meyer-Nieberg and H.-G. Beyer, “Self-adaptation in evolutionary algorithms,” *History*, vol. 75, pp. 47–75, 2007.
- [33] T. Bäck, “Self-adaptation,” in *Handbook of Evolutionary Computation*, pp. C7.1:1–C7.1:15, Oxford University Press, 1997.
- [34] S. Zeng, Z. Liu, C. Li, Q. Zhang, and W. Wang, “An evolutionary algorithm and its application in antenna design,” *Journal of Bioinformatics and Intelligent Control*, vol. 2, no. 1, pp. 129–137, 2012.
- [35] M. P. Poland, C. D. Nugent, H. Wang, and L. M. Chen, “Genetic algorithm and pure random search for exosensor distribution optimisation,” *International Journal of Bio-Inspired Computation*, vol. 6, no. 4, pp. 359–372, 2012.
- [36] A. F. Sheta, P. Rausch, and A. S. Al-Afeef, “A monitoring and control framework for lost foam casting manufacturing processes using genetic programming,” *International Journal of Bio-Inspired Computation*, vol. 2, no. 4, pp. 111–118, 2012.
- [37] J. Muñuzuri, P. C. Achedad, M. Rodríguez, and R. Grosso, “Use of a genetic algorithm for building efficient choice designs,” *International Journal of Bio-Inspired Computation*, vol. 1, no. 4, pp. 27–32, 2012.
- [38] B. B. Pal, D. Chakraborti, P. Biswas, and A. Mukhopadhyay, “An applicationof genetic algorithm method for solving patrol manpower deployment problems through fuzzy goal programming in traffic management system: a case study,” *International Journal of Bio-Inspired Computation*, vol. 1, no. 4, pp. 47–60, 2012.
- [39] L. Carlos Molina, L. Belanche, and À. Nebot, “Feature selection algorithms: a survey and experimental evaluation,” in *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM '02)*, pp. 306–313, December 2002.
- [40] R. Kohavi and G. H. John, “Wrappers for feature subset selection,” *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.
- [41] F. Provost and T. Fawcet, “Analysis and verification of classifier performance: classification under imprecise class and cost distributions,” in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD '97)*, pp. 43–48, 1997.
- [42] T. Fawcet, “An introduction to roc analysis,” *Pattern Recognition Letters*, vol. 27, pp. 861–874, 2006.
- [43] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: an update,” *SIGKDD Explorations*, vol. 1, no. 1, 2009.
- [44] X. Wu, “Top 10 algorithms in data mining,” *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2007.
- [45] J. Shawe-Taylor and N. Cristianini, *Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, 2000.
- [46] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [47] C. C. Chang and C. J. Lin, “Libsvm a library for support vector machines,” 2000, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [48] D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, 2001.
- [49] A. Carlos Coello, B. Gary Lamont, and D. A. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*, Springer, New York, NY, USA, 2006.
- [50] A. Gaspar-Cunha, “RPSGAe-reduced pareto set genetic algorithm: application to polymer extrusion,” in *Metaheuristics for Multiobjective Optimisation*, pp. 221–249, Springer, 2004.
- [51] A. Gaspar-Cunha and J. A. Covas, “Robustness in multi-objective optimization using evolutionary algorithms,” *Computational Optimization and Applications*, vol. 39, no. 1, pp. 75–96, 2008.
- [52] B. Ribeiro, C. Silva, A. Vieira, A. Gaspar-Cunha, and J. C. das Neves, “Financial distress model prediction using SVM+,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '10)*, pp. 1–7, July 2010.
- [53] B. Ribeiro, N. Lopes, and C. Silva, “High-performance bankruptcy prediction model using Graphics Processing Units,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '10)*, pp. 1–7, July 2010.
- [54] A. Frank and A. Asuncion, “UCI machine learning repository,” 2010, <http://archive.ics.uci.edu/ml>.
- [55] K. Wang, S. Zhou, A. W. Fu, and J. X. Yu, “Mining changes of classification by correspondence tracing,” in *SDM*, 2003.

- [56] J. A. Gonzalez, L. B. Holder, and D. J. Cook, "Graph based concept learning," in *AAAI/IAAI*, 2000.
- [57] J. R. Quinlan, "Simplifying decision trees," *International Journal of Man-Machine Studies*, vol. 27, no. 3, pp. 221–234, 1987.
- [58] M. A. Hal, *Correlation-based feature selection for machine learning [Ph.D. thesis]*, Department of Computer Science, University of Waikato, Waikato, New Zealand, 1999.
- [59] K. Deb and A. Raji Reddy, "Reliable classification of two-class cancer data using evolutionary algorithms," *BioSystems*, vol. 72, no. 1-2, pp. 111–129, 2003.
- [60] A. Bagirov, A. Rubinov, N. Soukhoroukova, and J. Yearwood, "Unsupervised and supervised data classification via nonsmooth and global optimization," *TOP*, vol. 11, pp. 1–75, 2003.
- [61] A. Vieira, J. Duarte, B. Ribeiro, and J. Neves, "Accurate prediction of financial distress of companies with machine learning algorithms," in *Adaptive and Natural Computing Algorithms*, M. Kolehmainen, P. Toivanen, and B. Beliczynski, Eds., vol. 5495 of *Lecture Notes in Computer Science*, pp. 569–576, Springer, Berlin, Germany, 2009.
- [62] N. Kwak and C.-H. Choi, "Input feature selection for classification problems," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 143–159, 2002.

Research Article

A Hybrid Evolutionary Algorithm for Wheat Blending Problem

Xiang Li,¹ Mohammad Reza Bonyadi,¹ Zbigniew Michalewicz,^{1,2,3} and Luigi Barone⁴

¹ School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia

² Institute of Computer Science, Polish Academy of Sciences, Ulica Orłona 21, 01-237 Warsaw, Poland

³ Polish-Japanese Institute of Information Technology, Ulica Koszykowa 86, 02-008 Warsaw, Poland

⁴ SolvelT Software, 99 Frome Street, Adelaide, SA 5000, Australia

Correspondence should be addressed to Xiang Li; xiang.li01@adelaide.edu.au

Received 9 November 2013; Accepted 30 December 2013; Published 20 February 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 Xiang Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents a hybrid evolutionary algorithm to deal with the wheat blending problem. The unique constraints of this problem make many existing algorithms fail: either they do not generate acceptable results or they are not able to complete optimization within the required time. The proposed algorithm starts with a filtering process that follows predefined rules to reduce the search space. Then the linear-relaxed version of the problem is solved using a standard linear programming algorithm. The result is used in conjunction with a solution generated by a heuristic method to generate an initial solution. After that, a hybrid of an evolutionary algorithm, a heuristic method, and a linear programming solver is used to improve the quality of the solution. A local search based posttuning method is also incorporated into the algorithm. The proposed algorithm has been tested on artificial test cases and also real data from past years. Results show that the algorithm is able to find quality results in all cases and outperforms the existing method in terms of both quality and speed.

1. Introduction

Wheat is Australia's most important grain crop. About 80 percentage of Australia's wheat is exported. Australia is the world's fourth-largest exporter of wheat. Usually, wheat is sold to the central collection sites via truck in batches, called *loads*. When submitted, each load is weighted and sampled and the result of quality checks is given. There are 10 to 20 attributes, such as protein content and moisture, that are checked. A grade is assigned to each load according to the result of the quality check. This grade is used to deliver products (wheat) within given specifications. There are 26 grades in total; each one has its own quality requirements and price. The value of the wheat is determined by its grade (see Table 1).

For example, consider grades G1 and G2 (for simplicity, only the protein content is presented here). To be graded as G1, the protein content of wheat must be within the 11.0%–12.5% range, and for G2 the range is from 10% to 11.0%. G1 has a higher requirement on protein and has a higher price.

Now let us consider three loads of wheat and how they will to be graded.

As shown in Table 2, L1 (with 11.5% of protein) is graded as G1, L2 (with 10.5% of protein) is graded as G2, and L3 (with 10.0% of protein) is graded as G2. Note that, although L1, L2 have a higher protein value than the required lower bounds, the price does not increase.

In fact, there are many cases where the quality of wheat is above the minimum requirement or cases where the wheat is just short of obtaining a higher grade. One way to improve the overall value is to *blend* the wheat.

Blending is the process of mixing wheat of different qualities. This is usually done by blending low-quality (low price) wheat with some high-quality wheat to achieve a better overall value. Blending is a vital part of the entire wheat supply chain and, as discussed below, plays a major role in generating profit.

By blending different loads, the mixture (called a *lot*) could be assigned with a new grade based on the weighted average quality result.

TABLE 1

Grade	Protein lower bound	Protein upper bound	Price per tonne
G1	11.0%	12.5%	\$240
G2	10.0%	11.0%	\$220

TABLE 2

Load	Protein	Grade	Price per tonne	Tonne
L1	11.5%	G1	\$240	100
L2	10.5%	G2	\$220	100
L3	10.0%	G2	\$220	80

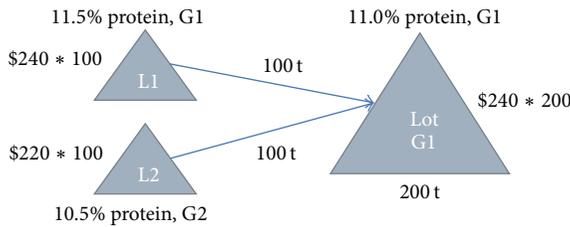


FIGURE 1

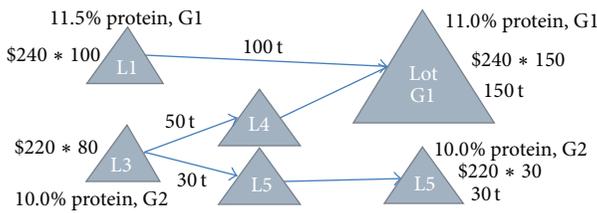


FIGURE 2

Figures 1 and 2 present two examples to illustrate the basics of blending.

In Figure 1, there are two loads, L1 and L2. L1 is 100 tonnes, with a protein percentage of 11.5% that would be graded as G1. L2 is 100 tonnes, with a protein percentage of 10.5% that would be graded as G2. The price of G1 is \$240 per tonne and G2 is \$220 per tonne.

Suppose that the requirement of G1 is to have at least 11.0% of protein. Clearly, L1 exceeds the protein requirement of G1 (with no additional benefit) and can be mixed with L2 to achieve a better total value. If L1 and L2 are blended together, the mixed lot will have a protein percentage of 11.0% and thus still meet the requirement of G1. This results in an increase of total value: the value before blending sums to \$46,000 and the value after blending is \$48,000, realising an *uplift* of \$2,000.

Figure 2 represents a more complicated example. There are two loads, L1 and L3. L1 is 100 tonnes, with a protein percentage of 11.5% that would be graded as G1. L3 is 80 tonnes, with a protein percentage of 10.0% that would be graded as G2. The price of G1/G2 is still \$240/220 per tonne. Since L3 has less protein than L2, in this case, the blending of L1 and L3 no longer meets the protein requirement of G1.

Instead, we can split L3 into two subloads, L4 with 50 tonnes and L5 with 30 tonnes, and then blend L1 and L4 together to form a G1 lot. In this case, the value before blending sums to \$41,600 and the value after blending is \$42,600, increasing profit by \$1,000.

The growers in Australia do not actually do the physical blending work. However, they could sell their wheat at the blended price if a blending plan is provided. Thus, the result of a provided blending plan is directly related to their profit. In fact, for growers with hundreds of loads of wheat, the profit of blending can be easily beyond \$200,000.

However, building a good blending plan is a very complex task even for experts. As an example, in Australia, the grading standard includes up to 20 attributes (protein, moisture, screening, earth, etc.) and 2 unique constraints (discussed in Section 2) to determine the grade of wheat. Moreover, one individual grower might have more than 500 loads of wheat, all with different qualities. As a result, a good blending plan often needs hours or days of work.

During the harvest season, the prices of wheat change daily or even more frequently. As the price of wheat changes, the optimal way to blend also changes. This indicates that not only the quality of the blending plan is important, but the time taken to generate that plan is also important. A good plan created after hours of work might be already outdated due to the changes of the price. There is not enough time to manually build a good blending plan every time the price changes. A tool which can generate the blending plan in a short period of time is in much demand.

This paper extends our previous work [1] where we proposed a linear programming guided hybrid evolutionary algorithm to address the wheat blending problem. The proposed algorithm is hybridized with an evolutionary algorithm, a heuristic algorithm, and a linear programming algorithm. In addition to these, a heuristic based initialization method is used to reduce the search space and a local search is also applied to fine-tune the final result. During the 2013 harvest season, the proposed algorithm helped thousands of growers build their blending plans and generate tens of millions dollars profit for the growers. In this paper, more experiment results are presented and detailed stage-by-stage performance analysis is included as well.

The rest of the paper is organized as follows. Section 2 introduces the blending problem in detail. Then Section 3

provides some background on related work for solving the underlined problem. The proposed hybrid evolutionary algorithm is described in Section 4. In Section 5, the proposed algorithm is applied to the test cases and the results of comparison with a heuristic algorithm in current use are provided. The impact of each stage is included in Section 6 and Section 7 concludes the paper.

2. Model of the Problem

Wheat is one of the most important agricultural commodities in Australia and is one of Australia's most valuable exports. Blending is an important stage in the whole wheat supply chain. Before milling, wheat with different levels of quality may be mixed together to balance the cost and quality. The price of wheat is based on many quality attributes and some wheat may have higher quality values than required. In these cases, high-quality wheat can be blended with low-quality wheat to balance the quality, thereby having a better overall value.

This problem can be described by the following model. Assume that L is the number of loads, G is the number of grades, l represents a load, and g represents a grade. Also, S_L, S_G, P_L, P_G represent the set of all loads, the set of all grades, the price per tonne of load l , and the price per tonne of grade g wheat, respectively. Consider that t is the decision variables vector defined by $t_{l,g}$ which is the number of tonnes from load l that has been blended into grade g lot. The objective of this blending problem is to

$$\begin{aligned} &\text{find } t \in R^L \times R^G \text{ such that} \\ &\text{maximize } \sum_{l \in S_L} \left(\sum_{g \in S_G} (P_G - P_L) t_{l,g} \right). \end{aligned} \quad (1)$$

In (1), $(P_G - P_L)$ is the earned profit when load l is blended into a lot with grade g . This refers to the fact that maximizing the profit generated by blending is desirable.

Then, M_l is the number of tonnes of load l originally and (2) and (3) indicate that the total tonnes of load l used in blending should always be greater than or equal to 0 and less than or equal to its original tonne weight:

$$\sum_{g \in S_G} t_{l,g} \geq 0 \quad \forall l, \quad (2)$$

$$\sum_{g \in S_G} t_{l,g} \leq M_l \quad \forall l. \quad (3)$$

Equations (4) and (5) are the constraints on the quality standards of each grade. q represents one quality attribute, for example, the protein percentage. $Q_{q,l}, Q_{-Max_{q,g}}$, and $Q_{-Min_{q,g}}$ are the quality attribute q of load l , the maximum requirement of quality attribute q for grade g , and the minimum requirement of quality attribute q for grade g , respectively. The weighted average result of quality attribute q

for the blended lot with grade g should always be within the min/max range:

$$\sum_{l \in S_L} (Q_{q,l} t_{l,g}) \leq Q_{-Max_{q,g}} \sum_{l \in S_L} (t_{l,g}) \quad \forall g, q, \quad (4)$$

$$\sum_{l \in S_L} (Q_{q,l} t_{l,g}) \geq Q_{-Min_{q,g}} \sum_{l \in S_L} (t_{l,g}) \quad \forall g, q. \quad (5)$$

Linear constraints usually cannot model real-world problems precisely. As in our problem, there are two nonlinear constraints involved which makes the problem quite unique.

Firstly, the Australian standards suggest that the weighting of wheat is precise down to the 10 kilo range. Thus $100t_{l,g}$ is required as an integer vector since it used tonnes based weighting. This hard constraint corresponds to

$$100t_{l,g} \text{ is integer } \quad \forall l, g. \quad (6)$$

There is also one more constraint which further complicates the problem. As proposed, it is possible to just take a part from load l to use in blending, known as a *split*. However, the total number of splits allowed for the entire blending plan is limited, and this may differ from grower to grower. This constraint is included in

$$\sum_{l \in S_L} \left(\sum_{g \in S_G} \left\lceil \frac{t_{l,g}}{M_l} \right\rceil - 1 \right) \leq S, \quad (7)$$

where S is the number of splits allowed and $\lceil x \rceil$ is the ceiling function which returns the smallest integer not less than x .

3. Related Work

In this section, related algorithms for solving the general blending problem are detailed and a brief introduction to the epsilon level constraint handling is included.

3.1. Linear Programming. Linear programming (LP) is an optimization technique that has been designed for addressing continuous space (decision variables are continuous) optimization problems. LP requires that the objective function and constraints are all linear and LP algorithms are able to solve such optimization problem to optimality. There are many methods to solve linear programming problems such as simplex, criss-cross, and interior point methods [2].

In this blending problem, the objective function (1) and constraints (2), (3), (4), and (5) are all linear. Thus the linear relaxed version, which only considers (1) to (5), can be solved efficiently using a linear programming algorithm. There have been a few attempts to solve similar blending problems (with only linear constraints) using linear programming algorithms, especially before the early 1990s [3].

However, for this problem, (6) and (7) affect the model significantly. The result from a linear-relaxed model might break either or both of the constraints. Firstly, linear programming is operated in the continuous space; thus there is no guarantee that the result is feasible for (6). Secondly, the result might use any number of splits, which breaks (7). Both

of the constraints are important for business. Constraint (6) is clearly stated in the Australian standards and (7) comes from the capacity limitation for the shared storage space. In addition, (7) is also used by the business to control hidden operational cost.

We can partially solve the problem of (6) by rounding the results. A simple half-up rounding will do the job but then the result is no longer guaranteed to satisfy all the constraints from (2) to (5). During our experiments, there are around 30% of the cases in which the result after the half-up rounding is still feasible. Some heuristic based rounding methods could increase the chance to 60%, but those methods are computationally expensive and are not the focus of this paper.

Again, we can use rounding (if the variable representation is transformed from $t_{i,g}$ to $(t_{i,g}/M_i)$ and capping the cases where the value is rounded up to 1) to solve the problem of (7). However, the variations needed are significant and feasibility of the solution is not guaranteed either. The result is also quite possibly a suboptimal solution, since those extra splits used usually contribute major sources of profit.

3.2. Mixed Integer Programming. Blending problems are also often modelled as mixed integer programming problems, especially for real-world cases [4]. Integer programming (IP) is a type of linear programming in which decision variables are integers and mixed integer linear programming (MILP) is a variety where only some of the variables are constrained to be integers. There are different methods for IP/MILP: some are exact (such as the methods which use branch and bound or cutting plane) and some are approximation methods. In the exact methods, normally the relaxed version of the problem is solved by LP and then this information is used (e.g., in branch and bound) to find optimal solutions. However, the time complexity of these methods is exponential [3].

There are many studies using exact algorithms to solve blending problems. Bilgen and Ozkarahan proposed a mixed-integer linear programming model for optimizing a wheat supply chain. The objective is to minimize the total cost for blending, loading, transportation, and storage [5]. Ashayeri et al. apply the model to the blending of chemical fertilizers [6]. Jia and Ierapetritou also use a mixed-integer linear programming model to optimize the blending of gasoline [7]. The MILP model is also used in the blending of water [8] and oil [9].

MILP could model the problem more precisely than a LP since (6) is not relaxed. However, (7) is still not solved. In addition, the execution speed is limiting the usage of exact methods in here. One grower could have up to 700 loads and it may need days of time for those exact algorithms to finish. Thus, those exact methods are not applicable for this problem. Actually, unlike academic researchers, real-world users are usually more concerned of the speed of the tool, instead of the optimality of the solution. Users might be happy to have a cup of coffee while waiting for the result, but, in general, waiting for hours is not acceptable, especially in decision support systems. As a rule of thumb, a casual user usually prefers a tool that is fast and generates a quality result, but not necessarily the optimal result.

3.3. Metaheuristic. Metaheuristic algorithms are also a popular choice for solving complex mixed-integer programming problems [10]. Examples include applying evolution strategy to the problem of optimal multilayer coating design [11] and to optimize chemical engineering plants [12]. Other cases include an ant colony system for optimizing electrical power distribution networks [13], a genetic algorithm to optimize the design of antenna [14], to optimize the deployment of patrol manpower [15], and to optimize exosensor distribution for smart home systems [16]. Yokota et al. proposed a genetic algorithm to solve nonlinear mixed integer programming [17] and there are many other algorithms created for solving the general MILP [18–20].

However, those algorithms are either too general or too specific for the underlying problems. To obtain the best result, real-world constraints like (7) usually need specially designed methods and intense tuning [21].

3.4. Evolutionary Algorithm. An evolutionary algorithm (EA) is a stochastic population-based metaheuristic that mimics biologically inspired operators such as *mutation*, *recombination*, and *selection*. In an EA, a set (known as the *population*) of initially generated solution candidates (known as *individuals*) is processed (generations: the main loop of the EA). In each generation, a subset of the individuals in the population is selected (via the selection operator, to mimic the competition between individuals). The selected individuals are then modified (via the mutation and/or recombination operator), resulting in a new set of individuals. This subset is merged into the original population and after a selection process (to mimic the “survival of the fittest” process), a new population is generated. This process is repeated until a certain termination criterion is met (such as reaching the maximum generation limit or the solution is not being improved for a long time) [22].

In many practical cases, it has been reported that hybridizing an EA with other methods is effective [23]. There are many ways to hybridize an EA with other methods. For example, one way would be to incorporate with other methods to create problem dependent operators [24]. Another way would be to apply another method to improve the final solutions found by the EA [25]; It is also possible to use problem specific representation [26] or to run one or more algorithms interactively [27].

3.5. Heuristic Algorithm. One existing tool has been used by growers to help them build the blending plan, and it uses a heuristic based algorithm. The heuristic is based on the fact that protein percentage is the main attribute to differentiate grades. Thus the algorithm tries to find a load that has the best (profit/protein cost) ratio. If given a load l and a target lot with grade g , the ratio can be calculated as

$$\frac{P \cdot G_g - P \cdot L_l}{Q \cdot \text{Min}_{\text{protein},g} - Q_{\text{protein},l}}, \quad (8)$$

where $P \cdot G_g$ is the unit price of grade g wheat, $P \cdot L_l$ is the unit price of the load l , $Q \cdot \text{Min}_{\text{protein},g}$ is the minimum protein

```

hasNext = true
While hasNext
    hasNext = false
    loadtarget, lottarget = SELECT_BY_BEST_RATIO(L)
    blends = COMBINATIONS(L, loadtarget, lottarget)
    Blendbest = null
    for each blend in blends
        if NOT_VALID(blend)
            continue
        end if
        if NO_PROFIT(blend)
            continue
        end if
        If blendbest or blend is better than blendbest
            blendbest = blend
        end if
    end for
    if blendbest is not null
        APPLY(blendbest)
        hasNext = true
    end if
end while
    
```

ALGORITHM 1: Heuristic algorithm.

requirement of grade g , and $Q_{\text{protein},l}$ is the protein percentage of load l respectively.

After that, the algorithm tries to find one or more companion loads which have better quality attributes to improve the weighted average quality. The combination of the selected load, the target lot, and the companion loads is called a *blend*. The algorithm stops if it cannot find any blend with profit.

The method used to find the companion loads is to do an exhaustive search with all the combination of 3 (or less) loads. The whole process is summarized as in Algorithm 1. The NOT_VALID method tests whether any constraint violation is introduced. The NO_PROFIT method tests whether any profit is generated.

This algorithm was a lot faster than doing the blending manually, but the generated result was often suboptimal. This tool could solve some simple problems but, for more complex cases, the user typically used the tool to generate a base solution and then tweaked it to get a better result (in fact, our proposed algorithm follows the same ways as the users. It generates an initial solution first and then tweaks it to get a better result). The users were generally happy with the tool but always seek for a better tool that could generate a quality blending plan all the time while keeping the execution time short.

3.6. *Epsilon Level Constraint Handling*. Epsilon (ϵ) level constraint handling (ϵ LCH) is a method that transforms the constrained optimization problems into unconstrained problems [28]. The transformation is done by replacing the ordinary comparison operator by the ϵ level comparison operator. The ϵ level comparison operator combines the

constraint violation values and objective values for evaluating candidate solutions.

In short, the ϵ level comparison compares two solutions by their constraint violation values first. The solution which has a lower constraint violation value is ranked higher. However, if both the violation values are under a small threshold ϵ , then the constraint violation values are ignored, and the two solutions are only compared by their objective function values.

Suppose there are two candidate solutions x_1 and x_2 , f_1 and f_2 are the objective values, and h_1 and h_2 are constraint violation values of x_1 and x_2 ; then the ϵ level comparison operator $<_\epsilon$ and \leq_ϵ is defined by the following:

$$\begin{aligned}
 x_1 <_\epsilon x_2 &\equiv \begin{cases} f_1 < f_2 & \text{if } h_1, h_2 \leq \epsilon \\ f_1 < f_2 & \text{if } h_1 = h_2 \\ h_1 < h_2 & \text{otherwise,} \end{cases} \\
 x_1 \leq_\epsilon x_2 &\equiv \begin{cases} f_1 \leq f_2 & \text{if } h_1, h_2 \leq \epsilon \\ f_1 \leq f_2 & \text{if } h_1 = h_2 \\ h_1 \leq h_2 & \text{otherwise.} \end{cases}
 \end{aligned}
 \tag{9}$$

There are many ways to control the threshold ϵ . The formula used by the proposed algorithm is included in Section 4.

4. The Proposed Hybrid Evolutionary Algorithm

The proposed algorithm contains four stages: search space reduction, initialization, evolutionary loop, and local search. The working flow is shown in Algorithm 2.

```

BEGIN
  search space reduction(Stage 1)
  initialization(Stage 2)
  while not terminated
    mutation(Stage 3-1)
    heuristic(Stage 3-2)
    simplex(Stage 3-3)
  end while
  local search(Stage 4)
END ALGORITHM

```

ALGORITHM 2

In Sections 4.1 to 4.4, each of those stages is presented. Firstly, the algorithm tries to eliminate all the obvious bad choices using predefined rules. Then the algorithm solves the linear-relaxed version of the problem and uses the result as a clue to build an initial solution of the nonrelaxed version. After that, the algorithm tries to tweak the solution in an iterative fashion. In each iteration, there is an evolutionary algorithm to optimize the loads to blend, a heuristic to choose the right loads to split, and the use of a linear programming algorithm to find the optimal way to split. Final tune-up is done by a local search.

Additionally, Section 4.5 introduces a specially designed constraint handling method that is proposed into the algorithm to encourage the exploration of infeasible regions. Local search in the main loop is included in Section 4.6.

4.1. Search Space Reduction. In this stage, the algorithm tries to eliminate some obvious bad choices before it starts the stochastic process. This is done by a rule-based filtering process. These rules are based on advice from domain experts and experimental results. Some rules include the following.

- (i) Never blend a load to a lot which requires at least another 2% of protein. As the protein percentage is generally from 10% to 14%, overcoming the 2% margin is too costly.
- (ii) Never blend a load that has an extra 1.5% of protein above the grade requirement, unless there are only few choices. This rule attempts to save the good quality loads for a better global result.

After this filtering process, the search space (number of possible blends) of the problem is greatly reduced.

The key of this stage is to ensure that there is no bad choice made. To do so, the thresholds of the rules are carefully chosen. Those values could almost guarantee that it does not have a negative impact on finding the optimal solution.

4.2. Initialization. Since the execution speed is crucial for this problem, the algorithm uses a heuristic based initialization method instead of any random initialization method. This might sacrificed the diversity of solutions but the algorithm could get a good basic solution with the least computation.

The algorithm starts with applying the simplex algorithm [3] to solve a linear-relaxed version of the problem. The

linear-relaxed version is the same problem but only consider constraints (2) to (5). Then the algorithm uses the heuristic (8) to build a solution of the nonrelaxed problem, but with a threshold of 15. Only loads that have the profit-protein ratio greater than 15 are considered. After that, the algorithm extracts the common parts from both solutions and generates an initial solution based on them.

The threshold value 15 is a very high number for the profit-protein ratio. This is to ensure that the algorithm is not too greedy at the beginning. The simplex result is used to double check that and also serves as a clue for reaching the global optimal. The decisions made in this step are then fixed, not possible to modify by the latter stages.

The purpose of this stage is to generate a basic solution with no or few bad choices and further reduce the search space. Since we have chosen a very high threshold number, we can ensure that the decisions are all obvious good ones.

4.3. Evolutionary Loop. This is the main loop where the new solutions are generated. It contains an evolutionary algorithm to optimize the loads to blend, a heuristic to choose the right loads to split, and the use of a linear programming algorithm to find the optimal way to split. The operators used in this stage are as follows.

- (i) Mutation: for a randomly selected load, change its allocation to a random lot.
- (ii) Heuristic: it is to choose which load to split. For all the possible combinations of load l and target lot with grade g , this applies the 2-way tournament selection to choose S combinations from the top $2S$ that have the best value of

$$\frac{P \cdot G_g - P \cdot L_l}{Q \cdot \text{Min}_{\text{protein},g} - Q_{\text{protein},l}} M_l. \quad (10)$$

- (iii) Simplex algorithm: the selected loads in the heuristic step form a subproblem and the problem is solved with unlimited splits allowed using the simplex algorithm.

In each iteration, the algorithm tries to modify the existing solution by the mutation operator one or more times (by some probabilities). The probabilities are to ensure that

```

input:  $n, p_m, p_l$ 
output:  $t_{best}$ 
SEARCH_SPACE_REDUCTION()
 $t_{init}$  = INITIALIZE()
 $t_{best}$  =  $t_{init}$ 
While none of termination condition was met
     $t_{base}$  =  $t_{best}$ 
    for  $i = 1 : n$ 
         $t_{new1}$  = MUTATE( $t_{base}$ )
         $r$  = RANDOM(0,1)
        while  $r < p_m$ 
             $t_{new1}$  = MUTATE( $t_{new1}$ )
             $r$  = RANDOM(0,1)
        end while
         $t_{new2}$  = ROUND(SIMPLEX( $t_{new1}$ , GET_SPLIT( $t_{new1}$ )))
         $r$  = RANDOM(0,1)
        if  $r < p_l$  and  $t_{new2}$  is feasible
             $t_{new2}$  = LOCAL_SEARCH( $t_{new2}$ )
        end if
        if  $t_{new2}$  is better than  $t_{best}$ 
             $t_{best}$  =  $t_{new2}$ 
        end if
    end for
     $t_{best}$  = LOCAL_SEARCH_ALL( $t_{best}$ )
end while
    
```

ALGORITHM 3

the algorithm is possible to perform bigger variations. The generated new solution contains no split and is called *raw solution*. Then, the algorithm iterates over all the loads using the heuristic mentioned above and tries to find S good candidates to do the split. After that, the algorithm builds a linear-relaxed model with only the selected S loads and solves it using the simplex algorithm. The generated solution is called *split solution* and always satisfies (7) since there are no more than S variables in the model.

Thus, each solution has actually two forms: raw and split form. Note that the mutation only operates on the raw form and the simplex algorithm resulting in a solution in the split form. Also, the result of the simplex algorithm might not satisfy (6). In those cases, rounding is applied.

4.4. Local Search. Random modification is usually very inefficient when the result is close to the optimal point. It needs to be really lucky to find any improvement and it is often much more time consuming than doing an exhaustive search. Thus, a local search method is applied at the end to fine-tune the result. It tries to find all possible combinations that could give an increase of profit. The procedure is as follows.

- (i) For all possible combinations of load l and target lot with grade g , apply the one which gives the most profit until there are no combinations that could generate any profit.

4.5. Constraint Handling. As a highly constrained problem, the search space of this problem is generally separated by the

constraints into many isolated feasible regions. The simplex result from the initialization stage is used here to guide the search jumping out of a single feasible region. The idea is to de-penalise any blend that also can be found in the simplex result. Such blend might be a bad move by itself but is also possibly a vital part of a bigger profitable blend.

More detailedly, if any blend violates any of the constraints (2) to (5) and the same blend can be found in the linear-relaxed result, its constraint violation value is reduced. The formula used is

$$c_{new} = \begin{cases} 0.5c & \text{if } \left(\frac{S_L - S}{S}\right) \geq 0.5 \\ \left(\frac{S_L - S}{S}\right)c & \text{if } 0.5 \geq \left(\frac{S_L - S}{S}\right) \geq 0.1 \\ 0.1c & \text{otherwise,} \end{cases} \quad (11)$$

where c is the original constraint violation value, c_{new} is the reduced constraint violation value, S is the number of splits allowed, and S_L is the number of splits used by the simplex result, respectively.

Also, in this algorithm, solutions are compared using the ϵ level comparison operators. The value of ϵ is set according to the following equations:

$$\begin{aligned} \epsilon_0 &= h(x_0), \\ \epsilon_t &= \begin{cases} \epsilon_0 \left(1 - \frac{1.5t}{T-I}\right) & \text{if } 0 < 1.5t < T-I \\ 0 & \text{if } t \geq T-I, \end{cases} \end{aligned} \quad (12)$$

where ϵ_0 is the initial ϵ value, $h(x_0)$ is the constraint violation value for the best solution in the initialization step, ϵ_t is the ϵ

TABLE 3: Result of test cases with unlimited splits allowed.

Test case	Number of loads	Known best Splits used	Heuristic algorithm			Hybrid algorithm		
			Percentage to known best	Splits used	Time used (seconds)	Percentage to known best	Splits used	Time used (seconds)
R1	34	3	10.3	5	15.3	0	3	1.4
R2	145	2	17.9	3	11.9	0	2	2.3
R3	26	0	0	0	1.4	0	0	1.0
R4	332	1	0	0	6.5	0	0	5.9
R5	127	2	23.7	5	22.1	0	2	2.2
R6	718	2	36.5	46	732.4	0	3	49.9
R7	129	2	15.8	2	28.9	0	3	2.2
R8	610	5	20.4	22	139.7	1.2	6	25.3
R9	49	2	2.2	7	10.1	0	2	1.6
R10	47	2	9.6	3	14.5	0	2	2.9

TABLE 4: Result of test cases with 1 split allowed.

Test case	Number of loads	Heuristic algorithm			Hybrid algorithm			
		Percentage to upper bound	Splits used	Time used (seconds)	Percentage to upper bound	Splits used	Improvement over heuristic	Time used (seconds)
RS1	34	10.6	1	11.7	6.6	1	4.4	1.4
RS2	145	18.1	1	9.4	1.8	1	19.8	2.6
RS3	26	0	0	1.4	0	0	0	1.0
RS4	332	0	0	6.5	0	0	0	5.9
RS5	127	23	1	17.2	0.1	0	29.6	3.5
RS6	718	18.4	1	572.5	0.3	1	22.2	58.6
RS7	129	15.8	1	28.4	0.5	0	18.1	2.3
RS8	610	18.4	1	85.9	1.3	0	20.9	37.8
RS9	49	2.7	0	7.8	0.5	0	2.2	1.6
RS10	47	9.8	1	12.1	0.3	1	10.4	2.9

value at iteration t , and T_I is the iterations limit. This formula suggests that the methods will be focusing on finding feasible solutions when $1.5t \geq T_I$.

4.6. Local Search in the Evolutionary Loop. Within the main evolutionary loop, the generated solution also gets a chance to perform a single local search step and is used to speed up the convergence. Many different quality solutions are generated during the evolutionary loop and they are all good starting points for the local search. The algorithm only performs the local search by a single step. This is to ensure that the result is not suffering from premature convergence significantly.

4.7. Summary. The complete steps are shown in Algorithm 3. The parameters are

- (i) n : the number of offspring;
- (ii) p_m : the probability of applying additional mutation;
- (iii) p_l : the probability of applying one local search step within the evolutionary loop.

And the termination conditions are defined as

- (1) no improvement after T_I iterations;
- (2) total number of evaluations is over T_E .

5. Experimental Results

In this section, the proposed algorithm is applied to 20 selected real-world and 73 artificial test cases. All real-world test cases are created using the data from past years and should cover the most typical scenarios. The proposed algorithm is compared with the existing heuristic based algorithm here and the results were averaged over 20 runs for each test case.

5.1. Parameters Setting. The proposed algorithm has been implemented as a web service, running on distributed servers. To improve convergence, we always set the population size to 1 and use elitism selection. The main parameters in this experiment were set as follows:

- (i) $n = 7$,
- (ii) $p_m = 0.6$,

TABLE 5: Result of test cases (A1–A28).

Test case	Number of loads	Heuristic algorithm	Hybrid algorithm
A1	4	Pass	Pass
A2	7	Pass	Pass
A3	6	Pass	Pass
A4	6	Pass	Pass
A5	3	Pass	Pass
A6	5	Pass	Pass
A7	4	Pass	Pass
A8	6	Pass	Pass
A9	3	Pass	Pass
A10	3	Fail	Pass
A11	3	Pass	Pass
A12	3	Pass	Pass
A13	5	Pass	Pass
A14	3	Pass	Pass
A15	5	Pass	Pass
A16	4	Pass	Pass
A17	7	Pass	Pass
A18	5	Pass	Pass
A19	4	Fail	Pass
A20	5	Pass	Pass
A21	7	Pass	Pass
A22	7	Pass	Pass
A23	3	Fail	Pass
A24	6	Pass	Pass
A25	4	Fail	Pass
A26	4	Fail	Pass
A27	3	Pass	Pass
A28	4	Pass	Pass

- (iii) $p_l = 0.1$,
- (iv) $T_I = 100$,
- (v) $T_E = 50,000$.

The values of those parameters are selected manually. This set of parameters gives the best averaged result on the 10 real-world test cases (R1–R10, see Section 5.2).

A larger population size is also tested. It is completely applicable but there is no fundamental improvement up to the size of 4. After that, the execution time is increased significantly. In cases where the population size is more than 1, the 2-way tournament selection is used.

5.2. *Test Cases.* The 10 real-world test cases (R1–R10) are selected by domain experts, aiming to cover the most typical scenarios. The number of loads ranges from 26 to 718 in each case and is the dominant factor in the complexity. R8 is the largest case and quite possibly the most complex. R6 is a combined case (loads from two growers) to test the extreme scenario. The profit generated ranges from thousands to a

quarter-million dollars. Note that test cases R1–R10 do not have any limitation on the split allowed.

The result of the proposed algorithm is compared with the heuristic based tool in current use. The benchmark here is the known best results which are optimized manually by domain experts (supported by computer tools). The experts have spent weeks of time on those cases and they believe the results are good enough to be used as the benchmark.

Test cases RS1–RS10 are the same ones as R1–R10, but with only 1 split allowed. Those cases are more constrained and are harder (slower) to optimize. Note that there is no known best result in these cases (the proposed hybrid algorithm outperforms them). Instead, we use the known best result from R1–R10 to serve as the upper bound.

The 28 artificial tests (A1–A28) are simple test cases which contains many typical pitfalls. The number of loads ranges from 3 to 7. The first 20 tests (A1–A20) do not require any split to obtain the optimal solution but the rest (A21–A28) do.

There are 45 more artificial tests (AC1–AC45) which are pair-wise combination of the real-world test cases R1–R10. Those tests are more time consuming but also have more potential to optimize. The linear-relaxed result is served as the upper bound.

5.3. *Results.* Table 3 shows the result on cases where there are an unlimited allowed number of splits. With the split limit constraint relaxed, those cases are relatively easy. The proposed algorithm found a close-to-optimal result in all cases, while the heuristic algorithm only succeeded in the most simple cases. N8 is the only case where the hybrid algorithm is greater than 1% from the known-best result. In all cases, the hybrid algorithm is significantly faster.

Table 4 shows the result on cases with only 1 split allowed. In real-world cases, the split limit is normally set as 1 to 10 depending on the choice made by the user. The proposed algorithm still outperforms the heuristic algorithm in terms of both quality and speed, and the results are very close to the upper bound except for RS1. Note that for RS5, RS6, and RS8, the heuristic algorithm generates better results than the cases with an unlimited split allowed. This suggests that the heuristic algorithm can easily get stuck in local optima.

Table 5 shows the result of the artificial tests (A1–A28). The generated blending plan is required to be the same or equal-valued with the precalculated optimal result to be able to pass the test. The proposed algorithm passes all the tests while the heuristic fails on 5 cases.

Table 6 shows the result of the combination cases (AE1–AE45). Again, the proposed algorithm outperforms the heuristic algorithm. AE5, AE13, AE20, AE27, and AE44 are the only cases where the results are greater than 3% from the linear-relaxed upper bound. It also shows that the heuristic algorithm is rarely generating good solutions for large test cases (like AE26, AE28, AE33, AE36, AE37, etc.). This suggests that the heuristic algorithm might be too greedy at the beginning and cannot get out of the local optima. In contrast to this, the results from the proposed algorithm do not suffer much from a large number of loads. Additionally, the running time of the proposed algorithm grows significantly slower than the heuristic algorithm.

TABLE 6: Result of test cases (AE1-AE45).

Test case	Number of loads	Heuristic algorithm			Hybrid algorithm			
		Percentage to upper bound	Splits used	Time used (seconds)	Percentage to upper bound	Splits used	Improvement over heuristic	Time used (seconds)
AE1	179	19.7	2	21.4	2.4	2	21.49	5.5
AE2	60	4.3	1	18.0	1.8	1	2.67	3.0
AE3	366	38.0	8	192.6	1.3	5	59.36	7.5
AE4	161	22.7	1	22.3	1.4	4	27.46	5.5
AE5	752	36.0	6	912.1	7.8	12	43.92	46.4
AE6	163	12.5	6	13.5	2.2	4	11.75	4.9
AE7	644	28.5	9	491.7	1.5	8	37.85	62.5
AE8	83	4.2	1	19.1	0.7	2	3.57	2.9
AE9	81	5.2	3	18.1	1.4	1	4.00	3.7
AE10	171	26.2	1	21.9	2.5	3	32.20	6.3
AE11	477	20.5	8	57.4	0.9	8	24.70	9.1
AE12	272	15.3	4	39.0	1.6	5	16.17	4.1
AE13	863	16.5	4	738.1	4.8	7	14.01	39.2
AE14	274	19.1	5	24.7	1.0	3	22.41	5.8
AE15	755	31.1	7	438.1	1.0	9	43.67	83.1
AE16	194	20.4	2	35.6	2.1	5	23.01	3.2
AE17	192	12.5	4	20.4	1.5	4	12.54	5.5
AE18	358	28.0	5	145.8	2.3	5	35.67	7.3
AE19	153	5.3	4	16.6	1.9	5	3.66	2.8
AE20	744	31.3	9	478.6	10.6	12	30.12	77.7
AE21	155	25.2	4	15.9	1.3	4	32.05	6.3
AE22	636	36.3	6	405.6	2.2	9	53.64	48.1
AE23	75	15.1	3	13.9	0.8	4	16.94	4.4
AE24	73	17.6	2	16.7	0.3	2	20.86	3.2
AE25	459	25.9	4	89.5	1.4	7	33.05	18.0
AE26	1050	36.0	18	654.7	1.7	13	53.62	164.4
AE27	461	25.3	2	358.6	3.8	1	28.70	26.8
AE28	942	26.1	5	1796.6	0.8	10	34.26	120.9
AE29	381	8.1	12	746.7	0.6	6	8.19	12.2
AE30	379	15.4	2	510.5	0.2	4	18.00	9.6
AE31	845	32.5	9	757.2	1.7	8	45.59	89.2
AE32	256	24.6	2	289.0	0.6	5	31.83	12.8
AE33	737	52.2	6	1038.5	0.2	5	108.92	41.9
AE34	176	28.9	4	23.6	0.9	3	39.36	7.0
AE35	174	9.3	5	32.5	1.4	6	8.68	5.1
AE36	847	22.7	11	998.2	0.8	9	28.21	130.7
AE37	1328	51.6	10	1104.3	2.6	6	101.24	219.1
AE38	767	54.7	7	501.3	1.6	4	117.31	31.5
AE39	765	36.1	4	695.0	2.3	4	52.87	72.5
AE40	739	34.3	6	390.0	2.1	7	48.82	60.8
AE41	178	25.4	3	47.3	0.3	4	33.58	4.1
AE42	176	21.0	4	34.8	0.6	3	25.90	8.7
AE43	659	41.6	8	767.3	1.3	5	69.17	37.7
AE44	657	26.4	5	568.2	3.8	8	30.73	23.1
AE45	96	11.3	2	27.4	0.3	3	12.37	3.7

TABLE 7: Result of performance evaluation I.

Test case	No search space reduction		No initialization		No main loop	
	Value (percentage)	Time used (percentage)	Value (percentage)	Time used (percentage)	Value (percentage)	Time used (percentage)
R1	100	95.83	100	135.09	75.53	25.65
R2	99.13	92.86	100	178.67	80.15	32.48
R3	100	93.88	100	114.50	100	33.18
R4	100	92.24	100	120.91	100	33.12
R5	100	92.88	100	140.90	93.34	21.39
R6	98.79	92.01	100	119.67	86.92	29.90
R7	100	92.34	100	155.93	95.11	32.20
R8	98.77	93.76	99.94	154.80	72.92	22.78
R9	100	95.43	100	111.32	100	31.79
R10	100	94.53	100	135.21	95.46	26.58
RS1	100	92.22	100	112.37	75.53	25.65
RS2	97.50	92.37	100	155.02	80.15	32.48
RS3	100	95.68	100	115.96	100	33.18
RS4	100	94.95	100	143.69	100	33.12
RS5	100	93.14	100	125.32	93.34	21.39
RS6	98.69	94.32	100	112.92	86.92	29.90
RS7	100	95.30	100	195.11	95.11	32.20
RS8	96.50	94.41	99.21	115.04	72.92	22.78
RS9	100	94.87	100	118.70	100	31.79
RS10	100	93.47	100	157.74	95.46	26.58

6. Performance Evaluation

The proposed hybrid algorithm consists of 4 stages: search space reduction, initialization, evolutionary loop, and final tune-up. In this section, the performance evaluation on each stage is investigated.

Tables 7 and 8 show the results if each of the functions is disabled. The value column is used to indicate the loss of quality, and the time used is used to indicate the loss of speed (less than 100 means the algorithm runs faster than the full version).

The search space reduction stage requires around 7% of the running time. However, it builds a good base for further optimizing, as in some cases the quality of the solution drops if these two stages are missing.

The initialization stage greatly reduces the processing time required. For RS7, the time required is almost doubled if initialization is missing. And for R8 and RS8, it also improves the quality of result.

The main loop contributes a huge improvement on the quality of the generated solution. For cases like R3, R4, and R9, the algorithm can still provide good solutions just using initialization and posttuning, but not for the other cases. The result for RS1 to RS10 is the same as R1 to R10, since, without the main loop, the algorithm is not able to use any split.

The constraint handling methods (linear-guided and ε level comparison) do not require much time but could

improve the result up to 30% for nontrivial cases. This suggests that the proposed algorithm is able to escape from the local optima with the constraint handling methods.

The local search in the main loop plays a major role in improving the quality of the solution. It also requires a significant chunk of time but is worthwhile. As mentioned before, at the later stage of the optimization process, an exhaustive search is usually more efficient than a stochastic variation.

The final tune-up improves the result slightly for some cases without much execution time needed. Note that the local search in the main loop could partially replace the effect of the final tune-up since they are basically the same method. This stage is to ensure that there is no missing profit.

7. Conclusions and Future Work

In this paper, a hybrid evolutionary algorithm for solving the Australian wheat blending problem is proposed. The algorithm starts with a filtering process to reduce the search space. The filtering is based on predefined rules suggested by domain experts. Then the algorithm generates its initial solution by extracting the common parts from both the result from the linear-relaxed version of the problem and the result from a heuristic method. The main loop of the algorithm uses a combination of an evolutionary algorithm, a heuristic method, and the simplex algorithm to improve

TABLE 8: Result of performance evaluation II.

Test case	No constraint handling		No local search in loop		No final tune-up	
	Value (percentage)	Time used (percentage)	Value (percentage)	Time used (percentage)	Value (percentage)	Time used (percentage)
R1	100	97.36	97.59	82.74	100	99.87
R2	85.09	95.09	82.64	77.85	100	99.14
R3	100	96.62	100	84.63	100	98.68
R4	100	97.63	100	80.67	100	97.59
R5	73.62	95.70	83.74	77.46	100	99.33
R6	70.15	96.09	88.22	79.76	100	98.63
R7	100	94.52	95.59	86.82	100	98.20
R8	82.74	93.06	87.95	79.54	99.9	97.37
R9	100	95.97	100	87.22	100	99.73
R10	98.28	94.47	97.99	81.94	100	97.88
RS1	99.52	97.72	97.69	83.66	100	98.51
RS2	70.42	95.53	88.15	81.55	99.9	99.08
RS3	100	97.37	100	89.30	100	98.07
RS4	100	94.66	100	81.50	100	98.63
RS5	95.14	95.01	81.10	78.79	100	99.62
RS6	80.95	96.62	82.00	82.49	99.9	98.35
RS7	97.60	97.92	92.81	80.91	100	99.31
RS8	93.69	97.99	97.92	79.60	100	97.73
RS9	100	95.70	100	89.28	100	98.26
RS10	98.16	97.31	98.63	80.85	100	99.68

the solution while maintaining the feasibility of the solution. For the constraint handling part, the result from the linear-relaxed problem is used in conjunction with the epsilon level comparison. Those constraint handling methods help the algorithm explore the infeasible regions more efficiently. Final tune-up is performed by a local search method. The proposed algorithm is tested on 20 real-world test cases and 73 artificial test cases. Result shows that the proposed algorithm always finds equal or better results compared with the existing heuristic algorithm.

For further study, the parameter setting of this algorithm could be investigated. One promising way to improve the algorithm is to design an adaptive way to control the mutation probability, the local search probability, and especially the threshold used in ϵ LCH.

There are also additional functionalities requested by the growers. The growers have signed a few supply contracts before the harvest and they want to fulfil their contracts with minimum cost and maximize the profit of the rest products. Additionally, sometimes it is beneficial for the growers to buy some wheat from the other growers. Therefore, the growers also want the optimizer to generate a blending plan with the consideration of trading between multiple growers. The blending of other types of wheat is also requested.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was partially funded by the ARC Discovery Grants DP0985723, DP1096053, and DP130104395, as well as by Grant N N519 5788038 from the Polish Ministry of Science and Higher Education (MNiSW).

References

- [1] X. Li, M. R. Bonyadi, Z. Michalewicz, and L. Barone, "Solving a real-world wheat blending problem using a hybrid evolutionary algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 2665–2671, Cancún, Mexico, June 2013.
- [2] H. Karloff, *Linear Programming*, Birkhäuser, Boston, Mass, USA, 2nd edition, 2008.
- [3] G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, USA, 1998.
- [4] Y. Pochet and L. A. Wolsey, *Production Planning by Mixed Integer Programming*, Springer, New York, NY, USA, 2006.
- [5] B. Bilgen and I. Ozkarahan, "A mixed-integer linear programming model for bulk grain blending and shipping," *International Journal of Production Economics*, vol. 107, no. 2, pp. 555–571, 2007.
- [6] J. Ashayeri, A. G. M. van Eijs, and P. Nederstigt, "Blending modelling in a process manufacturing: a case study," *The European Journal of Operational Research*, vol. 72, no. 3, pp. 460–468, 1994.
- [7] Z. Jia and M. Ierapetritou, "Mixed-integer linear programming model for gasoline blending and distribution scheduling,"

- Industrial and Engineering Chemistry Research*, vol. 42, no. 4, pp. 825–835, 2003.
- [8] D. Randall, L. Cleland, C. S. Kuehne, G. W. B. Link, and D. P. Sheer, “Water supply planning simulation model using mixed-integer linear programming engine,” *Journal of Water Resources Planning and Management*, vol. 123, no. 2, pp. 116–124, 1997.
- [9] L. F. L. Moro and J. M. Pinto, “Mixed-integer programming approach for short-term crude oil scheduling,” *Industrial and Engineering Chemistry Research*, vol. 43, no. 1, pp. 85–94, 2004.
- [10] L. Costa and P. Oliveira, “Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems,” *Computers and Chemical Engineering*, vol. 25, no. 2–3, pp. 257–266, 2001.
- [11] T. Bäck and M. Schütz, “Evolution strategies for mixed-integer optimization of optical multilayer systems,” in *Evolutionary Programming IV: Proceedings of the 4th Annual Conference on Evolutionary Programming*, pp. 33–51, MIT Press, Cambridge, Mass, USA, 1st edition, 1995.
- [12] M. Emmerich, M. Grötzner, B. Groß, and M. Schütz, “Mixed-integer evolution strategy for chemical plant optimization with simulators,” in *Evolutionary Design and Manufacture*, I. C. Parmee, Ed., pp. 55–67, Springer, Berlin, Germany, 2000.
- [13] J. F. Gómez, H. M. Khodr, P. M. de Oliveira et al., “Ant colony system algorithm for the planning of primary distribution circuits,” *IEEE Transactions on Power Systems*, vol. 19, no. 2, pp. 996–1004, 2004.
- [14] R. L. Haupt, “Antenna design with a mixed integer genetic algorithm,” *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3, pp. 577–582, 2007.
- [15] B. B. Pal, D. Chakraborti, P. Biswas, and A. Mukhopadhyay, “An application of genetic algorithm method for solving patrol manpower deployment problems through fuzzy goal programming in traffic management system: a case study,” *International Journal of Bio-Inspired Computation*, vol. 4, no. 1, pp. 47–60, 2012.
- [16] M. P. Poland, C. D. Nugent, H. Wang, and L. Chen, “Genetic algorithm and pure random search for exosensor distribution optimisation,” *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 359–372, 2012.
- [17] T. Yokota, M. Gen, and Y.-X. Li, “Genetic algorithm for non-linear mixed integer programming problems and its applications,” *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 905–917, 1996.
- [18] Y. C. Lin, F. S. Wang, and K. S. Hwang, “A hybrid method of evolutionary algorithms for mixed-integer nonlinear optimization problems,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '99)*, vol. 3, Washington, DC, USA, 1999.
- [19] C. T. Su and C. S. Lee, “Network reconfiguration of distribution systems using improved mixed-integer hybrid differential evolution,” *IEEE Transactions on Power Delivery*, vol. 18, no. 3, pp. 1022–1027, 2003.
- [20] K. Deep, K. P. Singh, M. L. Kansal, and C. Mohan, “A real coded genetic algorithm for solving integer and mixed integer optimization problems,” *Applied Mathematics and Computation*, vol. 212, no. 2, pp. 505–518, 2009.
- [21] F. G. Lobo, C. F. Lima, and Z. Michalewicz, *Parameter Setting in Evolutionary Algorithm*, Springer, Berlin, Germany, 2007.
- [22] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, New York, NY, USA, 1st edition, 1996.
- [23] C. Grosan, A. Abraham, and H. Ishibuchi, *Hybrid Evolutionary Algorithms*, Springer, Berlin, Germany, 2007.
- [24] M. Lin and L. T. Yang, “Hybrid genetic algorithms for scheduling partially ordered tasks in a multi-processor environment,” in *Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications (RTCSA '99)*, pp. 382–387, Hong Kong, 1999.
- [25] P. Moscato, C. Cotta, and A. Mendes, “Memetic algorithms,” in *New Optimization Techniques in Engineering*, chapter 3, pp. 53–85, Springer, Berlin, Germany, 2004.
- [26] L. M. O. Queiroz and C. Lyra, “Adaptive hybrid genetic algorithm for technical loss reduction in distribution networks under variable demands,” *IEEE Transactions on Power Systems*, vol. 24, no. 1, pp. 445–453, 2009.
- [27] C. P. Gomes and B. Selman, “Algorithm portfolio design: theory vs. practice,” in *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI '97)*, pp. 190–197, Providence, RI, USA, 1997.
- [28] T. Takahama and S. Sakai, “Constrained optimization by ϵ constrained particle swarm optimizer with ϵ -level control,” in *Soft Computing as Transdisciplinary Science and Technology*, A. Abraham, Y. Dote, T. Furuhashi, M. Köppen, A. Ohuchi, and Y. Ohsawa, Eds., pp. 1019–1029, Springer, Berlin, Germany, 2005.

Research Article

Designing a Multistage Supply Chain in Cross-Stage Reverse Logistics Environments: Application of Particle Swarm Optimization Algorithms

Tzu-An Chiang,¹ Z. H. Che,² and Zhihua Cui^{3,4}

¹ Department of Business Administration, National Taipei College of Business, Taipei 10051, Taiwan

² Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan

³ Complex System and Computational Intelligent Laboratory, Taiyuan University of Science and Technology, Taiyuan 030024, China

⁴ State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

Correspondence should be addressed to Z. H. Che; zhche@ntut.edu.tw

Received 11 October 2013; Accepted 23 December 2013; Published 18 February 2014

Academic Editors: C. H. Aladag and C.-C. Jane

Copyright © 2014 Tzu-An Chiang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This study designed a cross-stage reverse logistics course for defective products so that damaged products generated in downstream partners can be directly returned to upstream partners throughout the stages of a supply chain for rework and maintenance. To solve this reverse supply chain design problem, an optimal cross-stage reverse logistics mathematical model was developed. In addition, we developed a genetic algorithm (GA) and three particle swarm optimization (PSO) algorithms: the inertia weight method (PSOA_IWM), V_{Max} method (PSOA_VMM), and constriction factor method (PSOA_CFM), which we employed to find solutions to support this mathematical model. Finally, a real case and five simulative cases with different scopes were used to compare the execution times, convergence times, and objective function values of the four algorithms used to validate the model proposed in this study. Regarding system execution time, the GA consumed more time than the other three PSOs did. Regarding objective function value, the GA, PSOA_IWM, and PSOA_CFM could obtain a lower convergence value than PSOA_VMM could. Finally, PSOA_IWM demonstrated a faster convergence speed than PSOA_VMM, PSOA_CFM, and the GA did.

1. Introduction

Intense competition within the global market has prompted enterprise competition to change from a competition among companies to that among supply chains. In addition to reducing operating costs and improving competitiveness, effectively integrating the upstream and downstream suppliers and manufacturers of a supply chain can reflect market changes and meet consumer needs efficiently.

Previous studies on the design problems of supply networks include [1–8]. In addition, Che and Cui [9] addressed the network design on unbalanced supply chain system. For the integrity of supply chain circulation, reverse logistics should be implemented to form a complete logistics circulation. Reverse logistics was first proposed by Stock [10]; then Trebilcock [11] indicated that, in the past, most enterprises focused only on forward logistics and misunderstood reverse logistics as a non-profitable activity.

Cohen [12] suggested that enterprises could save 40%–60% of manufacturing costs annually by adopting the remake method, compared with using new materials. In recent years, enterprises have begun paying increased attention to reverse logistics activities such as customer returns, product maintenance, replacement, and recycling. White et al. [13] described in detail the essential aspects and challenges in acquiring, assessing, disassembling, and reprocessing computer equipment as it moves through this reverse manufacturing process. Proper planning of a comprehensive product recycling plan can reduce the environmental damage caused by disposing of used equipment.

Based on literature review, reverse logistics includes management functions related to returned products, depot repair, rework, material reprocessing, material recycling, and disposal of waste and hazardous materials. These allow products to be returned upstream for processing in a reverse logistics system; thus, the circulation of an integral supply chain can be

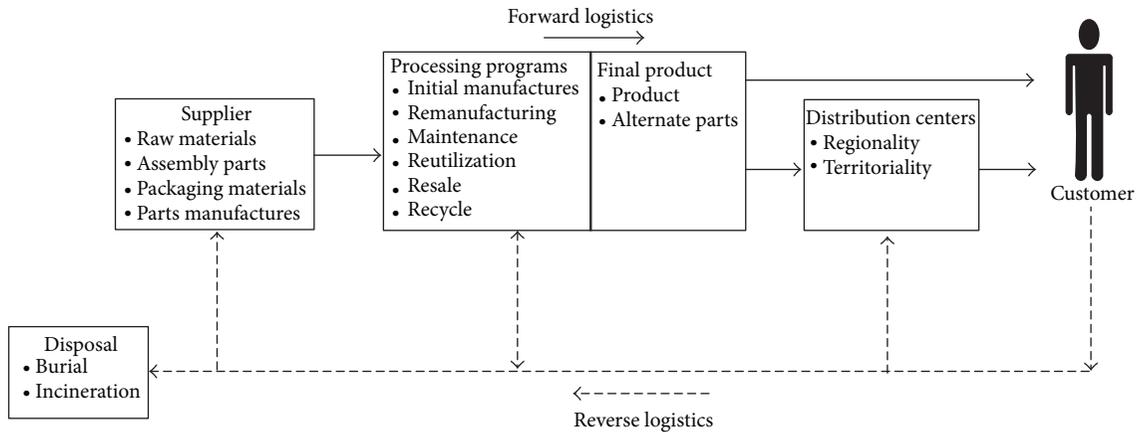


FIGURE 1: The reverse logistics flow of the products (Gattorna [14]).

implemented. The reverse logistics flow of products is shown in Figure 1.

Many scholars have defined reverse logistics briefly and clearly [10, 15–18], and some have studied the reverse logistics network design for different fields such as the steel industry [19], electronic equipment [20], sand recycling [21], reusable packaging [22], and general recovery networks [23]. Amini et al. [24] demonstrated how an effective and profitable reverse logistics operation for an RSSC was designed for an MDM in which customer operations demanded a quick repair service. Fleischmann et al. [25] considered a logistics network design in a reverse logistics context and presented a generic facility location model by discussing the differences compared with traditional logistics settings. This model was then used to analyze the impact of product return flow on logistics networks.

In addition, Savaskan et al. [26] developed a detailed understanding of the implications that a manufacturer's reverse channel choice has on forward channel decisions and the used product return rate from customers. Chouinard et al. [27] addressed problems related to integrating reverse logistics activities within an organization and to coordinating this new system. Kainuma and Tawara [28] proposed the multiple-attribute utility theory method for assessing a supply chain, including reusing and recycling throughout the life cycle of products and services. Nagurney and Toyasaki [29] developed a model linking these decisions to prices and material shipments among end-of-life electronics sources, recyclers, processors, and suppliers for deterministic scenarios. Nikolaidis [30] proposed a single-period mathematical model for determining a reverse supply chain plan that considers procurement and returns' remanufacturing, and Nenes and Nikolaidis [31] extended Nikolaidis's model to a multi-period model. Salema et al. [32] developed a multiperiod, multiproduct model for designing supply chain networks regarding reverse flows. More recently, Pinto-Varela et al. [33] considered an environmental perspective to develop a mixed-integer linear programming model for planning reverse supply chains. Amin and Zhang [34] presented a mixed-integer linear programming model for designing a closed-loop supply chain network regarding product life cycles. In

addition, Huang et al. [35] analyzed strategies of a closed-loop supply chain containing a dual recycling channel. Although cross-stage logistics in reverse supply chains generally exists in practice, our research suggests that it has yet to be adequately addressed. Hence, the motivation of this study is to design the reverse supply chain with cross-stage logistics.

Reverse logistics is more complex than forward logistics, and this study aimed to develop a mathematical foundation for modeling a cross-stage reverse logistics plan that enables defective products with differing degrees of damage to be returned to upstream partners in the stages of a supply chain for maintenance, replacement, or restructuring. This cross-stage reverse logistics model can help save time, lessen unnecessary deliveries, and, more importantly, meet the conditions of reverse logistics operation more efficiently.

Recently, GAs have been regarded as a novel approach to solving complex, large-scale, and real-world optimization problems [6, 36–42]. Moreover, the PSO proposed by Kennedy and Eberhart [43] was an iteration optimization instrument, generating a group of initial solutions at the beginning and then acquiring the optimal value through iteration. Liao and Rittscher [44] applied this instrument to scheduling problems related to industrial piece work requiring minimal completion time. Zhang et al. [45] applied PSO to solve the minimization problems of the project duration for resource-constrained scheduling. Shi et al. [46] applied a PSO to the traveling salesman problem. Che [47] developed a PSO-based back-propagation artificial neural network for estimating the product and mold costs of plastic injection molding and Che [48] proposed a modified PSO method for solving multiechelon unbalanced supply chain planning problems. Priya and Lakshmi [49] applied PSO for performing the real time control of spherical tank system and Ali et al. [50] used the PSO for solving the constrained numerical and engineering benchmark problems. Other related studies concerning the use of PSO for the optimization problems are [51–54].

In addition, Dong et al. [55] compared the improved PSO, a combinatorial particle swarm optimization (CPSO), with GA, and the results showed that the improved PSO was more effective in solving nonlinear problems. Yin and Wang [56]

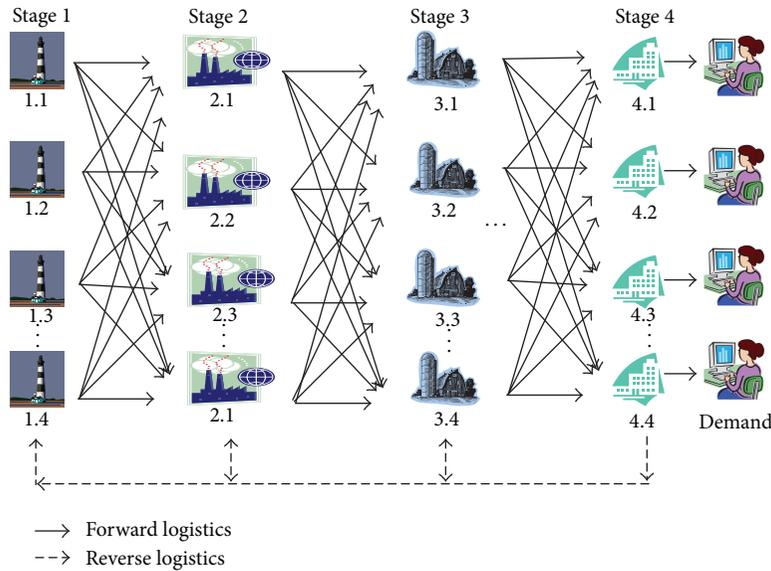


FIGURE 2: The transportation model of reverse logistics.

used PSO to solve nonlinear resource allocation problems and compared PSO with the GA. They found that the efficiency and potency of a PSO were higher than those of a GA. Salman et al. [57] applied PSO to solve the efficiency rates of tasks assigned to computers or parallel computer systems and compared the results with those of GA. The results showed that PSO has faster execution and convergence speeds than the GA. Based on our research, no previous studies have applied PSO to cross-stage reverse logistics problems; therefore, to solve this problem, this study used three updated PSO methods: the inertia weight method (PSOA_IWM), constriction factor method (PSOA_CFM), and V_{Max} method (PSOA_VMM). The results were then compared with those using the GA regarding system execution time, convergence time, and objective function value.

The remainder of this paper is structured as follows. Section 2 introduces the proposed mathematical foundation and solving algorithms for modeling and solving cross-stage reverse logistics problems. Section 3 presents illustrative examples and the comparative and analytical results of the algorithms. Finally, Section 4 provides the conclusion of this study and offers suggestions for future research.

2. Mathematical Foundation and Solving Models for Cross-Stage Reverse Logistics Problems

2.1. Problem Description. Reverse logistics activities include recycling, rework, replacement, and waste disposal; however, the reverse logistics activity of each function differs. Therefore, this study designed a forward and reverse cross-stage logistics system for maintaining, reassembling, and packaging recycled defective products. The structure is shown in Figure 2.

When downstream partners generate defective products, the products can be returned directly to upstream supply chain partners for maintenance to restore product function and value, based on the degree of damage. Therefore, this study supposed that, when defective products are generated, they can be divided into N parts according to the average volume of defective products generated by a particular supplier. Downstream partners can then return defective products, based on the divided quantity, to upstream partners for maintenance. For example, when the first partner of the fourth stage generates defective products, the total defective amount is divided into three parts and then sent to the first, second, and third stage partners separately in the supply chain, thereby reducing general reverse logistics costs and transportation time.

For supply chain partner selection, this study considered productivity restrictions, transportation costs, manufacturing costs, transportation time, manufacturing quality, and other parameters. The T -transfer approach is a common statistical technology that is employed to integrate variables. In this study, the T -transfer of transportation costs, manufacturing costs, transportation time, and manufacturing quality was integrated into the objective function standards. T -transfer is a common statistics technology first proposed by McCall [58]; it is defined as follows: “ T -Scores are a transformation of raw scores into a standard form, where the transformation is made when there is no knowledge of the population’s mean and standard deviation.” T -scores have a mean of 50 and a standard deviation of 10. Che [59] considered the manufacturing cost and time, transportation cost and time, product quality, and green appraisal score in selecting green suppliers when establishing a green supply chain and used T -transfer technology to transform the data. Cost, time, quality, and green appraisal score are measurable criteria with different units; thus, in this study the T -transfer approach was

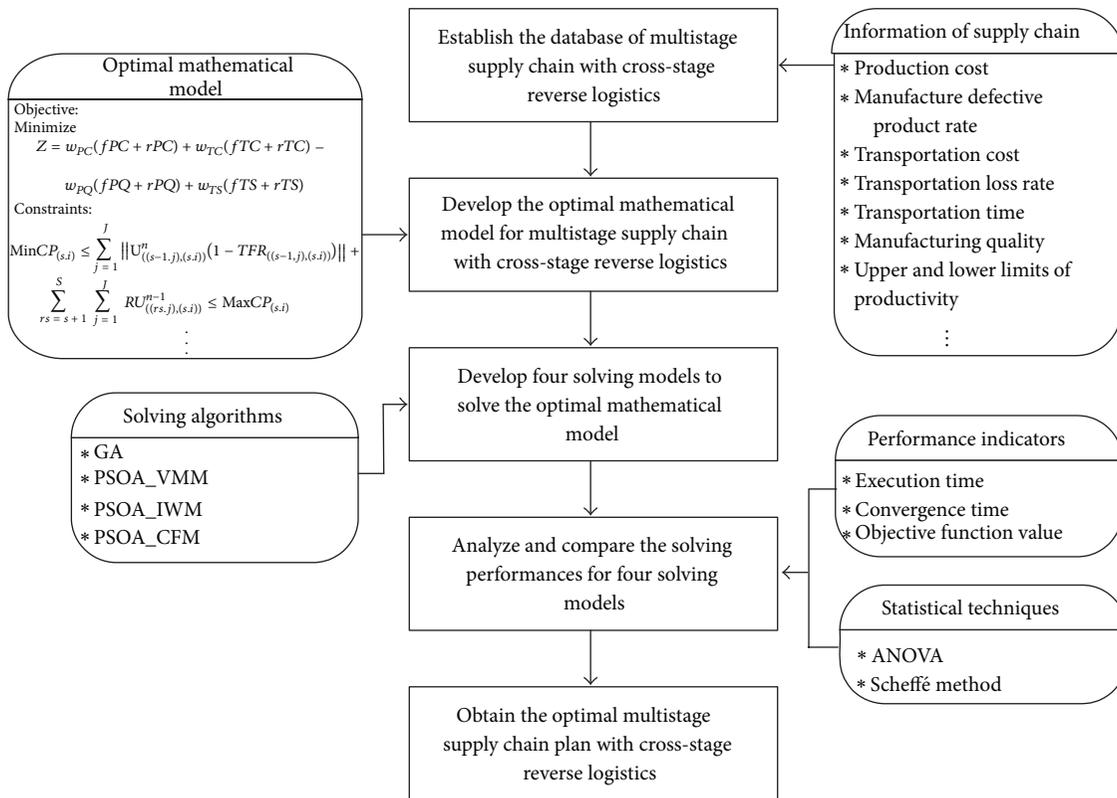


FIGURE 3: The structure of this study.

also employed to first transform the original scores of each criterion into a standard form and then to integrate them.

To satisfy the conditions of the actual production situation, this study used transportation losses and manufacturing losses to construct an unbalanced supply chain network. In considering the characteristics of all the suppliers addressed in this study, we developed a cross-stage reverse logistics course planning system for single-product and multiperiod programming.

We programmed the reverse logistics for recycled defective products, which were returned directly to the upstream supply chain partners for maintenance, reassembly, and repackaging through the cross-stage reverse logistics course programming based on the degree and nature of the damage. For selecting supply chain partners, this study considered the manufacturing characteristics (transportation costs, production costs, upper and lower limit of productivity, manufacturer’s defective product rate, transportation losses rate, and manufacturing quality) to construct the reverse logistics programming model. Based on these data, optimal manufacturing quality with minimal production cost, transportation cost, and transportation time can be determined.

In considering the different evaluation criteria, this study *T*-transferred the database and used the Visual Basic program language to compile four solution models, including GA, PSOA_IWM, PSOA_VMM, and PSOA_CFM. The considered parameters in the supplier database were combined to develop a set for designing reverse logistics course planning systems. The framework of this study is shown in Figure 3.

Analysis of variance (ANOVA) and Scheffé analyses were performed to compare the objective function values (*T*-score), convergence times, and run times of the four algorithms to verify the validity of this study and the performance of the four algorithms.

2.2. Mathematical Foundation for Cross-Stage Reverse Logistics Problems. The optimal mathematical model of cross-stage reverse logistics was developed as described in the following steps. The definitions of notations used in this model are listed as follows.

Notations for developing the optimal mathematical model:

Parameters

- i, j*: Serial number of supplier
 $i = 1, 2, 3, \dots, I; j = 1, 2, 3, \dots, J$
- n*: Production period $n = 1, 2, 3, \dots, N$
- s, rs*: Stages of the supply chain network,
 $s = 1, 2, 3, \dots, S; rs = 1, 2, 3, \dots, S$
- I, J*: Total number of suppliers
- N*: Total production periods
- S*: Total stages of supply chain network
- $CD^n_{(s,i)}$: Customer requirement of supplier *i* at stage *s* for period *n*
- $Min CP_{(s,i)}$: Minimal starting up productivity of supplier *i* at stage *s*
- $Max CP_{(s,i)}$: Maximal starting up productivity of supplier *i* at stage *s*

$PC_{(s,i)}$:	Manufacturing cost of supplier i at stage s
$PQ_{(s,i)}$:	Product quality of supplier i at stage s
$TC_{((s,i),(s+1,j))}$:	Transportation cost from supplier i at stage s to supplier j at stage $s + 1$
$TS_{((s,i),(s+1,j))}$:	Transportation time from supplier i at stage s to supplier j at stage $s + 1$
$\overline{PC}_{(s,i)}$:	Average manufacturing cost of supplier i at stage s
$\overline{PQ}_{s,i}$:	Average product quality of supplier i at stage s
$\overline{TC}_{((s,i),(s+1,j))}$:	Average transportation cost from supplier i at stage s to supplier j at stage $s + 1$
$\overline{TS}_{((s,i),(s+1,j))}$:	Average transportation time from supplier i at stage s to supplier j at stage $s + 1$
${}^T PC_{(s,i)}$:	Manufacturing cost of supplier i at stage s after T -transfer
${}^T PQ_{s,i}$:	Product quality of supplier i at stage s after T -transfer
${}^T TC_{((s,i),(s+1,j))}$:	Transportation cost from supplier i at stage s to supplier j at stage $s + 1$ after T -transfer
${}^T TS_{((s,i),(s+1,j))}$:	Transportation time from supplier i at stage s to supplier j at stage $s + 1$ after T -transfer
$SG_{PC_{s,i}}$:	Manufacturing cost standard deviation of supplier i at stage s
$SG_{TC_{((s,i),(s+1,j))}}$:	Transportation cost standard deviation of supplier i at stage s to supplier j at stage $s + 1$
$SG_{PQ_{s,i}}$:	Product quality standard deviation of supplier i at stage s
$SG_{TS_{((s,i),(s+1,j))}}$:	Transportation time standard deviation of supplier i at stage s to supplier j at stage $s + 1$
$FR_{(s,i)}$:	Defective product rates of supplier i at stage s
$TFR_{((s,i),(s+1,j))}$:	Transportation loss rate from supplier i at stage s to supplier j at stage $s + 1$
$w_{PC}, w_{TC}, w_{TS}, w_{PQ}$:	Weights of manufacturing cost, transportation cost, transportation time, and product quality
$\ \ $:	Integer function for obtaining the integer value of the real number by eliminating its decimal.

Decision Variables

$U^n_{((s,i),(s+1,j))}$:	Transportation quantity from supplier i at stage s to supplier j at stage $s + 1$ for period n
$RU^n_{((rs,j),(s,i))}$:	Defective products quantity from supplier j at stage rs to supplier i at stage s stage for term n .

Notations for developing the update models for the position and velocity of each particle:

c_1, c_2 :	Learning factors
K :	Constriction factor
rand():	Random numbers between 0 and 1
s_i^* :	$Pbest$ memory value of particle i
$s_i^\#$:	$Gbest$ memory value of particle i
s_i^{new} :	New position of particle i
v_i^{old} :	Original velocity of particle i
v_i^{new} :	New velocity of particle i
v_{max} :	The set maximal velocity
w :	Inertia weight
ϕ :	Totaling of cognition parameter and social parameter, which must exceed 4.

Notations for performing hypotheses on the objective function value, convergence time, and completion time among four proposed approaches:

CT_{GA} :	Convergence time of GA
CT_{PSOA_IWM} :	Convergence time of PSOA_IWM
CT_{PSOA_VMM} :	Convergence time of PSOA_VMM
CT_{PSOA_CFM} :	Convergence time of PSOA_CFM
FT_{GA} :	Completion time of GA
FT_{PSOA_IWM} :	Completion time of PSOA_IWM
FT_{PSOA_VMM} :	Completion time of PSOA_VMM
FT_{PSOA_CFM} :	Completion time of PSOA_CFM
Obj_{GA} :	Objective function value of GA
Obj_{PSOA_IWM} :	Objective function value of PSOA_IWM
Obj_{PSOA_VMM} :	Objective function value of PSOA_VMM
Obj_{PSOA_CFM} :	Objective function value of PSOA_CFM.

Acquire the minimization of manufacturing costs, transportation costs, and transportation time, as well as the maximization of the manufacturing quality of the different suppliers, at various stages of forward and reverse logistics.

Manufacturing cost for forward logistics:

$$\begin{aligned}
 fPC &= \sum_{n=1}^N \sum_{s=1}^S \sum_{i=1}^I {}^T PC_{(s,i)} \\
 &\quad \times \left[\left\| \frac{U^n_{((s,i),(s+1,1))}}{1 - FR_{(s,i)}} \right\| \right. \\
 &\quad \left. + \sum_{j=2}^J \left\| U^n_{((s-1,j),(s,i))} (1 - TFR_{((s-1,j),(s,i))}) \right\| \right]. \tag{1}
 \end{aligned}$$

Transportation cost for forward logistics:

$$fTC = \sum_{n=1}^N \sum_{s=1}^S \sum_{i=1}^I \sum_{j=1}^J {}^T TC_{(s,i),(s+1,j)} U^n_{((s,i),(s+1,j))}. \tag{2}$$

Product quality for forward logistics:

fPQ

$$= \sum_{n=1}^N \sum_{s=1}^S \sum_{i=1}^I T PQ_{(s,i)} \times \left[\left\| \frac{U_{((s,i),(s+1.1))}^n}{1 - FR_{(s,i)}} \right\| + \sum_{j=2}^J \left\| U_{((s-1,j),(s,i))}^n (1 - TFR_{((s-1,j),(s,i))}) \right\| \right] \quad (3)$$

Transportation time for forward logistics:

$$fTS = \sum_{n=1}^N \sum_{s=1}^S \sum_{i=1}^I \sum_{j=1}^J T TS_{(s,i),(s+1,j)} U_{((s,i),(s+1,j))}^n \quad (4)$$

Manufacturing cost for reverse logistics:

$$rPC = \sum_{n=2}^N \sum_{s=1}^{S-1} \sum_{i=1}^I T PC_{(s,i)} \sum_{rs=s+1}^S \sum_{j=1}^J RU_{((rs,j),(s,i))}^n \quad (5)$$

Transportation cost for reverse logistics:

$$rTC = \sum_{n=2}^N \sum_{s=1}^S \sum_{i=1}^I T TC_{((rs,j),(s,i))} \sum_{rs=s+1}^S \sum_{j=1}^J RU_{((rs,j),(s,i))}^n \quad (6)$$

Product quality for reverse logistics:

$$rPQ = \sum_{n=2}^N \sum_{s=1}^{S-1} \sum_{i=1}^I T PQ_{(s,i)} \sum_{rs=s+1}^S \sum_{j=1}^J RU_{((rs,j),(s,i))}^n \quad (7)$$

Transportation time for reverse logistics:

$$rTS = \sum_{n=2}^N \sum_{s=1}^S \sum_{i=1}^I T TS_{((rs,j),(s,i))} \sum_{rs=s+1}^S \sum_{j=1}^J RU_{((rs,j),(s,i))}^n \quad (8)$$

The objective function is expressed as follows:

$$\begin{aligned} \text{Minimize } Z = & w_{PC} (fPC + rPC) \\ & + w_{TC} (fTC + rTC) \\ & - w_{PQ} (fPQ + rPQ) \\ & + w_{TS} (fTS + rTS) \end{aligned} \quad (9)$$

s.t.

Upper and lower limits of productivity of all the suppliers:

$$\begin{aligned} \text{Min } CP_{(s,i)} \leq & \sum_{j=1}^J \left\| U_{((s-1,j),(s,i))}^n (1 - TFR_{((s-1,j),(s,i))}) \right\| \\ & + \sum_{rs=s+1}^S \sum_{j=1}^J RU_{((rs,j),(s,i))}^{n-1} \leq \text{Max } CP_{(s,i)} \end{aligned} \quad (10)$$

for $n = 1, 2, 3, \dots, N$;

$s = 2, 3, \dots, S$; $i = 1, 2, 3, \dots, I$;

$$\begin{aligned} \text{Min } CP_{(s,i)} \leq & \sum_{j=1}^J \left\| \frac{U_{((s,i),(s+1,j))}^n}{1 - FR_{(s,i)}} \right\| \\ & + \sum_{rs=s+1}^S \sum_{j=1}^J RU_{((rs,j),(s,i))}^{n-1} \leq \text{Max } CP_{(s,i)} \end{aligned} \quad (11)$$

for $n = 1, 2, 3, \dots, N$; $s = 1$;

$i = 1, 2, 3, \dots, I$.

Ensure the balance between the input and output of all partners by considering the transportation defective rate:

$$\begin{aligned} & \sum_{j=1}^J \left\| U_{((s-1,j),(s,i))}^n (1 - TFR_{((s-1,j),(s,i))}) \right\| \\ & + \sum_{rs=s+1}^S \sum_{j=1}^J RU_{((rs,j),(s,i))}^{n-1} \\ & = \sum_{j=1}^J U_{((s+1,j),(s,i))}^n + \sum_{rs=1}^{s-1} \sum_{j=1}^J RU_{((s,i),(rs,j))}^n \end{aligned} \quad (12)$$

for $n = 1, 2, 3, \dots, N$;

$s = 1, 2, 3, \dots, S$; $i = 1, 2, 3, \dots, I$;

$$\sum_{rs=1}^s \sum_{j=1}^J RU_{((rs,j),(s,i))}^n$$

$$= \left\| \left(\sum_{j=1}^J \left\| U_{((s-1,j),(s,i))}^n (1 - TFR_{((s-1,j),(s,i))}) \right\| + \sum_{rs=s+1}^S \sum_{j=1}^J RU_{((rs,j),(s,i))}^{n-1} \right) FR_{(s,i)} \right\| \quad (13)$$

for $n = 1, 2, 3, \dots, N$;

$s = 1, 2, 3, \dots, S$; $i = 1, 2, 3, \dots, I$.

The product quantity should meet customer requirements:

$$\sum_{j=1}^J \left\| U_{((s-1,j),(s,i))}^n \left(1 - TFR_{((s-1,j),(s,i))} \right) \right\| - \sum_{r=1}^{s-1} \sum_{j=1}^J RU_{((s,i),(s-r,j))}^n = CD_{(s,i)}^n \quad (14)$$

for $n = 1, 2, 3, \dots, N$; $s = S$;
 $i = 1, 2, 3, \dots, I$.

The weights of manufacturing cost, transportation cost, transportation time, and product quality should be not less than 0 and not more than 1:

$$0 \leq w_{PC}, w_{TC}, w_{TS}, w_{PQ} \leq 1, \quad (15)$$

$$w_{PC} + w_{TC} + w_{TS} + w_{PQ} = 1.$$

The defective products returned in the first period during the multiperiods of production number zero:

$$RU_{((rs,j),(s,i))}^n = 0 \quad \text{for } n \leq 0; \quad s = 1, 2, 3, 4, \dots, S-1;$$

$$rs = s + 1, \dots, S; \quad i = 1, 2, 3, \dots, I; \quad (16)$$

$$j = 1, 2, 3, \dots, J.$$

The forward and reverse transportation volume must be larger than zero and be an integer:

$$U_{((s,i),(s+1,j))}^n \geq 0 \quad \text{and} \quad U_{((s,i),(s+1,j))}^n \in \text{Integer}$$

$$\text{for } n = 1, 2, 3, \dots, N; \quad s = 1, 2, 3, \dots, S; \quad (17)$$

$$i = 1, 2, 3, \dots, I; \quad j = 1, 2, 3, \dots, J;$$

$$RU_{((rs,j),(s,i))}^n \geq 0 \quad \text{and} \quad U_{((s,i),(s+1,j))}^n \in \text{Integer}$$

$$\text{for } s = 1, 2, 3, \dots, S; \quad rs = S + 1, \dots, S;$$

$$n = 1, 2, 3, \dots, N; \quad i = 1, 2, 3, \dots, I; \quad j = 1, 2, 3, \dots, J. \quad (18)$$

Manufacturing costs, transportation costs, manufacturing quality, and transportation time should be T -transferred:

$${}^T PC_{(s,i)} = \frac{PC_{(s,i)} - \overline{PC}_{(s,i)}}{SG_{PC_{(s,i)}}/10} + 50 \quad \text{for } s = 1, 2, 3, \dots, S;$$

$$i = 1, 2, 3, \dots, I;$$

$${}^T TC_{((s,i),(s+1,j))} = \frac{TC_{((s,i),(s+1,j))} - \overline{TC}_{((s,i),(s+1,j))}}{SG_{TC_{((s,i),(s+1,j))}}/10} + 50$$

$$\text{for } s = 1, 2, 3, \dots, S; \quad i = 1, 2, 3, \dots, I;$$

$$j = 1, 2, 3, \dots, J;$$

$${}^T PQ_{(s,i)} = \frac{PQ_{(s,i)} - \overline{PQ}_{(s,i)}}{SG_{PQ_{(s,i)}}/10} + 50 \quad \text{for } s = 1, 2, 3, \dots, S;$$

$$i = 1, 2, 3, \dots, I;$$

$${}^T TS_{((s,i),(s+1,j))} = \frac{TS_{((s,i),(s+1,j))} - \overline{TS}_{((s,i),(s+1,j))}}{SG_{TS_{((s,i),(s+1,j))}}/10} + 50$$

$$\text{for } s = 1, 2, 3, \dots, S; \quad i = 1, 2, 3, \dots, I;$$

$$j = 1, 2, 3, \dots, J. \quad (19)$$

2.3. Proposed Models for Solving Cross-Stage Reverse Logistics Problems

2.3.1. *GA-Solving Model.* The detailed procedures of a GA-solving model are described as follows.

Step 1. The encoding of this study was performed according to the cross-stage reverse logistics problem including forward and reverse transportation routes; therefore, one route is one encoding value. The scope is randomly generated based on the demands and (10)–(14). The chromosome structure is shown in Figure 4. The gene cell index 1.1–2.1 in the figure represents the products sent from the first supplier of the first stage to the initial supplier of the second stage within the supply chain structure, whereas the gene value represents the transportation volumes from upstream to downstream.

Step 2. Substitute all the generated encoding values in the objective function equation (1) of this study to acquire the fitness function value of each gene.

Step 3. This study adopted the roulette wheel selection proposed by Goldberg [60], which is performed before cloning to solve the minimization problem of this study. It then selects the reciprocal of fitness function generated in Step 2 and calculates the cumulative probability of each strip of chromosome; the larger probability value indicates that this chromosome has a greater likelihood of being duplicated. One probability value between 0 and 1 is generated, the suitable fitness function is determined, and cloning is carried out.

Step 4. The crossover of this study involves using the single-point crossover method. Randomly select two chromosomes from the parent body for crossover, and generate one crossover point, then exchange the genes of the chromosome. The crossover course is shown in Figure 5.

Step 5. The mutation of this study also adopts a single-point mutation method and treats the delivery route of one supplier as a “single-point” of value. The mutation method is shown in Figure 6.

Step 6. The new filial generation was generated through the gene evolution of Steps 3–5; if the optimal fitness function value of the filial generation is higher than that of the parental

Gene cell index	1.1-2.1	1.2-2.1	1.3-2.1	1.1-2.2	1.2-2.2	1.3-2.2	...	3.1-4.6	3.2-4.6	3.3-4.6	3.4-4.6	3.5-4.6
Gene value	236	224	115	75	55	69	...	75	65	78	99	63

FIGURE 4: Chromosome structure.

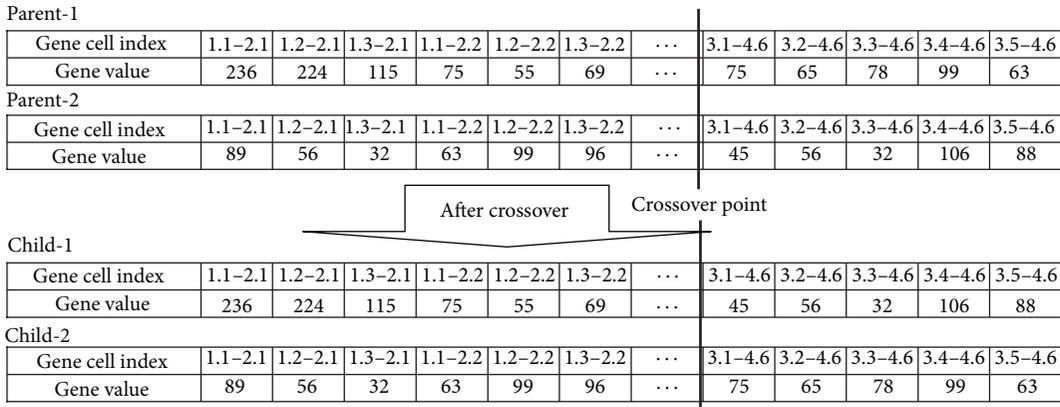


FIGURE 5: Crossover process.

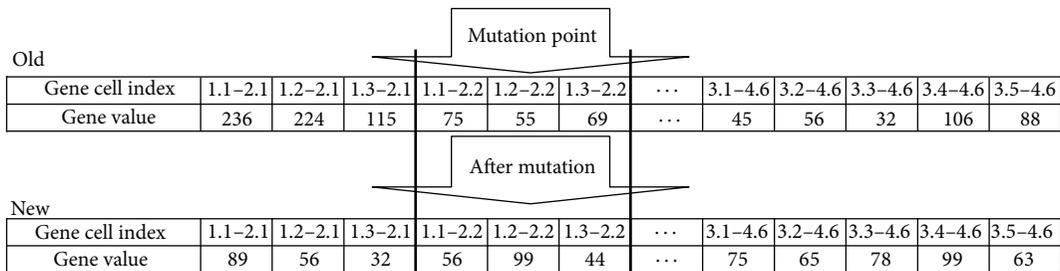


FIGURE 6: Mutation process.

Particle	1	2	3	4	5	6	...	60	61	62
Line	1.1-2.2 _F	1.1-2.2 _F	1.1-2.3 _F	1.1-2.4 _F	1.2-2.1 _F	1.2-2.2 _F	...	3.3-4.6 _F	3.4-4.6 _F	3.5-4.6 _F
Volume	103	27	30	140	158	18	...	308	15	61

FIGURE 7: Particle swarm encoding for forward logistics.

Particle	1	2	3	4	5	6	...	117	118	119
Line	2.1-1.1 _R	2.1-1.2 _R	2.1-1.3 _R	2.2-1.1 _R	2.2-1.2 _R	2.2-1.3 _R	...	4.6-3.3 _R	4.6-3.4 _R	4.6-3.5 _R
Volume	0	0	42	9	0	0	...	0	2	0

FIGURE 8: Particle swarm encoding for reverse logistics.

generation, then this would replace the parental generation as the new parent generation; otherwise, the original parental generation would be reserved to conduct the evolution of the next generation.

Step 7. This study sets the iteration times as the termination condition for gene evolution.

2.3.2. PSO-Solving Models. The detailed procedures involved in PSO-solving models are described as follows.

Step 8. Set the relative coefficients as particle population, velocity, weight, and iteration times; then all forward and

reverse transportation routes are viewed as one particle based on the supply chain structure. The forward and reverse particle swarm encodings are shown in Figures 7 and 8, 1.1-2.1_F in Figure 7 represents the products sent from the first supplier of the first stage to the first supplier of the second stage, and 2.1-1.1_R in Figure 8 represents the products returned to the first suppliers of the first stage from the first suppliers of the second stage.

The forward transportation volume produces the parental generation solution, adopting demand, transportation loss, manufacturer's defective products, and (10)-(14) as the random variant scope for the particles. Each particle has its own

initial parameters of velocity and position, generated within the scope of 0–1. The velocity and position would be renewed, and the reverse part is delivered according to the proportion, based on the quantity of defective products generated by the downstream suppliers of each stage. For example, the defective products generated by the fourth stage retailer would first be divided into 30%, 30%, and 40%, according to the proportion, and then delivered to the suppliers of the third, second, and first stages.

Step 9. All particles received by the initial solutions of objective function equation (1) are carried to conduct the operation to achieve minimal transportation costs, transportation times, and manufacturing costs, as well as maximizing the manufacturing quality for each granule particle.

Step 10. The target value of each particle generated in Step 9 is compared to receive *Gbest*.

Step 11. Modify the *Pbest* and *Gbest*. If the *Pbest* is better than the *Gbest*, then the *Pbest* would replace the *Gbest*.

Step 12. For the renewal portion of this study, the inertia weight method (PSOA_IWM) proposed by Eberhart and Shi [61], the constriction factor method (PSOA_CFM) proposed by Clerc [62], and the V_{Max} method (PSOA_VMM) proposed by Eberhart and Kennedy [43, 63] were used to update the position and velocity of each particle. The updated modes are listed as follows (descriptions of notations are listed in the appendix).

(1) PSOA_IWM (Eberhart and Shi [61]):

$$v_i^{new} = wv_i^{old} + c_1 \times \text{rand}() \times (s_i^* - s_i^{old}) + c_2 \times \text{rand}() \times (s_i^\# - s_i^{old}),$$

$$s_i^{new} = s_i^{old} + v_i^{new}.$$
(20)

(2) PSOA_VMM (Eberhart and Kennedy [43, 63]):

$$v_i^{new} = v_i^{old} + c_1 \times \text{rand}() \times (s_i^* - s_i^{old}) + c_2 \times \text{rand}() \times (s_i^\# - s_i^{old}),$$

$$s_i^{new} = s_i^{old} + v_i^{new}$$

$$\text{if } v_i > v_{max}, \quad v_i = v_{max}$$

$$\text{else if } v_i < -v_{max}, \quad v_i = -v_{max}.$$
(21)

When the particle velocity was too extreme, it could be guided to the normal velocity vector.

(3) PSOA_CFM (Clerc [62]):

$$v_i^{new} = k \times \left(v_i^{old} + c_1 \times \text{rand}() \times (s_i^* - s_i^{old}) + c_2 \times \text{rand}() \times (s_i^\# - s_i^{old}) \right),$$

$$s_i^{new} = s_i^{old} + v_i^{new},$$

$$K = \frac{2}{2 - \phi - \sqrt{\phi^2 - 4\phi}},$$

$$\phi = c_1 + c_2, \quad \phi > 4.$$
(22)

Step 13. After the velocity and position of the particles are updated, they must be verified to determine whether they met (10)–(18) and the set maximal velocity; if these conditions are not met, then the renewal formulae would be used until the renovation meets the restriction formula.

Step 14. Steps 9–13 would be repeated based on iteration times, the *Gbest* of each iteration time would be compared, and then the iteration times would be used as the condition for stopping the calculation. The final algorithm presents the delivery quantity and target value of the forward and reverse routes.

3. Illustrative Example and Results Analysis

This section presents an illustrative example involving a semiconductor supply chain network to demonstrate the effectiveness of the proposed approaches. A typical semiconductor supply chain network is shown in Figure 9. The chain includes a multistage process: obtaining silicon material, material fabrication, wafer fabrication, and a final test. In each stage, there are many enterprises that perform the production processes to fulfill the demand of the customer.

This case programmed one unbalanced supply chain network structure, including forward and reverse logistics, so that downstream suppliers or retailers can return defective products directly to upstream supply chain partners. The manufacturer can restore a broken product's function, depending on the damage, so that the product's purpose is recovered. This case addressed forward and reverse logistics partner selection and quantity delivery problems using a {3-4-5-6} network structure. It also programmed a three-period customer requirement list for a single product. This case supposed that the initial inventory of the first period was zero, transportation losses were considered waste and cannot be reproduced, and different reverse logistics for defective products of different damage levels were programmed. For example, when 10 defective products were generated by the first supplier of the fourth stage, this study assumes that 30% were returned to the third stage, 30% were returned to the second stage, and the rest were returned to the first stage. Therefore, the reverse logistics of this study would generate a cross-stage reverse delivery status.

This study considered the productivity restrictions, manufacturing costs, delivery costs, manufacturing quality, and transportation time for all suppliers in selecting supply chain partners. This study also considered the manufacturer's defective product rate and the transportation loss rate of suppliers to form a so-called "unbalanced" supply chain network. The details of all of the suppliers are shown in Figure 10 and Table 1. In addition, the weights of manufacturing costs,

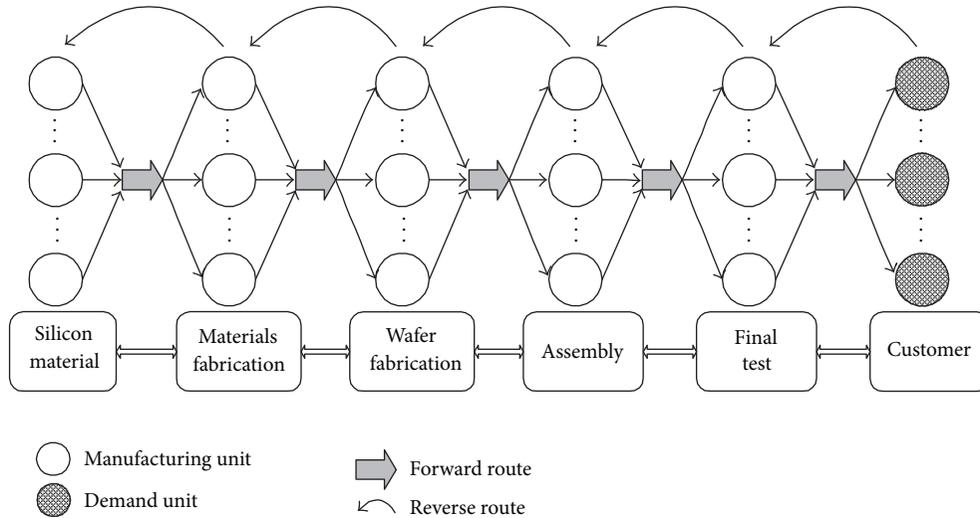


FIGURE 9: Typical supply chain network for semiconductor.

transportation costs, transportation time, and product quality were assumed to be equal.

This study used GA, PSOA_IWM, PSOA_CFM, and PSOA_VMM both to solve the problem of the optimal mathematical model of cross-stage reverse logistics constructed by this study and to determine the optimal parameter values. We used the experimental design to determine the optimal parameter values, and the parameters of the GA used in this study refer to the proposal of Eiben et al. [64]. It is possible to determine the optimal solution when the mutation rate is 0.005–0.01 and the crossover rate is 0.75–0.95. This study conducted 16 groups of experimental designs for the parental bodies (10, 20), crossover rates (0.6, 0.95), mutation rates (0.02, 0.05), and generation (1000, 2000). Each group was repeated 10 times to obtain the average, and the optimal parameter values were as follows: parental generation (20); crossover rate (0.6); mutation rate (0.05); generation (2000). The experimental results are shown in Table 2.

For the PSO, this study used PSOA_IWM, PSOA_CFM, and PSOA_VMM to solve the problems. PSOA_IWM was suggested by Eberhart and Shi [61], so, when W was between 0.9–1.25, it had a higher chance of achieving the optimal solution; the design of PSOA_IWM parameters was as follows: particle population (10, 20), velocity (30, 50), weight (0.4, 0.9), and generation (1000, 2000). Sixteen groups of experiments were designed and each group was repeated 10 times to gain the average convergence value, completion time, and convergence time. The optimal parameters of the experimental results were as follows: particle: 20; weight: 0.4; velocity: 50; generation: 2000. The experimental results are shown in Table 3. PSOA_CFM refers to the $c_1 = 2.05, c_2 = 2.05$ proposed by Clerc [62], $c_1 = 2.8, c_2 = 1.3$ proposed by Zhang et al. [45], the particle (10, 20), and the generation (1000, 2000); 16 groups of experiments were designed, respectively, with each group being repeated 10 times to acquire the average convergence value, completion time, and convergence time. The optimal parameters of the experimental result were as follows: particle = 20, $c_1 = 2.8, c_2 = 1.3$, velocity = 50,

and generation = 2000. The experimental results are shown in Table 4. PSOA_VMM used the following values: particle (10, 20), velocity (30, 50), and generation (1000, 2000), to conduct eight groups of experimental designs, respectively, with each group repeated 10 times to acquire the average convergence value, completion time, and convergence time. The optimal parameters of the experimental results were as follows: particle = 20; velocity = 50; generation = 2000. The experimental results are shown in Table 5.

For the hardware configuration of this experiment, the CPU was P4-3.0 GHz and the RAM DDR was 512 MB. This study used ANOVA and Scheffé to verify system operation times and convergence times and to select the indices for the GA and the three renovation methods. The Scheffé method was first promoted by Scheffé [65] to assess the relationship among the selection factors. ANOVA is a statistical technique that can be used to evaluate whether there are differences between the average values or means across several population groups. The Scheffé method, one of the multiple-comparison approaches, refers to tests designed to establish whether there are differences between particular levels in an ANOVA design, that is, to determine which variable among several independent variables is statistically the most different. The verification results are shown in Tables 6, 7, and 8.

Tables 6–8 show that all H_0 are rejected. Finally, the Scheffé method was used to make multiple comparisons of the selection index, system execution time, and convergence time of all the algorithms, and their differences. The Scheffé formula is presented as

$$\left(x_i - x_j - \sqrt{(k-1) F_{\alpha(k-1)(n-k)}} \sqrt{\text{MSE} \left(\frac{1}{n_i} + \frac{1}{n_j} \right)}, \right. \\ \left. x_i - x_j + \sqrt{(k-1) F_{\alpha(k-1)(n-k)}} \sqrt{\text{MSE} \left(\frac{1}{n_i} + \frac{1}{n_j} \right)} \right). \tag{23}$$

TABLE 1: Data of transportation cost and defect rate.

Transportation line	1.1-2.1	1.1-2.2	1.1-2.3	1.1-2.4	1.2-2.1	1.2-2.2	1.2-2.3	1.2-2.4	1.3-2.1	1.3-2.2	1.3-2.3	1.3-2.4	2.1-3.1	2.1-3.2	2.1-3.3	2.1-3.4
Transportation defect rate	0.01	0.03	0.02	0.02	0.02	0.01	0.04	0.05	0.04	0.03	0.05	0.01	0.03	0.02	0.01	0.04
Transportation cost	5	6	10	3	7	5	6	4	4	3	5	10	4	5	3	4
Transportation line	2.1-3.5	2.2-3.1	2.2-3.2	2.2-3.3	2.2-3.4	2.2-3.5	2.3-3.1	2.3-3.2	2.3-3.3	2.3-3.4	2.3-3.5	2.4-3.1	2.4-3.2	2.4-3.3	2.4-3.4	2.4-3.5
Transportation defect rate	0.02	0.02	0.03	0.04	0.02	0.05	0.03	0.04	0.02	0.03	0.01	0.03	0.04	0.02	0.01	0.02
Transportation cost	5	4	5	6	2	2	5	8	5	4	6	5	6	5	4	5
Transportation line	3.1-4.1	3.1-4.2	3.1-4.3	3.1-4.4	3.1-4.5	3.1-4.6	3.2-4.1	3.2-4.2	3.2-4.3	3.2-4.4	3.2-4.5	3.2-4.6	3.3-4.1	3.3-4.2	3.3-4.3	3.3-4.4
Transportation defect rate	0.03	0.02	0.01	0.02	0.01	0.03	0.02	0.03	0.01	0.04	0.02	0.03	0.02	0.03	0.04	0.05
Transportation cost	4	5	4	6	4	6	4	6	3	5	4	5	5	6	4	8
Transportation line	3.3-4.5	3.3-4.6	3.4-4.1	3.4-4.2	3.4-4.3	3.4-4.4	3.4-4.5	3.4-4.6	3.5-4.1	3.5-4.2	3.5-4.3	3.5-4.4	3.5-4.5	3.5-4.6		
Transportation defect rate	0.02	0.01	0.02	0.03	0.05	0.04	0.02	0.03	0.03	0.02	0.03	0.04	0.05	0.03		
Transportation cost	6	5	4	4	6	5	5	6	2	5	6	5	4	2		

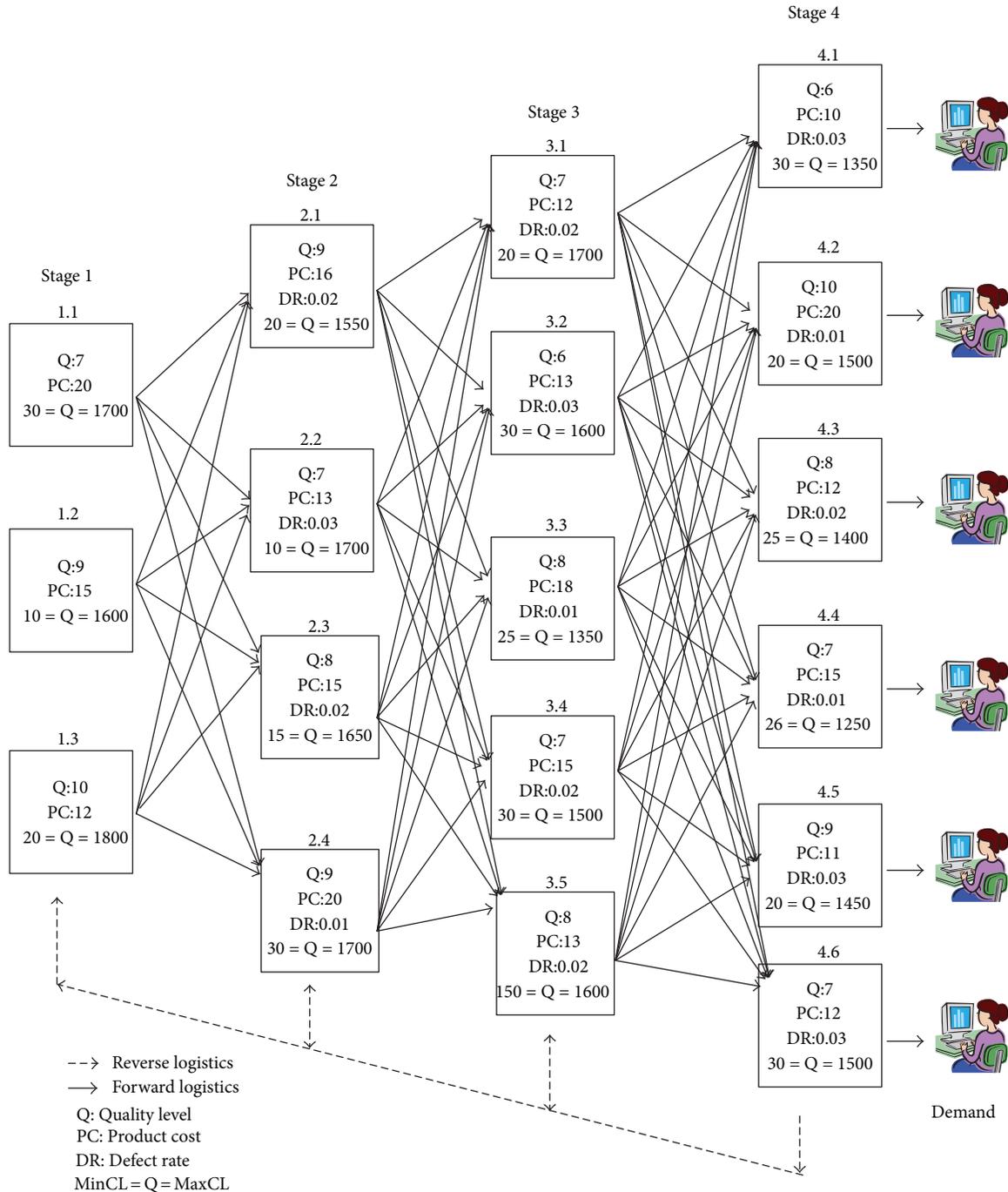


FIGURE 10: {3-4-5-6} forward and reverse supply chain network.

Table 9 shows $Obj_{PSOA_VMM} > Obj_{GA} = Obj_{PSOA_IWM} = Obj_{PSOA_CFM}$; that is, GA, PSOA_IWM, and PSOA_CFM are all better than PSOA_VMM, and there are no clear differences in the selection indices of the three algorithms. The comparative result of system execution is shown in Table 10, and $FT_{GA} > FT_{PSOA_IWM} = FT_{PSOA_VMM} = FT_{PSOA_CFM}$ is the three PSO updating methods that are all superior to GA. The convergence times of the algorithms are shown in Table 11, and $CT_{GA} > CT_{PSOA_CFM} > CT_{PSOA_VMM} >$

CT_{PSOA_IWM} , that is, PSOA_IWM, has faster convergence speed than PSOA_VMM, PSOA_VMM, and GA. The results show that PSOA_IWM performs better in objective function value solutions, execution times, and convergence times.

For validating the solving capabilities of the proposed approaches in cross-stage reverse logistics problems, more large-scale network structures {6-6-6-6}, {6-6-6-6-6}, {3-10-10-60}, {6-8-8-10-30}, and {8-10-20-20-60} were demonstrated. The analysis results also show that PSOA_IWM has

TABLE 2: Experimental design results of GA with different groups of parameters.

Generation	Population	Mutation rate	GA		Objective function value	
			Crossover rate	Convergence time (S)		
1000	10	0.02	0.6	45.65	65.22	585845.5
			0.95	36.91	52.72	586401.9
		0.05	0.6	58.88	84.12	582052.4
			0.95	60.12	85.88	585355.2
	20	0.02	0.6	99.07	141.53	585731.7
			0.95	58.49	83.57	579156.2
		0.05	0.6	111.26	158.94	578879.4
			0.95	94.15	134.51	578760.4
2000	10	0.02	0.6	59.42	112.12	583037.4
			0.95	51.87	97.88	587388.3
		0.05	0.6	90.85	171.42	576988.8
			0.95	89.87	169.58	580298.8
	20	0.02	0.6	88.23	166.49	576920.3
			0.95	72.51	136.81	579347.1
		0.05	0.6	155.42	293.72	575504.7
			0.95	133.79	252.45	576902.9

TABLE 3: Experimental design results of PSOA.IWM with different groups of parameters.

Generation	Particle	Velocity	PSOA.IWM		Objective function value	
			Weight	Convergence time (S)		
1000	10	20	0.4	1.17	2.49	581660.2
			0.9	1.26	2.69	585355.5
		50	0.4	2.28	4.86	588147.3
			0.9	2.94	5.62	593565.2
	20	20	0.4	3.45	5.21	593858.8
			0.9	2.71	5.77	582468.5
		50	0.4	5.76	10.12	581133.8
			0.9	6.17	11.21	589921.1
2000	10	20	0.4	2.96	4.89	612950.3
			0.9	2.38	5.21	581756.6
		50	0.4	4.69	9.24	584003.4
			0.9	5.10	10.26	585192.6
	20	20	0.4	4.73	9.31	591258.4
			0.9	5.02	10.06	580391.2
		50	0.4	7.56	18.88	573972.1
			0.9	11.94	22.34	580979.8

better capabilities for the proposed problems, as shown in Table 12. Therefore, this study used PSOA.IWM to solve cross-stage reverse logistics problems.

Tables 13, 14, and 15 show the received forward and reverse transportation volume of the three periods; since there were no defective products generated in the first period, there is no returned transportation volume. While this study considers the transportation losses and manufacturer's losses, upstream suppliers produced more products than required to ensure

that final demand was met. The quantity of defective products from the second stage was acquired through the defective product rate of all the suppliers. The reverse transportation volume was divided and returned to the upstream supply chain partners, respectively, according to the splitting ratio of defective product quantity. For example, 30% of the defective products generated by the fourth stage retailer would be returned to the third stage, 30% to the second stage, and the rest would be returned to the first stage; the third stage would

TABLE 4: Experimental design results of PSOA_CFM with different groups of parameters.

Generation	Particle	Velocity	PSOA_CFM		Execution time (S)	Objective function value
			c_1, c_2	Convergence time (S)		
1000	10	20	2.05, 2.05	2.77	3.76	596860.2
			2.8, 1.3	1.84	3.92	580909.7
	50	2.05, 2.05	4.85	8.18	591679.5	
		2.8, 1.3	4.14	7.95	575782.2	
	20	20	2.05, 2.05	3.37	5.05	604557
		50	2.8, 1.3	4.43	7.29	588076.4
2000	10	20	2.05, 2.05	6.45	9.08	579670.6
			2.8, 1.3	6.20	8.68	574969.5
		50	2.05, 2.05	11.53	19.22	601022.2
			2.8, 1.3	9.86	16.44	583479.2
	20	20	2.05, 2.05	8.34	15.57	594554.6
			2.8, 1.3	9.91	16.53	579629.1
		50	2.05, 2.05	22.64	37.74	605729.6
			2.8, 1.3	18.84	31.40	574033.9

TABLE 5: Experimental design results of PSOA_VMM with different groups of parameters.

Generation	Particle	Velocity	PSOA_VMM		Objective function value
			Convergence time (S)	Execution time (S)	
1000	10	20	1.97	2.95	620767.6
		50	3.89	5.81	626354.4
	20	20	3.91	5.83	621080.6
		50	8.74	13.04	616409.3
2000	10	20	3.53	5.21	617751.6
		50	7.27	12.12	614025.9
	20	20	7.13	10.05	614276.1
		50	14.45	24.09	609603.7

TABLE 6: ANOVA verification of objective function.

Algorithm	Total	Average	Variance
GA	17244286.9	574809.5	32263393.4
PSOA_IWM	17206940.5	573564.6	25588275.7
PSOA_VMM	18113738.4	603791.2	106679602.3
PSOA_CFM	17252708.0	575090.2	157150309.7

Hypothesis: $H_0 : \text{Obj}_{\text{GA}} = \text{Obj}_{\text{PSOA_IWM}} = \text{Obj}_{\text{PSOA_VMM}} = \text{Obj}_{\text{PSOA_CFM}}$ $H_1 : \text{otherwise}$
 $P \text{ value} = 3.59E - 28 \Rightarrow H_0 \text{ is rejected.}$

TABLE 7: ANOVA verification of completion time.

Algorithm	Total	Average	Variance
GA	92075	306.9	3999.0
PSOA_IWM	567.3	18.9	8.3
PSOA_VMM	608.7	20.2	12.5
PSOA_CFM	661.2	22.0	15.5

Hypothesis: $H_0 : FT_{\text{GA}} = FT_{\text{PSOA_IWM}} = FT_{\text{PSOA_VMM}} = FT_{\text{PSOA_CFM}}$ $H_1 : \text{otherwise}$
 $P \text{ value} = 7.62E - 71 \Rightarrow H_0 \text{ is rejected.}$

TABLE 8: ANOVA verification of convergence time.

Algorithm	Total	Average	Variance
GA	4662.6	155.4	96.7
PSO_IWM	226.9	7.5	3.2
PSO_VMM	433.2	14.4	10.1
PSO_CFM	567.2	18.9	12.0

Hypothesis: $H_0 : CT_{GA} = CT_{PSOA_IWM} = CT_{PSOA_VMM} = CT_{PSOA_CFM}$ H_1 : otherwise
 P -value = $3.25E - 122 \Rightarrow H_0$ is rejected

TABLE 9: Multiple comparison on objective function.

	Obj _{GA}	Obj _{PSOA_IWM}	Obj _{PSOA_VMM}
Obj _{PSOA_IWM}	(-, +)		
Obj _{PSOA_VMM}	(-, -)	(-, -)	
Obj _{PSOA_CFM}	(+, -)	(-, +)	(+, +)

TABLE 10: Multiple comparison on execution time.

	FT _{GA}	FT _{PSOA_IWM}	FT _{PSOA_VMM}
FT _{PSOA_IWM}	(+, +)		
FT _{PSOA_VMM}	(+, +)	(-, +)	
FT _{PSOA_CFM}	(+, +)	(-, +)	(-, +)

TABLE 11: Multiple comparison on convergence time.

	CT _{GA}	CT _{PSOA_IWM}	CT _{PSOA_VMM}
CT _{PSOA_IWM}	(+, +)		
CT _{PSOA_VMM}	(+, +)	(-, -)	
CT _{PSOA_CFM}	(+, +)	(-, -)	(-, -)

TABLE 12: Analysis results on different network structures.

	Network	GA	PSOA_IWM	PSOA_CFM	PSOA_VMM
Objective function	3-4-5-6	574809.5 ^a /1 ^b	573564.6/1	575096.2/1	603791.2/2
	6-6-6-6	644482.1/2	642426.8/1	650475.1/3	725523.7/4
	3-10-10-60	972412.2/2	954457.3/1	980211.5/3	1022415.6/4
	6-6-6-6-6	760460.1/2	758655.5/1	761552.1/3	823544.4/4
	6-8-8-10-30	1201225.3/2	1153252.1/1	1242273.4/3	1345758.7/4
	8-10-20-20-60	1685442.3/2	1637241.6/1	1711412.5/3	1811279.4/4
Execution time	3-4-5-6	306.9/2	18.9/1	22.0/1	20.2/1
	6-6-6-6	326.4/2	41.8/1	42.4/1	39.0/1
	3-10-10-60	621.4/4	74.5/3	65.8/2	61.3/1
	6-6-6-6-6	533.1/3	61.0/2	51.3/1	48.4/1
	6-8-8-10-30	782.6/4	112.5/3	92.1/2	85.2/1
	8-10-20-20-60	997.8/4	187.4/3	138.5/2	102.7/1
Convergence time	3-4-5-6	155.4/4	7.5/1	18.9/3	14.4/2
	6-6-6-6	196.6/2	18.6/1	22.7/1	22.6/1
	3-10-10-60	415.3/3	28.7/1	30.2/2	31.2/2
	6-6-6-6-6	302.9/3	23.6/1	26.5/2	27.2/2
	6-8-8-10-30	557.4/3	35.2/1	40.7/2	42.5/2
	8-10-20-20-60	632.5/4	39.8/1	44.2/2	48.6/3

^aAverage value; ^branking (by multiple comparison).

TABLE 13: The first period transportation plan by PSOA_IWM.

From	To	Stage 1			Stage 2				Stage 3					Stage 4					
		1.1	1.2	1.3	2.1	2.2	2.3	2.4	3.1	3.2	3.3	3.4	3.5	4.1	4.2	4.3	4.4	4.5	4.6
Stage 1	1.1				0	0	0	34											
	1.2				5	646	17	0											
	1.3				1615	154	6	15											
Stage 2	2.1								379	392	190	237	327						
	2.2								38	0	8	261	459						
	2.3								0	10	10	0	2						
	2.4								0	0	48	0	0						
Stage 3	3.1													6	28	194	163	1	5
	3.2													0	9	114	13	146	100
	3.3													0	0	0	27	168	56
	3.4													126	61	1	154	0	137
	3.5													291	366	0	8	0	72
Demand													400	450	300	350	300	350	

TABLE 14: The second period transportation plan by PSOA_IWM.

From	To	Stage 1			Stage 2				Stage 3					Stage 4					
		1.1	1.2	1.3	2.1	2.2	2.3	2.4	3.1	3.2	3.3	3.4	3.5	4.1	4.2	4.3	4.4	4.5	4.6
Stage 1	1.1				29	142	72	158											
	1.2				120	43	150	24											
	1.3				576	845	194	128											
Stage 2	2.1	12	0	19					177	115	148	90	164						
	2.2	9	6	8					189	139	51	183	421						
	2.3	0	0	0					104	78	130	74	12						
	2.4	0	0	0					136	30	15	91	37						
Stage 3	3.1	1	2	1	3	1	0	0						67	95	130	69	126	95
	3.2	2	4	0	2	0	3	0						41	54	111	106	27	5
	3.3	0	0	1	0	1	0	0						11	2	74	73	81	95
	3.4	2	0	3	1	1	2	0						36	45	23	148	127	43
	3.5	2	1	5	0	1	1	5						162	165	131	24	92	25
Stage 4	4.1	2	0	2	0	4	0	0	0	1	1	0	2						
	4.2	0	2	0	0	1	0	0	0	1	0	0	0						
	4.3	2	0	0	0	0	0	2	0	0	0	1	1						
	4.4	1	0	0	0	0	1	0	0	0	0	0	1						
	4.5	0	3	0	1	2	0	0	2	1	0	0	0						
	4.6	3	1	0	1	2	0	0	2	0	0	0	1						
Demand													300	350	450	400	430	250	

Bold data are the reverse transportation volumes.

return 50% to the second stage, the rest would be returned to the first stage, and the second stage supplier would directly return the defective products to the first stage.

4. Conclusion and Suggestion

Enterprises should react to market changes to meet consumer demands in a timely manner to maintain and enhance competitive advantages in this rapidly changing market.

The cross-stage reverse logistics course described in this study could help downstream partners return defective products to the upstream partners directly for maintaining and recovering product function, which in turn could reduce transportation costs and time. With this paper, we have accomplished three tasks. (1) We presented a mathematical model for partner selection and production-distribution planning in multistage supply chain networks with cross-stage reverse logistics. Based on our research, a mathematical

TABLE 15: The third period transportation plan by PSOA_IWM.

From	To	Stage 1			Stage 2				Stage 3					Stage 4					
		1.1	1.2	1.3	2.1	2.2	2.3	2.4	3.1	3.2	3.3	3.4	3.5	4.1	4.2	4.3	4.4	4.5	4.6
Stage 1	1.1				227	24	48	73											
	1.2				17	47	0	65											
	1.3				900	738	84	37											
Stage 2	2.1	7	3	3					133	156	180	239	383						
	2.2	4	2	23					172	121	144	234	96						
	2.3	2	4	1					45	0	66	27	0						
	2.4	2	1	0					3	0	73	33	71						
Stage 3	3.1	5	0	1	2	2	1	4						0	4	129	11	134	64
	3.2	3	2	0	2	0	2	1						4	3	100	53	57	49
	3.3	1	0	1	0	1	0	0						0	81	75	50	139	107
	3.4	0	1	3	0	0	4	0						80	50	49	174	36	125
	3.5	1	5	0	1	3	1	1						180	172	64	27	0	77
Stage 4	4.1	1	1	1	0	0	2	1	1	1	0	1	0						
	4.2	0	1	0	0	0	1	0	0	1	0	0	0						
	4.3	1	0	2	0	0	0	3	0	0	2	0	1						
	4.4	2	0	0	0	0	1	0	0	0	0	1	0						
	4.5	3	0	2	3	0	0	1	3	0	1	0	0						
	4.6	1	0	2	1	0	1	0	0	1	0	0	1						
Demand														250	300	400	300	350	400

Bold data are the reverse transportation volumes.

model for solving multistage supply chain design problems considering the cross-stage reverse logistics has yet to be presented. However, cross-stage reverse logistics should meet the practical logistics operation conditions; therefore, (2) we applied a GA and three PSO algorithms to efficiently solve the mathematical model of cross-stage reverse logistics problems. In this paper, we emphasized the suitability of adopting a GA and three PSOs to find the solution to the mathematical model; hence, (3) we compared four proposed algorithms to find which one works best with the proposed problem. The comprehensive results show that PSOA_IWM has the qualities and capabilities for dealing with a multi-stage supply chain design problem with cross-stage reverse logistics. Further research should be conducted to employ other heuristic algorithms such as ant colony and simulated annealing for solving this problem. Consideration should also be given to extending this developed approach to encompass more complex problems such as problems involving resource constraints, transportation, and economic batches.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank Mr. K. Hsiao for supporting writing of programs and the National Science Council of Taiwan for their partial financial support (Grants no. NSC 102-2410-H-027-009 and NSC 101-2410-H-027-006). The authors

would also like to acknowledge the editors and anonymous reviewers for their helpful comments and suggestions, which greatly improved the presentation of this paper.

References

- [1] C. J. Vidal and M. Goetschalckx, "Strategic production-distribution models: a critical review with emphasis on global supply chain models," *European Journal of Operational Research*, vol. 98, no. 1, pp. 1-18, 1997.
- [2] B. M. Beamon, "Supply chain design and analysis: models and methods," *International Journal of Production Economics*, vol. 55, no. 3, pp. 281-294, 1998.
- [3] Ş. S. Erengüç, N. C. Simpson, and A. J. Vakharia, "Integrated production/distribution planning in supply chains: an invited review," *European Journal of Operational Research*, vol. 115, no. 2, pp. 219-236, 1999.
- [4] J. Xu, Q. Liu, and R. Wang, "A class of multi-objective supply chain networks optimal model under random fuzzy environment and its application to the industry of Chinese liquor," *Information Sciences*, vol. 178, no. 8, pp. 2022-2043, 2008.
- [5] R. A. Aliev, B. Fazlollahi, B. G. Guirimov, and R. R. Aliev, "Fuzzy-genetic approach to aggregate production-distribution planning in supply chain management," *Information Sciences*, vol. 177, no. 20, pp. 4241-4255, 2007.
- [6] S.-W. Chiou, "A non-smooth optimization model for a two-tiered supply chain network," *Information Sciences*, vol. 177, no. 24, pp. 5754-5762, 2007.
- [7] D. Y. Sha and Z. H. Che, "Virtual integration with a multi-criteria partner selection model for the multi-echelon manufacturing system," *The International Journal of Advanced Manufacturing Technology*, vol. 25, no. 7-8, pp. 793-802, 2005.

- [8] D. Y. Sha and Z. H. Che, "Supply chain network design: partner selection and production/distribution planning using a systematic model," *Journal of the Operational Research Society*, vol. 57, no. 1, pp. 52–62, 2006.
- [9] Z. H. Che and Z. Cui, "Unbalanced supply chain design using the analytic network process and a hybrid heuristic-based algorithm with balance modulating mechanism," *The International Journal of Bio-Inspired Computation*, vol. 3, no. 1, pp. 56–66, 2011.
- [10] J. R. Stock, *Reverse Logistics*, Council of Logistics Management, Oak Brook, Ill, USA, 1992.
- [11] B. Trebilcock, "Reverse logistics heroes," *Modern Materials Handling*, vol. 56, no. 10, pp. 63–65, 2001.
- [12] M. Cohen, "Replace. Rebuild or remanufacture," *Equipment Management*, vol. 16, no. 1, pp. 22–26, 1988.
- [13] C. D. White, E. Masanet, C. M. Rosen, and S. L. Beckman, "Product recovery with some byte: an overview of management challenges and environmental consequences in reverse manufacturing for the computer industry," *Journal of Cleaner Production*, vol. 11, no. 4, pp. 445–458, 2003.
- [14] J. Gattorna, *Strategic Supply Chain Alignment-Best Practice in Supply Chain Management*, Ashgate, 1998.
- [15] C. R. Carter and L. M. Ellram, "Reverse logistics: a review of the literature and framework for future investigation," *Journal of Business Logistics*, vol. 19, no. 1, pp. 85–102, 1998.
- [16] S. Dowlatshahi, "Developing a theory of reverse logistics," *Interfaces*, vol. 30, no. 3, pp. 143–155, 2000.
- [17] M. Fleischmann, J. M. Bloemhof-Ruwaard, R. Dekker, E. van der Laan, J. A. E. E. van Nunen, and L. N. van Wassenhove, "Quantitative models for reverse logistics: a review," *European Journal of Operational Research*, vol. 103, no. 1, pp. 1–17, 1997.
- [18] D. S. Rogers and R. Tibben-Lembke, "An examination of reverse logistics practices," *Journal of Business Logistics*, vol. 22, no. 2, pp. 129–148, 2001.
- [19] T. Spengler, H. Püchert, T. Penkuhn, and O. Rentz, "Environmental integrated production and recycling management," *European Journal of Operational Research*, vol. 97, no. 2, pp. 308–326, 1997.
- [20] V. Jayaraman, V. D. R. Guide Jr., and R. Srivastava, "A closed-loop logistics model for remanufacturing," *Journal of the Operational Research Society*, vol. 50, no. 5, pp. 497–508, 1999.
- [21] A. I. Barros, R. Dekker, and V. Scholten, "A two-level network for recycling sand: a case study," *European Journal of Operational Research*, vol. 110, no. 2, pp. 199–214, 1998.
- [22] L. Kroon and G. Vrijens, "Returnable containers: an example of reverse logistics," *International Journal of Physical Distribution & Logistics Management*, vol. 25, no. 2, pp. 56–68, 1995.
- [23] M. Fleischmann, *Quantitative Models for Reverse Logistics*, Springer, Berlin, Germany, 2001.
- [24] M. M. Amini, D. Retzlaff-Roberts, and C. C. Bienstock, "Designing a reverse logistics operation for short cycle time repair services," *International Journal of Production Economics*, vol. 96, no. 3, pp. 367–380, 2005.
- [25] M. Fleischmann, P. Beullens, J. M. Bloemhof-Ruwaard, and L. N. van Wassenhove, "The impact of product recovery on logistics network design," *Production and Operations Management*, vol. 10, no. 2, pp. 156–173, 2001.
- [26] R. C. Savaskan, S. Bhattacharya, and L. N. V. Wassenhove, "Closed loop supply chain models with product remanufacturing," *Management Science*, vol. 50, no. 2, pp. 239–252, 2004.
- [27] M. Chouinard, S. D'Amours, and D. Aït-Kadi, "Integration of reverse logistics activities within a supply chain information system," *Computers in Industry*, vol. 56, no. 1, pp. 105–124, 2005.
- [28] Y. Kainuma and N. Tawara, "A multiple attribute utility theory approach to lean and green supply chain management," *International Journal of Production Economics*, vol. 101, no. 1, pp. 99–108, 2006.
- [29] A. Nagurney and F. Toyasaki, "Reverse supply chain management and electronic waste recycling: a multitiered network equilibrium framework for e-cycling," *Transportation Research E*, vol. 41, no. 1, pp. 1–28, 2005.
- [30] Y. Nikolaidis, "A modelling framework for the acquisition and remanufacturing of used products," *International Journal of Sustainable Engineering*, vol. 2, no. 3, pp. 154–170, 2009.
- [31] G. Nenes and Y. Nikolaidis, "A multi-period model for managing used product returns, international," *Journal of Production Research*, vol. 50, pp. 1360–1376, 2012.
- [32] M. Salema, A. Barbosa-Póvoa, and A. Novais, "Simultaneous design and planning of supply chains with reverse flows: a generic modelling framework," *European Journal of Operational Research*, vol. 203, no. 2, pp. 336–349, 2010.
- [33] T. Pinto-Varela, A. P. Barbosa-Póvoa, and A. Q. Novais, "Bi-objective optimization approach to the design and planning of supply chains: economic versus environmental performances," *Computers and Chemical Engineering*, vol. 35, no. 8, pp. 1454–1468, 2011.
- [34] S. Amin and G. Zhang, "A proposed mathematical model for closed-loop network configuration based on product life cycle," *International Journal of Advanced Manufacturing Technology*, vol. 58, no. 5–8, pp. 791–801, 2012.
- [35] M. Huang, M. Song, L. H. Lee, and W. K. Ching, "Analysis for strategy of closed-loop supply chain with dual recycling channel," *International Journal of Production Economics*, vol. 144, no. 2, pp. 510–520, 2013.
- [36] P. L. Meena and S. P. Sarmah, "Multiple sourcing under supplier failure risk and quantity discount: a genetic algorithm approach," *Transportation Research E*, vol. 50, pp. 84–97, 2013.
- [37] B. Stojanovic, M. Milivojevic, M. Ivanovic, N. Milivojevic, and D. Divac, "Adaptive system for dam behavior modeling based on linear regression and genetic algorithms," *Advances in Engineering Software*, vol. 65, pp. 182–190, 2013.
- [38] R. Belevičius, D. Jatulis, and D. Šešok, "Optimization of tall guyed masts using genetic algorithms," *Engineering Structures*, vol. 56, pp. 239–245, 2013.
- [39] Z. H. Che and C. J. Chiang, "A modified Pareto genetic algorithm for multi-objective build-to-order supply chain planning with product assembly," *Advances in Engineering Software*, vol. 41, no. 7–8, pp. 1011–1022, 2010.
- [40] S. P. Nachiappan and N. Jawahar, "A genetic algorithm for optimal operating parameters of VMI system in a two-echelon supply chain," *European Journal of Operational Research*, vol. 182, no. 3, pp. 1433–1452, 2007.
- [41] H. S. Wang and Z. H. Che, "An integrated model for supplier selection decisions in configuration changes," *Expert Systems with Applications*, vol. 32, no. 4, pp. 1132–1140, 2007.
- [42] Z. H. Che and T. A. Chiang, "Designing a collaborative supply chain plan using the analytic hierarchy process and genetic algorithm with cycle time estimation," *International Journal of Production Research*, vol. 50, no. 16, pp. 4426–4443, 2012.
- [43] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.

- [44] Z. Liao and J. Rittscher, "A multi-objective supplier selection model under stochastic demand conditions," *International Journal of Production Economics*, vol. 105, no. 1, pp. 150–159, 2007.
- [45] L.-P. Zhang, H.-J. Yu, and S.-X. Hu, "Optimal choice of parameters for particle swarm optimization," *Journal of Zhejiang University Science A*, vol. 6, no. 6, pp. 528–534, 2005.
- [46] X. H. Shi, Y. C. Liang, C. Lu, H. P. Lee, and Q. X. Wang, "Particle swarm optimization-based algorithms for TSP and generalized TSP," *Information Processing Letters*, vol. 103, no. 5, pp. 169–176, 2007.
- [47] Z. H. Che, "PSO-based back-propagation artificial neural network for product and mold cost estimation of plastic injection molding," *Computers and Industrial Engineering*, vol. 58, no. 4, pp. 625–637, 2010.
- [48] Z. H. Che, "A particle swarm optimization algorithm for solving unbalanced supply chain planning problems," *Applied Soft Computing*, vol. 12, no. 4, pp. 1279–1287, 2012.
- [49] C. Priya and P. Lakshmi, "Particle swarm optimisation applied to real time control of spherical tank system," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 206–216, 2012.
- [50] L. Ali, S. L. Sabat, and S. K. Udgate, "Particle swarm optimization with stochastic ranking for constrained numerical and engineering benchmark problems," *International Journal of Bio-Inspired Computation*, vol. 4, no. 3, pp. 155–166, 2012.
- [51] E. García-Gonzalo and J. L. Fernández-Martínez, "A brief historical review of particle swarm optimization (PSO)," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 3–16, 2012.
- [52] L. Ali and S. L. Sabat, "Particle swarm optimization based universal solver for global optimization," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 95–105, 2012.
- [53] M. Salehi Maleh, S. Soleymani, R. Rasouli Nezhad, and N. Ghadimi, "Using particle swarm optimization algorithm based on multi-objective function in reconfigured system for optimal placement of distributed generation," *Journal of Bioinformatics and Intelligent Control*, vol. 2, no. 2, pp. 119–124, 2013.
- [54] H. M. Abdelsalam and A. M. Mohamed, "Optimal sequencing of design projects' activities using discrete particle swarm optimisation," *International Journal of Bio-Inspired Computation*, vol. 4, no. 2, pp. 100–110, 2012.
- [55] Y. Dong, J. Tang, B. Xu, and D. Wang, "An application of swarm optimization to nonlinear programming," *Computers and Mathematics with Applications*, vol. 49, no. 11–12, pp. 1655–1668, 2005.
- [56] P.-Y. Yin and J.-Y. Wang, "A particle swarm optimization approach to the nonlinear resource allocation problem," *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 232–242, 2006.
- [57] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, 2002.
- [58] W. A. McCall, *Measurement*, Macmillan, New York, NY, USA, 1939.
- [59] Z. H. Che, "A genetic algorithm-based model for solving multi-period supplier selection problem with assembly sequence," *International Journal of Production Research*, vol. 48, no. 15, pp. 4355–4377, 2010.
- [60] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, New York, NY, USA, 1988.
- [61] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Proceedings of the 7th Annual Conference on Evolutionary Programming*, pp. 611–616, Springer, Berlin, Germany, 1998.
- [62] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," in *Proceeding of the IEEE Congress on Evolutionary Computation*, vol. 3, pp. 1951–1957, 1999.
- [63] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, October 1995.
- [64] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.
- [65] H. A. Scheffé, "A method for judging all contrasts in the analysis of variance," *Biometrika*, vol. 40, no. 1–2, pp. 87–104, 1953.

Research Article

Enhancing Artificial Bee Colony Algorithm with Self-Adaptive Searching Strategy and Artificial Immune Network Operators for Global Optimization

Tinggui Chen¹ and Renbin Xiao²

¹ College of Computer Science & Information Engineering, Zhejiang Gongshang University, Zhejiang Province, Hangzhou 310018, China

² Institute of Systems Engineering, Huazhong University of Science and Technology, Hubei Province, Wuhan 430074, China

Correspondence should be addressed to Renbin Xiao; rbxiao@hust.edu.cn

Received 10 November 2013; Accepted 30 December 2013; Published 18 February 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 T. Chen and R. Xiao. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Artificial bee colony (ABC) algorithm, inspired by the intelligent foraging behavior of honey bees, was proposed by Karaboga. It has been shown to be superior to some conventional intelligent algorithms such as genetic algorithm (GA), artificial colony optimization (ACO), and particle swarm optimization (PSO). However, the ABC still has some limitations. For example, ABC can easily get trapped in the local optimum when handling in functions that have a narrow curving valley, a high eccentric ellipse, or complex multimodal functions. As a result, we proposed an enhanced ABC algorithm called EABC by introducing self-adaptive searching strategy and artificial immune network operators to improve the exploitation and exploration. The simulation results tested on a suite of unimodal or multimodal benchmark functions illustrate that the EABC algorithm outperforms ACO, PSO, and the basic ABC in most of the experiments.

1. Introduction

With the rapid development of communication technology, computer technology, and network technology, humans put forward higher demand for efficient intelligent technologies. However, in view of the complexity, constraint, and nonlinearity of practical issues, searching for all kinds of emerging intelligent computing technologies for solving large and complex problems has been paid attention by more and more scholars.

As one of typical intelligent computing approaches, swarm intelligence that combines biology and social based heuristics has become a research interest to many research scientists of related fields in recent years. It is based on the collective behavior of social insects, flock of birds, or schools of fish. The key components of swarm intelligence are self-organization and division of labor. In a self-organization system, each of the covered units may respond to local stimuli individually and act together to accomplish a global task

via division of labor without a centralized supervision. The entire system can adapt to internal and external changes efficiently [1, 2]. Particle swarm optimization (PSO) algorithm introduced by Hsieh et al. in 2008 [3] can be thought of as a typical swarm whose individual agents are birds and has been widely used in all kinds of combination optimization problems [4–7]. What is more, other algorithms such as ant colony optimization (ACO) [8, 9] and artificial immune network (aiNet) [10, 11] can also be considered as subfields of swarm intelligence.

Nowadays, an artificial bee colony (ABC) algorithm, inspired by the intelligent foraging behavior of honey bees, was proposed by Karaboga [12]. Due to its simplicity and ease of implementation, the ABC algorithm has captured much attention and has been widely applied to solve many practical optimization problems such as supply chain management [13] and scheduling optimization [14]. In addition, a set of well-known numerical comparisons have demonstrated that the performance of ABC algorithm is competitive to

other intelligent ones including genetic algorithm (GA), PSO, differential evolution (DS), and evolution strategy (ES) although it uses fewer control parameters [15–17].

However, similar to other intelligent algorithms, the ABC still has some limitations. For example, the convergence speed of ABC is slow because of its stochastic nature. What is more, ABC can easily get trapped in the local optimum when handling in functions that have a narrow curving valley, a high eccentric ellipse, or complex multimodal functions [18]. All these insufficiencies prevent the further applications of the ABC algorithm.

Therefore, in this work, some modifications to the standard ABC algorithm are introduced for global optimization of numerical functions. Firstly, ABC algorithm is extended by employing the chaotic systems and the diversity-based method when producing the initial population. Next, self-adaptive searching strategy is incorporated into the employed bee search process to improve the exploitation. In addition, the enhanced algorithm retains the main steps of ABC and incorporates an aiNet-based search technique, where aiNet algorithm has powerful multimodal searching ability as well as good stabilization due to its negative selection and network compression operators. Therefore, ABC and aiNet have complementary advantages, and a hybrid of the two may be a possible strategy to improve the performances of ABC.

The remainder of this paper is organized as follows. In Section 2, the works related to the ABC algorithm are summarized. In Section 3, the basic ABC algorithm is described. Section 4 describes the modified ABC algorithm combined with self-adaptive searching strategy and artificial immune network operators. In Section 5, the testing of the proposed algorithm through 15 benchmark functions problems is carried out and the simulation results are compared. Finally, conclusions and future works are provided in Section 6.

2. Previous Works on the ABC Algorithm

The ABC algorithm imitated the foraging behavior of honeybee and was first applied to numerical optimization problems. However, due to its weaknesses mentioned in Section 1, some researchers proposed many improved strategies. For example, Alatas [19] used different chaotic maps to generate sequences substituting random numbers for different parameters of ABC when producing initial population. Moreover, Gao and Liu [18, 20] also employed both the chaotic systems and opposition-based learning methods to enhance the global convergence. In these two literature works, authors also developed an improved solution search equation which was based on that the bee searched only for the best solution of the previous iteration to improve the exploitation. The experiments derived from a set of 28 benchmark functions demonstrated that the performance of this method was better than the other methods. Unlike these studies mentioned above, inspired by PSO, Zhu and Kwong [21] proposed Gbest-guided ABC algorithm by incorporating the information of the global best solution into the solution search equation to improve the exploitation. Banharnsakun et al. [22] presented a best-so-far method for solution updates in the ABC

algorithm, and the searching method based on a dynamic adjustment of search range depending on the iteration was also introduced for scout bees. The test results showed that the proposed method was able to produce higher quality solutions with faster convergence than either the original ABC or the current state-of-the-art ABC-based algorithm.

Besides numerical optimization, the ABC algorithm has been widely used to solve large-scale problems and engineering design optimization. Some representative applications are introduced as follows. Kang et al. [23] used a hybrid ABC algorithm which combines Nelder-Mead simplex method with ABC algorithm for structural inverse analysis problems, and its performance outperforms other heuristic methods. Singh [24] applied the ABC algorithm for the leaf-constrained minimum spanning tree (LCMST) problem and compared the approach against GA, ACO and tabu search. In the literature [24], it was reported that the proposed algorithm was superior to the other methods in terms of solution qualities and computational time. Zhang et al. [25] developed the ABC clustering algorithm to optimally partition N objectives into K cluster and Deb's rules were used to direct the search direction of each candidate. Pan et al. [26] used the discrete ABC algorithm to solve the lot-streaming flow shop scheduling problem with the criterion of total weighted earliness and tardiness penalties under both the idling and no-idling cases. Samanta and Chakraborty [27] employed ABC algorithm to search out the optimal combinations of different operating parameters for three widely used nontraditional machining (NTM) processes, that is, electrochemical machining, electrochemical discharge machining, and electrochemical micromachining processes. Alejandro et al. [28] used the ABC algorithm in order to find the optimal distribution of material with the aim of establishing a standard time for this duty by examining how this was applied in a local manufacturing plant. The simulation results showed that using this approach might be convenient to set the standard times in the selected company. All these researches illustrated that the ABC algorithm has powerful ability to solve much more complex engineering problems.

3. The Original Artificial Bee Colony Algorithm

The artificial bee colony has been inspired by the intelligent behavior of real honey bees. The honey bees in this algorithm are categorized into three groups: employed bees, onlooker bees, and scout bees. The first half of the colony consists of employed bees, and the other half includes the onlookers. Each solution in the search space consists of a set of optimization parameters which represent a food source population. The number of employed bees is equal to the number of food sources around the hive. In other words, for every food source, there is only one employed bee. What is more, onlooker bees wait in the hive and decide on a food source to exploit based on the information shared by the employed bees. Scout bees are translated from a few employed bees whose food source has been exhausted by the bees.

Similar to the other swarm intelligence algorithms, ABC is an iterative process. The units of the original ABC algorithm can be explained as follows.

3.1. The Initial Population of Solutions. The initial population of solutions is filled with SN number of randomly generated D -dimensional real-valued vectors (i.e., food sources). Each food source is generated as follows:

$$x_i^j = x_{\min}^j + \text{rand}(0, 1)(x_{\max}^j - x_{\min}^j), \quad (1)$$

where $i = 1, 2, \dots, \text{SN}$, $j = 1, 2, \dots, D$. x_{\min}^j and x_{\max}^j are the lower and upper bounds for the dimension j , respectively. These food sources are randomly assigned to SN number of employed bees and their fitness is evaluated.

After initialization, the population of the food source is subjected to repeat cycle of the search processes of the employed bees, the onlooker bees, and the scout bees.

3.2. The Search Phase of Employed Bees. In this phase, in order to produce a candidate food position from the old one, the ABC uses the following equation:

$$v_i^j = x_i^j + \varphi_i^j(x_i^j - x_k^j), \quad (2)$$

where $j \in \{1, 2, \dots, D\}$ and $k \in \{1, 2, \dots, \text{SN}\}$ are randomly chosen indexes. Although k is determined randomly, it has to be different from i . φ_i^j is a random number in the range $[-1, 1]$. Equation (2) denotes that, within the neighborhood of every food source site represented by x_i , a food source v_i is determined by changing one parameter of x_i .

Once v_i is obtained, it will be evaluated and compared to x_i . A greedy selection is applied between x_i and v_i ; then the better one is selected depending on fitness values representing the nectar amount of the food sources at x_i and v_i . If the fitness of v_i is equal to or better than that of x_i , v_i will replace x_i and become a new member of the populations; otherwise x_i is retained.

3.3. The Selection Phase of Onlooker Bees. In this phase, each onlooker bee selects one of the food sources depending on the fitness value obtained from the employed bees. The fitness-based probability selection, scheme may be a roulette wheel, ranking based, stochastic universal sampling, tournament selection or another selection scheme. In original ABC, roulette wheel selection scheme is employed described as an equation below:

$$P_i = \frac{\text{fit}(x_i)}{\sum_{m=1}^{\text{SN}} \text{fit}(x_m)}, \quad (3)$$

where $\text{fit}(x_i)$ is the fitness value of solution i . Obviously, the higher the $\text{fit}(x_i)$ is, the more probability is that the i th food source is selected. After the food source is selected, onlooker bees will go to the selected food source and produce a new candidate position in the neighborhood of the selected food source by using (2).

3.4. Scout Bee Phase. In a cycle, after all employed and onlooker bees complete their searches, the ABC algorithm checks if there is any exhausted source to be abandoned. If a position cannot be improved further through a predetermined number of cycles, then that food source is assumed to be abandoned. The scouts can accidentally discover rich, entirely unknown food sources. This operation can be defined as in (4) shown as follows. This process helps avoid suboptimal solutions. The value of predetermined number of cycles is called "limit" for abandoning a food source, which is an important control parameter of ABC algorithm:

$$x_i = x_{\min} + \text{rand}(0, 1)(x_{\max} - x_{\min}), \quad (4)$$

where x_{\min} and x_{\max} are the lower and upper bounds of variable x_i .

3.5. Main Steps of the Original Artificial Colony Bee Algorithm. Based on the above explanation, there are three control parameters used in the original ABC: the number of the food sources which is equal to the number of employed bees (SN), the value of *limit* and the maximum cycle number (MEN). Detailed pseudocode of the ABC algorithm is given in Algorithm 1 [15].

4. Enhancing Artificial Bee Algorithm with Artificial Immune Network

The original version of ABC algorithm is very efficient for multidimensional basic functions. However, the convergence rate of the algorithm is poor when working with some complex multimodal functions and composite functions. Furthermore, due to its poor exploration process, the ABC algorithm easily gets trapped in a local optimum. In order to improve these limitations existing in the ABC algorithm, some modifications inspired by the artificial immune network (ai-Net) algorithm so as to accelerate the convergence rate have been introduced in the search process of the original ABC algorithm. In addition, an improved search mechanism based on the self-adaptive strategy as well as a novel generation method of the initial population is also proposed.

4.1. Generation of the Initial Population. One of the modifications in the ABC algorithm is generating effective initial population, which can affect the convergence rate and the quality of the final solution. Generally, random initialization is the most adopted approach to generate initial population, which often makes solutions concentrated in a local area. Chaotic sequences derived from a chaotic map have been proven easy and fast to store; there is no need for storage of long sequences. Recently, chaotic sequences have been used instead of random sequences and shown somewhat good results in many applications. Therefore, chaotic maps are introduced in ABC to improve the global convergence by escaping the local solutions in [19]. Meanwhile, in order to increase the population diversity, similar individuals should be gotten rid of. The main principle is to compare the affinity inspired by ai-Net algorithm between two different

```

(1) Generate the initial population  $x_i$  ( $i = 1, 2, \dots, SN$ )
(2) Evaluate the fitness ( $fit(x_i)$ ) of the population
(3) Set cycle to 1
(4) Repeat
(5) For each employed bee {
    Produce new solution  $v_i$  by using (2)
    Calculate its fitness value  $fit(v_i)$ 
    Apply greedy selection process}
(6) Calculate the probability values  $P_i$  for the solution ( $x_i$ ) by (3)
(7) For each onlooker bee {
    Select a solution  $x_i$  depending on  $P_i$ 
    Produce new solution  $v_j$ 
    Calculate its fitness value  $fit(v_j)$ 
    Apply greedy selection process}
(8) If there is an abandoned solution for the scout,
    then replace it with a new solution which will be randomly produced by (4)
(9) Memorize the best solution so far
(10) Cycle = cycle +1
(11) Until cycle = MEN

```

ALGORITHM 1: Pseudocode of main body of ABC algorithm.

individuals. As a result, this work proposes a novel initialization approach which uses chaotic systems and affinity-based compression method to produce initial population. Here, according to the literature [19], sinus map is selected and its equation is defined as follows:

$$cm_{n+1} = 2.3(cm_n)^{2 \sin(\pi cm_n)}, \quad (5)$$

$$cm_n = 0, 1, 2, \dots, N,$$

where n is the iteration counter and N is the maximum number of chaotic iterations. Furthermore, the affinity equation between two different individuals is defined as Euclidean distance shown in (6):

$$\text{affinity}(x_i, x_k) = \sqrt{\sum_{j=1}^D (x_i^j - x_k^j)^2}, \quad (6)$$

$$(i \neq k, j \in (1, 2, \dots, D)) > \xi,$$

where ξ is a threshold value defined in advance so as to control the difference between two individuals. D is the number of optimization parameters. Based on these operators, we propose the following algorithm to generate initial population and its corresponding pseudocode is given in Algorithm 2.

4.2. An Improved Search Mechanism Based on the Self-Adaptive Strategy. As mentioned above, the original ABC algorithm is good at exploration but poor at exploitation due to two reasons. On the one hand, in (2), the coefficient ϕ_j^i is a uniform random number in $[-1, 1]$ and x_j^k is a random individual in the population; therefore, the search process of solutions illustrated by (2) is random enough for exploration [21]. On the other hand, a greedy selection mechanism is employed between the old and candidate solutions, which may easily make solutions get trapped in

a local optimal. What is more, the slower convergence rate of the algorithm is another limitation when working with some complex composite functions. As a result, we introduce a self-adaptive strategy to improve its search process and the detailed explanations are as follows.

Firstly, we can see from (2) that there is only one different element between the candidate solution and the old one (i.e., the j th element). This search strategy may be efficient in earlier iterations. However, when the solution approaches to a local optimum, its search efficiency becomes poor in later iterations. To handle this limitation, similar to [29, 30], we introduce a parameter L to control the difference between the candidate solution and the old one, where how to choose the value of the parameter L is very important. Generally, the higher the value of L is, the more information is brought into the candidate solution. In the literature [18], the parameter L is a fixed constant and chosen according to simulation experiments. Different from this approach mentioned above, this paper proposes a self-adaptive adjustable strategy to determine the parameter L and the corresponding search equation is given below:

$$v_i^l = x_i^l + cm_i^l (x_i^l - x_k^l), \quad l = \{1, 2, \dots, L\}, \quad (7)$$

$$L = 1 + \left\| D \left(\frac{\text{iteration}}{2 \times \text{MEN}} \right) \right\|, \quad (8)$$

where cm_i^l is a chaotic map defined by (5). D is the number of optimization parameters. A symbol $\| \cdot \|$ denotes a rounding operator. We can see from (8) that onlooker bees will search better solutions in only one direction in the first iteration and they will search in the whole space with the increase in the value of L when the solutions are closely to the local optimum in later iterations. However, we limit the value of L no more than $\|1 + D/2\|$. This is because, with the higher value of L , the solution generated by the search equations (7)-(8)

```

Set the population size SN, the number of randomly generated individuals SNr ≫ SN, the maximum number of chaotic
iterations N > 200, the number of optimization parameters D, and the individual counter i = 1, j = 1
{- Chaotic systems- }
for i = 1 to SNr do
    for j = 1 to D do
        Randomly initialize the first chaotic variable cm0,j ∈ (0, 1) and set iteration counter n = 0
        for n = 1 to N do
            cmn+1,j = 2.3(cmn,j)2 sin(πcmn,j)
        end for
        xij = xminj + cmnj(xmaxj - xminj)
    end for
end for
{- Affinity-based compression method- }
Set the threshold value ξ and the individual counter i = 1, j = 1
for i = 1 to (SNr - 1) do
    for k = i + 1 to SNr do
        if affinity(xi, xk) = √(∑j=1D (xij - xkj)2) < ξ
            SNr = SNr - 1, delete the individual k in the population
        end if
    end for
end for
Selecting SN fittest individuals form set (X(SNr)) as initial population
    
```

ALGORITHM 2: Modified initialization step of ABC algorithm.

is more likely random search operator. As a result, (7)-(8) will dynamically adjust the position of onlooker bees by allowing them to explore with a wider search space in later iterations. As the number of the iterations increases, the corresponding search space of onlooker bees will also increase.

The second modification in the ABC algorithm lies in the selection probability of onlooker bees associated with the food source. In the original ABC algorithm, the fitness values obtained from the employed bees are adopted to determine the selection probabilities of onlooker bees. Nevertheless, the fitness comparisons among different individuals only reflect the qualitative information. In this work, based on the idea of the fitness evolution, we introduce an environment factor η_i corresponding to every food source so as to evaluate its exploitation potential. In other words, the parameter η_i is used to evaluate quantitatively the environment situations of exploitation for every food source. As the number of iterations increases, the higher the value of η_i , the better exploitation environment it corresponds to. At this moment, more onlooker bees will follow the corresponding employed one to its food source position with higher nectar amount in order to accelerate exploitation efficiency. Conversely, if the value of η_i is lower, its corresponding solution lies in the worse exploitation environment, which means that it is difficult to find out a better solution around the old food source and less onlooker bees will be attracted by the employed one. As a result, how to define η_i needs explain. Generally, the parameter η_i is associated with both a fitness change amount Δf and a count accumulator C , where Δf denotes the fitness difference associated with the same food source between two adjacent generations which reflects the exploitation

potential of the corresponding food source position. Besides, a count accumulator C will be explained in the following text. The equations below reflect the fitness change amount Δf between two adjacent generations:

$$\Delta f_t(x_i) = \left| \frac{\text{fit}_t(x_i) - \text{fit}_{t-1}(x_i)}{\text{fit}_{t-1}(x_i)} \right| \tag{9}$$

$$\Delta f'_t(x_i) = \exp[\Delta f_t(x_i)] = \log \left| \frac{\text{fit}_t(x_i) - \text{fit}_{t-1}(x_i)}{\text{fit}_{t-1}(x_i)} \right|, \tag{10}$$

where the parameter t denotes the number of iterations and the symbol $| |$ means absolute value sign. As can be seen from (9), the value range of Δf is between 0 and 1. When the value of Δf is higher, it means the corresponding food source has a higher exploitation potential and is largely possible to find out a better solution and vice versa. However, at later iterations, the value of Δf will be very small and need appropriate amplification. As a result, we adopt power function to amplify Δf here and take the number e as its base. Due to this reason, (9) is substituted for (10).

In addition, if Δf is less than a given small value in advance (i.e., $\Delta f \leq \Delta f_0$, and Δf_0 is the threshold of Δf), the improved ABC algorithm may trigger a count accumulator called *Counter* (represented by C in this paper) which is used to record how many times the quality of the solution has not improved (it corresponds to the number of iterations in the algorithm). The following equation is used to express the count accumulator C at the t th iteration. Generally, the larger

the value of C , the less exploitation potential that the food source position corresponds to:

$$C(t) = \begin{cases} C(t-1) + T(t) = \sum_{i=s}^t T(i), & T(t) = 1, \\ 0, & T(t) = 0, \end{cases} \quad (11)$$

where T represents a pulse signal. When $\Delta f > 0$ or $\Delta f \geq \Delta f_0$, $T = 0$; conversely, when $\Delta f = 0$ or $\Delta f \leq \Delta f_0$, $T = 1$. It means that

$$T(t) = \begin{cases} 1, & (\Delta f(t) = 0) \text{ or } (\Delta f(t) \leq \Delta f(t)_0), \\ 0, & (\Delta f(t) > 0) \text{ or } (\Delta f(t) \geq \Delta f(t)_0). \end{cases} \quad (12)$$

Note that, according to the performance requirement of specific problems, the maximum times of the stagnation of global extrema, C_{\max} , should be defined in advance, which means, if a minimum of a function has not been updated for continuous C_{\max} iterations, the current exploitation area has few potentials to find out the better solution and computation resources should be redistributed. Generally, we define that the C_{\max} is equal to or not less than 5. In addition, $C(t)$ is normalized for simplifying the problem and $C'(t)$ can be obtained which is expressed as follows:

$$C'(t) = \frac{\sum_{i=s}^t T(i) + 1}{C_{\max} + 1}. \quad (13)$$

On the basis of the definitions of the fitness change amount Δf and the count accumulator C , the environment factor η is defined as follows:

$$\eta(t) = \frac{\Delta f'(t)}{C'(t)}. \quad (14)$$

Submitting (10) and (13) into (14), we can achieve the following equation:

$$\eta(t) = \frac{(C_{\max} + 1) \cdot \exp\left[\frac{(\text{fit}_t(x_i) - \text{fit}_{t-1}(x_i))}{\text{fit}_{t-1}(x_i)}\right]}{\sum_{i=s}^t T(i) + 1}. \quad (15)$$

In accordance with the expression of the environment factor $\eta(x_i)$ corresponding to every food source, the selection probability of onlooker bees associated with the food source can be substituted with the following equation:

$$P_i = \frac{\eta_t(x_i)}{\sum_{m=1}^{\text{SN}} \eta_t(x_m)}. \quad (16)$$

4.3. Enhancing Convergence Efficiency with Artificial Immune Network Operators. In the basic ABC system, artificial bees fly around in the search space. Some (like employed and onlooker bees) choose food source depending on the experience of themselves and their nest mates and then adjust their positions, but others (like scouts) fly and choose the food sources randomly without using experience. If the nectar

amount of a new source is higher than that of the previous one in their memory, they memorize the new food source position and forget the previous one. Thus, the ABC system combines local search methods, carried out by employed and onlooker bees, with global search methods, managed by Karaboga and Basturk [15]. However, unlike the ABC system, the concept of artificial immune system (AIS) was originated by observing how the defense mechanism of natural immune system protects against attacks by antigens. There are numerous AIS algorithms developed for a variety of applications, where artificial immune network (aiNet for short) is a typical one and its algorithms and models are originally proposed to perform information compression and data clustering based on artificial immune system (AIS) theory [31]. Immune network-based algorithms are similar to clonal selection algorithms in that they both measure the goodness of antibodies by affinities, and both methods include a series of steps for selecting, cloning, and mutating antibodies. The major difference is that the immune network-based algorithms are represented by network graph structures [32]. Compared with other ones, the immune network-based algorithms employ extra procedures of antibody pruning and suppressing. This allows the models to generate a smaller, less-redundant population of antibody representatives, which is desirable for solving multimodal function optimization. Comparing ABC optimization with ai-Net algorithm, we can see that the advantages of ABC optimization lie in its neighborhood search method according to the profitability of food sources. However, ai-Net algorithm adopts fixed clonal individuals to perform local search which has certain blindness. In addition, due to introducing network compression, negative selection, and other operators, ai-Net can maintain the diversity of the population and reduce the possibility of being trapped into a local minimum. Unlike the ai-Net algorithm, ABC optimization maintains population diversity through random search of scout bees, which has obvious limitation. Based on the analysis mentioned above, if network compression and negative selection deriving from ai-Net algorithm are introduced into ABC optimization, this improved one may have a powerful and efficient multimodal searching ability as well as good stabilization. The detailed process is as follows.

Different employed bee individual corresponds to different food source position. In order to eliminate redundant and similar food sources, negative selection and network compression are used to compare with the similarities among various individuals. The Euclidean distance of two employed bee individuals X_i and X_k is adopted as shown in (17):

$$d(X_i, X_k) = \sqrt{\sum_{j=1}^D (X_i^j - X_k^j)^2} \quad (i \neq k). \quad (17)$$

In order to simplify the problem, the affinity concept is introduced which is obtained by the following equation using the normalization method:

$$A(X_i, X_k) = \frac{1}{1 + d(X_i, X_k)}, \quad (18)$$

- (1) Generate the initial population x_i based on chaotic maps and affinity strategy ($i = 1, 2, \dots, SN$)
- (2) Evaluate the fitness ($fit(x_i)$) of the population
- (3) Set cycle to 1
- (4) Repeat
- (5) For each employed bee {
 - Produce new solution v_i by using (7)
 - Calculate its fitness value $fit(v_i)$
 - Apply greedy selection process}
- (6) Adopt negative selection and network compression to eliminate redundant and similar food sources by using (18)
- (7) Randomly generate the same number of new individuals
- (8) Calculate the probability values P_i for the solution (x_i) by (16)
- (9) For each onlooker bee {
 - Select a solution x_i depending on P_i
 - Produce new solution v_j
 - Calculate its fitness value $fit(v_j)$
 - Apply greedy selection process}
- (10) **If** there is an abandoned solution for the scout,
then replace it with a new solution which will be randomly produced by (4)
- (11) Memorize the best solution so far
- (12) Cycle = cycle + 1
- (13) **Until** cycle = MEN

ALGORITHM 3: Pseudocode of main body of the enhanced ABC algorithm.

where the value range of $A(X_i X_k)$ is between 0 and 1. The smaller the value of $A(X_i X_k)$ is, the larger the value of $A(X_i X_k)$ is, which means that two different employed bee individuals have a higher similarity. Specially, when $A(X_i X_k)$ equals 1, these two ones are identical. According to negative selection and network compression operators, redundant and similar food sources should be eliminated. We predefine a threshold value, ϵ , so as to realize wipe-off of redundant individuals. It also means when $A(X_i X_k)$ is equal to or great than ϵ , we think these two ones are identical and only one can be retained and other should be wiped off. Repeat this process until the affinity of any two individuals in a population is less than ϵ . In doing so, the population size may be reduced. Nevertheless, in order to keep the population size unchanged, the same number of new individuals need generating randomly. Through negative selection and network compression operators, the exploitation efficiency will be improved and the corresponding convergence rate of the algorithm will also be accelerated.

4.4. Main Steps of the Enhanced Artificial Bee Colony Algorithm. Based on the above analysis, three main improvements including novel generation of initial population, self-adaptive searching strategy, and redundant individual compression operator are presented and the detailed pseudo-code is given in Algorithm 3.

5. Experimental Studies on Function Optimization Problems

5.1. Benchmark Functions and Parameter Settings. In this section, numerical experiment is used to test the performance of the enhanced ABC (shorthand for EABC) proposed in this

paper. Summarized in Table 1 are the 15 scalable benchmark functions. $f_1 \sim f_{10}$ are continuous unimodal functions. $f_{11} \sim f_{15}$ are multimodal functions and the number of their local minima increases exponentially with the problem dimension.

In order to testify the performance of different intelligent algorithms, we compare the EABC with the standard ACO, PSO, and ABC. In all simulations, the population size of ACO, PSO, ABC, and EABC is 50. The maximum number of function evaluations (FE) is set to 5000. The threshold value of the affinity, ϵ , is 0.9. Other related parameter values of ACO, PSO, and ABC are referred in the literature [17]. All experiment results reported are obtained based on 30 independent runs. The experiment results are the best, worst, mean, and standard deviation of the statistical experimental data.

5.2. Simulation Results. The performance on the solution accuracy of EABC is compared with that of ACO, PSO, and ABC. Table 2 shows the optimization of the 15 benchmark functions obtained in the 30 independent runs by each algorithm and some interesting results can be found in Table 2.

Firstly, almost all algorithms have identical performance on most of unimodal functions f_1 to f_4 , f_7 , f_8 , and f_{14} . However, on other functions, these four algorithms show different performance, especially for multimodal ones such as f_{11} , f_{12} , f_{13} , and f_{15} . For example, on function f_{15} , the best values obtained by ACO, PSO, ABC, and EABC are -10296 , -6993.47 , -12566.9 , and -152568.7 , respectively. It means that EABC can be efficiently applied for solving multimodal and multidimensional function optimization problems due to its abundant operators such as clonal

TABLE 1: Benchmark functions used in experiments.

Number	Function	Dimension	Property	Range	Min
1	$f_1(x) = \sum_{i=1}^D x_i^2$	30	Unimodal	[-100, 100]	0
2	$f_2(x) = \sum_{i=1}^D ix_i^2$	30	Unimodal	[-10, 10]	0
3	$f_3(x) = 25 + \sum_{i=1}^D [x_i]$	5	Unimodal	[-5.12, 5.12]	0
4	$f_4(x) = \sum_{i=1}^D ([x_i + 0.5])^2$	30	Unimodal	[-100, 100]	0
5	$f_5(x) = \sum_{i=1}^D ix_i^4 + \text{random}(0, 1)$	30	Unimodal	[-1.28, 1.28]	0
6	$f_6(x) = \max_i \{\ x_i\ , 1 \leq i \leq D\}$	30	Unimodal	[-100, 100]	0
7	$f_7(x) = -\prod_{i=1}^D \cos(x_i) \exp\left(-\sum_{i=1}^D (x_i - \pi)^2\right)$	2	Unimodal	[-100, 100]	-1
8	$f_8(x) = 0.26 \sum_{i=1}^D x_i^2 - 0.48 \prod_{i=1}^D x_i$	2	Unimodal	[-10, 10]	0
9	$f_9(x) = (x_1 - 1)^2 + \sum_{i=2}^D i(2x_i^2 - x_{i-1})^2$	30	Unimodal	[-10, 10]	0
10	$f_{10}(x) = \sum_{i=1}^D 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$	30	Unimodal	[-30, 30]	0
11	$f_{11}(x) = \sum_{i=1}^D (x_i - 10 \cos(2\pi x_i) + 10)$	30	Multimodal	[-5.12, 5.12]	0
12	$f_{12}(x) = \frac{1}{4000} \left(\sum_{i=1}^D (x_i - 100)^2 \right) - \left(\prod_{i=1}^D \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) \right) + 1$	30	Multimodal	[-600, 600]	0
13	$f_{13}(x) = 10 \times D + \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i))$	10	Multimodal	[-50, 50]	0
14	$f_{14}(x) = 0.5 + \frac{\sin^2\left(\sqrt{\sum_{i=1}^D x_i^2}\right) - 0.5}{\left(1 + 0.001\left(\sum_{i=1}^D x_i^2\right)\right)^2}$	2	Multimodal	[-100, 100]	0
15	$f_{15}(x) = \sum_{i=1}^D -x_i \sin\left(\sqrt{\ x_i\ }\right)$	30	Multimodal	[-500, 500]	-12569.5

selection and negative selection which outperforms ACO, PSO, and ABC. In addition, on the one hand, on basic unimodal functions, both the basic ABC and EABC have almost identical solving performance; on the other hand, on the multimodal functions, these two ones display huge difference.

Secondly, the EABC algorithm can find optimal or closer-to-optimal solutions on the complex multimodal functions f_{11} , f_{12} , f_{13} , and f_{14} . Although the result of multi-dimension function f_{15} is slightly far from the known global optimum, the EABC is superior to the other algorithms all the same. At the same time, for almost all benchmark functions, standard deviations of the EABC obtained from the statistical experimental data are no greater than those of others expect for f_{10} . In addition, the differences of EABC between the best and worst solutions for these 15 benchmark functions are

relatively smaller than those of others in the 30 independent simulation runs. All these mean that the EABC algorithm has better robustness than others. It is also clear that EABC can work better in almost all cases and gets better performance than ACO, PSO, and ABC.

Summarizing the statements mentioned above, the EABC can prevent bees from being trapped into the local minimum, accelerate convergence process, search with more efficiency, and improve exploitation abilities for basic ABC.

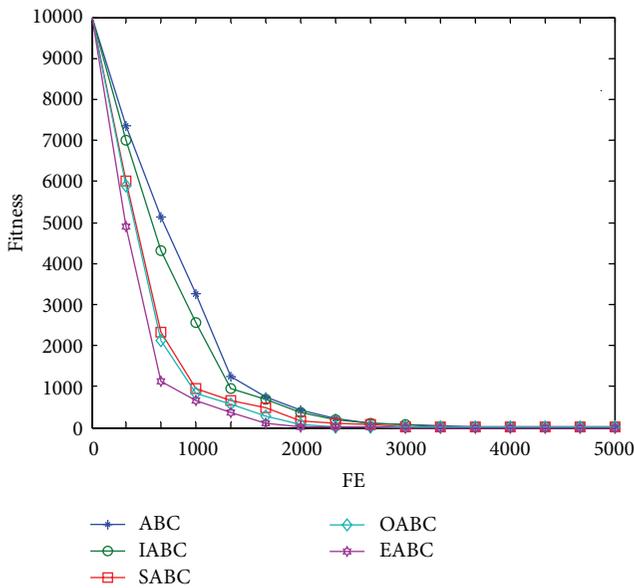
5.3. Analysis and Discussion. In this section, the effects of each modification on the performance of EABC are discussed. First of all, corresponding to three modifications, we named the basic ABC with the proposed initialization as IABC, the one with the proposed self-adaptive searching strategy as SABC, and the one with the proposed immune

TABLE 2: Benchmark functions used in experiments for testing the performances of EABC, ACO, PSO, and ABC.

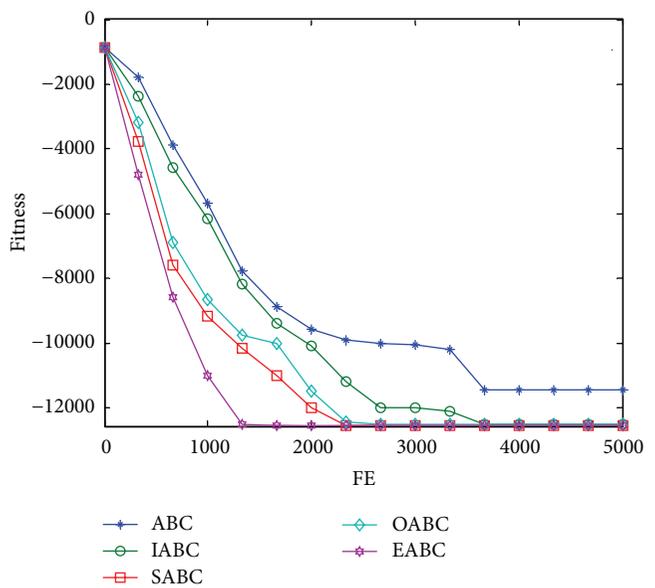
Function number	Min		ACO	PSO	ABC	EABC
$f_1(x) = \sum_{i=1}^D x_i^2$	0	Best	0	0	0	0
		Worst	0	0	0	0
		Mean	0	0	0	0
		SD	0	0	0	0
$f_2(x) = \sum_{i=1}^D ix_i^2$	0	Best	0	0	0	0
		Worst	0	0	0	0
		Mean	0	0	0	0
		SD	0	0	0	0
$f_3(x) = 25 + \sum_{i=1}^D [x_i]$	0	Best	0	0	0	0
		Worst	0	0	0	0
		Mean	0	0	0	0
		SD	0	0	0	0
$f_4(x) = \sum_{i=1}^D ([x_i + 0.5])^2$	0	Best	0	0	0	0
		Worst	0	0	0	0
		Mean	0	0	0	0
		SD	0	0	0	0
$f_5(x) = \sum_{i=1}^D ix_i^4 + \text{random}(0, 1)$	0	Best	0	0	0	0
		Worst	0.00289	0.00321	0.00543	0.00422
		Mean	0.00136	0.00116	0.00300	0.00196
		SD	0.00219	0.00276	0.00387	0.00208
$f_6(x) = \max_i \{\ x_i\ , 1 \leq i \leq D\}$	0	Best	0	0	0	0
		Worst	0.00246	0.00305	0.00110	0.00400
		Mean	0.00180	0.00156	0.00666	0.00210
		SD	0.00039	0.00058	0.00092	0.00037
$f_7(x) = -\prod_{i=1}^D \cos(x_i) \exp\left(-\sum_{i=1}^D (x_i - \pi)^2\right)$	-1	Best	-1	-1	-1	-1
		Worst	-1	-1	-1	-1
		Mean	-1	-1	-1	-1
		SD	0	0	0	0
$f_8(x) = 0.26 \sum_{i=1}^D x_i^2 - 0.48 \prod_{i=1}^D x_i$	0	Best	0	0	0	0
		Worst	0	0	0	0
		Mean	0	0	0	0
		SD	0	0	0	0
$f_9(x) = (x_1 - 1)^2 + \sum_{i=2}^D i(2x_i^2 - x_{i-1})^2$	0	Best	0.66667	0.6667	0	0
		Worst	0.66667	0.6667	0	0
		Mean	0.66667	0.6667	0	0
		SD	0.00001	0.00001	0	0
$f_{10}(x) = \sum_{i=1}^D 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$	0	Best	8.7513	10.5433	19.6788	9.1578
		Worst	32.4215	24.6711	54.2333	26.9874
		Mean	18.2039	15.0886	33.1227	17.3558
		SD	5.0361	24.1702	154.1443	11.4774
$f_{11}(x) = \sum_{i=1}^D (x_i - 10 \cos(2\pi x_i) + 10)$	0	Best	52.6677	43.5774	0	0
		Worst	53.2331	44.1131	0	0
		Mean	52.9226	43.9771	0	0
		SD	4.5649	11.7286	0	0
$f_{12}(x) = \frac{1}{4000} \left(\sum_{i=1}^D (x_i - 100)^2 \right) - \left(\prod_{i=1}^D \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) \right) + 1$	0	Best	0.01470	0.017112	0.008531	0
		Worst	0.01499	0.017989	0.017356	0
		Mean	0.01479	0.017391	0.011447	0
		SD	0.00296	0.020808	0.001223	0

TABLE 2: Continued.

Function number	Min		ACO	PSO	ABC	EABC
$f_{13}(x) = 10 \times D + \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i))$	0	Best	4.56781	21.44755	3.34788	0
		Worst	8.77993	39.55741	10.91447	0
		Mean	5.85411	26.3991	5.59331	0
		SD	13.1142	155.6380	10.04216	0
$f_{14}(x) = 0.5 + \frac{\sin^2\left(\sqrt{\sum_{i=1}^D x_i^2}\right) - 0.5}{\left(1 + 0.001\left(\sum_{i=1}^D x_i^2\right)\right)^2}$	0	Best	0	0	0	0
		Worst	0	0	0	0
		Mean	0	0	0	0
		SD	0	0	0	0
$f_{15}(x) = \sum_{i=1}^D -x_i \sin\left(\sqrt{\ x_i\ }\right)$	-12569.5	Best	-10296	-6993.47	-11566.9	-12568.7
		Worst	-10237	-6883.33	-11498.8	-12514.3
		Mean	-10266	-6909.12	-11544.1	-12551.1
		SD	521.849	457.9577	125.4471	101.3217



(a) Function f_{13} with $D = 10$



(b) Function f_{15} with $D = 30$

FIGURE 1: Convergence speed of the different ABCs on the two test functions (f_{13} , f_{15}).

operators as OABC. We compare the convergence speed of these different ABCs through two complex high-dimension multimodal functions f_{13} and f_{15} in order to find the contributions of three modifications in EABC to improve the performance of the algorithm, respectively. The corresponding simulation results are shown in Figure 1. We can see from Figure 1 that IABC, SABC and OABC outperform the basic ABC, which means that the three modification measures mentioned in Section 4 have positive effect on the convergence speed of the algorithm. In addition, SABC, and OABC are obviously superior to IABC, which implies that searching strategy and immune operators play more important roles than that of initialization. However, it is difficult to compare the contributions between searching

strategy and immune operators on the two test functions; the reasons may be that the characteristic of test functions will also affect the problem-solving efficiency of the algorithm.

6. Conclusion

In this paper, we have proposed an enhanced artificial bee colony algorithm, called EABC through introducing self-adaptive searching strategy and artificial immune network operators. Subsequently, a suite of unimodal or multimodal benchmark functions are used to testify the performance of the proposed algorithm. The simulation results illustrate that the EABC algorithm outperforms ACO, PSO, and the basic ABC.

The future work includes the studies on how to apply EABC to more complex discrete dynamic optimization problems including product design optimization problem, dynamic project scheduling problem, and data clustering problem.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research is supported by National Natural Science Foundation of China (Grant nos. 71071141 and 71171089), the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant nos. 20130142110051 and 20103326120001), and Humanity and Sociology Foundation of Ministry of Education of China (Grant no. I1YJC630019).

References

- [1] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York, NY, USA.
- [2] X. S. Yang, Z. H. Cui, R. B. Xiao, A. H. Gandomi, and M. Karamanoglu, *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, Elsevier, Waltham, Mass, USA, 2013.
- [3] S. T. Hsieh, T. Y. Sun, C. L. Lin, and C. C. Liu, "Effective learning rate adjustment of blind source separation based on an improved particle swarm optimizer," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 242–251, 2008.
- [4] Z. H. Cui and X. J. Cai, "Integral particle swarm optimization with dispersed accelerator information," *Fundamenta Informaticae*, vol. 95, no. 4, pp. 427–447, 2009.
- [5] Z. H. Cui, X. J. Cai, J. C. Zeng, and Y. F. Yin, "PID-controlled particle swarm optimization," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 16, no. 6, pp. 585–610, 2010.
- [6] C. Priya and P. Lakshmi, "Particle swarm optimisation applied to real time control of spherical tank system," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 206–216, 2012.
- [7] A. H. Gandomi, G. J. Yun, X. S. Yang, and S. Talatahari, "Chaos-enhanced accelerated particle swarm optimization," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 2, pp. 327–340, 2013.
- [8] K. M. Salama and A. A. Freitas, "Learning Bayesian network classifiers using ant colony optimization," *Swarm Intelligence*, vol. 7, no. 2-3, pp. 229–254, 2013.
- [9] H. Ahangarikiasari, M. R. Saraji, and M. Torabi, "Investigation of code complexity of an innovative algorithm based on ACO in weighted graph traversing and compare it to traditional ACO and Bellman-Ford," *Journal of Bioinformatics and Intelligent Control*, vol. 2, no. 1, pp. 73–78, 2013.
- [10] P. B. Cao and R. B. Xiao, "Assembly planning using a novel immune approach," *International Journal of Advanced Manufacturing Technology*, vol. 31, no. 7-8, pp. 770–782, 2007.
- [11] M. Bateni, A. Baraani, and A. Ghorbani, "Alert correlation using artificial immune recognition system," *International Journal of Bio-Inspired Computation*, vol. 4, no. 3, pp. 181–195, 2012.
- [12] D. Karaboga, "An idea on honeybee swarm for numerical optimization," Tech. Rep. TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
- [13] T. Chen and C. Ju, "A novel artificial bee colony algorithm for solving the supply chain network design under disruption scenarios," *International Journal of Computer Applications in Technology*, vol. 47, no. 2-3, pp. 289–296, 2013.
- [14] T. Chen and R. Xiao, "A dynamic intelligent decision approach to dependency modeling of project tasks in complex engineering system optimization," *Mathematical Problems in Engineering*, vol. 2013, Article ID 398123, 12 pages, 2013.
- [15] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.
- [16] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 687–697, 2008.
- [17] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.
- [18] W. Gao and S. Liu, "Improved artificial bee colony algorithm for global optimization," *Information Processing Letters*, vol. 111, no. 17, pp. 871–882, 2011.
- [19] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.
- [20] W. Gao and S. Liu, "A modified artificial bee colony algorithm," *Computers and Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.
- [21] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3166–3173, 2010.
- [22] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul, "The best-so-far selection in artificial bee colony algorithm," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 2888–2901, 2011.
- [23] F. Kang, J. Li, and Q. Xu, "Structural inverse analysis by hybrid simplex artificial bee colony algorithms," *Computers and Structures*, vol. 87, no. 13-14, pp. 861–870, 2009.
- [24] A. Singh, "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem," *Applied Soft Computing Journal*, vol. 9, no. 2, pp. 625–631, 2009.
- [25] C. Zhang, D. Ouyang, and J. Ning, "An artificial bee colony approach for clustering," *Expert Systems with Applications*, vol. 37, no. 7, pp. 4761–4767, 2010.
- [26] Q. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Information Sciences*, vol. 181, no. 12, pp. 2455–2468, 2011.
- [27] S. Samanta and S. Chakraborty, "Parametric optimization of some non-traditional machining processes using artificial bee colony algorithm," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 6, pp. 946–957, 2011.
- [28] A. Alejandro, L. G. Jorge, I. R. Manuel, and M. Aide, "Optimization of the material flow in a manufacturing plant by use of artificial bee colony algorithm," *Expert Systems With Applications*, vol. 40, no. 12, pp. 4785–4790, 2013.

- [29] S. Sundar, A. Singh, and A. Rossi, "An artificial bee colony algorithm for the 0-1 multidimensional knapsack problem," in *Proceedings of the 3rd International Conference on Contemporary Computing*, vol. 94 of *Communications in Computer and Information Science*, pp. 141–151, 2010.
- [30] S. Sundar and A. Singh, "A hybrid heuristic for the set covering problem," *Operational Research*, vol. 12, no. 3, pp. 345–365, 2012.
- [31] Y. Liu and R. Xiao, "Optimal synthesis of mechanisms for path generation using refined numerical representation based model and AIS based searching method," *Journal of Mechanical Design*, vol. 127, no. 4, pp. 688–691, 2005.
- [32] B. Gong, J. Im, and G. Mountrakis, "An artificial immune network approach to multi-sensor land use/land cover classification," *Remote Sensing of Environment*, vol. 115, no. 2, pp. 600–614, 2011.

Research Article

Operational Optimization of Large-Scale Parallel-Unit SWRO Desalination Plant Using Differential Evolution Algorithm

Jian Wang,^{1,2} Xiaolong Wang,² Aipeng Jiang,² Shu Jiangzhou,² and Ping Li¹

¹ School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China

² Institute of Energy Utilization & Automation, Hangzhou Dianzi University, Hangzhou 310018, China

Correspondence should be addressed to Ping Li; pli@iipc.zju.edu.cn

Received 15 November 2013; Accepted 30 December 2013; Published 17 February 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 Jian Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A large-scale parallel-unit seawater reverse osmosis desalination plant contains many reverse osmosis (RO) units. If the operating conditions change, these RO units will not work at the optimal design points which are computed before the plant is built. The operational optimization problem (OOP) of the plant is to find out a scheduling of operation to minimize the total running cost when the change happens. In this paper, the OOP is modelled as a mixed-integer nonlinear programming problem. A two-stage differential evolution algorithm is proposed to solve this OOP. Experimental results show that the proposed method is satisfactory in solution quality.

1. Introduction

The shortage of fresh water has become a bottleneck of the economic development in many countries. Seawater desalination is an effective way to solve this problem. Reverse osmosis (RO) desalinating is one of the most popular ways to generate freshwater from seawater and has made a rapid progress over the past four decades [1–3]. The scale of seawater reverse osmosis (SWRO) desalination plant is continually expanding, whose capacity of freshwater has exceeded 100,000 tons per day in recent years.

Now, a kind of large-scale parallel-unit SWRO desalination plant, which is composed of multiple parallel RO units, has appeared. This kind of plant has huge capital cost and more complicated operation processes. So, before it is built, an optimal design is made to select the suitable devices and system performance to match the operating condition [4–8]. These optimal designs are made based on static condition [9, 10], but the actual situation is changing [11]. For example, the seawater temperature varies with changing seasons, the freshwater supply changes according to the user's demand, the permeate rate is declining with the membrane fouling [12], and so on. The result is that these devices would not work at the optimal designed points in most time. Therefore,

an operational optimization scheduling is necessary to make these machines work in a best way under the changed conditions.

In this paper, a mathematical model of operational optimization problem (OOP) for a large-scale parallel-unit SWRO desalination plant, which includes objective function and constraint functions, is made. In order to solve this OOP, a two-stage differential evolution (TSDE) algorithm is proposed. The simulating results show that the TSDE is excellent in searching ability than basic DE and genetic algorithm (GA).

2. Problem Description and Formulation

2.1. SWRO Desalination Plant Representation. The single SWRO desalination unit is a multistage process, which includes seawater intake, pretreatment, RO desalination, and posttreatment [13]. Figure 1 shows the schematic diagram of the unit.

In the intake stage, the raw seawater is pumped from a deep well, which is located close to the shoreline, to the flocculators to filter; in the pretreatment stage, the most suspended matters and colloids are filtered out of seawater

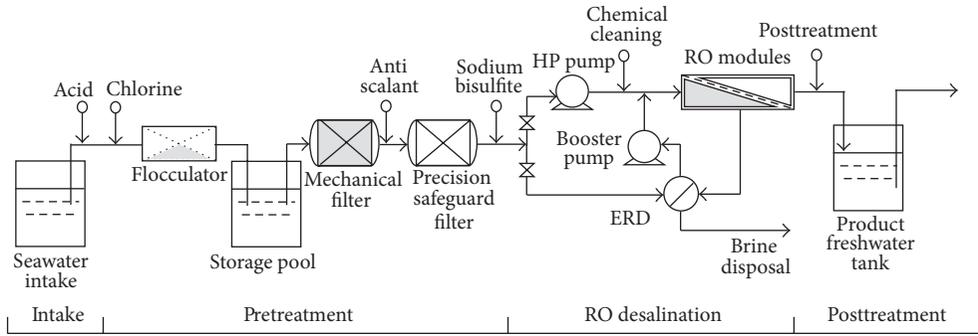


FIGURE 1: Schematic diagram of a single SWRO desalination unit.

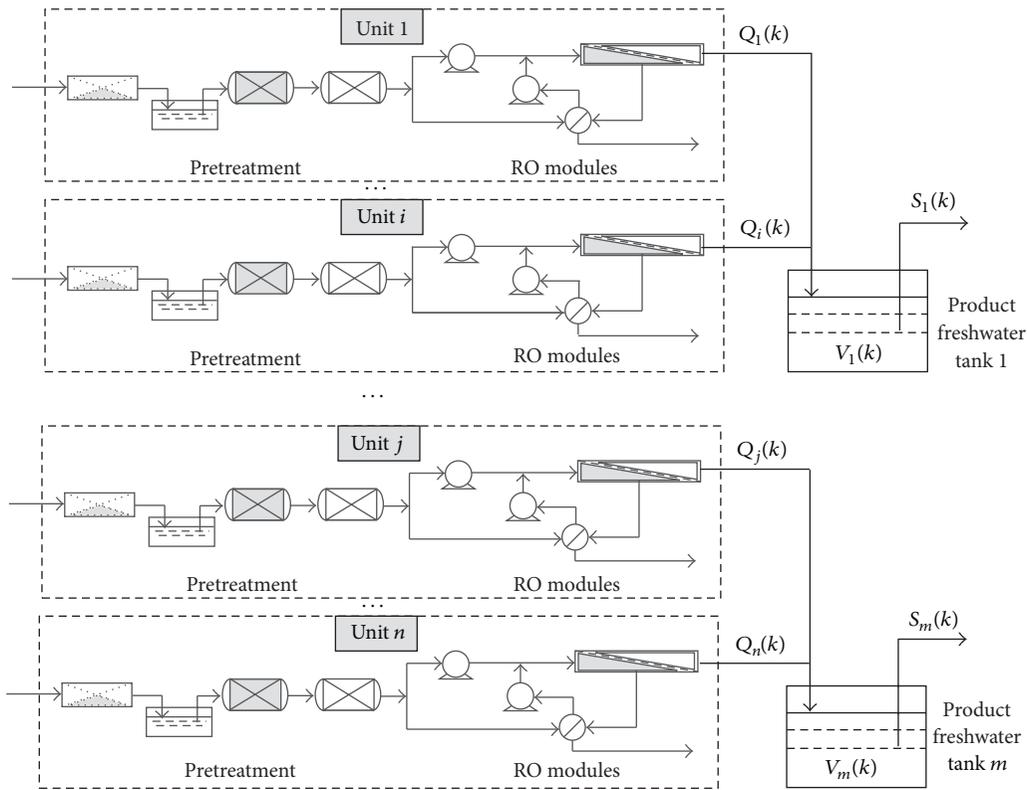


FIGURE 2: Schematic diagram of a parallel-unit SWRO desalination plant.

by flocculators, mechanical filters, and precision safeguard filters successively to ensure that SDI (silt density index) is lower than 5 to meet the RO modules requirement; in the RO desalination stage, one part of the fed seawater is pressurized by the high pressure (HP) pump and the other by energy recovery device (ERD) and booster pump up to 40 ~ 50 bar and then is desalinated by RO modules; in the posttreatment stage, the produced freshwater flows into the product freshwater tanks (PFWT) to supply into the municipal water network; the brine is disposed reasonably.

The schematic diagram of a large-scale parallel-unit SWRO desalination plant is shown in Figure 2, which is structured with a number of independent RO units in parallel.

Each RO unit has different permeate rate. The freshwater, produced by RO Unit i ($i \in n_j$), flows into PFWT j and then is supplied for user.

2.2. Problem Mathematical Formulation. The OOP of a large-scale SWRO desalination plant is to make an optimal scheduling plan to minimize the plant's total running cost (TRC). This optimal schedule plan determines the on/off status and amount of generated freshwater of each RO unit, and the amount of supplied freshwater by each PFWT at each time period. When computing the TRC, the price factors, such as time-of-use electricity price, the operation cost, and the maintenance cost, are taken into consideration.

2.2.1. *Objective Function of OOP.* A lumped parameter model of this problem is built in this paper. The TRC of the plant consists of capital depreciation cost (CDC), operating cost (OC), labor and chemical cost (LCC), and energy cost (EC), which is presented as

$$\min \text{TRC} = \text{CDC} + \text{OC} + \text{LCC} + \text{EC}. \quad (1)$$

(1) *The Capital Depreciation Cost, CDC, is*

$$\text{CDC} = \text{CC} \times 1.411 \times \eta \times T. \quad (2)$$

The CDC is presented as (2), where CC is the capital cost of the SWRO desalination plant, 1.411 is the coefficient that is used to calculate the practical investment, η is the capital charge rate, which is usually a constant, and T is the operational optimization periods. When the SWRO desalination plant has been built, the CC is a fixed value; therefore, the CDC is a fixed cost in a certain time period T under a constant capital charge rate η . So, when computing the minimum value of TRC in this condition, this part cost CDC can be ignored. When the other parts of TRC reach their minimum values, the TRC gets its optimal value by adding this constant cost.

(2) *The Operating Cost, OC.* The OC includes maintenance expense and the repair and replacement expense of devices. The maintenance expense refers to the conventional maintenance expense during machines running; the repair and replacement expense refers to the cost of repair and replacement when the equipment stops. The OC is presented as

$$\text{OC} = \sum_{k=1}^K \sum_{i=1}^n M_i(k), \quad (3)$$

where K is the time horizon of the operational optimization; n is the number of RO units in the plant; $M_i(k)$ is the operating cost of Unit i at the time k , which is presented as

$$M_i(k) = \begin{cases} C_1 \times Q_i(k), & \alpha_i(k) = 1, \\ C_2, & \alpha_i(k) = 0, \end{cases} \quad (4)$$

where C_1 is the coefficient that is used to calculate the maintenance cost; $Q_i(k)$ is the amount of freshwater generated by Unit i at time k ; C_2 is the repair and replacement expense when Unit i stops at time k ; $\alpha_i(k)$ is the on/off status of Unit i at time k ; $\alpha_i(k) = 1$ when Unit i is running; $\alpha_i(k) = 0$ when it stops.

(3) *The Labor and Chemical Cost, LCC.* Usually, we consider $\text{LCC} = 12\% \times \text{TRC}$. This cost includes labor cost, chemical expense, and other expenses. When TRC is the smallest, LCC has its minimum value.

(4) *The Energy Cost, EC.* The energy cost is the energy consumption which is necessary to drive the devices. It is

obviously variable in different scheduling plans. By optimizing the scheduling plan, the plant's energy consumption can achieve the minimum value. The EC is presented as

$$\text{EC} = \sum_{k=1}^K \left(P_e(k) \sum_{i=1}^n (\alpha_i(k) \times C_3 \times Q_i(k)) \right), \quad (5)$$

where C_3 is the coefficient between energy and amount of freshwater generated by Unit i at time k ; $P_e(k)$ is time-of-use electricity price at time k .

2.2.2. *Constraints of OOP.* The constraints of OPP include technical limitations and the design requirements, which are as follows.

(1) *The Amount of Freshwater, $Q_i(k)$.* $Q_i(k)$ is the amount of freshwater generated by Unit i at time k , which is subject to

$$Q_{i,\min} \leq Q_i(k) \leq Q_{i,\max}, \quad i = 1, 2, \dots, n, \quad \forall k, \quad (6)$$

where $Q_{i,\max}$, $Q_{i,\min}$ are the upper and the lower limit of the amount of freshwater generated by Unit i , respectively.

(2) *The Capacity of Each PFWT, $V_j(k)$.* $V_j(k)$ is the capacity of PFWT j at time k , which is subject to

$$V_{j,\min} \leq V_j(k-1) + \sum_{i=1}^{n_j} \alpha_i(k) \cdot Q_i(k) - S_j(k) \leq V_{j,\max}, \quad (7)$$

$$j = 1, 2, \dots, m; \quad i = 1, 2, \dots, n_j, \quad \forall k,$$

where $V_j(k-1)$ is the capacity of PFWT j at the beginning of time k ; m is the number of PFWT; n_j is the number of RO units which feed freshwater to the PFWT j ; $S_j(k)$ is the amount of supplied freshwater by PFWT j at time k ; $V_{j,\max}$, $V_{j,\min}$ are the upper limit and the lower limit of capacity of PFWT j .

(3) *The Amount of Supplied Freshwater, $S_j(k)$.* $S_j(k)$ is the amount of supplied freshwater by PFWT j at time k , which is subject to

$$\sum_{j=1}^m S_j(k) = D(k), \quad \forall k, \quad (8)$$

$$S_j(k) \geq 0, \quad j = 1, 2, \dots, m, \quad \forall k.$$

3. Two-Stage Differential Evolution

The OOP of the large-scale parallel-unit SWRO desalination plant is a mixed-integer nonlinear programming problem (MINLP) over the time horizon. In the OOP, $\{0, 1\}$ binary variables represent the on/off statuses of RO units; the real variables represent the amounts of freshwater generated by RO units and supplied freshwater by PFWTs. In order to compute this OOP, a novel differential evolution, TSDE, is proposed in this paper, and the basic steps of this TSDE are addressed in the following subsections.

3.1. Basic DE. Differential evolution algorithm was originally proposed by Storn and Price for solving continuous optimization in the mid-1990s [14–18]. It is an evolutionary algorithm, including three important operators: mutation, crossover, and selection [19, 20]. The basic DE works are as follows.

Step 1 (initialization). The original population is generated in the search space randomly, which contains NP individual vectors: $\mathbf{x}_{i,1} = [x_{i1,1}, x_{i2,1}, \dots, x_{iD,1}]$, $i = 1, 2, \dots, \text{NP}$, where D is the dimension of individual.

Step 2 (mutation). For each individual vector $\mathbf{x}_{i,G}$, a mutated solution $\mathbf{v}_{i,G}$ is created according to a DE/rand/1/bin mutate operator (9) in each generation G [15]. Consider

$$\mathbf{v}_{i,G} = \mathbf{x}_{r_1,G} + F \times (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}), \quad (9)$$

where G is the current generation number, F is a scale factor, and r_1, r_2 , and r_3 are randomly selected integers from the set $\{1, 2, \dots, \text{NP}\}$ ($r_1 \neq r_2 \neq r_3 \neq i$).

Step 3 (crossover). The crossover operator generates an offspring $\mathbf{u}_{i,G}$ according to

$$u_{ij,G} = \begin{cases} v_{ij,G}, & \text{if } (\text{rand}(j) \leq \text{CR}) \text{ or } j = \text{rnbr}(j), \\ x_{ij,G}, & \text{otherwise} \end{cases} \quad (10)$$

$(i = 1, 2, \dots, \text{NP}; j = 1, 2, \dots, D),$

where CR is a crossover rate; $\text{rnbr}(j)$ is a randomly chosen index ($\text{rnbr}(j) \in \{1, 2, \dots, D\}$), which ensures $\mathbf{u}_{i,G}$ getting at least one component from $\mathbf{v}_{i,G}$.

Step 4 (selection). The selection operator is to generate next population $\mathbf{x}_{i,G+1}$ according to (11). The objective function of $\mathbf{x}_{i,G}$ is compared to one of $\mathbf{u}_{i,G}$ and the smaller one is selected as the next generation. Consider

$$\mathbf{x}_{i,G+1} = \begin{cases} \mathbf{u}_{i,G}, & \text{if } (f(\mathbf{u}_{i,G}) \leq f(\mathbf{x}_{i,G})), \\ \mathbf{x}_{i,G}, & \text{otherwise} \end{cases} \quad (11)$$

$(i = 1, 2, \dots, \text{NP}).$

Step 5 (stopping criterion). If the stopping criterion (maximum number of iterations) is satisfied, computation is terminated; otherwise, Steps 3–5 are repeated.

3.2. Treatment of Constraints. In the OOP, there are two types of constraints: boundary constraints and technical limited constraint functions. The following is the treatment strategies of them.

3.2.1. Boundary Constraints. Sometimes $\mathbf{u}_{i,G}$ is out of the range of search space. When it happens, it is necessary to

replace this value to guarantee it is into its allowed range (12). Consider

$$u_{ij,G} = \begin{cases} x_{ij}^L + \text{rand} \cdot (x_{ij}^U - x_{ij}^L) & \text{if } (u_{ij,G} < x_{ij}^L), \text{ or } (u_{ij,G} > x_{ij}^U) \\ u_{ij,G}, & \text{otherwise} \end{cases} \quad (12)$$

$i = 1, 2, \dots, \text{NP}; j = 1, 2, \dots, D,$

where x_{ij}^L is the lower bound of $u_{ij,G}$; x_{ij}^U is the upper bound of $u_{ij,G}$.

3.2.2. Constraint Functions. Penalty function is one of the most effective methods to solve the evolutionary constraint optimization problem [21–25].

In this paper, the MINLP with constraints (13) is converted into an unconstrained MINLP by using a penalty function shown as (14). Consider

$$\begin{aligned} \min \quad & f(x, y) \\ & g_k(x, y) \leq 0, \quad k = 1, 2, \dots, p \\ & h_l(x, y) = 0, \quad l = 1, 2, \dots, q \\ \text{s.t.} \quad & x_i^L \leq x_i \leq x_i^U, \quad i = 1, 2, \dots, D \\ & y_k \in \{0, 1\}, \quad k = 1, 2, \dots, m, \end{aligned} \quad (13)$$

$$\min F(x, y) = f(x, y) + M$$

$$\times \left[\sum_{k=1}^p \max\{0, g_k(x, y)\}^2 + \sum_{l=1}^q \max\{0, |h_l(x, y) - \varepsilon|\}^2 \right], \quad (14)$$

where $x = [x_1, x_2, \dots, x_n]^T$ is a continuous vector; $y = [y_1, y_2, \dots, y_m]^T$ is a binary vector; $g_k(x, y)$ represents the k th inequality constraint; $h_l(x, y)$ represents the l th equality constraint; $F(x, y)$ represents new objective function, which consists of two parts: the old objective function $f(x, y)$ and a penalty function; p is the number of inequality constraints; q is the number of equality constraints; ε is a small positive constant, so that the l th equality constraint $h_l(x, y) = 0$ is converted into the inequality constraint $h_l(x, y) - \varepsilon < 0$. In addition, M is defined as penalty coefficient, which is a large positive constant, so that it imposes penalty on unfeasible solutions.

3.3. Conventional Technique for Binary Variables. In this paper, the $\{0, 1\}$ binary variables represent on/off status of RO units, but the DE algorithm is only capable of handling continuous variables. Therefore, some real variables within the range of $[0, 1]$ are used to represent the statuses of RO units in TSDE instead of these binary variables. When the objective

function and the constraint functions are calculated, these real variables are rounded off to the nearest integer with (15). Consider

$$\bar{y}_i = \text{INT}(y_i), \quad (15)$$

where INT() is a function to convert a real number to an integer value by rounding off.

3.4. *Two-Stage DE.* The OOP of a large-scale parallel-unit SWRP desalination plant has such features: the permeate rate of each RO unit is huge; that is, the amount of freshwater generated by each RO unit is usually more than 10000 tons a day; the permeate rate changes within an allowable range, but this range is much smaller than its amount. So, when a RO unit's status changes, the TRC of the plant will change sharply. Usually, this change cannot be made up by adjusting the RO unit's permeate rate within its allowable range.

In this paper, a novel DE, TSDE, is proposed to solve the OOP. The TSDE is divided into two periods: Stage One and Stage Two. In Stage One, the permeate rate of each RO unit is supposed be a constant, which equals the median of its allowable range. A DE algorithm is used to compute the run/off status of each RO unit in this stage. When the DE is satisfying the stopping criterion, a preliminary scheduling will be worked out.

Then, the best 30 percent individuals in Stage One remain to Stage Two. The other 70 percent individuals in Stage Two are generated in the search space randomly (here, these two stages have the same population size NP). All these individuals are as the original values of another DE algorithm to evolve again in Stage Two. When the second DE stops, a final scheduling will be got.

4. Experimentation

For numerical experimentation, a large-scale parallel-unit SWRO desalination plant in Liuheng Island, China, which has the capacity of 100,000 m³ freshwater a day, is considered over a 24-hour time horizon. The basic parameters of this OOP are shown in Tables 1, 2, and 3.

Here, the length of individual $D = 20$; each individual consists of 20 variables. The first 8 variables are {0, 1} binary variables to represent the on/off statuses of 8 RO units. The next 8 variables are real values, which are the amounts of freshwater generated by 8 RO units. And the last 4 variables are real values to represent the supplied freshwater by 4 PFWTs.

The CDC of this plant is considered as a constant and ignored when calculating the TRC.

4.1. *Parameters of TSDE.* The DE's search ability for different problems depends on its parameters [26]. So, before it is working, these parameters must be tuned.

4.1.1. *The Maximum Number of Iterations G_m and the Population Size NP.* Firstly, the effects of two important parameters, the maximum number of iterations G_m and the population

TABLE 1: The basic parameters of OOP in Liuheng SWRO desalination plant.

Parameters	Sign	Value
The number of RO units	n	8
The number of PFWTs	m	4
	n_1	2
The number of RO units which feed freshwater to each PFWT	n_2	2
	n_3	2
	n_4	2
	$Q_{1,max}$	460
	$Q_{2,max}$	460
	$Q_{3,max}$	570
The upper limit of the permeate rate of RO Unit i , (m ³ /h)	$Q_{4,max}$	570
	$Q_{5,max}$	570
	$Q_{6,max}$	800
	$Q_{7,max}$	570
	$Q_{8,max}$	570
	$Q_{1,min}$	380
	$Q_{2,min}$	380
	$Q_{3,min}$	470
The lower limit of the permeate rate of RO Unit i , (m ³ /h)	$Q_{4,min}$	470
	$Q_{5,min}$	470
	$Q_{6,min}$	655
	$Q_{7,min}$	470
	$Q_{8,min}$	470
	$V_{1,0}$	340
The initial value of PFWT (m ³)	$V_{2,0}$	340
	$V_{3,0}$	340
	$V_{4,0}$	340
	$V_{1,max}$	1680
The maximum capacity of PFWT i , (m ³)	$V_{2,max}$	1680
	$V_{3,max}$	2280
	$V_{4,max}$	1680
	$V_{1,min}$	320
The minimum capacity of PFWT i , (m ³)	$V_{2,min}$	320
	$V_{3,min}$	320
	$V_{4,min}$	320
The maintenance cost coefficient	C_1	11.5
The repair and replacement fees	C_2	155
The correlation coefficient of energy consumption and generated freshwater	C_3	2.86

size NP, on the search ability of DE for the OOP are explored. A basic DE is used to study the relationship, setting $F = \text{Rand}[0.1 : 0.2]$, $\text{CR} = \text{Rand}[0.7 : 0.9]$, where $\text{Rand}[a : b]$ represents a uniformly distributed random value that ranges from a to b .

The different pairs $\{G_m, \text{NP}\}$ for DE are used to compute the solution of the OOP, and each algorithm is running 10

TABLE 2: The user's demands for freshwater $D(k)$.

Time k	Demands for flash water (m^3)	Time k	Demands for flash water (m^3)
1	1660	13	2525
2	1520	14	2430
3	1345	15	2300
4	1370	16	2490
5	1865	17	2765
6	2235	18	3030
7	2700	19	3095
8	2830	20	2970
9	2770	21	2655
10	2555	22	2350
11	2705	23	2085
12	2640	24	1870

TABLE 3: The time-of-use electricity price $P_e(k)$.

Time k	1-8	9-11	12-13	14-19	20-21	22	23-24
Electricity price (¥/kWh)	0.27	0.69	0.27	0.69	0.89	0.69	0.27

times. The results are listed in Table 4, in which the bold font is the best solution. It can be observed that the bigger these two parameters are, the stronger the search ability is.

It is worth noting that if G_m and NP are too big, the search speed of DE will decline sharply; and if the NP is small to a certain value, the algorithm is easy to fall into the locally optimal solution. Therefore, there is a compromise between the search speed and the accuracy when turning these two parameters.

4.1.2. The Scale Factor F and Crossover Rate CR . The scale factor F and crossover rate CR are another two important parameters in DE; the search ability of DE is sensitive to them too. In order to find out the effects of F and CR on the search ability of DE in this problem, a sensitivity analysis of these two parameters is presented.

Usually the F is in the range of $[0.0, 1.0]$ and CR is in the range of $[0.0, 1.0]$ [15]. In order to compare the effects of different F and CR fairly, the same initial individuals are used to calculate the optimal values of this OOP under different F and CR . The population size $NP = 100$ and the maximum number of iterations $G_m = 1000$. A basic DE is used to do this job, and the results are shown in Figures 3 and 4.

In Figure 3, the effects of different CR on the search ability of DE when $F = 0.1, 0.3, 0.5, 0.8, 1.0,$ and $\text{Rand}[0.1 : 0.3]$, respectively, are explored, in which it is found that OOP gets its best solution when $CR \geq 0.7$. In Figure 4, the effects of different F on the search ability of DE when $CR = 0.1, 0.3, 0.5, 0.8, 1.0,$ and $\text{Rand}[0.7 : 0.9]$, respectively, are studied, and it is found that OOP gets its minimal value when $F \leq 0.3$. So, we set $F = \text{Rand}[0.1 : 0.3]$ and $CR = \text{Rand}[0.7 : 0.9]$ for DE algorithm to compute the best solutions of OOP in the following sections.

4.1.3. The Two Maximum Numbers of Iterations G_1 and G_2 of TSDE. As mentioned in the above sections, the TSDE algorithm is divided into two stages. The two maximum iteration numbers of these two stages, G_1 and G_2 , must be determined before the TSDE working. Here, it is supposed that $G_1 + G_2 = 1000$, $NP = 100$, $F = \text{Rand}[0.1 : 0.3]$, and $CR = \text{Rand}[0.7 : 0.9]$, and each algorithm runs 10 times. The results are listed in Table 5, in which the bold font is the best solution. It is known that the TSDE gets its best solution when $G_1 = 300$ and $G_2 = 700$.

4.2. Pseudocode of TSDE. The pseudocode of TSDE is shown in Algorithm 1.

4.3. Comparative Study

4.3.1. Comparison between TSDE and Other Global Minimizing Algorithms. In this paper, genetic algorithm (GA) and basic DE are chosen to compare the search ability with STDE for this problem. The main parameters of GA are set as follows: the maximum number of iterations $G_m = 1000$, the length of chromosome $L_c = 20$, mutation factor $F = 0.6$, and crossover rate $CR = 0.1$; the main parameters of basic DE: the maximum number of iterations $G_m = 1000$, the population size $NP = 100$, mutation factor $F = \text{Rand}[0.1 : 0.3]$, and crossover rate $CR = \text{Rand}[0.7 : 0.9]$; the main parameters of TSDE: the maximum two numbers of iterations $G_1 = 300$, $G_2 = 700$, the population size $NP = 100$, mutation factor $F = \text{Rand}[0.1 : 0.3]$, and crossover rate $CR = \text{Rand}[0.7 : 0.9]$. All algorithms are coded in MATLAB 7.0 and executed in HP desktop 6300 MT with Intel Core i5-3470 CPU @3.2 GHz and 4 GB RAM.

Each of algorithm runs 10 times, and the statistical results are listed in Table 6, in which the bold font are the best solution. The average searching quality of TSDE is better than the others. Moreover, the standard deviations by basic DE and TSDE are much smaller than GA.

4.3.2. Comparison between Optimal Operation and Manual Operation. Finally, we compare the optimal operation with manual operation in the OOP. The scheduling strategy of manual operation is that all RO units are running until reaching the high amount limits of the PFWTs from time 1 to time 8 every day, when the time-of-use electricity price is the lowest. When the freshwater amount of each PFWT is at high limit, the corresponding RO units stop. At other time periods, the running status of each RO unit is only determined by the user's freshwater demand, and the time-of-use electricity price will no longer be considered. The optimal operation obeys the scheduling plan computed by TSDE.

Figure 5 is the comparison of freshwater generation of each RO unit between optimal operation and manual operation. As Figure 5 shows, those RO units which have smaller freshwater generation capacity, such as Unit 1 and Unit 2, are almost running because of its smaller running cost. On the contrary, the RO unit which has large capacity (its running cost increases according to its capacity), such as Unit 6, is opened only when necessary. So, it has low

TABLE 4: Analysis of NP and G_m of TSDE.

NP	G_m	Best solution (¥)	Mean value (¥)	Std. dev.
200	250	89758	90376.7	351.6491
250	200	89662	90471.4	556.8511
334	150	89116	89666.7	410.5998
500	100	88822	89926.1	423.0399
625	80	88725	89508.5	451.2566
1000	100	88429	88955.4	376.5983

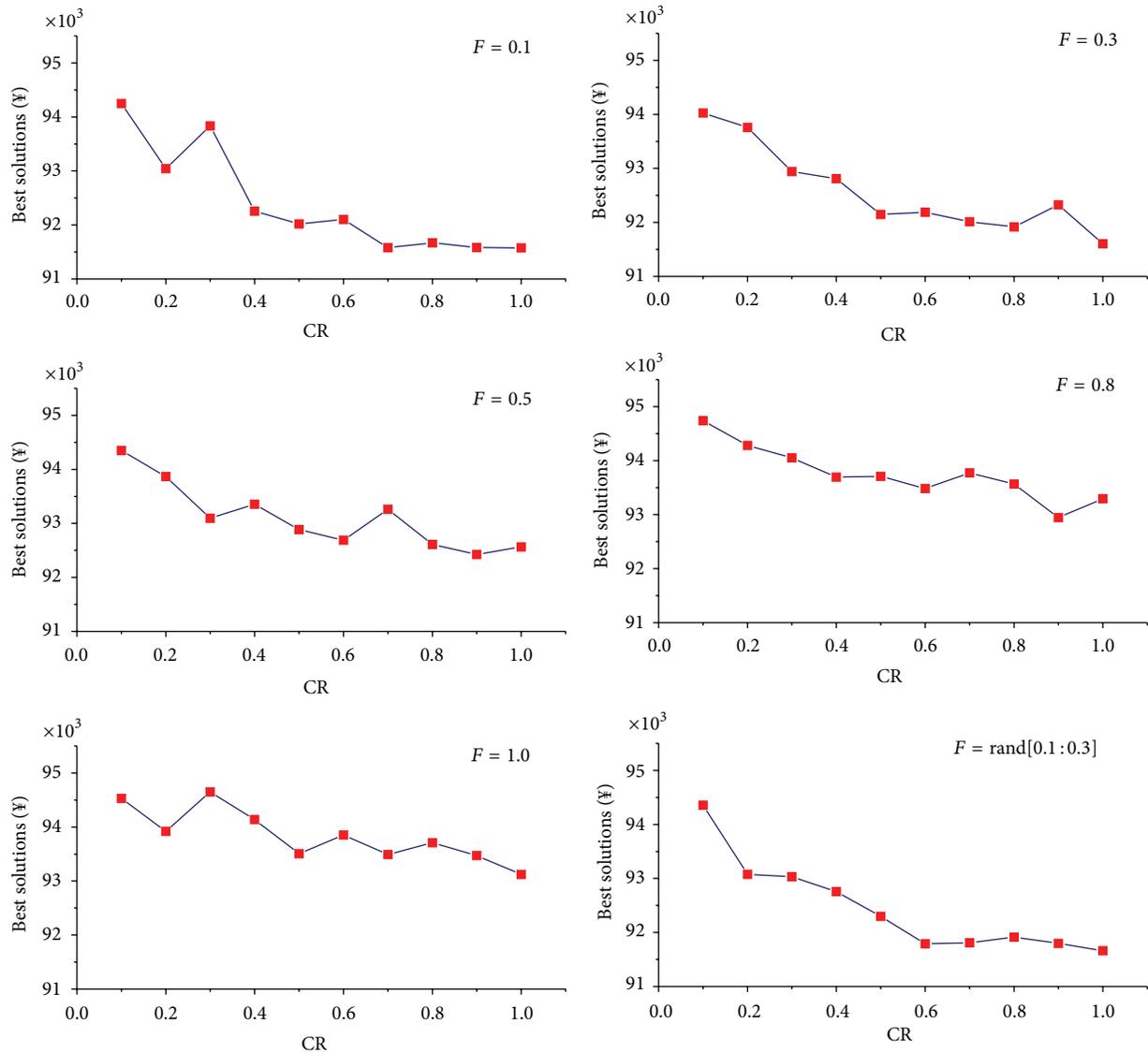


FIGURE 3: Analysis of CR in different F .

utilized efficiency. The others which have medium capacity are working intermittently.

Figure 6 is the comparison of capacity of each PFWT. The sum of capacity of all PFWTs fluctuates with the profile of freshwater demand.

From Figure 6, it is observed as follows.

- (1) Both of these two schedules open the RO units to generate freshwater from time 1 to time 8, and the sum of freshwater reaches its peak at time 8;
- (2) the manual operation does not take the time-of-use electricity price as the cost factor of TRC after time 8, so even at the highest electricity price period, such

TABLE 5: Analysis of G_1 and G_2 of TSDE.

G_1	G_2	Best solution (¥)	Mean value (¥)	Std. dev.
300	700	88140	88745.38	467.6048
500	500	88464	88878.9	383.1357
700	300	88556	88850	166.2274
1000	0	88275	89052	489.2724

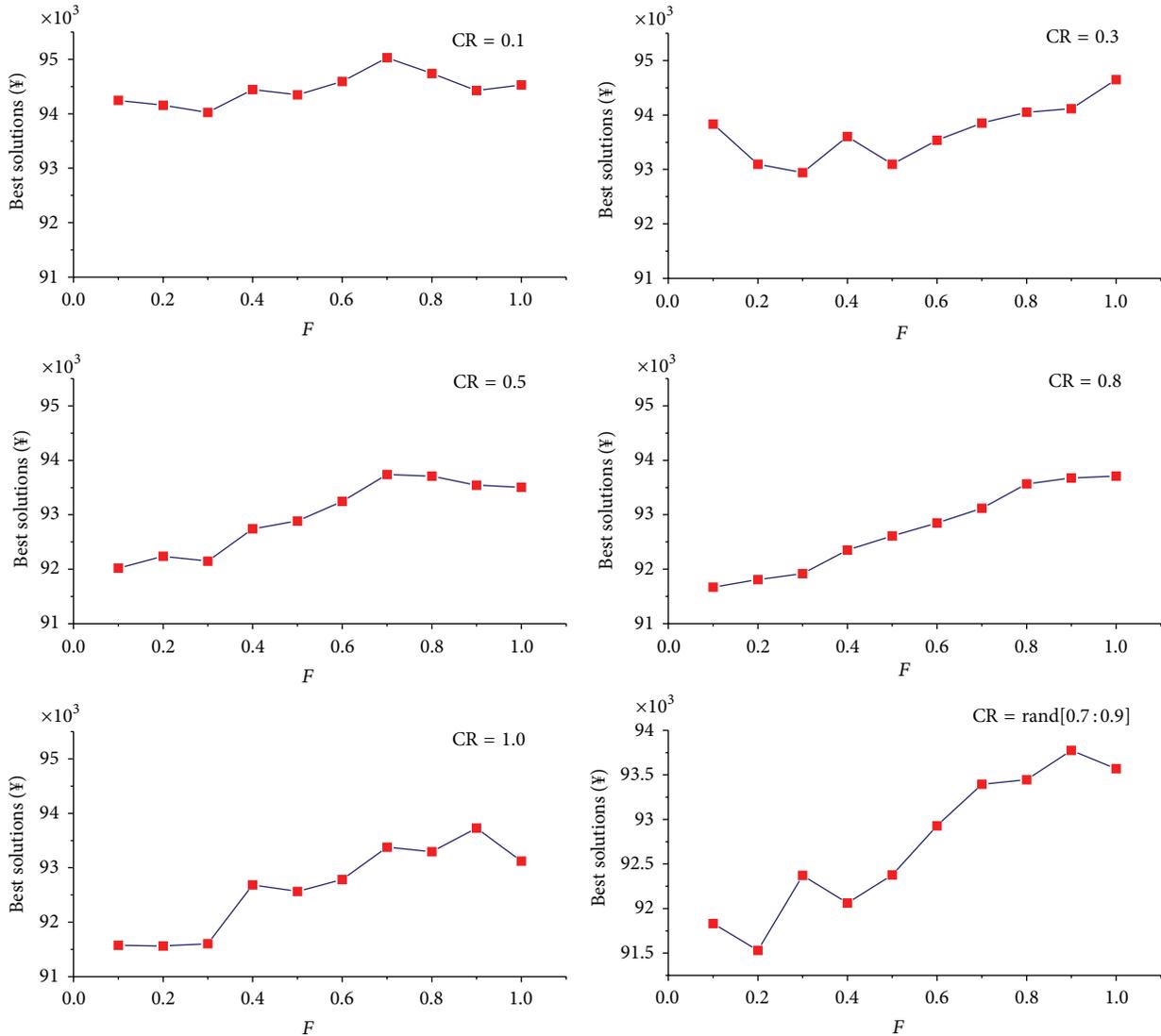


FIGURE 4: Analysis of F in different CR.

as time 15, the sum of freshwater of PFWTs is still increasing. That is, the RO units are still opening at these time periods, so that the TRC under manual operation cannot get its optimal value;

- (3) the optimal operation takes full use of its advantage in global optimization, so at times 8, 13, and 18, the sum of generated freshwater reaches its local peaks before the time-of-use electricity price gets higher. Therefore, when the electricity price is higher, the storage of freshwater is used to satisfy user's demand,

and the RO units will not be opened unless the storage of freshwater reaches its lower limit. In this way, the TRC of this plant under optimal operation is 5% lower than manual operation (Figure 7).

5. Conclusions

The OOP of large-scale parallel-unit SWRO desalination plant is modeled as a MINLP, in which binary-valued vectors indicate the on/off statuses of RO units and real-valued

```

Line      % Pseudocode of TSDE
(1)      Begin
(2)      Set NP = 100; D = 20; G1 = 300; G2 = 700;
(3)      % Stage One
(4)      Initialize population xi,G (i = 1, 2, ..., NP) (the vectors which represent the status of RO are initialized
          randomly in the range of [0, 1]; the vectors which represent the amount of freshwater are
          initialized as constants, which equal the average capacity of each RO unit)
(5)      G = 1
(6)      For G = 1 to G1
(7)          For k = 1 to K
(8)              For i = 1 to NP
(9)                  Randomly selected three integers r1, r2, and r3 from the set {1, 2, ..., NP}
                    (r1 ≠ r2 ≠ r3 ≠ i)
(10)                 F = 0.2 * rand + 0.1
(11)                 vi,G = xr1,G + F * (xr2,G - xr3,G)
(12)                 Randomly selected index rnbr(j) (rnbr(j) ∈ {1, 2, ..., D})
(13)                 CR = rand * 0.2 + 0.7
(14)                 For j = 1 to D
(15)                     If (rand < CR) or (j = rnbr(j))
(16)                         uij,G = vij,G
(17)                     Elseif
(18)                         uij,G = xij,G
(19)                     End If
(20)                     If (uij,G < xijL) or (uij,G > xijU)
(21)                         uij,G = xijL + rand * (xijU - xijL)
(22)                     End If
(23)                 End For
(24)                 If F(ui,G) > F(xi,G)
(25)                     xi,G+1 = ui,G
(26)                 Else
(27)                     xi,G+1 = xi,G
(28)                 End If
(29)             End For
(30)         End For
(31)     End For
(32)     % Stage Two
(33)     Initialize population xi,G (i = 1, 2, ..., NP) (30 percent of them are remained the best values
          from Stage One and 70 percent are generated randomly)
(34)     G = 1
(35)     For G = 1 to G2
(36)         For k = 1 to K
(37)             For i = 1 to NP
(38)                 Randomly selected three integers r1, r2, and r3 from the set {1, 2, ..., NP}
                    (r1 ≠ r2 ≠ r3 ≠ i)
(39)                 F = 0.2 * rand + 0.1
(40)                 vi,G = xr1,G + F * (xr2,G - xr3,G)
(41)                 Randomly selected index rnbr(j) (rnbr(j) ∈ {1, 2, ..., D})
(42)                 CR = rand * 0.2 + 0.7
(43)                 For j = 1 to D
(44)                     If (rand < CR) or (j = rnbr(j))
(45)                         uij,G = vij,G
(46)                     Elseif
(47)                         uij,G = xij,G
(48)                     End If
(49)                     If (uij,G < xijL) or (uij,G > xijU)
(50)                         uij,G = xijL + rand * (xijU - xijL)
(51)                     End If
(52)                 End For
(53)                 If F(ui,G) > F(xi,G)

```

```

(54)            $x_{i,G+1} = u_{i,G}$ 
(55)           Else
(56)            $x_{i,G+1} = x_{i,G}$ 
(57)           End If
(58)           End For
(59)           End For
(60)           End For
(61)           End
    
```

ALGORITHM 1: Pseudocode of TSDE.

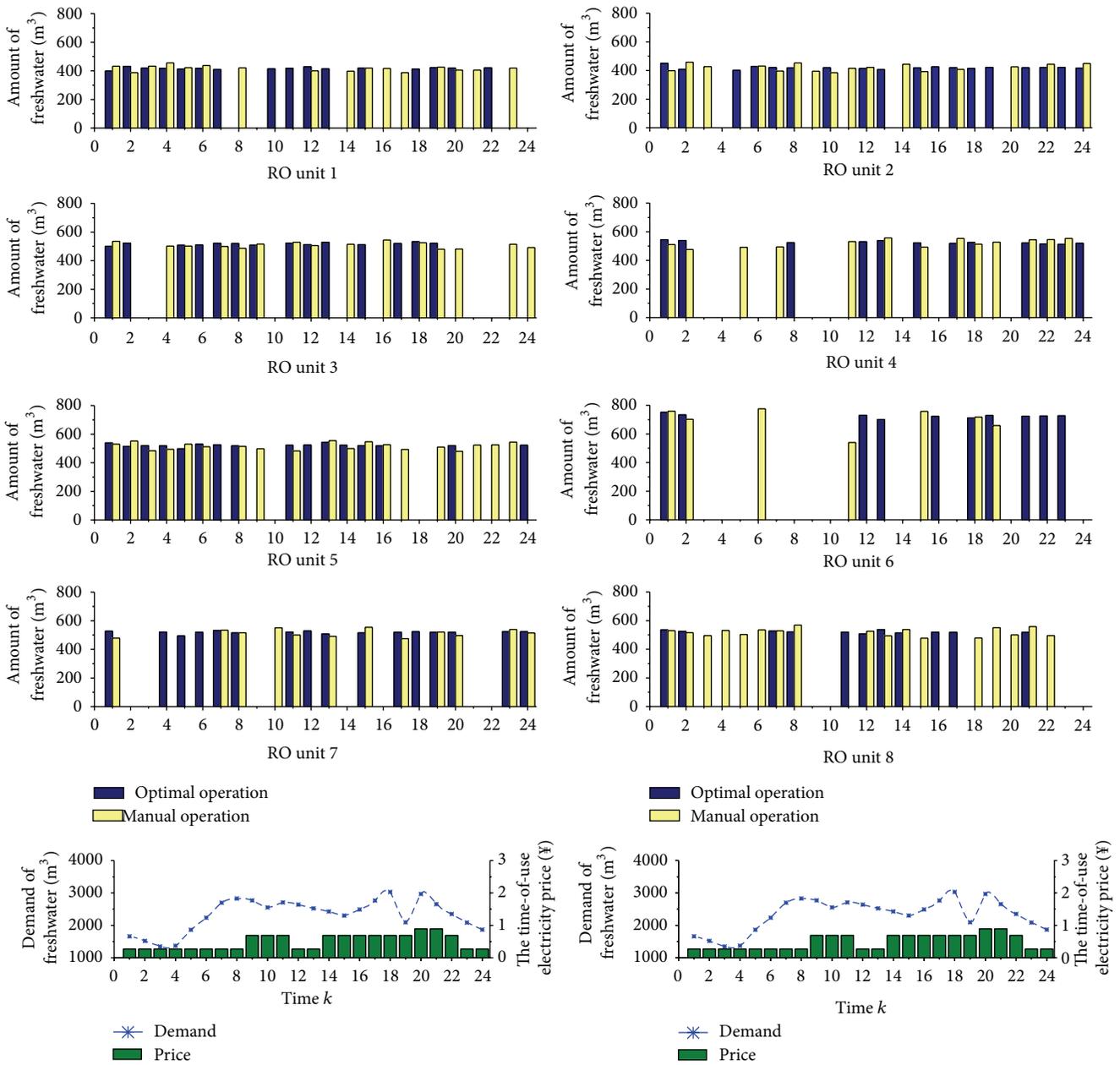


FIGURE 5: The comparison of capacity of each RO unit.

TABLE 6: Comparison of the best solutions of STDE with basic DE and GA (¥).

Algorithm	Best solution	Worst solution	Mean value	Std. dev.
GA	89680	91102	90556	622.9885
Basic DE	88429	89266	88955.4	376.5983
TSDE	88140	89246	88791.59	391.3258

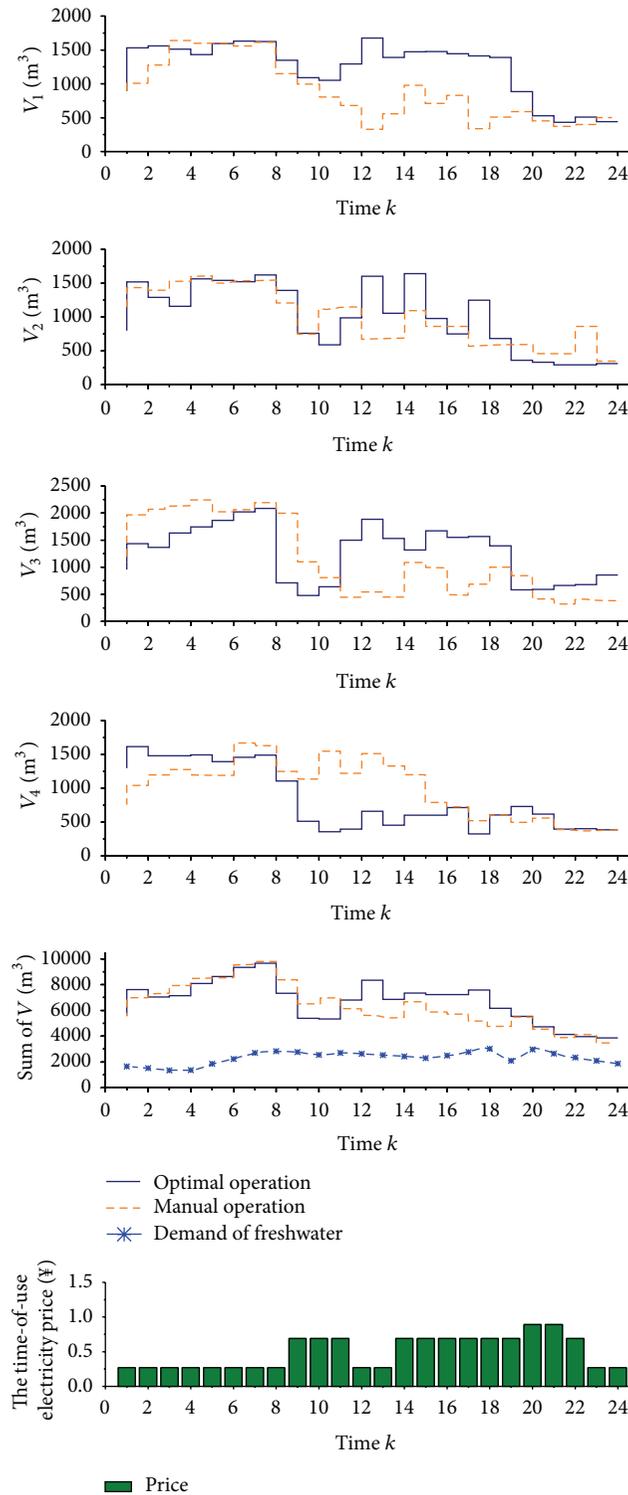


FIGURE 6: The comparison of capacity of each RO unit.

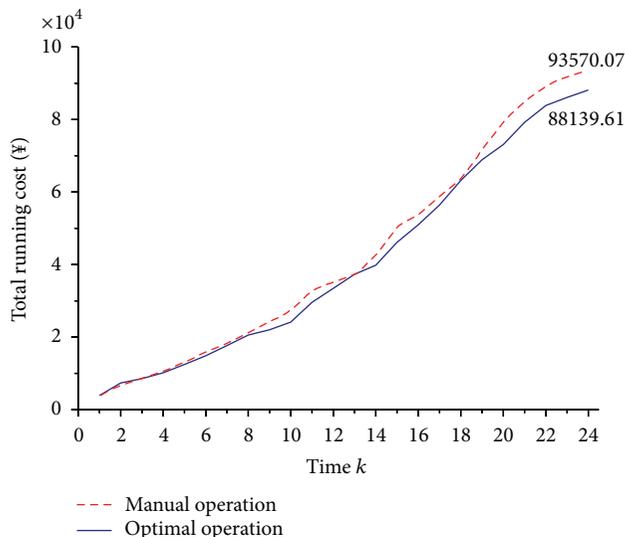


FIGURE 7: The comparison of TRC.

vectors indicate the amounts of freshwater generated by RO units and the amounts of supplied freshwater by PFWFs. The objective function of the OOP is the total running cost of the desalination plant, and the constraint functions include the technical limitations, the design requirements of each RO unit, and the freshwater demand of user. A novel DE, two-stage DE, is presented to solve this OOP, and the effects of its main parameters on the search ability are analyzed in this paper. Applying this TSDE to a 100,000 ton SWRO desalination plant in Liuheng Island, China, it is observed that the proposed TSDE can successfully improve the utilization rate of RO units to reduce the TRC.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by National Key Technology Research and Development Program of the Ministry of Science and Technology of China (2009BAB47B06) and the grants of National Natural Science Foundation of China (no. 61374142).

References

- [1] N. M. Wade, "Distillation of plant development and cost update," *Desalination*, vol. 136, no. 1-3, pp. 3-12, 2001.
- [2] E. El-Zanati and S. Eissa, "Development of a locally designed and manufactured small-scale reverse osmosis desalination system," *Desalination*, vol. 165, pp. 133-139, 2004.
- [3] A. Villafafila and I. M. Mujtaba, "Fresh water by reverse osmosis based desalination: simulation and optimisation," *Desalination*, vol. 155, no. 1, pp. 1-13, 2003.
- [4] M. G. Marcovecchio, P. A. Aguirre, and N. J. Scenna, "Global optimal design of reverse osmosis networks for seawater desalination: modeling and algorithm," *Desalination*, vol. 184, no. 1-3, pp. 259-271, 2005.
- [5] Y. Y. Lu, Y. D. Hu, D. M. Xu, and L. Y. Wu, "Optimum design of reverse osmosis seawater desalination system considering membrane cleaning and replacing," *Journal of Membrane Science*, vol. 282, no. 1-2, pp. 7-13, 2006.
- [6] Y. Y. Lu, Y. D. Hu, X. L. Zhang, L. Y. Wu, and Q. Z. Liu, "Optimum design of reverse osmosis system under different feed concentration and product specification," *Journal of Membrane Science*, vol. 287, no. 2, pp. 219-229, 2007.
- [7] Y. Saif, A. Elkamel, and M. Pritzker, "Optimal design of reverse-osmosis networks for wastewater treatment," *Chemical Engineering and Processing*, vol. 47, no. 12, pp. 2163-2174, 2008.
- [8] P. Sarkar, D. Goswami, S. Prabhakar, and P. K. Tewari, "Optimized design of a reverse osmosis system with a recycle," *Desalination*, vol. 230, no. 1-3, pp. 128-139, 2008.
- [9] F. Majali, H. Ettouney, N. Abdel-Jabbar, and H. Qiblawey, "Design and operating characteristics of pilot scale reverse osmosis plants," *Desalination*, vol. 222, no. 1-3, pp. 441-450, 2008.
- [10] M. Verhuelsdonk, T. Attenborough, O. Lex, and T. Altmann, "Design and optimization of seawater reverse osmosis desalination plants using special simulation software," *Desalination*, vol. 250, no. 2, pp. 729-733, 2010.
- [11] K. M. Sassi and I. M. Mujtaba, "Optimal operation of RO system with daily variation of freshwater demand and seawater temperature," *Computer and Chemical Engineering*, vol. 59, pp. 101-110, 2013.
- [12] Y. G. Lee, A. Gambier, E. Badreddin, S. Lee, D. R. Yang, and J. H. Kim, "Application of hybrid systems techniques for cleaning and replacement of a RO membrane," *Desalination*, vol. 247, no. 1-3, pp. 25-32, 2009.
- [13] D. W. Jing, *Optimization Design of Reverse Osmosis System*, Chemical Industrial Press, Beijing, China, 2006.
- [14] R. Storn and K. Price, "Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep. TR-95-012, International Computer Science Institute, Berkeley, Calif, USA, 1995.
- [15] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341-359, 1997.
- [16] R. Storn and K. Price, "Differential evolution—a simple evolution strategy for fast optimization," *Dr. Dobb's Journal*, vol. 78, pp. 18-24, 1997.
- [17] R. Storn and K. Price, *Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*, University of California, Berkeley, Calif, USA, 2006.
- [18] R. Storn, K. Price, and J. Lampinen, *Differential Evolution—A Practical Approach to Global Optimization*, Springer, New York, NY, USA, 2005.
- [19] Y. W. Zhong, L. J. Wang, C. Y. Wang, and H. Zhang, "Multi-agent simulated annealing algorithm based on differential evolution algorithm," *International Journal of Bio-Inspired Computation*, vol. 4, pp. 217-228, 2012.
- [20] X. S. Yang and S. Deb, "Two-stage eagle strategy with differential evolution," *International Journal of Bio-Inspired Computation*, vol. 4, pp. 1-5, 2012.

- [21] C. W. Carroll, "The created response surface technique for optimizing nonlinear restrained systems," *Operations Research*, vol. 9, pp. 169–184, 1961.
- [22] A. V. Fiacco and G. P. McCormick, "Extensions of SUMT for nonlinear programming: equality constraints and extrapolation," *Management Science*, vol. 12, pp. 816–828, 1968.
- [23] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002.
- [24] F.-Z. Huang, L. Wang, and Q. He, "An effective co-evolutionary differential evolution for constrained optimization," *Applied Mathematics and Computation*, vol. 186, no. 1, pp. 340–356, 2007.
- [25] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 561–579, 2010.
- [26] A. Ponsich and C. A. C. Coello, "Differential evolution performances for the solution of mixed-integer constrained process engineering problems," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 399–409, 2011.

Research Article

An Adaptive Evolutionary Algorithm for Traveling Salesman Problem with Precedence Constraints

Jinmo Sung and Bongju Jeong

Department of Information & Industrial Engineering, Yonsei University, 50 Yonsei-Ro, Seodaemaun-gu, Seoul 120-749, Republic of Korea

Correspondence should be addressed to Bongju Jeong; bongju@yonsei.ac.kr

Received 9 November 2013; Accepted 30 December 2013; Published 17 February 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 J. Sung and B. Jeong. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Traveling sales man problem with precedence constraints is one of the most notorious problems in terms of the efficiency of its solution approach, even though it has very wide range of industrial applications. We propose a new evolutionary algorithm to efficiently obtain good solutions by improving the search process. Our genetic operators guarantee the feasibility of solutions over the generations of population, which significantly improves the computational efficiency even when it is combined with our flexible adaptive searching strategy. The efficiency of the algorithm is investigated by computational experiments.

1. Introduction

In most business processes, a set of operations or activities are constrained by precedence relationships that are imposed by technological characteristics of products or services. The precedence constraints represent a typical operational structure constrained by sequentially and/or related operations. Therefore, the determination of operation sequences considering precedence constraints is one of the most important issues in many industrial problems such as production planning, scheduling, and project network planning problems. In most practices, however, sequencing problem is very difficult to solve because of its combinatorial complexity. More often than not, the solution may be impractical to be used in real environment due to system variability, while most managers in charge of operation sequencing want more sturdy and robust solutions. This study considers, specifically, traveling salesman problem with precedent constraints (TSPPC) and aims to obtain more robust and efficient operation sequence with minimum setups.

TSPPC is a variant of traveling salesman problem (TSP) because all nodes should be visited once but in predetermined order. Precedence constraints make TSPPC have the wider range of industrial applications such as scheduling, project management, and process routing. In solution methods,

however, TSP is known to be a class of NP-hard problem and TSPPC is even more complicated with additional constraints. Since no exact solution can be obtained in reasonable computational time and a good solution needs to be practical and implementable in real environment, careful development of new solution method is crucial. As a solution technique, we will consider the evolutionary algorithm which is proven to be very effective in large scale of solution space and provides high flexibility in searching strategy.

We will first present a mathematical model of TSPPC as an appropriate network flow model. A new evolutionary algorithm will be introduced and in subsequent sections modifications of algorithm will be proposed for further improvement of solution. Finally, experimental analyses will validate the efficiency of our algorithms.

2. Literature Review

Abundant researchers have been interested in various types of sequencing problem using TSP. Reinelt [1] developed a traveling salesman problem library (TSPLIB) which is meant to provide researchers with a broad set of test problems from various sources and with various properties. Chen [2] discussed an AND/OR precedence constrained sequencing

problem and formulated it as a state-constrained traveling salesman model and applied the model to assembly scheduling problem. Also, He and Kusiak [3] developed a mixed integer formulation and a simple and easy heuristic for a single machine scheduling problem with sequence-dependent setup times and precedence constraints. Recently, Su [4] proposed a unique reasoning method supported by an artificial intelligent technique of case-based reasoning with evolutionary algorithm to solve the TSPPC problem. Lee et al. [5] suggested a tree-structured precedence graph to solve the problem of selecting and sequencing operations in process planning with the objectives of minimizing the sum of operation processing, setup, and tool change costs with precedence constraints. Lambert [6, 7] formulated a disassembly scheduling problem by modification of two-commodity network flow model including TSPPC problem. Mingozi et al. [8] proposed an exact algorithm to solve the problem using dynamic programming and bounding functions to reduce state space graph. Ascheuer et al. [9] described the implementation of a branch and cut-algorithm and gave computational results for real-world instances and benchmark problems from TSPLIB. Many exact and heuristic algorithms have been developed in the field of operations research (OR) to solve the variants of TSP. However, precedent constraints always bother the researchers in making efficient algorithms because with them the problems become dramatically difficult.

Many researched also used the memetic computational algorithms to solve TSP. Jati and Suyanto [10] used the firefly algorithm with the discrete distance between two fireflies and the movement scheme. Ouaarab et al. [11] employed cuckoo search (CS) algorithm inspired by the breeding behavior of cuckoos. They extended and improved CS by reconstructing its population and introducing a new category of cuckoos. In spite of using memetic computation, precedent constraints are still challenging in TSP.

Meanwhile, genetic algorithm is widely accepted as efficient algorithm to be applied in many intractable problems like TSP. Zeng et al. [12] used genetic algorithm for antenna design. Poland et al. [13] solved exosensor distribution optimization problem using genetic algorithm to generate globally optimal sensor distributions for a smart home replica kitchen.

Some researchers used genetic algorithms to tackle TSPPC. As one of the first researchers using genetic algorithm for this problem, Potvin [14] introduced genetic algorithms for traveling salesman problem and its extensions. Recently, more efficient genetic algorithms were developed using some additional techniques such as special decoder, penalty function, and special genetic operator, all of which efficiently generate good feasible solutions by Michalewicz and Fogel [15]. Ghazalli and Murata [16] used genetic algorithm to find optimal disassembly sequences for disassembling the end-of-life product. He used topological sort method to generate feasible solutions and fix infeasible solutions. Moon et al. [17–19] also proposed a topological sort based evolutionary approach to solve TSPPC and tool selection problem. Yun and Moon [20] also used a similar technique to solve precedence-constrained sequencing problem. Even though most of these

approaches are very simple and easy to implement, a major drawback is that the quality of solution is not satisfactory enough because of randomness of solution generation. Avoiding randomness and guiding to the better choice of solution are essential to improve GA based algorithms.

In this study, we propose an adaptive evolutionary algorithm to improve the computational efficiency and the quality of solutions. The proposed algorithm searches only the feasible solution space by adopting efficient genetic operators and employs adaptive search strategies to adaptively apply the genetic operations on the current population. Throughout this paper, we use “evolutionary algorithm” as the same meaning as “genetic algorithm,” “evolutionary strategies,” and “evolutionary programming,” which are found in many literatures. The objective of this study is to develop an efficient evolutionary process scheme for TSPPC. The algorithm seeks a solution to minimize the total processing time for implementing all the operations with sequence dependent setup times.

3. Mathematical Model of Traveling Salesman Problem with Precedent Constraints (TSPPC)

The mathematical model of TSPPC is quite similar to the traveling salesman problem (TSP). The $G = (V, A)$ graph is used to define TSPPC, where $V = \{0, 1, \dots, n\}$ and $A = \{(i, j) \mid i, j \in V\}$ which indicate nodes and arcs, respectively, in graph G . In TSPPC, a node corresponds to an operation and an arc to a processing time. To formulate TSPPC, we need to modify the two-commodity network flow model (as discussed by Lambert [6]) used in TSP as follows.

TSP. The model assumes that each node has one demand unit and the starting node has N units to meet the demand of each node. In other words, the salesman starts with N units at the starting node and travels to the end node till meeting the demand of all nodes. The simplified commodity network flow model considers only the amounts of belonging when the salesman enters or leaves a node. Then, a mathematical model can be formulated as follows:

(i) parameters

N : the number of nodes;

t_{ij} : the distance from node i to node j ;

(ii) decision variables

a_k : integer variable which is decreasing aggregate counter;

$$x_{ij} : \begin{cases} 1, & \text{if node } j \text{ is visited next to } i \\ 0, & \text{otherwise;} \end{cases} \quad (1)$$

p_{ij} : decreasing partial counter which is decreased with 1 if node j is visited next to i ;

(iii) objective

$$\text{Minimize } \sum_{ij} t_{ij} \times x_{ij}; \quad (2)$$

(iv) subject to

$$\sum_j x_{ij} = 1 \quad \forall i, \tag{3}$$

$$\sum_i x_{ij} = 1 \quad \forall j, \tag{4}$$

$$\sum_i p_{ij} = a_j \quad \forall j, \tag{5}$$

$$\sum_i p_{i0} = N, \quad \sum_i p_{0i} = 0, \quad p_{jj} = 0 \quad \forall j, \tag{6}$$

$$p_{ij} \leq (N + 1) \times x_{ij} \quad \forall i, j, \tag{7}$$

$$\sum_j p_{ij} = a_i - 1 \quad \forall i, \tag{8}$$

$$x_{ij} = \text{binary}, \quad a_i = \text{integer} \quad \forall i, j. \tag{9}$$

The objective function (2) is to minimize total traveling distance in TSP, for example, the total processing time of operations in scheduling problem. Constraints (3) and (4) indicate that all nodes have one preceding and one subsequent node. Constraint (5) represents counter aggregation, where the amount of belonging at each node is calculated by sum of preceding belongings. The total belonging size is restricted by constraint (6). And constraint (6) also makes starting and end condition for the start and end point. Constraint (7) couples counter and flows. Constraint (8) is for counter decrement. The last constraint (9) restricts the decision variables to integer.

Then TSPPC is formulated as follows.

TSPPC. TSPPC has the same constraint sets of TSP and one additional constraint set for precedence relationships between node i and j as follows:

$$a_i > a_j. \tag{10}$$

Constraint (10) indicates that node i precedes node j .

4. Algorithm

This section presents our proposed evolutionary method which searches only feasible solution spaces to improve the computational efficiency. The overall procedure of the algorithm is shown in Figure 1. The procedure includes crossover and mutation processes along with adaptive search strategies. The concept of the adaptive scheme is to adaptively change the values of parameters to enhance the search efficiency.

The details of algorithms are described in the following subsections.

4.1. Solution Representation. A chromosome representing the sequence of operations is shown in Figure 2. Data are represented using the linked-list format for the evolutionary algorithm (Horowitz et al. [21], Michalewicz [22]).

Since each gene corresponds to an operation, Figure 2 represents an operation sequence as 2-3-1-5-4-6. Also we have

information of the precedence constraints as $n \times n$ matrix with parameters pre_{ij} . If $\text{pre}_{ij} = 1$, then i operation should be done before j operation.

4.2. Initial Solution and Parameter Setup. The n initial solutions can be generated by topological sort (Moon et al. [19], Yun and Moon [20]). All solutions of the population are evaluated by the degree of fitness. In terms of TSP, the objective is to find a visit sequence with the shortest traveling time through all nodes. The fitness value of each solution is computed as follows:

$$\text{fit}_k = \sum_{ij} t_{ij} \times x_{ijk}, \tag{11}$$

where fit_k is the fitness-function value of k th chromosome, t_{ij} indicates the setup time of changing operation i to operation j , and x_{ijk} is a binary variable for k th chromosome with a value of 1 if node i precedes node j and 0 otherwise.

We have four parameters to be set for our algorithm: crossover acceptance probability (pc_0), crossover selection probability (psc_k), mutation acceptance probability (pm_0), and mutation selection probability (psm_{gk}).

The crossover acceptance probability is used to obtain the number of crossover operations over the current population. Similarly, the mutation acceptance probability is used to decide whether a chromosome accepts mutation operation or not and consequently is used to obtain the number of mutation operations over the current population. The crossover selection probability is a probability of selecting a chromosome k (as a parent chromosome) on which crossover operation is performed and computed as follows:

$$\text{psc}_k = \frac{(\text{Max} - \text{fit}_k)}{\sum_c (\text{Max} - \text{fit}_c)}, \tag{12}$$

where Max is the maximum fit_k (fitness-function value) among n chromosomes.

According to (12), a chromosome with the low fitness-function value has a high probability for selection.

The mutation selection probability is a probability of selecting a gene (as both end points) in chromosome k to determine the mutation region and computed is follows:

$$\text{psm}_{gk} = \frac{t_{[g-1][g]} + t_{[g][g+1]}}{\text{fit}_k}, \tag{13}$$

where psm_{gk} is the probability to select g th gene as a start or end point of mutation region. $[i]$ means the value of i th gene. For example, $t[1][2]$ with operation sequence 2-5-3-4-6-1 indicates the setup time of changing operation 2 to operation 5. If a gene is the first or the last one, calculate one side value.

These all parameter values are computed before the algorithm proceeds.

4.3. Genetic Operations. Genetic operations are designed to search only feasible solutions over the entire generations. The modified topological sort that merges two instances into one

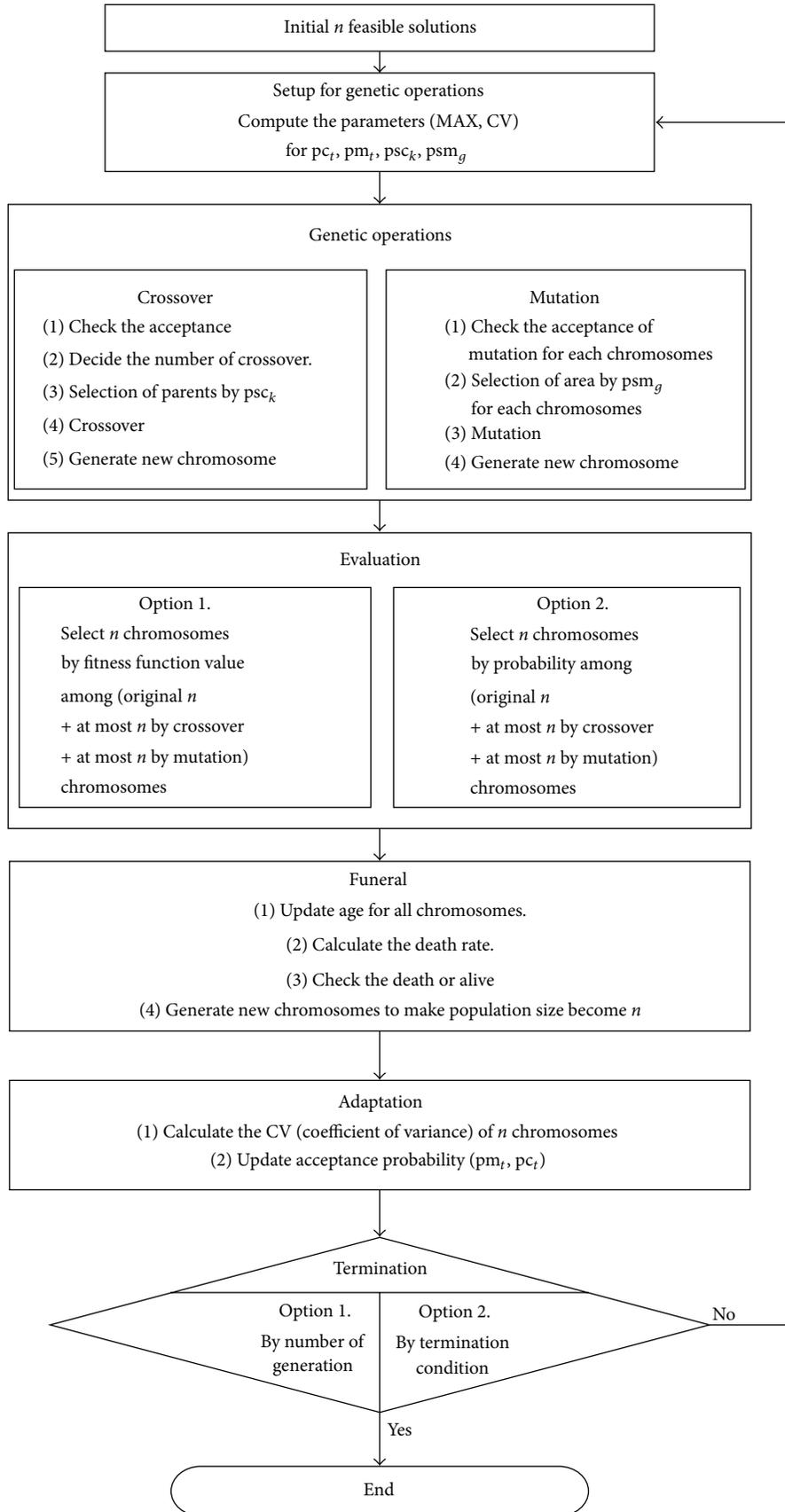


FIGURE 1: Procedure of the adaptive evolutionary algorithm.

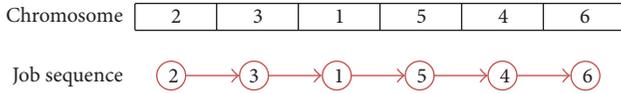


FIGURE 2: Representation of chromosome.

is considered to generate feasible chromosomes. We can use pc_0 and pm_0 , which are the probabilities of crossover and mutation, respectively.

4.3.1. Crossover. A crossover is operated according to the value of pc_0 , which is a crossover probability. If the crossover is accepted, two chromosomes will be selected for crossover. With a cumulative probability of crossover selection probabilities of each chromosome, that is, cumulative psc_k , two chromosomes are chosen using a random number generator. Thus, the higher chance of selection goes to a chromosome with the higher crossover selection probability. After two chromosomes are selected in this manner, an offspring through crossover operation on the chromosome is created by incremental inclusion of “selectable” nodes. A node is selectable if all precedent nodes are already included in the current chromosome or no precedent node exists. Our procedure maintains a selectable node set, E , until the set is empty. Let L be the length of a chromosome and let l be the position of genes in the chromosome. The crossover procedure is described as follows.

Step 1. Create a graph $G = (N, A)$ of TSPPC. Set $l = 1$

Step 2. Create a selectable node set E from G .

Step 3. Select two l th genes from both parent chromosomes. We have four possible cases as follows:

Case 1. Two selected genes are different and found in E . Select one arbitrarily.

Case 2. Two selected genes are same and found in E . Then select that one.

Case 3. Only one gene is selected and found in E . Then select that one.

(This happens when a gene has been removed in the previous iteration.

Case 4. No selected genes are found in E . Then select a gene (node) from E arbitrarily.

Step 4 (update G). Delete the selected node in Step 3 with the corresponding arcs.

Step 5. If $l = L$, terminate. Otherwise, set $l = l + 1$ and go to Step 2.

Since all genes in a new chromosome are selected from the selectable node set E , the precedent constraints are always satisfied, which means the crossover operations search are

always the feasible solution spaces. Over the crossover operations, the maximum n chromosomes are newly generated from n original chromosomes,

4.3.2. Mutation. A mutation is performed on a chromosome in order to create, by chance, an unexpected good solution. In our algorithm, the mutation is operated according to the value of pm_0 , mutation acceptance probability. If mutation is accepted, a mutation region is determined so that the mutation is performed on an interval between two genes. With a cumulative probability of mutation selection probabilities of each gene of given chromosome, that is, cumulative psm_{gk} , two genes are chosen using a random number generator. This means that the higher chance of selection goes to a gene with the higher mutation selection probability.

After selecting a pair of genes, all genes between the pair are topologically sorted and subject to only their precedence constraints. It is obvious that this mutation operation keeps the feasibility of newly created chromosome because the front part (or the latter part) of the mutation region satisfies the precedence constraints with the mutation region in the original chromosome, and even the mutation region is replaced with other feasible region, the feasibility is still maintained.

The mutation operations create newly maximum n chromosomes with n original chromosomes,

4.4. Termination Criterion. We have two options to terminate the algorithm. The first one is to set the fixed number of generations, which is easy and popular way. The other criterion is to use the best fitness function value of each generation and check the trend of improvement over the generations. If the best fitness function value at generation t ($\min fit_t$) is not changed for the certain number of generations, the algorithm stops. This is represented as follows:

$$\min fit_t = \min fit_{t+1} = \dots = \min fit_{t+R}, \quad (14)$$

where $\min fit_t$ is the best fitness-function value at the t th generation.

If termination criterion is not satisfied, the algorithm selects n chromosomes for the next generation. We may have two options for selection. Firstly, we can choose n chromosomes by evaluating the fitness function values among all chromosomes from the original, crossover, and mutation operations. The second way for selection is to use the probability of each chromosome. The chromosome with lower fit_t -value has the higher probability for selection as follows:

$$psxg_k = \frac{\text{Max} - fit_k}{\sum_i (\text{Max} - fit_i)}, \quad (15)$$

where $psxg_k$ is the probability that the k th chromosome is selected for next generation.

4.5. The Aging of Chromosome. As generations continue, some survival chromosomes are getting older. All organisms experience aging process and may have a peak time to achieve

TABLE 1: Adaptive search strategies.

Factor	Adaptive search strategy	
	Crossover acceptance probability	Mutation acceptance probability
The number of generations	Increase	Decrease
Variance of fitness value		
High	Increase	Decrease
Low	Decrease	Increase

the best performance. We can employ this natural property to improve the searching spaces. With a lifespan value, funeral probability is computed as follows:

$$\text{fnr}_k = 1 - e^{-\text{age}_k/\text{als}}, \quad (16)$$

where fnr_k is the probability of death and age_k is the age of k th chromosome and als is the average lifespan.

A chromosome with the higher funeral probability has the lower chance of survival over the next generation.

4.6. Adaptive Search Strategy. In evolutionary algorithm, the population of generation changes as the generations proceeds. In order to obtain the better solutions, the better population needs to be generated at each generation. We can use this idea to create the search strategy for getting the better solution spaces. Due to the necessity of adaptability to each generation, we use a term of “adaptive” strategy.

For development of adaptive search strategy, we need to identify the characteristics of population. The parameters used in our algorithm can be quite useful to do so. Basically, new chromosomes are created by genetic operations, that is, crossover and mutation operations. This means that modifying the parameters of genetic operations makes the algorithm have the adaptability. We consider the crossover and mutation acceptance probabilities as the parameters for the algorithm to have the adaptability. Table 1 shows how we change the two parameters according to the number of generations and variance of fitness function values of the current population.

Without loss of generality, in evolutionary algorithm we assume that as the number of generations increases, the chance of finding the better solution also increases. In this context, we think that as the number of generations increases, the more crossover operations are necessary because the better solution can be obtained from the more crossover among them, while the number of mutation operations need to be reduced in order to avoid unnecessary solution regions. At the same time, we look into the distribution of fitness function values of chromosomes. If the variance of the fitness function values is high, we need more crossover operations on the population to get the unexplored solutions in the current solution region. In this case, however, the mutation operation does not have to be encouraged in order to avoid unnecessary solution search. We have the opposite

statements in case of high variance of fitness function value in population.

The above adaptive search strategies can be simply implemented with the following parameters:

$$\text{pc}_t = (\text{pc}_0)^{\alpha \times t \times \text{CV}_t}, \quad (17)$$

$$\text{pm}_t = (\text{pm}_0)^{\beta/t \times \text{CV}_t}, \quad (18)$$

where pc_t and pm_t are the acceptance probabilities of crossover and mutation at t th generation, respectively. α and β are the parameters reflecting the characteristics of problem. CV_t is coefficient of variance of population at t th generation.

5. Computational Experiments

In this section, we present the results of our experimentation with the proposed algorithm. We investigate the efficiency of genetic operations that avoid the generation of infeasible solution in the next subsection, the behavior of adaptive strategies, and then compare the results against other algorithm, and finally, where possible, the results are compared with the best known solutions.

5.1. Efficiency of Feasible Solution Search. Our approach uses the topological sort and special genetic operation procedures to generate feasible populations. In order to focus on our genetic operations that guarantee the feasibility of solution, we present a general approach that uses a separate feasibility check module and compare the computation results. The overall procedure of the general approach is shown in Figure 3.

For experiments, the initial parameters are given as $\text{pm}_0 = 0.8$, $\alpha = 1$, and $\beta = 1$. The experiments are performed on micro-PC with 3.0 GHz processor and 2 GB RAM. Table 2 shows the results of the general approach and our adaptive evolutionary algorithm.

From the results, we observe that two approaches find the best solutions for the small size examples. However, our algorithm is far efficient in computation time for especially large size problems, which is because our algorithm explores only the limited feasible solution space. Also our algorithm shows much better solution results than the general approach. The computation time of the general approach significantly increases as the size of problem increases. This is because the crossover operations on the infeasible chromosomes generate infeasible chromosomes again and it repeats without improving the solution. Since the probability of infeasibility increases as the number of nodes increases, the general approach is getting worse for the large size problems.

5.2. Behavior of Adaptive Search Strategy. Sets of computational experiments have been conducted to explore the behavior of adaptive search strategy. The experiment for 7 nodes sequencing problem is performed and the results are shown in Figure 4.

In the results, we observe that the crossover acceptance probabilities increase according to the number of generations. And when CV of population is high, the acceptance

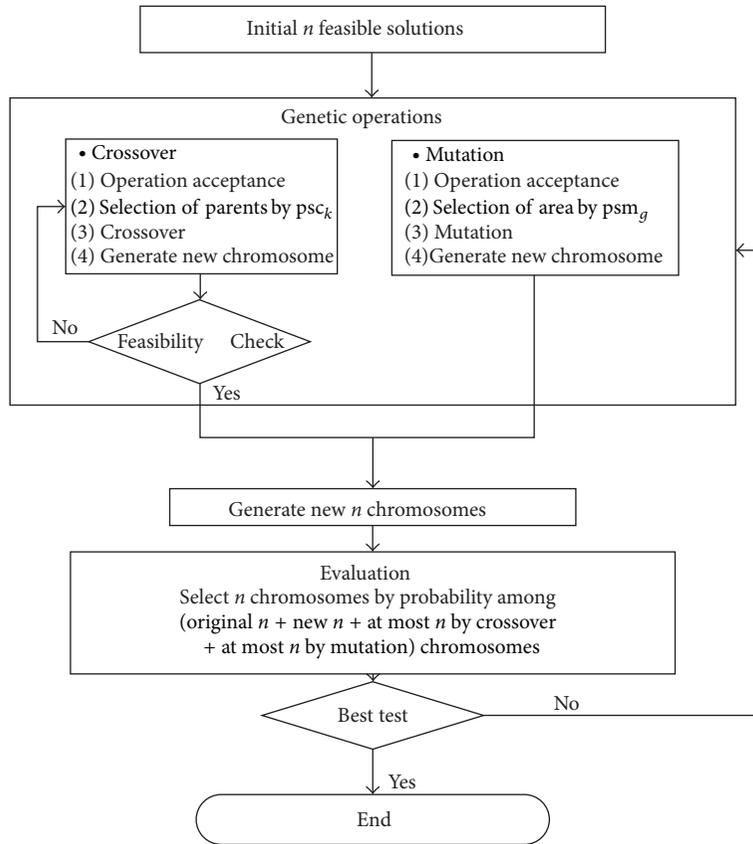


FIGURE 3: The procedure of a general approach that uses the separate feasibility check module.

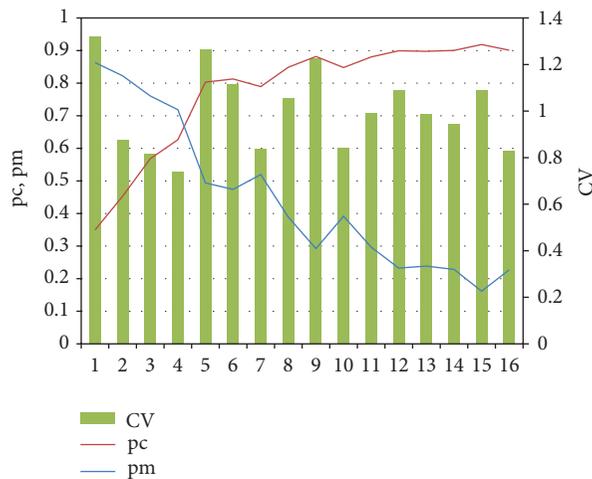


FIGURE 4: pc, pm, and CV graph over the generation.

probability of crossover is also high. For example, in Figure 4, the CV of 5th generation is high and so the acceptance probability of crossover is also high. And then the 6th CV is forced to decrease by the increased probability of crossover, which means that the algorithm tries to generate the solutions near the original solutions. When CV is low, the acceptance probability of mutation is high. When the CV at the 7th generation is low, then the acceptance probability of mutation

increases and the 8th CV is forced to increase by the increased acceptance probability of mutation. This means that the algorithm generates new solutions far different from the original solutions.

5.3. Comparison against Other Algorithms. In this section, the performance of our approach is compared with the

TABLE 2: The performance experiment result 1.

Number of nodes	Parameter		General approach			AEA*		
	n	pc_0	Best solution	Frequency of best	CPU Time (sec)	Best solution	Frequency of best	CPU time (sec)
7	7	0.6	26	6	0.922	26	1	0.006
	14	0.7	26	3	0.015	26	4	0.016
25	25	0.6	134	10	10.422	134	4	0.047
	50	0.7	134	5	6.042	134	5	0.094
35	30	0.6	177	11	18.803	177	9	0.218
	35	0.7	180	12	23.844	177	10	0.235
	70	0.8	180	11	19.424	177	16	0.625
45	45	0.6	214	15	44.314	209	18	1.047
	45	0.7	214	17	38.688	207	19	1.75
	50	0.8	209	18	42.124	207	17	1.547
70	70	0.6	383	33	532.406	363	43	9.016
	70	0.7	372	21	229.319	363	40	9.796
	140	0.8	372	27	245.247	363	45	14.281

*The proposed adaptive evolutionary algorithm.

TABLE 3: The performance experiment result 2.

Number of nodes	Optimization technique		AEA		
	Optimal solution	CPU Time (sec)	n	Best solution	CPU time (sec)
10	18	1.51	25	18	0.023
			50	18	0.047
12	27	156.72	25	27	0.062
			50	27	0.062
15	49	70626	25	49	0.088
			50	49	0.078
20	—	—	25	68	0.086
			50	73	0.125

optimization technique by OPL-studio (IBM, Available from: <http://www.ilog.com/products/optimization/>) [23] in terms of the computational time and the quality of solutions. Ten experiments are performed for each problem size on micro-PC with 3.0 GHz processor and 1 GB RAM. The results are shown in Table 3. The initial probabilities are given as $pc_0 = 0.8, 0.7$ and $pm_0 = 0.6, 0.7$ and $\alpha = 1$ and $\beta = 1$.

The results show that our algorithm finds the best solution in negligible computation time, while the optimization algorithm requires the considerable computation time and even for 20 nodes problem, it cannot provide a solution in reasonable time.

For the larger size experiments, we use the examples of networks with 25, 35, 45, 70, 85, and 100 nodes. We compared our algorithm with the evolutionary algorithm proposed by Yun and moon [20] in terms of the computing times and the best solution values. The experiments have been performed on micro-PC with 3.0 GHz processor and 2 GB RAM. The results are shown in Table 4.

In Table 4, we observe that two approaches find the best solutions for various examples, but our algorithm is much more efficient in terms of the computing time.

5.4. Computational Performance against the Best Known Solutions. TSPLIB problems sets are used to verify the performance of the adaptive evolutionary algorithm. We compare the results of our algorithm to the best known values. We use Sequential Ordering Problem (SOP) set. This problem is an asymmetric traveling salesman problem with additional constraints. Given a set of n nodes and distances for each pair of nodes, find a Hamiltonian path from node 1 to node n of minimal length which takes given precedence constraints into account. Each precedence constraint requires that node i have to be visited before node j . The initial probabilities are given as pc_0 and $pm_0 = 1$, $\alpha = 1$, and $\beta = 1$. And $10 * 8$ (8 types of option) = 80 times of experiments for each problem were performed to eliminate the random effects.

The results are shown in Table 5. The table shows the % best value to the best known value for the problem set. We used combined options of termination (T), selection (S), and funeral (F) criterion for each problem and compared all the results. On average, 128–131% performances are obtained for all the problems and T1S2F shows the best option. However, we can employ the best option for the specific problem. For example, T1S2 option is the best for the problem *rbg050c*.

TABLE 4: The performance experiment result 3.

Node	Yun and Moon (2011) [20]						AEA					
	Best value			CPU time			Best value			CPU time		
	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
7	26	26.4	28	0.00	0.01	0.01	26	26.4	28	0.00	0.01	0.02
25	134	134.8	141	0.34	0.98	2.56	134	134.2	136	0.31	0.97	2.39
35	177	179.9	186	0.78	5.62	10.84	177	180.3	183	2.36	5.62	11.31
45	209	216.3	224	12.92	29.73	48.66	207	214.9	223	12.64	29.87	57.18
70	368	375.8	386	83.05	146.20	223.80	364	374.5	380	46.38	93.59	161.20
85	420	432.8	471	133.60	326.04	587.30	394	427.3	448	60.22	290.93	467.80
100	458	492.25	520	303.00	676.87	1314.00	448	480.7	505	178.90	491.39	735.60

TABLE 5: Comparison against best known values with TSPLIB problem set.

Name of problem	(% Best value*/best known									Min
	T1S1**	T1S2	T2S1	T2S2	T1S1F	T1S2F	T2S1F	T2S2F		
br17.10	100%	100%	101%	103%	100%	100%	100%	100%	100%	100%
br17.12	100%	100%	100%	103%	100%	100%	100%	100%	100%	100%
ESC07	100%	100%	100%	103%	100%	100%	100%	100%	100%	100%
ESC12	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
ESC25	142%	155%	144%	150%	140%	156%	146%	146%	140%	140%
ESC47	363%	373%	365%	351%	378%	373%	345%	323%	323%	323%
ESC63	119%	120%	125%	122%	122%	118%	119%	122%	118%	118%
ft53.1	152%	141%	143%	152%	145%	141%	144%	145%	141%	141%
ft53.2	141%	138%	132%	138%	137%	134%	136%	136%	132%	132%
ft53.3	121%	123%	123%	126%	125%	121%	122%	125%	121%	121%
ft53.4	122%	108%	129%	128%	125%	108%	130%	111%	108%	108%
ft70.1	128%	118%	121%	120%	125%	121%	124%	123%	118%	118%
ft70.2	115%	123%	122%	124%	118%	118%	118%	125%	115%	115%
p43.1	111%	104%	103%	103%	107%	103%	104%	112%	103%	103%
p43.2	110%	103%	117%	114%	109%	102%	111%	108%	102%	102%
p43.3	107%	101%	106%	100%	103%	102%	103%	102%	100%	100%
p43.4	106%	101%	107%	106%	102%	100%	104%	102%	100%	100%
prob.42	190%	182%	197%	196%	192%	187%	189%	191%	182%	182%
rbg048a	111%	107%	112%	107%	108%	107%	111%	115%	107%	107%
rbg050c	118%	108%	126%	118%	115%	109%	120%	117%	108%	108%
ry48p.1	126%	121%	118%	123%	122%	122%	122%	122%	118%	118%
ry48p.2	121%	111%	122%	115%	117%	116%	119%	124%	111%	111%
ry48p.3	119%	113%	119%	121%	115%	115%	118%	124%	113%	113%
ry48p.4	116%	111%	108%	108%	111%	108%	108%	112%	108%	108%
avg	131%	128%	131%	130%	130%	128%	129%	129%		

*The best value obtained by the adaptive evolutionary algorithm.
 **T: termination option, 1 = the first option, 2 = the second option.
 S: selection option of *n* chromosomes, 1 = the first option, 2 = the second option.
 F: funeral option if applied.

In Table 6, the best results obtained in Table 5 are compared with the best known values. We use the selection option 2 in Figure 1 and the first termination criterion with max generation = 10,000. On average, 124% result is obtained. When ESC47 and prob.42 are excluded, the average percentage of best value decreases to 112%.

6. Conclusion

In this paper, we presented an adaptive evolutionary algorithm for solving a traveling salesman problem with precedence constraints (TSPPC). Our algorithm employs genetic operators that guarantee the feasibility of solutions, and as

TABLE 6: Comparison against best known values using the best options for each problem.

Name of problem	Best	Avg.	Worst	% Best	% Avg.	% Worst	Best known value
br17.10	55	55.9	58	100%	102%	105%	55
br17.12	55	55	55	100%	100%	100%	55
ESC07	2125	2125	2125	100%	100%	100%	2125
ESC12	1675	1720.6	1791	100%	103%	107%	1675
ESC25	2354	3378.4	3972	140%	201%	236%	1681
ESC47	4160	5648.3	6628	323%	439%	515%	1288
ESC63	73	80.9	88	118%	130%	142%	62
ft53.1	10619	11537.4	12323	141%	153%	164%	7531
ft53.2	11000	12092.1	13340	132%	145%	160%	8335
ft53.3	13231	13797	14236	121%	126%	130%	10935
ft53.4	15579	16263.3	16793	108%	113%	116%	14425
ft70.1	46390	48696.2	49623	118%	124%	126%	39313
ft70.2	46485	49162.1	51920	115%	122%	128%	40422
p43.1	28830	29107	29685	103%	104%	106%	27990
p43.2	28896	47979.5	56225	102%	169%	198%	28330
p43.3	28680	29362.5	29555	100%	102%	103%	28680
p43.4	82960	83393.5	83590	100%	101%	101%	82960
prob.42	443	530.7	586	182%	218%	241%	243
rbg048a	376	412.6	435	107%	118%	124%	351
rbg050c	505	528.7	564	108%	113%	121%	467
ry48p.1	18650	21732.2	25057	118%	138%	159%	15805
ry48p.2	18499	22210.8	25969	111%	133%	156%	16666
ry48p.3	22480	25029.2	28744	113%	126%	144%	19894
ry48p.4	33961	34944.7	36421	108%	111%	116%	31446
Average				124%	141%	154%	
Average excluding ESC47 and prob. 42				112%	121%	129%	

the results the searching spaces are significantly reduced. The algorithm also adopts an adaptive search strategy to adaptively apply the crossover and mutation operations on the current generation, which improves the quality of solution as well as the computational efficiency.

The experimental results show that our proposed approach outperforms other evolutionary algorithm in terms of the quality of solution and computation time. Also, we observe that for the small size problems. The algorithm mostly finds the best solution and for even the large size problems, the quality of solution is not much deteriorated and the computation time is quite less. Finally, the proposed algorithm has wide range of options to be applied in various types of problems. We can use combined options of termination, selection of chromosome, and funeral probability.

For future research, our algorithm can be improved in a way to generate more robust solutions, which means that the deviation from a proposed solution by algorithm does not deteriorate the objective function value much so that more practical applications can be possible in real practices.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] G. Reinelt, "TSPLIB. A traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.
- [2] C. L. P. Chen, "AND/OR precedence constraint traveling salesman problem and its application to assembly schedule generation," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 560–562, November 1990.
- [3] W. He and A. Kusiak, "Scheduling manufacturing systems," *Computers in Industry*, vol. 20, no. 2, pp. 163–175, 1992.
- [4] Q. Su, "Applying case-based reasoning in assembly sequence planning," *International Journal of Production Research*, vol. 45, no. 1, pp. 29–47, 2007.
- [5] D.-H. Lee, D. Kiritsis, and P. Xirouchakis, "Iterative approach to operation selection and sequencing in process planning," *International Journal of Production Research*, vol. 42, no. 22, pp. 4745–4766, 2004.
- [6] A. J. D. Lambert, "Exact methods in optimum disassembly sequence search for problems subject to sequence dependent costs," *Omega*, vol. 34, no. 6, pp. 538–549, 2006.
- [7] A. J. D. Lambert, "Optimizing disassembly processes subjected to sequence-dependent cost," *Computers and Operations Research*, vol. 34, no. 2, pp. 536–551, 2007.
- [8] A. Mingozzi, L. Bianco, and S. Ricciardelli, "Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints," *Operations Research*, vol. 45, no. 3, pp. 365–377, 1997.

- [9] N. Ascheuer, M. Jünger, and G. Reinelt, "Branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints," *Computational Optimization and Applications*, vol. 17, no. 1, pp. 61–84, 2000.
- [10] G. K. Jati and S. Suyanto, "Evolutionary discrete firefly algorithm for travelling salesman problem," in *Adaptive and Intelligent Systems*, vol. 6943 of *Lecture Notes in Computer Science*, pp. 393–403, Springer, 2011.
- [11] A. Ouaarab, B. Ahiod, and X. S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Computing and Applications*, 2013.
- [12] S. Zeng, Z. Liu, C. Li, Q. Zhang, and W. Wang, "An evolutionary algorithm and its application in antenna design," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 2, pp. 129–137, 2012.
- [13] M. P. Poland, C. D. Nugent, H. Wang, and L. M. Chen, "Genetic algorithm and pure random search for exosensor distribution optimisation," *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 359–372, 2012.
- [14] J.-Y. Potvin, "Genetic algorithms for the traveling salesman problem," *Annals of Operations Research*, vol. 63, pp. 339–370, 1996.
- [15] Z. Michalewicz and D. Fogel, *How to Solve It: Modern Heuristics*, Springer, New York, NY, USA, 2004.
- [16] Z. Ghazalli and A. Murata, "An integrated TSP-GA with EOL cost model for selecting the best EOL option," *International Journal of Industrial Engineering Computations*, vol. 2, no. 4, pp. 775–792, 2011.
- [17] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," *European Journal of Operational Research*, vol. 140, no. 3, pp. 606–617, 2002.
- [18] C. Moon, M. Lee, Y. Seo, and Y. H. Lee, "Integrated machine tool selection and operation sequencing with capacity and precedence constraints using genetic algorithm," *Computers and Industrial Engineering*, vol. 43, no. 3, pp. 605–621, 2002.
- [19] C. Moon, Y. Yun, and C. Leem, "Evolutionary algorithm based on topological sort for precedence constrained sequencing," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 1325–1332, September 2007.
- [20] Y. Yun and C. Moon, "Genetic algorithm approach for precedence-constrained sequencing problems," *Journal of Intelligent Manufacturing*, vol. 22, no. 3, pp. 379–388, 2011.
- [21] E. Horowitz, S. Sahni, and S. Anderson-Freed, *Fundamentals of Data Structures in C*, Silicon Press Summit, Summit, NJ, USA, 2nd edition, 2007.
- [22] Z. Michalewicz, *Genetic Algorithms + Data Structures*, Springer, New York, NY, USA, 1996.
- [23] IBM, "IBM ILOG Optimization and Analytical Decision Support Solutions," <http://www.ilog.com/products/optimization/>.

Review Article

Bioinspired Evolutionary Algorithm Based for Improving Network Coverage in Wireless Sensor Networks

Mohammadjavad Abbasi, Muhammad Shafie Bin Abd Latiff, and Hassan Chizari

Universiti Teknologi Malaysia, Malaysia

Correspondence should be addressed to Mohammadjavad Abbasi; mj_abbasi55@yahoo.com

Received 20 October 2013; Accepted 2 January 2014; Published 12 February 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 Mohammadjavad Abbasi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Wireless sensor networks (WSNs) include sensor nodes in which each node is able to monitor the physical area and send collected information to the base station for further analysis. The important key of WSNs is detection and coverage of target area which is provided by random deployment. This paper reviews and addresses various area detection and coverage problems in sensor network. This paper organizes many scenarios for applying sensor node movement for improving network coverage based on bioinspired evolutionary algorithm and explains the concern and objective of controlling sensor node coverage. We discuss area coverage and target detection model by evolutionary algorithm.

1. Introduction

Wireless sensor network (WSN) has drawn a lot of attention in recent years. Developments of wireless sensor network enable them to operate with lower cost, lower power consumption, simpler computation, and better sensing of area when sensors move around. Furthermore, sensors also can sense the environment behind the movement, compute the data, and send the collected data to the sink node that can route the data to the other analyzing center through the internet [1].

Wireless sensor network has potential in many applications, such as healthcare, environment, industry, and environment monitoring surveillance in military, wildlife monitoring, and battle field. For instance, sensor network can be deployed in the environment for monitoring and controlling of plants and animal behavior [2] or in the ocean for controlling of temperature and seismic activities. However, in many places that are hostile, manual deployment is impossible and nodes have to be deployed randomly [3, 4].

The main problem in the wireless sensor network is deployment, coverage, and mobility strategy of sensor node; however, the coverage problem depends on a deployment sensor node in the wireless sensor network. There are

some optimization methods which grow exponentially as the problem size increases. Therefore, an optimization technique that requires appropriate memory and computational process and yet produces great results is favorable, especially for implementation on sensor node. Bioinspired optimization techniques are computationally efficient alternatives to traditional analytical techniques.

Deployment of the sensor nodes can be placed randomly in a target area. When network size is large and sensor field is hostile, the only choice for deployment of nodes is to scatter with aircraft. However, when sensor nodes are scattered randomly, it is difficult to find best strategy for random deployment that could minimize the coverage hole and communication overhead. Minimizing of the coverage hole can improve the quality of service for sensor network [5, 6].

Recently, mobile sensor node has great impact on network coverage. They are equipped with vehicle and move around the area after random deployment to enhance network coverage. However, mobile sensor node is very expensive in comparison to the stationary node. It has maximum utility to increase the network coverage and lifetime and provide fault tolerance and quality service for network. The key objective for mobile node is to cover all area in

the network and ensure each position has at least one sensor node for coverage. According to the monitoring area, three types of coverage have been identified: area coverage, target coverage, and barrier coverage. The mobile sensor node moves to exact location and connects to the other sensor node to form path coverage.

This paper presents how the mobility control can increase the coverage. In Section 2, we describe the model related to the sensing, coverage, and connectivity. Section 3 describes the evolutionary algorithms for optimization coverage. In Section 4 the classification of mobility exploited coverage is described. In particular, a concept of the coverage holes is explained in detail. Section 5 describes the dynamic optimization coverage using evolutionary algorithm with mobility to improve the network coverage. Section 6 summarizes our contributions and research challenge in this open area.

2. Coverage Criteria

One of the fundamental issues in the wireless sensor network is achieving optimum coverage. The goal of optimum coverage in physical space is sensing area within sensing range covered at least with one sensor node. There are different criteria for the design coverage scheme based on different objective and application in WSN. This section is reviewing several models that have effect in WSN network.

2.1. Randomly Deployment Sensor Node. The main problem in the wireless sensor network is deployment of the sensor node; however, the coverage problem depends on a deployment sensor node in the wireless sensor network. Deployment of the sensor nodes can be placed randomly in a sensor field that is the only choice scattered with aircraft for deployment when network size is large and sensor field is hostile. However, randomly deployment where sensor nodes are scattered within the field (such as continues or grid) environment probability and probability from the aircraft exceptionality is needed [7].

2.2. Sensing Detection Model. There are two types of a sensor detection model: one is a unit-disk-based model and the other one is non-unit-disk-based model. Unit-disk-based model has fixed sensing range, which sensor node able to sense the environment inside the disk range. When using sensor network as a unit graph, it is reasonable that the connectivity information of a graph contains sufficient information. Non-unit-disk based model has probabilistic sensing range, which sensing range is less than the distance.

2.3. Coverage Type. The key objective for mobile node is to maximize coverage in the network and ensure that area has at least one sensor node for coverage. According to the network to be covered, three types of coverage have been identified: area, target coverage, and barrier coverage. Area coverage addresses the problem of maximizing the detection rate in all spaces of sensing area with sensor node movement. Target coverage, on the other hand, moves a number of nodes to the specific point with the exact location for full coverage.

The main concern of barrier coverage is about finding the point in the path after deployment. Furthermore, mobile sensor node moves to exact location and connects to the other sensor node in the barrier coverage.

2.4. Fitness Function. Fitness function is a specific type of function that measures the optimality of a solution in evolutionary algorithm. Depending on the goals of the research, fitness function could be designed differently: single objective fitness function and multiobjective fitness function. In single objective fitness function, just one parameter for measuring the quality of the objective is used. With more than one objective, this entire independent objective should be combined, and this interaction is usually called hypostasis. All of above mentioned consist of objective about evaluating the solution in an evaluation algorithm.

2.5. Sensor Mobility. Sensor network used mobile node to enhance the coverage area. However, random deployment is not able to guarantee full coverage where a node is not in the exact position of sensing area. But, there are some mobility strategy to relocated sensor node in the exact position of sensing area after random deployment for improved network coverage. Mobility performance of the sensor has great effect on the sensor network to improve network QoS [8].

3. Evolutionary Algorithm

Evolutionary algorithms (EAs) are inspired from natural evaluation that helps to find optimum strategy for solving problem. The EAs continue a group of potential solutions to a problem. Hence, EAs use operator to create favorable potential solution. This operation is based on their optimal solution for problem. The EAs use this processes constantly to generate new population for optimal solution.

3.1. Particle Swarm Optimization. Particle swarm optimization (PSO) is relative of calculative models for outstanding by evolution. The purpose of PSO is responsibility to the best compound for problem under the presumption. PSO by Kennedy and Eberhart was first intended for simulating the social behavior. A difficulty optimized with the PSO is population of candidate solution. PSO model tagged particles and active these particles nearby in the search space similar to be common arithmetical formulae. Some fundamental concepts of particle swarm optimization that will be applied in bioinspired evaluation are described in the literature [9, 10]. The pursuits of the particle is aimed for establishing of best positions in the search space [11]. The particle swarm optimization for the d th dimension of position and velocity of i th particle is presented with following equation:

$$V(t+1) = \omega \cdot v(t) + C1 \cdot (ld(t) - xd(t)) + C2 \cdot (gd(t) - xd(t)), \quad (1)$$

$$xd(t+1) = xd(t) + vd(t+1), \quad (2)$$

where $v(t)$ is velocity for particle i , $xd(t)$ is the distance to be moved by this particle from its current position, $xd(t)$ is

the current particle location, $ld(t)$ is its best previous local location, and $gd(t)$ is the best global position. C_1 and C_2 are positive constant parameters called acceleration coefficients. The inertia weigh, ω , is a user-specified parameter that controls, with C_1 and C_2 , the impact of previous historical principles of particle velocities is based on its present one [12].

3.2. Genetic Algorithm. Genetic algorithm is a family of computational models inspired by evolution. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures so as to preserve critical information. Some general steps are required to solve a problem with GA. Amongst them, there is a main component that is problem dependent, and it is chromosome design. Chromosome is a set of string, which consists of all the genes, and indicates a solution to the problem. Each string is sometimes referred to as a genotype or alternatively a chromosome [13]. Although at first chromosomes are generated randomly and they could not be the good answers, during each generation the overall fitness of them would be increased [14]. In the literature [15–17] are some basic concepts of genetic algorithm application that will be applied in bioinspired computation.

4. Mobility Exploited Coverage Classification

Many researchers have been able to develop mobility schemes for improving network coverage with high QoS. Based on deployment objective, mobility can generally classified into three major categories: repair the coverage hole, optimizing coverage, and event based coverage [8].

4.1. Repairing Coverage Hole. Coverage hole (some spots are not covered) may be happening when some mobile node is not located in the exact position after deployment. The main objective of using sensor node is to repair the coverage hole in sensing area with redeployment of sensor network [5].

4.2. Optimizing Coverage. Optimizing coverage objective is leverage mobility to reduce the node overlap and maximize the network coverage. In a random deployment network, some node in the area has overlaps with the other sensor node. Hence, mobile sensor node can move around sensing area and adjust their position in order to optimize the network coverage [6].

4.3. Event Based Coverage. The goal of event base coverage is improving the target coverage by using mobile nodes. Event coverage has limited lifetime and does not need to be longer coverage [18].

5. Optimization Coverage

The difficulty in the wireless sensor network is coverage; however, the coverage problem turns on the coverage model in the wireless sensor network. Coverage model can be vouching for the quality of service of sensing area and allocated in the large diversity of the application. In this section, we introduce

how to estimate the network coverage hole and optimize the coverage area. Set of sensor nodes deployed in sensing area and coverage-estimate problem is to determine if all point in target area have k -coverage, where each point is at least covered with one sensor node. Optimization coverage problem is mostly studied in coverage optimization problem, while it also emphasizes the network lifetime and balanced energy consumption for k -coverage of sensor network with minimum mobility of sensor node.

Authors in [2] proposed P-BEEG algorithm which prolonged survival time of the sensor network, but all cluster heads can communicate with a base station, and the nodes are stationary in P-BEEG approach. The deployment method of sensors and optimizing movement strategy are developed for MSN, which taken into account the important issues and the key parameters about affecting energy consumption of nodes and maximizing the network lifetime in mobile sensor networks.

The authors in [19] proposed Optimization Movement control (OMC), for controlling of movement strategy in mobile sensor network. In the first deployment, S mobile node is randomly deployed and F node will be deployed in the rectangle grid with communication radius R ; the hop of communication between F and S is one. The proposed algorithm works based on the evolutionary algorithms (SPSO) for optimal coverage in the mobile sensor network. The aim of this algorithm is to find particle position which is based on evaluation fitness function. The evaluation fitness function used simulated annealing (SA) and PSO. Simulated annealing is the combinational optimization and can help PSO algorithm to get high rate convergence and success in search space (SPSO). The SPSO is used for movement strategy; the algorithm accepted the new criteria that it helps fitness function to become worse in a limited area instead of extra criteria to accept the new optimal solution. The new criteria Δf calculated the new particle position between two fitness functions where $\Delta f < \epsilon$. In evolutionary algorithm, fitness function has great effect in mobile sensor network. Also the author makes some improvement in PSO algorithm where velocity has magnitude director, which velocity have X -velocity and Y -velocity based on the following equation:

$$v_x(t+1) = \omega \cdot v(t) + C1 \cdot (ld(t) - X_x d(t)) + C2 \cdot (gd(t) - X_x d(t)), \quad (3)$$

$$v_x(t+1) = X_x d(t) + V_x d(t+1), \quad (4)$$

$$V_y(t+1) = \omega \cdot v(t) + C1 \cdot (ld(t) - X_y d(t)) + C2 \cdot (gd(t) - X_y d(t)), \quad (5)$$

$$V_y d(t+1) = X_y d(t) + V_y d(t+1). \quad (6)$$

When nodes are deploying in monitoring area, the value calculated by (4) and (6) is not able to map to the corresponding sensor node. For corresponding sensor node, they use fitness function. The fitness function requires the following issue: first, consider additional energy; second, calculate the neighbor energy and then consider the surplus

and the consumption of energy. As illustrated on the three considerations, the author used new fitness function to make reasonable model as follows:

$$f(x) = a_1 E_c + a_2 \frac{e_{ave}}{E_{max} - E_c} + a_3 \frac{1}{n-1} \times \sum_{i=1, i \neq j}^n E_{c(i)} * \frac{1}{(2R_i)^2 + 1}, \quad (7)$$

where $a_1 + a_2 + a_3 = 1$, a_1 , a_2 and a_3 are impact factor of node and their neighbor E_c is energy for current node and $E_{c(i)}$ is equivalent energy for neighbor node. According to SPSO algorithm for movement control, at first, they initiate the stage to collect the statistic of F node position and then, based on fitness function, to calculate the speed and change the position of F mobile sensor node. Therefore, SPSO and rectangle grid for mobile sensor node guaranteed fundamental network topology and can improve network coverage.

Deployment is one issue in wireless sensor (WSN) while network consists of stationary and mobile node allows sensor network to enhance coverage by self-organization technique. Author in [20] proposed the parallel particle swarm optimization (PPSO) to enhance the coverage for large area. The mobile node will use PPSO to relocate them to find optimal deployment in large area for various coverage optimizations. Actually, mobile node deployment based PPSO is appropriate for finding optimum solution in contentious area which some position needs cooperative and dynamically can change their position based on environment requirement. They assumed that all nodes know their position and use detection range r_d , dependability r_t , and communication range. When area is in sensing range of n sensor node at time t , the area detection dependability can be computed as

$$R(t) = 1 - \prod_{i=1}^n (1 - r_i(t)), \quad (8)$$

where $r_i(t)$ is the dependability of i th sensor nodes.

After sensor node deployment based PSO in position $X_i = x_{i1}, x_{i2}, \dots, x_{i3}$, coordinates of all mobile sensor nodes and associative objective are presented by detection of environment. The velocity of particle adjusts the granularity for tradeoff between speed and precision should be randomly change and calculate the local best and global best fitness correlated with new granularity for utiliz validity based on PSO equation. However, each node has minimum ability for huge computation based PSO. Therefore, authors used PPSO for deployment optimization which divided all detecting environment in n group and each group includes some intelligence sensor node. In random deployment, the coverage hole of each part is not equal, then mobile node is divided into n parts as $s_i = (s_i / \sum s) * N$, where s_i is coverage hole and N is the sum of mobile sensor nodes. The sensor nodes are considered about neighbor node during optimization, because mobile node is intelligent and performs optimization independently. So, if the distance is less than detection range, the mobile node should be redeployed in monitoring area.

Furthermore, PPSO algorithm has great effect in deployment optimization which can improve network coverage and connectivity performance in wireless sensor network according to detection ranges and the position of node. However, computational time of particle swarm optimization (PSO) will increase exponentially as the search space increases.

The main issue of coverage and target detection in wireless sensor network are dynamic deployment in wireless sensor network. The main problem for coverage balancing between local and global best position is coefficient for current speed of particle in next steps. Authors in [21] proposed three dynamic PSO algorithms that decrease the computation cost in sensor coverage. The first approach is PSO-LA which is used Learning Automata to adjust the searching method to continue the current route for particle. The proposed algorithm suppose mobile node equipped with Learning Automata that has two exploits, which will flow the best and continue your way. Choosing the flow as the best action means the use of best experience and team experience which has great effect on next iteration current particle velocity and present particle velocity is unnoticed. In this case, they use PSO equation for velocity and position update for particle. On the other hand, continue-your-way action has great effect on global search in unknown search space. In this algorithm, the mobile sensor node with minimum repetition and relocation moves around. In the next algorithm, PSO-LA and Learning Automata are guides each particle to move around search space. Allocation of learning automates helps each particle to make decision for moving around without considering the other particle. Then, Learning Automata uses two actions based on PSO algorithm for best global position and moves in search space with current velocity in the right way. In this algorithm, each particle uses previous phase to determine its action with minimum energy consumption. For both previous algorithms, that authors have proposed based on PSO algorithm, cyclic and zigzag movement were the major problem in long distance movement. However, cyclic and zigzag movement are the major problem based on PSO-LA algorithm in long movement. They use high energy consumption and computation time. To solve this problem, PSO-LA algorithm with logical movement, same as PSO-LA algorithm just in last step, has not zigzag movement. Therefore, logical movement method has best local and global search with high convergence rate and increases the network coverage and lifetime with minimum number of movements. In PSO-LA approach, PSO and Learning Automata are hybridized where velocity of particles is corrected by using the existing knowledge and the feedback from the real implementation of the approach. To improve the performance of the PSO-LA, improved PSO-LA approach is proposed, movement strategy of a node without an impact from the movement of other mobile nodes and based on the result learned from its previous step movement. In the third one, Improved PSO-LA with logical movement, sensors virtually change new positions by computing their target areas with the same procedure of the improved PSO-LA, but the real strategy movement of the sensor nodes only happens at the final round after last destinations are defined.

The goal of existing algorithms is best deployment, where coverage guaranteed the quality of service of the WSN. The PSO is responsible for maximizing coverage under the presumption as set of role. An adjustment problem includes a fitness function delineating in problem. The approach in [22] proposed coverage optimization based PSO and Voronoi diagram. Voronoi diagram is very useful in sampling model for coverage hole. Furthermore, this model can calculate coverage hole based on particle encoded. Voronoi diagram can be used for WSN deployment for N sensor s_1, s_2, \dots, s_N , and put sensor nodes act as the site. The measure of a coverage holes is requires set of point. These sets of points include the voronoi diagram and have distributed point on the boundary of a polygon. A particle is encoded a final explanation which represents of the best location of sensor nodes. The location of sensor i is made clear by two coordinators (x_i, y_i) . Particle is consider of an N sensor nodes and can determine the sensor nodes best location. The fitness function objective is minimizing coverage holes. The voronoi diagram measures the coverage hole as set of interest points. The interest point is vertex of the voronoi polygon which determines as voronoi diagram and number of spot consistently in a boundary of voronoi polygons. These interest points can help sensors to meet each other in the region of interest by pulling force. The proposed algorithm estimated the coverage hole as follow. Firstly, compute the interest point and distance of each nearby sensor node. After the calculation of the distance if distance (d) is larger than the sensing range (r_s), then its shows there is a coverage hole around the interest point. Thus, the optimum coverage is a total coverage hole in interest of the region. The complexity computation of this fitness objective depends on the number of sensor nodes and the size of the grid. PSO-Voronoi algorithm helps to optimize the coverage with sensible computational time. The complexity computation of this fitness objective depends on the number of sensor nodes and size of the grid. PSO-Voronoi algorithm improves network coverage but ignores the time complexity of defining Voronoi polygons.

Multiobjective Genetic Algorithm (GA) is used to examine the optimization of WSN layout which are considered two competing objectives. Two competing objectives are consist of overall sensor coverage and the lifetime of the network [23]. During the coverage operational time, sensors move to form a uniformly distribute based on the execution of the approach at a destination. However, the computation of this algorithm is not minimum. Authors in [24] applied particle swarm optimization (PSO) algorithm to increase the 1-coverage in mobile sensor networks and to minimize cost by finding the optimum positions for cluster head based on a well-known energy model.

The aforementioned algorithms mainly consider 1-coverage optimization, which each point in monitoring region is covered by at least 1 sensor node. However, 1-coverage optimization is not capable of providing a uniform sensor distribution over the monitoring region. Any point in monitoring regain can be covered by k -coverage ($k > 1$) which k can be set of sensor nodes and the area of the monitored region. k -coverage can improve the network performance, which is propitious to the maximum possible utilization of

the available nodes and balancing the node energy consumption.

Finding the best position for sensor node is a favorable use of availability of the sensor nodes, increasing network lifetime and stability of sensor energy. In [25] authors proposed optimization deployment for k -coverage with minimum mobility. Hence, the random deployment does not guarantee full coverage and there is some vacancy. In [25] first nodes are randomly deployed, then the sensor nodes analyze the coverage hole with the following: k -coverage, coverage hole, and coverage vacancy. k -coverage where any point of sensing area should at least coverage with $k > 1$ sensor node (k is the number of sensors). Coverage hole where the monitoring area i , not monitored by any sensor node. Coverage vacancy where the monitored area i is covered by $n_i < k$ sensor. At the end, algorithms calculated the total uncovered area. In such proposed algorithm there are $n = \lambda S$ homogeneous sensor in sensing area and each sensor used-unit-disk based covered πr^2 . Based on randomly distribution, vacancy density λ is $\sqrt{k}/r^2 \sqrt{2\pi^3}$ and the number of vacancy in M sensing area used by λs as $S \rightarrow \infty$ combine with $\lambda \pi^2$. The authors applied PSO algorithm for optimization deployment with minimum mobility. The PSO algorithm determined each particle in the d -dimensional space as $X_i = (x_{i1}, x_{i2}, x_{i3}, \dots, x_{id})$ where i represents a number of particles and d is the dimension and the previous best position of particle is $P_i = (p_{i1}, p_{i2}, p_{i3}, \dots, p_{id})$ and velocity among the search space is $V_i = (v_{i1}, v_{i2}, v_{i3}, \dots, v_{id})$ and best position of particles can be updates by (1) and (2). The proposed algorithm shows reduced distance movement significantly with the unlimited model and the PSO algorithm. Also it can help to get high convergence rate and increase scalability of sensor network.

The coverage and lifetime are two important issues in mobile sensor network that ensure high quality service for sensor network. Authors in [26] proposed PSO algorithm for improving the coverage with maximum movement of sensor node. Finding optimal position is executed by updating particle velocity and position based on (1) and (2). The proposed algorithm used PSO method to find optimal position according to penalty based on fitness objective. The penalty factor can be found with fuzzy penalty. The algorithm uses voronoi diagram to get best coverage and time efficiency. Voronoi diagram is specified beforehand and for each sites there will be a corresponding region consisting of all points closer to that site than to any other. They used fitness function to evaluate the solution to encode coverage problem in particle. The objective is to minimize the coverage hole in wireless sensor network. The coverage hole is calculated by using voronoi diagram, which the sensor act as sites, if all polygon vertexes are covered by sensor node, then the region of interest is fully covered. Otherwise, coverage hole exists in sensing area. Therefore the fitness function of coverage hole can be calculated based on (9):

$$\text{minimize : } \sum \text{coverage_hole} \quad (9)$$

$$\text{subject to } d_{\text{mov}} \geq D_{\text{max}}, \quad (10)$$

where coverage gap is the place not fully covered by sensor node around the target area and d_{mov} is the maximum distance moved by sensor and D_{max} is maximum distance that a sensor is allowed to move. Hence (2) can be rewritten with penalty function as follows:

$$\begin{aligned} \text{minimize : } & \sum \text{coverage_hole size} + \gamma P(d_{\text{mov}}) \\ \text{subject to } & D_{\text{mov}} \geq D_{\text{max}}, \end{aligned} \quad (11)$$

where γ is the value action variable and $p_{(d_{\text{mov}})}$ is penalty function and suitable $p_{(d_{\text{mov}})}$ function as n:

$$p_{(d_{\text{mov}})} = \max(0, (p_{(d_{\text{mov}})} - D_{\text{max}})). \quad (12)$$

$p_{(d_{\text{mov}})}$ is equal to zero as long as the limitation is submitted, but when the constraint is disobeyed, $p_{(d_{\text{mov}})}$ is equal to some positive value. The fuzzy system uses penalty parameter, γ , based on the d_{mov} value. The following equation is determined to return the value of γ :

$$\gamma = \exp^a, \quad (13)$$

where

$$a = \begin{cases} 0 & \text{if } d_{\text{mov}} < D_{\text{max}}, \\ \frac{d_{\text{mov}} - D_{\text{max}}}{\Delta} & \text{if } D_{\text{max}} < d_{\text{mov}} < D_{\text{max}} + \Delta, \\ 2 \left(\frac{d_{\text{mov}} - (D_{\text{max}} + \Delta)}{D_{\text{ROI}} - (D_{\text{max}} + \Delta)} \right) & \text{if } D_{\text{max}} + \Delta < d_{\text{mov}} < D_{\text{ROI}}. \end{cases} \quad (14)$$

In proposed algorithm, first, sensor deployed is randomly in two-dimensional area and all sensors have similar sensing range when sensor node is homogeneous. In every step, maximum distance movement by sensor passed to the fuzzy system to calculate the new value of penalty parameter γ and the value passed to the PSO for fitness objective. This condition continues when one stopping condition happens. The proposed WSNPSO_{con} algorithm as method based on PSO improves sensor network coverage and minimize energy consumption but the complexity of computational for voronoi polygon is huge.

The approach in [27] works based on particle swarm optimization (PSO) to solve the movement coverage problem. The main objective in movement strategy is to decrease the distance between the neighboring nodes, thus increasing coverage in the network. The proposed algorithm does not consider the stationary nodes which are not able to change their initial positions. However, to minimize energy consumption and to decrease cost, stationary nodes are widely used in real applications.

Wireless sensor node is randomly deployed in grid filed. For evaluation of sensor deployment, sensor field can be two-dimensional grid and use probabilistic detection model. Dynamic deployment provides coverage and target detection for wireless sensor network. The proposed dynamic deployment algorithm is "with virtual force directed coevolutionary particle swarm optimization" (VFPCSO). The proposed

algorithm use coevolutionary algorithm for dynamic deployment, because the PSO algorithm uses particle in search space to find optimal position. However, it is difficult for PSO to find solution in large search space and it also has some disadvantage; when some particles are closer to the optimal positions the other particles move away from the best position. In VFPCSO, best deployment of global search is accomplished by the hybrid CPSO algorithm for improving network deployment. At first, initialize the swarm as Q in n -dimensional, $Q \cdot x_k$ is recent location of particle k , $Q \cdot y_k$ is optimal local position of particle k , and $Q \cdot \bar{y}$ is global optimal position of particle. And then, calculate the coverage area based on $f(b(k, p_k \cdot x_i))$. After evaluating the effective coverage area, by using (15) calculate the attractive and repulsive virtual force between sensor nodes. Perform PSO update on p_k using (1) and (2) and compute the virtual force of i th particle in j th dimension with the following equation:

$$g_{ij} = \begin{cases} \frac{F_x^{(i,j/2)}}{F_{xy}^{(i,j/2)}} * \text{Max step} * e^{F_{xy}^{-(1/(i,j/2))}}, & j = 1, 3, 5, \dots, 2n - 1, \\ \frac{F_{xy}^{(i,j/2)}}{F_{xy}^{(i,j/2)}} * \text{Max step} * e^{F_{xy}^{-(1/(i,j/2))}}, & j = 2, 4, 6, \dots, 2n, \end{cases} \quad (15)$$

where the superscript of each factor is the index of sensor and index of particle which is virtual force using coordinate of virtual force. In proposed algorithm the potential processing capability of multiple nodes may contribute to best optimization performance. However, for WSNs, the energy efficiency should be taken into account in the deployment.

The approach in [28] proposed 2.5D and studied PSO based coverage optimization for WSNs on digital elevation models (DEMs). To compute network coverage on DEMs, a method of computing individual sensor node coverage is introduced. The authors also proposed an improved algorithm based on dissipative particle swarm optimization (DPSO). The basic steps of PSO based coverage optimization are as follows.

Step 1. Randomly initialize the speed and position of each particle. The range of speed is $[-m/2, m/2]^n$ and the range of position is $[0, m]^n$. Compute the fitness value of each particle using (1). Set the position of each particle as its best position p_i and set the position of the particle having the best fitness as the group best position p_g .

Step 2. Update the speed and position of each particle using (1) and (2).

Step 3. Compute the fitness value of each particle.

Step 4. Compare the fitness value of each particle with the fitness value of its best position p_i . Set the current position of the particle as p_i if it has better fitness.

Step 5. Compare the fitness value of each particle with the fitness value of the group best position p_g . Set the current position of the particle as p_g if it has better fitness.

Step 6. Stop if the maximum number of generations G_{\max} is reached and the optimized deployment is represented by p_g ; otherwise return to Step 2 and continue.

If the sensing radius of each node is r , then computing the coverage of a node requires $O(r^2)$ time and computing the coverage of n nodes requires $O(nr^2)$ time. It requires $O(n)$ time to update the speed and position of a particle using (1) and (2) and $O(n)$ time to mutate its position. If there are p particles, then it takes $O(pnr^2)$ time to update them and compute their fitness values in each generation of the algorithm. Therefore, the time complexity of the algorithm is $O(G_{\max} pnr^2)$. Authors introduced better sensor coverage with significantly minimum computational effort. The method involves significant energy consumption in broadcasting initial and final positions. It also necessitates algorithms for collision avoidance and localization.

6. Concluding Remarks

Scale and density of deployment and constraints in battery, storage device, bandwidth, and computational resources pose serious challenges to the developers of WSNs. Issues of the node deployment, coverage, and mobility are often formulated as optimization problems. Most optimization techniques suffer from slow or weak convergence to the optimal solutions. This calls for high performance optimization methods that produce high quality solutions by using minimum resources. Bioinspired algorithm has been a popular method applied to solve optimization problems in WSNs due to its simplicity, best solution, fast convergence, and minimum computational complexity. However, nature of bioinspired algorithm can forbid its use for real-time applications which need high speed, especially if optimization require to be carried out mostly. Bioinspired algorithms require large sizes of storage device, which may limit their implementation for resource. Literature has numerous successful WSN applications that utilized advantages of bioinspired algorithms.

In this paper, we surveyed recent contributions to the problem of improving network coverage by evolutionary algorithm. Network coverage is an important performance metric for various applications in WSNs. The concept of mobility as it can be used for wireless sensor networks is improving network coverage. However, traditional approach is used stationary node for improve network coverage based on schedule for control of activity in a best way. Hence, by control of mobile node the network coverage can significantly improve for wide performance application.

This paper organizes many scenarios for applying sensor node movement for improving network coverage based on evolutionary algorithm and explains the concern and objective of controlling sensor node coverage. However, there are many kinds of coverage control algorithms that have been proposed for different coverage based on different sensing models. These new sensing models depend on more than one sensor node and also this new model require to call new node mobility control. Also there is another feature for node mobility and this objective is not only to improve network

coverage but also to increase network lifetime and enhance data details timeline and reliability at the same time. Some future research work may take into account how to minimize energy consumption for those coverage's with holes and how to control sensor node movement strategy to heal network coverage and improve network lifetime. Furthermore, other issues such as an energy consumption model about mobile nodes and their moving strategy need to be taken into account in developing movement strategies.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] I. F. Akylidiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.
- [2] A. Ghosh and S. K. Das, "Coverage and connectivity issues in wireless sensor networks: a survey," *Pervasive and Mobile Computing*, vol. 4, no. 3, pp. 303–334, 2008.
- [3] G. J. Pottie, "Wireless sensor networks," in *Proceedings of the Information Theory Workshop*, pp. 139–140, June 1998.
- [4] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.
- [5] B. Wang, H. B. Lim, and D. Ma, "A survey of movement strategies for improving network coverage in wireless sensor networks," *Computer Communications*, vol. 32, no. 13-14, pp. 1427–1436, 2009.
- [6] C.-F. Huang and Y.-C. Tseng, "A survey of solutions to the coverage problems in wireless sensor networks," *Journal of Internet Technology*, vol. 6, no. 1, pp. 1–8, 2005.
- [7] H. Zhang and J. Hou, "On deriving the upper bound of a lifetime for large sensor networks," in *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MoBiHoc '04)*, pp. 121–132, May 2004.
- [8] Y.-C. Tseng, Y.-C. Wang, K.-Y. Cheng, and Y.-Y. Hsieh, "iMouse: an integrated mobile surveillance and wireless sensor system," *IEEE Computer*, vol. 40, no. 6, pp. 76–82, 2007.
- [9] X. S. Yang, Z. H. Cui, R. B. Xiao, A. H. Gandomi, and M. Karamanoglu, *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, Elsevier, Waltham, Mass, USA, 2013.
- [10] A. H. Gandomi, G. J. Yun, X. S. Yang, and S. Talatahari, "Chaos-enhanced accelerated particle swarm optimization," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 2, pp. 327–340, 2013.
- [11] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*, Kafman Morgan, Los Altos, Calif, USA, 2001.
- [12] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC' 98)*, pp. 69–73, Anchorage, Ala, USA, May 1998.
- [13] Lawrence, *Handbook of Genetic Algorithms.*, Van Nostr and Reinhold, New York, NY, USA, 1996.
- [14] M. Davoodi, M. Mohades, and J. Rezaei, *A Genetic Algorithm for the Constrained Coverage Problem*, vol. 58 of *Applications of Soft Computing*, Springer, 2009.

- [15] S. Zeng, Z. Liu, C. Li, Q. Zhang, and W. Wang, "An evolutionary algorithm and its application in antenna design," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 2, pp. 129–137, 2012.
- [16] M. P. Poland, C. D. Nugent, H. Wang, and L. M. Chen, "Genetic algorithm and pure random search for exosensor distribution optimisation," *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 359–372, 2012.
- [17] J. Muñuzuri, P. C. Achedad, M. Rodríguez, and R. Grosso, "Use of a genetic algorithm for building efficient choice designs," *International Journal of Bio-Inspired Computation*, vol. 4, no. 1, pp. 27–32, 2012.
- [18] M. Cardei and J. Wu, "Energy-efficient coverage problems in wireless ad-hoc sensor networks," *Computer Communications*, vol. 29, no. 4, pp. 413–420, 2006.
- [19] W.-W. Huang, C.-W. Zou, M.-X. Deng, and M. Yu, "An optimizing movement control strategy for mobile sensor networks," in *Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '09)*, September 2009.
- [20] X. Wang, S. Wang, and J. Ma, "Dynamic deployment optimization in wireless sensor networks," *Lecture Notes in Control and Information Sciences*, vol. 344, pp. 182–187, 2006.
- [21] R. Soleimanzadeh, B. J. . Farahani, and M. Fathy, "PSO based deployment algorithms in hybrid sensor networks," vol. 10, no. 7, pp. 167–171.
- [22] N. A. B. Ab Aziz, A. W. Mohemmed, and M. Y. Alias, "A wireless sensor network coverage optimization algorithm based on particle swarm optimization and voronoi diagram," in *Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC '09)*, pp. 602–607, Okayama, Japan, March 2009.
- [23] D. B. Jourdan and O. L. De Weck, "Layout optimization for a wireless sensor network using a multi-objective genetic algorithm," in *Proceedings of the IEEE 59th Vehicular Technology Conference (VTC '04)*, pp. 2466–2470, Milan, Italy, May 2004.
- [24] X. Wu, S. Lei, W. Jin, J. Cho, and S. Lee, "Energy-efficient deployment of mobile sensor networks by PSO," in *Proceedings of the International Workshop on Sensor Networks (IWSN '06)*, pp. 373–382, Harbin, China, January 2006.
- [25] X. Bai, S. Li, and J. Xu, "Mobile sensor deployment optimization for k -Coverage in wireless sensor networks with a limited mobility model," *IETE Technical Review*, vol. 27, no. 2, pp. 124–137, 2010.
- [26] N. A. A. Aziz, A. W. Mohemmed, and M. Zhang, "Particle swarm optimization for coverage maximization and energy conservation in wireless sensor networks," in *Applications of Evolutionary Computation*, vol. 6025 of *Lecture Notes in Computer Science*, pp. 51–60, 2010.
- [27] N. Kukunuru, B. R. Thella, and R. L. Davuluri, "Sensor deployment using particle swarm optimization," *International Journal of Engineering Science*, vol. 2, pp. 5395–5401, 2010.
- [28] W. Li, "PSO based wireless sensor networks coverage optimization on DEMs," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, pp. 371–378, Springer, Berlin, Germany, 2012.

Research Article

A Multipopulation Coevolutionary Strategy for Multiobjective Immune Algorithm

Jiao Shi, Maoguo Gong, Wenping Ma, and Licheng Jiao

Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, Xidian University, Xi'an 710071, China

Correspondence should be addressed to Maoguo Gong; gong@ieee.org

Received 21 October 2013; Accepted 22 December 2013; Published 12 February 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 Jiao Shi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to maintain the population diversity is an important issue in designing a multiobjective evolutionary algorithm. This paper presents an enhanced nondominated neighbor-based immune algorithm in which a multipopulation coevolutionary strategy is introduced for improving the population diversity. In the proposed algorithm, subpopulations evolve independently; thus the unique characteristics of each subpopulation can be effectively maintained, and the diversity of the entire population is effectively increased. Besides, the dynamic information of multiple subpopulations is obtained with the help of the designed cooperation operator which reflects a mutually beneficial relationship among subpopulations. Subpopulations gain the opportunity to exchange information, thereby expanding the search range of the entire population. Subpopulations make use of the reference experience from each other, thereby improving the efficiency of evolutionary search. Compared with several state-of-the-art multiobjective evolutionary algorithms on well-known and frequently used multiobjective and many-objective problems, the proposed algorithm achieves comparable results in terms of convergence, diversity metrics, and running time on most test problems.

1. Introduction

Optimization problems widely exist in real life, especially in engineering applications [1–4]. The optimization problem with only one objective is called single-objective optimization problem. The optimization problem with more than one objective to be simultaneously solved is called multiobjective optimization problem (MOP). In practical optimization applications, there is a great demand for optimizing multiple objectives simultaneously. As a heuristic searching method, evolutionary computation has already been successfully used in the field of MOPs and gradually develops into a hot research direction, named evolutionary multiobjective optimization (EMO) [5–8]. The search technique based on population is proved to have a good ability of global searching and can find a set of solutions in one-shot operation. Thus, evolutionary computation achieves comparable results in solving nonconvex, nonlinear, discontinuous and differentiable problems and overcomes the deficiency of traditional mathematical programming [9–13].

The first study on multiobjective evolutionary algorithm (MOEA) is probably the vector evaluated genetic algorithm

(VEGA) [14]. Since then, MOEAs have obtained increasing attention, and the amount of literatures about MOEAs has increased in which many MOEAs were designed one after another, such as multiobjective genetic algorithm (MOGA) [15], niched Pareto genetic algorithm (NPGA) [16], and nondominated sorting genetic algorithm (NSGA) [17]. These algorithms are regarded as the typical representatives of the first generation of MOEAs which are characterized by using Pareto ranking-based selection and fitness sharing strategy [18]. The second generation of MOEAs are characterized by using elite strategy, including strength Pareto evolutionary algorithm (SPEA) [19], improved version of SPEA (SPEA2) [20], Pareto envelop-based selection algorithm (PESA) [21], niched Pareto genetic algorithm 2 (NPGA2) [22], and nondominated sorting genetic algorithm II (NSGA-II) [23]. Recently, researches on evolutionary multiobjective optimization present new characteristics. The concepts of simulated annealing [24], particle swarm [25], quantum [26, 27], and messiness [28] were proposed and introduced into the framework of evolutionary algorithms. At the same time, many new-type evolutionary mechanisms were introduced, including regularity-model-based multiobjective estimation

of distribution algorithm (RM-MEDA) [29] and multiobjective evolutionary algorithm based on decomposition (MOEA/D) [30].

The concept of artificial immune systems (AIS) was first put forward in 1996. Since then, AIS have stepped into a high-speed development period and become one of the hot topics in the field of artificial intelligence. AIS that get the inspiration from biological immune systems attempt to develop computational tools for solving science and engineering problems. Some AIS-based multiobjective optimization algorithms have been proposed [31–34], including immune optimization algorithm for constrained nonlinear multiobjective optimization problems [34], a hybrid immune multiobjective optimization algorithm [31], and chaos-based multiobjective immune algorithm [32]. Recently, a multiobjective immune algorithm with nondominated neighbor-based selection (NNIA) was proposed by Gong et al. [35]. From the comparison with representative algorithms, it is apparent that NNIA is an effective immune multiobjective algorithm in solving MOPs. Although the employment of elite strategy improves the convergence rate of MOEA, it leads to the loss of population diversity as well. Like the common problem existing in evolutionary algorithms, premature convergence also haunts NNIA. It may be trapped into local optimal solution, thus the population diversity of NNIA needs to be improved.

An enhanced version of nondominated neighbor-based immune algorithm with a multipopulation coevolutionary strategy is proposed for improving the population diversity. Subpopulations employ evolutionary operations independently; thus the unique characteristics of each subpopulation can be effectively maintained. During evolutionary search, information exchanges among subpopulations thus expanding the search range of the entire population. As a matter of fact, most of the evolutionary algorithms employ regular operations throughout the whole evolutionary process, and few of them take advantage of online discovered information. The adaptive operator which dynamically applies evolution operations to subpopulations based on the online discovered information is designed. Therefore, evolutionary search becomes more directional and purposeful and the unnecessary waste of computational cost is reduced.

The remainder of this paper is organized as follows. The problem statement is described in Section 2. Section 3 presents the proposed algorithm in detail. Section 4 presents experimental results, ZDT problems, DTLZ problems, and some extensional problems are adopt, and the sensitivity of the introduced parameter, the scalability of the proposed algorithm, and the comparison of running time are also investigated in this section. Finally, we outline the conclusions of this paper.

2. Problem Statement

The mathematical description of multiobjective optimization problems can be expressed as follows [36, 37]:

$$\min y = F(x) = (f_1(x), f_2(x), \dots, f_m(x))^T,$$

$$\text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \dots, q,$$

$$h_j(x) = 0, \quad j = 1, 2, \dots, p,$$

(1)

where $x = (x_1, x_2, \dots, x_n) \in X \subset R^n$ is a decision variable vector, X is the decision space, $F(x)$ is the set of objective functions to be optimized simultaneously, $g_i(x)$ defines the inequality constraint, and $h_j(x)$ defines the equality constraint. Based on these mathematical descriptions, several important definitions of multiobjective optimization problems are given as follows.

Definition 1 (feasible solution and feasible solution set). For a certain decision variable vector $x \in X$, if it satisfies both equality constraints and inequality constraints, then x can be called a feasible solution. The feasible solution set is made up of all the feasible solutions, which can be denoted as X_f , where $X_f \subseteq X$.

Definition 2 (Pareto domination). For any two feasible solutions, if and only if they satisfy condition (2), it is called that x_a dominates x_b , which can be denoted as $x_a > x_b$. Consider the following:

$$\forall i = 1, 2, \dots, m, \quad f_i(x_a) \leq f_i(x_b) \wedge \exists j = 1, 2, \dots, m, \\ f_j(x_a) < f_j(x_b). \quad (2)$$

Definition 3 (Pareto-optimal solution and Pareto-optimal set). For a certain feasible solution x^* , if and only if it satisfies the condition: $\neg \exists x \in X_f : x > x^*$, then x^* can be regarded as the Pareto-optimal solution. The Pareto-optimal set is made up of all the Pareto-optimal solutions in the decision space, which can be denoted as

$$P^* = \{x^* \mid \neg \exists x \in X_f : x > x^*\}. \quad (3)$$

Definition 4 (Pareto-optimal front). The corresponding image of the Pareto-optimal set in the objective space is called the Pareto-optimal front, which can be denoted as

$$PF = \{F(x^*) = (f_1(x^*), f_2(x^*), \dots, f_m(x^*))^T \mid x^* \in P^*\}. \quad (4)$$

In solving MOPs, it is expected that the set of nondominated solutions obtained by the proposed algorithm can well approximate the true Pareto-optimal front and the diversity of the solutions can be maximized.

3. Multipopulation Coevolution Multiobjective Immune Algorithm

Many new-type evolutionary methods have been introduced into the area of MOEAs. Immune-based algorithm is one of these late-model methods. Artificial immune systems (AIS) get the inspiration from biological immune systems. They have learnable, parallel, and distributed characteristics, therefore possessing an efficient information processing ability.

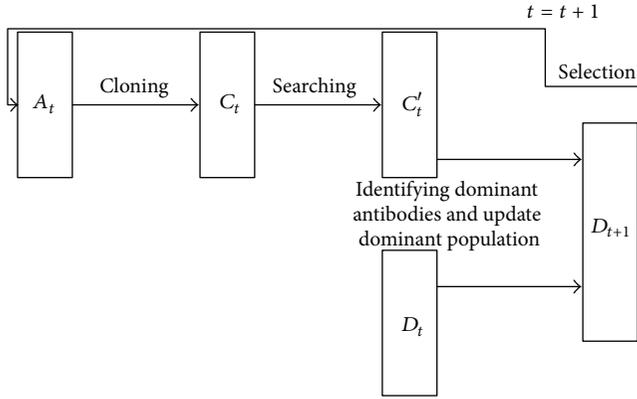


FIGURE 1: Population evolution of NNIA.

AIS-based algorithms have attracted a lot of attention and have been applied to many complex MOPs, including constrained nonlinear MOPs and dynamic MOPs [38, 39]. Recently, Gong et al. [35] presented a multiobjective immune algorithm with nondominated neighbor-based selection (NNIA), which is one of the representative immune-based multiobjective algorithms.

3.1. The Original NNIA. In NNIA, a nondominated neighbor-based selection and a crowding-distance-based proportional cloning were proposed. The fitness of a nondominated individual is assigned according to its crowding distance. The individual with greater crowding distance is reproduced more times and then less-crowded regions will have more chances to be searched, which improves the search ability of NNIA on less-crowded regions. Besides, only a minority of nondominated individuals will be selected to form an active population, and then a series of evolutionary operations are applied to this active population. Therefore, NNIA evolves very fast by performing evolutionary operations on a small-scale active population. The specific framework of population evolution in a single generation at time t in NNIA is shown in Figure 1.

From Figure 1, it is easy to observe that the evolutionary search in NNIA is very fast and effective due to its specific framework. Although such efficient mechanism achieves a high evolutionary rate, it introduces errors as well. Due to the efficient mechanism, population diversity is quickly decreased, and the resulting solutions may fall into local optimum, which is not a rare case. Under normal circumstances, the use of the elite strategy in MOEAs will lead to the loss of population diversity. The special evolutionary framework of NNIA further exacerbates this knotty problem. Table 1 shows the results of NNIA on ZDT2 and ZDT4, where NNIA performs 30 independent runs and the maximum size of the active population is 20. However, NNIA always obtains only one nondominated solution on ZDT2 and ZDT4 during 30 runs. The solutions obtained by NNIA are always trapped into local optimum on ZDT2 and ZDT4, which demonstrates the assertion of the analysis above.

TABLE 1: The situation of falling into local optimum.

Test problems	ZDT2	ZDT4
Times of falling into local optimum	12 times	10 times
The number of nondominated solutions	1	1

3.2. Description of the Proposed Algorithm. In this paper, we present an enhanced multipopulation coevolutionary strategy for nondominated neighbor-based immune algorithm, called CONNIA. Different from the traditional evolution, coevolution recognizes the simultaneous existence of competition and cooperation among populations, which provides a theoretical basis for maintaining the population diversity.

3.2.1. Adaptive Operator. When it comes to the adaptive operator in the field of MOEAs, it mainly refers to adaptively tuning some parameters, such as population size, crossover probability, and mutation probability. However, the adjustment of evolutionary strategy based on evolutionary conditions is seldom involved. The major contribution of the designed adaptive operator is that each subpopulation adaptively selects corresponding operators during the evolution, which makes the evolution become more purposeful and directional. Therefore, the need for unnecessary computing resource existing in random search is avoided effectively.

After performing a series of evolutionary operations on each subpopulation, a way for measuring the evolutionary condition is to identify the nondominated solutions of each subpopulation. The set coverage metric is employed for measuring the relationship between two subpopulations [40]. If a subpopulation has a higher value of the set coverage metric, it contributes more to the formation of the entire approximated Pareto front. The adaptive operator which consists of two different cases is designed on the basis of measuring the relationship between two subpopulations. Different evolutionary operators are designed for different cases. A threshold is introduced to decide which case is activated. The influence of the threshold on the performance is analyzed in Section 4.3.

Case 1. If the difference of the set coverage metric between subpopulations is not obvious, a local search operator would be employed. Two subpopulations perform independently evolutionary operations and search within different solution space for maintaining the diversity of the entire population. Meanwhile, some appropriate perturbations are applied around the obtained nondominated solutions for seeking the possible better solutions and reducing the probability of getting into local optimum.

Case 2. If the difference of the coverage metric between subpopulations is obvious, a cooperation operator would be employed. The information exchanges among subpopulations, which reflects a mutually beneficial relationship between two subpopulations. The subpopulation with lower value of the set coverage metric could make use of the reference experience from another subpopulation to improve its own evolution. Two subpopulations make progress together by

means of cooperation to ultimately complete the evolutionary task.

3.2.2. Local Search Operator and Cooperation Operator. We get the inspiration from traditional differential evolution (DE) operator [41] to design the local search operator and the cooperation operator. DE operator uses the differences between the structures of antibodies to guide the antibody variation and make the generated antibody closer to the optimal point.

Local Search Operator. Assume that P_t is a population, D_t is the nondominated population of P_t , and two individuals (x_1, x_2, \dots, x_n) and (y_1, y_2, \dots, y_n) are randomly selected from P_t and D_t , respectively. A new individual (z_1, z_2, \dots, z_n) is generated through the following operation:

$$z_i = y_i + U(-1, 1) * (y_i - x_i), \quad (5)$$

where $i = 1, 2, \dots, n$, $U(\cdot, \cdot)$ is a uniformly distributed random number. As we know, there may be some better solutions around the obtained Pareto-optimal solutions, particularly in the case that the obtained Pareto-optimal solutions are trapped into local optimum. The designed local search operator inflicts appropriate disturbances around the obtained Pareto-optimal solutions, and then the opportunity of finding some better solutions is increased. After the local search operation, a $(\mu + \lambda)$ selection strategy is adopted [42]. This elite strategy ensures the effectiveness of the local search operation and accelerates the rate of evolutionary search.

Cooperation Operator. Assume that there are two populations P_{t1} and P_{t2} . D_{t1} and D_{t2} are two nondominated populations of P_{t1} and P_{t2} , respectively. P_{t2} is better than P_{t1} in terms of the set coverage metric. Two individuals (x_1, x_2, \dots, x_n) and (y_1, y_2, \dots, y_n) are randomly selected from P_{t1} and D_{t2} , respectively. A new individual (z_1, z_2, \dots, z_n) is generated through the following operation:

$$z_i = y_i + U(-1, 1) * (y_i - x_i), \quad (6)$$

where $i = 1, 2, \dots, n$, $U(\cdot, \cdot)$ is a uniformly distributed random number. By applying the cooperation operator, subpopulations gain the opportunity to exchange information, thus expanding the search range of their own. The subpopulation with larger value of the set coverage metric may possess more effective convergence information. In this case, the subpopulation with lower value of the set coverage metric can improve its evolutionary capacity by gaining the experience from the outstanding antibodies in another subpopulation. This directed cooperation operator provides good evolutionary paths towards antibodies, thereby making antibodies evolve faster when compared with the noncooperation strategy.

The designed local search operator and cooperation operator reflect a mutually beneficial relationship between subpopulations. Both operators transmitting information among antibodies within the same generation are combined with traditional evolutionary operators such as crossover and mutation, for transmitting information effectively.

3.2.3. Multipopulation Coevolutionary Nondominated Neighbor-Based Immune Algorithm. The details of the proposed algorithm are described in this part. To be specific, the following parts are designed. (1) As each subpopulation evolves independently, the differences between subpopulations can be well kept. (2) By means of information exchange among subpopulations, the search range of each subpopulation can be effectively expanded. (3) The way of information exchange depends on the gap of the set coverage metric between subpopulations. Such online-decision strategy has an adaptive character, which improves the global search efficiency. The main steps of CONNIA are presented as follows.

Step 1. Generate two initial subpopulations P_{a0} and P_{b0} .

Step 2. The nondominated antibodies of the two subpopulations P_{at} and P_{bt} form two nondominated populations D_{at} and D_{bt} , respectively. Then the two nondominated populations are combined together to form the entire nondominated population T_t .

Step 3. If the terminal condition is satisfied, export T_t as the output. Stop; otherwise, $t = t + 1$.

Step 4. Select the individuals which have more contributions to the population diversity from D_{at} and D_{bt} , respectively. Then the selected individuals form two active populations A_{at} and A_{bt} .

Step 5. Two clone populations C_{at} and C_{bt} are formed by applying cloning to A_{at} and A_{bt} , respectively.

Step 6. Perform recombination and mutation on C_{at} and C_{bt} ; then obtain two resulting populations C'_{at} and C'_{bt} .

Step 7. If the condition of information exchange is satisfied, perform cooperation operator between C'_{at} and C'_{bt} . Otherwise, perform guided local search operator on C'_{at} and C'_{bt} , respectively. Then recalculate the nondominated solutions of C'_{at} and C'_{bt} , respectively.

Step 8. Get subpopulations P_{at} and P_{bt} by combining C'_{at} and D_{at} , C'_{bt} and D_{bt} , respectively; go to Step 2.

3.3. Solution Pruning Based on Crowding Distance. In the proposed algorithm, the crowding distance [23] is used to estimate the density around a solution and the contribution of a solution to the diversity of objective function values. The definition of the crowding distance is described as follows:

$$D(x) = \sum_{i=1}^k \frac{f_i(x') - f_i(x'')}{f_i^{\max} - f_i^{\min}}, \quad (7)$$

where f_i^{\max} and f_i^{\min} are the maximum and minimum values of the i th objective and k is the number of objective functions. $f_i(x')$ and $f_i(x'')$ are the values of the i th objective of the top two nearest points to x . If x is an extreme point, $D(x) = \infty$. Otherwise, the crowding distance of x is calculated by (7).

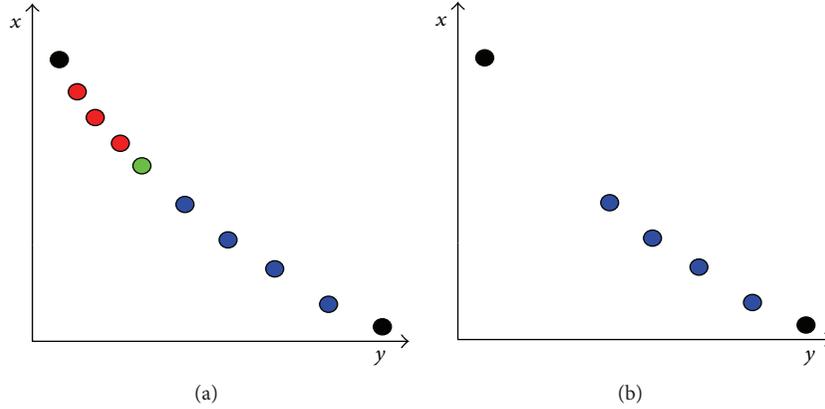


FIGURE 2: The static method of solution pruning.

The density around a dominant antibody is estimated by calculating its crowding distance. The larger the crowding distance of a dominant antibody is, the sparser the distribution around it will be, which also means that the contribution of this antibody to the population diversity is relatively greater. When it is required to delete some solutions, the antibody with small crowding distance will be deleted firstly. The traditional way of solution pruning is to calculate the crowding distance of all solutions only once, and then some solutions are deleted based on such one-shot result. However, such mechanism is unreasonable sometimes.

After calculating the crowding distance of all points shown in Figure 2(a), it is evident that two black extreme points have the largest crowding distances. In addition to black points, four blue points have larger crowding distances than other points. Points are sorted according to the crowding distance, from black points, blue points, green point, to red points in a decline order. Assume that four points need to be deleted, and then the red and green points are deleted by using the original static method. It is obvious that the points after pruning are not well-distributed as shown in Figure 2(b). It has been mentioned that the dynamic way is more reasonable than the traditional static method [43]. After deleting a point, recalculate the crowding distance of the remaining points and sort them based on the recalculated crowding distance.

3.4. Computational Complexity Analysis of CONNIA. Assume that the maximum size of the dominant population is n_d , the maximum size of the active population is n_a , and the size of the clone population is n_c . The time complexity for CONNIA in a single generation without information exchange can be calculated as follows.

The time complexity for identifying nondominated individuals in the population is $O((n_c + n_d)^2)$; the worst time complexity for dynamic selection is $\sum_{i=n_a}^{n_d} O(i \log i)$; the time complexity for cloning is $O(n_c)$; the worst time complexity for updating the dominant population is $\sum_{i=n_d}^{n_d+n_c} O(i \log i)$; and the time complexity for recombination and mutation is $O(n_c)$.

Therefore the worst total time complexity is:

$$\sum_{i=n_a}^{n_d} O(i \log i) + \sum_{i=n_d}^{n_d+n_c} O(i \log i) + 2O(n_c) + O((n_d + n_c)^2). \tag{8}$$

Owing to the fact that the operational rule of the symbol “O” can be simplified, the worst time complexity of one generation without information exchange for CONNIA can be written as: $O((n_c + n_d)^2)$.

The time complexity for CONNIA in a single generation with information exchange can be calculated as follows.

The time complexity for identifying nondominated individuals in the population is $O((n_c + 2n_d)^2)$; the worst time complexity for dynamic selection is $\sum_{i=n_a}^{n_d} O(i \log i)$; the time complexity for cloning is $O(n_c)$; the worst time complexity for updating the dominant population is $\sum_{i=2n_d}^{2n_d+n_c} O(i \log i)$; and the time complexity for recombination and mutation is $O(n_c)$.

So the worst total time complexity is:

$$\sum_{i=n_a}^{n_d} O(i \log i) + \sum_{i=2n_d}^{2n_d+n_c} O(i \log i) + 2O(n_c) + O((2n_d + n_c)^2). \tag{9}$$

Owing to the fact that the operational rule of the symbol “O” can be simplified, the worst time complexity of one generation with information exchange for CONNIA can be written as $O((n_c + 2n_d)^2)$.

In real applications, the key factor to decide whether a technique can be applied is the running time. The further research on the practical running time of the proposed algorithm will be presented in Section 4.9.

4. Experimental Study

In this section, we compare CONNIA with three state-of-the-art MOEAs, including NNIA, NSGA-II, and SPEA2, on benchmark MOPs. Besides, some extensional problems based

on the benchmark MOPs are also tested. It is well known that the parameter setting has significant impact on MOEAs. Therefore, the parameter setting of the four algorithms is consistent with the original references and has some adjustments appropriately. For SPEA2, the size of the population is 100; the size of an external population is 100. For NSGA-II, the size of the population is 100. For NNIA, the maximum size of the dominant population is 100; the maximum size of an active population is 20. For CONNIA, the maximum sizes of the two dominant subpopulations are both 50, and the maximum sizes of the two active subpopulations are both 10. A given number of function evaluations are used as the stopping criteria. We obtain statistical experimental results by running the four algorithms 30 times independently. To simplify the expression, Arabic numerals 1, 2, 3, and 4 are used to denote CONNIA, NNIA, NSGA-II, and SPEA2.

4.1. Evaluation Metrics. To evaluate various performances of the compared algorithms, some numerical metrics are adopted, including generation distance [44], spacing [45], maximum spread [36], hypervolume [19, 46], and the coverage of two sets [40]. These numerical metrics are summarized as follows.

Generation Distance. The metric which measures the distance from the approximate Pareto-optimal front to the true Pareto-optimal front is defined as follows:

$$GD(P, P^*) = \frac{\sum_{v \in P} d(v, P^*)}{|P|}, \quad (10)$$

where P^* is a set of uniformly distributed points in the objective space along the Pareto front, P is an approximation to the Pareto front, $|P|$ is the number of solutions in P , and $d(v, P^*)$ is the minimum Euclidean distance between a point v in P and the solutions in P^* .

Spacing. The metric measures the uniformity of nondominated solutions in the objective space and is described as follows:

$$S = \sqrt{\frac{1}{|A| - 1} \sum_{i=1}^{|A|} (\bar{d} - d_i)^2}, \quad (11)$$

where $d_i = \min\{\sum_{m=1}^k |f_m(a_i) - f_m(a_j)|\}$, ($a_i, a_j \in A; i, j = 1, 2, \dots, |A|$), d_i is the distance between the solution i and another solution which is nearest to i , and \bar{d} is the average value of all d_i s.

Maximum Spread. The metric measures how “well” the true Pareto-optimal front is covered by the approximate Pareto-optimal front. It can be described as follows:

$$MS = \sqrt{\frac{1}{m} \sum_{i=1}^m \left\{ \frac{\min(f_i^{\max}, F_i^{\max}) - \max(f_i^{\min}, F_i^{\min})}{F_i^{\max} - F_i^{\min}} \right\}^2}, \quad (12)$$

where m is the number of objectives and f_i^{\max} and f_i^{\min} are the maximum and minimum values of the i th objective in

the approximate Pareto-optimal front, respectively. F_i^{\max} and F_i^{\min} are the maximum and minimum values of the i th objective in the true Pareto-optimal front, respectively.

Hypervolume. The metric measures the “volume” in the objective domain covered by a set of nondominated solutions. The definition of the metric is

$$HV = \text{volume} \left(\bigcup_{i=1}^{n_{PF}} v_i \right), \quad (13)$$

where n_{PF} is the number of nondominated solutions; for any nondominated solution i , a hypercube can be formed with a reference point and the solution i as the diagonal corners of the hypercube. Finally, the HV is the amount of domain occupied by the union of hypercubes.

Coverage of Two Sets. This metric measures the dominant relationship between two approximate Pareto-optimal sets A and B . The definition of the metric is described as follows:

$$I_C(A, B) \triangleq \frac{|\{\mathbf{b} \in B; \exists \mathbf{a} \in A : \mathbf{a} \succeq \mathbf{b}\}|}{|B|}, \quad (14)$$

where the symbol “ \succeq ” means domination. Note that both $I_C(B, A)$ and $I_C(A, B)$ have to be considered simultaneously, because the relationship between them is not completely linear.

4.2. Test Problems. To verify the versatility of the proposed algorithm, five ZDT [47] and five DTLZ problems [48] with diverse complexities in the field of multiobjective optimization are selected. Table 1 demonstrates that NNIA may fall into local optimum in solving ZDT2 and ZDT4. So as to further explore the performance of CONNIA in solving some extreme problems, five test problems based on ZDT2 and ZDT4 are designed.

The related problems based on ZDT2 are described as follows. When p equals the values of 2 and 3, the corresponding problems are named ZDT21 and ZDT22, respectively. Consider the following:

$$g(\mathbf{x}) = 1 + 9 \left(\frac{(\sum_{i=2}^n x_i)}{(n-1)} \right)^{0.25},$$

$$f_1(\mathbf{x}) = x_1, \quad f_2(\mathbf{x}) = \left\{ g(\mathbf{x}) \left[1 - \left(\frac{x_1}{g(\mathbf{x})} \right)^p \right] \right\}^{1/p}. \quad (15)$$

The shape of the Pareto-optimal front changes with the value of p . When p is greater than 1, the formative Pareto-optimal front is convex. If not, the formative Pareto-optimal front is concave. The curvature of the Pareto-optimal front also changes with the value of p . In Figure 3(a), we use Arabic numerals 1, 2, and 3 to concisely denote the Pareto-optimal fronts of ZDT2, ZDT21, and ZDT22, respectively.

Similar to the related problems based on ZDT2, the related problems based on ZDT4 are described as follows.

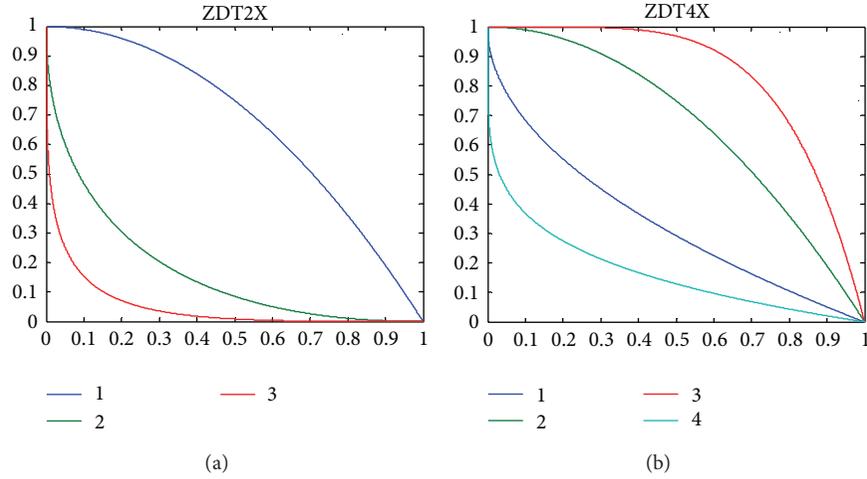


FIGURE 3: The true Pareto-optimal fronts of the related problems based on ZDT2 and ZDT4, respectively.

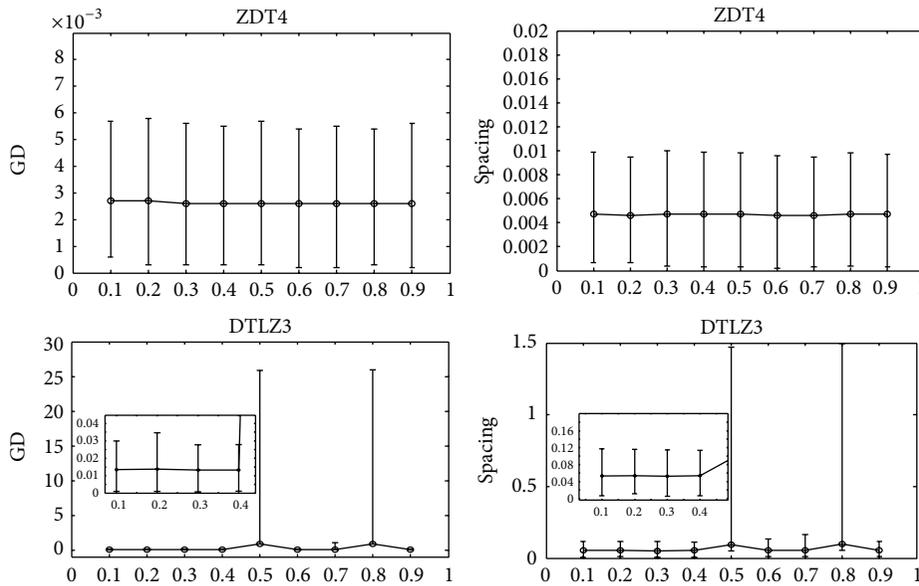


FIGURE 4: Mean values of GD and spacing versus the introduced parameter in solving DTLZ3 and ZDT4 by the proposed algorithm.

When q equals the values of 0.5, 2, 5, and 0.2, the corresponding problems are named ZDT4, ZDT41, ZDT42, and ZDT43, respectively. Consider the following:

$$f_1(\mathbf{x}) = x_1, \quad f_2(\mathbf{x}) = g(x) \left[1 - \left(\frac{x_1}{g(x)} \right)^q \right], \quad x_1 \in [0, 1],$$

$$g(x) = 1 + 10(n-1) + \sum_{i=2}^n [x_i^2 - 10 \cos(4\pi x_i)],$$

$$x \in [-5, 5], \quad i = 2, \dots, n, \quad n = 10. \tag{16}$$

The shape of the Pareto-optimal front changes with the value of q . When q is greater than 1, the formative Pareto-optimal front is convex. If not, the formative Pareto-optimal front is concave. The curvature of the Pareto-optimal front

also changes with the value of q . In Figure 3(b), we use Arabic numerals 1, 2, 3, and 4 to concisely denote the Pareto-optimal fronts of ZDT4, ZDT41, ZDT42, and ZDT43, respectively.

4.3. Sensitivity to the Introduced Parameter. The influence of the threshold is discussed in this part. Considering the representative of multi-objective problems with two-objectives and three-objectives, respectively, ZDT4 and DTLZ3 are selected for parameter analysis. Figure 4 shows that the mean values of GD and spacing are rather stable in dealing with ZDT4, whatever the value of the threshold is. However, the mean values of GD and spacing change greatly with the variation of the threshold in solving DTLZ3. Figure 4 indicates that the proposed algorithm is not sensitive to the threshold on simple problems. The performance has some differences with the variation of the threshold on difficult problems.

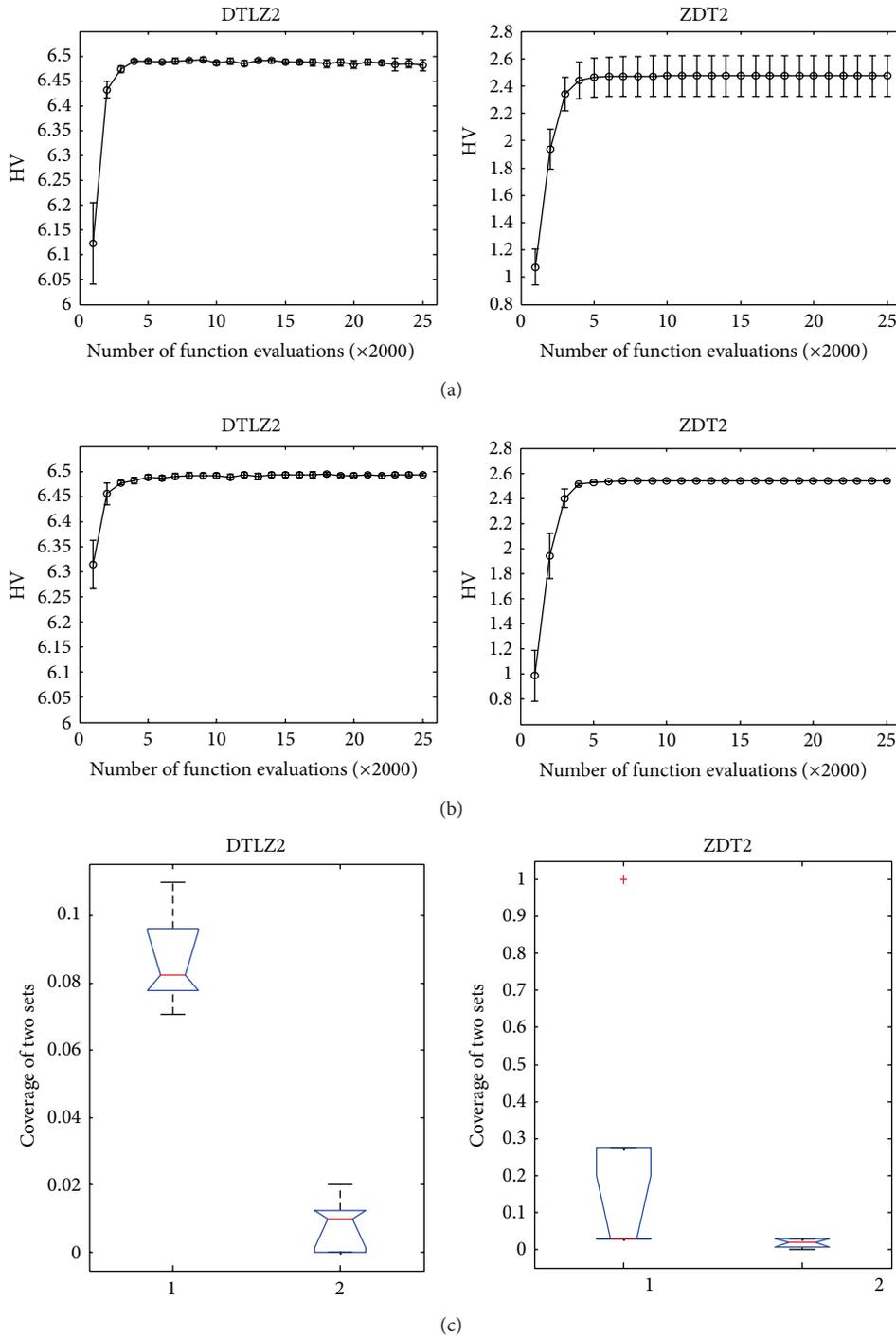
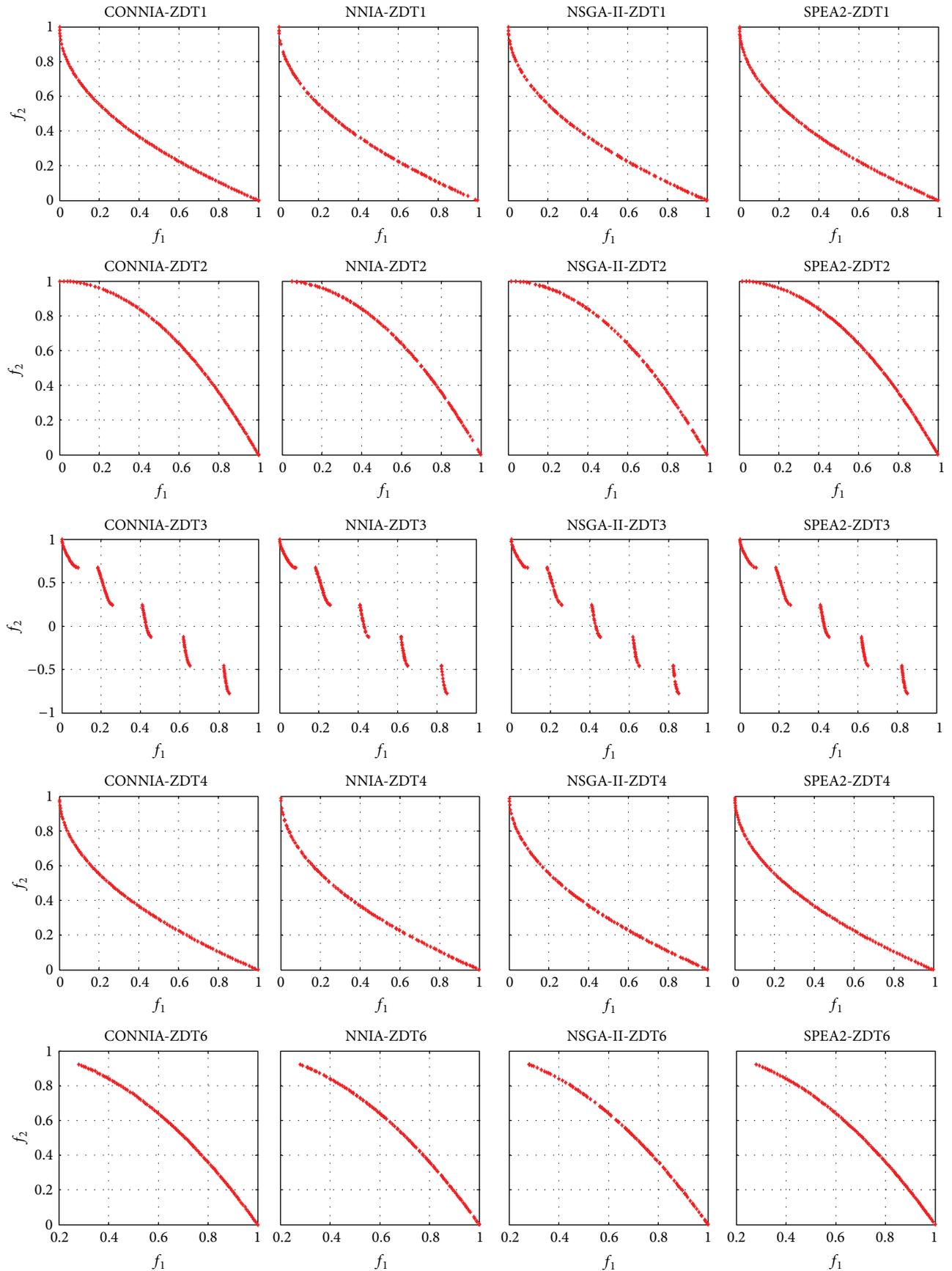


FIGURE 5: (a) The error bar of HV of the nondominated antibodies in final population with different number of function evaluations by CONNIA'. (b) The error bar of HV of the nondominated antibodies in final population with different number of function evaluations by CONNIA. (c) Box plots of the coverage of the two sets obtained by CONNIA with and without information exchange.

When the cooperation among subpopulations happens with a small value of the threshold, the information among subpopulations will keep coincidence with each other which leads to the ineffectiveness of the cooperation. On the contrary, when the late cooperation appears with a large value of the threshold, the differences among subpopulations are apparent.

Thus, there is little chance for the inferior subpopulation to gain experience from the superior one.

4.4. Comparison of CONNIA with and without Information Exchange. The cooperation operator reflects a mutually beneficial relationship between two subpopulations. By applying



(a)

FIGURE 6: Continued.

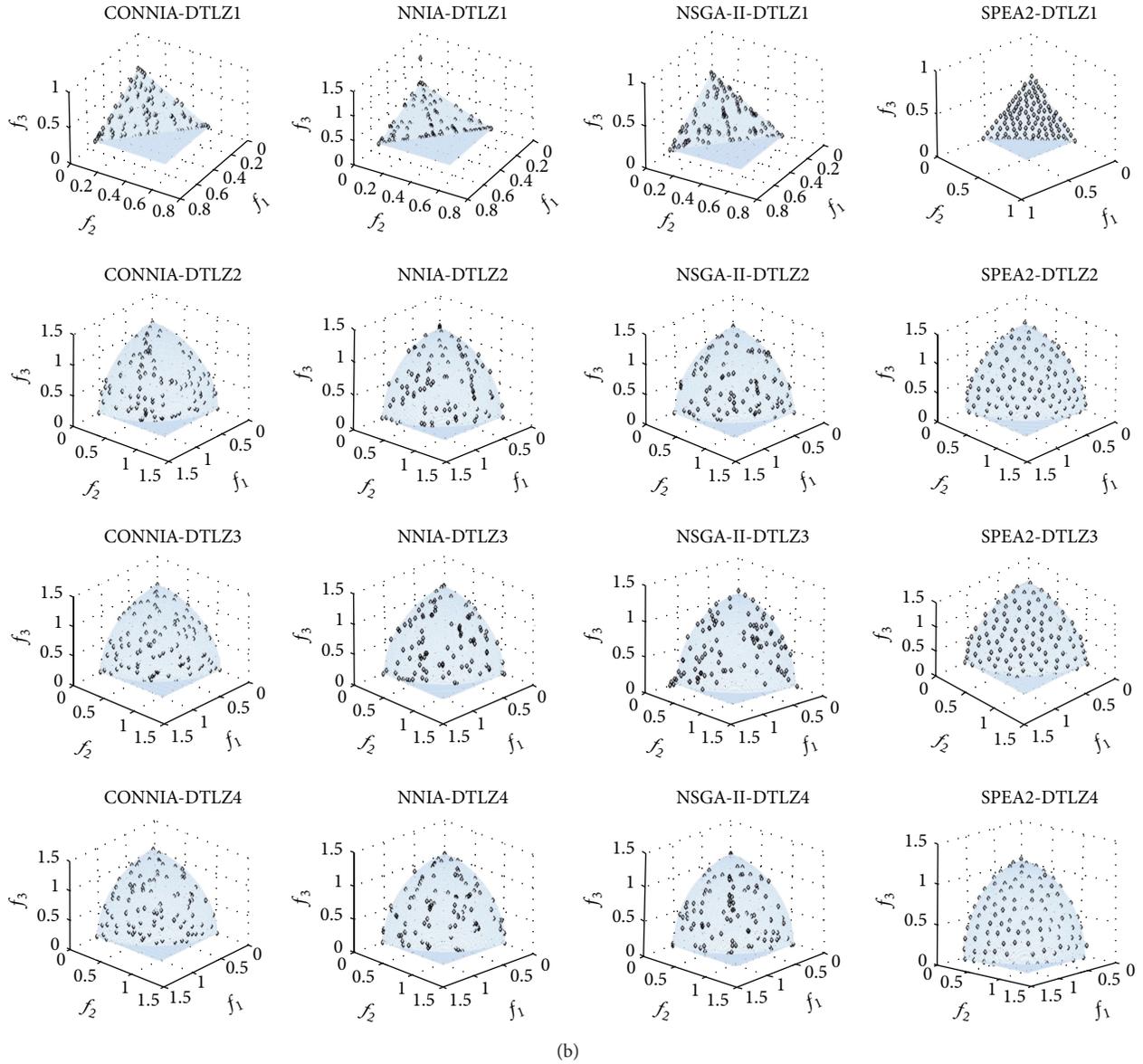


FIGURE 6: The approximate Pareto-optimal fronts obtained by CONNIA, NNIA, NSGA-II, and SPEA2 in solving the 9 test problems.

the cooperation operator, two subpopulations gain the opportunity to exchange information and expand the search range of the entire population. The subpopulation could make use of the reference experience from each other to improve its own evolution. This directed cooperation operator provides good evolutionary paths towards antibodies, thereby making antibodies evolve faster.

In this part, the effectiveness of information exchange among subpopulations is discussed. The proposed algorithm without information exchange is denoted by CONNIA'. Figures 5(a) and 5(b) show the error bars of hypervolume metric of nondominated antibodies in final population with different number of function evaluations by CONNIA and CONNIA', respectively. From Figures 5(a) and 5(b), some conclusions can be obtained: (1) the evolution curves of CONNIA are more flat than those of CONNIA'; (2) the

standard deviation of the error bar obtained by CONNIA becomes near to zeros; (3) with the same number of function evaluations, CONNIA obtains a higher value of HV metric than CONNIA'. Figure 5(c) shows the box plots of CONNIA against CONNIA' in terms of the coverage of two sets. In each plot, the left box represents the distribution of $I_C(\text{CONNIA}, \text{CONNIA}')$ and the right box represents the distribution of $I_C(\text{CONNIA}', \text{CONNIA})$. The box plots of $I_C(\text{CONNIA}, \text{CONNIA}')$ are higher than the corresponding box plots of $I_C(\text{CONNIA}', \text{CONNIA})$. Therefore, we can get the conclusion that CONNIA performs better than CONNIA' as far as the coverage is concerned.

4.5. Experimental Results on ZDT and DTLZ Problems. Figure 6 shows the distribution of approximate Pareto-optimal solutions obtained by four algorithms on ZDT and

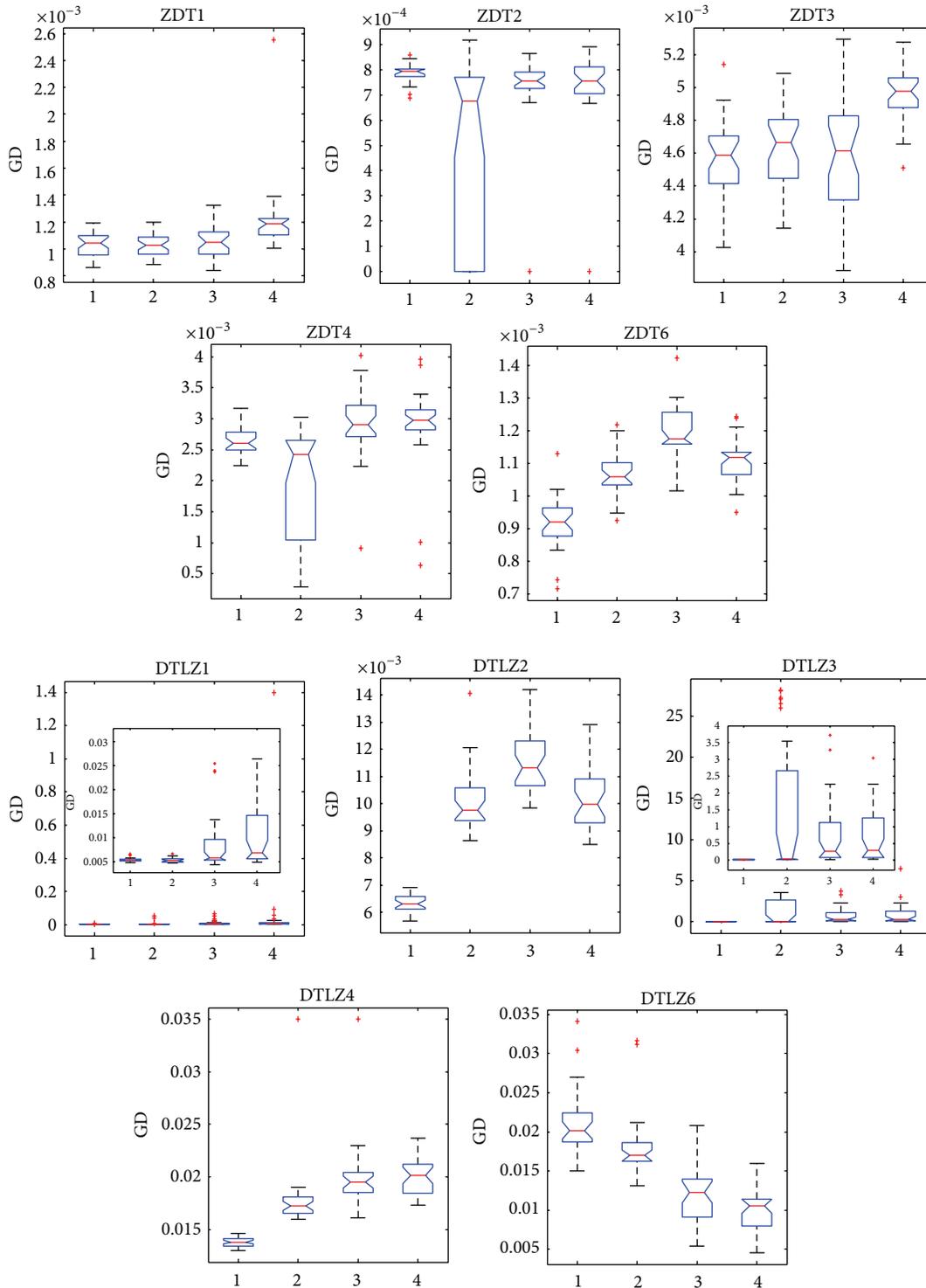


FIGURE 7: Statistical values of convergence obtained by CONNI, NNIA, NSGA-II, and SPEA2 in solving the 10 test problems.

DTLZ problems. The distributions of the approximate Pareto-optimal solutions obtained by CONNIA and SPEA2 are more uniform than those obtained by other two algorithms on five ZDT problems. The approximate Pareto-optimal solutions obtained by NNIA can not well cover the extreme solutions

of ZDT2 and ZDT4. For DTLZ problems, the distribution of the approximate Pareto-optimal solutions obtained by SPEA2 is the most uniform among the four algorithms; nevertheless the computational complexity of SPEA2 is the highest. The distribution of the approximate Pareto-optimal

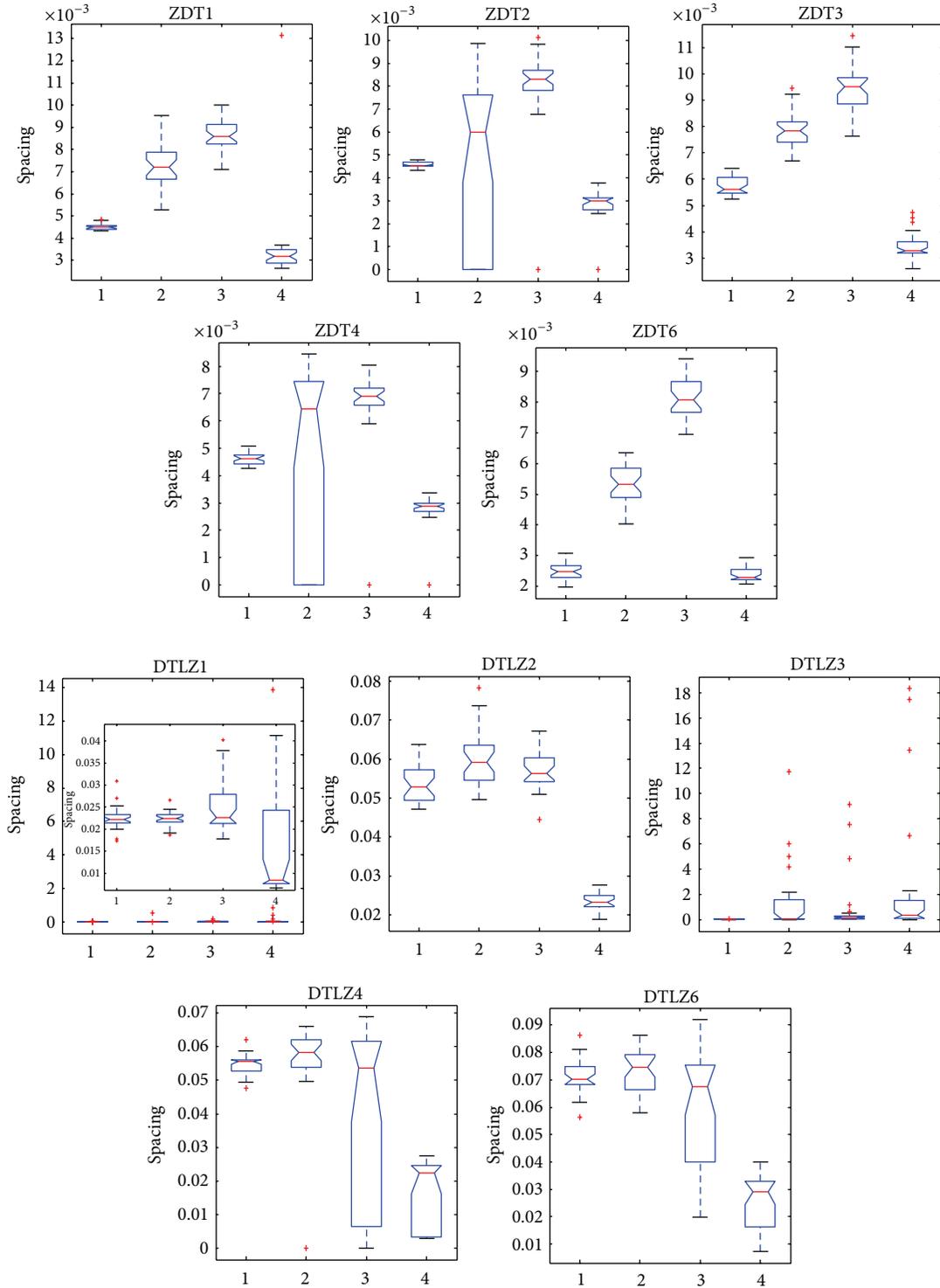


FIGURE 8: Statistical values of spacing obtained by CONNI, NNIA, NSGA-II, and SPEA2 in solving the 10 test problems.

solutions obtained by CONNIA is the most uniform among the remaining three algorithms. In addition to the qualitative analysis of the results, we also analyze statistical results obtained by four algorithms. The statistical results of convergence, spacing, maximum spread, and hypervolume are shown in Figures 7–10.

Figure 7 shows that the values of convergence can reach 10^{-3} in almost all the 30 independent runs by four algorithms on five ZDT problems. The box plots obtained by NNIA on ZDT2 and ZDT4 are quite broad which indicates that the stability of NNIA in solving these problems is quite poor. However, CONNIA is more robust than NNIA on ZDT2 and

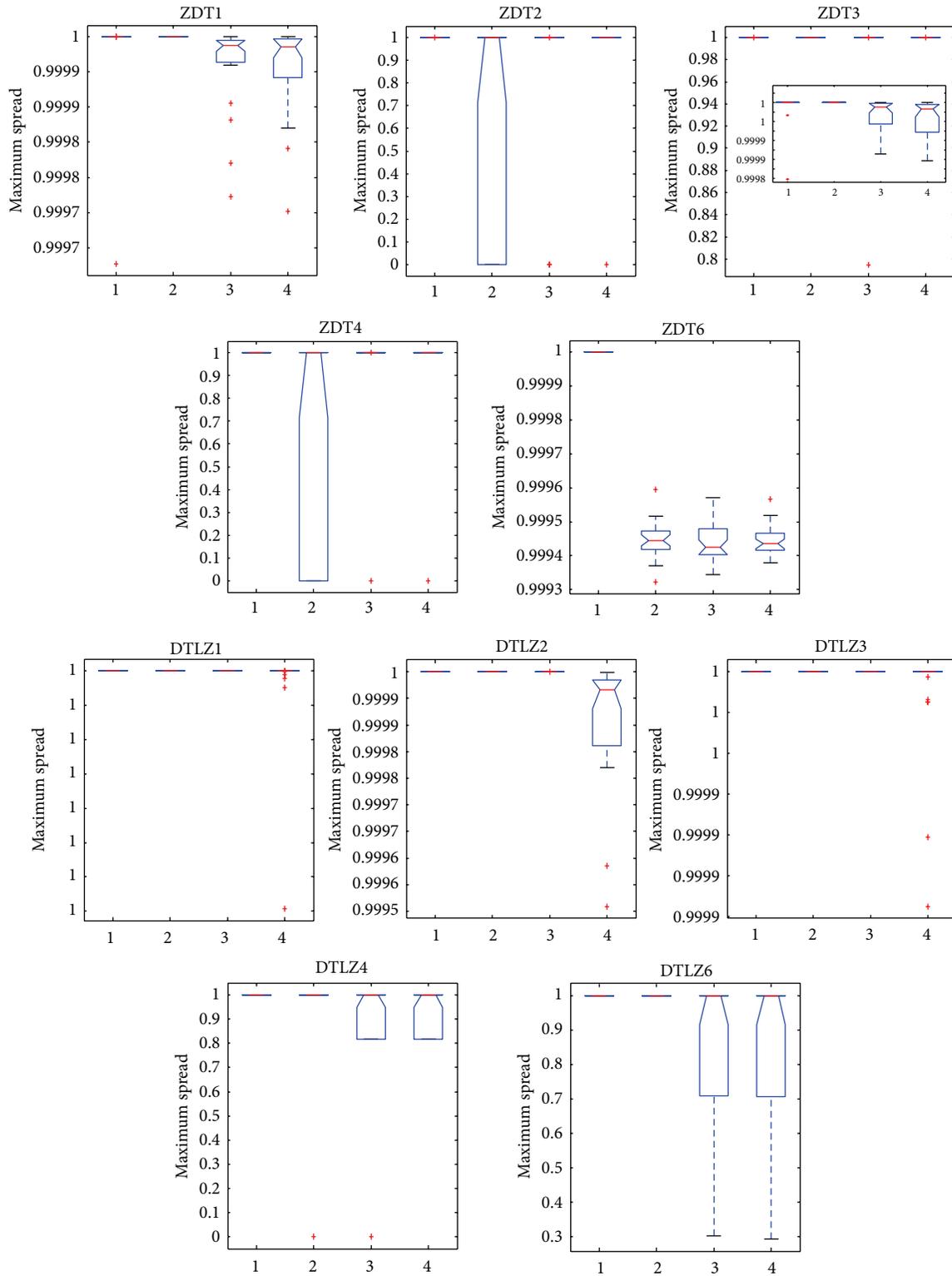


FIGURE 9: Statistical values of maximum spread obtained by CONNI, NNIA, NSGA-II, and SPEA2 in solving the 10 test problems.

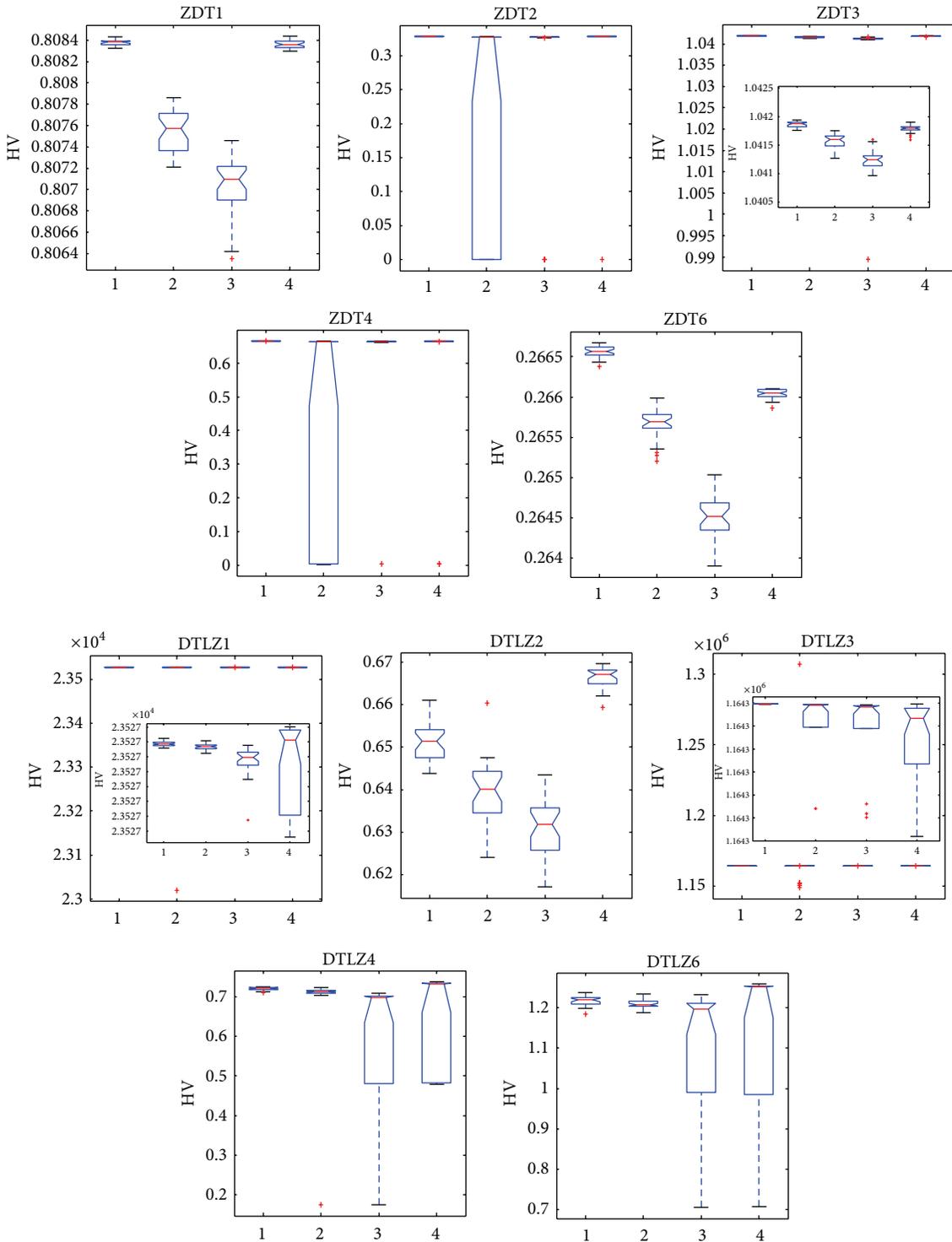


FIGURE 10: Statistical values of HV obtained by CONNI, NNIA, NSGA-II, and SPEA2 in solving the 10 test problems.

ZDT4, owing to the multipopulation coevolutionary strategy which plays an important role in maintaining the population diversity. In general, except for the appearance of local optimum when NNIA deals with ZDT2 and ZDT4, the differences among four algorithms on five ZDT problems are relatively small. Hereinto, CONNIA obtains the smallest values of convergence on ZDT3, ZDT4, and ZDT6. It has been

pointed out that NSGA-II and SPEA2 could not completely converge onto the true Pareto-optimal fronts in a limited number of function evaluations on DTLZ3 which has some local Pareto-optimal fronts [35, 49]. However, CONNIA obtains the best results in terms of convergence on DTLZ3. As far as convergence is concerned, CONNIA performs best on DTLZ1, DTLZ2, DTLZ3, and DTLZ4.

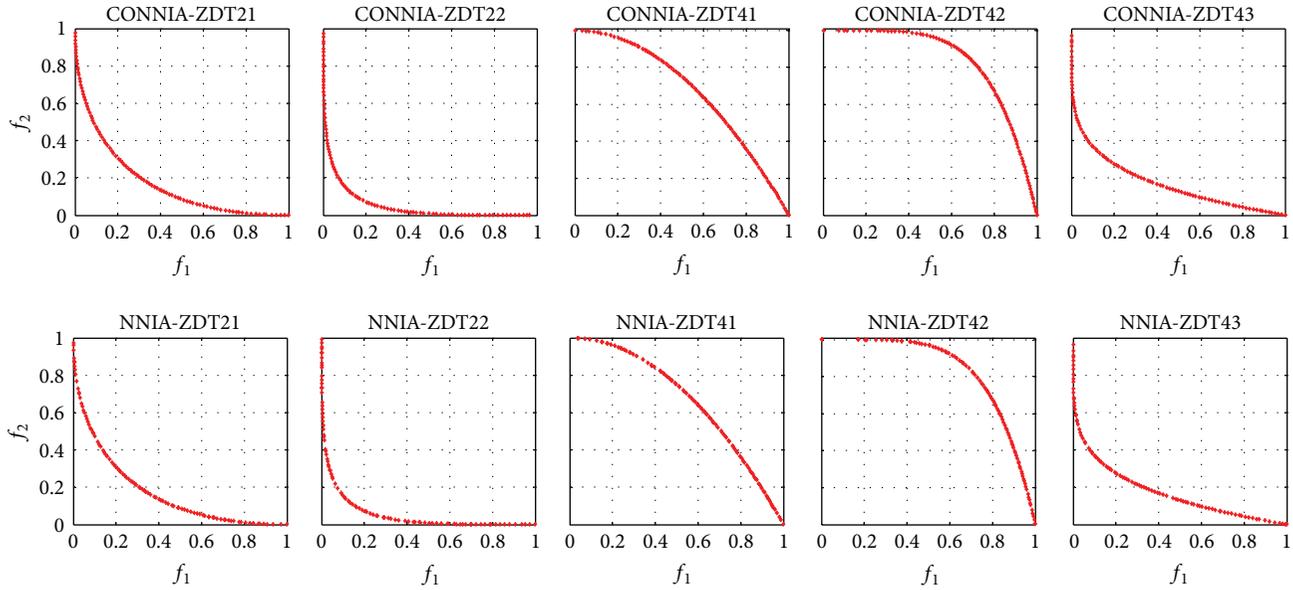


FIGURE 11: The approximate Pareto-optimal fronts obtained by CONNIA and NNIA in solving the extensional problems.

Figure 8 shows that, compared with the other three algorithms, SPEA2 performs best in most problems in terms of spacing. Apart from SPEA2, statistical values obtained by CONNIA are smaller than those obtained by other two algorithms in 9 out of the 10 problems. The statistical value obtained by CONNIA is even smaller than that obtained by SPEA2 on DTLZ3. The reason is that SPEA2 can not quite converge onto the true Pareto-optimal fronts in a limited number of function evaluations. In general, SPEA2 exhibits the best performance in diversity maintaining among the four algorithms. However, the complicated calculation of SPEA2 costs a large amount of computing resources. The proposed algorithm gets the smallest values of spacing among the remaining three algorithms in solving ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, DTLZ1, DTLZ2, DTLZ3, and DTLZ4.

Figure 9 demonstrates that NNIA obtains broad box plots on ZDT2 and ZDT4, thereby suggesting that the stability of NNIA on ZDT2 and ZDT4 is poor. Compared with the other three algorithms, CONNIA obtains the largest statistical values of MS on all the 10 test problems, while NSGA-II and SPEA2 perform slightly poor on ZDT1, ZDT3, ZDT6, DTLZ4, and DTLZ6. Figure 10 shows that the stability of NNIA is quite poor on ZDT2 and ZDT4 in terms of HV. However, in most of the 10 test problems except DTLZ2, the result obtained by the CONNIA is not inferior to that obtained by other three algorithms as far as HV is concerned. Apparently, SPEA2 does well in diversity maintenance in the field of MOEAs. In solving five ZDT problems, SPEA2 gets the results similar to CONNIA in terms of HV. However, SPEA2 can not well converge onto the true Pareto-optimal fronts in 50000 function evaluations in solving difficult problems. CONNIA achieves the results which are not worse than, or even better than, those of SPEA2 with much lower complexity on the nine test problems.

4.6. *Comparing the Robustness of NNIA and CONNIA.* The comparison of CONNIA and NNIA on some difficult problems (DTLZ1 and DTLZ3) and some extreme problems (ZDT21, ZDT22, ZDT41, ZDT42, and ZDT43) is carried out in this part. Figure 11 shows the distribution of approximate Pareto-optimal solutions obtained by CONNIA and NNIA. The distributions of approximate Pareto-optimal solutions obtained by CONNIA are relatively more uniform than those of NNIA. The solutions obtained by NNIA can not well cover extreme solutions in solving ZDT21, ZDT41, and ZDT42. Nevertheless, CONNIA can well cover these solutions in solving the same problems.

Figure 12 shows the box plots of CONNIA against NNIA based on the coverage of two sets. NNIA obtains a relatively wider range of box plot measures on ZDT2, ZDT4, and ZDT41; that is, the stability of NNIA is relatively weak in dealing with these problems. However, the performance obtained by CONNIA is more stable on the same problems. The box plots of $C(1, 2)$ are higher than the corresponding box plots of $C(2, 1)$ in all the test problems as shown in Figure 12. Therefore, we can get the conclusion that the solutions obtained by CONNIA almost weakly dominate those obtained by NNIA.

4.7. *Tests on Convergence of the Four Algorithms.* In the field of MOEAs, the number of function evaluations is commonly used as the stopping criteria. It is difficult to set the accurate stopping criteria for an MOEA on different problems, while uniform stopping criteria which are applied to different problems always provide a plethora of information [49]. After investigating the running convergence with different function evaluations, the effective stopping criteria of CONNIA on different problems can be discovered. To demonstrate the

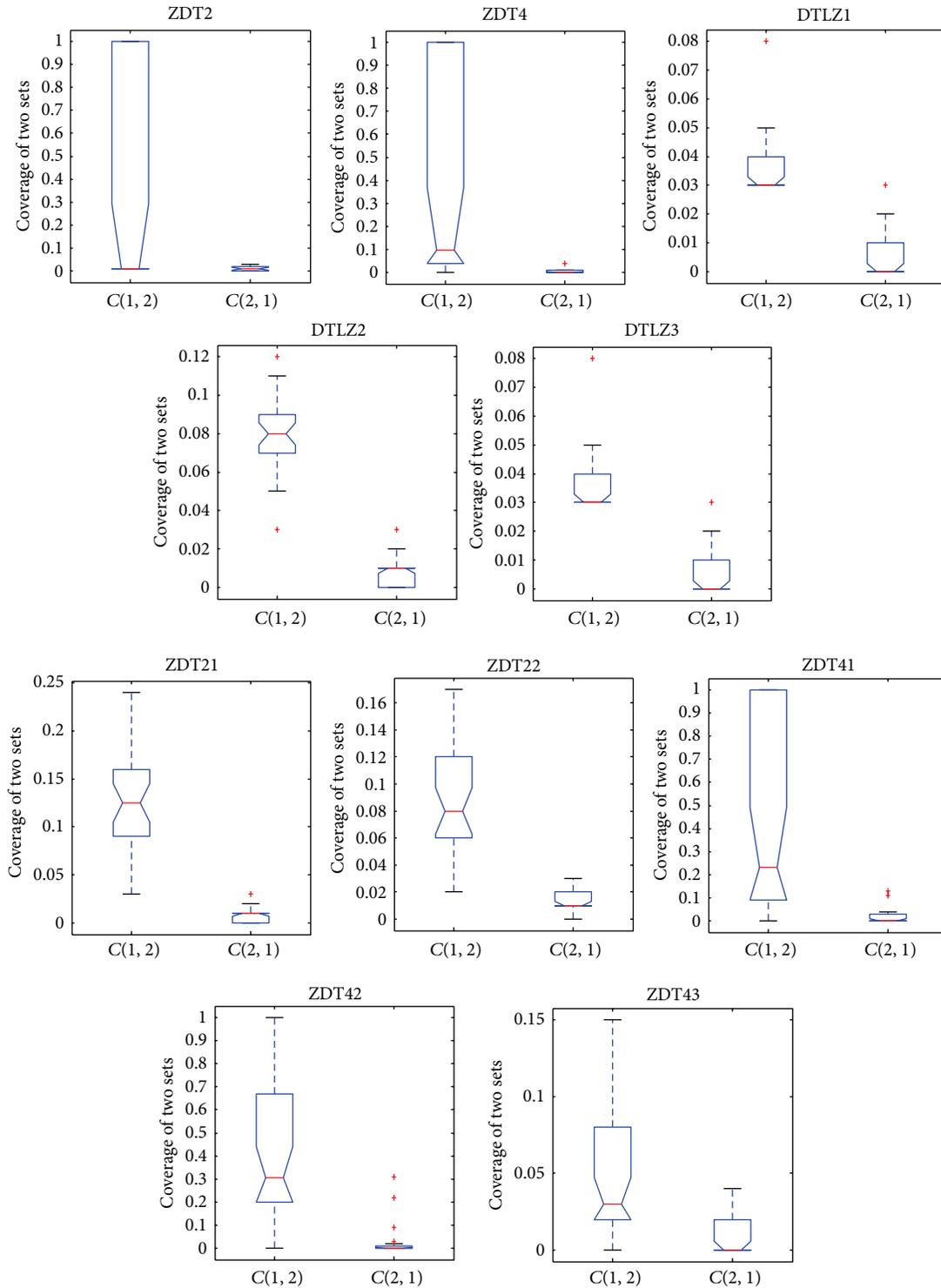


FIGURE 12: Box plots of the coverage of two sets by CONNIA and NNIA in solving 10 test problems.

convergence of four algorithms more explicitly, results are showed with Y coordinate in the form of log 10.

Figure 13 shows the mean value in terms of convergence with different function evaluations by four algorithms. The differences among four algorithms are not obvious on five ZDT problems. However, the disparities among them are

apparent on five DTLZ problems. CONNIA obtains better performance than the other three algorithms on DTLZ1, DTLZ2, DTLZ3, and DTLZ4. In particular on some intractable problems, such as DTLZ1 and DTLZ3, SPEA2 and NSGA-II can not well converge onto the true Pareto-optimal front with a limited number of function evaluations, while

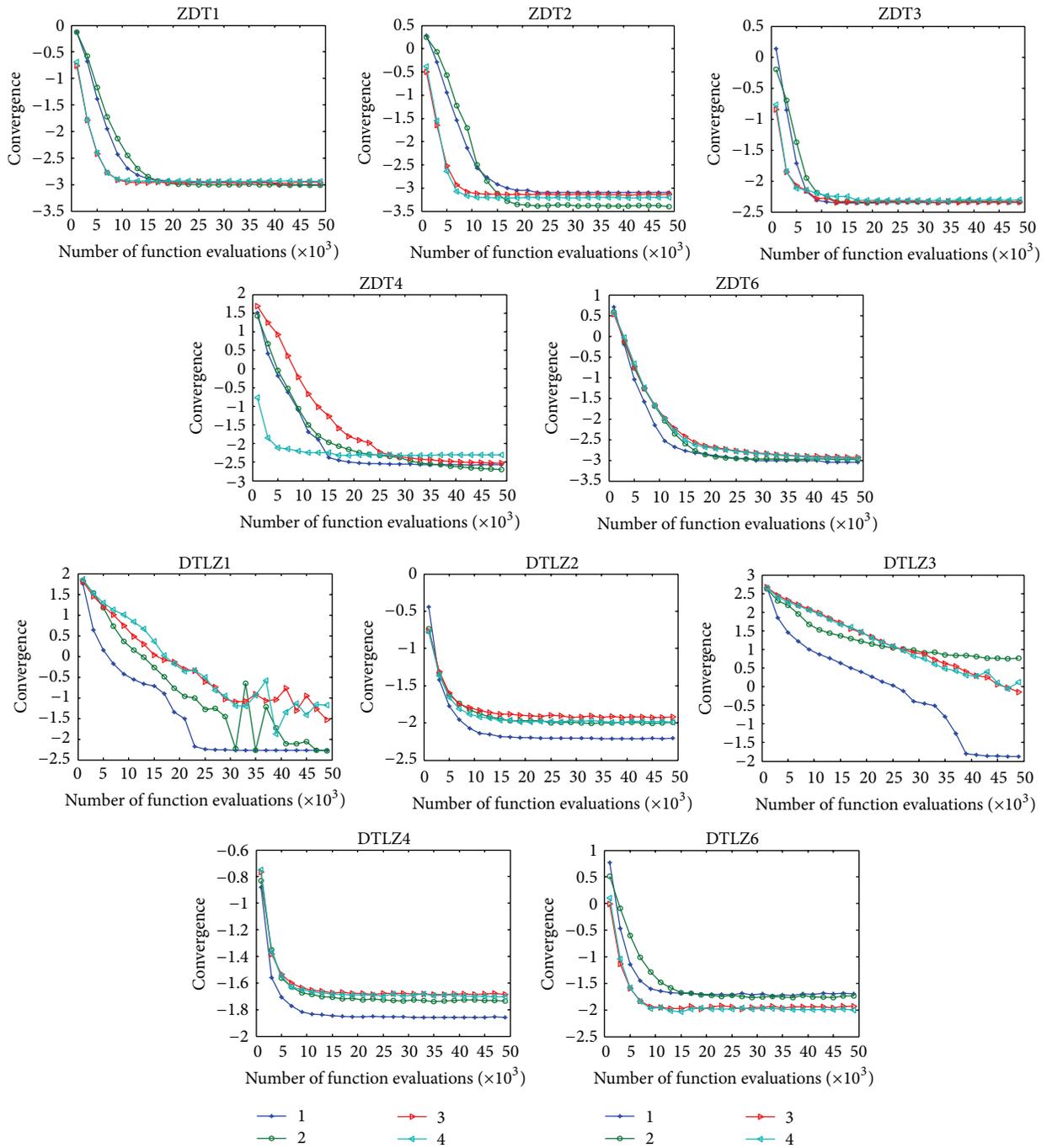


FIGURE 13: Mean value of convergence with different function evaluations by the four algorithms in solving the 10 test problems.

under the same condition CONNIA shows distinct advantages.

4.8. Experimental Results of the Four Algorithms on Many-Objective Problems. In this section, the performance of four algorithms on many-objective problems is investigated. Multiobjective problems with more than three objectives are defined as many-objective problems. The test problems are

the extensional problems of DTLZ1 and DTLZ2 with 4 to 7 objectives and are named DTLZ14–DTLZ17 and DTLZ24–DTLZ27, respectively. Due to the fact that the number of non-dominated solutions dramatically enlarges with the number of objectives increasing, many MOEAs have difficulty in converging onto the true Pareto-optimal front with a limited number of function evaluations. Therefore, the size of population and the number of function evaluations are doubled as those in Section 4.5 [35].

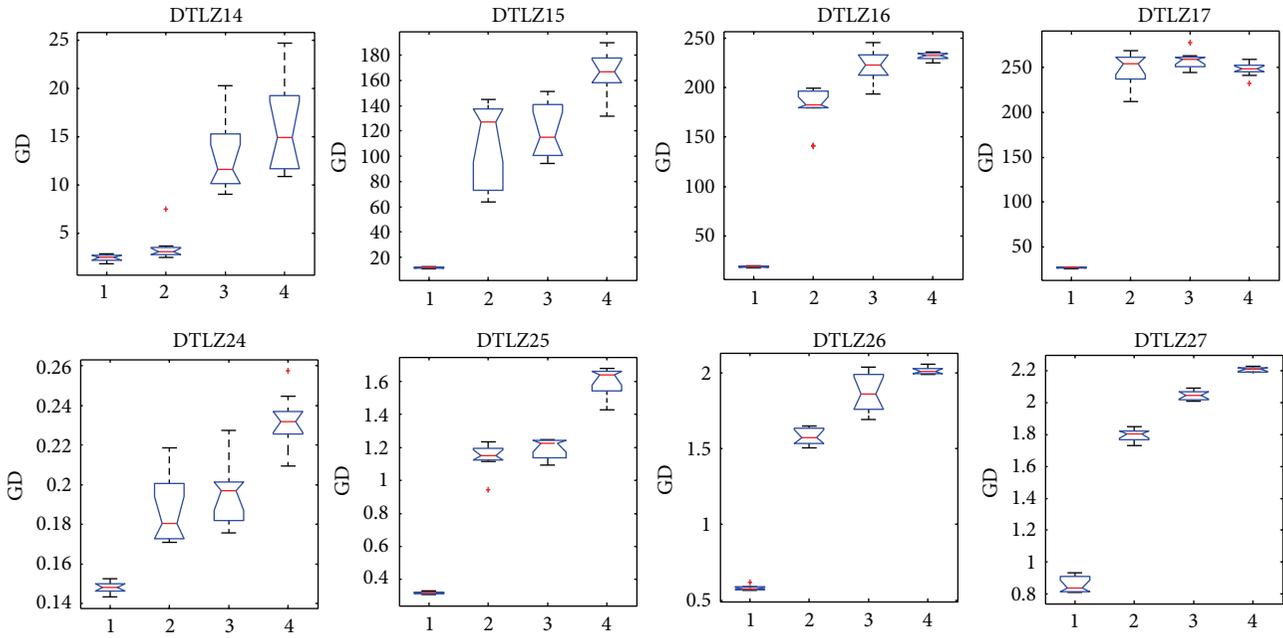


FIGURE 14: Statistical values of convergence obtained by the four algorithms in solving many-objective problems.

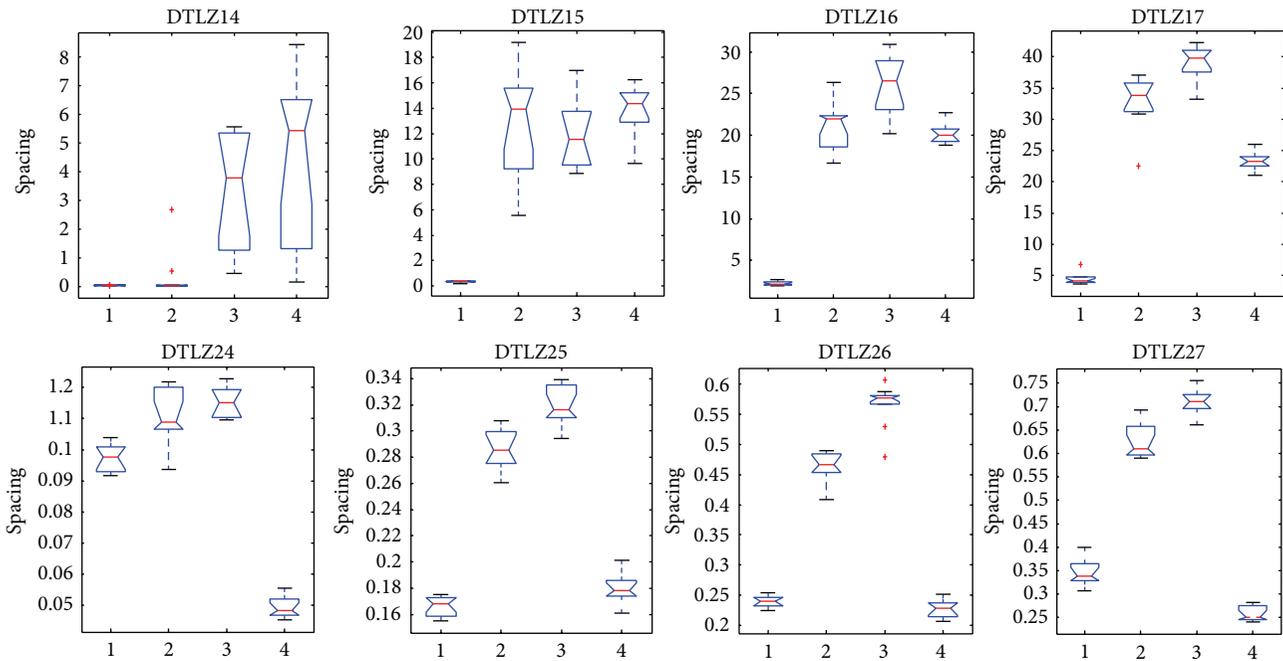


FIGURE 15: Statistical values of spacing obtained by the four algorithms in solving many-objective problems.

Figure 14 shows that CONNIA obtains the largest statistical values of convergence among four algorithms on all the test problems, closely followed by NNIA. While results obtained by SPEA2 and NSGA-II are relatively worse in terms of convergence, Figure 15 indicates that the result of CONNIA is even better than SPEA2 in terms of spacing. SPEA2 cannot converge onto the true Pareto-optimal fronts with a limited

number of function evaluations on eight many-objective problems. In this case, the diversity maintaining mechanism used in SPEA2 is no longer effective. The statistical values of MS on eight many-objective problems are shown in Figure 16. In terms of MS, four algorithms obtain similar results, except SPEA2 which does slightly worse. Overall, CONNIA performs much better than the other three algorithms on

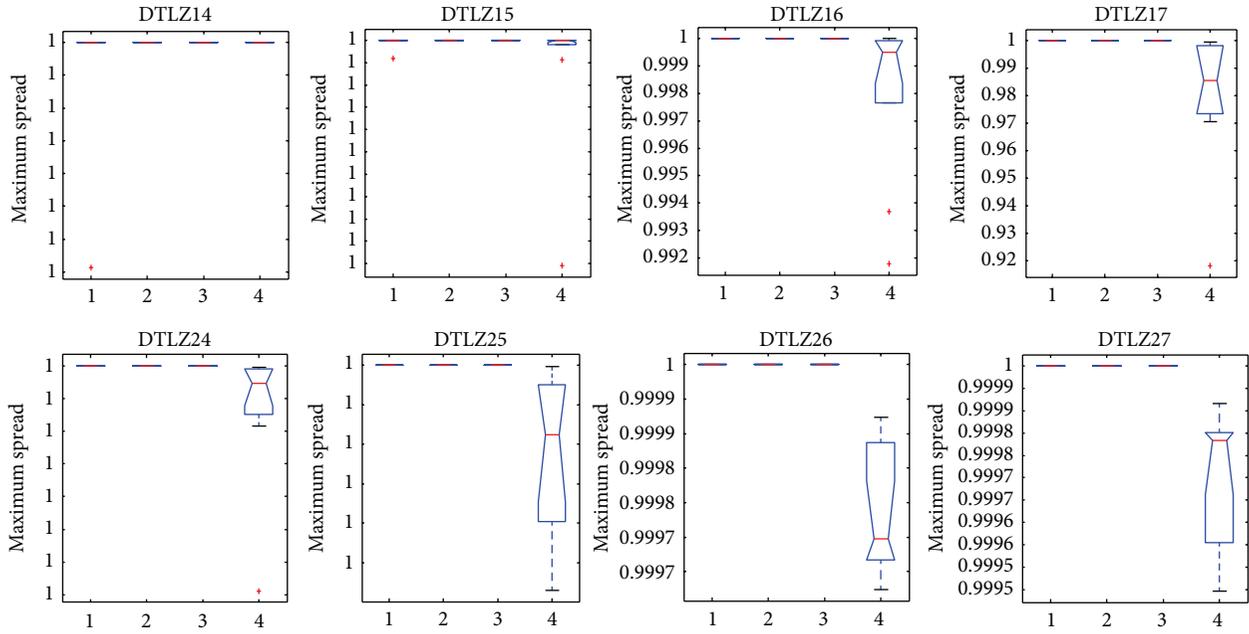


FIGURE 16: Statistical values of maximum spread obtained by the four algorithms in solving many-objective problems.

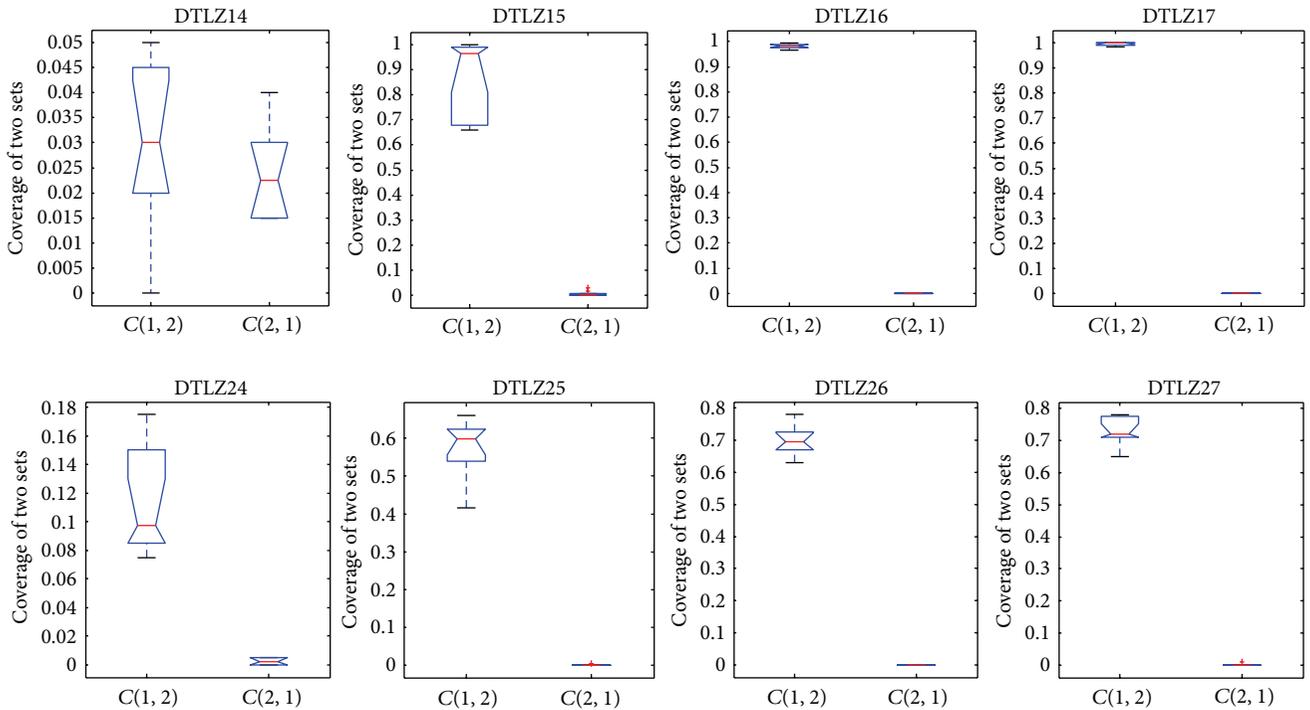


FIGURE 17: Statistical values of the coverage of two sets obtained by CONNIA and NNIA in solving many-objective problems.

eight many-objective problems. The performance of NSGA-II and SPEA2 is seriously degenerated in solving many-objective problems.

The convergence metric can be only used under the condition of which knowledge of the true Pareto-optimal fronts is available, which is unsuitable for many-objective problems. Hence, the metric of the coverage of two sets is employed to measure the dominant relationship between solutions

obtained by different algorithms. Figures 17, 18, and 19 show the comparison between CONNIA and the other three algorithms on many-objective problems in terms of the coverage of two sets. Figures 17–19 indicate that the values of $C(*, 1)$ are smaller than the corresponding values of $C(1, *)$. Hereinto, 1 denotes the solution set obtained by CONNIA, and the symbol “*” stands for the solution set obtained by any one of the other three algorithms. The gap between $C(*, 1)$ and $C(1, *)$

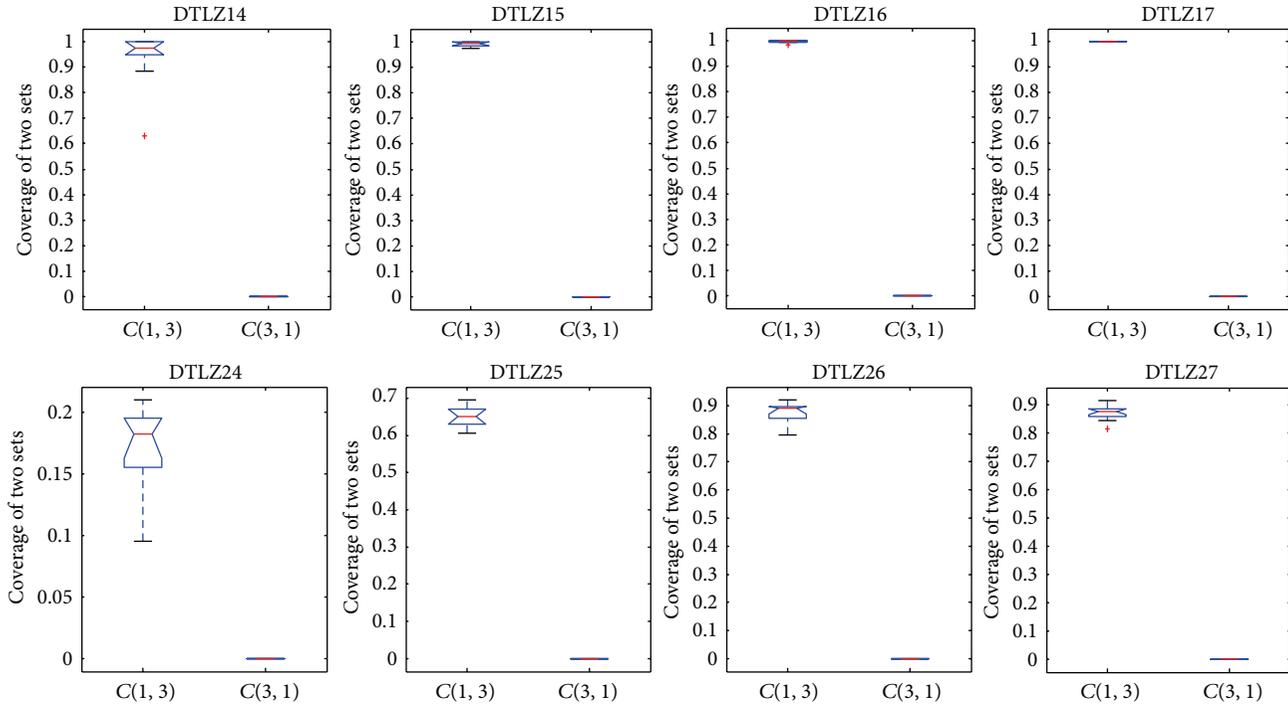


FIGURE 18: Statistical values of the coverage of two sets obtained by CONNIA and NSGA-II in solving many-objective problems.

is enlarged with the number of objectives increasing, which indicates that the dominant relationship between the solutions obtained by CONNIA and other three algorithms is more apparent on complex many-objective problems. In the special case that $C(1, *) = 1$ and $C(*, 1) = 0$, the solution set obtained by CONNIA almost dominates that obtained by any one of the other three algorithms. For example, in solving DTLZ15, DTLZ16, DTLZ17, and DTLZ27, the values of $C(*, 1)$ are almost equal to 0, while the values of $C(1, *)$ are almost equal to 1. CONNIA outperforms the other three algorithms in most cases as coverage is concerned.

4.9. Running Time Study. Figure 20 shows the average running time on the extensional problems of DTLZ1 and DTLZ2 with 3 to 7 objectives, respectively. As shown in Figure 20, the cost of the average running time of four algorithms increases with the number of objectives increasing. NNIA exhibits the best performance in terms of computational time, closely followed by CONNIA. The running time of NSGA-II and SPEA2 is relatively longer; particularly for SPEA2, the required running time is the longest among the four algorithms. This is because SPEA2 adopts a relatively expensive diversity maintaining mechanism whose worst run-time complexity is $O(N^3)$, where N is the number of nondominated solutions. NNIA is an effective immune inspired MOEA, which is famous for good performance in convergence [35, 50]. Although its special evolutionary framework results in fast convergence, solutions obtained by NNIA are occasionally trapped into local optimum. It is required to focus on the pursuit of not only a high convergence rate, but also good evolutionary quality. CONNIA is an enhanced version of NNIA

by introducing the multipopulation coevolutionary strategy and an adaptive operator. Although the computational cost of CONNIA is a little larger than NNIA, the improvement on the performance is evident.

5. Conclusion

To the best of our knowledge, slow convergence rate is a ubiquitous problem in MOEAs. AIS have the learnable, parallel, and distributed characteristics and possess an efficient information processing ability. AIS-based algorithms have already been widely used for dealing with MOPs, in which NNIA obtains a fast convergence rate solving such knotty problem in MOEAs. However, the population diversity can not be well maintained in NNIA, which leads the solutions obtained by NNIA to be trapped into local optimum on some difficult problems. Co-evolution is a high-level evolutionary method, which confirms that all the populations are beneficial mutually, thus providing a theoretical basis for maintaining diversity. In this paper, a multipopulation coevolutionary strategy is designed. Subpopulations implement evolutionary operation independently; thus the diversity of the entire population can be well maintained. The information exchange among subpopulations is available, thereby expanding the search range of the entire population and improving the efficiency of evolutionary search.

In the field of MOEAs, when it comes to adaptive algorithms, most of them adaptively adjust some parameters, such as population size, crossover probability, and mutation probability. However, an adaptive algorithm with online-decision strategy is seldom involved. Based on this idea, an adaptive

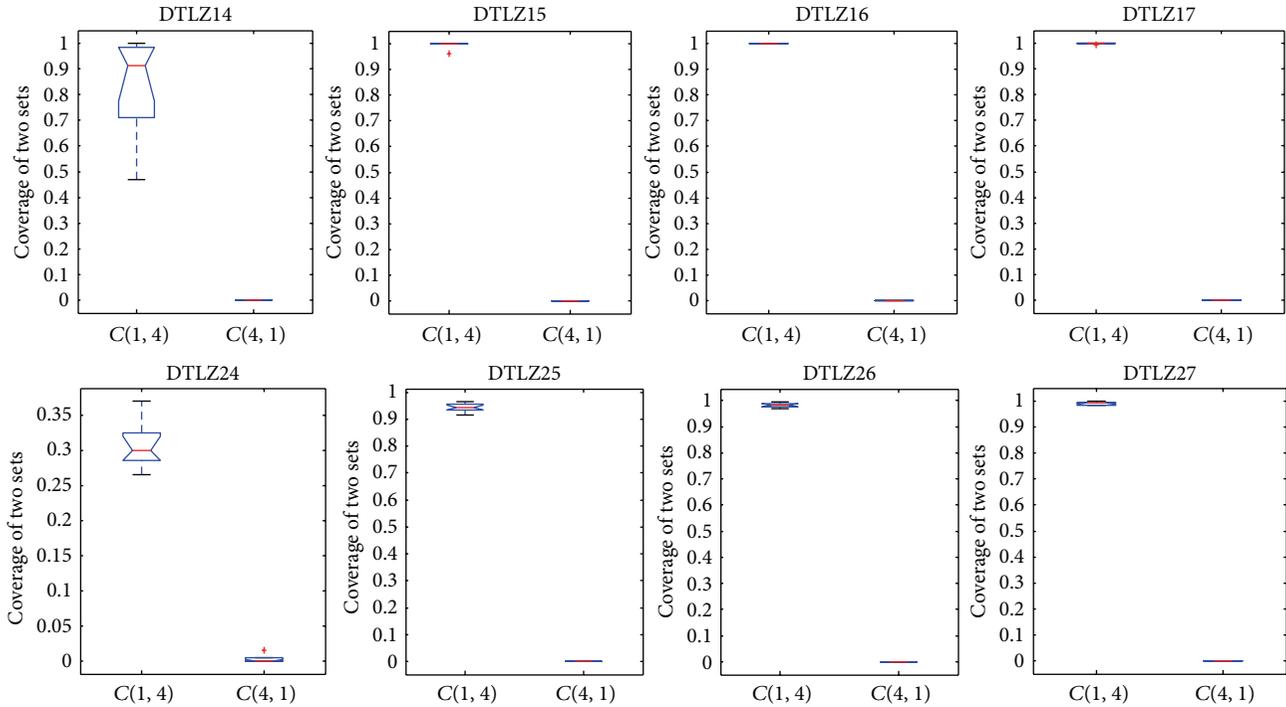


FIGURE 19: Statistical values of the coverage of two sets obtained by CONNIA and SPEA2 in solving many-objective problems.

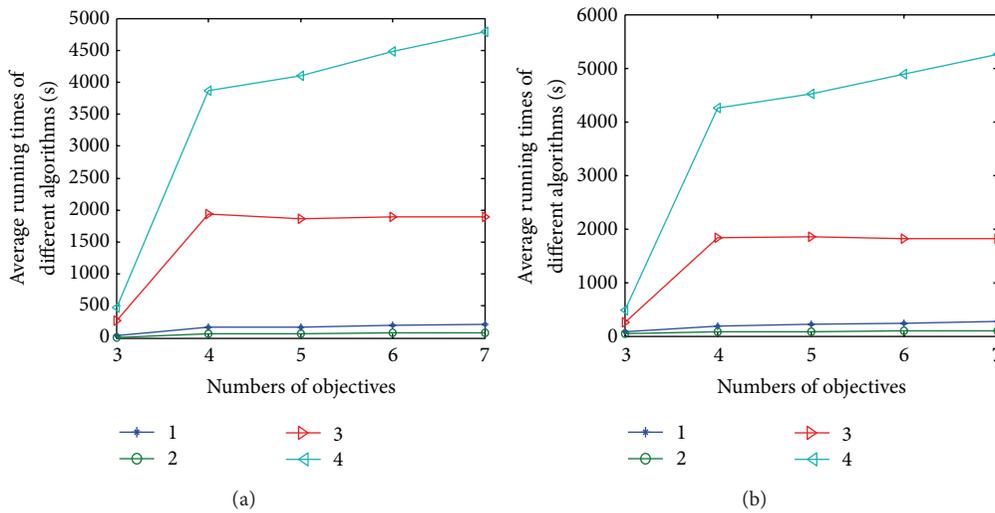


FIGURE 20: Mean value of running time by four algorithms on the extensional problems of DTLZ1 and DTLZ2 with 3 to 7 objectives, respectively.

strategy is designed in the proposed algorithm. Subpopulations adopt corresponding operations according to the conditions of themselves which ensures that evolutionary search is not random or blind.

In dealing with many-objective problems, the rapid increase of nondominated solutions requires a large size of population or a large number of function evaluations. However, in many MOEAs, the size of population is constant. No matter how difficult the problem is, the size of population is the same. According to the characteristics of CONNIA, it is

more reasonable to adaptively adjust the number of subpopulations according to the difficulty of the problem. We can imagine that it is more reasonable to employ more subpopulations together to cooperatively overcome the difficulty in solving many-objective problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 61273317, 61202176, and 61203303), the National Top Youth Talents Program of China, the Specialized Research Fund for the Doctoral Program of Higher Education (Grant no. 20130203110011), and the Fundamental Research Fund for the Central Universities (Grant nos. K50510020001 and K5051202053).

References

- [1] T. Pichpibal and R. Kawtummachi, "A heuristic approach based on clarke-wright algorithm for open vehicle routing problem," *The Scientific World Journal*, vol. 2013, Article ID 874349, 11 pages, 2013.
- [2] S. Y. Zeng, Z. Liu, C. H. Li, Q. Zhang, and W. G. Wang, "An evolutionary algorithm and its application in antenna design," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 2, pp. 129–137, 2012.
- [3] R. K. Sahu, S. Panda, U. K. Rout, and P. Raul, "Application of gravitational search algorithm for load frequency control of multi area power system," *Journal of Bioinformatics and Intelligent Control*, vol. 2, no. 3, pp. 200–210, 2013.
- [4] B. Yu, Z. H. Cui, and G. Y. Zhang, "Artificial plant optimization algorithm with correlation branches," *Journal of Bioinformatics and Intelligent Control*, vol. 2, no. 2, pp. 146–155, 2013.
- [5] J. Muñozuri, P. C. Achedad, M. Rodríguez, and R. Grosso, "Use of a genetic algorithm for building efficient choice designs," *Journal of Bioinformatics and Intelligent Control*, vol. 4, no. 1, pp. 27–32, 2012.
- [6] B. B. Pal, D. Chakraborti, P. Biswas, and A. Mukhopadhyay, "An application of genetic algorithm method for solving patrol manpower deployment problems through fuzzy goal programming in traffic management system: a case study," *Journal of Bioinformatics and Intelligent Control*, vol. 4, no. 1, pp. 47–60, 2012.
- [7] A. F. Sheta, P. Rausch, and A. S. Al-Afeef, "A monitoring and control framework for lost foam casting manufacturing processes using genetic programming," *Journal of Bioinformatics and Intelligent Control*, vol. 4, no. 2, pp. 111–118, 2012.
- [8] D. Donmez, O. Simsek, T. Izgu, Y. A. Kacar, and Y. Y. Mendi, "Genetic transformation in citrus," *The Scientific World Journal*, vol. 2013, Article ID 491207, 8 pages, 2013.
- [9] P. M. Vasant, V. N. Dieu, and L. L. Dinh, "Artificial bee colony algorithm for solving optimal power flow problem," *The Scientific World Journal*, vol. 2013, Article ID 159040, 9 pages, 2013.
- [10] M. G. Gong, X. W. Chen, L. J. Ma, Q. F. Zhang, and L. C. Jiao, "Identification do multi-resolution network structures with multi-objective immune algorithm," *Applied Soft Computing*, vol. 13, no. 4, pp. 1705–1717, 2013.
- [11] M. Gong, L. Ma, Q. Zhang, and L. Jiao, "Community detection in networks by using multiobjective evolutionary algorithm with decomposition," *Physica A*, vol. 391, no. 15, pp. 4050–4060, 2012.
- [12] M. G. Gong, L. J. Zhang, J. J. Ma, and L. C. Jiao, "Community detection in dynamic social networks based on multiobjective immune algorithm," *Journal of Computer Science and Technology*, vol. 27, no. 3, pp. 455–467, 2012.
- [13] M. Gong, J. Zhang, J. Ma, and L. Jiao, "An efficient negative selection algorithm with further training for anomaly detection," *Knowledge-Based Systems*, vol. 30, pp. 185–191, 2012.
- [14] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 93–100, 1985.
- [15] C. M. Fonseca and P. J. Fleming, "Genetic algorithm for multi-objective optimization: formulation, discussion, and generation," in *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 416–423, 1993.
- [16] J. Horn, N. Nafpliotis, and D. E. Goldberg, "Niche Pareto genetic algorithm for multiobjective optimization," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pp. 82–87, June 1994.
- [17] N. Srinivas and K. Deb, "Multi-objective optimization using non-dominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.
- [18] C. A. Coello Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, 2006.
- [19] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.
- [20] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength Pareto evolutionary algorithm," in *Proceedings of Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pp. 95–100, 2002.
- [21] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelope-based selection algorithm for multiobjective optimization," in *Proceedings of the 6th Conference on Parallel Problem Solving from Nature*, pp. 839–848, 2000.
- [22] M. Erickson, A. Mayer, and J. Horn, "Multi-objective optimal design of groundwater remediation systems: application of the niched Pareto genetic algorithm (NPGA)," *Advances in Water Resources*, vol. 25, no. 1, pp. 51–65, 2002.
- [23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [24] C. García-Martínez, M. Lozano, and F. J. Rodríguez-Díaz, "A simulated annealing method based on a specialised evolutionary algorithm," *Applied Soft Computing Journal*, vol. 12, no. 2, pp. 573–588, 2012.
- [25] M. S. Arumugam and M. V. C. Rao, "On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 324–336, 2008.
- [26] C. Y. Chung, H. Yu, and K. P. Wong, "An advanced quantum-inspired evolutionary algorithm for unit commitment," *IEEE Transactions on Power Systems*, vol. 26, no. 2, pp. 847–854, 2011.
- [27] M. Karakose and U. Cigdem, "QPSO-based adaptive DNA computing algorithm," *The Scientific World Journal*, vol. 2013, Article ID 160687, 8 pages, 2013.
- [28] H. Kwasnicka and M. Przewozniczek, "Multi population pattern searching algorithm: a new evolutionary method based on the idea of messy genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 715–734, 2011.
- [29] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: a regularity model-based multiobjective estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, 2008.

- [30] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.
- [31] J. Chen, Q. Lin, and Z. Ji, "A hybrid immune multiobjective optimization algorithm," *European Journal of Operational Research*, vol. 204, no. 2, pp. 294–302, 2010.
- [32] J. Chen, Q. Lin, and Z. Ji, "Chaos-based multi-objective immune algorithm with a fine-grained selection mechanism," *Soft Computing*, vol. 15, no. 7, pp. 1273–1288, 2011.
- [33] K. Vijayalakshmi and S. Radhakrishnan, "A novel hybrid immune-based GA for dynamic routing to multiple destinations for overlay networks," *Soft Computing*, vol. 14, no. 11, pp. 1227–1239, 2010.
- [34] Z. Zhang, "Multiobjective optimization immune algorithm in dynamic environments and its application to greenhouse control," *Applied Soft Computing Journal*, vol. 8, no. 2, pp. 959–971, 2008.
- [35] M. Gong, L. Jiao, H. Du, and L. Bo, "Multiobjective immune algorithm with nondominated neighbor-based selection," *Evolutionary Computation*, vol. 16, no. 2, pp. 225–255, 2008.
- [36] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK, 2001.
- [37] C. A. Coello Coello, D. A. van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic, New York, NY, USA, 2002.
- [38] Z. Zhang, "Immune optimization algorithm for constrained nonlinear multiobjective optimization problems," *Applied Soft Computing Journal*, vol. 7, no. 3, pp. 840–857, 2007.
- [39] K. C. Tan, C. K. Goh, A. A. Mamun, and E. Z. Ei, "An evolutionary artificial immune system for multi-objective optimization," *European Journal of Operational Research*, vol. 187, no. 2, pp. 371–392, 2008.
- [40] E. Zitzler and L. Thiele, "Multi-objective optimization using evolutionary algorithms-A comparative study," in *Proceedings of the 5th Conference on Parallel Problem Solving from Nature*, pp. 292–301, 1998.
- [41] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.
- [42] H. G. Beyer and H. P. Schwefel, "Evolution strategies-A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.
- [43] S. Kukkonen and K. Deb, "A fast and effective method for pruning of nondominated solutions in many-objective problems," in *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature*, pp. 553–562, 2006.
- [44] D. A. Van Veldhuizen, *Multi-objective evolutionary algorithms: classifications, analyses, and new innovations [Ph.D. thesis]*, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, Ohio, USA, 1999.
- [45] J. R. Schott, *Fault tolerant design using single and multicriteria genetic algorithm optimization, [MA thesis]*, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, UK, 1995.
- [46] M. Laumanns, E. Zitzler, and L. Thiele, "Unified model for multi-objective evolutionary algorithms with elitism," in *Proceedings of the Congress on Evolutionary Computation (CEC '00)*, pp. 46–53, July 2000.
- [47] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.
- [48] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 825–830, 2002.
- [49] K. Deb and S. Jain, "Running performance metrics for evolutionary multiobjective optimization," in *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, pp. 13–20, 2002.
- [50] D. Yang, L. Jiao, M. Gong, and J. Feng, "Adaptive ranks clone and k-nearest neighbor list-based immune multi-objective optimization," *Computational Intelligence*, vol. 26, no. 4, pp. 359–385, 2010.

Research Article

Heterogeneous Differential Evolution for Numerical Optimization

Hui Wang,¹ Wenjun Wang,² Zhihua Cui,^{3,4} Hui Sun,¹ and Shahryar Rahnamayan⁵

¹ School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China

² School of Business Administration, Nanchang Institute of Technology, Nanchang 330099, China

³ Complex System and Computational Intelligent Laboratory, Taiyuan University of Science and Technology, Taiyuan 030024, China

⁴ State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

⁵ Department of Electrical, Computer and Software Engineering, University of Ontario Institute of Technology, 2000 Simcoe Street North Oshawa, ON, Canada L1H 7K4

Correspondence should be addressed to Hui Wang; huiwang@whu.edu.cn

Received 28 September 2013; Accepted 23 December 2013; Published 5 February 2014

Academic Editors: G. C. Gini and J. Zhang

Copyright © 2014 Hui Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Differential evolution (DE) is a population-based stochastic search algorithm which has shown a good performance in solving many benchmarks and real-world optimization problems. Individuals in the standard DE, and most of its modifications, exhibit the same search characteristics because of the use of the same DE scheme. This paper proposes a simple and effective heterogeneous DE (HDE) to balance exploration and exploitation. In HDE, individuals are allowed to follow different search behaviors randomly selected from a DE scheme pool. Experiments are conducted on a comprehensive set of benchmark functions, including classical problems and shifted large-scale problems. The results show that heterogeneous DE achieves promising performance on a majority of the test problems.

1. Introduction

Differential evolution (DE) [1] is a well-known algorithm for global optimization over continuous search spaces. Although DE has shown a good performance over many optimization problems, its performance is greatly influenced by its mutation scheme and control parameters (population size, scale factor, and crossover rate). To enhance the performance of DE, many improved DE variants have been proposed based on modified mutation strategies [2, 3] or adaptive parameter control [4–6].

The standard DE and most of its modifications [7–10] make use of homogeneous populations where all of the individuals follow exactly the same behavior. That is, individuals implement the same DE scheme, such as DE/rand/1/bin and DE/best/1/bin. The effect is that individuals in the population behave with the same exploration and/or exploitation characteristics [11]. An ideal optimization algorithm should balance exploration and exploitation during the search process. Initially, the algorithm should concentrate on exploration. As

the iteration increases, it would be better to use exploitation to find more accurate solutions. However, it is difficult to determine when the algorithm should switch from an explorative behavior to an exploitative behavior. To tackle this problem, a new concept of heterogeneous swarms [11] is proposed and applied to particle swarm optimization (PSO) [12, 13], where particles in the swarm use different velocity and position update rules. Therefore, the swarm may consist of explorative particles as well as exploitative particles. This makes the heterogeneous PSO have the ability to balance exploration and exploitation during the search process.

In this paper, a simple and effective heterogeneous DE (HDE) algorithm is proposed inspired by the idea of heterogeneous swarms [11]. In HDE, individuals will be allocated to different search behaviors randomly selected from a DE scheme pool. However, the concept of heterogeneous swarms used in DE is not new. Qin et al. [6] proposed a self-adaptive DE (SaDE), where individuals are also allowed to implement different mutation schemes according to a complex probability model. Gong et al. [14] combined

a strategy adaptation mechanism with four mutation strategies proposed in JADE [3]. For other self-adaptive DE variants [8], the search characteristics of individuals dynamically change according to the adaptation of the control parameters (scale factor and/or crossover rate). The HDE proposed in this paper differs from the above DE variants. In HDE, the behaviors are randomly assigned from a DE scheme pool. By the suggestions of heterogeneous PSO [11], two heterogeneous models are proposed. This first one is static HDE (sHDE), where the randomly selected behaviors are fixed during the evolution. The second one is dynamic HDE (dHDE), where the behaviors can randomly change during the search process. Experimental studies are conducted on a comprehensive set of benchmark functions, including classical problems and shifted large-scale problems. Simulation results demonstrate the efficiency and effectiveness of the proposed heterogeneous DE.

The rest of the paper is organized as follows. In Section 2, the standard DE algorithm is briefly introduced. The HDE is proposed in Section 3. Experimental simulations, results, and discussions are presented in Section 4. Finally, the work is concluded in Section 5.

2. Differential Evolution

There are several variants of DE [1], which use different mutation strategies and/or crossover schemes. To distinguish these different DE schemes, the notation “DE/*x/y/z*” is used, where “DE” indicates the DE algorithm, “*x*” denotes the vector to be mutated, “*y*” is the number of difference vectors used in the mutation, and “*z*” stands for the type of crossover scheme, exponential (exp) or binomial (bin). The following section discusses the mutation and crossover operations.

2.1. Mutation. For each vector $X_{i,G}$ at generation G , this operation creates mutant vectors $V_{i,G}$ based on the current parent population. The following are five well-known variant mutation strategies:

(1) DE/rand/1

$$V_{i,G} = X_{i_1,G} + F \cdot (X_{i_2,G} - X_{i_3,G}), \quad (1)$$

(2) DE/best/1

$$V_{i,G} = X_{\text{best},G} + F \cdot (X_{i_1,G} - X_{i_2,G}), \quad (2)$$

(3) DE/current-to-best/1

$$V_{i,G} = X_{i,G} + F \cdot (X_{\text{best},G} - X_{i,G}) + F \cdot (X_{i_1,G} - X_{i_2,G}), \quad (3)$$

(4) DE/rand/2

$$V_{i,G} = X_{i_1,G} + F \cdot (X_{i_2,G} - X_{i_3,G}) + F \cdot (X_{i_4,G} - X_{i_5,G}), \quad (4)$$

(5) DE/best/2

$$V_{i,G} = X_{\text{best},G} + F \cdot (X_{i_1,G} - X_{i_2,G}) + F \cdot (X_{i_3,G} - X_{i_4,G}), \quad (5)$$

where the indices i_1, i_2, i_3, i_4 , and i_5 are mutually different random indices chosen from the set $\{1, 2, \dots, N_p\}$, N_p is the population size, and all are different from the base index i . The scale factor F is a real number that controls the difference vectors. $X_{\text{best},G}$ is the best vector in terms of fitness value at the current generation G .

2.2. Crossover. Similar to EAs, DE also employs a crossover operator to build trial vectors $U_{i,G}$ by recombining the current vector $X_{i,G}$ and the mutant one $V_{i,G}$. The trial vector is defined as follows:

$$u_{i,j,G} = \begin{cases} v_{i,j,G}, & \text{if } \text{rand}_j(0, 1) \leq \text{CR} \vee j = j_{\text{rand}} \\ x_{i,j,G}, & \text{otherwise,} \end{cases} \quad (6)$$

where $\text{CR} \in (0, 1)$ is the predefined crossover probability, $\text{rand}_j(0, 1)$ is a uniform random number within $[0, 1]$ for the j th dimension, and $j_{\text{rand}} \in \{1, 2, \dots, D\}$ is a random index.

2.3. Selection. After the crossover, a greedy selection mechanism is used to select the better one from the parent vector $X_{i,G}$ and the trial vector $U_{i,G}$ according to their fitness values $f(\cdot)$. Without losing of generality, this paper only considers minimization problems. If, and only if, the trial vector $U_{i,G}$ is better than the parent vector $X_{i,G}$, then $X_{i,G+1}$ is set to $U_{i,G}$; otherwise, we keep $X_{i,G+1}$ the same with $X_{i,G}$:

$$X_{i,G+1} = \begin{cases} U_{i,G}, & \text{if } f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G}, & \text{otherwise.} \end{cases} \quad (7)$$

3. Heterogeneous DE

In the standard DE and most of its modifications, individuals in the population behave with the same search characteristics, exploration, and/or exploitation, because of the use of the same DE scheme. The effect is that the algorithms could hardly balance exploration and exploitation during the search process. Inspired by the heterogeneous swarms [11], a simple and effective heterogeneous DE (HDE) algorithm is proposed in this paper. Compared to DE and most of its variants, individuals in HDE are allowed to implement different behaviors.

To implement different DE schemes in HDE, we need to address two questions. First, which DE scheme should be chosen to construct the DE scheme pool? Second, how do we assign DE schemes to individuals?

As mentioned before, there are several DE schemes, and different DE schemes have different search characteristics. In this paper, three different DE schemes are chosen to construct the DE scheme pool: (1) DE/rand/1/bin; (2) DE/best/1/bin; and (3) DE/BoR/1/bin [16]. The first two schemes are two basic DE strategies proposed in [1], and the last one was recently proposed in [16], where a new mutation strategy called “best-of-random” (BoR) is defined as follows:

$$V_{i,G} = X_{i_b,G} + F \cdot (X_{i_1,G} - X_{i_2,G}), \quad (8)$$

where the individuals X_{i_1} , X_{i_2} , and X_{i_b} are randomly chosen from the current population, $i \neq i_1 \neq i_2 \neq i_b$, and X_{i_b} is the best one of them; that is, $f(X_{i_b}) \leq \min(f(X_{i_1}), f(X_{i_2}))$.

Variables	DE scheme
$X_{1,G}$	ID ₁
$X_{2,G}$	ID ₂
$X_{3,G}$	ID ₃
...	...
$X_{N_p,G}$	ID _{N_p}

FIGURE 1: The encoding of individuals in heterogeneous DE.

There are two reasons for choosing these DE schemes. First, these three DE schemes are very simple and easy to implement. Second, each of them has different search characteristics. For the DE/rand/1/bin, it obtains higher population diversity. DE/best/1/bin shows faster convergence speed. The last one provides a middle phase between the first two DE schemes. Note that this paper only chooses three DE schemes to construct the DE scheme pool and other DE schemes can also be possibly used.

Figure 1 presents the encoding method in HDE, where ID_{*i*} denotes the employed DE scheme for the *i*th individual. In this paper, ID_{*i*} ∈ {1, 2, 3}; that is, ID_{*i*} = 1 indicates the DE/rand/1/bin scheme, ID_{*i*} = 2 denotes the DE/best/1/bin scheme, and ID_{*i*} = 3 stands for the DE/BoR/1/bin scheme.

In order to address the second question, two different heterogeneous models, namely, static HDE (sHDE) and dynamic HDE (dHDE), are used to assign DE schemes to individuals [11].

- (i) In the static HDE (sHDE), DE schemes are randomly assigned to individuals in population initialization. The assigned DE schemes do not change during the search process.
- (ii) In the dynamic HDE (dHDE), DE schemes are randomly assigned to individuals during initialization. As the iteration increases, the assigned DE schemes of individuals are not fixed. They can randomly change during the search process. An individual randomly selects a new DE scheme from the DE scheme pool when the individual fails to improve its objective fitness value. In the standard DE and its variants, the objective fitness value of each individual X_i satisfies $f(X_{i,1}) \leq f(X_{i,2}) \leq \dots \leq f(X_{i,G})$. If the new generated individual (trail vector U_i) could not improve its previous position (X_i), it may indicate early stagnation. This can be addressed by assigning a new DE scheme to the individual.

The main steps of dynamic heterogeneous DE (dHDE) are presented in Algorithm 1, where P is the current population, FEs is the number of fitness evaluations, and MAX_FEs is

the maximum number of FEs. For static HDE (sHDE), please delete line 21 in Algorithm 1.

4. Experimental Verifications

This section provides experimental studies of HDE on 18 well-known benchmark optimization problems. According to the properties of these problems, two series of experiments are conducted: (1) comparison of HDE on classical optimization problems and (2) comparison of HDE on shifted large-scale optimization problems.

4.1. Results on Classical Optimization Problems. In this section, 12 classical benchmark problems are used to verify the performance of HDE. These problems were considered in [2, 5, 17–19]. Table 1 presents a brief description of these benchmark problems. All the problems are to be minimized.

Experiments are conducted to compare HDE with other six DE variants. The involved algorithms are listed below:

- (i) DE/rand/1/bin,
- (ii) DE/best/1/bin,
- (iii) DE/BoR/1/bin [16],
- (iv) self-adapting DE (jDE) [5],
- (v) DE with neighborhood search (NSDE) [20],
- (vi) DE using neighborhood-based mutation (DEGL) [2],
- (vii) the proposed sHDE and dHDE.

In the experiments, we have two series of comparisons: (1) comparison of sHDE/dHDE with basic DE schemes and (2) comparison of sHDE/dHDE with state-of-the-art DE variants. The first comparison aims to check whether the heterogeneous method is helpful to improve the performance of DE. The second comparison investigates whether the proposed approach is better or worse than some recently proposed DE variants.

For these two comparisons, we use the same parameter settings as follows. For the three basic DE schemes (DE/rand/1/bin, DE/best/1/bin, and DE/BoR/1/bin) and sHDE/dHDE,

TABLE 1: The 12 classical benchmark optimization problems

Problem	Name	D	Properties	Search range
f_1	Sphere	25	Unimodal	$[-100, 100]$
f_2	Schewefel 2.22	25	Unimodal	$[-10, 10]$
f_3	Schewefel 1.2	25	Unimodal	$[-100, 100]$
f_4	Schewefel 2.21	25	Unimodal	$[-100, 100]$
f_5	Rosenbrock	25	Multimodal	$[-30, 30]$
f_6	Step	25	Unimodal	$[-100, 100]$
f_7	Quartic with noise	25	Unimodal	$[-1.28, 1.28]$
f_8	Schewefel 2.26	25	Multimodal	$[-500, 500]$
f_9	Rastrigin	25	Multimodal	$[-5.12, 5.12]$
f_{10}	Ackley	25	Multimodal	$[-32, 32]$
f_{11}	Griewank	25	Multimodal	$[-600, 600]$
f_{12}	Penalized	25	Multimodal	$[-50, 50]$

```

(1) Randomly initialize the population  $P$  including the variables and ID;
(2) Evaluate the objective fitness value of each individuals in  $P$ ;
(3)  $FES = N_p$ ;
(4) while  $FES \leq MAX\_FES$  do
(5)   for  $i = 1$  to  $N_p$  do
(6)     if  $ID_i == 1$  then
(7)       Use DE/rand/1/bin to generate a trail vector  $U_i$ ;
(8)     end
(9)     if  $ID_i == 2$  then
(10)      Use DE/best/1/bin to generate a trail vector  $U_i$ ;
(11)    end
(12)    if  $ID_i == 3$  then
(13)      Use DE/BoR/1/bin to generate a trail vector  $U_i$ ;
(14)    end
(15)    Evaluate the objective fitness value of  $U_i$ ;
(16)     $FES++$ ;
(17)    if  $f(U_i) \leq f(X_i)$  then
(18)       $X_i = U_i$ 
(19)    end
(20)    else
/* For static HDE (sHDE), please delete lines 20–22 */
(21)      Randomly assign a new different DE scheme to  $X_i$ , (change the value of  $ID_i$ );
(22)    end
(23)  end
(24) end

```

ALGORITHM 1: The dynamic heterogeneous DE (dHDE).

the control parameters, F and CR, are set to 0.5 and 0.9, respectively [5]. For jDE, NSDE, and DEGL, the parameters F and CR are self-adaptive. For all algorithms, the population size (N_p) and maximum number of FEs are set to $10 \cdot D$ and $5.00E+05$, respectively [2]. All the experiments are conducted 30 times, and the mean error fitness values are reported.

4.1.1. Comparison of HDE with Basic DE Schemes. The comparison results of HDE with the three basic DE schemes are presented in Table 2, where “Mean” denotes the mean error fitness values. From the results, sHDE and dHDE outperform other three basic DE schemes on a majority of test problems. For unimodal problems ($f_1 - f_4$), DE/best/1/bin shows faster

convergence than other algorithms for the attraction of the global best individual. For f_6 and f_7 , all the algorithms obtain similar performance. DE/rand/1/bin provides high population diversity but slows down convergence rate. That is why DE/rand/1/bin performs better than DE/best/1/bin on most multimodal problems, but worse on unimodal problems. DE/BoR/bin/1 is a middle phase of DE/rand/1/bin and DE/best/1/bin. It obtains better performance than the other two basic DE schemes. For the heterogeneity of these basic DE schemes, sHDE and dHDE significantly improve the performance of DE. For unimodal problems, DE/best/1/bin obtains the best performance, while sHDE and dHDE perform better than DE/rand/1/bin and DE/BoR/1/bin. For

TABLE 2: Comparison of HDE with basic DE schemes on classical benchmark problems.

Problem	DE/rand/1/bin	DE/best/1/bin	DE/BoR/1/bin	sHDE	dHDE
	Mean	Mean	Mean	Mean	Mean
f_1	0.00E + 00	4.38E - 27	2.37E - 46	6.58E - 79	0.00E + 00
f_2	0.00E + 00	2.64E - 13	3.02E - 22	3.32E - 39	4.96E - 103
f_3	2.68E - 125	2.41E - 04	1.51E - 11	3.10E - 21	7.21E - 57
f_4	5.34E - 64	1.77E - 07	1.89E - 11	1.42E - 07	2.49E - 40
f_5	5.75E - 29	2.91E - 04	1.45E - 16	1.11E - 29	1.73E - 29
f_6	0.00E + 00	0.00E + 00	0.00E + 00	0.00E + 00	0.00E + 00
f_7	1.38E - 03	2.50E - 03	1.52E - 03	8.31E - 04	1.84E - 03
f_8	3.36E + 03	1.68E + 03	2.78E + 03	0.00E + 00	8.56E + 02
f_9	4.38E + 01	1.32E + 02	2.62E + 01	5.22E + 01	0.00E + 00
f_{10}	2.81E + 00	2.55E - 14	4.14E - 15	4.14E - 15	4.14E - 15
f_{11}	2.95E - 02	0.00E + 00	0.00E + 00	0.00E + 00	0.00E + 00
f_{12}	9.96E - 01	5.00E - 28	1.88E - 32	1.88E - 32	1.88E - 32

TABLE 3: Comparison of HDE with jDE, NSDE, and DEGL on classical benchmark problems.

Problem	jDE	NSDE	DEGL	sHDE	dHDE
	Mean	Mean	Mean	Mean	Mean
f_1	4.04E - 35	9.55E - 35	8.78E - 37	6.58E - 79	0.00E + 00
f_2	8.34E - 26	8.94E - 30	4.95E - 36	3.32E - 39	4.96E - 103
f_3	4.76E - 14	3.06E - 09	1.21E - 26	3.10E - 21	7.21E - 57
f_4	3.02E - 14	2.09E - 11	4.99E - 15	1.42E - 04	2.49E - 40
f_5	5.64E - 26	2.65E - 25	6.89E - 25	1.11E - 29	1.73E - 29
f_6	1.67E - 36	4.04E - 28	9.56E - 48	0.00E + 00	0.00E + 00
f_7	3.76E - 02	4.35E - 03	1.05E - 07	8.31E - 04	1.84E - 03
f_8	0.00E + 00	2.60E + 00	0.00E + 00	0.00E + 00	8.56E + 02
f_9	6.74E - 24	4.84E - 21	5.85E - 25	5.22E + 01	0.00E + 00
f_{10}	7.83E - 15	5.97E - 10	5.98E - 23	4.14E - 15	4.14E - 15
f_{11}	1.83E - 28	7.93E - 26	2.99E - 36	0.00E + 00	0.00E + 00
f_{12}	9.37E - 24	5.85E - 21	7.21E - 27	1.88E - 32	1.88E - 32

multimodal problems, DE/best/1/bin is the worst algorithm, while sHDE and dHDE obtain better performance than other algorithms.

For the comparison of sHDE and dHDE, sHDE performs better on 3 problems, while dHDE achieves better results on 5 problems. For the remaining 4 problems, both of them could search the global optimum. Although the dynamic heterogeneous model improves the performance of HDE on many problems, it may lead to premature convergence. In the dHDE, if the current individual could not improve its fitness value, a new DE scheme is assigned to the individual. If the new scheme is DE/best/1/bin, the individual will be attracted by the global best individuals. This will potentially run the risk of premature convergence.

4.1.2. Comparison of HDE with Other State-of-the-Art DE Variants. The comparison results of HDE with other three state-of-the-art DE variants are presented in Table 3, where “Mean” denotes the mean error fitness values. From the results, sHDE and dHDE perform better than other three DE algorithms on the majority of test problems. dHDE

TABLE 4: Average rankings achieved by Friedman test.

Algorithms	Ranking
dHDE	4.17
sHDE	3.58
DEGL	3.50
jDE	2.25
NSDE	1.50

outperforms jDE and NSDE on all test problems except for f_8 . On this problem, dHDE falls into the local minima, while sHDE could successfully solve it. dHDE achieves better results than DEGL on 9 problems, while DEGL performs better than dHDE on the remaining 3 problems.

In order to compare the performance of multiple algorithms on the test suite, we conducted Friedman test according to the suggestions of [21]. Table 4 shows the average ranking of jDE, NSDE, DEGL, sHDE, and dHDE. As seen, the performance of the five algorithms can be sorted by average ranking into the following order: dHDE, sHDE, DEGL,

TABLE 5: The 6 shifted large-scale benchmark optimization problems proposed in [15].

Problem	Name	D	Properties	Search range
F_1	Shifted Sphere	500	Unimodal, separable, scalable	$[-100, 100]$
F_2	Shifted Schewefel 2.21	500	Unimodal, nonseparable	$[-100, 100]$
F_3	Shifted Rosenbrock	500	Multimodal, nonseparable	$[-100, 100]$
F_4	Shifted Rastrigin	500	Multimodal, separable	$[-5, 5]$
F_5	Shifted Griewank	500	Multimodal, nonseparable	$[-600, 600]$
F_6	Shifted Ackley	500	Multimodal, separable	$[-32, 32]$

TABLE 6: Comparison of HDE with ODE and MDE on shifted large-scale benchmark problems.

Problem	ODE	MDE	sHDE	dHDE
	Mean	Mean	Mean	Mean
F_1	$8.02E + 01$	$1.95E + 01$	$8.45E + 00$	$1.14E - 10$
F_2	$5.78E + 00$	$2.70E + 01$	$8.54E + 01$	$1.01E + 02$
F_3	$1.54E + 05$	$4.67E + 05$	$1.52E + 05$	$1.29E + 03$
F_4	$4.22E + 03$	$4.14E + 03$	$5.27E + 03$	$3.42E + 03$
F_5	$1.77E + 00$	$1.52E + 00$	$6.94E - 01$	$1.83E - 01$
F_6	$4.51E + 00$	$4.02E + 00$	$1.26E + 00$	$1.58E + 01$

jDE, and NSDE. The best average ranking was obtained by the dHDE algorithm, which outperforms the other four algorithms.

4.2. Results of HDE on Shifted Large-Scale Optimization Problems. To verify the performance of HDE on complex optimization problems, this section provides experimental studies of HDE on 6 shifted large-scale problems. These problems were considered in CEC 2008 special session and competition on large-scale global optimization [15]. Table 5 presents a brief descriptions of these benchmark problems. All the problems are to be minimized.

Experiments are conducted to compare four DE algorithms including the proposed sHDE and dHDE on 6 test problems with $D = 500$. The involved algorithms are listed below:

- (i) opposition-based DE (ODE) [22],
- (ii) modified DE using Cauchy mutation (MDE) [23],
- (iii) the proposed sHDE and dHDE.

To have a fair comparison, all the four algorithms use the same parameter settings by the suggestions of [22]. The population size N_p is set to D . The control parameters, F and CR, are set to 0.5 and 0.9, respectively [22]. For ODE, the rate of opposition J_r is 0.3. The maximum number of FEs is $5000 \cdot D$. All the experiments are conducted 30 times, and the mean error fitness values are reported.

Table 6 presents the comparison results of HDE with ODE and MDE on shifted large-scale problems. As seen, sHDE and dHDE outperform ODE and MDE on four problems. The static heterogeneous model slightly improves the final solutions of DE, while the dynamic model significantly improves the results on F_1 and F_3 . However, the two heterogeneous models do not always work. For some problems, sHDE and/or dHDE perform worse than ODE and MDE. The

possible reason is that the heterogeneous models may hinder the evolution. For the static model, the entire population is divided into three groups. Each group uses a different basic DE scheme to generate new individuals. Though these groups share the search information of the entire population, each group with small population may obtain slow convergence rate. As mentioned before, the dynamic heterogeneous model runs the risk of premature convergence.

5. Conclusion

In the standard DE and most of its modifications, individuals follow the same search behavior for using the same DE scheme. The effect is that the algorithms could hardly balance explorative and exploitative abilities during the evolution. Inspired by the heterogeneous swarms, a simple and effective heterogeneous DE (HDE) is proposed in this paper, where individuals could follow different search characteristics randomly selected from a DE scheme pool. To implement the HDE, two heterogeneous models are proposed, one static, where individuals do not change their search behaviors, and a dynamic model where individuals may change their search behaviors. Both versions of HDE initialize individual behaviors by randomly selecting DE schemes from a DE scheme pool. In the dynamic HDE, when an individual does not improve its fitness value, a new different DE scheme is randomly selected from the DE scheme pool.

To verify the performance of HDE, two types of benchmark problems, including classical problems and shifted large-scale problems, are used in the experiments. Simulation results show that the proposed sHDE and dHDE outperform the other eight DE variants on a majority of test problems. It demonstrates that the heterogeneous models are helpful to improve the performance of DE.

For the dynamic heterogeneous model, the frequency of changing new DE schemes may affect the performance of

dHDE. More experiments will be investigated. The concept of heterogeneous swarms can be applied to other evolutionary algorithms to obtain good performance. Future research will investigate different algorithms based on heterogeneous swarms. Moreover, we will try to apply our approach to some real-world problems [24, 25].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the Humanity and Social Science Foundation of Ministry of Education of China (no. 13YJCZH174), the National Natural Science Foundation of China (nos. 61305150, 61261039, and 61175127), the Science and Technology Plan Projects of Jiangxi Provincial Education Department (no. GJJ13744 and GJJ13762), and the Foundation of State Key Laboratory of Software Engineering (no. SKLSE2012-09-19).

References

- [1] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
- [2] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.
- [3] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.
- [4] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.
- [5] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.
- [6] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.
- [7] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61–106, 2010.
- [8] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.
- [9] X. S. Yang and S. Deb, "Two-stage eagle strategy with differential evolution," *International Journal of Bio-Inspired Computation*, vol. 4, no. 1, pp. 1–5, 2012.
- [10] Y. W. Zhong, L. J. Wang, C. Y. Wang, and H. Zhang, "Multi-agent simulated annealing algorithm based on differential evolution algorithm," *International Journal of Bioinspired Computation*, vol. 4, no. 4, pp. 217–228, 2012.
- [11] A. P. Engelbrecht, "Heterogeneous particle swarm optimization," *International Conference on Swarm Intelligence*, vol. 6234, pp. 191–202, 2010.
- [12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.
- [13] E. GarGarcíaI-Gonzalo and J. L. Fernández-Martínez, "A brief historical review of particle swarm optimization (PSO)," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 3–16, 2012.
- [14] W. Gong, Z. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 41, no. 2, pp. 397–413, 2011.
- [15] K. Tang, X. Yao, P. N. Suganthan et al., Benchmark functions for the CEC' 2008 special session and competition on high-dimensional real-parameter optimization, Nature Inspired Computation and Applications Laboratory, USTC, Hefei, China, 2007.
- [16] C. Lin, A. Qing, and Q. Feng, "A new differential mutation base generator for differential evolution," *Journal of Global Optimization*, vol. 49, no. 1, pp. 69–90, 2011.
- [17] L. Ali and S. L. Sabat, "Particle swarm optimization based universal solver for global optimization," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 95–105., 2012.
- [18] X. J. Cai, S. J. Fan, and Y. Tan, "Light responsive curve selection for photosynthesis operator of APOA," *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 373–379, 2012.
- [19] L. P. Xie, J. C. Zeng, and R. A. Formato, "Selection strategies for gravitational constant G in artificial physics optimisation based on analysis of convergence properties," *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 380–391, 2012.
- [20] Z. Yang, J. He, and X. Yao, "Making a difference to differential evolution," *Advance in Metaheuristics for Hard Optimization*, pp. 397–414, 2008.
- [21] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.
- [22] R. S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.
- [23] M. Ali and M. Pant, "Improving the performance of differential evolution algorithm using Cauchy mutation," *Soft Computing*, vol. 15, no. 5, pp. 991–1007, 2011.
- [24] A. Y. Abdelaziz, R. A. Osama, and S. M. Elkhodary, "Application of ant colony optimization and harmony search algorithms to reconfiguration of radial distribution networks with distributed generations," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 1, pp. 86–94, 2012.
- [25] S. Y. Zeng, Z. Liu, C. H. Li, Q. Zhang, and W. Wang, "An evolutionary algorithm and its application in antenna design," *Journal of Bioinformatics and Intelligent Control*, vol. 1, no. 2, pp. 129–137, 2012.

Research Article

Correction of Faulty Sensors in Phased Array Radars Using Symmetrical Sensor Failure Technique and Cultural Algorithm with Differential Evolution

S. U. Khan,¹ I. M. Qureshi,^{2,3} F. Zaman,⁴ B. Shoaib,⁴ A. Naveed,⁴ and A. Basit⁴

¹ School of Engineering & Applied Sciences, ISRA University, Islamabad 44000, Pakistan

² Electrical Department, Air University, Islamabad 44000, Pakistan

³ Institute of Signals, Systems and Soft Computing (ISSS), Islamabad 44000, Pakistan

⁴ Electronic Engineering Department, IIU, H-10, Islamabad 44000, Pakistan

Correspondence should be addressed to S. U. Khan; shafqatphy@yahoo.com

Received 30 August 2013; Accepted 17 November 2013; Published 29 January 2014

Academic Editors: Z. Cui and X. Yang

Copyright © 2014 S. U. Khan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Three issues regarding sensor failure at any position in the antenna array are discussed. We assume that sensor position is known. The issues include raise in sidelobe levels, displacement of nulls from their original positions, and diminishing of null depth. The required null depth is achieved by making the weight of symmetrical complement sensor passive. A hybrid method based on memetic computing algorithm is proposed. The hybrid method combines the cultural algorithm with differential evolution (CADE) which is used for the reduction of sidelobe levels and placement of nulls at their original positions. Fitness function is used to minimize the error between the desired and estimated beam patterns along with null constraints. Simulation results for various scenarios have been given to exhibit the validity and performance of the proposed algorithm.

1. Introduction

In adaptive beamforming, null steering and beam steering are hot research areas. It has direct application in radar, sonar, and mobile communication [1–3]. In the literature various analytical and computational methods are available to concentrate on the issue of null steering [4–6]. The condition becomes more demanding and complicated when a sensor fails in the active antenna array. The excitation of these sensors is to accomplish desired radiation pattern. In case of sensor failure, the sidelobe level (SLL) raise and nulls are displaced, which is highly unwanted. It is very expensive in terms of time and budget to replace the defective sensor regularly. Hence the weights of active sensors in the same array should be recalculated and readjusted to create a new pattern close to the original one. Recently few algorithms have been proposed to correct the damaged pattern of the array [7–10].

In the last few decades Radar technology has developed very rapidly. The radar commonly used nowadays is known

as phased array radar. In this radar the whole input array transmits the same signal with different delay and a beam is formed towards the area of interest [11]. The advantages of beam include the electronic steering instead of mechanical steering and a high processing gain at the transmitter. The phased array radar used the phase shifting in the input waveform to steer a beam electronically in the direction of the target instead of mechanical steering. Array design is one of the most active research area in phased array radars in which the sensors are arranged together to form an array. The phase shifters adjust the phase in such a way that a beam is formed in the desired direction. The width of the beam depends on the number of sensors in the array. By increasing the number of sensors in an array, the beam becomes sharper and thus more efficient in detecting the targets with smaller size. Now if one or more sensors become damaged, the radars cannot detect the target correctly. Researchers are still working to recalculate and adjust the weights of the active array to get the pattern near the original one. By recalculating the weights of

the faulty array, the radar will improve its capabilities in such a way that the radar can perform searching, tracking, and weapon guidance at the same time.

The evolutionary computing technique has been doing well in solving numerous problems in search and optimization due to the impartial nature of their operations which can still be present in situations with no domain knowledge. The search method used by evolutionary algorithms (EAs) is impartial having no domain knowledge to guide the search method. Domain knowledge serves as a method to reduce the search space by pruning unnecessary parts of the solution space and by promoting required parts. Cultural algorithm is based on the principle to bias the search method with prior knowledge about the domain, as well as the knowledge during the evolution method.

Among EC techniques, differential evolution (DE) is considered to be one of the powerful and reliable tools to optimize the problems in any engineering field [12–14]. The DE is a technique based on stochastic searches, in which function parameters are programmed as floating-point variables. The DE algorithm has a simple structure, convergence speed, flexibility, and robustness, with only some parameters required to be put by a user. The application of CAs in DE is a different strategy to get the performance and local search better. The previous work on null steering in failed antenna arrays is presented in [15]. The technique tries to restore the previous nulls pattern by using particle swarm optimization (PSO). All the above EC based techniques have discussed the SLL reduction and null steering in failed array, but no one is solving the issue of null depth and null steering at their original positions using CADE for the correction of faulty arrays. Authors in their previous work [16] have used the symmetrical element failure technique to achieve the required null depth level and first null beamwidth and [17] for fault finding in failed array antenna. Memetic computing algorithms are stochastic population based methods that have been established to be dominant and forceful to solve optimization problems. The advantages of cultural algorithm (CA) with evolutionary algorithms (EAs) include global search capability and consistent performance in any field of engineering and technology [18–20].

In this paper, the proposed algorithm developed three issues in case of sensor failure. These are raised in sidelobe levels, displacement of nulls from their original positions, and diminishing of null depth. We propose a symmetrical sensor failure (SSF) method that provides better results in terms of null depth. Moreover, the SSF method has deeper first null which is another big improvement over single sensor failure. The first null depth in beamforming is of great importance. To address the other two issues, we have used a cultural algorithm with differential evolution (CADE) to reduce the sidelobe levels and positions of nulls reverse to their original positions by adjusting the current weights of active sensors. A hybrid method based on the memetic computing algorithm is proposed, which combines the cultural algorithm with differential evolution (CADE) for the reduction of sidelobes and placement of nulls. Different simulation results are provided to confirm the performance of the proposed approach. The rest of the paper is organized as follows.

The problem formulation is discussed in Section 2, while in Section 3 the proposed solution is provided. Section 4 describes the simulations and the results while Section 5 concludes the paper and proposes some future work.

2. Problem Formulation

Consider a linear array of 17 sensors in which all the sensors are placed symmetrically about the origin. The total number of sensors is $N = 2M + 1$. The array factor in this healthy set up with uniformly spaced sensors, nonuniform weight, and progressive phase excitation will be [21]

$$AF(\theta_i) = \sum_{n=-M}^M w_n \exp(jn(kd \cos \theta_i + \beta)), \quad (1)$$

where w_n is the nonuniform weight of n th sensor whereas $n = 0, \pm 1, \pm 2, \dots, \pm M$. The spacing between the adjacent sensors is d , while θ is the angle from broadside. $k = 2\pi/\lambda$ is the wave number with λ as wavelength. The progressive phase shift $\beta = -kd \cos \theta_s$ and θ_s is steering angle for the main beam. The damage array factor for 7th sensor failure is given by the expression below:

$$AF(\theta_i) = \sum_{\substack{n=-M \\ n \neq 7}}^M w_n \exp(jn(kd \cos \theta_i + \beta)). \quad (2)$$

The nonuniform weights of 17 sensors with 7th symmetry sensor failures are as follows:

$$[w_{-8}, w_{-7}, w_{-6}, w_{-5}, w_{-4}, \dots, w_{-1}, w_0, w_1, \dots, w_4, w_5, w_6, w_7, w_8]. \quad (3)$$

It is assumed that the w_7 sensor fails in the antenna array given in (3). One can clearly monitor from Figure 1 that due to single sensor failure the radiation pattern is damaged in terms of sidelobe levels, null depth, and displacement of the nulls from their original position. So, the goal of this job is to recover the null depth, sidelobe levels, and null steering at their original positions. Different methods are available in the literature to correct the damage pattern of sensor failures; however, none of them is able to achieve the required null depth level.

3. Proposed Solution

In this section, we develop the proposed solution based on SSF. As we had assumed the damage of w_7 sensor, we lost the null depth as given in Figure 1. For SSF method we also force the sensor w_{-7} to be zero as shown in (3). From Figure 2, it is clear that symmetric sensor failure maintains the null depth almost as close to that of the original array. The damage array factor for 7th symmetrical sensor failure is given by

$$AF(\theta_i) = \sum_{\substack{n=-M \\ n \neq \pm 7}}^M w_n \exp(jn(kd \cos \theta_i + \beta)). \quad (4)$$

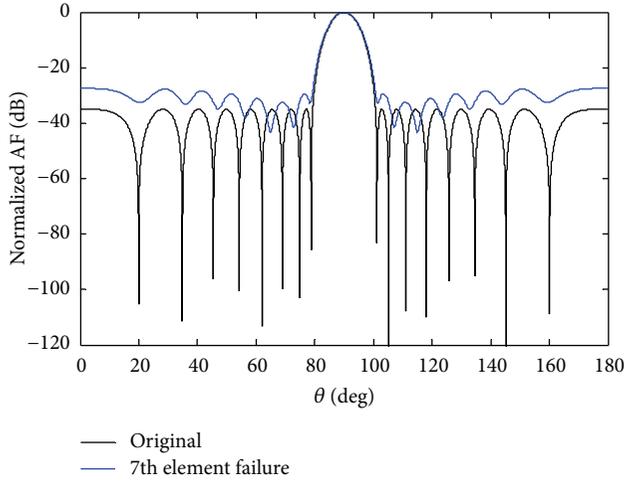


FIGURE 1: The original Chebyshev array and the w_7 sensor damage pattern.

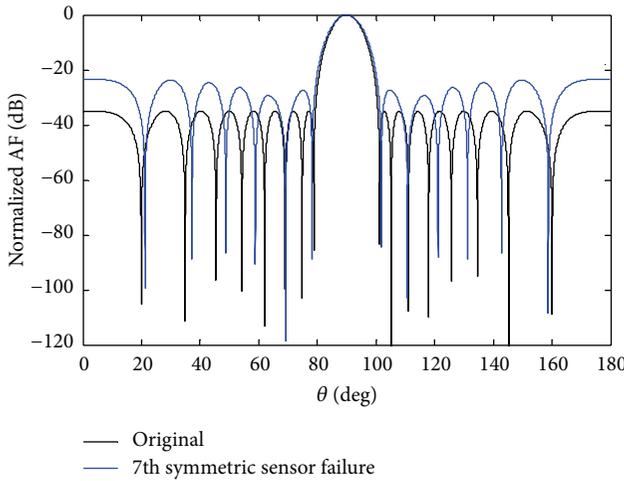


FIGURE 2: The original Chebyshev array and the w_7 symmetric sensor failure.

Although we have achieved better null depth level due to SSF, but the sidelobe levels and positioning of nulls are still a problem to be taken into account, for which we will use the cultural algorithm with differential evolution (CADE) for the reduction of sidelobes and placement of nulls in the required positions.

3.1. Differential Evolution (DE). DE is an EA and was developed by Storn and Price which is used to solve real valued optimization problems [22]. The DE is a method based on stochastic searches. The DE algorithm presents easy structure, convergence speed, flexibility, and robustness, with only some parameters required to be set by a user. However, this faster convergence of DE results in a higher possibility of searching near a local optimum or getting early convergence. Differential evolution is based on a mutation operator, which adds an amount obtained by the difference of two randomly chosen individuals of the present population. The problem to

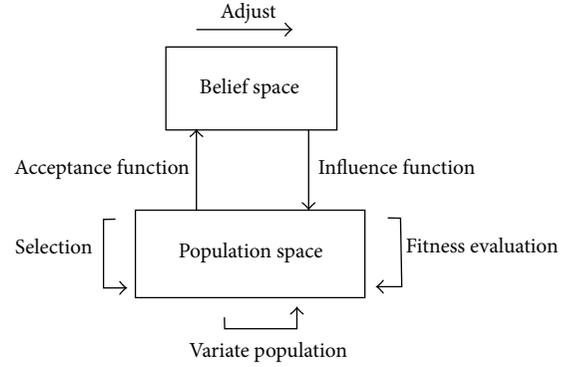


FIGURE 3: The flow diagram of the cultural algorithm.

TABLE 1: Parameter used in CADE.

CADE	
Parameters	Setting
Population size	300
Number of generation	500
Value of F	0.5
Value of CR	$0.5 \leq CR \leq 1$

be solved has N decision variables; F and CR are parameters given by the user and given in Table 1. Computing the difference between two individuals which are selected randomly from the population, in fact the algorithm estimating the gradient in that region and this technique is a proficient way to self-adapt the mutation operator. The pseudocode for the DE is given in Pseudocode 1.

3.2. The Cultural Algorithm with Differential Evolution (CADE). Differential evolution is used as a population space in the cultural algorithm. CAs have been developed to model the evolution of the cultural component of an evolutionary computational system over time as it accumulates experience. As a result, CAs can provide a clear method for global knowledge and a framework within which to model self-adaptation in an evolutionary system.

Cultural algorithms consist of three components, a population space, belief space, and a communication protocol as shown in Figure 3. First one is population space that contains the population to be evolved and the mechanisms for its estimate. The population space consists of a set of possible solutions to the problem; in our problem, the population space is DE. Second, one is a belief space that represents the bias that has been acquired by the population during its problem solving process. In CAs, the information acquired by a member of the population can be shared with the entire population. The third one is the communication protocol that is used to determine the interface between the population and the beliefs.

CAs model has two levels of evolution. One is the population level and the other is belief space level. In addition to a population space, CA has a belief space in which the beliefs acquired from the population's evolution can be stored and

```

Generate the initial population of individuals
Do
  For each individual j in the population
    Choose three numbers  $n_1, n_2,$  and  $n_3$  that is,  $1 \leq n_1, n_2, n_3 \leq N$  with  $n_1 \neq n_2 \neq n_3 \neq j$ 
    Generate a random integer  $i_{\text{rand}} \in (1, N)$ 
    For each parameter i
       $y^{i,g} = x^{n_1,g} + F(x^{n_2,g} - x^{n_3,g})$ 
       $z_j^{i,g} = \begin{cases} y_j^{i,g} & \text{if } \text{rand}() \leq \text{CR or } j = j_{\text{rand}} \\ x_j^{i,g} & \text{otherwise} \end{cases}$ 
    End For
    Replace  $x^{i,g}$  with the child  $z^{i,g}$  if  $z^{i,g}$  is better
  End For
Until the termination condition is achieved

```

PSEUDOCODE 1: The pseudocode of the differential evolution algorithm.

integrated. An acceptance function is used to generate beliefs by gleaning the experience of individuals from the population space. In return, this belief space can bias the evolution of the population component by means of the influence function. The belief space itself also evolves by the adjust function [23]. In the present work, the belief space is divided into two knowledge components.

3.2.1. Situational Knowledge. Situational knowledge stores individuals from population space which provides the direction for other individuals. Situational knowledge consists of the best example P found along the evolutionary process. It represents a leader for the other individuals to follow. The variation operators of differential evolution are influenced in the following way

$$y^{i,g} = P_i + F(x^{n_2,g} - x^{n_3,g}), \quad (5)$$

where P_i is the i th component of the individual stored in the situational knowledge. This way, we use the leader instead of a randomly chosen individual for the recombination, getting the children closer to the best point found. The update of the situational knowledge is done by replacing the stored individual P , by the best individual found in the current population, x_{best} only if x_{best} is better than P .

3.2.2. Normative Knowledge. The normative knowledge contains the intervals for the decision variables where good solutions have been found, in order to move new solutions towards those intervals. The normative knowledge includes a scaling factor, ds_i to influence the mutation operator adopted in differential evolution. The following expression shows the influence of the normative knowledge of the variation operators

$$z_j^{i,g} = \begin{cases} x^{n_1,g} + F(x^{n_2,g} - x^{n_3,g}) & \text{if } x^{n_1,g} < l_i \\ x^{n_1,g} - F(x^{n_2,g} - x^{n_3,g}) & \text{if } x^{n_1,g} > u_i \\ x^{n_1,g} + \left(\frac{u_i - l_i}{ds_i}\right) * F(x^{n_2,g} - x^{n_3,g}) & \text{otherwise,} \end{cases} \quad (6)$$

where l_i and u_i are the lower and upper bounds, respectively, for the i th decision variable. The update of the normative knowledge can reduce or expand the intervals stored on it. An extension takes place when the accepted individuals do not fit in the current interval, while a reduction occurs when all the accepted individuals lie in the current interval, and the extreme values have an improved fit and are feasible. The values ds_i are updated with the difference $(x^{n_2,g} - x^{n_3,g})$ found of the variation operators of the prior generation. Normative knowledge leads individuals to jump into the good range if they are not already there. The normative knowledge is updated as follows, let us consider $x_{a1}, x_{a2}, x_{a3}, \dots, x_{a n_{\text{accepted}}}$ be the accepted individuals in the current generation and x_{min_i} and x_{max_i} belong to $(a1, a2, a3, \dots, n_{\text{accepted}})$ and the accepted individuals with minimum and maximum values for the parameter i :

$$u_i = \begin{cases} x_{i,\text{max}_i} & \text{if } x_{i,\text{max}_i} > u_i \text{ or } f(x_{\text{max}_i}) < U_i \\ u_i & \text{otherwise,} \end{cases} \quad (7)$$

$$l_i = \begin{cases} x_{i,\text{min}_i} & \text{if } x_{i,\text{min}_i} < l_i \text{ or } f(x_{\text{min}_i}) < L_i \\ l_i & \text{otherwise.} \end{cases}$$

If l_i and u_i are updated, the values of L_i and U_i will be done in the same way. The ds_i are updated with the greatest difference of $|x_{i,r1} - x_{i,r2}|$ found during the variation operators at the prior generation.

The flow chart and pseudocode for CADE is shown in Pseudocode 2 and Figure 3.

3.3. Null Constraint (NC). A jamming signal located at a particular angle wants to be eliminated, in case of satellite, radar, and mobile communication applications. For a uninformed array, to put a null at a particular angle θ_i , we want [24]

$$\text{AF}(\theta_i) = \mathbf{w}^H \mathbf{s}(\theta_i) = 0, \quad (8)$$

```

Generate the initial population
Calculate the initial population
Initialize the belief space
Do
  For each individual in the population
    Apply the variation operator influenced by a randomly knowledge component
    Calculate the child generated
    Replace the individual with child, if the child is better
  End for
  Update the belief space with the accepted individuals
Until the termination condition is achieved
    
```

PSEUDOCODE 2: The pseudocode of the cultural algorithm.

where

$$\mathbf{s}(\theta_i) = \begin{bmatrix} \exp\left(-j\left(\frac{N-1}{2}\right)kd \cos \theta_i\right) \\ \exp\left(-j\left(\frac{N-3}{2}\right)kd \cos \theta_i\right) \\ \vdots \\ \exp\left(j\left(\frac{N-3}{2}\right)kd \cos \theta_i\right) \\ \exp\left(j\left(\frac{N-1}{2}\right)kd \cos \theta_i\right) \end{bmatrix}_{N \times 1} \tag{9}$$

and \mathbf{w}^H is $N \times 1$ vector which is defined as

$$\mathbf{w} = [w_{-M}, w_{-M+1}, \dots, w_0, \dots, w_{M-1}, w_M]^T \tag{10}$$

The null constraint is given as

$$\mathbf{w}^H \mathbf{s}(\theta_i) = 0, \quad i = 1, 2, \dots, M_0. \tag{11}$$

We may define an $N \times M_0$ constraint matrix \mathbf{C} as

$$\mathbf{C} = [\mathbf{s}(\theta_1), \mathbf{s}(\theta_2), \dots, \mathbf{s}(\theta_{M_0})], \tag{12}$$

where θ_i for $i = 1, 2, \dots, M_0$ is the direction of null. Our goal is to optimize the squared weighting error subject to the condition that

$$\mathbf{w}^H \mathbf{C} = \mathbf{0}. \tag{13}$$

Our constraint is that the columns of \mathbf{C} should be orthogonal to the weight vector \mathbf{w} . Accordingly we may define G_i , $i = 1, 2$ and G as follows:

$$G_1 = \sum_{i=1}^P [|\text{AF}_d(\theta_i) - \text{AF}_{\text{CADE}}(\theta_i)|]^2, \tag{14}$$

$$G_2 = \|\mathbf{w}^H \mathbf{C}\|^2, \tag{15}$$

$$G = G_1 + G_2. \tag{16}$$

Hence G is the fitness function for the problem given above which are to be minimized. Best chromosome will give the minimal value of G . The first term in (16) is used for SLL reduction, where $\text{AF}_d(\theta_i)$ represents the desired pattern and $\text{AF}_{\text{CADE}}(\theta_i)$ is the pattern obtained by using CADE. The second term in (16) is used for jammer suppression and placement of nulls at their original positions after sensor failure.

4. Simulation Results

In the simulation, a classical Dolph-Chebyshev linear array of 17 sensors with $\lambda/2$ intersensor spacing is used as the test antenna. The array factor in this case represents a -35 dB constant SLL with the nulls at specific angles. Analytical techniques are used to find out the nonuniform weights for classical Dolph-Chebyshev array. In case of sensor failure, cultural algorithm with differential evolution (CADE) is used for the reduction of sidelobes and placement of nulls in the required positions.

Case a. At the first instant the w_7 sensor is assumed to fail. After sensor failure the radiation pattern is destroyed, which results in an increase of the SLL and displacement of null positions. In order to regain the symmetry, its mirror sensor weight w_{-7} is forced to zero. We achieve the desired null depth level (NDL) and deeper first null depth level (FNDL) as compared to that of nonsymmetric case. The SLL rises to -32.32 dB due to the w_7 sensor failure, while due to SSF of the w_7 sensor, the SLL is -26.53 dB. The advantage of SSF is deeper nulls, especially, the first null. The SLL and FNDL for a damage array of single sensor failure and SSF are shown in Table 2. It is clear from Figure 2 that SSF maintains better FNDL as compared to that of single sensor failure.

After optimization by a cultural algorithm with differential evolution (CADE), the SLL of the w_7 sensor failure are reduced to -32.99 dB while due to SSF, the SLL is reduced to -28.35 dB. The recovery of one null due to 7th sensor failure and SSF to its original position $\theta_1 = 19.93^\circ$ is shown in Figures 4 and 5. The comparison of recovered pattern with 7th sensor and SSF for one null imposed is given in Table 3. The recovered NDL of SSF is seven dB deeper than that of 7th sensor failure.

Figures 6 and 7 show the recovery of two nulls at angles of $\theta_1 = 19.93^\circ$ and $\theta_2 = 34.88^\circ$, respectively, for 7th sensor failure and SSF. The SLL and NDL for the corresponding nulls are given in Table 4. The NDL of the SSF is deeper as compared to the 7th sensor failure.

Now we check the recovery of three nulls at the required positions. Figures 8 and 9 show the recovery of three nulls originally at angles of $\theta_1 = 19.93^\circ$, $\theta_2 = 34.88^\circ$, and $\theta_3 = 45.44^\circ$ for 7th sensor failure and SSF. The SLL and NDL for

TABLE 2: Comparison of FNDL and SLL of the damaged pattern.

Comparison of FNDL and SLL of damage pattern of 7th sensor and SSF			
7th sensor failure		SSF	
FNDL (dB)	SLL (dB)	FNDL (dB)	SLL (dB)
-32.32	-29.45	-88.98	-26.53

TABLE 3: Recovery of one null.

Comparison of NDL and SLL of one sensor failure and SSF				
Correction of 7th sensor failure		Correction of SSF		Recovery of nulls
NDL (dB)	SLL (dB)	NDL (dB)	SLL (dB)	
-104.7	-32.99	-111.5	-28.35	1st null recovered.

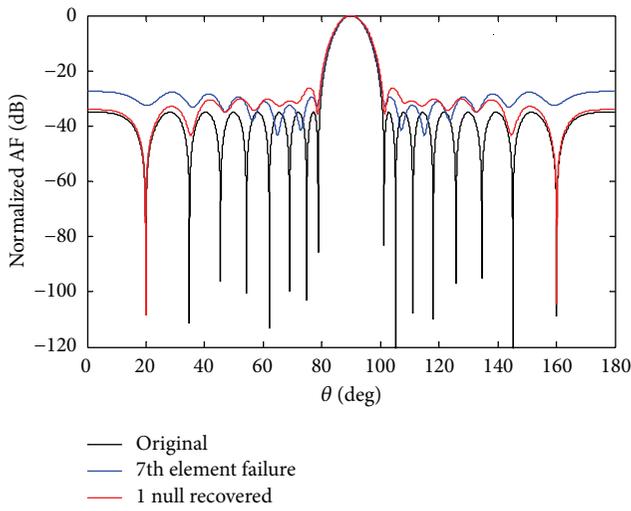


FIGURE 4: The original radiation pattern, the w_7 sensor damage, and recovery of one null.

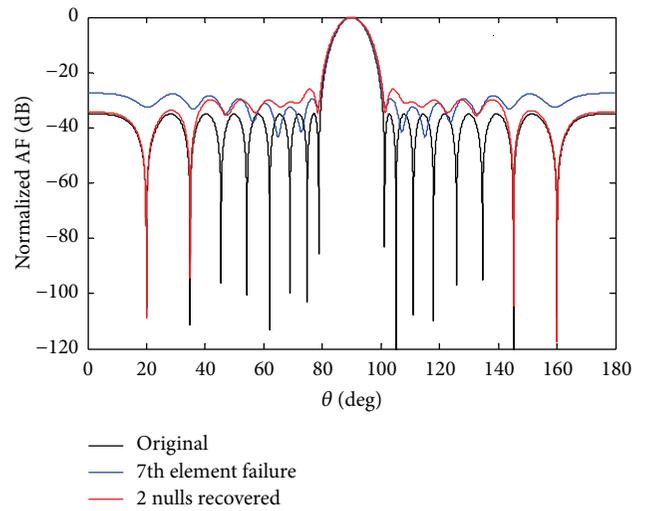


FIGURE 6: The original radiation pattern, the w_7 sensor damage, and recovery of two nulls.

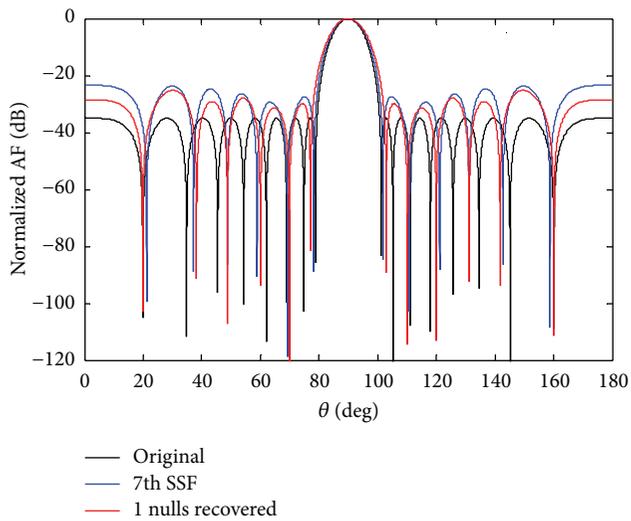


FIGURE 5: The original radiation pattern, the w_7 SSF, and recovery of one null.

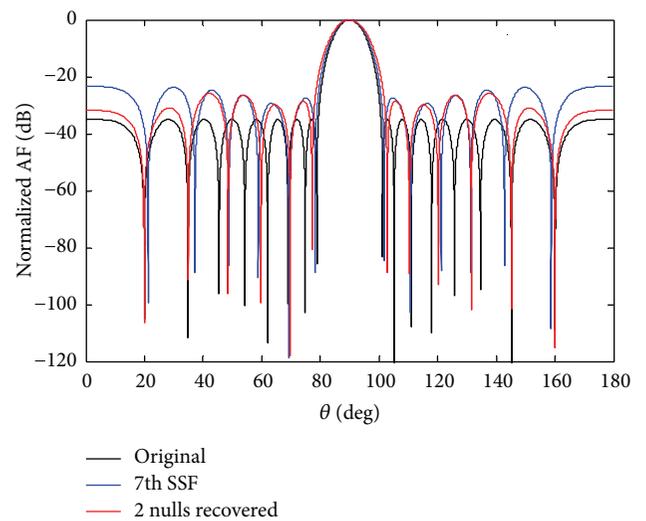


FIGURE 7: The original radiation pattern, the w_7 SSF, and recovery of two nulls.

TABLE 4: Recovery of two nulls.

Correction of 7th sensor failure		Correction of SSF		Recovery of nulls
NDL (dB)	SLL (dB)	NDL (dB)	SLL (dB)	
-95.3	-29.87	-115.1	-31.12	1st null recovered.
-93.65	-30.75	-101.4	27.07	2nd null recovered.

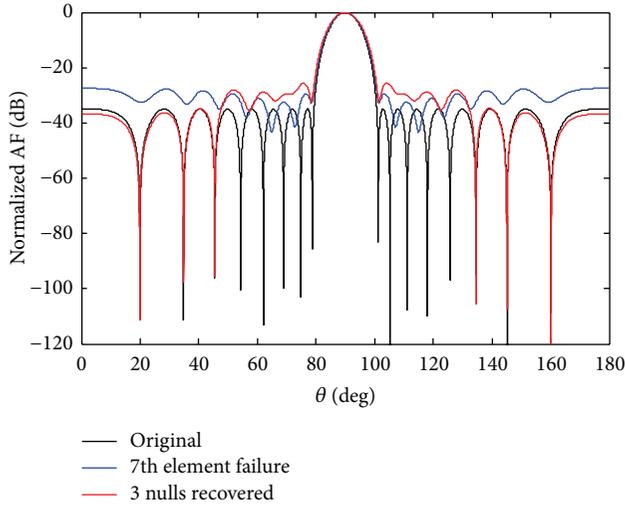


FIGURE 8: The original radiation pattern, the w_7 sensor damage, and recovery of three nulls.

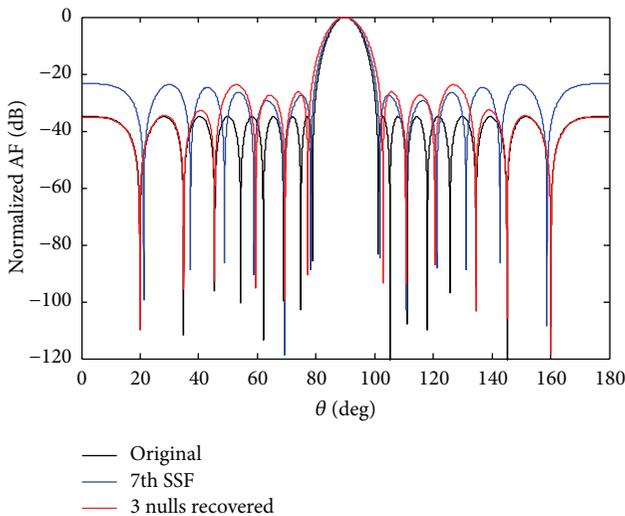


FIGURE 9: The original radiation pattern, the w_7 SSF, and recovery of three nulls.

the corresponding nulls are given in Table 5. The NDL of all nulls in SSF is deeper than that of 7th sensor failure.

Now the recovery of five nulls for 7th sensor failure and SSF originally at positions 19.93° , 34.88° , 45.44° , 62.02° , and 68.94° is carried out and shown in Figures 10 and 11. A comparison of SLL and NDL for the recovery of five nulls is given in Table 6. In each case SSF produces deeper nulls

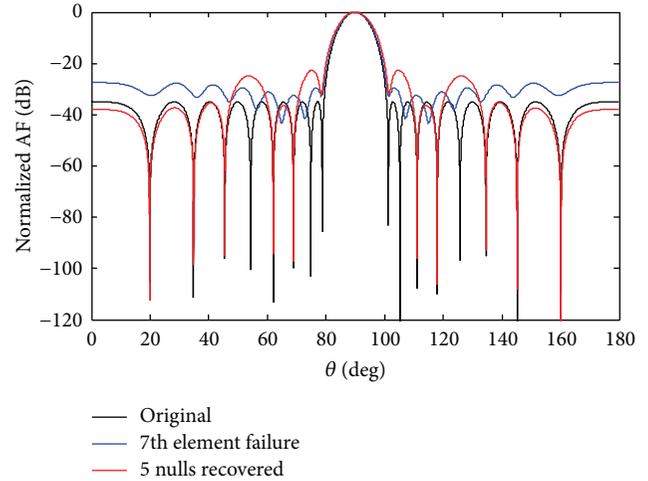


FIGURE 10: The original radiation pattern, the w_7 sensor damage, and recovery of five nulls.

compared to the 7th sensor failure. From simulation it is observed that we have received deeper depth of nulls in SSF scenarios as compared to 7th sensor failure discussed above.

Case b. In this case, we discuss the failure of w_4 sensor. If the sensor w_4 fails due to any reason, the whole radiation pattern became damage. After optimization, the SLL reduces and nulls are steered back to their original positions as shown in Figure 12. Then to create symmetry we also force w_{-4} equal to zero, to achieve the required null depth level. The advantage of symmetry sensor failure is to get deeper null depth level. The number of nulls achieved in 7th symmetry sensor failure is six, and in case of 4th symmetry sensor failure the number of nulls received are three. From the simulation results, it is clear that the number of nulls reduces by one as the sensors get damage near the centre sensor. After optimization by CADE, the SLL reduces and nulls are steered back to their previous positions at angles $\theta_1 = 34.89^\circ$, $\theta_2 = 54.31^\circ$, and $\theta_3 = 68.9^\circ$ as shown in Figure 13. The null depth level for single and SEF is given in Table 7.

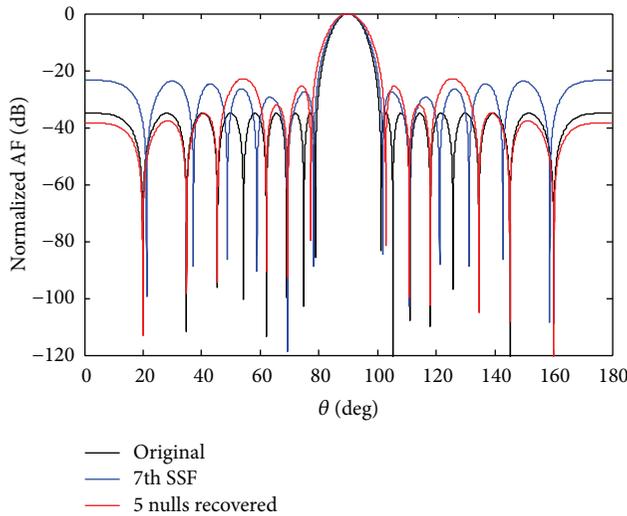
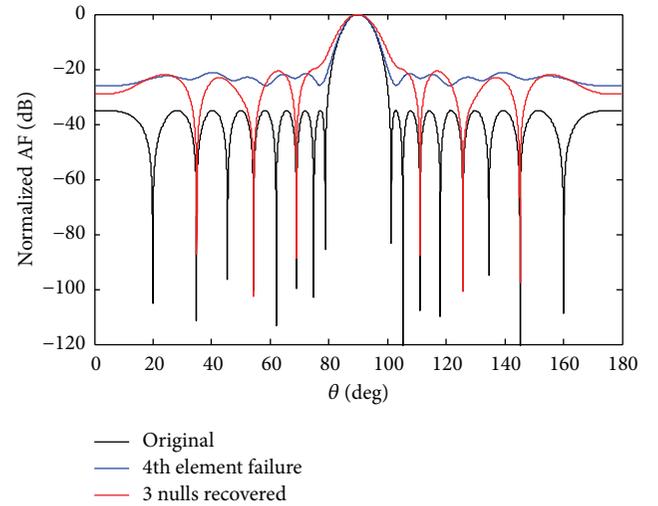
Case c. In this section, we discuss the possibility of getting failure near the centre sensor, if the sensor w_1 fails due to unforeseen reason. From Figure 14 it is clear that its SLL increases and nulls are damaged and also lose null depth. To create the symmetry, we also force its mirror sensor weight w_{-1} equal to zero. The one advantage of symmetry sensor failure is to get the deeper null depth level and, on the other hand, due to w_1 SSF its beamwidth also decreases. In case of

TABLE 5: Recovery of three nulls.

Comparison of NDL and SLL of 7th sensor failure and SSF				
Correction of 7th sensor failure		Correction of SSF		Recovery of nulls
NDL (dB)	SLL (dB)	NDL (dB)	SLL (dB)	
-111.6	-36.32	-118.5	-34.59	1st null recovered.
-97.57	-34.92	-105.6	-32.68	2nd null recovered.
-95.01	-28.1	-103.3	-24.19	3rd null recovered.

TABLE 6: Recovery of five nulls.

Comparison of NDL and SLL of 7th sensor failure and SSF				
Correction of 7th sensor failure		Correction of SSF		Recovery of nulls
NDL (dB)	SLL (dB)	NDL (dB)	SLL (dB)	
-112.7	-37.34	-120	-37.8	1st null recovered.
-98.16	-35.31	-108.4	-34.93	2nd null recovered.
-95.33	-24.9	-105.3	-22.91	3rd null recovered.
-94.6	-36.42	-102.5	-32.27	4th null recovered.
-97.39	-22.69	-99.68	-25.832	5th null recovered.

FIGURE 11: The original radiation pattern, the w_7 SSF, and recovery of five nulls.FIGURE 12: The original radiation pattern, the w_4 sensor failure, and recovery of three nulls.

7th symmetry sensor failure, the nulls are six, and in 4th SSF the achievable nulls are three but we received only one null in case of w_1 SSF as shown in Figure 15. The null depth level for single and SEF is given in Table 8.

Case d. The main beam can be steered at any desired angle. If the user changes their position than the main beam can be steered in the desired direction. Figure 16 shows the corrected pattern with recovered nulls at main beam pointing at $\theta_s = 110^\circ$. The main beam can be steered in the direction of the desired user at any particular angle. The array factor for $2M + 1$ sensors in terms of main beam direction θ_s is given by

$$AF(\theta_i) = \sum_{n=-M}^M w_n \exp jnkd(\cos \theta_i - \cos \theta_s), \quad (17)$$

where θ_s is the main beam direction to which it can be steered to the desired angles.

5. Conclusion and Future Work

We have proposed symmetric sensor failure (SSF) technique along cultural algorithm with differential evolution for the correction of faulty arrays. The null depth of all nulls, especially the first one, has been achieved with the help of SSF technique. Null steering at their original positions and sidelobe reduction has been achieved by a cultural algorithm with differential evolution and using a proper fitness function demanding the sidelobe reduction and null constraints. Due to 7th SSF the number of nulls is six and in 4th SSF the achievable nulls are three but in case of w_1 SSF we received

TABLE 7: Recovery of three nulls.

Comparison of NDL and SLL of 4th sensor failure and SSF					
Correction of 7th sensor failure		Correction of SSF		Recovery of nulls	
NDL (dB)	SLL (dB)	NDL (dB)	SLL (dB)		
-87.56	-22.1	-86.24	-22.64	1st null recovered.	
-102.7	-23.7	-97.97	-19.65	2nd null recovered.	
-88.81	-21.11	-94.01	-20.01	3rd null recovered.	

TABLE 8: Recovery of one nulls.

Comparison of NDL and SLL of w_1 sensor failure and SSF					
Correction of w_1 sensor failure		Correction of SSF		Recovery of nulls	
NDL (dB)	SLL (dB)	NDL (dB)	SLL (dB)		
-113.3	-20.02	-115.1	-21.1	1st null recovered.	

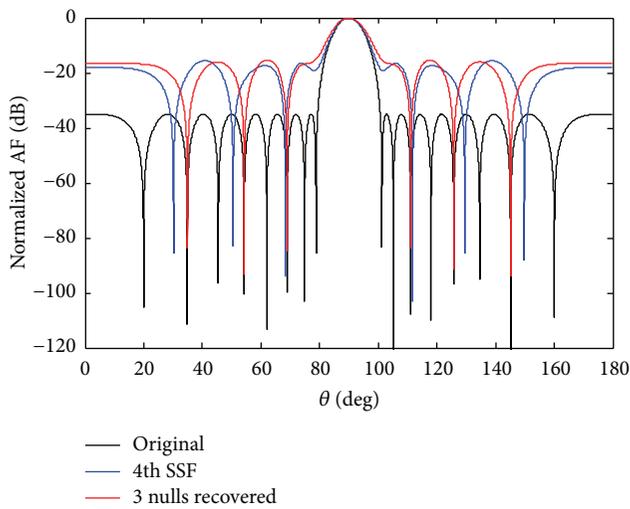


FIGURE 13: The original radiation pattern, the w_4 SSF, and recovery of three nulls.

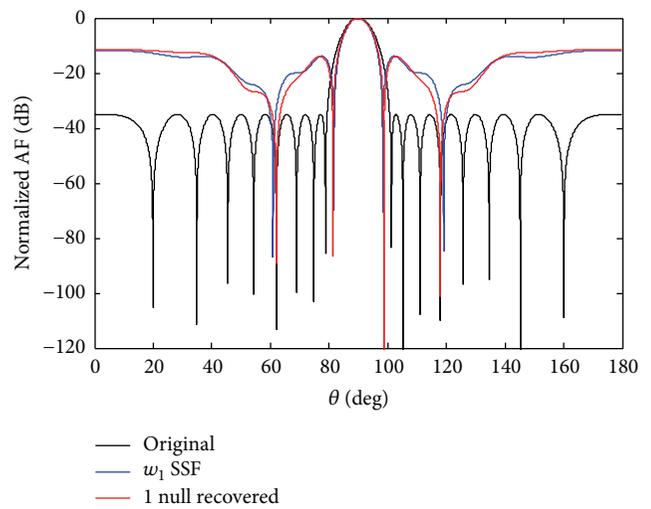


FIGURE 15: The original radiation pattern, the w_1 SSF, and recovery of one null.

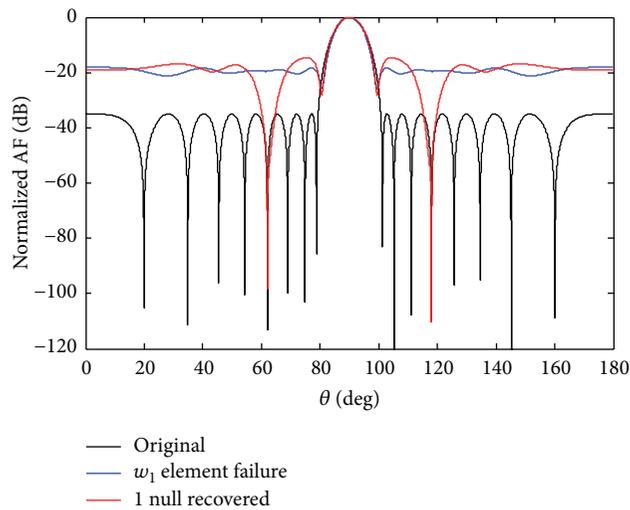


FIGURE 14: The original radiation pattern, the w_1 sensor failure, and recovery of one null.

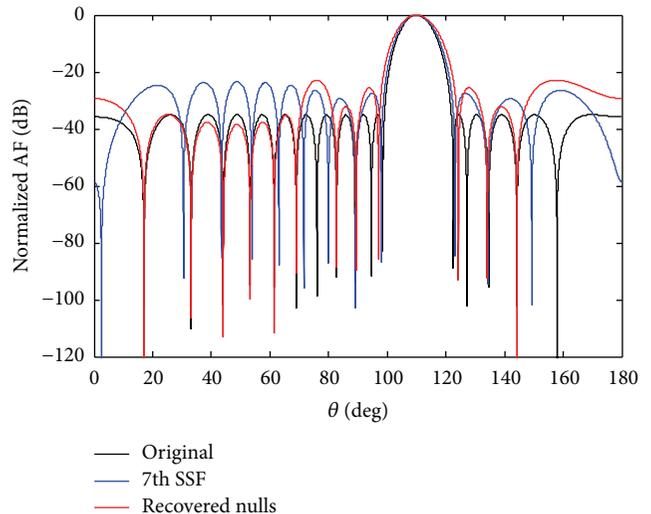


FIGURE 16: The corrected pattern with main beam pointing at 110° with recovered nulls.

only one null with reduced beamwidth. The simulation result shows that as the faulty sensor gets near the central sensor the number of nulls reduces by one. The reduction in the corrected sidelobe level comes at the cost of broader main beam. The corrected pattern has beamwidth broader than that of the original one. Using the approach of symmetric sensor failure, with the reduction of the SLL, we can steer single, double, and multiple nulls in the direction of known interferences. This method can be extended to planar arrays.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] S. Applebaum, "Adaptive arrays," *IEEE Transactions on Antennas and Propagation*, vol. 24, no. 5, pp. 585–598, 1976.
- [2] J. A. Hejres, "Null steering in phased arrays by controlling the positions of selected elements," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 11, pp. 2891–2895, 2004.
- [3] F. Zaman, I. M. Qureshi, J. A. Khan, and Z. U. Khan, "An application of artificial intelligence for the joint estimation of amplitude and two-dimensional direction of arrival of far field sources using 2-L-shape array," *International Journal of Antennas and Propagation*, vol. 2013, Article ID 593247, 10 pages, 2013.
- [4] F. Zaman, "Amplitude and directional of arrival estimation: comparison between different techniques," *Progress in Electromagnetics Research B*, vol. 39, pp. 319–335, 2012.
- [5] J. A. Hejres, A. Peng, and J. Hijres, "Fast method for sidelobe nulling in a partially adaptive linear array using the elements positions," *IEEE Antennas and Wireless Propagation Letters*, vol. 6, pp. 332–335, 2007.
- [6] J. A. Hejres, "Null steering in phased arrays by controlling the positions of selected elements," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 11, pp. 2891–2895, 2004.
- [7] T. J. Peters, "A conjugate gradient-based algorithm to minimize the sidelobe level of planar arrays with element failures," *IEEE Transactions on Antennas and Propagation*, vol. 39, no. 10, pp. 1497–1504, 1991.
- [8] R. J. Mailloux, "Array failure correction with a digitally beam-formed array," *IEEE Transactions on Antennas and Propagation*, vol. 44, no. 12, pp. 1543–1550, 1996.
- [9] K. Guney and S. Basbug, "Interference suppression of linear antenna arrays by amplitude-only control using a bacterial foraging algorithm," *Progress in Electromagnetics Research*, vol. 79, pp. 475–497, 2008.
- [10] K. Guney and M. Onay, "Amplitude-only pattern nulling of linear antenna arrays with the use of bees algorithm," *Progress in Electromagnetics Research*, vol. 70, pp. 21–36, 2007.
- [11] H. Li and B. Himed, "Transmit subaperturing for MIMO radars with co-located antennas," *IEEE Journal on Selected Topics in Signal Processing*, vol. 4, no. 1, pp. 55–65, 2010.
- [12] Y. W. Zhong, L. J. Wang, C. Y. Wang, and H. Zhang, "Multi-agent simulated annealing algorithm on differential evolution algorithm," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 217–228, 2012.
- [13] T. J. Choi, C. W. Ahn, and J. An, "An adaptive Cauchy differential evolution algorithm for global numerical optimization," *The Scientific World Journal*, vol. 2013, Article ID 969734, 12 pages, 2013.
- [14] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.
- [15] O. P. Acharya, A. Patnaik, and S. N. Sinha, "Null steering in failed antenna arrays," *Applied Computational Intelligence and Soft Computing*, vol. 2011, Article ID 692197, 9 pages, 2011.
- [16] S. U. Khan, I. M. Qureshi, F. Zaman, and A. Naveed, "Null placement and sidelobe suppression in failed array using symmetrical element failure technique and hybrid heuristic computation," *Progress in Electromagnetics Research B*, vol. 52, pp. 165–184, 2013.
- [17] S. U. Khan, I. M. Qureshi, F. Zaman, A. Basit, and W. Khan, "Application of firefly algorithm to fault finding in linear arrays antenna," *World Applied Sciences Journal*, vol. 26, no. 2, pp. 232–238, 2013.
- [18] R. G. Reynolds, "An introduction to cultural algorithms," in *Evolutionary Programming: Proceedings of the Third Annual Conference*, A. V. Sebald and L. J. Fogel, Eds., pp. 131–139, World Scientific Publishing, River Edge, NJ, USA, 1994.
- [19] R. G. Reynolds and C. Chung, "The use of cultural algorithms to evolve multi agent cooperation," in *Proceedings of the Micro-Robot World Cup Soccer Tournament*, pp. 53–56, Taejon, Republic of Korea, 1996.
- [20] X. Jin and R. G. Reynolds, "Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach," in *Proceedings of the Congress on Evolutionary Computation*, pp. 1672–1678, Washington, DC, USA, 1999.
- [21] I. Wolf, "Determination of the radiating system which will produce a specified directional characteristic," *Proceedings of the Institute of Radio Engineers*, vol. 25, pp. 630–643, 1937.
- [22] R. Storn and K. Price, "Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep. TR-95-012, International Computer Science Institute, Berkeley, Calif, USA, 1995.
- [23] R. L. Becerra and C. A. C. Coello, "Cultured differential evolution for constrained optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 33–36, pp. 4303–4322, 2006.
- [24] H. Steyskal, R. A. Shore, and R. L. Haupt, "Methods for null control and their effects on the radiation pattern," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 404–409, 1986.

Research Article

A Novel Harmony Search Algorithm Based on Teaching-Learning Strategies for 0-1 Knapsack Problems

Shouheng Tuo, Longquan Yong, and Fang'an Deng

School of Mathematics and Computer Science, Shaanxi University of Technology, Hanzhong 723001, China

Correspondence should be addressed to Shouheng Tuo; tuo_sh@126.com

Received 17 August 2013; Accepted 17 September 2013; Published 8 January 2014

Academic Editors: R. Alex and Z. Cui

Copyright © 2014 Shouheng Tuo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

To enhance the performance of harmony search (HS) algorithm on solving the discrete optimization problems, this paper proposes a novel harmony search algorithm based on teaching-learning (HSTL) strategies to solve 0-1 knapsack problems. In the HSTL algorithm, firstly, a method is presented to adjust dimension dynamically for selected harmony vector in optimization procedure. In addition, four strategies (harmony memory consideration, teaching-learning strategy, local pitch adjusting, and random mutation) are employed to improve the performance of HS algorithm. Another improvement in HSTL method is that the dynamic strategies are adopted to change the parameters, which maintains the proper balance effectively between global exploration power and local exploitation power. Finally, simulation experiments with 13 knapsack problems show that the HSTL algorithm can be an efficient alternative for solving 0-1 knapsack problems.

1. Introduction

Harmony search (HS) [1, 2] is a new population-based meta-heuristic optimization algorithm. It has received much attention regarding its application potential as continuous and discrete optimal problem. Inspired by the process of the musicians' improvisation of the harmony, the HS algorithm improvises its instruments' pitches searching for a perfect state of harmony. The effort to find a new harmony in music is analogous to finding a better solution in an optimization process. HS has been applied to optimization problems in different areas [3–11]. The HS algorithm has powerful exploration ability in a reasonable time but is not good at performing a local search. In order to improve the performance of the harmony search method, several variants of HS have been proposed [12–20]. These variants have some improvement on continuous optimization problems. However, their effectiveness in dealing with discrete problems is still unsatisfactory.

The knapsack problem is one of the classical combinatorial optimization problems. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. The knapsack problem often applies to resource allocation

where there are financial constraints and is studied in fields such as combinatorics, computer science, complexity theory, cryptography, and applied mathematics.

The 0-1 knapsack problem is as follows. Given a set of D items and a knapsack, with

$$\begin{aligned} p_j &= \text{profit of item } j, \\ w_j &= \text{weight of item } j, \\ c_j &= \text{volume of item } j, \\ W &= \text{weight capacity of knapsack,} \\ V &= \text{volume capacity of knapsack,} \end{aligned} \quad (1)$$

select some items so that the total profit of the selected items is maximum, and the total weight and the total volumes of selected items are not more than the weight capacity and the volume capacity of the knapsack. Formally,

$$\text{maximize } z = \sum_{j=1}^D p_j x_j, \quad (2)$$

subject to

$$\sum_{j=1}^D w_j x_j \leq W, \quad (3)$$

$$\sum_{j=1}^D c_j x_j \leq V,$$

where

$$x_j = \begin{cases} 1, & \text{if item } j \text{ is assigned to knapsack} \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Many methods have been employed to solve 0-1 knapsack problems. Zou et al. proposed a novel global harmony search algorithm (NGHS) for 0-1 knapsack problems [21]. Y. Liu and C. Liu presented an evolutionary algorithm to solve 0-1 knapsack problems [22]. Shi used an improved ant colony algorithm to solve 0-1 knapsack problems [23]. Lin solved the knapsack problems with imprecise weight coefficients by using genetic algorithm [24]. Boyer et al. solved knapsack problems on GPU [25]. Hill et al. proposed a heuristic method for 0-1 multidimensional knapsack problems [26]. Gherboudj et al. propose a discrete binary cuckoo search (BCS) algorithm in order to deal with binary optimisation problems [27]. A novel quantum inspired cuckoo search for knapsack problems is present in the literature [28].

In recent years, more and more discrete optimization problems are solved by HS method. To some extent, this is due to the memory consideration rule that is appropriate to be employed to resolve the discrete optimization problems. However, for a high-dimensional discrete optimization problem, classical HS algorithm can be easy to cause premature convergence and stagnation behavior. Therefore, we present a dynamic parameters-adjustment mechanism for solving the high-dimensional 0-1 knapsack problems. To enhance the performance of dealing with discrete problems by HS method, this paper proposed an improved HS algorithm based on teaching-learning (HSTL) strategies.

The rest of the paper is organized as follows. Section 2 introduces the classical HS algorithm and three state-of-the-art variants of HS. The teaching-learning-based optimization (TLBO) algorithm and the proposed approach (HSTL) are introduced in Section 3. Section 4 presents related constraint-handling technique and integer processing method. Experimental results are reported in Section 5. Finally, Section 6 concludes this paper.

2. HS Algorithm and Other Variants

In this section, we introduce the classical HS algorithm and three state-of-the-art variants of HS algorithms: NGHS algorithm [17], intelligent tuned harmony search algorithm (ITHS) [18], and exploratory power of harmony search algorithm (EHS) [19].

2.1. Classical Harmony Search Algorithm (HS). Classical harmony search (HS) is derivative-free meta-heuristic algorithm. It mimics the improvisation process of music players

and uses three rules (memory consideration, pitch adjustments, and randomization) to optimize the harmony memories. The steps in the procedure of classical harmony search algorithm are as follows.

Step 1 (initialize the harmony memory). The harmony memory (HM) consists of HMS harmony. Each harmony is generated from a uniform distribution in the feasible space, as

$$x_i^j = x_i^L + \text{rand}() \cdot (x_i^U - x_i^L), \quad (5)$$

$$i = 1, 2, \dots, D; \quad j = 1, 2, \dots, \text{HMS},$$

where $\text{rand}()$ is a uniform distribution random number between 0 and 1.

Consider the following:

$$\text{HM} = \begin{bmatrix} X^1 \\ X^2 \\ \vdots \\ X^{\text{HMS}} \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_D^1 \\ x_1^2 & x_2^2 & \dots & x_D^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & \dots & x_D^{\text{HMS}} \end{bmatrix}. \quad (6)$$

Step 2 (improvise a new harmony via three rules). Improvise a new harmony X^{new} via three rules: memory consideration, pitch adjustment, and random generation.

(a) *Memory Consideration.* Decision variable value of the new harmony will be generated by choosing from the harmony memory with probability HMCR.

(b) *Pitch Adjustment.* Get a component randomly from an adjacent value of one decision variable of a harmony vector with probability PAR.

(c) *Random Generation.* Generate a component randomly in the feasible region with probability $1-\text{HMCR}$.

The improvisation procedure of a new harmony works as Algorithm 1.

A new potential variation (or an offspring) is generated in Step 2, which is equivalent to mutation and crossover operator in standard Evolution Algorithms (EAs).

Step 3 (update the worst harmony). Consider the following:

If X^{new} is better than X^{worst}

$$X^{\text{new}} = X^{\text{worst}}, \quad (7)$$

Endif,

where $X^{\text{worst}} = (x_1^{\text{worst}}, x_2^{\text{worst}}, \dots, x_D^{\text{worst}})$ denotes the worst harmony in HM.

Step 4 (check stopping criterion). If the stopping criterion (maximum function evaluation times: MaxFEs) is satisfied, computation is terminated. Otherwise, Step 2 and Step 3 are repeated.

```

For  $i = 1$  to  $D$ 
  If  $\text{rand}() < \text{HMCR}$ 
     $x_i^{\text{new}} = x_i^j, j \in U\{1, 2, \dots, \text{HMS}\}$ 
    If  $\text{rand}() < \text{PAR}$ 
       $x_i^{\text{new}} = x_i^{\text{new}} \pm \text{rand}() \times \text{BW}(i)$ 
       $x_i^{\text{new}} = \min(\max(x_i^{\text{new}}, x_i^L), x_i^U)$ 
    Endif
  Else
     $x_i^{\text{new}} = x_i^L + (x_i^U - x_i^L) \times \text{rand}()$ 
  Endif
EndFor
    
```

ALGORITHM 1: The improvisation procedure of new harmony by classical HS.

```

For  $i = 1$  to  $D$ 
   $x_i^r = 2 \times x_i^{\text{best}} - x_i^{\text{worst}}$ 
   $x_i^r = \min(\max(x_i^r, x_i^L), x_i^U)$ 
   $x_i^{\text{new}} = x_i^{\text{worst}} + \text{rand}() \times (x_i^r - x_i^{\text{worst}})$ 
  If  $\text{rand}() \leq p_m$  %random mutation
     $x_i^{\text{new}} = x_i^L + (x_i^U - x_i^L) \times \text{rand}()$ 
  Endif
EndFor
    
```

ALGORITHM 2: The improvisation procedure of new harmony by NGHS.

2.2. *The NGHS Algorithm.* In NGHS algorithm, three significant parameters, harmony memory considering rate (HMCR), bandwidth (BW), and pitch adjusting rate (PAR), are excluded from NGHS, and a random select rate (p_m) is included in the NGHS. In Step 3, NGHS works as Algorithm 2, where $X^{\text{best}} = (x_1^{\text{best}}, x_2^{\text{best}}, \dots, x_D^{\text{best}})$ and $X^{\text{worst}} = (x_1^{\text{worst}}, x_2^{\text{worst}}, \dots, x_D^{\text{worst}})$ denote the best harmony and the worst harmony in HM, respectively. We set parameter $p_m = 2/D$ for the 0-1 knapsack problem and $p_m \in [0.005, 0.1]$ for continuous optimal problem.

2.3. *The EHS Algorithm.* The EHS algorithm uses the same structure with the classical HS algorithm and does not introduce any complex operations. The only difference between HS and EHS is that the EHS algorithm proposed a new scheme of tuning BW, and it works as follows (proportional to the current population variance):

$$\text{BW} = k \sqrt{\frac{1}{\text{HMS}} \sum_{i=1}^{\text{HMS}} (x_i - \bar{x})^2}, \quad (8)$$

where k is the proportionality constant. If the value of k is high, the population variance will maintain or increase, and thus the global exploration power of algorithm will enhance. While if the value of k is low, the population variance will decrease and the local exploitation performance will increase. The experimental results tested by Das et al. [19] have shown that keeping k around 1.2 provided reasonably accurate results.

The EHS algorithm can enhance the exploratory power, and can provide a better balance between diversification and intensification. However, the exploratory power of EHS leads to the slower convergence [18], and the computational time of BW is greatly increased.

2.4. *The ITHS Algorithm.* The ITHS algorithm [18] proposed a subpopulation formation technique. The population is divided into two groups: Group A and Group B, which is carried out based on $\bar{\text{fit}}$ (the mean objective function value of all harmony vectors in HM). The harmony vectors, whose objective function value is less than $\bar{\text{fit}}$, belong to Group A, and the rest belong to Group B. The improvisation procedure of new harmony by ITHS is shown in Algorithm 3. In ITHS algorithm, the parameter PAR is updated with the number of iterations:

$$\text{PAR} = \text{PAR}_{\text{max}} - \frac{(\text{PAR}_{\text{max}} - \text{PAR}_{\text{min}}) \times t}{T_{\text{max}}}, \quad (9)$$

where t and T_{max} are the current iterations and the maximum iterations.

3. HSTL Algorithm

In this section, we proposed a novel harmony search with teaching-learning strategy which derived from Teaching-Learning-Based Optimization (TLBO) algorithm. Above all, the TLBO algorithm is introduced and analyzed, and then we

```

For  $i = 1$  to  $D$  do
  If  $\text{rand}() < \text{HMCR}$ 
     $x_i^{\text{new}} = x_i^j, j \in U \{1, 2, \dots, \text{HMS}\}$ 
    If  $\text{rand}() < \text{PAR}$ 
      If  $\text{fit}(x^j) \leq \overline{\text{fit}}$  //Group A
        If  $\text{rand}() < 0.5$ 
           $y_i = x_i^{\text{best}} - \text{rand}() \times (x_i^{\text{best}} - x_i^{\text{new}})$ 
        Else
           $y_i = x_i^{\text{best}} - \text{rand}() \times (x_i^{\text{worst}} + x_i^{\text{new}})$ 
        Endif
      Else //Group B
         $x_m^{\text{best}} = x_m^{\text{best}} \times (x_i^U / x_m^U), m \in U \{1, 2, \dots, \text{HMS}\}$ 
         $y_i = x_i^{\text{new}} + \text{rand}() \times (x_m^{\text{best}} - x_i^{\text{new}})$ 
      Endif
       $x_i^{\text{new}} = \min(\max(y_i, x_i^L), x_i^U)$ 
    Endif
  Else
     $x_i^{\text{new}} = x_i^L + (x_i^U - x_i^L) \times \text{rand}()$ 
  Endif
EndFor

```

ALGORITHM 3: The improvisation procedure of new harmony by ITHS.

focus on the details of HSTL algorithm and the strategies of dynamically adjusting the parameters.

Since its origination, HS algorithm has been applied to many practical optimization problems. However, for large scale optimization problems, HS has slow convergence and low precision, which is because a new decision variable value in HM can be generated only by pitch adjustment and randomization strategies during the search process, the memory consideration rule is only used to adjust the decision variable values according to the current HM. HS can maintain a strong exploration power in the early stage, but it does not have a good exploitation in the later stage, and thus it is characterized by earlier mature and slow convergence. Therefore, for solving large scale optimization problem, the key is how to balance between global exploration performance and local exploitation ability.

3.1. Dimension Reduction Adjustment Strategy. As we know, for a complex optimization problem, its optimization may experience a process from extensive exploration in a large range to fine adjustment in a small range. For a D -dimensional optimization problem, we assume that its optimal solution is $X^* = (x_1^*, x_2^*, \dots, x_D^*)$. Let initial HM be as follows:

$$\text{HM} = \begin{bmatrix} X^1 \\ X^2 \\ \vdots \\ X^{\text{HMS}} \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_D^1 \\ x_1^2 & x_2^2 & \cdots & x_D^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & \cdots & x_D^{\text{HMS}} \end{bmatrix}. \quad (10)$$

After several iterations, the HM turns into HM2. It can be seen from HM2 that each solution X^j ($j = 1, 2, \dots, \text{HMS}$)

in HM2 has nearly achieved the best solution except for one dimension ($y_i^j, j = 1, 2, \dots, \text{HMS}; i = 1, 2, \dots, D$):

$$\text{HM2} = \begin{bmatrix} X^1 \\ X^2 \\ \vdots \\ X^{\text{HMS}} \end{bmatrix} = \begin{bmatrix} y_1^1 & x_2^* & \cdots & x_D^* \\ x_1^* & y_2^2 & \cdots & x_D^* \\ \vdots & \vdots & \vdots & \vdots \\ x_1^* & x_2^* & \cdots & y_D^{\text{HMS}} \end{bmatrix}. \quad (11)$$

Then we assume that only the harmony memory consideration rule of HS algorithm is employed to optimize the HM2. In the following, we employ the two methods to generate two new harmonies: $X^{\text{new1}} = (x_1^{\text{new1}}, x_2^{\text{new1}}, \dots, x_D^{\text{new1}})$ and $X^{\text{new2}} = (x_1^{\text{new2}}, x_2^{\text{new2}}, \dots, x_D^{\text{new2}})$, respectively, and then analyze which method is better.

Method 1. Generate the solution X^{new1} on each variable x_i^{new1} ($i = 1, 2, \dots, D$) by using the harmony memory consideration rule, as in Algorithm 4.

Method 2. Let $X^{\text{new2}} = X^{\text{worst}}$ and then adjust one of the variables of the new solution X^{new2} by using the harmony memory consideration rule, as in Algorithm 5.

In the following, we analyze the two methods.

In Method 1, all of the decision variables of harmony X^{new1} are chosen from HM2. If X^{new1} hopes to be the optimal solution $X^* = (x_1^*, x_2^*, \dots, x_D^*)$, we should choose $x_i^{\text{new1}} = x_i^*, i = 1, 2, \dots, D$. In other words, y_i^j ($j = 1, 2, \dots, \text{HMS}; i = 1, 2, \dots, D$) in HM2 cannot be chosen as

```

For  $i = 1$  to  $D$ 
  If  $\text{rand}() < \text{HMCR}$ 
     $x_i^{\text{new1}} = x_i^j, j \in U \{1, 2, \dots, \text{HMS}\}$ 
  EndIf
EndFor
    
```

ALGORITHM 4

```

 $X^{\text{new2}} = X^{\text{worst}}$ 
 $i = \text{round}(\text{rand}() * D)$  //Generating a random integer between 1 and D.
 $x_i^{\text{new2}} = x_i^j, j \in U \{1, 2, \dots, \text{HMS}\}$ 
    
```

ALGORITHM 5

x_i^{new1} in the latter iteration. We can see that the probability x_i^{new1} that turns into x_i^* is $\text{HMS} - 1/\text{HMS}$ in HM2, so the probability X^{new1} turns into $X^* = (x_1^*, x_2^*, \dots, x_D^*)$ is $((\text{HMS} - 1)/\text{HMS})^D$.

In Method 2, assume that the $X^{\text{new2}} = X^{\text{worst}} \triangleq (y_1^1, x_2^*, \dots, x_D^*)$, during each iteration, and only one decision variable of X^{new2} needs to be adjusted with harmony memory consideration rule. So the probability that decision variable y_1^1 is chosen to adjust is $1/D$, and then the probability that y_1^1 will be replaced with x_1^* is $\text{HMS} - 1/\text{HMS}$. Therefore, in one iteration, the probability that X^{new2} turns into $X^* = (x_1^*, x_2^*, \dots, x_D^*)$ by Method 2 is $1/D \cdot \text{HMS} - 1/\text{HMS}$.

Next, we compare the success rate between Method 1 and Method 2 at different dimensions.

When $\text{HMS} = 10, D = 10$, the success rate of Method 1 is

$$P \{X^{\text{new1}} \rightarrow X^*\} = \left(\frac{\text{HMS} - 1}{\text{HMS}}\right)^D = \left(\frac{10 - 1}{10}\right)^{10} = 0.34867844, \tag{12}$$

and the success rate of Method 2 is

$$P \{X^{\text{new2}} \rightarrow X^*\} = \frac{1}{D} \cdot \frac{\text{HMS} - 1}{\text{HMS}} = \frac{1}{10} \cdot \frac{10 - 1}{10} = 0.09. \tag{13}$$

Apparently, under this condition, Method 1 is superior to Method 2. However, if we set $\text{HMS} = 10, D = 100$, the success rate of Method 1 is

$$P \{X^{\text{new1}} \rightarrow X^*\} = \left(\frac{\text{HMS} - 1}{\text{HMS}}\right)^D = \left(\frac{100 - 1}{100}\right)^{100} = 2.65614E - 05, \tag{14}$$

while the success rate of Method 2 is

$$P \{X^{\text{new2}} \rightarrow X^*\} = \frac{1}{D} \cdot \frac{\text{HMS} - 1}{\text{HMS}} = \frac{1}{100} \cdot \frac{100 - 1}{100} = 9.0E - 03. \tag{15}$$

For different D , a more detailed success rate of Method 1 and Method 2 is listed in Table 1.

From Table 1, it can be seen that, for low dimensional problem ($D < 50$), success rate of Method 1 is greater than that of Method 2; however, Method 2 has higher success rate than Method 1 when $D \geq 50$, and with the increasing of dimensionality, the success rate of Method 1 will drop dramatically, but Method 2 can maintain a higher success rate.

By above idea, in the beginning stages, exploring is on a wider domain so as to search fast, and in the later stages, the search focus on a small space to improve the accuracy of solution. Aiming at the HS algorithm, we design a dynamic dimension selection strategy to adjust some selected decision variables. A simple process is shown in Figure 1.

In Figure 1, the parameter TP represents the tune probability of each decision variable. TP changes from TP_{max} to TP_{min} . In other words, each decision variable of the objective harmony vector X^{new} will be tuned with probability TP which decreases from TP_{min} to TP_{min} with the increase of iterations.

3.2. The TLBO Algorithm. Teaching-Learning-Based Optimization (TLBO) algorithm [29–38] is a new nature-inspired algorithm; it mimics the teaching process of teacher and learning process among learners in a class. TLBO shows a better performance with less computational effort for large scale problems [31]. In addition, TLBO needs very few parameters.

In the TLBO method, the task of a teacher is to try to increase mean knowledge of all learners of the class in the subject taught by him or her depending on his or her capability. Learners make efforts to increase their knowledge by interaction among themselves. A learner is considered as a solution or a vector, different design variables of a vector will be analogous to different subjects offered to learners, and the learners' result is analogous to the "fitness" as in other population-based optimization techniques. The teacher is considered as the best solution obtained so far. The process of TLBO is divided into two phases, "Teacher Phase" and "Learner Phase."

3.2.1. Teacher Phase. Assume that there are D number of subjects (i.e., design variables) and NP number of learners (i.e., population size); $X^{\text{teacher}} = (x_1^{\text{teacher}}, x_2^{\text{teacher}}, \dots, x_D^{\text{teacher}})$

TABLE 1: Success rate of Methods 1 and 2.

HMS	D	The success rate	
		Method 1	Method 2
10	10	3.48678E-01	9.00000E-02
10	20	1.21577E-01	4.50000E-02
10	30	4.23912E-02	3.00000E-02
10	40	1.47809E-02	2.25000E-02
10	50	5.15378E-03	1.80000E-02
10	60	1.79701E-03	1.50000E-02
10	70	6.26579E-04	1.28571E-02
10	80	2.18475E-04	1.12500E-02
10	90	7.61773E-05	1.00000E-02
10	100	2.65614E-05	9.00000E-03
10	110	9.26139E-06	8.18182E-03
10	120	3.22925E-06	7.50000E-03
10	130	1.12597E-06	6.92308E-03
10	140	3.92601E-07	6.42857E-03
10	150	1.36891E-07	6.00000E-03
10	160	4.77311E-08	5.62500E-03
10	170	1.66428E-08	5.29412E-03
10	180	5.80299E-09	5.00000E-03
10	190	2.02338E-09	4.73684E-03
10	200	7.05508E-10	4.50000E-03
10	300	1.87393E-14	3.00000E-03
10	400	4.97741E-19	2.25000E-03
10	500	1.32207E-23	1.80000E-03
10	600	3.51161E-28	1.50000E-03
10	700	9.32731E-33	1.28571E-03
10	800	2.47747E-37	1.12500E-03
10	900	6.58049E-42	1.00000E-03
10	1000	1.74787E-46	9.00000E-04

is the best learner (i.e., teacher). For each learner $X^j = (x_1^j, x_2^j, \dots, x_D^j)$, the works of teaching are as follows:

$$x_i^{j,\text{new}} = x_i^{j,\text{old}} + \text{rand}() \times (x_i^{\text{teacher}} - T_F \times \text{Mean}_i),$$

$$\text{Mean}_i = \frac{1}{NP} \sum_{j=1}^{NP} x_i^j, \quad (16)$$

$$j = 1, 2, \dots, NP, \quad i = 1, 2, \dots, D,$$

where $x_i^{j,\text{old}}$ and $x_i^{j,\text{new}}$ denote the knowledge of the j th learner (X^j) before and after learning the i th subject, respectively. T_F is the teaching factor which decides the value of mean Mean_i to be changed. T_F is decided by $T_F = \text{round}[1 + \text{rand}()]$.

3.2.2. Learner Phase. Another important approach to increase knowledge for a learner is to interact with other learners. Learning method is expressed as in Algorithm 6.

Even since the TLBO algorithm proposed by Rao et al. [29], it has been applied to the fields of engineering optimization, such as mechanical design optimization [29, 32, 37], heat exchangers [33], thermoelectric cooler [34], and

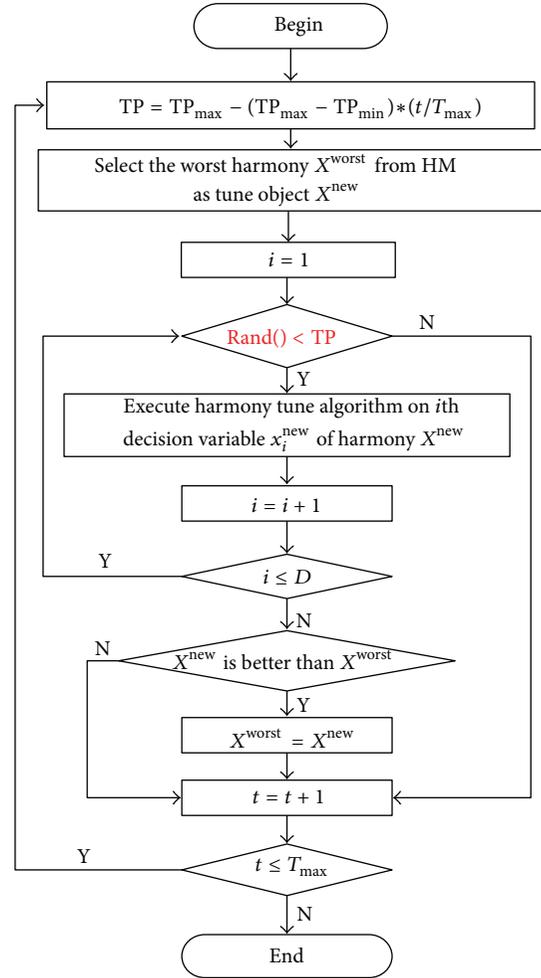


FIGURE 1: Dynamic dimension selection strategy for HS algorithm.

unconstrained and constrained real parameter optimization problems [35, 36]. Some improved TLBO algorithm was present in last two years. An elitist TLBO algorithm for solving unconstrained optimization problems [38] by Rao and Patel and an improved harmony search based on teaching-learning strategy for unconstrained optimization problems by Tuo et al. [39] are proposed.

In the TLBO method, the teacher phase relying on the best solution found so far usually has the fast convergence speed and the well ability of exploitation; it is more suitable for improving the accuracy of the global optimal solution. Learner phase relying on other learners usually has the slow convergence speed; however, it bears stronger exploration capability for solving multimodal problems.

3.3. The HSTL Algorithm. In order to achieve satisfactory optimization performance by applying the HS algorithm to a given problem, we develop a novel harmony search algorithm combined teaching-learning strategy, in which both new harmony generation strategies and associated control parameter values can be dynamically changed according to the process of evolution.

```

For each learner  $X^j$ ,  $j = 1, 2, \dots, NP$ 
  Randomly select another learner  $X^k$  ( $j \neq k$ )
  If  $X^j$  is superior to  $X^k$ 
     $X^{j,new} = X^{j,old} + \text{rand}(1, D) \cdot (X^j - X^k)$ 
  Else
     $X^{j,new} = X^{j,old} + \text{rand}(1, D) \cdot (X^k - X^j)$ 
  Endif
Endfor
If  $X^{j,new}$  is superior to  $X^{j,old}$ 
   $X^j = X^{j,new}$ 
Endif
    
```

ALGORITHM 6: The procedure of learner phase.

It is of great importance to realize the balance between the convergence and the diversity. In the classical HS algorithm, a new harmony is generated in Step 3. After the selecting operation in Step 4, the population diversity may increase or decrease. With high population diversity, the algorithm will have strong exploration power, and at the same time the convergence and the exploitation power will decrease accordingly. Conversely, with a low population variance, the convergence and the exploitation power will increase [18]; the diversity and the exploration power will decrease. So it is significant how to keep balance between the convergence and the diversity. Classical HS algorithm loses exploitation ability easily at later evolution process [19], because of improvising new harmony from HM with a high HMCR and local adjusting with PAR. Diversity of HM decreases gradually from the early iteration to the last. Moreover, in HS algorithm, a low HMCR employed will increase the probability (1-HMCR) of random selection in search space; the exploration power will enhance, but the local search ability and the exploitation accuracy cannot be improved by single pitch adjusting strategy.

To overcome the inherent weaknesses of HS, in this section, we propose an HSTL method. In the HSTL method, an improved teaching-learning strategy is employed to improve the search ability. The HSTL algorithm works as follows.

- (1) Optimization target vector preparation is as follows: $X^{new} = X^{worst}$, where X^{worst} is the worst harmony in the current HM.
- (2) Improve the target vector X^{new} with the following 4 strategies.

(a) *Harmony Memory Consideration.* The values of the target vector x_i^{new} ($i = 1, 2, \dots, D$) are randomly from HM with a probability of HMCR:

$$x_i^{new} = x_i^j, \quad j \in U\{1, 2, \dots, HMS\}, \quad i = 1, 2, \dots, D. \quad (17)$$

(b) *Teaching-Learning Strategy.* If the i th ($i = 1, 2, \dots, D$) design variable of the target vector x_j^{new} has not been considered in (a), it will learn from the best harmony (i.e., teacher) with probability TLP in the teacher phase or from the other harmony (i.e., learner) in the learner phase. The TLP is

the rate of performing teaching-learning operator on design variables that have not been carried out in (a): harmony memory consideration. It works as follows.

Teacher Phase. In this phase, the learner will learn from the best learner (i.e., teacher) in the class. Learner modification is expressed as

$$x_i^{new} = x_i^{new} + \text{rand}() \times [x_i^{best} - T_F \times M_i], \quad (18)$$

$$M_i = \frac{x_i^{worst} + x_i^{new}}{2}, \quad i = 1, 2, \dots, D,$$

where X^{best} is the best harmony in HM and X^{worst} is the worst harmony in HM.

The contribution of this paper is that M_i is replaced by $(x_i^{worst} + x_i^{new})/2$, instead of the mean value of population. This replacement will enhance diversity of population more than standard TLBO algorithm.

Learner Phase. Randomly select r_1 and r_2 from $\{1, 2, \dots, HMS\}$, and $r_1 \neq r_2$:

```

If  $x^{r_1}$  is better than  $x^{r_2}$ 
   $x_i^{new} = x_i^{new} + \text{rand}() \times (x_i^{r_1} - x_i^{r_2})$ 
Else
   $x_i^{new} = x_i^{new} + \text{rand}() \times (x_i^{r_2} - x_i^{r_1})$ 
Endif.
    
```

(c) *Local Pitch Adjusting Strategy.* To achieve better solutions in search space, it will carry out the local pitch adjusting strategy with probability PAR if design variables have not been selected to perform harmony memory consideration and teaching-learning strategy:

$$x_i^{new} = x_i^{new} \pm \text{rand}() \times BW(i). \quad (20)$$

(d) *Random Mutation Operator.* HSTL carries out random mutation in feasible space with probability P_m as follows:

$$x_i^{new} = x_i^L + (x_i^U - x_i^L) \times \text{rand}(). \quad (21)$$

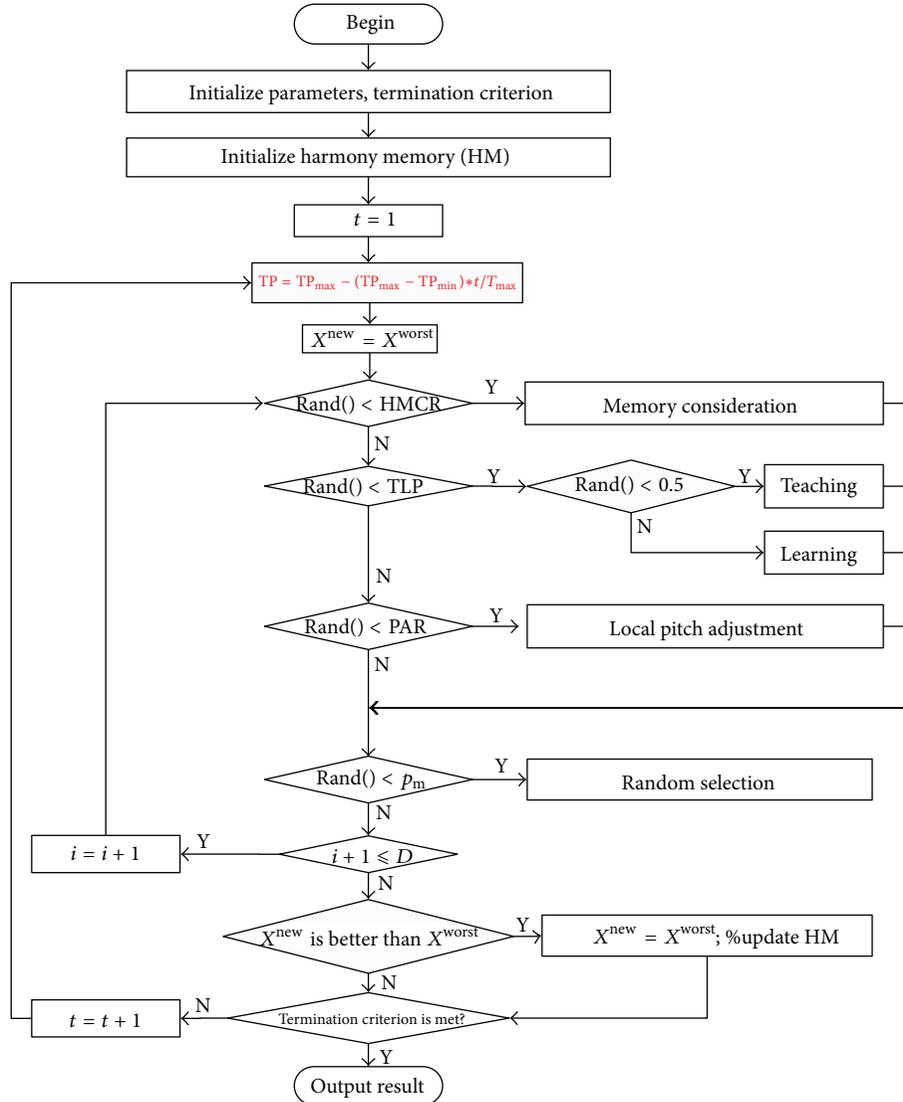


FIGURE 2: The flowchart of HSTL algorithm.

The improvisation of new target harmony in HSTL algorithm is given in Algorithm 7.

The flow chart of HSTL algorithm is shown in Figure 2.

(i) *Update Operation.* In HSTL algorithm, update operation has some changes, as follows.

Get the best harmony X^{best} and the worst harmony X^{worst} from the HM:

If X^{new} is better than X^{best}
 $X^{\text{worst}} = X^{\text{best}}$,
 $X^{\text{best}} = X^{\text{new}}$,
ElseIf X^{new} is better than X^{worst}
 $X^{\text{worst}} = X^{\text{new}}$,
EndIf. (22)

(ii) *Parameters Changed Dynamically.* To efficiently balance the exploration and exploitation power of the HSTL algorithm, HMCR, PAR, BW, and TLP are dynamically adapted to a suitable range with the increase of generations:

$$\text{HMCR} = \text{HMCR}_{\min} + (\text{HMCR}_{\max} - \text{HMCR}_{\min}) \times \left(\frac{t}{T_{\max}} \right), \quad (23)$$

$$\text{TLP} = \text{TLP}_{\min} + (\text{TLP}_{\max} - \text{TLP}_{\min}) \times \left(\frac{t}{T_{\max}} \right)^k, \quad k = 5, \quad (24)$$

$$\text{PAR} = \text{PAR}_{\max} - \frac{(\text{PAR}_{\max} - \text{PAR}_{\min}) \times t}{T_{\max}}, \quad (25)$$

$$\text{BW} = \text{BW}_{\max} + \exp \left[\ln \left(\frac{\text{BW}_{\min}}{\text{BW}_{\max}} \right) \times \sqrt{\frac{t}{T_{\max}}} \right], \quad (26)$$

```

 $X_i^{new} = X_i^{worst};$       % select  $X_i^{worst}$  as Optimization target vector
For  $i = 1$  to  $D$ 
  If  $\text{rand}() \leq \text{HMCR}$       % (a) Harmony memory consideration
     $x_i^{new} = x_i^j$  ( $j = 1, 2, \dots, \text{HMS}$ )
  Elseif  $\text{rand}() \leq \text{TLP}$     % (b) Teaching-Learning strategy
    If  $\text{rand}() \leq 0.5$       % Teaching
       $x_i^{new} = x_i^{new} + \text{rand}() \times [x_i^{best} - 0.5\text{TF} \times (x_i^{worst} + x_i^{new})]$ 
    Else                    % Learning
      Randomly select  $r_1$  and  $r_2$  from  $\{1, 2, \dots, \text{HMS}\}$ 
      If  $x^{r_1}$  is better than  $x^{r_2}$ 
         $x_i^{new} = x_i^{new} + \text{rand}() \times (x_i^{r_1} - x_i^{r_2})$ 
      Else
         $x_i^{new} = x_i^{new} + \text{rand}() \times (x_i^{r_2} - x_i^{r_1})$ 
      end
    Endif
     $x_i^{new} = \min(\max(x_i^{new}, x_i^L), x_i^U)$ 
  Elseif  $\text{rand}(0,1) \leq \text{PAR}$  % (c) Local pitch adjusting strategy
     $x_i^{new} = x_i^{new} \pm \text{rand}() \times \text{BW}(i)$ 
     $x_i^{new} = \min(\max(x_i^{new}, x_i^L), x_i^U)$ 
  Endif
  If  $\text{rand}(0,1) \leq P_m$       % (d) Random mutation operator in feasible space
     $x_i^{new} = x_i^L + (x_i^U - x_i^L) \times \text{rand}()$ 
  Endif
Endfor

```

ALGORITHM 7: Improvisation of new harmony in HSTL algorithm.

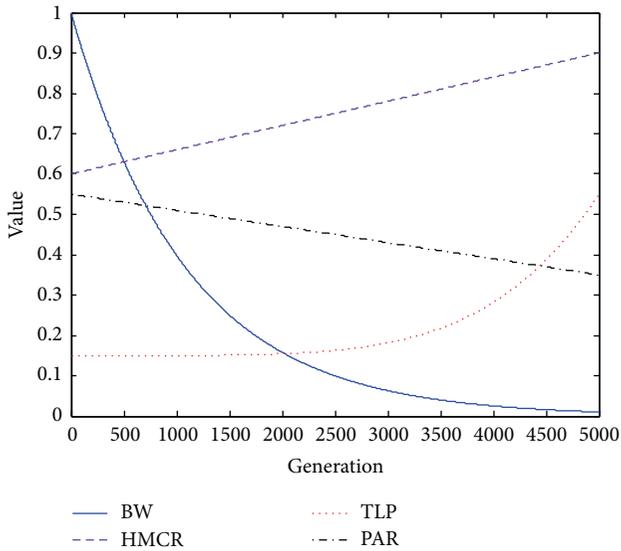


FIGURE 3: The evolution curves of parameters (HMCR, TLP, PAR, and BW) of HSTL algorithm.

where (25) and (26) are quoted from the literature studies [18] and [16], respectively.

Let $\text{HMCR}_{\max} = 0.9$, $\text{HMCR}_{\min} = 0.6$, $\text{TLP}_{\max} = 0.55$, $\text{TLP}_{\min} = 0.15$, $\text{PAR}_{\max} = 0.5$, $\text{PAR}_{\min} = 0.33$, $\text{BW}_{\max} = 1$, and $\text{BW}_{\min} = 0.001$. The changing curves of parameters (HMCR, TLP, PAR, and BW) are shown in Figure 3.

It can be seen that the parameter HMCR increases gradually from 0.6 to 0.9 linearly. TLP increases with low

velocity in the early stage and rises sharply in the final stage. That is to say, in the beginning, the harmony consideration rule and teaching-learning strategy are carried out with a smaller probability; in the later stage, HSTL algorithm begins to focus on local exploitation with harmony consideration and teaching-learning strategy. The benefits of doing so can get more opportunities to reinforce the global exploration by strengthening disturbance in the early stage and in the final stage to intensify local search step by step, thereby acquiring high-precision solution. For the same reason, BW decreases gradually in order to reduce perturbation gradually, and the PAR's variation from 0.5 to 0.3 is to reduce the probability of pitch adjustment.

Equation (27) shows that the random mutation operation is changed dynamically from $5/D$ to $3/D$ with iterations. It can make the random disturbance change from strong to weak, and thus HSTL algorithm has strong global exploration ability in the early stage and has effective local exploitation ability in the latter stage:

$$P_m = \frac{5}{D} - \frac{2t/T_{\max}}{D}. \tag{27}$$

4. Other Related Techniques for Solving 0-1 Knapsack Problem

The 0-1 knapsack problem is a large scale, multiconstraint, and nonlinear integer programming problem. So solving 0-1 knapsack problems with HSTL algorithm needs to employ constraint handling technique and integer processing method.

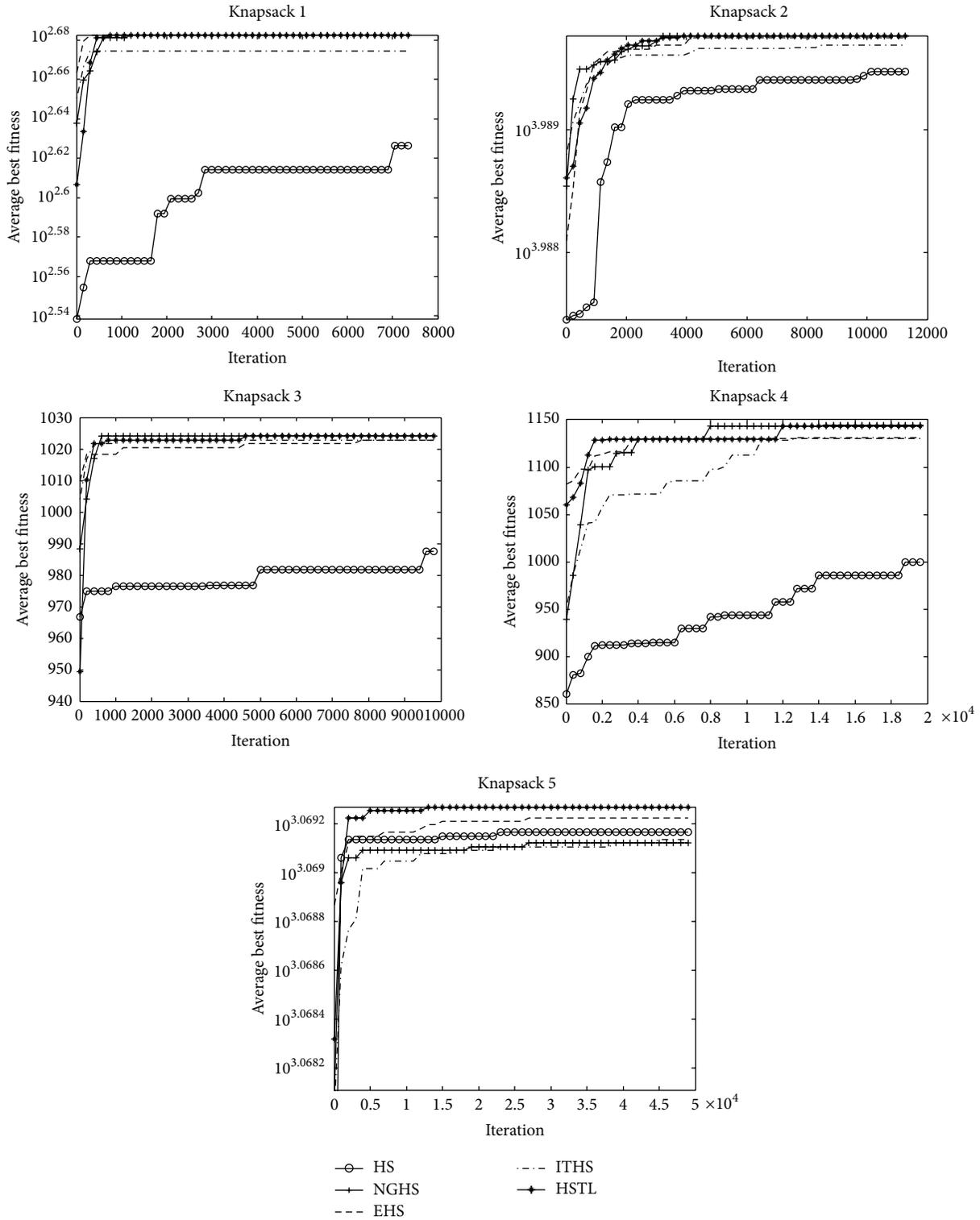


FIGURE 4: The convergence graphs of KP_1 – KP_5 .

4.1. *Constraint-Handling Technique.* For constrained optimization problems, there have existed many successful constraint-handling techniques on solving constrained optimization problems, such as penalty function method, special representations and operators, repair method, and

multiobjective method [40–53]. In this paper, because we mainly do some research on discrete optimization problems by using the HS algorithm, so special handling technique and the revision methods [23–28] for adjusting the 0-1 knapsack problems have not been adopted in this paper.

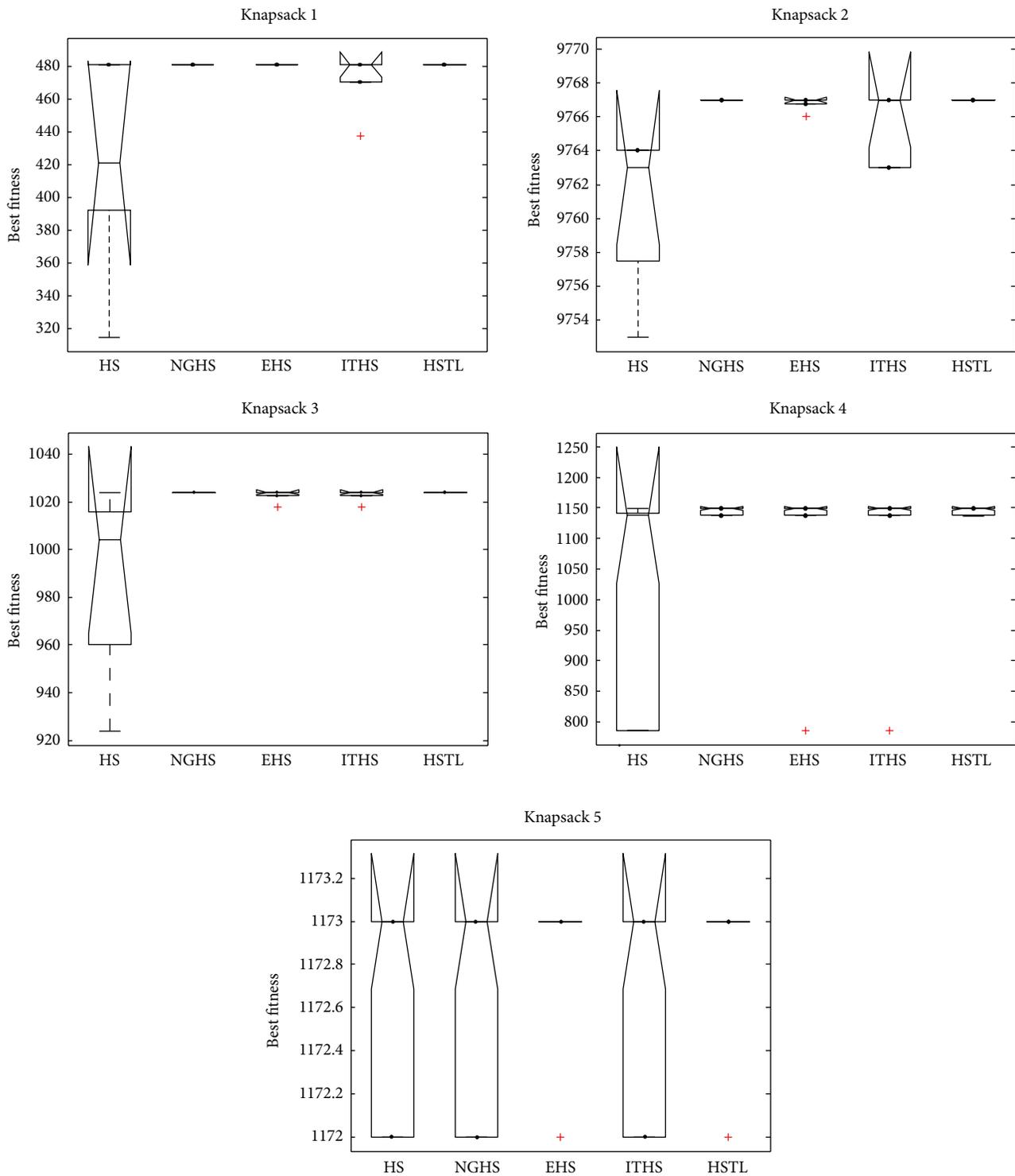


FIGURE 5: The box plots of $KP_1 - KP_5$.

In this paper, the multiobjective method has been used for handling constrained 0-1 knapsack problems. The multiobjective method [42] is as follows.

(1) Any feasible solution is better than any infeasible solution.

(2) Among two feasible solutions, the one that has a better objective function value is preferred.

(3) Among two infeasible solutions, the one that has a smaller degree of constraint violation is preferred. The purpose is to render the infeasible solution with large

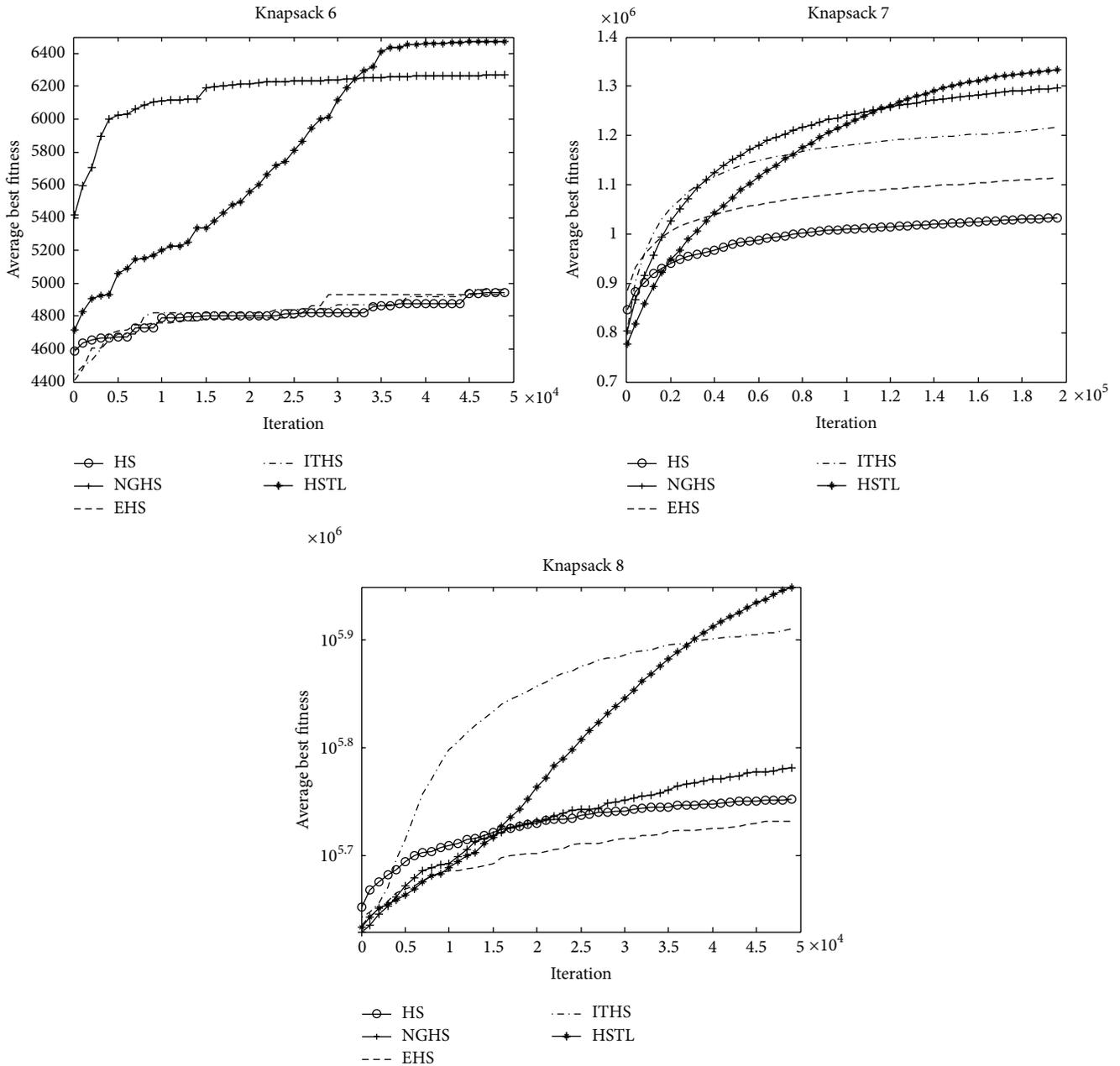


FIGURE 6: The convergence graphs of KP_6 – KP_8 .

degree of constraint violation from moving gradually towards the solution with no constraint violation or with smaller degree of constraint violation.

However, for a knapsack problem with many items ($D > 10000$) and the knapsack with small capacity, if we only execute HS algorithm without any revise method to this small-capacity knapsack, all of the harmony vectors in HM are probably infeasible. In this case, even in the very good methods it is hard to obtain the feasible solutions. So, for an infeasible solution, we will do simple revise through randomly removing some item from this knapsack. In this

way, the infeasible solution can gradually turn into the feasible solution.

4.2. Integer Processing Method. In HSTL algorithm, because the variables are adjusted by the teaching-learning strategy, local pitch adjusting strategy and random mutation operator are real numbers, so every variable is replaced by the nearest integer, that is, as follows:

$$x' = \text{round}(x). \tag{28}$$

Let $x = (0.8, 0.3, 0.9, 0.6, 1.2, 0.4)$. Then $x' = \text{round}(x) = (1, 0, 1, 1, 1, 0)$.

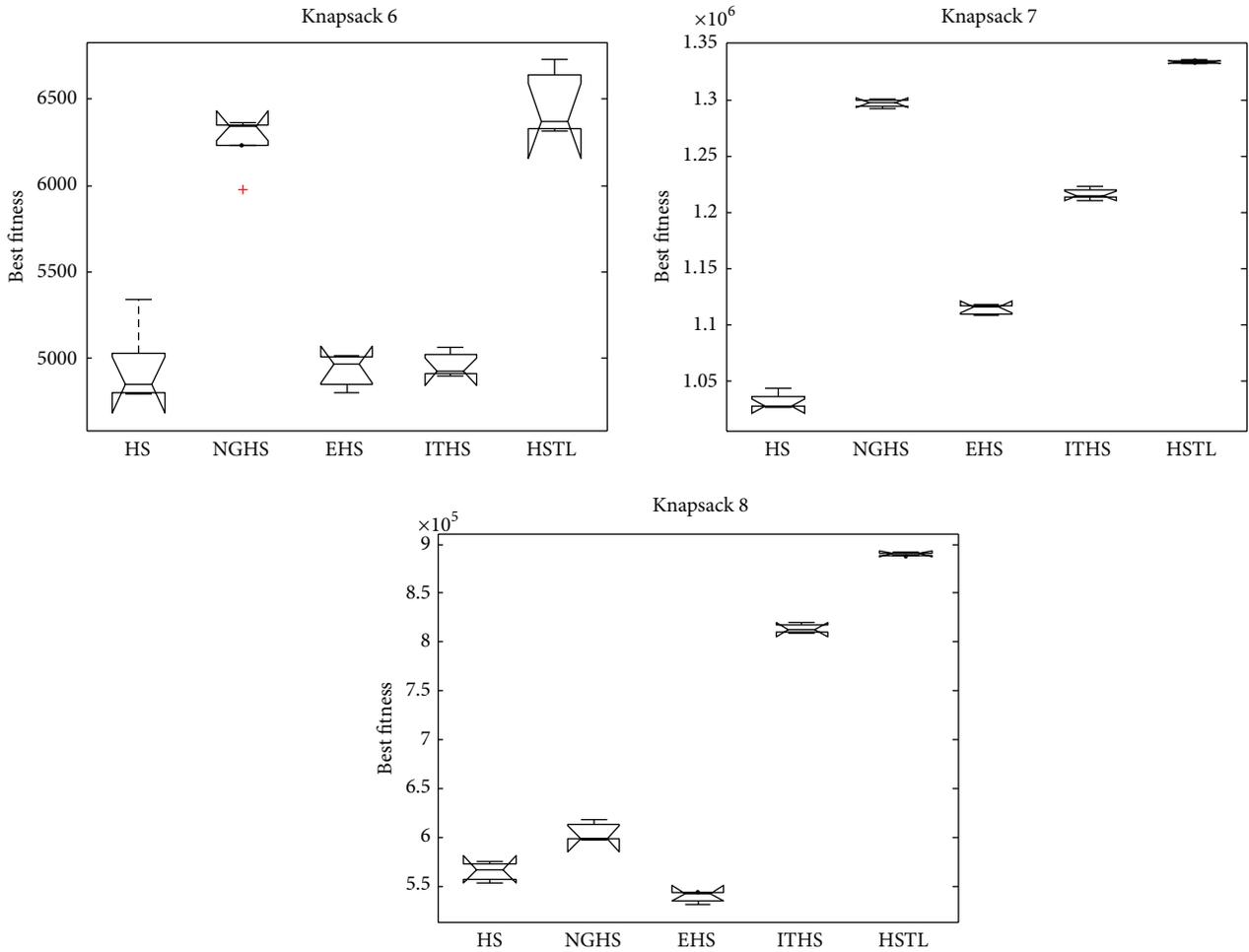


FIGURE 7: The box plots of KP_6 - KP_8 .

5. Solving 0-1 Knapsack Problems with HSTL Algorithm

5.1. Experimental Setup and Parameters Setting. In order to evaluate the performance of the HSTL algorithm, we used a set of 13 knapsack problems (KP_1 - KP_{13}). KP_1 - KP_3 is quoted, respectively, from f_5 , f_8 and f_{10} in the literature [21]. KP_4 - KP_{13} is quoted from website <http://homepage.ntlworld.com/walter.barker2/Knapsack%20Problem.htm>. In all the knapsack problems (KP_1 - KP_{13}), KP_1 - KP_8 are called one-dimensional problems that only include weight constraint and KP_9 - KP_{13} are called two-dimensional problems that include both weight constraint and volume constraint.

All simulation experiments are carried out to compare the optimization performance of the presented method (HSTL) with respect to (a) classical HS, (b) NGHS, (c) EHS, and (d) ITHS. In the experiments, the parameters setting for the compared HS algorithms is shown in Table 2. To make the comparison fair, the populations for all the competitor algorithms were initialized using the same random seeds. The variants of HS algorithm were set at the same termination

criteria: the number of improvisations (function evaluation times: FEs) $FEs = 500 \times D$, respectively. However, if the $FEs > 5E + 05$, then set $FEs = 5E + 05$.

The best and worst fitness value of each test problem are recorded for 30 independent runs; the mean fitness, standard deviation (Std), and mean runtime of each knapsack problem are calculated for 30 independent runs.

5.2. The Experimental Results and Analysis. Table 3 reports the worst, mean, best, and Std of problem results by applying the five algorithms (HS, NGHS, EHS, ITHS, and HSTL) to optimize the knapsack problems KP_1 - KP_8 , respectively. The best results are emphasized in boldface. Figures 4 and 6 illustrate the convergence characteristics in terms of the best values of the median run of each algorithm for knapsack problems KP_1 - KP_8 . Figures 5 and 7 demonstrate the performance and stability characteristics according to the distributions of the best values of 30 runs of each algorithm for knapsack problems KP_1 - KP_8 .

Based on the resulting data in Table 3, the optimal objective values (best, mean, worst, and Std) can be easily

TABLE 2: Parameter settings for the compared HS algorithms (HS, NGHS, EHS, ITHS, and HSTL).

Algorithm	HMS	HMCR	PAR	BW	others
HS	5	0.99	0.33	$BW = (X^U - X^L)/1000$	/
NGHS	5	/	/	/	/
EHS	50	0.99	0.33	$BW = 1.17\sqrt{(1/HMS) \sum_{i=1}^{HMS} (x_i - \bar{x})^2}$	/
ITHS	10	0.99	0.33	$BW_{max} = (X^U - X^L)/2$ $BW_{min} = (X^U - X^L)/10$	/
HSTL	10	$HMCR_{max} = 0.95$ $HMCR_{min} = 0.6$	$PAR_{max} = 0.5$ $PAR_{min} = 0.2$	$BW_{max} = (X^U - X^L)/2$ $BW_{min} = (X^U - X^L)/10$	$TLP_{max} = 0.55,$ $TLP_{min} = 0.15$

TABLE 3: The result of 1-dimensional (weight versus value) knapsack problems.

Problem	D	Target weight	Total weight	Total values	Optimal result	Algorithm	Outcomes				Runtime
							Worst	Mean	Best	Std	
KP ₁	15	375	741.9172	62.9963	481.069	HS	314.9297	423.191	481.069	67.92161	0.169532
						NGHS	481.069	481.069	481.069	6.4E-14	0.173909
						EHS	481.069	481.069	481.069	6.4E-14	0.814028
						ITHS	437.9345	472.4424	481.069	19.2905	0.447476
						HSTL	481.069	481.069	481.069	6.4E-14	0.330757
KP ₂	23	10000	19428	19309	9767	HS	9747	9760	9767	5.533596	0.313597
						NGHS	9767	9767	9767	0	0.305331
						EHS	9643	9751.6	9767	22.64509	1.595463
						ITHS	9756	9765.7	9767	2.641186	0.790986
						HSTL	9767	9767	9767	0	0.585312
KP ₃	20	878	1098	1085	1024	HS	924	987.4	1024	40.395544	0.254834
						NGHS	1024	1024	1024	0	0.245683
						EHS	1018	1022.8	1024	2.6832816	1.274226
						ITHS	1018	1022.8	1024	2.6832816	0.654352
						HSTL	1024	1024	1024	0	0.485799
KP ₄	40	15	374	14049	1149	HS	786	1109.4	1149	113.7494	12.51123
						NGHS	1138	1147.9	1149	3.478505	12.72128
						EHS	786	1109.4	1149	113.7494	15.64561
						ITHS	786	1112.7	1149	114.7907	13.6885
						HSTL	1138	1147.9	1149	3.478505	13.29233
KP ₅	100	27	1360	34965	1173	HS	1172	1172.5	1173	0.508548	37.14283
						NGHS	1172	1172.633	1173	0.490133	37.22416
						EHS	1172	1172.567	1173	0.504007	53.20237
						ITHS	1172	1172.567	1173	0.504007	41.50243
						HSTL	1172	1172.9	1173	0.305129	37.11053
KP ₆	10000	431	349354	3011792	unknown	HS	4795	4941	5338	228.20495	2902.844
						NGHS	5976	6270.4	6363	165.43821	1881.969
						EHS	4797	4929.8	5014	94.099416	16979.43
						ITHS	4899	4960.6	5064	72.081898	5047.957
						HSTL	6318	6474.4	6730	185.7156	1506.648
KP ₇	10000	1765326	5033006	2063406	unknown	HS	1068628	1070926.5	1073225	3250.5699	4526.0894
						NGHS	1126584	1127050.5	1127517	659.73063	5060.0062
						EHS	1041435	1043733.5	1046032	3250.5699	5435.1693
						ITHS	1124261	1129389	1134517	7252.0871	4555.54
						HSTL	1193743	1194524	1195304	1103.794	3326.1473
KP ₈	11000	1000000	5526981	2263955	unknown	HS	529153	532801.8	537353	3001.1646	10633.825
						NGHS	738654	762855.4	798991	26865.404	11912.341
						EHS	480038	486268.8	490627	4481.0616	13084.292
						ITHS	795202	800908	807164	4864.5424	10785.623
						HSTL	940656	944780	947850	2630.108	9606.735

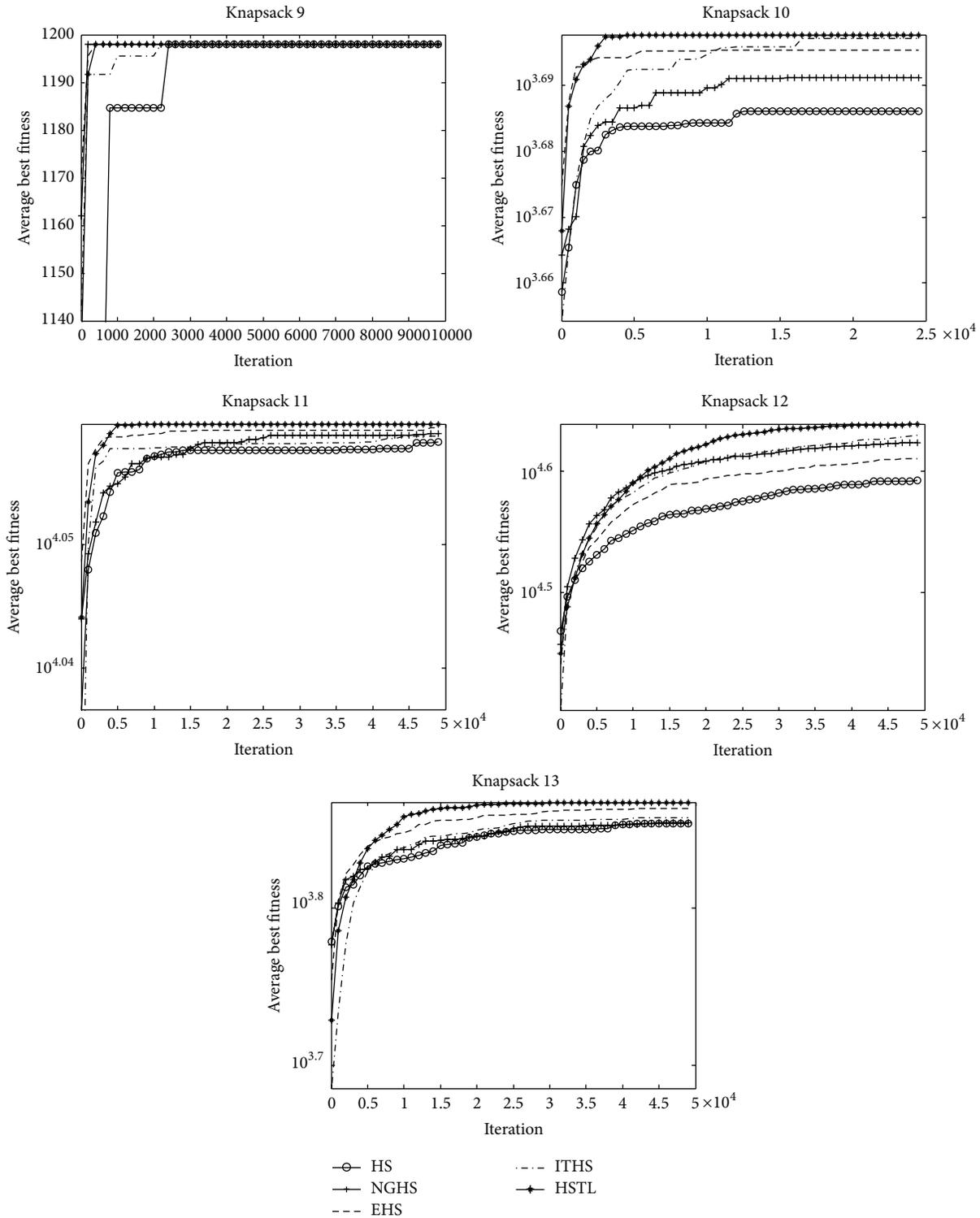


FIGURE 8: The convergence graphs of KP₉–KP₁₃.

obtained by NGHS, EHS, ITHS, and the HSTL algorithms with high accuracy on the knapsack problems KP₁–KP₅. However, comparatively speaking, NGHS and the HSTL algorithm have better optimal results on worst, mean, best,

and Std, and thus NGHS method and the HSTL algorithm are more effective and stabilized to solve the problems KP₁–KP₅.

For the high-dimensional knapsack problems KP₆–KP₈, Table 3 shows that the HSTL algorithm has obvious

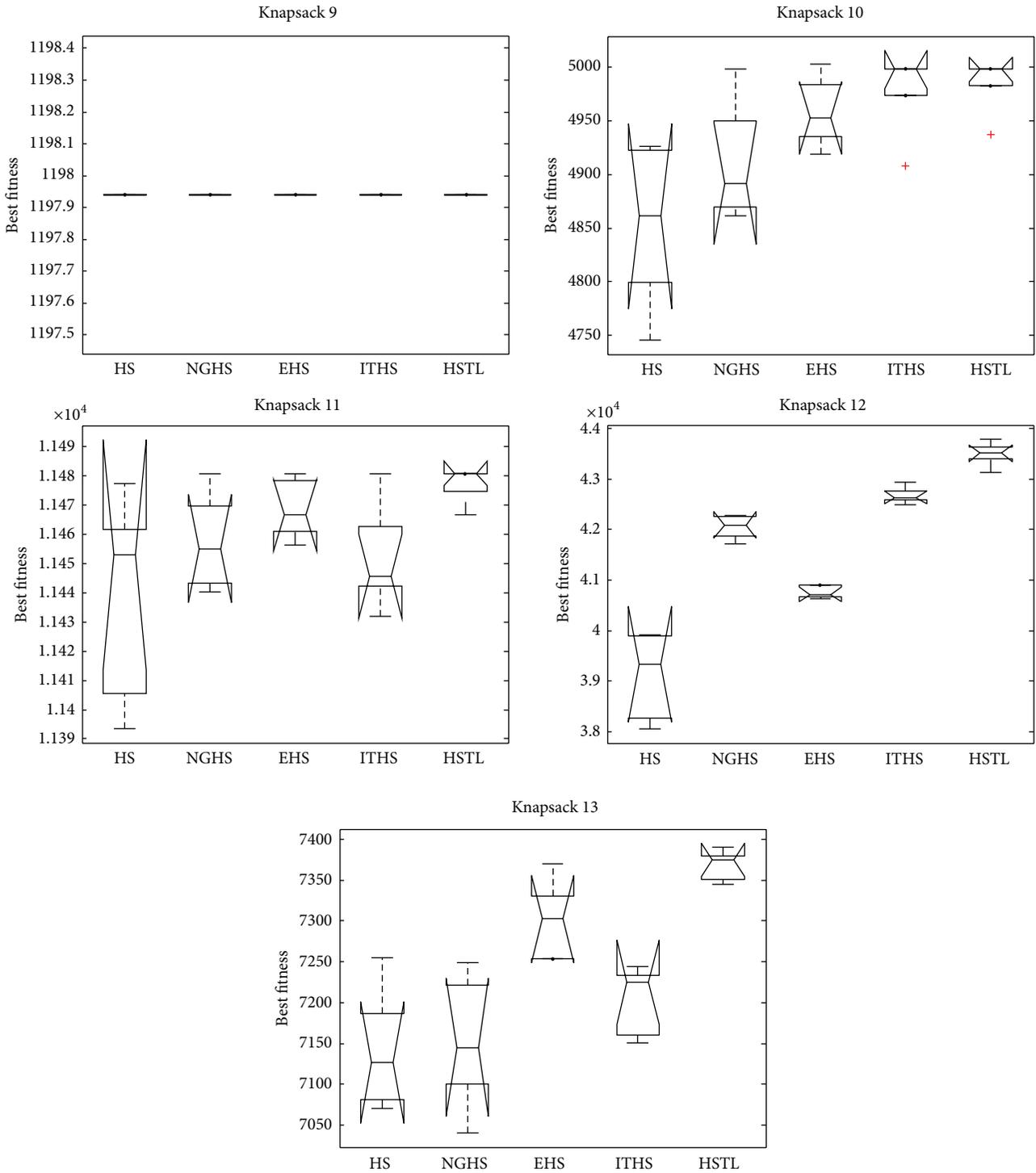


FIGURE 9: The box plots of KP₉-KP₁₃.

advantages over other variants of HS algorithms. Comparing with other HS algorithms, although HSTL algorithm has slow convergence speed in the early stage, it can be constant to optimize the solutions and obtain high precision solutions in the latter stage, which can be seen from Figure 6.

It is evident from Figure 7 that HSTL algorithm has better convergence, stability, and robustness in most cases than HS, NGHS, EHS, and ITHS algorithms.

Table 4 shows the experimental results on algorithms HS, NGHS, EHS, ITHS, and HSTL for two-dimensional knapsack problems: KP₉-KP₁₃. Figure 8 shows the convergence graphs, and Figure 9 is the box plots of independent 30 runs of knapsack problems KP₉-KP₁₃.

It can be seen evidently from Table 4 that the HSTL algorithm attained the optimal best, mean, worst, and Std results among all two-dimensional knapsack problems.

TABLE 4: The result of 2-dimensional (weight and volume versus value) knapsack problems.

Problem	D	Target weight volume	Total weight	Total volume	Total values	Optimal result	Algorithm	Outcomes				Runtime
								Worst	Mean	Best	Std	
KP ₉	20	20 25	111.63	170.27	3813.6	1197.94	HS	1197.94	1197.94	1197.94	0	7.90831
							NGHS	1197.94	1197.94	1197.94	0	8.092175
							EHS	1197.94	1197.94	1197.94	0	8.978277
							ITHS	1197.94	1197.94	1197.94	0	8.588568
							HSTL	1197.94	1197.94	1197.94	0	8.31578
KP ₁₀	50	114 133	400.56	408.57	10211	5002.81	HS	4849.19	4935.062	5002.81	45.16856	19.03619
							NGHS	4917.42	4963.421	5002.81	32.60289	19.15758
							EHS	4933.24	4983.169	4998.27	24.18182	23.48809
							ITHS	4926.07	4975.32	5002.81	29.72861	20.82369
							HSTL	4995.48	5000.063	5002.81	2.981432	18.9528
KP ₁₁	100	1500 3000	3090.5	4599.2	15278	11480.84	HS	11393.59	11438.024	11477.52	35.010842	51.07192
							NGHS	11440.36	11457.378	11480.84	16.483244	51.2287
							EHS	11456.44	11468.79	11480.84	10.221502	65.89753
							ITHS	11431.96	11452.068	11480.84	18.277327	54.87239
							HSTL	11466.63	11477.33	11480.84	6.153989	45.31429
KP ₁₂	1000	30000 40000	250320	400090	157890	unknown	HS	38064.94	39108.266	39924.89	866.36169	414.50957
							NGHS	41709.51	42048.386	42280.62	235.6772	424.71994
							EHS	40627.36	40766.678	40903.3	123.45544	454.46526
							ITHS	42485.83	42671.15	42928.94	163.78948	439.51866
							HSTL	43953.97	44119.987	44409.66	119.80076	393.5962
KP ₁₃	300	800 700	13605	14157	48464	unknown	HS	7070.4	7140.222	7254.76	73.854458	36.347165
							NGHS	7040.92	7153.61	7249.1	81.266043	36.41226
							EHS	7253.67	7299.32	7369.58	48.490527	39.440654
							ITHS	7150.24	7202.6	7244.01	42.540662	39.450648
							HSTL	7345.46	7367.95	7390.52	18.42674	29.889155

Bold indicates best results.

From the convergence graphs (Figure 8), it can be seen that the HSTL algorithm has a strong search ability and convergence throughout the search process for the two-dimensional knapsack problems. As can be seen from the box plots (Figure 9), the HSTL has demonstrated some advantage over the other four algorithms on solving two-dimensional 0-1 knapsack problems.

6. Conclusion

In this paper, a novel harmony search with teaching-learning strategies is presented to solve 0-1 knapsack problems. The HSTL algorithm employs the idea of teaching and learning. Four strategies are used to maintain the proper balance between exploration power and exploitation power. With the process of evolution, the dynamic strategy is adopted to change the parameters HMCR, TLP, BW, and PAR. Experimental results showed that the HSTL algorithm has stronger ability to resolve the high-dimensional 0-1 knapsack problems. However, the HSTL algorithm has more parameters. In the future, we should focus on improving the structure of HSTL algorithm, decreasing the running cost and enhancing efficiency for solving complex optimization problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant no. 81160183, Scientific Research Program funded by Shaanxi Provincial Education Department under Grant no. 12JK0863, Ministry of Education "Chunhui Plan" project (no. Z2011051), Natural Science Foundation of Ningxia Hui Autonomous Region (no. NZ12179), and Scientific Research Fund of Ningxia Education Department (no. NGY2011042).

References

- [1] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.
- [2] Z. W. Geem, "Novel derivative of harmony search algorithm for discrete design variables," *Applied Mathematics and Computation*, vol. 199, no. 1, pp. 223–230, 2008.
- [3] L. D. S. Coelho and V. C. Mariani, "An improved harmony search algorithm for power economic load dispatch," *Energy Conversion and Management*, vol. 50, no. 10, pp. 2522–2526, 2009.
- [4] A. Askarzadeh and A. Rezaeizadeh, "A grouping-based global harmony search algorithm for modeling of proton exchange membrane fuel cell," *International Journal of Hydrogen Energy*, vol. 36, no. 8, pp. 5047–5053, 2011.
- [5] S. Ghosh, D. Kundu, K. Suresh, S. Das, and A. Abraham, "Design of optimal digital IIR filters by using a bandwidth adaptive harmony search algorithm," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 481–486, Coimbatore, India, December 2009.
- [6] D. C. Hoang, P. Yadav, R. Kumar, and S. K. Panda, "A robust harmony search algorithm based clustering protocol for wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications Workshops (ICC '10)*, Cape Town, South Africa, May 2010.
- [7] M. Jaberipour and E. Khorram, "Solving the sum-of-ratios problems by a harmony search algorithm," *Journal of Computational and Applied Mathematics*, vol. 234, no. 3, pp. 733–742, 2010.
- [8] M. Poursha, F. Khoshnoudian, and A. S. Moghadam, "Harmony search based algorithms for the optimum cost design of reinforced concrete cantilever retaining walls," *International Journal of Civil Engineering*, vol. 9, no. 1, pp. 1–8, 2011.
- [9] A. H. Khazali and M. Kalantar, "Optimal reactive power dispatch based on harmony search algorithm," *International Journal of Electrical Power and Energy Systems*, vol. 33, no. 3, pp. 684–692, 2011.
- [10] A. H. Khazali, A. Parizad, and M. Kalantar, "Optimal voltage/reactive control by an improve harmony search algorithm," *International Review of Electrical Engineering*, vol. 5, no. 1, pp. 217–224, 2010.
- [11] E. Khorram and M. Jaberipour, "Harmony search algorithm for solving combined heat and power economic dispatch problems," *Energy Conversion and Management*, vol. 52, no. 2, pp. 1550–1554, 2011.
- [12] D. Zou, L. Gao, J. Wu, and S. Li, "Novel global harmony search algorithm for unconstrained problems," *Neurocomputing*, vol. 73, no. 16–18, pp. 3308–3318, 2010.
- [13] Q.-K. Pan, P. N. Suganthan, M. F. Tasgetiren, and J. J. Liang, "A self-adaptive global best harmony search algorithm for continuous optimization problems," *Applied Mathematics and Computation*, vol. 216, no. 3, pp. 830–848, 2010.
- [14] Q.-K. Pan, P. N. Suganthan, J. J. Liang, and M. F. Tasgetiren, "A local-best harmony search algorithm with dynamic subpopulations," *Engineering Optimization*, vol. 42, no. 2, pp. 101–117, 2010.
- [15] S. Tuo and L. Yong, "An improved harmony search algorithm with chaos," *Journal of Computational Information Systems*, vol. 8, no. 10, pp. 4269–4276, 2012.
- [16] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation*, vol. 198, no. 2, pp. 643–656, 2008.
- [17] D. Zou, L. Gao, J. Wu, S. Li, and Y. Li, "A novel global harmony search algorithm for reliability problems," *Computers and Industrial Engineering*, vol. 58, no. 2, pp. 307–316, 2010.
- [18] P. Yadav, R. Kumar, S. K. Panda, and C. S. Chang, "An intelligent tuned harmony search algorithm for optimisation," *Information Sciences*, 2012.
- [19] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, and B. K. Panigrahi, "Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 41, no. 1, pp. 89–106, 2011.
- [20] H. Sarvari and K. Zamanifar, "Improvement of harmony search algorithm by using statistical analysis," *Artificial Intelligence Review*, vol. 37, no. 3, pp. 181–215, 2012.
- [21] D. Zou, L. Gao, S. Li, and J. Wu, "Solving 0-1 knapsack problem by a novel global harmony search algorithm," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1556–1564, 2011.

- [22] Y. Liu and C. Liu, "A schema-guiding evolutionary algorithm for 0-1 knapsack problem," in *Proceedings of the International Association of Computer Science and Information Technology—Spring Conference (IACSIT-SC '09)*, pp. 160–164, April 2009.
- [23] H. Shi, "Solution to 0/1 knapsack problem based on improved ant colony algorithm," in *Proceedings of the IEEE International Conference on Information Acquisition (ICIA '06)*, pp. 1062–1066, August 2006.
- [24] F.-T. Lin, "Solving the knapsack problem with imprecise weight coefficients using genetic algorithms," *European Journal of Operational Research*, vol. 185, no. 1, pp. 133–145, 2008.
- [25] V. Boyer, D. El Baz, and M. Elkihel, "Solving knapsack problems on GPU," *Computers and Operations Research*, vol. 39, no. 1, pp. 42–47, 2012.
- [26] R. R. Hill, Y. K. Cho, and J. T. Moore, "Problem reduction heuristic for the 0-1 multidimensional knapsack problem," *Computers and Operations Research*, vol. 39, no. 1, pp. 19–26, 2012.
- [27] A. Gherboudj, A. Layeb, and S. Chikhi, "Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 229–236, 2012.
- [28] A. Layeb, "A novel quantum inspired cuckoo search for knapsack problems," *International Journal of Bio-Inspired Computation*, vol. 3, no. 5, pp. 297–305, 2011.
- [29] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems," *Computer Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.
- [30] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems," *Information Sciences*, vol. 183, no. 1, pp. 1–15, 2012.
- [31] R. V. Rao and V. Patel, "An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 3, pp. 535–560, 2012.
- [32] R. V. Venkata, R. Savsani, and V. J. Rao, *Mechanical Design Optimization Using Advanced Optimization Techniques*, Springer, London, UK, 2012.
- [33] R. V. Rao and V. Patel, "Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm," *Applied Mathematical Modelling*, vol. 37, no. 3, pp. 1147–1162, 2013.
- [34] R. Venkata Rao and V. Patel, "Multi-objective optimization of two stage thermoelectric cooler using a modified teaching-learning-based optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 430–445, 2013.
- [35] R. V. Rao, V. J. Savsani, and J. Balic, "Teaching-learning-based optimization algorithm for unconstrained and constrained real-parameter optimization problems," *Engineering Optimization*, vol. 44, no. 12, pp. 1447–1462, 2012.
- [36] R. V. Rao and V. Patel, "An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems," *Scientia Iranica*, vol. 20, no. 3, pp. 710–720, 2013.
- [37] P. J. Pawar and R. V. Rao, "Parameter optimization of machining processes using teaching-learning-based optimization algorithm," *The International Journal of Advanced Manufacturing Technology*, pp. 1–12, 2012.
- [38] R. Rao and V. Patel, "Comparative performance of an elitist teaching-learning-based optimization algorithm for solving unconstrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 4, no. 1, 2013.
- [39] S. Tuo, L. Yong, and T. Zhou, "An improved harmony search based on teaching-learning strategy for unconstrained optimization problems," *Mathematical Problems in Engineering*, vol. 2013, Article ID 413565, p. 29, 2013.
- [40] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002.
- [41] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.
- [42] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 658–675, 2006.
- [43] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.
- [44] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 658–675, 2006.
- [45] T.-C. Chiang and Y.-P. Lai, "MOEA/D-AMS: Improving MOEA/D by an adaptive mating selection mechanism," in *Proceedings of the IEEE Congress of Evolutionary Computation (CEC '11)*, pp. 1473–1480, June 2011.
- [46] N. Bacanin and M. Tuba, "Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators," *Studies in Informatics and Control*, vol. 21, no. 2, pp. 137–146, 2012.
- [47] D. Bianchi, S. Genovesi, and A. Monorchio, "Constrained Pareto Optimization of Wide Band and Steerable Concentric Ring Arrays," *IEEE Transactions on Antennas and Propagation*, vol. 60, no. 7, pp. 3195–3204, 2012.
- [48] P.-T. Chang and J.-H. Lee, "A fuzzy DEA and knapsack formulation integrated model for project selection," *Computers and Operations Research*, vol. 39, no. 1, pp. 112–125, 2012.
- [49] M. Daneshyari and G. G. Yen, "Constrained multiple-swarm particle swarm optimization within a cultural framework," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 42, no. 2, pp. 475–490, 2012.
- [50] R. Datta and K. Deb, "A fast and accurate solution of constrained optimization problems using a hybrid bi-objective and penalty function approach," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '12)*, Barcelona, Spain, July 2012.
- [51] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-operator based evolutionary algorithms for solving constrained optimization problems," *Computers and Operations Research*, vol. 38, no. 12, pp. 1877–1896, 2011.
- [52] A. H. Gandomi, X.-S. Yang, S. Talatahari, and S. Deb, "Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization," *Computers and Mathematics with Applications*, vol. 63, no. 1, pp. 191–200, 2012.
- [53] A. W. Mohamed and H. Z. Sabry, "Constrained optimization based on modified differential evolution algorithm," *Information Sciences*, vol. 194, pp. 171–208, 2012.

Research Article

A Game Theoretical Model for Location of Terror Response Facilities under Capacitated Resources

Lingpeng Meng,^{1,2} Qi Kang,¹ Chuanfeng Han,² Weisheng Xu,¹ and Qidi Wu¹

¹ Department of Control Science and Engineering, Tongji University, Shanghai 201804, China

² Institute of Urban Construction and Emergency Management, Tongji University, Shanghai 200092, China

Correspondence should be addressed to Chuanfeng Han; hanchuanfeng@hotmail.com

Received 2 September 2013; Accepted 17 November 2013

Academic Editors: Z. Cui and X. Yang

Copyright © 2013 Lingpeng Meng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper is concerned with the effect of capacity constraints on the locations of terror response facilities. We assume that the state has limited resources, and multiple facilities may be involved in the response until the demand is satisfied consequently. We formulate a leader-follower game model between the state and the terrorist and prove the existence and uniqueness of the *Nash* equilibrium. An integer linear programming is proposed to obtain the equilibrium results when the facility number is fixed. The problem is demonstrated by a case study of the 19 districts of Shanghai, China.

1. Introduction

Different from regular natural calamity such as earthquakes, terrorist attacks have low frequency but tremendous magnitude [1]. As a result, the attacks can lead to a big demand for medical supplies, which overwhelm the local emergency responders, and regional/national assistance will be required. Moreover, it contains human intelligence and intent in terrorist attacks, just as the 9/11 Commission Report states, “The terrorists analyze defenses. They plan accordingly.” The interaction between the government and the terrorist increases the complication in the issue of counterterrorism.

Faced with terrorist attacks, implementing an efficient disbursement of the requisite emergency resources is critical in reducing morbidity and mortality; therefore, the need to be prepared to respond to the terrorist attacks is highlighted. Consequently, there is increasing interest in strategic facility location for protecting assets of value (including human lives) against terrorist attacks. Facilities that contain resources required to respond to terror attacks are referred to as terror response facility [2]. This paper will address the location problem of terror response facility under capacitated resources.

Location problem of emergency service facilities such as ambulance, police, and fire-fighting assistance has been

previously investigated, and many models and their extensions have been developed to formulate and solve the problem, including covering models, p -median models, p -center models, and their extensions, for example, [3–6]. One could see [7, 8] for recent reviews. Such emergency facility location models (covering models, p -median models, p -center models, and their extensions) typically have not considered the attacker’s analysis and response to the defender’s strategy and tend to focus more on emergency response than on prevention. Many literatures address the strategic nature of facility location problems by modeling either stochastic or dynamic integer programming formulations, but as empirical data are likely to be sparse, classical statistical methods have been of relatively little use. The distinction between probabilistic and strategic risks has emerged recently, see; [9–11] and so forth. Snyder [12] provides a complete review on facility location problems with uncertainties. Unfortunately, most concentrate on common situations and few focus on conditions with a tremendous demand and low frequency which may occur in terror emergencies.

In addition, one critical assumption in most literatures is that the assigned facility can meet all the demands, which is not always the case for terrorist attacks. In fact, the state have not enough resources to cover all potential targets; thus situations with insufficient supplies will occur facing large

uncertain demands. How to allocate the limited resources among the terror response facilities is another challenge besides the facility location.

Game theory is an appropriate tool for investigating counterterrorism as it captures the strategic interactions between terrorists and targeted state whose choices are interdependent. With respect to applications of game theory and related methods to security in general, there is a large number of work already, that is, to inform policy-level decisions such as public versus private funding of defensive investments [13] or the relative merits of deterrence and other protective measures, for instance, [14–17]; however, few has gone deep into operational level.

Both empirical and theoretical analysis showed that raising a potential target's defense level will increase the probability for another target to be attacked, and limited resources should be allocated to maximize the overall utility [18, 19]. The framework of game theory can effectively deal with resource allocation problems [20]. The efficiency and equity of capacitated resource allocation problems have been studied [21]. It is verified that making decision in global manner is superior to that in decentralized manner when allocating resources between two targets [22]. Powell [23] proposed a two-stage game between the terrorist and the state, to allocate capacitated resources between two targets, and extended the model to the case of N targets. Based on this analysis framework, some more points were studied, including the effect of the state's information [24], the interaction of more variables [25], the attacking behavior of the terrorist [10], and preassessment for utility of both sides [26]. Some correct but counterintuitive conclusions were deduced. First, as the arms race, the attack of the terrorist is positively correlated with the defense level of the state; second, the state has first-mover advantage in the sense that the state can force the terrorist to attack a target with smaller vulnerability; third, it will benefit the state to make the resource allocation policies open; finally, some targets with less loss for the state will be left unprotected under capacitated resources.

In a game-theoretical model, the introduction of capacity constraint will usually change the equilibrium results, even when the initial model is simple [27]. The pure equilibrium may be replaced by mixed equilibrium, or there will no equilibrium, as a result the complication of solution will increase [28]. At the same time, the change of the rationing rule, that is, the allocation of residual demand when one facility could not satisfy the whole demand, will probably change the equilibrium results [29]. We will solve the Nash equilibrium of the problem using an integer program, that could simplify the solution of the game.

To our knowledge, the only rigorous model to date to deal with the terror response facility problem is exploratory analyses by Berman and Gavius [2] and Berman et al. [30]. The state decides where to locate the terror response facilities, and the terrorist decides which node to strike with complete or incomplete information of the distribution of the facilities. Both papers assume that there is no capacity constraints on the facilities; that is, the facility could provide enough resource to the nearest node attacked. We will extend this assumption to the case that the state has capacitated

resources, and he has to decide how to allocate the resources to each facility besides the location of facilities taking into consideration the behavior of the terrorist.

This paper assumes that the state has capacitated resources and could not satisfy all the demand simultaneously. When the facility nearest to the attacked city could not meet the demand, the second nearest facility will be involved in the rescue, and so on, until the required resources are satisfied. The state could spread the limited resource among more facilities to improve the overall coverage or adopt a centralized manner, among fewer facilities to protect key areas. A balance between resource quantity and allocation manner must be considered carefully.

The next section will formulate the problem. In Section 3, the Nash equilibrium of the game is discussed, and a simple example is given. Section 4 proposes a linear integer programming to solve the equilibrium when the facility number is fixed. A case study is given in Section 5. Finally, we give conclusions in Section 6.

2. Problem Formulation

Consider an undirected connected network $G(V, E)$, where $V = (v_1, v_2, \dots, v_n)$ is the set of nodes, and $E = (e_1, e_2, \dots, e_m)$ is the set of edges. The nodes are demand cities, and each city v_i is associated with a weight ω_{v_i} to represent the amount of resources needed every unit time. The shortest distance between node v_i and v_j is $d[v_i, v_j]$, representing the delay of shipment of resources from v_j to v_i . Let $R = \{r_1, r_2, \dots, r_m\}$ be the potential locations of the terror response facilities, $m \leq n$. Without loss of generality, let $R \subset V$.

It is assumed that the resources needed of city v_i is ω_{v_i} . Let the total resources of the state be a constant C , $\max\{\omega_{v_i}\} \leq C < \sum_{i=1}^n \omega_{v_i}$; that is to say, the state could satisfy the demand of every single city but could not eliminate the loss of terrorist attacks completely by establishing terror response facilities at each city. The resources associated with each facility r_j , $j = 1, 2, 3, \dots, m$ are c_j . For simplicity, we consider the symmetric condition; that is, $c_1 = c_2 = \dots = c_m = c$. Obviously, $C = mc$.

Once the city v_i is attacked, the nearest terror response facility will begin to provide the resources. If the nearest facility could not satisfy the demand, that is, $c < \omega_{v_i}$, the second nearest facility will be involved in the rescue, and so on. Let

$$D^{(v_i)} = \{d[v_i, r_1], \dots, d[v_i, r_m]\}, \quad (1)$$

in which when $i < j$, $d[v_i, r_i] < d[v_i, r_j]$. The facilities will be involved in the rescue sequentially from r_1 to r_m . Let $p = \lceil \omega_{v_i}/c \rceil$ be the smallest integer not less than ω_{v_i}/c then p facilities will be needed.

As Berman and Gavius [2], the players of the game are the state and the terrorist. We assume that the loss of the state f_G is a linear function of the time delay of the resources,

$$f_G = (d[v_i, r_1] + \dots + d[v_i, r_{p-1}]) \cdot c + d[v_i, r_p] \cdot (\omega_{v_i} - (p-1)c). \quad (2)$$

The utility of the terrorist is the loss of the state,

$$U_T = f_G. \tag{3}$$

We consider a zero-sum game; then the utility of the state is $U_G = -U_T$.

The interaction is modeled as a sequential game with complete information. The state acts first. He must decide the resource allocated to each facility and the facility location such that the city attacked is the one that minimizes his disutility. The terrorist plays second and chooses a city to attack to maximize his utility observing the arrangement of the state.

3. Nash Equilibrium Solutions

The strategy of the state is $S_G = (c, R)$, and the strategy of the terrorist is $S_T = (v_i)$. A Nash equilibrium mean that, for all $(S_G$ and $S_T)$, we have

$$\begin{aligned} U_G(S_G^*, S_T^*) &\geq U_G(S_G, S_T^*), \\ U_T(S_G^*, S_T^*) &\geq U_T(S_G^*, S_T). \end{aligned} \tag{4}$$

Obviously, the equilibrium is also the subgame perfect Nash equilibrium of the game.

3.1. Facility Number is Fixed. When the facility number m is a constant, the resources c of every facility are fixed. Then, the strategy of the state is $S_G = R = (r_1, \dots, r_m)$, and the strategy of the terrorist is $S_T = v(R)$.

Given the state's strategy R , the best response of the terrorist is

$$\begin{aligned} v^*(R) = \arg \max_{v_i} & (d[v_i, r_1] + \dots + d[v_i, r_{p-1}]) \cdot c \\ & + d[v_i, r_p] \cdot (\omega_{v_i} - (p-1)c). \end{aligned} \tag{5}$$

Forecasting the reaction of the terrorist, the state will adopt the strategy

$$\begin{aligned} R^* = \arg \min_{RCV} & (d[v^*, r_1] + \dots + d[v^*, r_{p-1}]) \\ & \cdot c + d[v^*, r_p] \cdot (\omega_{v^*} - (p-1)c). \end{aligned} \tag{6}$$

It is assumed that the storage and distribution manner of the resources is similar to the Strategic National Stockpile (SNS) of Centers for Disease Control and Prevention (CDC), USA. That is to say, the resource c at every facility will be shipped to the city attacked as a whole when the facility is involved in the rescue. In fact, this manner would increase the redundancy of resource supply and is more suitable for terrorist attacks.

Lemma 1. *When the facility number is fixed, the state equilibrium solution is the solution of a minimax problem in the nodal set, and the terrorist equilibrium solution could be obtained by (5).*

Proof. For any strategy of the state R , the best response of the terrorist is (5), and the best strategy of the state is decided

by (6). From the definition of p , $\omega_{v_i}/c \leq p < \omega_{v_i}/c + 1$. Let $f'_G = (d[v_i, r_1] + \dots + d[v_i, r_p]) \cdot c$; then $f'_G = f_G + d[v_i, r_p](pc - \omega_{v_i})$. Apparently, $0 \leq pc - \omega_{v_i} < c$; (6) could be rewritten accordingly as

$$R^* = \arg \min_{RCV} \left(\max_{v_i} ((d[v^*, r_1] + \dots + d[v^*, r_p]) \cdot c) \right). \tag{7}$$

Compared with (6), (7) could simplify the solution of the equilibrium to some extent but will not change the behavioral characteristics of the state and the terrorist. The equilibrium of the terrorist could be obtained by substituting (7) into (5). \square

3.2. Facility Number is Variable. When the facility number is variable, if there are no capacity constraints, the optimal facility number is obviously n . With the capacity constraints, one facility could not satisfy the demand, and multiple facilities will be involved. As m increases, the resources at each facility will decrease, and more facilities may be involved. Smaller m corresponds to the centralized mode, in which the state will distribute the limited resources to protect the key areas, while larger m corresponds to the dispersed mode, in which the state will distribute the resources to get larger coverage. Different m will induce different disutility, and the state must strike a balance between the two mode. The relationship between the utility of the state and the parameters c and m is demonstrated by the following lemmas.

Lemma 2. *When c is a constant, the equilibrium solution of the state $\bar{U}_G(R^*, v^*(R))$ is a monotonically increasing function of m .*

Proof. From (2) and (3), $\bar{U}_G(R^*, v^*(R)) = -((d[v^*, r_1] + \dots + d[v^*, r_{p-1}]) \cdot c + d[v^*, r_p] \cdot (\omega_{v^*} - (p-1)c))$. When c is fixed, \bar{U}_G is a monotonically increasing function of $d[v^*, r_i]$. As m increasing, there must exist a $d'[v^*, r_i] \leq d[v^*, r_i]$, such that for the new utility function \bar{U}'_G , $\bar{U}'_G \geq \bar{U}_G$. The proof is completed. \square

Lemma 3. *When m is a constant, the equilibrium solution of the state $\bar{U}_G(R^*, v^*(R))$ is a monotonically increasing function of c .*

Proof. From $p = \lceil \omega_{v^*}/c \rceil$, when c increases to c' , there will be $p' = \lceil \omega_{v^*}/c' \rceil \leq p$. Two conditions will be discussed.

When $p' = p$,

$$\begin{aligned} \bar{U}_G - \bar{U}'_G &= - (d[v^*, r_1] + \dots + d[v^*, r_{p-1}]) \cdot c - d[v^*, r_p] \\ &\quad \cdot (\omega_{v^*} - (p-1)c) + (d[v^*, r_1] + \dots + d[v^*, r_{p-1}]) \\ &\quad \cdot c' + d[v^*, r_p] \cdot (\omega_{v^*} - (p-1)c') \\ &= (d[v^*, r_1] + \dots + d[v^*, r_{p-1}]) \\ &\quad \cdot (c' - c) + d[v^*, r_p] \cdot (p-1)(c - c') \end{aligned}$$

$$\begin{aligned}
 &= (d[v^*, r_1] + \dots + d[v^*, r_{p-1}] - (p-1)d[v^*, r_p]) \\
 &\quad \cdot (c' - c) < 0.
 \end{aligned}
 \tag{8}$$

Accordingly, $\bar{U}_G < \bar{U}'_G$.

When $p' < p$, we just need to prove the correctness of the conclusion when $p' = p - 1$. Consider

$$\begin{aligned}
 &\bar{U}_G - \bar{U}'_G \\
 &= -(d[v^*, r_1] + \dots + d[v^*, r_{p-1}]) \\
 &\quad \cdot c - d[v^*, r_p] \cdot (\omega_{v^*} - (p-1)c) \\
 &\quad + (d[v^*, r_1] + \dots + d[v^*, r_{p-2}]) \\
 &\quad \cdot c' + d[v^*, r_{p-1}] \cdot (\omega_{v^*} - (p-2)c') \\
 &= (d[v^*, r_1] + \dots + d[v^*, r_{p-2}]) \\
 &\quad \cdot (c' - c) + d[v^*, r_{p-1}] \cdot (\omega_{v^*} - (p-1)c' - c) \\
 &\quad - d[v^*, r_p] \cdot (\omega_{v^*} - (p-1)c) \\
 &< (d[v^*, r_1] + \dots + d[v^*, r_{p-2}]) \\
 &\quad \cdot (c' - c) + d[v^*, r_{p-1}] \cdot (\omega_{v^*} - (p-1)c' - c) \\
 &\quad - d[v^*, r_{p-1}] \cdot (\omega_{v^*} - (p-1)c) \\
 &= (d[v^*, r_1] + \dots + d[v^*, r_{p-2}]) \cdot (c' - c) + d[v^*, r_{p-1}] \\
 &\quad \cdot ((p-1)(c - c') - c) \\
 &< (d[v^*, r_1] + \dots + d[v^*, r_{p-2}]) \\
 &\quad \cdot (c' - c) + d[v^*, r_{p-1}] \cdot (p-1)(c - c') \\
 &= (d[v^*, r_1] + \dots + d[v^*, r_{p-2}] - (p-1)d[v^*, r_{p-1}]) \\
 &\quad \cdot (c' - c) < 0.
 \end{aligned}
 \tag{9}$$

Accordingly, $\bar{U}_G < \bar{U}'_G$. The proof is completed. \square

Evidently, the facility number m and the resources c at each facility will affect the equilibrium solutions simultaneously; hence, it is not so straightforward to decide the optimal facility number and the location. However, when the total resources of the state C are constant, there is an inversely proportional relationship between c and m , so we could compare the disutility of the state under different facility numbers, and the optimal number is the one that induces the smallest disutility. We could get the following theorem.

Theorem 4. *There exists a unique c^* , such that $\{S_G^* = \{c^*, R^*\}, S_T^* = \{v^*\}$ is the Nash equilibrium solution of the game, and the optimal m could be obtained by*

$$m^* = \arg \max_{m=1, \dots, n} U_G(R^*, v^*(R)). \tag{10}$$

3.3. An Example. We will use a simple example to demonstrate the solution of the equilibrium. Consider the network in Figure 1, in which the numbers next to the edges are the distance between node pairs; that is, $d[1, 2] = 5$, $d[2, 3] = 8$, $d[1, 3] = 6$. Let $\omega_1 = 16$, $\omega_2 = 10$, $\omega_3 = 21$ and $C = 24$.

When $m = 1$, $c = 24$, the strategy of the state is $R = \{1, 2, 3\}$. From (5), the best responses of the terrorist are

$$\begin{aligned}
 v^*(1) &= \arg \max \{16 \times 0, 10 \times 5, 21 \times 6\} = 3, \\
 v^*(2) &= \arg \max \{16 \times 5, 10 \times 0, 21 \times 8\} = 3, \\
 v^*(3) &= \arg \max \{16 \times 6, 10 \times 8, 21 \times 0\} = 1.
 \end{aligned}
 \tag{11}$$

From (6), the state equilibrium is

$$\begin{aligned}
 R_1^* &= \arg \min \{U_G(1, 3), U_G(2, 3), U_G(3, 1)\} \\
 &= \arg \min \{126, 168, 96\} = 3.
 \end{aligned}
 \tag{12}$$

The equilibrium solution of the game is $\{3, 1\}$, and the utility of the state is $U_{G1}^* = -96$.

When $m = 2$, $c = 12$, the strategy of the state is $R = \{(1, 2), (2, 3), (1, 3)\}$, and the best responses of the terrorist are

$$\begin{aligned}
 v^*(1, 2) &= \arg \max \{12 \times 0 + 4 \times 5, 10 \times 0, 12 \times 6 + 9 \times 8\} = 3, \\
 v^*(1, 3) &= \arg \max \{12 \times 0 + 4 \times 6, 10 \times 5, 12 \times 0 + 9 \times 6\} = 3, \\
 v^*(2, 3) &= \arg \max \{12 \times 5 + 4 \times 6, 10 \times 0, 12 \times 0 + 9 \times 8\} = 1.
 \end{aligned}
 \tag{13}$$

The state equilibrium is

$$\begin{aligned}
 R_2^* &= \arg \min \{U_G((1, 2), 3), U_G((1, 3), 3), U_G((2, 3), 1)\} \\
 &= \arg \min \{144, 54, 84\} = (1, 3).
 \end{aligned}
 \tag{14}$$

The equilibrium solution of the game is $\{(1, 3), 3\}$, and the utility of the state is $U_{G2}^* = -54$.

When $m = 3$, $c = 8$, the strategy of the state is $R = \{(1, 2, 3)\}$, and the state equilibrium is $R_3^* = (1, 2, 3)$. The best response of the terrorist is

$$\begin{aligned}
 v^*(1, 2, 3) &= \arg \max \{8 \times 0 + 8 \times 5, 8 \times 0 + 2 \\
 &\quad \times 5, 8 \times 0 + 8 \times 6 + 5 \times 8\} = 3.
 \end{aligned}
 \tag{15}$$

The equilibrium solution of the game is $\{(1, 2, 3), 3\}$, and the utility of the state is $U_{G3}^* = -88$.

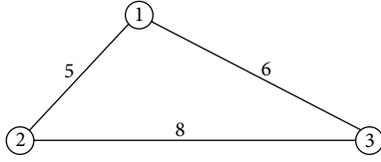


FIGURE 1: The network in the example.

From (10), the optimal m is $m^* = \arg \max_{m=1,2,3} \{U_{G1}^*, U_{G2}^*, U_{G3}^*\} = 2$. The Nash equilibrium of the game is $((12, (1, 3)), 3)$. The state will establish terror response facilities at node 1 and node 3, and the terrorist will attack node 3. The resource at each facility is 12, and the state will get a disutility of 54.

4. The Integer Programming

To solve the equilibrium efficiently, we formulate the game when the facility number is fixed as an integer program.

Define the decision variables as follows:

$$y_{ij} = \begin{cases} 1, & \text{if the city } v_i \text{ is assigned to facility } r_j \\ 0, & \text{otherwise} \end{cases} \quad \forall i, j = 1, 2, \dots, n,$$

$$x_j = \begin{cases} 1, & \text{if a facility is established at city } v_j \\ 0, & \text{otherwise} \end{cases} \quad \forall j = 1, 2, \dots, n, \quad (16)$$

$$z_i = \begin{cases} 1, & \text{if the terrorist attacks city } v_i \\ 0, & \text{otherwise} \end{cases} \quad \forall i = 1, 2, \dots, n.$$

Let Q be the terrorist utility in the equilibrium; then

$$\sum_{i=1}^n [(d[v_i, r_1] y_{i1} + \dots + d[v_i, r_{p-1}] y_{i,p-1}) \cdot c + d[v_i, r_p] y_{ip} \cdot (\omega_{v_i} - (p-1)c)] z_i \leq Q. \quad (17)$$

At the same time, the state will minimize his disutility; that is,

$$\sum_{i=1}^n [(d[v_i, r_1] y_{i1} + \dots + d[v_i, r_{p-1}] y_{i,p-1}) \cdot c + d[v_i, r_p] y_{ip} \cdot (\omega_{v_i} - (p-1)c)] z_i \geq Q. \quad (18)$$

Consequently, when the facility number is fixed, we could simplify (17) and (18) using (7).

The game could be formulated as follows:

$$\min Q \quad (19)$$

s.t.

$$\sum_{j=1}^n y_{ij} = p_i \quad \forall i \in I, \quad (20)$$

$$m \frac{\omega_{v_i}}{C} \leq p_i < m \frac{\omega_{v_i}}{C} + 1, \quad (21)$$

$$\sum_{j=1}^n x_j = m, \quad (22)$$

$$\sum_{i=1}^n z_i = 1, \quad (23)$$

$$y_{ij} \leq x_j, \quad \forall i, j = 1, \dots, n, \quad (24)$$

$$\sum_{k=1}^n y_{ij} d[v_i, v_k] + \sum_{l=1}^p (M - d[v_i, r_l]) x_l \leq pM, \quad (25)$$

$$\forall i, j = 1, 2, \dots, n,$$

$$\sum_{i=1}^n \sum_{j=1}^n \left(d[v_i, v_j] y_{ij} \frac{C}{m} \right) z_i = Q, \quad (26)$$

$$\sum_{j=1}^n d[v_i, v_j] y_{ij} \frac{C}{m} \leq Q, \quad \forall i \in I, \quad (27)$$

$$x_j \in \{0, 1\}, \quad z_i \in \{0, 1\}, \quad y_{ij} \in \{0, 1\}, \quad p_i \in N^+, \quad (28)$$

where M is a sufficiently large number, $M \geq p \cdot \max_{v_i, v_j \in V} d[v_i, v_j]$.

Equations (20) and (21) are used to ensure that each city is assigned to enough facilities while not wasting resources. Equation (22) guarantees that the facility number is m . In (23), we make sure that the terrorist will attack only one city. In (24), we make sure that each city is assigned to a node with a facility built at it. In (25), we verify that when the nearest facility could not meet the demand, the second nearest facility will be involved, and so on. In (26) and (27), we take the state and the terrorist objectives into account. Binary requirements are specified in (28).

Obviously, the program is a nonlinear programming because the formulation 10 has multiplications of variables. We could define a new variable u_{ij} to linearize it. Let

$$u_{ij} = y_{ij} z_i, \quad (29)$$

where

$$y_{ij} + z_i \leq 1 + u_{ij} \quad i, j = 1, \dots, n, \quad (30)$$

$$y_{ij} + z_i \geq 2u_{ij} \quad i, j = 1, \dots, n.$$

Then, (26) could be rewritten as

$$\sum_{i=1}^n \sum_{j=1}^n d[v_i, v_j] u_{ij} \frac{C}{m} = Q. \quad (31)$$

The linearized formulation of the game could be obtained by substituting (29)–(31) in (26).

TABLE 1: The state disutility under different scenarios.

C = 200	Facility number	1	2	3	4	5	6	7	8	9	10
	State disutility	12560	4770	5075.9	5045	4176	4009.3	3330	3702.2	3694.1	3472
	Facility number	11	12	13	14	15	16	17	18	19	
	State disutility	3336.1	3792.6	3497.3	3409.1	3742.6	3937.5	4060.4	4436.7	4899.3	
C = 750	Facility number	1	2	3	4	5	6	7	8	9	10
	State disutility	47100	17888	8629.3	6468.8	4455	3459	2402.4	2100	1864.8	1680
	Facility number	11	12	13	14	15	16	17	18	19	
	State disutility	2989.3	2743	2529.4	2348.8	2184.5	2053.1	1938.2	1823.2	1724.6	
C = 1300	Facility number	1	2	3	4	5	6	7	8	9	10
	State disutility	81640	31005	14957	10433	7202	5259.7	4164.2	3185	2583	1612
	Facility number	11	12	13	14	15	16	17	18	19	
	State disutility	1100.2	651.3	600.6	1161.9	1080.6	1454.4	1372.9	2489.2	2989.3	

TABLE 2: The 19 districts and their population sizes.

Node	District	Population size ($\times 10^4$)
1	Pudong	191.2
2	Huangpu	60.6
3	Luwan	31.2
4	Xuhui	89.2
5	Changning	61.1
6	Jing'an	31.0
7	Putuo	86.3
8	Zhabei	69.5
9	Hongkou	79.0
10	Yangpu	107.7
11	Baoshan	83.1
12	Minhang	88.6
13	Jiading	53.8
14	Jinshan	52.1
15	Songjiang	54.3
16	Qingpu	45.7
17	Nanhui	73.4
18	Fengxian	51.6
19	Chongming	69.7

5. Case Study

We solve the problem of terror response facility location under capacity constraints with a case study of districts in Shanghai, China. We assume that the resources needed are proportional to the population sizes of the districts. The 19 districts and their population sizes (10^4) are given in Table 2. The shortest distance matrix between the districts is given in Table 3. Without loss of generality, we considered three scenarios for resource capacity of the state, that is, insufficient, moderate, and abundant, which correspond to $C = 200, 750, 1300$, respectively. The equilibrium solutions were obtained by the linear programming in Section 4 using the optimization software Cplex 12.2. The software was run on an Intel Core2 Duo 2.0 GHz processor with 1.0 GB RAM under Windows XP.

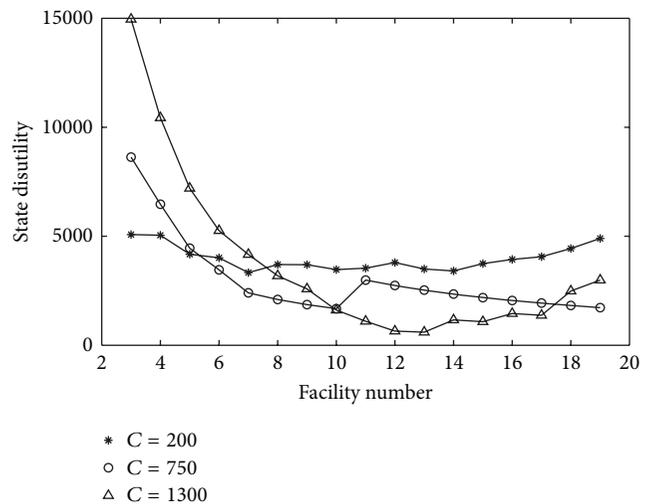


FIGURE 2: The relationship between facility number and the state disutility.

As stated above, different facility numbers could affect the state disutility, because the state could disperse the resources into more facilities to improve the coverage or could concentrate the resources to fewer facilities to protect the key areas. The state disutilities for different facility numbers are given in Table 1. From (10), the optimal facility number is 7, 10, and 13, respectively, when the total resources are 200, 750, and 1300, and the disutility is 3330, 1680, and 600.6. It shows that as the total resources increasing, the state could improve the coverage by establishing more facilities; at the same time, the resources at each facility will increase, that reduces the state disutility effectively.

In Figure 2, we depict the disutility of the state for different facility number under the three scenarios. Obviously, when the total resources are fixed, it is not a simple linear relationship between the state disutility and the facility number. For example, in the abundant scenario, when $m = 9$, there are 144.4 resources at each facility, the state disutility is almost the same when $m = 18$, in which there are 72.2 resources at each facility. However, when $m = 13$, there are 100 resources

TABLE 3: The shortest distance matrix between the districts.

Districts	Pudong	Huangpu	Luwan	Xuhui	Changning	Jingan	Putuo	Zhabei	Hongkou	Yangpu	Baoshan	Minhang	Jiading	Jinshan	Songjiang	Qingpu	Nanhui	Fengxian	Chongming
Pudong	0	6.3	7.6	12.4	12.4	10.3	16.4	9.8	6.5	6.9	24.3	22.7	36.0	66.9	41.6	43.1	35.8	40.6	67.3
Huangpu	6.3	0	2.3	7.0	6.1	3.9	10.1	3.4	4.7	5.7	21.2	17.3	29.7	65.1	36.3	37.1	42.2	39.7	63.0
Luwan	7.6	2.3	0	5.0	5.2	3.3	9.3	3.7	7.1	8.1	23.4	15.3	29.5	63.1	34.2	35.7	42.7	37.4	63.4
Xuhui	12.4	7.0	5.0	0	4.0	4.7	8.8	7.8	12.0	13.3	28.0	10.2	29.9	57.3	30.0	32.1	43.6	34.8	67.8
Changning	12.4	6.1	5.2	4.0	0	2.3	5.0	4.8	10.0	11.3	25.1	14.3	26.1	61.4	32.2	31.6	46.5	38.9	64.7
Jingan	10.3	3.9	3.3	4.7	2.3	0	6.1	2.6	7.4	8.5	22.7	15.0	27.3	62.8	33.9	33.8	44.8	38.8	62.7
Putuo	16.4	10.1	9.3	8.8	5.0	6.1	0	7.1	12.3	14.2	24.3	17.7	21.3	63.5	34.1	33.2	50.8	41.7	62.8
Zhabei	9.8	3.4	3.7	7.8	4.8	2.6	7.1	0	5.2	7.0	20.6	18.0	26.0	65.8	37.0	36.6	45.4	40.9	60.6
Hongkou	6.5	4.7	7.1	12.0	10.0	7.4	12.3	5.2	0	2.4	17.9	23.3	29.8	70.1	41.3	41.5	42.4	44.5	60.8
Yangpu	6.9	5.7	8.1	13.3	11.3	8.5	14.2	7.0	2.4	0	18.9	23.6	31.9	70.8	42.5	42.8	40.3	44.5	61.9
Baoshan	24.3	21.2	23.4	28.0	25.1	22.7	24.3	20.6	17.9	18.9	0	38.3	22.9	86.2	57.2	53.3	54.8	60.8	43.8
Minhang	22.7	17.3	15.3	10.2	14.3	15.0	17.7	18.0	22.3	23.6	38.3	0	34.6	47.7	19.7	29.1	44.6	24.7	78.1
Jiading	36.0	29.7	29.5	29.9	26.1	27.3	21.3	26.0	29.8	31.9	22.9	34.6	0	78.0	43.1	32.9	71.6	58.1	54.3
Jinshan	66.9	65.1	63.1	57.3	61.4	62.8	63.5	65.8	70.1	70.8	86.2	47.7	78.0	0	38.8	57.0	60.5	27.7	125.7
Songjiang	41.6	36.3	34.2	30.0	32.2	33.9	34.1	37.0	41.3	42.5	57.2	19.7	43.1	38.8	0	19.6	59.4	34.7	96.0
Qingpu	43.1	37.1	35.7	32.1	31.6	33.8	33.2	36.6	41.5	42.8	53.3	29.1	32.9	57.0	19.6	0	73.1	52.6	86.3
Nanhui	35.8	42.2	42.7	43.6	46.5	44.8	50.8	45.4	42.4	40.3	54.8	44.6	71.6	60.5	59.4	73.1	0	34.5	99.2
Fengxian	40.6	39.7	37.4	34.8	38.9	38.8	41.7	40.9	44.5	44.5	60.8	24.7	58.1	27.7	34.7	52.6	34.5	0	100.7
Chongming	67.3	63.0	63.4	67.8	64.7	62.7	62.8	60.6	60.8	61.9	43.8	78.1	54.3	125.7	96.0	86.3	99.2	100.7	0

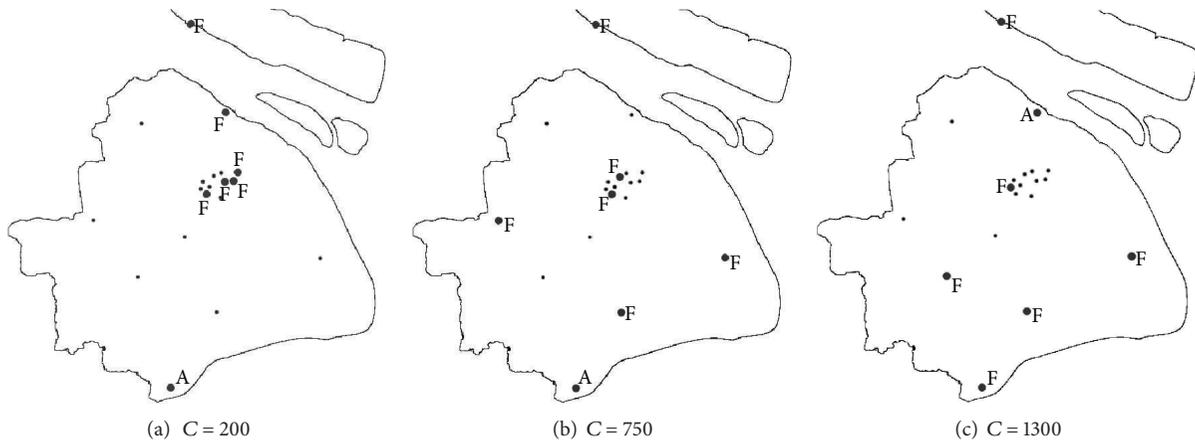


FIGURE 3: The facility location “F” and the district attacked “A” in equilibrium when $m = 6$ under different scenarios.

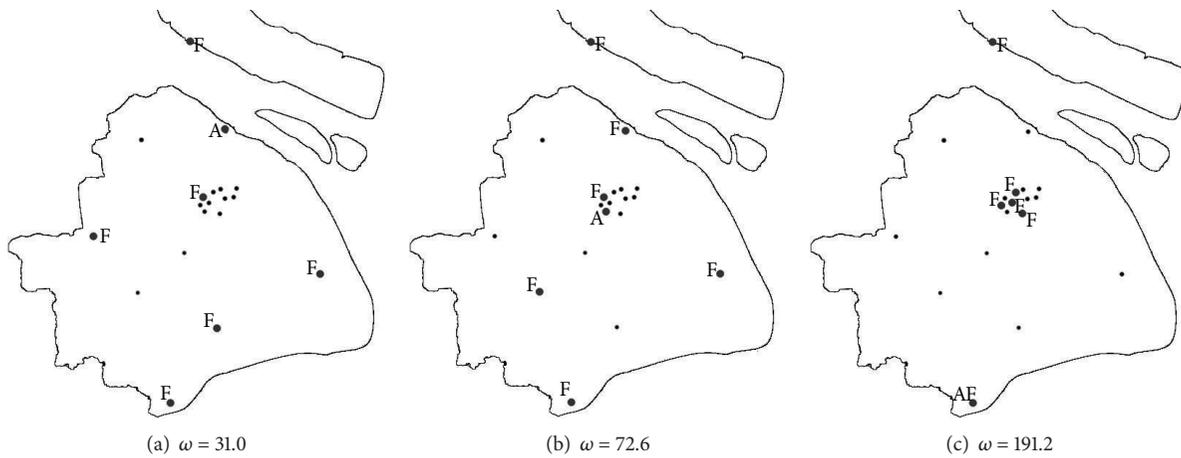


FIGURE 4: The facility location “F” and the district attacked “A” in equilibrium when $C = 200, m = 6$.

at each facility, and the state disutility is much less than the two cases above. It implies that, distributing the resources in limited facilities could protect key areas but will reduce the coverage, while distributing the resources in more facilities will increase the shipment cost although the coverage is enhanced. A balance should be achieved.

For fixed facility number, when the resources at each facility increase, the equilibrium solutions will vary, as Figure 3 shows. When the total resources are insufficient, once a district is attacked, more than one facility has to be involved. Then, the concentration mode of the resources could ensure the quick response to districts with more populations. The terrorist will attack Jinshan, which has less populations and is far away from all the terror response facilities, and the state will establish facilities at Pudong, Luwan, Xuhui, Yangpu, Baoshan, and Chongming. The state loss is mainly from the shipment delay of the resources. When the total resources are moderate, a single facility could satisfy the demand of some districts, and the facility distribution tends to be decentralized to improve the coverage. The terrorist will still attack Jinshan. When the total resources are abundant, a single facility could meet most demands, and the facility

distribution will be more dispersed to improve the coverage further. In this case, the terrorist will attack Baoshan, which has more populations, and the state will establish facilities at Putuo, Jinshan, Songjiang, Nanhui, Fengxian, and Chongming. The state loss comes more from the affected populations.

We consider a case that all districts have identical population, that is, $\omega_i = \omega$, for all $i \in V$, to check the sensitivity of the equilibrium solution to the variations of the node weights. Let the weights be 31.0, 72.6, and 191.2, respectively, which correspond to the lowest, highest, and average of the populations of all districts. The state has a total of 200 resources, dispersing in 6 facilities. As we can see from Figure 4, the equilibrium solutions are not related to population quantity but with the population distribution, namely, the network structure. Also, the results are much different from the cases when the populations vary across the districts. When $\omega = 31.0$, there are relatively many resources at each facility, and the distribution mode of the facilities tends to be decentralized; when $\omega = 72.6$, 3 facilities have to be involved in the response, and the terrorist attacks the district which has more populations and faraway from

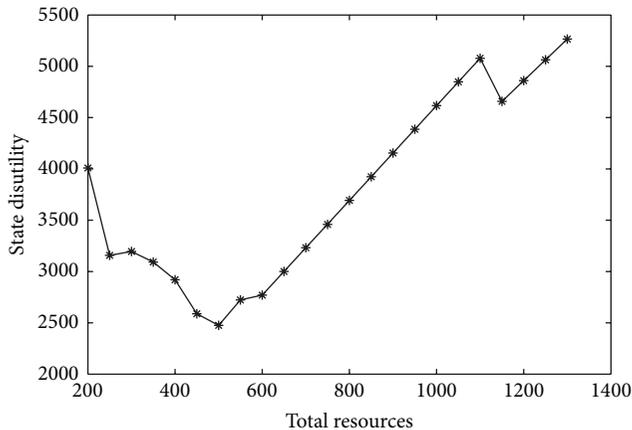


FIGURE 5: The relationship between total resources and state disutility when $m = 6$.

all facilities; when $\omega = 191.2$, all the 6 facilities must be involved. The problem for the state is equivalent to the p -median problem, and the terrorist will attack Jinshan.

In practical, the state could determine the most cost-effective total resource according to the tradeoff curve between the total resources and the state disutility. For example, when $m = 6$, the tradeoff curve is shown in Figure 5. Obviously, the state should prepare 500 resources in the sense of cost efficiency.

6. Conclusions

In this paper, we introduced the capacity constraints into the game between the state and the terrorist. The government locates terror response facilities under capacity constraints first. The terrorist observes the distribution of the facilities then chooses the weakest target to attack. The government, who knows that its move in the game is the common knowledge, must place itself in its opponents' position before deciding the appropriate strategic response. An integer programming was proposed to solve the sequential game when the facility number is fixed. A case study was provided to analyze the strategic interaction of both sides. The equilibrium results showed that resource capacity did affect the action of the state and the terrorist. Centralized mode or dispersed mode for facility distribution depended on the total resources the state has.

The proposed approach can be of interest for modelling and solving more general response facility location problems under deliberate attack. The assumption that all the resources will be shipped as a whole may be limited to a certain extent. Besides, in real life, two or more bombings or suicide terrorist attacks usually occur simultaneously or sequentially, for example, the tragedy in Madrid. Moreover, the assumption that a strike on one city has no effect on other cities may be not always valid when biological or chemical attacks take place. This is beyond the scope of this paper and is the subject of ongoing research.

Acknowledgments

This research was supported by Major Program of National Natural Science Foundation of China, Grant nos. 91024023 and 91224003, the National Natural Science Foundation of China, Grant nos. 70871093, 61005090, and 71371142, the Fundamental Research Funds for the Central Universities, and the Research Fund of State Key Lab of Management and Control for Complex systems, China. These supports are gratefully acknowledged.

References

- [1] H. Jia, F. Ordóñez, and M. Dessouky, "A modeling framework for facility location of medical services for large-scale emergencies," *IIE Transactions*, vol. 39, no. 1, pp. 41–55, 2007.
- [2] O. Berman and A. Gaviou, "Location of terror response facilities: a game between state and terrorist," *European Journal of Operational Research*, vol. 177, no. 2, pp. 1113–1133, 2006.
- [3] O. Berman, Z. Drezner, and D. Krass, "Cooperative cover location problems: the planar case," *IIE Transactions*, vol. 42, no. 3, pp. 232–246, 2010.
- [4] M. S. Daskin, "The maximal expected covering location model: formulation, properties and heuristic solution," *Transportation Science*, vol. 17, no. 1, pp. 48–70, 1983.
- [5] H. Pirkul and D. Schilling, "The capacitated maximal covering location problem with capacities on total workload," *Management Science*, vol. 37, no. 2, pp. 233–248, 1991.
- [6] J. Goldberg and L. Paz, "Locating emergency vehicle bases when service time depends on call location," *Transportation Science*, vol. 25, no. 4, pp. 264–280, 1991.
- [7] A. Klose and A. Drexl, "Facility location models for distribution system design," *European Journal of Operational Research*, vol. 162, no. 1, pp. 4–29, 2005.
- [8] L. V. Snyder and M. S. Daskin, "Stochastic p -robust location problems," *IIE Transactions*, vol. 38, no. 11, pp. 971–985, 2006.
- [9] D. Sarewitz, R. Pielke Jr., and M. Keykhah, "Vulnerability and risk: some thoughts from a political and policy perspective," *Risk Analysis*, vol. 23, no. 4, pp. 805–810, 2003.
- [10] J. Zhuang and V. M. Bier, "Balancing terrorism and natural disasters-defensive strategy with endogenous attacker effort," *Operations Research*, vol. 55, no. 5, pp. 976–991, 2007.
- [11] B. Golany, E. H. Kaplan, A. Marmur, and U. G. Rothblum, "Nature plays with dice-terrorists do not: allocating resources to counter strategic versus probabilistic risks," *European Journal of Operational Research*, vol. 192, no. 1, pp. 198–208, 2009.
- [12] L. V. Snyder, "Facility location under uncertainty: a review," *IIE Transactions*, vol. 38, no. 7, pp. 547–564, 2006.
- [13] D. Lakdawalla and G. Zanjani, "Insurance, self-protection, and the economics of terrorism," *Journal of Public Economics*, vol. 89, no. 9-10, pp. 1891–1905, 2005.
- [14] T. Sandler and W. Enders, "An economic perspective on transnational terrorism," *European Journal of Political Economy*, vol. 20, no. 2, pp. 301–316, 2004.
- [15] B. S. Frey and S. Luechinger, "How to fight terrorism: alternatives to deterrence," *Defence and Peace Economics*, vol. 14, no. 4, pp. 237–249, 2003.
- [16] T. Sandler and K. Siqueira, "Global terrorism: deterrence versus pre-emption," *Canadian Journal of Economics*, vol. 39, no. 4, pp. 1370–1387, 2006.

- [17] G. Daniel, M. Arce, and T. Sandler, "Counterterrorism: a game-theoretic analysis," *Journal of Conflict Resolution*, vol. 49, no. 2, pp. 183–200, 2005.
- [18] W. Enders and T. Sandler, "The effectiveness of anti-terrorism policies: Vector—autoregression—intervention analysis," *American Political Science Review*, vol. 87, no. 4, pp. 829–844, 1993.
- [19] W. Enders and T. Sandler, "What do we know about the substitution effect in transnational terrorism?" in *Research on Terrorism: Trends, Achievements and Failures*, A. Silke, Ed., pp. 119–137, Frank Cass, London, UK, 2004.
- [20] D. Ghose, J. L. Speyer, and J. S. Shamma, "A game theoretical multiple resource interaction approach to resource allocation in an air campaign," *Annals of Operations Research*, vol. 109, no. 1–4, pp. 15–40, 2002.
- [21] L. Zhang, "The efficiency and fairness of a fixed budget resource allocation game," in *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP '05)*, pp. 485–496, Springer, Lisbon, Portugal, July 2005.
- [22] T. Sandler and H. E. Lapan, "The calculus of dissent: an analysis of terrorists' choice of targets," *Synthese*, vol. 76, no. 2, pp. 245–261, 1988.
- [23] R. Powell, "Defending against terrorist attacks with limited resources," *American Political Science Review*, vol. 101, no. 3, pp. 527–541, 2007.
- [24] R. Powell, "Allocating defensive resources with private information about vulnerability," *American Political Science Review*, vol. 101, no. 4, pp. 799–809, 2007.
- [25] V. Bier, S. Oliveros, and L. Samuelson, "Choosing what to protect: strategic defensive allocation against an unknown attacker," *Journal of Public Economic Theory*, vol. 9, no. 4, pp. 563–587, 2007.
- [26] S. Farrow, "The economics of homeland security expenditures: foundational expected cost-effectiveness approaches," *Contemporary Economic Policy*, vol. 25, no. 1, pp. 14–26, 2007.
- [27] R. Levitan and M. Shubik, "Price duopoly and capacity constraints," *International Economic Review*, vol. 13, no. 1, pp. 111–122, 1972.
- [28] P. Dasgupta and E. Maskin, "The existence of equilibrium in discontinuous economic games—I: theory," *The Review of Economic Studies*, vol. 53, no. 1, pp. 1–26, 1986.
- [29] C. Davidson and R. Deneckere, "Long-run competition in capacity, short-run competition in price, and the cournot model," *The Rand Journal of Economics*, vol. 17, no. 3, pp. 404–415, 1986.
- [30] O. Berman, A. Gavious, and R. Huang, "Location of response facilities: a simultaneous game between state and terrorist," *International Journal of Operational Research*, vol. 10, no. 1, pp. 102–120, 2011.

Research Article

Naive Bayes-Guided Bat Algorithm for Feature Selection

Ahmed Majid Taha,¹ Aida Mustapha,² and Soong-Der Chen³

¹ College of Graduate Studies, Universiti Tenaga Nasional, 43000 Kajang, Selangor, Malaysia

² Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia

³ College of Information Technology, Universiti Tenaga Nasional, 43000 Kajang, Selangor, Malaysia

Correspondence should be addressed to Ahmed Majid Taha; ahmed.majid.taha@gmail.com

Received 30 September 2013; Accepted 14 November 2013

Academic Editors: Z. Cui and X. Yang

Copyright © 2013 Ahmed Majid Taha et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

When the amount of data and information is said to double in every 20 months or so, feature selection has become highly important and beneficial. Further improvements in feature selection will positively affect a wide array of applications in fields such as pattern recognition, machine learning, or signal processing. Bio-inspired method called Bat Algorithm hybridized with a Naive Bayes classifier has been presented in this work. The performance of the proposed feature selection algorithm was investigated using twelve benchmark datasets from different domains and was compared to three other well-known feature selection algorithms. Discussion focused on four perspectives: number of features, classification accuracy, stability, and feature generalization. The results showed that BANB significantly outperformed other algorithms in selecting lower number of features, hence removing irrelevant, redundant, or noisy features while maintaining the classification accuracy. BANB is also proven to be more stable than other methods and is capable of producing more general feature subsets.

1. Introduction

The motivations to perform feature selection in a classification experiment are two-fold. The first is to enhance the classifier performance by selecting only useful features and removing irrelevant, redundant, or noisy features. The second is to reduce the number of features when classification algorithms could not scale up to the size of feature set either in time or space. In general, feature selection consists of two essential steps, which are searching for desired subset using some search strategies and evaluating the subset produced. A search strategy in searching the feature subset could be exhaustive or approximate. While exhaustive search strategy evaluates all probabilities of the feature subset, approximate search strategy only generates high quality solutions with no guarantee of finding a global optimal solution [1].

One of the most prominent algorithms in exhaustive search is branch and bound method [2]. Exhaustive search guarantees optimal solution but this method is not practical for even a medium-sized dataset as finding the optimal subset of features is an NP-hard problem [3]. For N number of features, the number of possible solutions will be exponential

to 2^N . Since exhaustive search is not practical, research effort and focus on search strategies have since shifted to metaheuristic algorithms, which are considered as a subclass of approximate methods. The literature has shown that the metaheuristic algorithms are capable of handling large-size problem instances with satisfactory solutions within a reasonable time [4–7].

After searching for feature subset, each candidate from the resulting subset generated needs to be evaluated based on some predetermined assessment criteria. There are three categories of feature subset evaluations depending on how the searching strategy is being associated with the classification model, whether as filter, wrapper, or embedded methods. These three categories will be explained in more detail in the next section.

Nonetheless, the main challenge in feature selection is to select the minimal subset of features with modicum or no loss of classification accuracy. While the literature has shown numerous developments in achieving this [8–10], the basis of comparison is rather limited in terms of number of features, classification accuracy, stability, or feature generalization in

isolation. Generalization of the produced features is important to investigate their effect on the performance of different classifiers.

In view of this, the objectives of this paper are as follows: first to design a new hybrid algorithm that exploits a Naive Bayes algorithm to guide a Bat Algorithm, second to evaluate the performance of the proposed hybrid algorithm against other well-known feature selection algorithms, and third to test the effect of the resulting features in terms of generalization using three different classifiers. The remainder of this paper is organized as follows. Section 2 reviews the related works on searching and evaluating algorithms in feature selection. Section 3 details out the principles of Naive Bayes algorithms, Section 4 presents mechanics of the Bat Algorithm, and Section 5 introduces the proposed Naive Bayes-guided Bat Algorithm for feature selection. Next, Section 6 describes the experimental settings, Section 7 discusses implications of the results, and, finally, Section 8 concludes with some recommendations for future work.

2. Related Work

The application of metaheuristics algorithms in searching the feature subset has shown high effectiveness as well as efficiency to solve complex and large problems in feature selection. In general, there are two categories of metaheuristic search algorithms: single solution-based metaheuristics (SBM) that manipulate and transform a single solution during the search and population-based metaheuristics (PBM) where a whole population of solutions is evolved. The simplest and oldest SBM method used in feature selection is Hill Climbing (HC) algorithm [1, 11]. This algorithm starts with a random initial solution and swaps the current solution with a neighboring solution in the following iteration in order to improve the quality of solution. Searching will stop only when all the neighboring candidate subsets are poorer than the current solution, which means that the algorithm will be most probably trapped in local optimum [4].

In order to overcome this problem, Simulated Annealing (SA) is proposed [10]. SA accepts the worse moves that commensurate to the parameter determined at the initial stage, called the temperature, which is inversely proportional to the change of the fitness function. In more recent work, a modified SA algorithm called the Great Deluge Algorithm (GDA) is proposed [12] to provide a deterministic acceptance function of the neighboring solutions. Tabu Search (TS) also accepts nonimproving solutions to escape from local optima. TS stores information related to the search process, which is a list of all previous solutions or moves in what is termed as Tabu list [13, 14]. Nonetheless, SBM algorithms such as Hill Climbing and Simulated Annealing suffer from two major disadvantages. First, they often converge towards local optima and second they can be very sensitive to the initial solution [1].

The PBM methods have been equally explored in feature selection. Different from SBM, PBM represents an iterative improvement in a population of solutions that works as follows. Firstly, the population is initialized. Then, a new

population of solutions is generated. Next, the new population is integrated into the existing one by using some selection procedures. The search process is terminated when a certain criterion is satisfied. The most prominent and oldest population-based solution used in feature selection is Genetic Algorithm (GA) [5, 15, 16]. The major roles in GA are the crossover and mutation operations used to couple solutions and to arbitrarily adjust the individual content, to boost diversity aiming to decrease the risk of sticking in local optima.

Another PBM algorithm is the Ant Colony Optimization (ACO), which takes form as a multiagent system, whereby the building unit of this system represents virtual ants as inspired by the behavior of real ants. In nature, a chemical trace called pheromone is left on the ground and is used to guide a group of ants heading for the target point since ants are not able to see very well [6, 17, 18]. Another nature-inspired algorithm is the Particle Swarm Optimization (PSO) algorithm that simulates the social behavior of natural creatures such as bird flocking and fish schooling to discover a place with adequate food [7, 19]. Scatter Search (SS) is another PBM method that recombines solutions elected from a reference set to generate other solutions by building an initial population satisfactory to the criteria of quality and diversity [20].

The next step in feature selection is evaluating the feature subset produced. The evaluation methods can be broadly classified into three categories. First, the filter approach or independent approach evaluates candidate solutions by depending on intrinsic characteristics of the features themselves, without considering any mining algorithm. Filter approach includes several types such as distance [21], information [22], dependency [23], or consistency [24]. Second, the wrapper approach or dependent approach requires one predetermined learning model and selects features with the purpose of improving the generalization performance of that particular learning model [13]. Although the wrapper approach is known to outperform the filter approach with regard to prediction accuracy [25], the method is time-consuming. Third, the embedded approach in feature evaluation attempts to capitalize on advantages of both approaches by implementing the diverse evaluation criteria in different search phases [26]. By integrating the two approaches at different phases, the embedded approach is capable to achieve accuracy of a wrapper approach at the speed of a filter approach. Choosing an evaluation method for particular search method is a critical mission because the interaction between the evaluation method and the search strategy will affect the overall quality of solution.

3. Naive Bayes Algorithm

Naive Bayes (NB) algorithm is one of the most effective and efficient inductive learning algorithms for data mining along with machine learning. This algorithm belongs to the wrapper approach. NB is considered a simple classifier based on the classical statistical theory "Bayes theorem." The Bayesian algorithm is branded "naïve" because it is founded on Bayes Rule, which has a strong supposition that the features are

conditionally independent from each other with regard to the class [27]. In the literature, the NB algorithm has proven its effectiveness in various domains such as text classification [28], improving search engine quality [29], image processing [30, 31], fault prediction [32], and medical diagnoses [8].

Naive Bayes classifier works as follows: let X be a vector of random variables denoting the observed attribute values in the training set $X = [x_1, x_2, \dots, x_n]$ to certain class label c in the training set. The probability of each class given the vector of observed values for the predictive attributes can be computed using the following formula:

$$P(Y_j | X) = \frac{P(Y_j)P(X | Y_j)}{\sum_{i=1}^c P(Y_i)P(X | Y_i)}, \quad j = 1, \dots, c, \quad (1)$$

where $P(Y_i)$ is the prior probability of class Y_i and $P(Y_j | X)$ is the class conditional probability density functions. Basically put, the conditional independence assumption assumes that each variable in the dataset is conditionally independent of the other. This is simple to compute for test cases and to estimate from training data as follows:

$$P(X | Y_j) = \prod_{i=1}^n P(X_i | Y_j), \quad j = 1, \dots, c, \quad (2)$$

where X_i is the value of the i th attribute in X and n is the number of attributes. Let k be the number of classes, and c_i is the i th class; the probability distribution over the set of features is calculated using the following equation:

$$P(x) = \prod_{i=1}^k P(c_i) p(X | c_i). \quad (3)$$

Effectiveness of Naive Bayes algorithm in classification and learning is attributed to several characteristics such as the following [27].

- (i) High computational efficiency as compared to other wrapper methods because it is inexpensive since it is considered linear time complexity classifier.
- (ii) Low variance due to less searching.
- (iii) Incremental learning because NB functions work from approximation of low-order probabilities that are deduced from the training data. Hence, these can be quickly updated as new training data are obtained.
- (iv) High capability to handle noise in the dataset.
- (v) High capability to handle missing values in the dataset.

Furthermore, NB implementation has no required adjusting parameters or domain knowledge. The major drawback of NB only lies in the assumption of features independence [33]. Despite this, NB often delivers competitive classification accuracy and is widely applied in practice especially as benchmark results. Good survey on the variety of adaptations to NB in the literature can be found in [33].

4. Bat Algorithm

The idea of the Bat Algorithm (BA) is to mimic the behaviors of bats when catching their prey. BA was first presented in [34] and it was found to outperform Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) in evaluation using benchmark functions. BA has also been successfully applied to tough optimization problems such as motor wheel optimization problem [35], clustering problem [36], global engineering optimization, and constrained optimization tasks [37–40]. Very recently, two versions of bat-inspired algorithms have been proposed for feature selection [41, 42]. The implementation of BA is more complicated than many other meta-heuristic algorithms [43] because each agent (bat) is assigned a set of interacting parameters such as position, velocity, pulse rate, loudness, and frequencies. This interaction affects the quality of a solution and the time needed to obtain such solution.

The principle of bat algorithm is as follows. A swarm of bats is assumed to fly randomly with velocity V_i at position X_i with a fixed frequency f , varying wavelength λ , and loudness A_0 to search for a prey. They have the capability to adjust the wavelength of their emitted pulses and regulate the rate of pulse emission $r \in [0, 1]$, which is important to determine their closeness of the target. Although the loudness can be different in many ways, the loudness differs from a large (positive) A_0 to a minimum constant value A_{\min} . The frequency f is in the range $[f_{\min}, f_{\max}]$ that corresponds to a range of wavelengths $[\lambda_{\min}, \lambda_{\max}]$. For example, a frequency range of [20 kHz, 500 kHz] corresponds to a range of wavelengths from 0.7 mm to 17 mm.

5. Proposed Naive Bayes-Guided Bat Algorithm

5.1. Frequency. Frequency in the proposed algorithm is represented as a real number as defined in (4). The choice of minimum and maximum frequency depends on the application domain, where β is a random number range between 0 and 1. Frequency also affects the velocity as shown in (5). Consider the following:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta. \quad (4)$$

5.2. Velocity. The velocity of each bat is represented as a positive integer number. Velocity suggests the number of bat attributes that should change at a certain moment of time. The bats communicate with each other through the global best solution and move towards the global best position (solution). The following equation shows the formula for velocity:

$$v_i^t = v_i^{t-1} + (x_* - x_i^t) f_i, \quad (5)$$

where $(x_* - x_i^t)$ refers to the difference between the length of global best bat and the length of the i th bat. When the difference is positive, this means that the global best bat has more features than those of the i th bat. When the result is summed with the previous velocity, it will accelerate the i th

bat towards the global best bat. If the difference is negative, this means that the i th bat has more features than those of the global best bat. Therefore, when the output is summed with the previous velocity, it will decrease the velocity of i th bat and help to attract it closer to global best bat. In the proposed Bat Algorithm-Naive Bayes (BANB) algorithm, the maximum velocity was setting (V_{\max}) equal to $(1/3) * N$, where N is the number of features. In the proposed BANB, (2) is used to adjust the velocity during each iteration; therefore, the proposed algorithm is adaptive for feature selection problem in order to mimic the original algorithm behavior. Velocity representation is also one major difference between BANB and the Binary Bat Algorithm (BBA) [42]. In BBA, the velocity is calculated for each single feature; hence, the algorithm is more time-consuming and departing from the original algorithm attitude. On the contrary, the velocity in the proposed BANB is calculated once for the entire solution; hence, the velocity amount determines the piece of change.

5.3. Position Adjustment. In the proposed algorithm, each bat position is formulated as a binary string of length N , where N is the total number of features. Each feature is represented by bit, where “1” means that the corresponding feature is selected and the “0” means that it is not selected. The positions are categorized into two groups according to the bit difference between the i th bat and the global best bat in order to align exploitation and exploration during searching.

The bat’s position is adjusted depending on one of the following conditions. In the case where the velocity of i th bat is lower or equal to the number of different bits, i th bat will copy some features from global best bat, thus moving towards global best bat, while still exploring new search space. In the case where the velocities of i th bat are higher than the velocity of global best bat, then the i th bat will import all features from the global best bat to be the same as the global best bat with a few different bits to facilitate further exploitation. The following equation shows the position adjustment, where x is bat position, and v is the velocity of the i th bat at time t :

$$x_i^t = x_i^{t-1} + v_i^t. \quad (6)$$

5.4. Loudness. Loudness A_i in the proposed algorithm is represented as the change in number of features at certain time during local search around the global best bat, as well as local search around the i th bat. The formula for loudness is shown in (7), where A_i^t is the average loudness of all the bats at certain iteration and $\varepsilon \in [-1, 1]$. The value for sound loudness (A) ranges between the maximum loudness and minimum loudness. Consider the following:

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A^t. \quad (7)$$

Generally, the loudness value will decrease when the bat starts approaching the best solution. The following equation shows that the amount of decrease is determined by α :

$$A_i^{t+1} = \alpha A_i^t. \quad (8)$$

The value for sound loudness also plays an important role in obtaining good quality solutions within a reasonable amount of time. The choice of the maximum and minimum loudness depends on the domain of application and also the size of the dataset. In the proposed BANB algorithm, the maximum loudness has been determined empirically as $(1/5)*N$, where N is number of features. Value for maximum loudness is dynamic depending on number of features in certain dataset. For example, when $A_{\max} = 3$ and $A_{\min} = 1$, the bat begins to reduce the number of features from 3 features to 2 features and the value then becomes a single feature when it gets closer to the target.

5.5. Pulse Rate. Pulse rate r_i has the role to decide whether a local search around the global best bat solution should be skipped or otherwise. Higher pulse rate will reduce the probability of conducting a local search around the global best and vice versa. Therefore, when the bat approaches the best solution, pulse rate value will increase and subsequently reduce the chances to conduct a local search around the global best. The amount of increase is determined by γ as defined in the following:

$$r_i^{t+1} = r_i^0 [1 - \exp(-\gamma t)]. \quad (9)$$

5.6. Fitness Function. Each candidate solution is using a fitness function defined in (10), where $P(Y_j | X)$ is the classification accuracy, TF is the total number of all features, and SF is the number of selected features. δ and φ are two parameters corresponding to the weight of classification accuracy and subset length, where $\delta \in [0, 1]$ and $\varphi = 1 - \delta$. From (10), we can see that the importance of classification accuracy and subset size is weighted differently. Generally, classification accuracy is given more weight than the size of the subset. In this experiment, the two parameters have been set as follows: $\delta = 0.9$, $\varphi = 0.1$. Consider the following:

$$\text{Sol}_A = \delta \cdot P(Y_j | X) + \varphi \cdot \frac{\text{TF} - \text{SF}}{\text{TF}}. \quad (10)$$

The complete algorithm for the proposed hybrid BA guided by Naive Bayes classifier (BANB) is shown in Algorithm 1.

6. Experiments and Results

The objective of the experiments is to evaluate the performance of the proposed Naive Bayes-guided Bat Algorithm (BANB) in terms of number of features selected and the classification accuracy achieved. To achieve this objective, we compared the number of features and classification accuracies of BANB with several well-known algorithms, which are Genetic Algorithms (GA) [44], Particle Swarm Optimization (PSO) [45], and Geometric Particle Swarm Optimization (GPSO) [46]. Similar to the proposed BANB, we also used Naive Bayes classifier for all comparative algorithms as the attribute evaluator. However, the parameters for the algorithms had the same settings as those used by the original authors. For the proposed algorithm, the parameters were set

to the following values: population size = 25 and decrease sound loudness and increase pulse rate both are set to 0.6. The initial value of pulse rate is equal to 0.2. The proposed BANB algorithm and other algorithms were run for 20 times with different initial solutions. Following [4, 17], all the algorithms were terminated after 250 iterations.

6.1. Description of Dataset. For the experiments, twelve datasets were considered to cover both cases of binary and multiclass data. Three of the datasets, namely, M-of-N, Exactly, and Exactly2, were sourced from [47]. M-of-N is an artificial binary class since the decision attribute consisting of two class labels and the dataset were generated from a uniform distribution to create the artificial domain. Exactly and Exactly2 are artificial binary classification datasets, generated based on x-of-y concepts, which are not linearly separable and are known to be difficult for many classes of learning algorithms [47]. The remaining datasets were taken from the UCI data repository [48]. The datasets are Vote, Credit, LED, Derm, Derm2, Lung, WQ, Heart, and Mushroom. Vote is widely used as a binary classification dataset in the literature. The dataset represents votes for each of the U.S. House of Representatives congressmen with the class label democrat and republican. Credit dataset is a binary classification data that is concerned with credit card applications. LED dataset in display domain is a multiclass classification data as the class label includes ten possible values in which the first seven features determine the class label of a pattern, whilst the rest of the 17 features are irrelevant.

Derm and Derm2 represent real data in dermatology concerning differential diagnosis of erythematous diseases. The class labels contain six values, which refer to six different diseases. Lung dataset is the pathological types of Lung cancer that aims to demonstrate the power of the optimal discriminant plane even in ill-posed settings [49]. WQ is a multiclass label dataset that originated from the daily measures of sensors in an urban waste water treatment plant. The idea is to categorize the operational state of the plant with the purpose of predicting faults out of the state variables of the plant at each of the phases in the water treatment procedure. Heart is a binary class data that contains 76 attributes although all the published experiments reference to using only 14 of the original attributes. This data has been used to predict heart diseases, whereby the class label of zero and one refers to the absence or existence of heart disease in the patient. Finally, Mushroom is a binary class dataset that includes characterization of hypothetical samples identical to 23 types of gilled mushrooms in the *Agaricus* and *Lepiota* family. Table 1 shows the characteristics of the datasets.

6.2. Results for Feature Selection Experiment. In this experiment, we compared the proposed BANB against GA [44], PSO [45], and GPSO [46] in terms of the number of features selected from the original dataset. Table 2 provides the comparison results. The number of features obtained from the comparative algorithms in Table 2, and the best results are highlighted in bold. Then the results are statistically tested using two tests, Kolmogorov-Smirnov and Levene test

```

(1) Initialize parameters:  $A, A_{\min}, A_{\max}, r, f_{\min}, f_{\max}, P_{\max}, I_{\max}, V_{\max}, V_{\min}, \Phi, \delta, \gamma, \alpha$ 
(2) Generate a swarm with  $P_{\max}$  bats
(3) Calculate cost function for all bats
(4) Find the current best bat ( $x_*$ )
(5) While stop condition not met Do
(6)   For  $i = 1$  to  $P_{\max}$  Do
(7)     Frequency  $f_i = f_{\min} + (f_{\max} - f_{\min})\beta$ 
(8)     Velocity  $v_i^t = v_i^{t-1} + (x_i^t - x_i)f_i$ 
(9)     If ( $V_i > V_{\max}$ ) Then
(10)       ( $V_i = V_{\max}$ )
(11)     End-If
(12)     If ( $V_i < V_{\min}$ ) Then
(13)       ( $V_i = V_{\min}$ )
(14)     End-If
(15)     Locations  $x_i^t = x_i^{t-1} + v_i^t$ 
(16)     If (Rand >  $r_i$ ) Then
(17)       calculate  $\epsilon A^t$ 
(18)       If ( $\epsilon A^t > A_{\max}$ ) Then
(19)         ( $\epsilon A^t = A_{\max}$ )
(20)       End-If
(21)       If ( $\epsilon A^t < A_{\min}$ ) Then
(22)         ( $\epsilon A^t = A_{\min}$ )
(23)       End-If
(24)       Generate a local solution around the best solution ( $x_*$ ) [ $x_{gb} = x_{old} + \epsilon A^t$ ]
(25)     End-If
(26)     Calculate  $\epsilon A^t$ 
(27)     If ( $\epsilon A^t > A_{\max}$ ) Then
(28)       ( $\epsilon A^t = A_{\max}$ )
(29)     End-If
(30)     If ( $\epsilon A^t < A_{\min}$ ) Then
(31)       ( $\epsilon A^t = A_{\min}$ )
(32)     End-If
(33)     Generate a new solution around the current Solution  $x_i^t$  [ $x_i = x_{old} + \epsilon A^t$ ]
(34)     If  $x_i \geq x_{gb}$ 
(35)        $fx = x_i$ 
(36)     Else
(37)        $fx = x_{gb}$ 
(38)     End-If
(39)     If (Rand <  $A_i$ ) & ( $f(fx) < f(x_*)$ )
(40)       accept the new solution
(41)       Increase  $r_i$   $r_i^{t+1} = r_i^0 [1 - \exp(-\gamma t)]$ 
(42)       Decrease  $A_i$  [ $A_i^{t+1} = \alpha A_i^t$ ]
(43)     End-If
(44)   End-For
(45)   Find the current best solution ( $x_*$ )
(46) End-While

```

ALGORITHM 1: BANB Algorithm.

[50]. However, the Kolmogorov-Smirnov and Levene test did not meet the assumptions of normality distribution and equality of variance which then led us to use Wilcoxon test. Essentially, this test is an alternative to the paired t -test, when the assumption of normality or equality of variance is not met [51]. Wilcoxon test is rated to be a robust estimate tool that depended on the rank estimation [52]. Table 3 presents Wilcoxon test results for the proposed BANB algorithm

TABLE 1: Dataset characteristics.

Datasets	No. of features	No. of samples
Lung	56	32
WQ	38	521
Derm2	34	358
Derm	34	366
LED	24	2000
Mushroom	22	8124
Credit	20	1000
Vote	16	300
Heart	13	294
Exactly2	13	1000
Exactly	13	1000
M-of-N	13	1000

TABLE 2: Average of selected features.

Datasets	BANB	GA	GPSO	PSO
M-of-N	6	8	6.9	7.6
Exactly	1	7	7	6.9
Exactly2	1	3	3	2.9
Heart	4	6	5.7	6.25
Vote	1	3	3.1	3.55
Credit	1	10	11.7	11.9
Mushroom	1	5.75	6.15	5.9
LED	5	5	5	5
Derm	12.3	18.1	17.1	21.5
Derm2	11.45	17.75	17.3	20.75
WQ	5.9	20.1	20.15	21.05
Lung	2	8.6	7.95	8.7

TABLE 3: Wilcoxon test results.

	Wilcoxon test		
	GA-BANB	GPSO-BANB	PSO-BANB
M-of-N	.000 (BANB)	.013 (BANB)	.013 (BANB)
Exactly	.000 (BANB)	.000 (BANB)	.000 (BANB)
Exactly2	.000 (BANB)	.000 (BANB)	.000 (BANB)
Heart	.000 (BANB)	.000 (BANB)	.000 (BANB)
Vote	.000 (BANB)	.000 (BANB)	.000 (BANB)
Credit	.000 (BANB)	.000 (BANB)	.000 (BANB)
Mushroom	.000 (BANB)	.000 (BANB)	.000 (BANB)
LED	1	1	1
Derm	.000 (BANB)	.000 (BANB)	.000 (BANB)
Derm2	.000 (BANB)	.000 (BANB)	.000 (BANB)
WQ	.000 (BANB)	.000 (BANB)	.000 (BANB)
Lung	.000 (BANB)	.000 (BANB)	.000 (BANB)

against other feature selection algorithms. From Table 3, between the brackets refer to the algorithm that performs better than another algorithm. The results of Wilcoxon test are considered to be statistically significant at P less than 0.05 and are highly significant at P less than 0.01.

6.3. *Results for Classification Accuracy Experiment.* The second part of the experiment was to evaluate and compare the average classification accuracies achieved by BANB and

other comparative algorithms over 10 runs, using 10-fold cross-validation method. Three well-known classifiers were employed for the purpose of evaluating the resulting subsets among different classifiers, which were JRip PART and J48 [53]. Tables 4, 5, and 6 show the average classification accuracy and standard deviation values from the experiment.

7. Discussions

In selecting the feature subset, Table 2 shows that the proposed BANB algorithm obtained the smallest number of features across all datasets except for LED. Table 3 confirmed that the difference between BANB and the remaining comparative algorithms is highly significant except for LED and M-of-N datasets. More significantly, BANB is able to reduce the number of features up to a single feature in five datasets as shown in Table 2. In evaluating the feature subset, if we take into consideration the interaction between classification accuracy and number of features selected by the proposed BANB algorithm as compared to other algorithms, we can categorize the results into three cases. In the first case, a reduced number of features deliver the same classification accuracy. This is shown in the Exactly dataset that produced similar classification accuracy in both JRip and J48 classifiers and even higher accuracy in PART classifier. On the contrary, features selected by other algorithms included more features, which indicate that some of the features selected are redundant. This can be seen clearly in the Exactly2 dataset when all solutions achieved exactly the same accuracy in spite of variance in the number of selected features.

In the second case, the proposed algorithm reduced the number of features while at the same time increased the classification accuracy. For example, BANB selected only two features from the Lung dataset as opposed to additional eight features among other algorithms. The difference between the numbers of features selected is attributed to noisy features, which cause a decrease in classification accuracy such as in the Vote dataset. In the third case, smaller feature subset that is selected delivers a slightly lower classification accuracy, such as in Heart and Mushroom dataset with the exception of LED dataset. All algorithms could deliver the same accuracy with the same number of features because the LED dataset contains very protruding features.

Finally, it can be noted from Tables 4, 5, and 6 that the classification accuracies achieved by the proposed BANB algorithm are in less disagreement or very close across three different classifiers. This can be noted obviously from the experimental results using Exactly, Credit, Lung, and Derm datasets. To support this finding, we calculated standard deviation for each dataset over the three different classifiers and we averaged the values for each algorithm. The results were as follows: BANB equals 0.36, GPSO equals 0.99, PSO equals 1.04, and, finally, GA equals 1.11. This implies that the proposed feature selection algorithm BANB has better generalization as compared to other feature selection algorithms. Results from Table 2 also show that BANB is capable of selecting the same number of features for 9 out of 12 datasets over 20 iterations. This is followed by GA, GPSO,

TABLE 4: Average classification accuracy with standard deviation for JRip.

Datasets	JRip			
	BANB	GA	GPSO	PSO
M-of-N	98.9 ± 0	99.1 ± 0	97.62 ± 4.32	98.92 ± 0.23
Exactly	68.8 ± 0	68 ± 0	68 ± 0	68.01 ± 0.22
Exactly2	75.8 ± 0	75.8 ± 0	75.8 ± 0	75.8 ± 0
Heart	80.61 ± 0	81.97 ± 0	81.66 ± 0.49	82.47 ± 5.33
Vote	95 ± 0	93.66 ± 0	93.92 ± 0.43	94.42 ± 0.35
Credit	71.6 ± 0	70.7 ± 0	70.75 ± 0.75	70.48 ± 0.79
Mushroom	98.52 ± 0	100 ± 0	99.46 ± 0.48	99.33 ± 0.45
LED	100 ± 0	100 ± 0	100 ± 0	100 ± 0
Derm	93.30 ± 1.69	89.39 ± 0.90	90.18 ± 2.41	91.08 ± 0.78
Derm2	90.77 ± 1.77	89.64 ± 1.64	90.47 ± 1.31	89.35 ± 1
WQ	70.99 ± 1.57	69.49 ± 1.35	68.59 ± 1.61	68.34 ± 1.39
Lung	87.5 ± 0	84.68 ± 1.77	83.74 ± 4.37	84.05 ± 2.73

TABLE 5: Average classification accuracy with standard deviation for PART.

Datasets	PART			
	BANB	GA	GPSO	PSO
M-of-N	100 ± 0	100 ± 0	98.53 ± 4.62	100 ± 0
Exactly	68.8 ± 0	65.6 ± 0	65.6 ± 0	65.93 ± 0.82
Exactly2	75.8 ± 0	75.8 ± 0	75.8 ± 0	75.8 ± 0
Heart	79.59 ± 0	80.27 ± 0	80.57 ± 0.49	79.79 ± 1.44
Vote	94.33 ± 0	94.33 ± 0	94.42 ± 0.15	94.49 ± 0.36
Credit	71.7 ± 0	72.9 ± 0	72.24 ± 1.14	71.81 ± 1.05
Mushroom	98.52 ± 0	100 ± 0	99.33 ± 0.46	99.39 ± 0.41
LED	100 ± 0	100 ± 0	100 ± 0	100 ± 0
Derm	94.39 ± 1.76	94.73 ± 0.86	95.65 ± 0.99	95.62 ± 0.71
Derm2	93.15 ± 2.80	95.19 ± 2.82	95.27 ± 0.76	95.61 ± 0.76
WQ	68 ± 0.7	67.76 ± 1.78	68.70 ± 1.28	67.07 ± 2.92
Lung	78.12 ± 0	80.93 ± 4.52	81.21 ± 5.84	80.27 ± 4.36

TABLE 6: Average classification accuracy and standard deviation for J48.

Datasets	J48			
	BANB	GA	GPSO	PSO
M-of-N	100 ± 0	100 ± 0	98.53 ± 4.64	100 ± 0
Exactly	68.8 ± 0	68.8 ± 0	68.8 ± 0	68.8 ± 0
Exactly2	75.8 ± 0	75.8 ± 0	75.8 ± 0	75.8 ± 0
Heart	79.93 ± 0	79.25 ± 0	79.25 ± 0	79.08 ± 0.28
Vote	95 ± 0	94 ± 0	94.13 ± 0.23	94.39 ± 0.30
Credit	71.7 ± 0	72.2 ± 0	72.21 ± 0.46	72.08 ± 1.08
Mushroom	98.52 ± 0	100 ± 0	99.49 ± 0.45	99.39 ± 0.41
LED	100 ± 0	100 ± 0	100 ± 0	100 ± 0
Derm	93.92 ± 1.04	95.02 ± 0.52	94.64 ± 0.77	94.36 ± 0.68
Derm2	91.25 ± 2.44	94.38 ± 1.67	93.73 ± 0.51	94.32 ± 0.89
WQ	65.08 ± 0.46	69.11 ± 1.35	68.30 ± 2.01	68.32 ± 2.82
Lung	87.5 ± 0	79.37 ± 4.70	80.62 ± 6.21	82.18 ± 4.67

and, finally, PSO. It can be noted that the standard deviation values in Tables 4, 5, and 6 are zeros for 9 datasets. This means that our proposed BANB could obtain certain number of

features with exactly the same features for each iteration. As a consequence, BANB showed the highest stability among all comparative algorithms.

8. Conclusion

In this paper, a new hybrid feature selection algorithm has been presented. The Bat Algorithm employed Naïve Bayes Algorithm to intelligently select the most convenient feature that could maximize the classification accuracy while ignoring redundant and noisy features. We compared our proposed algorithm with three other algorithms using twelve well-known UCI datasets. The performance was evaluated from four perspectives, which are the number of features, classification accuracy, stability, and generalization. From the experiments, we can conclude that the proposed Naïve Bayes-guided Bat Algorithm (BANB) outperformed other meta-heuristic algorithms with a selection of feature subsets that are significantly smaller with a less number of features. In terms of classification accuracy, BANB has proven to achieve equal, if not better results as compared to other algorithms. For stability, the proposed algorithm is more stable than other algorithms. Finally, from the perspective of generalization of results, the resulting features produced by BANB are also more general than other algorithms in practice. For future work, further investigations are required to observe the behavior of the proposed algorithm in gene expression and very high-dimensional datasets.

References

- [1] E. G. Talbi, *Metaheuristics: From Design to Implementation*, John Wiley and Sons, 2009.
- [2] B. Yu and B. Yuan, "A more efficient branch and bound algorithm for feature selection," *Pattern Recognition*, vol. 26, no. 6, pp. 883–889, 1993.
- [3] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, vol. 209, no. 1-2, pp. 237–260, 1998.
- [4] R. Jensen and Q. Shen, "Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 12, pp. 1457–1471, 2004.
- [5] J. Yang and V. Honavar, "Feature subset selection using genetic algorithm," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 2, pp. 44–48, 1998.
- [6] L. Ke, Z. Feng, and Z. Ren, "An efficient ant colony optimization approach to attribute reduction in rough set theory," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1351–1357, 2008.
- [7] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen, "Feature selection based on rough sets and particle swarm optimization," *Pattern Recognition Letters*, vol. 28, no. 4, pp. 459–471, 2007.
- [8] M. Anbarasi, E. Anupriya, and N. Iyengar, "Enhanced prediction of heart disease with feature subset selection using genetic algorithm," *International Journal of Engineering Science and Technology*, vol. 2, pp. 5370–5376, 2010.

- [9] H. Vafaie and I. F. Imam, "Feature selection methods: genetic algorithms vs greedy-like search," in *Proceedings of the International Conference on Fuzzy and Intelligent Control Systems Louisville*, 1994.
- [10] J. C. W. Debus and V. J. Rayward-Smith, "Feature subset selection within a simulated annealing data mining algorithm," *Journal of Intelligent Information Systems*, vol. 9, no. 1, pp. 57–81, 1997.
- [11] R. Caruana and D. Freitag, "Greedy attribute selection," in *Proceedings of the 11th International Conference on Machine Learning*, pp. 28–36.
- [12] N. S. Jaddi and S. Abdullah, "Nonlinear great deluge algorithm for rough set attribute reduction," *Journal of Information Science and Engineering*, vol. 29, pp. 49–62, 2013.
- [13] M. A. Tahir, A. Bouridane, and F. Kurugollu, "Simultaneous feature selection and feature weighting using Hybrid Tabu Search/K-nearest neighbor classifier," *Pattern Recognition Letters*, vol. 28, no. 4, pp. 438–446, 2007.
- [14] A.-R. Hedar, J. Wang, and M. Fukushima, "Tabu search for attribute reduction in rough set theory," *Soft Computing*, vol. 12, no. 9, pp. 909–918, 2008.
- [15] A. Jain and D. Zongker, "Feature selection: evaluation, application, and small sample performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 153–158, 1997.
- [16] R. Leardi, "Application of a genetic algorithm to feature selection under full validation conditions and to outlier detection," *Journal of Chemometrics*, vol. 8, pp. 65–79, 1994.
- [17] R. Jensen and Q. Shen, "Finding rough set reducts with Ant colony optimization," in *Proceedings of the UK Workshop on Computational Intelligence*, pp. 15–22, 2003.
- [18] R. K. Sivagaminathan and S. Ramakrishnan, "A hybrid approach for feature subset selection using neural networks and ant colony optimization," *Expert Systems with Applications*, vol. 33, no. 1, pp. 49–60, 2007.
- [19] E.-G. Talbi, L. Jourdan, J. García-Nieto, and E. Alba, "Comparison of population based metaheuristics for feature selection: application to microarray data classification," in *Proceedings of the 6th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA '08)*, pp. 45–52, April 2008.
- [20] J. Wang, A.-R. Hedar, G. Zheng, and S. Wang, "Scatter search for rough set attribute reduction," in *Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO '09)*, pp. 531–535, chn, April 2009.
- [21] L. Bobrowski, "Feature selection based on some homogeneity coefficient," in *Proceedings of the 9th International Conference on Pattern Recognition*, pp. 544–546, 1988.
- [22] D. Koller and M. Sahami, *Toward Optimal Feature Selection*, 1996.
- [23] M. Modrzejewski, "Feature selection using rough sets theory," in *Machine Learning: ECML-93*, pp. 213–226, Springer, 1993.
- [24] H. Liu and R. Setiono, "A probabilistic approach to feature selection: a filter solution," in *Machine Learning*, pp. 319–327, 1996.
- [25] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005.
- [26] E. P. Xing, M. I. Jordan, and R. M. Karp, "Feature selection for high-dimensional genomic microarray data," in *Machine Learning*, pp. 601–608, 2001.
- [27] G. I. Webb, "Naïve bayes," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., pp. 713–714, Springer, New York, NY, USA, 2010.
- [28] G. Feng, J. Guo, B.-Y. Jing, and L. Hao, "A Bayesian feature selection paradigm for text classification," *Information Processing and Management*, vol. 48, no. 2, pp. 283–302, 2012.
- [29] H. T. T. Nguyen and D. K. Le, "An approach to improving quality of crawlers using naïve bayes for classifier and hyperlink filter," in *Computational Collective Intelligence. Technologies and Applications*, N. T. Nguyen, K. Hoang, and P. Jędrzejowicz, Eds., pp. 525–535, Springer, Berlin, Germany, 2012.
- [30] Z. Zhang, Q. Zhu, and Y. Xie, "A novel image matting approach based on naïve bayes classifier," in *Intelligent Computing Technology*, D. S. Huang, C. Jiang, V. Bevilacqua, and J. Figueroa, Eds., pp. 433–441, Springer, Berlin, Germany, 2012.
- [31] M. C. Yang, C. S. Huang, J. H. Chen, and R. F. Chang, "Whole breast lesion detection using naïve bayes classifier for portable ultrasound," *Ultrasound in Medicine and Biology*, vol. 38, pp. 1870–1880, 2012.
- [32] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Information Sciences*, vol. 179, no. 8, pp. 1040–1058, 2009.
- [33] Z. Hoare, "Landscapes of Naïve Bayes classifiers," *Pattern Analysis and Applications*, vol. 11, no. 1, pp. 59–72, 2008.
- [34] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies For Optimization (NICSO'10)*, J. González, D. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds., pp. 65–74, Springer, Berlin, Germany, 2010.
- [35] T. C. Bora, L. D. S. Coelho, and L. Lebensztajn, "Bat-inspired optimization approach for the brushless DC wheel motor problem," *IEEE Transactions on Magnetics*, vol. 48, no. 2, pp. 947–950, 2012.
- [36] K. Khan, A. Nikov, and A. Sahai, "A fuzzy bat clustering method for ergonomic screening of office workplaces," in *Proceedings of the 3rd International Conference on Software, Services and Semantic Technologies (S3T '11)*, D. Dicheva, Z. Markov, and E. Stefanova, Eds., pp. 59–66, Springer, Berlin, Germany, 2011.
- [37] X. S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, 2012.
- [38] A. H. Gandomi, X. S. Yang, A. H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing and Applications*, vol. 22, pp. 1239–1255, 2012.
- [39] I. Fister Jr., D. Fister, and X. S. Yang, "A hybrid bat algorithm," *Elektrotehnicki Vestnik*, vol. 80, pp. 1–7, 2013.
- [40] X. S. Yang, "Bat algorithm for multi-objective optimisation," *International Journal of Bio-Inspired Computation*, vol. 3, pp. 267–274, 2011.
- [41] A. M. Taha and A. Y. C. Tang, "Bat algorithm for rough set attribute reduction," *Journal of Theoretical and Applied Information Technology*, vol. 51, no. 1, 2013.
- [42] R. Nakamura, L. Pereira, K. Costa, D. Rodrigues, J. Papa, and X. S. Yang, "BBA: a binary bat algorithm for feature selection," in *Proceedings of the 25th Conference on Graphics, Patterns and Images (SIBGRAPI '12)*, pp. 291–297, 2012.
- [43] R. S. Parpinelli and H. S. Lopes, "New inspirations in swarm intelligence: a survey," *International Journal of Bio-Inspired Computation*, vol. 3, pp. 1–16, 2011.
- [44] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1989.
- [45] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, 1998.

- [46] A. Moraglio, C. Di Chio, and R. Poli, "Geometric particle swarm optimisation," in *Proceedings of the 10th European Conference on Genetic Programming (EuroGP '07)*, pp. 125–136, April 2007.
- [47] B. Raman and T. R. Ioerger, *Instance Based Filter for Feature Selection*, 2002.
- [48] C. L. Blake and C. J. Merz, *UCI Repository of Machine Learning Databases*, University of California, Irvine, Calif, USA, 1998.
- [49] Z.-Q. Hong and J.-Y. Yang, "Optimal discriminant plane for a small number of samples and design method of classifier on the plane," *Pattern Recognition*, vol. 24, no. 4, pp. 317–324, 1991.
- [50] H. W. Lilliefors, "On the Kolmogorov-Smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association*, vol. 62, pp. 399–402, 1967.
- [51] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, pp. 80–83, 1945.
- [52] F. Monti, R. Dell'Anna, A. Sanson, M. Fasoli, M. Pezzotti, and S. Zenoni, "A multivariate statistical analysis approach to highlight molecular processes in plant cell walls through ATR FT-IR microspectroscopy: the role of the α -expansin PhEXPA1 in *Petunia hybrida*," *Vibrational Spectroscopy*, vol. 65, pp. 36–43, 2013.
- [53] Q. Duan, D. Miao, H. Zhang, and J. Zheng, "Personalized web retrieval based on rough-fuzzy method," *Journal of Computational Information Systems*, vol. 3, no. 3, pp. 1067–1074, 2007.

Research Article

Hybrid Model Based on Genetic Algorithms and SVM Applied to Variable Selection within Fruit Juice Classification

**C. Fernandez-Lozano,¹ C. Canto,¹ M. Gestal,¹ J. M. Andrade-Garda,²
J. R. Rabuñal,¹ J. Dorado,¹ and A. Pazos¹**

¹ *Information and Communications Technologies Department, Faculty of Computer Science, University of A Coruña, Campus Elviña s/n, 15071, A Coruña, Spain*

² *Analytical Chemistry Department, Faculty of Sciences, University of A Coruña, Campus da Zapateira s/n, 15008, A Coruña, Spain*

Correspondence should be addressed to C. Fernandez-Lozano; carlos.fernandez@udc.es

Received 24 September 2013; Accepted 21 October 2013

Academic Editors: Z. Cui and X. Yang

Copyright © 2013 C. Fernandez-Lozano et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Given the background of the use of Neural Networks in problems of apple juice classification, this paper aim at implementing a newly developed method in the field of machine learning: the Support Vector Machines (SVM). Therefore, a hybrid model that combines genetic algorithms and support vector machines is suggested in such a way that, when using SVM as a fitness function of the Genetic Algorithm (GA), the most representative variables for a specific classification problem can be selected.

1. Introduction

The evolution of technology has made accessible a wide range of information in almost any area of human activity. The increased connectivity and the development of Internet have allowed access to large volumes of data. Moreover, the increase of the processing capacity of computers, as well as the current low cost of computer storage, allows preserving up to the last generated byte.

However, even having the means necessary for the data preservation, it is common to find a large number of irrelevant information when solving a problem. Only a fraction of the data has different and significant profiles, whereas the rest is redundant information, unwanted noise, or worse, information captured or stored incorrectly. In recent years, various variable selection methods have been developed with the aim of mitigating these problems as far as possible [1].

The feature selection or extraction is first introduced in the early 1960s, becoming a research topic widely studied and used over the last four decades [1–3]. The need for these techniques arises, as noted, from the emergence of certain predictive variables that may become irrelevant or redundant.

A predictive variable is considered irrelevant if its content does not provide any information that could clear uncertainty about the original set. Similarly, it is redundant when its value can be determined from other predictive variables. The variable selection methods are aimed at detecting redundant and/or trivial variables for a given problem. The purpose of using these techniques is to identify an “optimal” subset, made up of a minimum number of variables necessary to find a valid solution to a problem.

As a result of applying these methods, some benefits are obtained, consolidating their position throughout time [4–6]. Among others, it is worth mentioning that the variable selection methods allow decreasing the number of samples required to obtain optimum results in virtually any classification or clustering problem. This leads, for example, to the creation of a faster prediction model when it comes to analyzing a lower number of variables. Similarly, cost reduction is obtained, taking into account a temporal perspective as well as the one regarding the system complexity for data acquisition, since we are dealing with a smaller amount of information. As a result, the complexity of the problem to be solved is reduced.

In this paper, we propose the use of a prediction hybrid system combining GA and SVM. First, we will briefly mention the most representative selection techniques in recent times. Then, we will describe the methods used to develop the proposed system. Next, we will explain concisely the developed model, as well as the results obtained with it. These results, related to the classification of juice samples according to their sugar concentration, will be compared to those obtained in previous studies from the use of ANN, allowing a comparative study of both methods.

2. State of the Art

In recent years, the methodology used by different organizations has been forced to evolve due to the fact that storage and processing capacities have substantially increased. This progress has led researchers to develop a large number of selection and learning techniques that allow improving their working methods.

The variable selection techniques have become a very useful tool. Among the existing techniques it is worth mentioning, as the most representative ones, Principal Component Analysis (PCA)—a mathematical method [7], and Partial Least Squares (PSA) [8], which behave similarly from the point of view of variable selection.

The main component analysis method is characterized as a statistical technique for the information synthesis or dimension reduction [7]. We are dealing with a linear algorithm whose operation is based on the correlation between variables. The aim of these methods is to minimize the number of variables as much as possible with the minimum loss of information. However, its use entails a series of drawbacks. The most important one comes from the set of variables obtained from their use, as these latter one do not belong to the initial set but are a linear combination of them.

On the other hand, PLS is a mathematical approach used to establish a model that relates the information from two different datasets. The idea is to look not only for the directions with a larger amount of information within the set of predictive variables but also to select those which have a stronger relationship with the variables to be predicted [8, 9]. This is why it is said that the PLS models are governed by a criterion of predictive ability rather than by the fitness of the model to the data.

The use of these or other mathematical methods involves a series of drawbacks that encourage the search for new techniques, aimed at reducing them. Among the common disadvantages of using these methods, it is worth mentioning two main drawbacks. On the one hand, it is necessary to have a broad knowledge of the field to be discussed, since, depending on certain features (linearity, interdependence, etc.) it will be possible to apply only some methods or others. On the other hand, such methods tend to have a single valid solution, not a set of solutions, so that they would not be efficient in problems in which there are several global solutions, or a global solution and local variables.

In several studies, evolutionary methods such as variable selection techniques were used [10, 11] due to the fact that they minimize the drawbacks of using the above-mentioned

techniques. From the field of evolutionary computation and, more specifically, from the area of Gas, there has been a significant number of approaches towards variable selection methods.

That is why, in order to increase the reliability of the variable selection methods, we have opted for new strategies, among which we consider of great importance the hybrid methods that employ GAs and, on the one hand, prediction mechanisms, among which artificial neural networks (ANNs) are worth mentioning.

The ANNs are methods of nonparametric prediction-classification that allow obtaining a high degree of accuracy in many problems, especially those related to human knowledge [12–14]. They are based on a computational model established according to biological neural systems that try to simulate the operation of human brain neurons at small scale. The training of the weights of the ANN through traditional neuronal network learning methods has proven to be a very effective process. The creation of a hybrid model combining GAs and ANN allows the use of the former system as a guideline for improving the latter. As a result, GAs are employed to optimize the number of entries into the neural predictor classifier, thus decreasing the generalization error and consequently the size of the models.

There have been several studies that have opted for the combination of evolutionary methods with classification mechanisms [6, 13, 15–17]. However, even after having developed very efficient models, ANNs have the disadvantage of undergoing training processes on a strong stochastic basis, leading to nonrepeatability of the process. This is why many researchers prefer to choose new techniques based on robust statistical principles such as support vector machines (SVM) [18–23].

Developed by Vapnik [24, 25] and based on the statistical learning theory, the support vector machines are fast becoming one of the most used methods of prediction classification. Although their use is fairly recent, a considerable number of researchers have already reported states of the art of their performance in a variety of applications in pattern recognition, regression estimation, and prediction of time series. For example, the study carried out by Min et al. [26] combined the SVM and the GAs with the aim of predicting the business failure risk and avoiding bankruptcy, with a particular model being tested in predicting the crisis in Taiwan by Wu et al. [27]. Tan et al. [11], besides combining GAs and SVM, opted for the use of PLS for the identification of mitochondrial proteins. There are also some comparisons of SVM with regression methods [28]. Huerta et al. [29] combined a GA with SVM for gene selection and classification of microarray data. Venegas [30] used the SVM in the classification of academic texts according to their lexical-semantic content; in the study carried out by Donís et al. [31] the SVM were used to estimate the creep rupture stress of ferritic steel. Pérez et al. [32] applied the SVM to a typical classification problem of failure identification in distribution systems of electric power. In Fernandez-Lozano et al. [15, 16], a hybrid approach combines GAs and SVM for protein identification in two-dimensional gel electrophoresis images. Tong et al. [33] proposed a GA based ensemble SVM classifier built

on gene pairs. A GA-based for solving two dual quadratic programming problems with a twin parametric-margin SVM in the primal space is proposed by Wang et al. [34]. Won et al. [35] combined a novel GA-SVM to find features from biological sequences. A particle swarm optimization (PSO) [36] to optimize a GA-SVM method to predict single nucleotide polymorphisms (SNPs) and to select tag SNPs as pointed by Ilhan and Tezel [37] or to solve the heating system planning problem is presented [38]. Zhang et al. combined PSO with SVM for classifying magnetic resonance imaging (MRI), brain images [39]. Ocak [40] combined a GA with a SVM in a medical decision support system for the evaluation of fetal wellbeing. Uzer et al. [41] combined an artificial bee colony algorithm with SVM for classification. A hybrid approach with SVM and microarray data is presented by Li et al. [42]. A cross-study comparison of classification methods including ANN and SVM for predicting metastasis in breast cancer is presented by Burton et al. [43]. Other approaches using evolutionary computation techniques are presented in [44, 45].

3. Methods

3.1. Selection of Variables. Currently, there are several variable selection techniques. One of the guidelines used for their classification is given according to the approach used: the indirect or filter approach on the one hand [46] and the direct or wrapper approach [47] on the other hand. The filter techniques select subsets of variables in a preprocessing step regardless of the classification problem. On the contrary, the wrapper methods use machine learning to assign a rating to the subsets of samples according to their predictive ability. There is a third group of techniques, called embedded, which perform the selection of variables during the very learning process or classification of samples.

Once the type of variable selection technique to be used is defined, it is necessary to establish a mechanism which allows carrying out the search for significant variables. Ideally, the selection should be performed in terms of the entire subset of variables that can be formed, but however, this would involve analyzing numerous combinations, with their corresponding computation time loss. That is why we use search strategies that provide results as close as possible to the overall optimum value. As shown in Section 3, different methods have been developed in order to explore the set of variables, among which the best known is the one based on the use of GAs [48, 49].

Several studies have proposed the combination of GAs and selection and prediction methods such as selection techniques and data prediction, respectively. One of the selection-prediction methods most used in these hybrid models is the ANNs [50]. However, these models are not free of drawbacks. One of the criticisms associated with the use of the ANNs is due to their "black-box" characteristic, as it is difficult to understand their internal operation and the process which leads them to determine the appropriate solution when it comes to a set of patterns. Another drawback of their use as a selection or prediction method, and perhaps the most important one, is the one concerning the exact nonreproducibility of their training (or at least of their

difficulty) as a result of the stochastic process by which the weights are initialized. GAs are search techniques inspired by Darwinian Evolution and developed by Holland in the 1970s [51]. GAs for feature selection were first proposed by Siedlecki and Sklansky [52]. Many studies have been done on GA for feature selection since then [6, 53], concluding that GA is suitable for finding optimal solutions to large problems with more than 40 features to select from. GA for feature selection could be used in combination with a classifier such SVM, optimizing it.

Today, there are other prediction-selection techniques that allow making up for the shortcomings caused by the use of ANNs. This paper is aimed at emphasizing a method recently developed within the field of machine learning, the SVM. This is a classification and regression algorithm family which currently shows comparable or better results than those obtained with ANNs or other statistical models in problems of pattern recognition, prediction, classification, or data mining. From their inception to the present day, the SVM have evolved in such a way that they have become a successful tool when dealing with highly dimensional data.

3.2. Genetic Algorithms. The evolutionary computation (EC) has reintroduced concepts of evolution and genetics to solve problems, mainly those related to optimization tasks [48, 54]. However, it is also worth mentioning the important influence of the studies carried out by Turing and Samuel in the 1950s, "Can machines think?" and "How can computers learn to solve problems without being explicitly programmed?" [55, 56].

Broadly speaking, the EC methods are search and optimization techniques consisting of the application of heuristic rules based on principles of natural evolution. In other words, these are algorithms that look for solutions according to properties of genetics and evolution. Among these properties, it is worth mentioning the survival of the fittest individuals (which implies that once the best solutions to a problem are reached, they will keep being this way) and heterogeneity (we mean basic heterogeneity, so that algorithms could have numerous types of information when creating solutions).

3.2.1. General Outline of Operation. The evolutionary algorithms base their operation on a relatively simple outline, as shown in Figure 1. This iterative outline will refine solutions which will gradually be closer and closer to obtaining an overall solution of the problem.

But prior to the implementation of the evolutionary process specified by the algorithm, it is necessary to undertake two issues, perhaps the most important ones in the whole process: determining how to represent the solutions (encoding) and specifying a method which would allow us to know how good a solution is (fitness function).

One of the most used branches in the EC is made up of genetic algorithms (GAs) [49, 57]. In this case, the encoding of the solutions is performed through value chains (the chains being of fixed or variable length, with the values of these being bits, whole and real numbers, etc.).

The critical step when setting up an evolutionary algorithm is the definition of the fitness function. This function

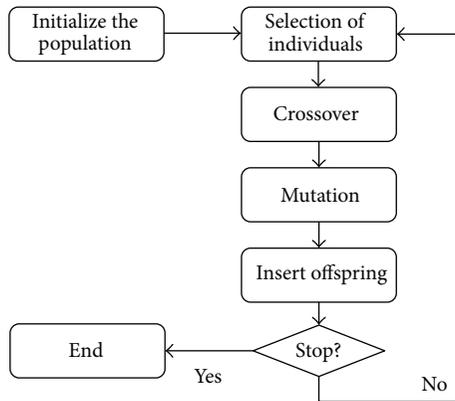


FIGURE 1: General outline of the operation of an evolutionary algorithm.

will have to evaluate every genetic individual, indicating a real value representing the goodness of the solution provided by the individual. This function will be responsible for guiding the search process in either direction. Precisely because we are dealing with a function responsible for verifying the goodness of each solution, this is an inherent aspect linked to the problem to be solved.

The above discussed evolution of solutions will happen due to crossover and mutation genetic operators which simulate processes of sexual and asexual reproduction that occur in a natural environment. Next, we will summarize each one of the remaining steps of which such algorithms consist.

Initialization. The power of evolutionary algorithms lies in the massively parallel exploration of the search space. This can occur due to the existence of numerous solutions, each exploring an area of the search space. The set of solutions, randomly initialized, is called genetic population.

Selection. The selection algorithms will be responsible for choosing the individuals which will and which will not have the opportunity to reproduce [58]. Since this is a simulation of what occurs in the natural environment, the fittest individuals have to be given more opportunities to reproduce. Therefore, the selection of an individual is related to its fitness value. However, the reproduction options of less fit individuals should not be completely eliminated, because in a few generations the population would become homogeneous in this way.

Crossover. Once the individuals are selected, they are recombined to produce offspring that will be inserted into the next generation. Their importance for the transition between generations is high because the usually employed crossover rates are around 90%.

The main idea of the crossover is based on that, if two individuals, properly adapted to their environment, are selected and the offspring obtained shares genetic information from both, there is a possibility that the inherited

information is precisely the cause of their parents' goodness. By sharing the good features of two individuals, the offspring, or at least part of them, should have better characteristics than each parent separately.

Mutation. An individual's mutation causes that the value of one of its genes or nodes, usually only one of them varies randomly.

Although individuals can be selected directly from the current population and mutated before being introduced into the new population, the mutation is often used together with the crossover operator. Thus, the behavior that occurs in the natural environment is simulated, since when generating the offspring there is always some kind of error, usually with no consequence throughout the transmission of the genetic load from parents to offspring. The mutation causes sometimes a reduction in the individual's fitness value (which can be remedied in subsequent generations). However, the new information contributes directly to a significant increase of the goodness of the solutions or it can be a part of a better solution in future generations.

Replacement. The traditional operation of ANNs often includes the use of a temporary population. This latter is being filled by copying individuals and with the offspring generated due to crossover operations (and mutation, if that is the case). When this temporary population is complete—in this case it is said to have passed to a new generation—it becomes the population of the current generation, ruling out the previous one and repeating the process from a new empty temporary population. Such algorithms are usually called generational algorithms.

However, there is another approach, called steady-state algorithms. This option consists of working with a single population, which undergoes selections and insertions, ruling out the use of a temporary population. In this case, since the number of individuals in the population remains constant, it should be noted that a new individual cannot be added unless another is eliminated before that.

Stopping Criterion. As previously explained, the evolution process of solutions is essentially an iterative process. Therefore, it will be necessary to specify a criterion that allows establishing when the execution is completed. Once more, there are different options, but the most common ones are shown as follows.

- (i) The fittest individuals in the population represent solutions good enough so that the problem could be solved.
- (ii) The population has converged. A gene has converged when 95% of the population has the same value (or a very similar one) for that gene. Once all the genes reach convergence it is said that the population has converged. When this phenomenon happens, the average goodness of the population is close to the goodness of the fittest individual.

- (iii) The difference of the best solutions found between different generations is reduced. This may indicate, at the very best, that the population has reached an overall solution or on the contrary that the population has come to a standstill at a local minimum value.
- (iv) A predetermined maximum number of generations have been reached.

It may be worth mentioning that the advantage of such techniques is the simplicity of their implementation. No technical knowledge is required to solve the problem, only one way that allows evaluating a possible solution (in order to define the fitness function). Moreover, it should be also noteworthy the simplicity of the ideas taken from the natural environment, on which the evolution of solutions is based.

In addition, this type of techniques is easily adaptable to multimodal problems (those with multiple solutions) [59] or multiobjective problems (those in which different criteria are optimized simultaneously) [60].

When the computational cost is a criterion to be considered due to its inherently parallel operation, we are dealing with easily distributable techniques (or at least the evaluation of the solutions, which often becomes a hurdle) with a marked improvement in the response time arising from such distribution.

Finally, we should note that these techniques, unlike others, always provide a solution to the problem raised and, in addition, this solution will be improving as implementation is carried out over time.

3.3. Support Vector Machines. The support vector machines are general methods for solving problems of classification, regression, and estimation. They are learning systems based on the studies performed by Vapnik on the statistical learning theory [24, 25]. From their inception to the present day, they have become the subject of continuous research and application. The interest raised by this method has increased considerably, becoming a referent for the other disciplines of machine learning and data mining.

At first, the SVM were developed to solve problems of binary classification (two classes), but currently, and throughout their evolution, they have widened their field of action, dealing with any kind of problems. The SVM are aimed at finding a linear optimal hyperplane distributing the data into two or more classes, so that all those elements which belong to the same class are located on the same side. This is equivalent to solving a classical quadratic programming problem, which guarantees the existence of a single solution and a reasonable efficiency for real problems with thousands of examples and attributes.

Intuitively, it seems obvious to come to the conclusion that when solving a linear classification problem, there is a high probability of obtaining several solutions which could correctly classify the information, as shown in Figure 2.

Therefore, the question to be answered is which of the alternatives is the ideal one? In his studies, Vapnik answered this question by defining the concept of optimal hyperplane. "A hyperplane is said to be optimal if it maximizes the margin over all hyperplanes (see Figure 3) [61].

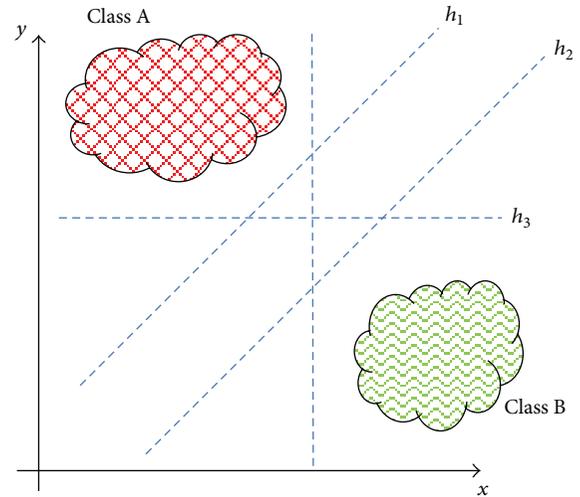


FIGURE 2: Linearly separable classification.

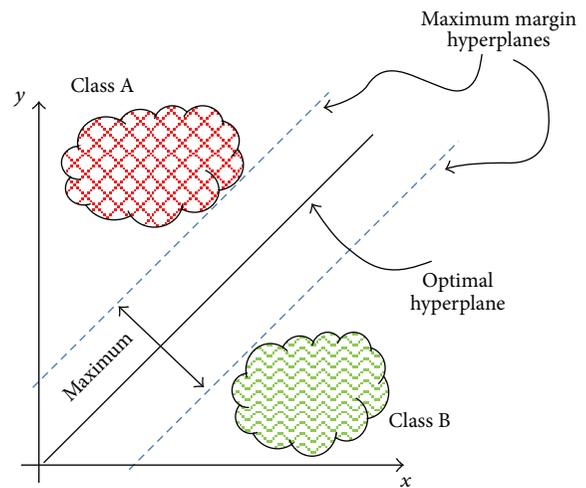


FIGURE 3: Linearly separable classification; hyperplane and separation margin.

Once defined the concept of optimal hyperplane, and after carrying out several studies, it was observed that the hyperplane could be defined only if considering certain data from the training set. These characteristic points are called "support vectors," and they are those instances of each class which are closest to the hyperplane with maximum margin.

However, in most of the existing problems, the data are not linearly separable, so that the implementation of the above-mentioned process does not achieve a good result. To solve this drawback, we should tackle these problems with different strategies, thus achieving a linear separation but in a different space. To this end, a transformation of input variables is performed in a dimensional space greater than the one to which they belong (the greater dimensional space being a Hilbert space):

$$x : \mathbb{R}^n \mapsto \varphi(x) : \mathcal{H}. \tag{1}$$

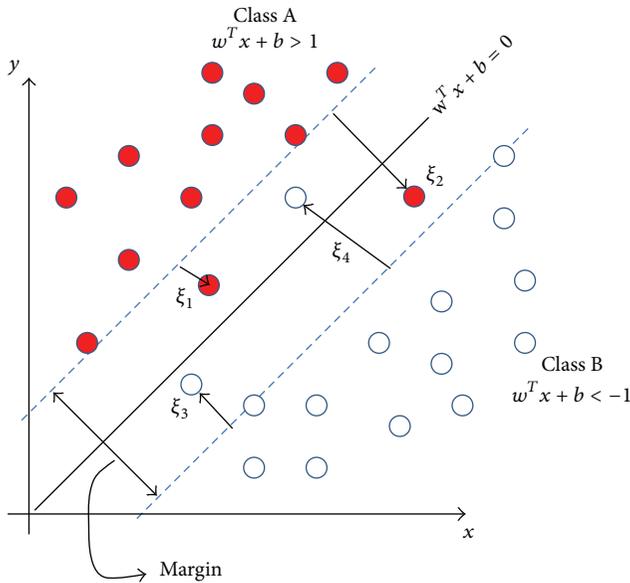


FIGURE 4: Slack variables.

The next step is to find a hyperplane (actually a scalar product of vectors that can be expressed as a function of the input space x) in this new dimension that allows separating the data linearly. The result of this scalar product is called kernel, and the most common ones are as follows:

- (i) linear kernel: $K(x_i, x_j) = x_i^T x_j$,
- (ii) Gaussian kernel: $K(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / 2\sigma^2)$,
- (iii) polynomial kernel (order n): $K(x_i, x_j) = (x_i^T x_j + 1)^n$.

In general, a kernel is any function $K(u, v)$ that verifies Mercer's theorem [62], that is, any function that verifies

$$\int_{u,v} K(u, v) g(u) g(v) du dv > 0 \tag{2}$$

for every $g(\cdot)$ function of integrable square.

Given the above, if the transformed function gives rise to a linearly separable space search, Vapnik and Chervonesky [63] showed that maximizing the separation margin between classes is equivalent to the minimization of the Euclidean norm of the weight vector. That is, considering the approximation of the set $\{x_i, y_i\} \text{ con } x \in \mathbb{R}^n, y \in \mathbb{R}$ by the linear function $f: X \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$,

$$f(x) = w^T x + b = \sum_{i=1}^n w_i x_i + b, \tag{3}$$

where $w \in \mathfrak{R}^n$ (weight vector) $yb \in \mathfrak{R}^n$ (vector of bias) are the parameters which define the hyperplane, as the lowest training and complexity error is obtained looking for the minimal $w \in \mathfrak{R}^n$.

But what happens when some datum is still not linearly separable in this new dimension? As shown in Figure 4, in this case the solution is to introduce a new set of slack

variables: $\xi_i, i = 1, \dots, N$, representing an estimate of the error on the optimal hyperplane.

Therefore, the problem leads to the search of a classification function $f(x)$ that minimizes the sum of these losses reflected by the slack variables.

In this case, the function to be minimized would be as follows:

$$\Phi(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i, \tag{4}$$

where C is a variable empirically specified by the user with the aim of controlling the tradeoff between the model complexity and the number of not separable data. More specifically, the greater is the value of this parameter, the higher is the assigned penalty to errors. Depending on the value of C , the margins of a boundary decision will vary their forms. As a result, we can conclude that the higher is the value, the narrower is the margin and the lower is the classification error in the training phase. On the contrary, the wider is the margin, the higher is the classification error in the training phase.

4. Proposed System

The method proposed in this work is based on creating a hybrid model that combines a GA and support vector machines, with the aim of classifying samples before selecting the minimum number of significant variables.

The GALib library [64], developed by Matthew Wall in 1996 and last modified in 2007, was used to encode the genetic algorithms. Similarly, the WinSVM code was used to implement vector machines, a code developed by Sewell [65]. WinSVM provides as output a mean squared error (MSE) to measure the distance between the samples incorrectly classified on the optimal hyperplane. Thus, the obtained MSE value will be deterministic; hence, this will be one of the criteria which will be subsequently considered when comparing different executions.

According to those mentioned in the previous section, the SVM are extremely useful when trying to make dichotomous classification of data, that is, to distinguish between two classes. However, this idea can be generalized to identify, among a set of n categories, to which a certain datum belongs. In this work we have chosen to raise the following approach: considering that in the total data set n categories (C_1, C_2, \dots, C_n) can be defined for each possible C_i category existing in the input set, and an SVM is created. This latter will try to distinguish whether a datum belongs to the given C_i category or to the remaining set. Finally, to determine to which specific category each datum belongs, we have simply implemented all the defined SVM and we have selected that output which indicated a greater degree of belonging to a particular class (e.g., that in which the MSE value is lower).

With the aim of creating the above-mentioned hybrid model, we have modified the traditional operation of the GA in such a way that it could generate a population of individuals of varying length [66]. To this end, we have implemented an initialization function that will be responsible

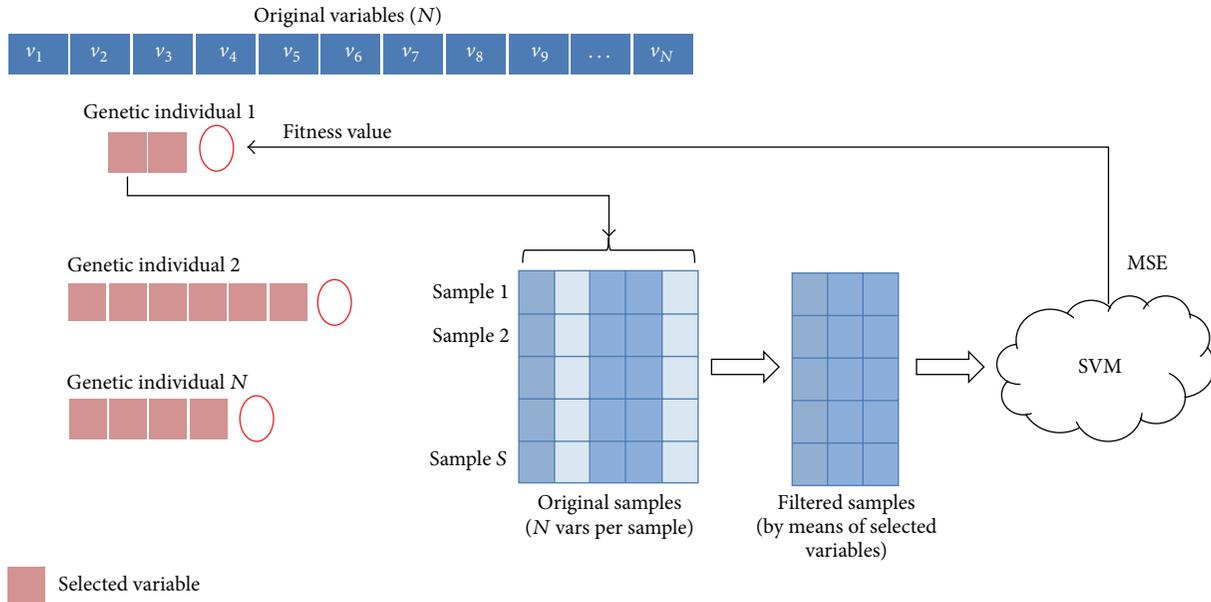


FIGURE 5: Evaluation of the genetic individuals.

```

M = data dimension
N = population size
for i = 1 to N
    perm = randomPermutation(M)
    for g = 1 tonumberOfGenes
        individual[i].Gene[g] = perm[g]
    end for
end for
    
```

ALGORITHM 1

for, firstly, either performing a random generation or with a predetermined size, of the length of each individual making up the population of the genetic algorithm and, secondly, initializing the value of each gene. For this purpose, we have selected from the total set of variables a subset of random size so that, if the variables generated are different, the subset will be assigned to the individual. Otherwise, a new random combination will be generated until the condition stipulated is met. This procedure is repeated until all individuals in the GA population have an assigned subset. Therefore, the result is a set of individuals as shown in Algorithm 1.

Once the GA population is created, it will be assessed by a fitness function.

In this paper, we suggest the use of the SVM as a fitness function of the genetic algorithm. Thus, for each individual, and depending on the indicated positions, a training and validation set is created from the initial data. These sets will be applied to the SVM which, once the prediction is made, will yield a mean squared error (MSE) to be used as a scale to determine the fittest individual (see Figure 5).

Nevertheless, when the information refers to a nonbinary (or dichotomic) classification, it is necessary to modify a

small aspect. It will be supposed that the problem has N classes $\{c_1, c_2, \dots, c_n\}$. In this case an individual SVM is applied, using the variables specified by the genetic individual, to discriminate between each of the classes c_i and the rest a MSE being obtained. In this case of multiclass problems, the value of kindness of the individual will be the sum of the MSE obtained for every classification.

This choice of the fitness function allows, among other advantages, a repeatability of results, which hardly ever applies to other techniques such as artificial neural networks, used in similar problems tackled in previous studies [13, 67, 68].

Therefore, this paper is a step forward regarding previous studies on selection of variables in an experimental field on which numerous tests have been carried out. Still, as shown below, the obtained results significantly improved the previous ones.

5. Materials

5.1. Data Description. Nowadays the society awareness has evolved into the need for a more and more healthy diet in order to improve the quality of life. Undoubtedly, the juice manufacturing industry, influenced by these new circumstances, has enjoyed a boom in both production and sales. However, the increasing production of these industries leads to an increase regarding the level of adulteration of their products. Consequently, the search for new methods that allow identifying the exact amount of pure juice used to produce these products has become an issue of great importance in recent years [69].

In order to prevent and detect adulteration in food, this latter must be subjected to an increasingly strict series of quality control tests. This is due to the fact that the commonly used analysis techniques have become obsolete

TABLE 1: Samples with high concentration.

%	Range: 20%–100%		Commercial
	Training	Test	
20%	20	6	
25%	19	18	2
50%	16	13	
70%	14	1	
100%	17	6	19
Total	86	44	21

with their development and progress. Different techniques such as HPLC (*high performance liquid chromatography*) gas chromatography, or isotope methods are too slow and relatively expensive and, therefore, they are not suitable for carrying out routine analysis. On the other hand, IR (infrared) spectroscopy provides a quick and cheap alternative, which, besides these already mentioned characteristics, provides great information about the main components of the juice.

Therefore, in order to perform testing, we have used information that allows verifying the authenticity of the apple juice quality. Using various types of apples, such as Golden Delicious, Gloster, Granny, Smith, Reineta, Royal Gala, and Starking, their juice has been extracted for subsequent centrifuging, filtering and classification using Fourier transform mid-infrared attenuated total reflectance (FTMIR-ATR). As a result, we have obtained a series of samples of diluted pure juice, which will be used to obtain the different training and validation sets.

Two sets of samples will be considered: one for samples with high concentration of juice (see Table 1) and one for samples with low concentration of juice (see Table 2). Consequently, beverages with a low concentration of juice fall within what is called energy drinks (among which soft drinks are included), while beverages with a higher concentration receive the generic name of juices. The samples made up of 20% of diluted juice (the boundary between what is considered low and high concentrations) are found in both sets.

All samples were characterized by means of an infrared spectroscopy. As a result of this characterization, a spectrum (as shown in Figure 6) is obtained for each sample, which represents the amount of energy absorbed (or absorbance) for a total of 176 wavelengths or variables.

The objective is to determine the amount of juice in a sample using only the information provided by this spectrum. However, using all the information of the spectrum does not lead to fully satisfactory results, as shown below. That is why the need for applying a process of selection of variables is arisen. This would have two obvious advantages. On the one hand, it is time-saving (both when obtaining the spectrum and in the subsequent classification, since a smaller amount of information is involved). On the other hand, and perhaps most importantly, the expert is provided further information about which part of the spectrum, that is, what specific type

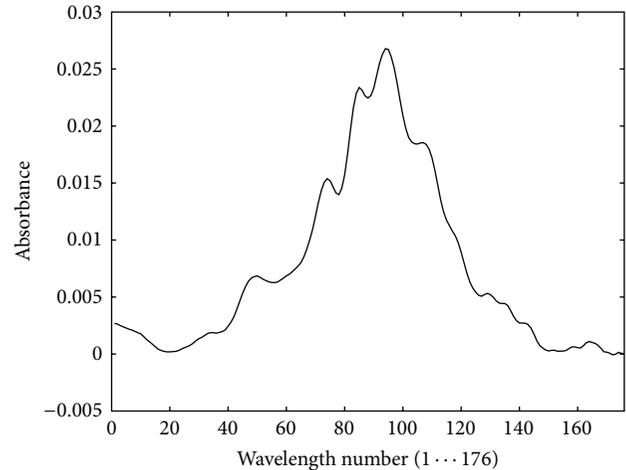


FIGURE 6: IR spectrum, specific to a sample.

TABLE 2: Samples with low concentration.

%	Range: 2%–20%		Commercial
	Training	Test	
2%	19	1	
4%	17	1	
6%	16	13	
8%	22	6	
10%	21	6	1
16%	20	6	1
20%	19	6	
Total	134	39	2

of sugar (fructose, sucrose, etc.), provides more information to carry out this classification.

5.2. Parameter Setting. The first experiment that should be performed before running the developed model is the choice of setting parameters for both vector machines and GA. To this end, we have applied the following procedure.

To determine the setting parameters of the SVM algorithm, the steps proposed by the author have been followed, using the “optimize” option provided by the algorithm itself. This function is based on a random generation of different combinations of the SVM parameters, which are then applied to the initial data obtaining a mean squared error (MSE) as a result of the application.

A total of 100 different combinations were generated for this purpose, and from among them, and taking as scale the training MSE, we have chosen those with the best results in the training cases. However, when performing this first test, there were no entirely satisfactory solutions (see Table 3). On the other hand, this fact is not really significant since the final aim in this phase is to determine the configuration parameters of the algorithm, not to carry out a real data classification.

Tables 3 and 4 show the three best combinations obtained from the 100 tested (for both high and low concentration),

TABLE 3: Setting of SVM parameters: samples with high concentration.

	SVM parameters			MSE train
	C	Epsilon	Radial	
1°	10000	0,000001	5	3,83E - 07
2°	10000	0,0001	2	4,49E - 07
3°	10000	0,0001	2	4,49E - 07

TABLE 4: Setting of SVM parameters: samples with low concentration.

	SVM parameters			MSE train
	C	Epsilon	Radial	
1°	1000	0,000001	2	5,26E - 07
2°	1000	0,000001	10	5,95E - 07
3°	1000	0,0001	2	5,95E - 07

selecting as optimal the one whose training mean squared error is lower. It is noted that the penalty variable C , in both cases, has a relatively high value ($C = 10,000$ for high concentration and $C = 1,000$ for low concentration), leading to, as mentioned above, a considerable reduction of the error training (3.83E-07 and 5.26E-07, resp.) finally obtained.

The configuration of the genetic algorithm parameters, carried out similarly as in the case of the SVM, should be selected after performing a series of tests by varying their value. From the range of tests, the configuration shown in Table 5 was taken as optimal.

6. Results and Discussion

First, to allow the comparison of results, a reference model should be established. In this case, we have chosen the original set of 176 variables provided by the IR spectrometer to build different classification models starting from the former. More specifically, we have chosen some of the most widely used models in the field of analytical chemistry with this type of data (partial least squares: PLS, SIMCA or potential functions), together with a model generated from the use of ANN and another one using SVM for the classification. Tables 6 and 7 show the results obtained with each of these models.

At first, given the great performance offered by the ANNs, we had chosen this technique to guide the search for the GA. However, as discussed below, they have the great disadvantage of the repeatability of results, which leads to the fact that the variables selected as the most relevant are different in each iteration.

On the other hand, the results obtained in this first test using SVM can seem discouraging. However, this can be explained according to the above classification criteria. Thus, for a given concentration, SVM are applied for each type of concentration (20%, 25%, ...) obtaining a good or bad classification result regarding the individual classification. A good result in the final classification of the sample will be taken into account only if the datum is always assigned (i.e., for all the developed SVM) to the right category. For example,

TABLE 5: Genetic algorithm configuration.

(i) GA: simple	(i) Selection: wheel-roulette
(ii) Number of individuals: 100	(ii) Number of generations: 100
(iii) Crossover rate: 90%	(iii) Crossover: uniform
(iv) Mutation rate: 10%	(iv) Mutation: uniform

TABLE 6: High concentration: classification errors using 176 variables.

	Training (86 samples)	Validation (44 samples)	Commercial (21 samples)
PLS	11	5	1
SIMCA	15	12	1
Potential functions	4	6	0
ANN	0	1	0
SVM	0	39	21

TABLE 7: Low concentration: classification errors using 176 variables.

	Training (134 samples)	Validation (39 samples)	Commercial (2 samples)
PLS	29	11	0
SIMCA	19	14	0
Potential functions	4	9	0
ANN	0	4	0
SVM	0	33	1

a classification of a datum with 20% juice is considered correct only if the SVM that discriminate the category of 20% allocate it to that category, while the remaining defined SVM (each for the discrimination of one category) always assign the datum to the class of "other categories" (among which there is one of 20%). If, on the contrary, the sample is incorrectly characterized by a SVM, then the end result will be regarded as wrong.

As mentioned earlier, the results achieved in this test using the SVM are not satisfactory. Undoubtedly, any of the techniques employed for performing the classification (ANN, PLS, ...) achieve better results than the SVM.

Previous studies, based on the use of a tool widely employed in the study of chemical data as Procrustes rotation [70], have allowed establishing that the information provided by two of the original variables would be enough to categorize the data. The existence of overinformation and the difficulty that this involves when generalizing may be some of the causes of the poor results provided by SVM. Thus, selecting the most important variables from their original set we may obtain similar or better results than using all the data. This is due to the fact that, in many cases, not all the original variables have the same influence, or they contain redundant information, when determining the actual content of juice in a sample.

Therefore, a second criterion of comparison or a reference model is established, in this case one consisting of the classifications made only from the two significant variables

TABLE 8: Classification errors using variables selected by Procrustes rotation. High concentration.

	Training (86 samples)	Validation (44 samples)	Commercial (21 samples)
PLS	6	6	1
SIMCA	44	27	8
Potential functions	5	6	0
ANN	6	8	0
SVM	5	3	3

TABLE 9: Classification errors using variables selected by Procrustes rotation. Low concentration.

	Training (134 samples)	Validation (39 samples)	Commercial (2 samples)
PLS	29	9	0
SIMCA	36	17	0
Potential functions	29	13	0
ANN	28	17	2
SVM	13	4	1

previously selected by Procrustes (specifically, the variable 94 and the variable 95).

The results for each of these models are shown in Tables 8 and 9.

In this case, the proper operation of SVM is shown with a smaller version of the data set provided by IR spectroscopy, reducing the number of errors obtained so far in other studies. Therefore, its use as a measure of goodness seems feasible when guiding the implementation process of variable selection by means of GAs.

As previously mentioned, the initial goal consists of determining by GAs a series of sets obtained from the original variables that allow the correct classification of a sample. Furthermore, by not providing a single solution—as does Procrustes Rotation—further information is provided to the expert about the importance of different regions of the spectrum. First, the GA is configured to generate individuals made up of only 2 genes, each of them representing each of the variables to be considered for performing the classification.

In previous studies [13, 17, 67, 71], due to the versatility of ANNs, this technique has been chosen as the evaluation function used by the GA. Thus, the GA selects variables considered as *significant*, which will be provided as inputs to an ANN that is responsible for classifying the samples according to their amount of juice. The MSE obtained in the training process of the ANN is used as fitness to determine the goodness of each individual gene.

Tables 10 and 11 show the best results obtained with this technique after various tests and configurations. We can observe that the results improve significantly compared to those obtained when classification is made according to the variables selected by Procrustes Rotation.

However, this approach raises a problem. Due to random initialization process of an ANN, the obtained results are

TABLE 10: Classification Errors using GA + ANN: high concentration.

Selected variables	High concentrations		Commercial (21 samples)
	Train (86 samples)	Test (44 samples)	
[52, 141]	4	8	1
[102, 129]	10	14	2
[23, 67]	5	16	1
[18, 120]	3	12	2

TABLE 11: Classification errors using GA + ANN: low concentration.

Selected variables	Low concentrations		Commercial (2 samples)
	Train (134 samples)	Test (39 samples)	
[52, 141]	19	17	2
[102, 129]	10	14	2
[23, 67]	5	16	1
[18, 120]	3	12	2

hardly repeatable. To solve this problem, this paper proposes the use of SVM as a fitness function employed to guide the search for the GA. On this basis, we propose two approaches for defining the fitness function. On the one hand, the training mean squared error (MSETrain) is used as a fitness function, while on the other hand, the sum between the training MSE and the validation MSE (MSETrain + MSETest) is used as a fitness function.

Depending on the selected option as a fitness function, we have obtained the results shown in Tables 12 and 13, which include the four best solutions reached during the execution of the hybrid system using the two possibilities raised here as a fitness function. They show the predictive variables, as well as the errors made during the training and validation phases. As it can be seen by analyzing the results, regardless of the fitness function chosen, the obtained results are not only improved when the classification is done starting from the variables selected by Procrustes, but they are comparable to those obtained from the initial data set (unlike other techniques). In addition, compared to the use of ANN, we can count on the advantage offered by the repeatability of results.

Comparing these results with those previously shown (Tables 10 and 11), we can observe that the results in the training phase are very similar, regardless of the use of ANN or SVM in the evaluation function. However, the results obtained by SVM on the validation and testing data significantly improve the ones obtained with ANN, coming to the conclusion that greater generalization ability is achieved.

On the other hand, according to the results presented in Tables 12 and 13, one can sense that the use of the sum of both MSE (MSETrain + MSETest) as a fitness function of the hybrid system is able to reduce the number of errors made during the classification. The graph shown in Figure 7 performs a comparison between the two implemented functions. The data displayed correspond to the average of errors made during the SVM classification of all the executions carried out

TABLE 12: Classification errors using GA + SVM: high concentration.

Selected variables	Train (86 samples)	Test (44 samples)	Commercial (21 samples)
MSE train			
[4, 69]	5	5	2
[81, 84]	9	5	2
[68, 28]	4	6	2
[91, 101]	4	6	1
MSE train + MSE test			
[9, 105]	5	3	3
[9, 113]	5	4	9
[34, 101]	4	5	11
[86, 104]	5	5	4

TABLE 13: Classification errors using GA + SVM: low concentration.

Selected variables	Train (134 samples)	Test (39 samples)	Commercial (2 samples)
MSE train			
[1, 107]	15	5	1
[5, 91]	16	5	1
[2, 107]	17	5	1
[11, 97]	14	6	1
MSE train + MSE test			
[5, 73]	13	4	1
[27, 90]	14	4	1
[2, 94]	15	4	1
[27, 88]	14	5	2

with the hybrid system. A slight superiority is observed when using the sum of both MSE as a fitness function, especially in the validation phase.

7. Conclusions

This paper has proposed the development of a hybrid model combining GAs and SVM. The aim pursued by using this approach is the identification, from the initial data set, of the smallest number of variables possible that allow determining optimally the amount of juice within a sample. To this end, the method proposed herein uses a GA that employs SVM as a fitness function. This would lead to the generation of a fast, effective model that is capable of reproducing the data obtained.

With the aim of demonstrating the effectiveness of the proposed system, a series of experiments have been raised, with their corresponding validation tests. Observing the values obtained, we can come to the conclusion that the system improves significantly the results obtained so far with other classification methods.

As noted above, SVM are useful and effective when performing prediction data with a margin of error, previously cited as ξ . This evaluation mechanism has the undeniable advantage of the repeatability of results. Thus, each set of vari-

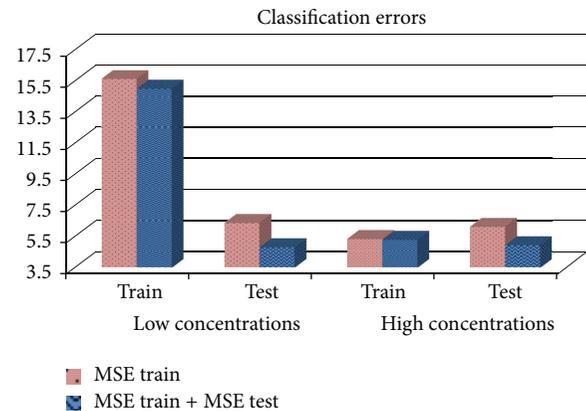


FIGURE 7: Classification errors according to the used fitness function.

ables will always have the same prediction system associated with the same fitness level, contrary to what would happen in the case of using ANN as a measure of goodness for genetic individuals.

Acknowledgments

This work is supported by the following projects: General Directorate of Research, Development and Innovation of Xunta de Galicia Ref. 07REM001CT, Spanish Ministry of Economy and Competitiveness Ref. CGL2012-34688, the Centre for Industrial Technological Development (CDTI) Ref. IDI 20120575, “Ibero-American Network of the Nano-Bio-Info-Cogno Convergent Technologies”, Ibero-NBIC Network (209RT-0366) funded by CYTED (Spain), “Development of new image analysis techniques in 2D Gel for biomedical research” (Ref. 10SIN105004PR) funded by Xunta de Galicia, RD07/0067/0005 funded by the Carlos III Health Institute, and “Galician Network for Colorectal Cancer Research” (REGICC, Ref. 2009/58), from the General Directorate of Scientific and Technologic Promotion of the Galician University System of Xunta de Galicia.

References

- [1] I. Guyon, A. Elisseeff, and L. P. Kaelbling, “An introduction to variable and feature selection,” *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.
- [2] P. Lewis, “The characteristic selection problem in recognition systems,” *IRE Transactions on Information Theory*, vol. 8, pp. 171–178, 1962.
- [3] G. S. Sebestyen, *Decision-Making Processes in Pattern Recognition*, ACM Monograph Series, 1962.
- [4] Y. Saeys, I. Inza, and P. Larrañaga, “A review of feature selection techniques in bioinformatics,” *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.
- [5] F. Alonso-Atienza, J. L. Rojo-Álvarez, A. Rosado-Muñoz, J. J. Vinagre, A. García-Alberola, and G. Camps-Valls, “Feature selection using support vector machines and bootstrap methods for ventricular fibrillation detection,” *Expert Systems with Applications*, vol. 39, no. 2, pp. 1956–1967, 2012.

- [6] C. Fernandez-Lozano, J. A. Seoane, M. Gestal Pose, T. R. Gaunt, and C. Campbell, "Texture classification using kernel-based techniques," in *International Work Conference on Artificial Neural Network*, G. J. Ignacio Rojas and J. Cabestany, Eds., pp. 427–434, Springer, Puerto de la Cruz, Spain, 2013.
- [7] I. T. Jolliffe, *Principal Component Analysis*, Springer, New York, NY, USA, 2 edition, 2002.
- [8] M. Barker and W. Rayens, "Partial least squares for discrimination," *Journal of Chemometrics*, vol. 17, no. 3, pp. 166–173, 2003.
- [9] R. Rosipal and N. Krämer, "Overview and recent advances in partial least squares," in *Proceedings of the International Conference on Subspace, Latent Structure and Feature Selection*, pp. 34–51, Springer, Bohinj, Slovenia, 2006.
- [10] Z. Ramadan, D. Jacobs, M. Grigorov, and S. Kochhar, "Metabolic profiling using principal component analysis, discriminant partial least squares, and genetic algorithms," *Talanta*, vol. 68, no. 5, pp. 1683–1691, 2006.
- [11] F. Tan, X. Feng, Z. Fang, M. Li, Y. Guo, and L. Jiang, "Prediction of mitochondrial proteins based on genetic algorithm—partial least squares and support vector machine," *Amino Acids*, vol. 33, no. 4, pp. 669–675, 2007.
- [12] J. A. Freeman and D. M. Skapura, *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley, Reading, Mass, USA, 1991.
- [13] M. Gestal, M. P. Gómez-Carracedo, J. M. Andrade et al., "Selection of variables by genetic algorithms to classify apple beverages by artificial neural networks," *Applied Artificial Intelligence*, vol. 19, no. 2, pp. 181–198, 2005.
- [14] C. M. Bishop, *Neural Networks For Pattern Recognition*, Oxford university press, New York, NY, USA, 1995.
- [15] C. Fernandez-Lozano, J. A. Seoane, P. Mesejo, Y. S. G. Nashed, S. Cagnoni, and J. Dorado, "2D-PAGE Texture classification using support vector machines and genetic algorithms," in *Proceedings of the 4th International Conference on Bioinformatics Models, Methods and Algorithms*, pp. 5–14, Scitepress, 2013.
- [16] C. Fernandez-Lozano, J. A. Seoane, M. Gestal Pose, D. Rivero, J. Dorado, and A. Pazos, "A texture-based classification method for proteins in two-dimensional electrophoresis gel images," in *Proceedings of the International Conference on Computer Vision Theory and Applications VISSAP*, pp. 401–404, Scitepress, Barcelona, Spain, 2013.
- [17] M. P. Gómez-Carracedo, M. Gestal, J. Dorado, and J. M. Andrade, "Chemically driven variable selection by focused multimodal genetic algorithms in mid-IR spectra," *Analytical and Bioanalytical Chemistry*, vol. 389, no. 7-8, pp. 2331–2342, 2007.
- [18] K.-Y. Chen and C.-H. Wang, "Support vector regression with genetic algorithms in forecasting tourism demand," *Tourism Management*, vol. 28, no. 1, pp. 215–226, 2007.
- [19] P.-F. Pai and W.-C. Hong, "Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms," *Electric Power Systems Research*, vol. 74, no. 3, pp. 417–425, 2005.
- [20] K. K. Kandaswamy, G. Pugalenth, S. Möller et al., "Prediction of apoptosis protein locations with genetic algorithms and support vector machines through a new mode of pseudo amino acid composition," *Protein and Peptide Letters*, vol. 17, no. 12, pp. 1473–1479, 2010.
- [21] D.-K. Kang and M.-J. Kim, "Performance enhancement of SVM ensembles using genetic algorithms in bankruptcy prediction," in *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE '10)*, pp. V2154–V2158, August 2010.
- [22] X. Chen and L. Wu, "Nonlinear demodulation and channel coding in EBPSK scheme," *The Scientific World Journal*, vol. 2012, Article ID 180469, 7 pages, 2012.
- [23] Y. F. Huang and S. Y. Chen, "Extracting physicochemical features to predict protein secondary structure," *The Scientific World Journal*, vol. 2013, Article ID 347106, 8 pages, 2013.
- [24] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.
- [25] V. N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, Chichester, UK, 1998.
- [26] S.-H. Min, J. Lee, and I. Han, "Hybrid genetic algorithms and support vector machines for bankruptcy prediction," *Expert Systems with Applications*, vol. 31, no. 3, pp. 652–660, 2006.
- [27] C.-H. Wu, G.-H. Tzeng, Y.-J. Goo, and W.-C. Fang, "A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy," *Expert Systems with Applications*, vol. 32, no. 2, pp. 397–408, 2007.
- [28] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.
- [29] E. Huerta, B. Duval, and J. K. Hao, "A hybrid GA/SVM approach for gene selection and classification of microarray data," in *Applications of Evolutionary Computing*, F. Rothlauf, J. Branke, S. Cagnoni et al., Eds., pp. 34–44, Springer, Berlin, Germany, 2006.
- [30] R. Venegas, "Academic text classification based on lexical-semantic content," *Revista Signos*, vol. 40, no. 63, pp. 239–271, 2007.
- [31] C. A. Donís, E. Valencia Morales, and C. Morell Pérez, "Support vector machine model for regression applied to the estimation of the creep rupture stress in ferritic steels," *Revista Facultad de Ingeniería Universidad de Antioquia*, no. 47, pp. 53–58, 2009.
- [32] L. Pérez, J. Mora, and J. Bedoya, "A linear approach to determining an SVM-based fault locator's optimal parameters," *Ingeniería e Investigación*, vol. 29, pp. 76–81, 2009.
- [33] M. Tong, K. H. Liu, C. Xu, and W. Ju, "An ensemble of SVM classifiers based on gene pairs," *Computers in Biology and Medicine*, vol. 43, pp. 729–737, 2013.
- [34] Z. Wang, Y. H. Shao, and T. R. Wu, "A GA-based model selection for smooth twin parametric-margin support vector machine," *Pattern Recognition*, vol. 46, pp. 2267–2277, 2013.
- [35] K. J. Won, C. Saunders, and A. Prügel-Bennett, "Evolving fisher kernels for biological sequence classification," *Evolutionary Computation*, vol. 21, pp. 83–105, 2013.
- [36] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.
- [37] T. İlhan and G. Tezel, "A genetic algorithm-support vector machine method with parameter optimization for selecting the tag SNPs," *Journal of Biomedical Informatics*, vol. 46, pp. 328–340, 2013.
- [38] R. J. Ma, N. Y. Yu, and J. Y. Hu, "Application of particle swarm optimization algorithm in the heating system planning problem," *The Scientific World Journal*, vol. 2013, Article ID 718345, 11 pages, 2013.
- [39] Y. Zhang, S. Wang, G. Ji, and Z. Dong, "An MR brain images classifier system via particle swarm optimization and kernel support vector machine," *The Scientific World Journal*, vol. 2013, Article ID 130134, 9 pages, 2013.

- [40] H. Ocak, "A medical decision support system based on support vector machines and the genetic algorithm for the evaluation of fetal well-being," *Journal of Medical Systems*, vol. 37, article 9913, 2013.
- [41] M. S. Uzer, N. Yilmaz, and O. Inan, "Feature selection method based on artificial bee colony algorithm and support vector machines for medical datasets classification," *The Scientific World Journal*, vol. 2013, Article ID 419187, 10 pages, 2013.
- [42] L. Li, H. Chen, C. Liu et al., "A robust hybrid approach based on estimation of distribution algorithm and support vector machine for hunting candidate disease genes," *The Scientific World Journal*, vol. 2013, Article ID 393570, 7 pages, 2013.
- [43] M. Burton, M. Thomassen, Q. Tan, and T. A. Kruse, "Gene expression profiles for predicting metastasis in breast cancer: a cross-study comparison of classification methods," *The Scientific World Journal*, vol. 2012, Article ID 380495, 11 pages, 2012.
- [44] M. P. Poland, C. D. Nugent, H. Wang, and L. Chen, "Genetic algorithm and pure random search for exosensor distribution optimisation," *International Journal of Bio-Inspired Computation*, vol. 4, pp. 359–372, 2012.
- [45] J. Muñuzuri, P. Cortés Achedad, M. Rodríguez, and R. Grosso, "Use of a genetic algorithm for building efficient choice designs," *International Journal of Bio-Inspired Computation*, vol. 4, pp. 27–32, 2012.
- [46] D. Koller and M. Sahami, "Toward optimal feature selection," in *Citeseer*, pp. 284–292, 1996.
- [47] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.
- [48] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 1975.
- [49] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.
- [50] S. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, New York, NY, USA, 1999.
- [51] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, 1975.
- [52] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognition Letters*, vol. 10, no. 5, pp. 335–347, 1989.
- [53] M. Kudo and J. Sklansky, "A comparative evaluation of medium- and large-scale feature selectors for pattern classifiers," *Kybernetika*, vol. 34, no. 4, pp. 429–434, 1998.
- [54] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, "Genetic programming III: darwinian invention and problem solving [book review]," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 251–253, 1999.
- [55] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433–460, 1950.
- [56] A. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 11, pp. 601–617, 1967.
- [57] M. Mitchell, *An Introduction to Genetic Algorithms*, The MIT press, 1998.
- [58] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Urbana*, vol. 51, pp. 61801–62996, 1991.
- [59] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the 2nd International Conference on Genetic Algorithms on Genetic Algorithms and their Application*, pp. 41–49, L. Erlbaum Associates, 1987.
- [60] K. Deb, "Multi-objective genetic algorithms: problem difficulties and construction of test problems," *Evolutionary computation*, vol. 7, no. 3, pp. 205–230, 1999.
- [61] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, no. 1–3, pp. 131–159, 2002.
- [62] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press, Cambridge, Mass, USA, 2002.
- [63] V. Vapnik and A. Chervonenkis, "The necessary and sufficient conditions for consistency in the empirical risk minimization method," *Pattern Recognition and Image Analysis*, vol. 1, pp. 283–305, 1991.
- [64] M. Wall, "GAlib: a C++ library of genetic algorithm components," Mechanical Engineering Department, Massachusetts Institute of Technology, 1996, <http://lancet.mit.edu/ga/>.
- [65] M. Sewell, "WinSVM: a windows implementation of a support vector machine," 2005, <http://winsvm.martinsewell.com/>.
- [66] U. Maulik and S. Bandyopadhyay, "Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 5, pp. 1075–1081, 2003.
- [67] M. Gestal, A. Cancela, J. Andrade, and M. Gomez-Carracedo, "Several approaches to variable selection by means of genetic algorithms," in *Intelligent Information Technologies: Concepts, Methodologies, Tools and Applications*, V. Sugaraman, Ed., Information Science Reference, pp. 274–292, Hershey, New York, NY, USA, 2007.
- [68] M. Gestal, M. P. Gómez-Carracedo, J. M. Andrade et al., "Classification of apple beverages using artificial neural networks with previous variable selection," *Analytica Chimica Acta*, vol. 524, no. 1-2, pp. 225–234, 2004.
- [69] D. R. Heldman and D. B. Lund, *Handbook of Food Engineering*, CRC Press, New York, NY, USA, 2007.
- [70] W. J. Krzanowski, *Principles of Multivariate Analysis: A User's Perspective*, Oxford University Press, New York, NY, USA, 2000.
- [71] M. P. Gómez-Carracedo, M. Gestal, J. Dorado, and J. M. Andrade, "Linking chemical knowledge and genetic algorithms using two populations and focused multimodal search," *Chemo-metrics and Intelligent Laboratory Systems*, vol. 87, no. 2, pp. 173–184, 2007.

Research Article

A Modified Dynamic Evolving Neural-Fuzzy Approach to Modeling Customer Satisfaction for Affective Design

C. K. Kwong,¹ K. Y. Fung,¹ Huimin Jiang,¹ K. Y. Chan,² and Kin Wai Michael Siu³

¹ Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong

² Department of Electrical and Computer Engineering, Curtin University of Technology, Perth, WA 6845, Australia

³ School of Design, The Hong Kong Polytechnic University, Kowloon, Hong Kong

Correspondence should be addressed to C. K. Kwong; c.k.kwong@polyu.edu.hk

Received 24 September 2013; Accepted 21 October 2013

Academic Editors: Z. Cui and X. Yang

Copyright © 2013 C. K. Kwong et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Affective design is an important aspect of product development to achieve a competitive edge in the marketplace. A neural-fuzzy network approach has been attempted recently to model customer satisfaction for affective design and it has been proved to be an effective one to deal with the fuzziness and non-linearity of the modeling as well as generate explicit customer satisfaction models. However, such an approach to modeling customer satisfaction has two limitations. First, it is not suitable for the modeling problems which involve a large number of inputs. Second, it cannot adapt to new data sets, given that its structure is fixed once it has been developed. In this paper, a modified dynamic evolving neural-fuzzy approach is proposed to address the above mentioned limitations. A case study on the affective design of mobile phones was conducted to illustrate the effectiveness of the proposed methodology. Validation tests were conducted and the test results indicated that: (1) the conventional Adaptive Neuro-Fuzzy Inference System (ANFIS) failed to run due to a large number of inputs; (2) the proposed dynamic neural-fuzzy model outperforms the subtractive clustering-based ANFIS model and fuzzy *c*-means clustering-based ANFIS model in terms of their modeling accuracy and computational effort.

1. Introduction

Today, manufacturers face a highly competitive environment, as marketing becomes global and large amount of choices are available in the market. Customers consider functionality, ease-of-use, and reliability as product requirements, they but equally consider intangible and emotional aspects such as the novelty, personality, aesthetics, and style of products [1, 2]. Therefore, design for performance and design for usability can no longer guarantee a competitive advantage [3]. Products with highly affective designs can attract customers, influence their choices and preferences, and gain their loyalty while giving them the joy of use [4–6]. Thus, affective design is essential to identify, measure, analyze, and understand the relationships between the affective values of the customer domain and the perceptual design parameters in the design domain [7, 8]. Furthermore, affective design provides

decision support for the design optimization process, such that appealing products can be successfully developed by satisfying the emotional needs of the target customers [9, 10].

Affective design in industries still heavily relies on the experience and intuition of designers, who may not fully recognize or perceive the affective values of their customers. Thus, cognitive gaps in design appreciation always exist between designers and customers [1, 11]. Although the customers' emotional and implicit needs can be obtained and collected from focus groups and in-depth interviews, the collected information is subjective and qualitative because the sampling size is limited. Given the importance of affective design, previous studies attempted to adopt quality function deployment and robust design approaches to affective design [12, 13]. However, these approaches are incapable of dealing with the subjective aspects of affective design [14].

Affective relationships are mostly nonlinear because of the inconsistencies in the customer survey data and their nonlinear relationships with design attributes [13]. Given the fuzzy and non-linear behavior of affective modeling, statistical methods based on the assumptions of simple linear relationships and normal distributions may not be capable of modeling affective values [15]. Studies on affective design have attempted to apply computational intelligence techniques, specifically artificial neural networks, to deal with the ambiguity of affective data and the nonlinearity of affective modeling [16]. Although these approaches are capable of modeling the non-linear relationships between parameters, they generate models that are implicit, that is, black-box models. Thus, the analysis of the behavior of these relationships remains difficult.

A hybrid approach, named the neural fuzzy (NF) model, has been used to model the relationships between design attributes and affective responses of customers [17–19]. The NF model combines the capability of fuzzy logic to generate the linguistic representation of knowledge and the adaptive learning capability of artificial neural networks for the automatic generation and optimization of a fuzzy inference system. The model is formed by complex networks that consist of significant numbers of hidden nodes, linkage weights, and fuzzy membership functions, among others. Based on these complex networks, the model effectively captures the highly non-linear relationships between the design attributes of a new product and its satisfaction of affective dimensions. Furthermore, the networks of the NF models overcome the limitations of artificial neural networks by obtaining explicit information that is useful for affective design.

A large number of design attributes may need to be studied in an affective design. Furthermore, the initialization of customer satisfaction models is based on a limited amount of customer surveys, which are collected by a focus group. When new survey data is collected, the customer satisfaction model will need to be updated with respect to the data from the new survey. The network structures of NF models tend to be too complex and too cumbersome. Thus, the effectiveness of these networks is questionable for modeling customer affections that have multiple dimensions and require updates. In this paper, a modified dynamic evolving neural-fuzzy inference systems (DENFIS) approach is proposed to overcome the limitations of the existing NF approaches with respect to modeling affective relationships with high dimensions and updating models with different batches of survey data. The effectiveness of the proposed approach is demonstrated using a case study of affective design for mobile phones.

2. Dynamic Evolving Neurofuzzy Inference Systems (DENFIS)

Adaptive neural fuzzy inference systems (ANFIS) have been used [20] to model customer satisfaction, where the inputs and the outputs of the NF models represent the design attributes and the satisfaction values of the new product, respectively. However, the fixed structure generated by

ANFIS hinders its adaption to new data sets. In addition, ANFIS cannot be used for high dimensional problems [21], because the complexity of the dynamic NF models generated by ANFIS typically grows exponentially with the number of input attributes. The number of rule nodes generated is equal to $\prod_{j=1}^J N_{MF,j}$, where $N_{MF,j}$ is the number of membership functions (MF) for the j th design attribute and J is the number of design attributes of the customer affection model. For instance, the structure of the NF model contains 6,561 ($= 3^8$) rule nodes if there are eight inputs, and each input contains three MFs. Excessive inputs and MFs could cause computational difficulties because of the insufficient machine memory [22]. Therefore, ANFIS is unsuitable for generating customer affection models with high dimensions of inputs. The DENFIS approach is considered in this research to develop dynamic NF models for modeling customer satisfaction for affective design.

2.1. Conventional DENFIS. The conventional DENFIS can be used to generate a streamlined model in NF form for affective design, which cannot be achieved by the modified DENFIS [23, 24], in which a reasonable number of fuzzy rule-based models are generated. Conventional DENFIS is a fast, one-pass, online incremental clustering algorithm for generating NF models for affective design; this system generates input spaces using the evolving clustering method [23, 25]. The survey data on affective values are partitioned by the system into clusters, which are specified by their centers and radii. Thus, the number of created clusters is self-determined based on a threshold value, D_{thr} , which controls the maximum distance between a data point of a cluster and the cluster center and acts as a constraint for updating the radii of the clusters.

In the clustering process, the evolving clustering method starts by initializing the first cluster C_1 with the first data survey data regarding affective values (Z_1); that is, $k = 1$, where k is the number of clusters created by the evolving clustering method. When new data on affective values (Z_n) is presented, its distance to the cluster centre (c_k), of each existing cluster (C_k) is computed, where $k = 1, 2, \dots, K$, and $n > 1$. Therefore, the distance D' between the n th survey data (Z_n) and its closest cluster (C') can similarly be found. The new point belongs to the cluster C' , and an update does not occur when $D' < r'$, where r' denotes the current radius of the cluster C' . If $D' < D_{thr}$, the centre c' and radius r' of the closest cluster C' are updated [23]. Otherwise, a new cluster C_{K+1} is created at the n th survey data Z_n , such that $K = K + 1$. Consequently, the newly collected survey data can be dynamically grouped by applying the evolving clustering method. This step facilitates the formulation of the optimal number of local models for effective modeling substantial data.

After determining the partitions of the input space using the clustering process, a set of fuzzy rules are then created to represent the cluster. In the rule antecedents, fuzzy numbers are used to represent the design attributes of the new product. In the rule consequents, first-order linear models are used to

represent the affective values of the new product. The fuzzy rules are expressed as follows:

if x_1 is MF_{11} and x_2 is MF_{21} and ... and x_j is MF_{j1} ,

then y is $f_1(x_1, x_2, \dots, x_j)$;

if x_1 is MF_{12} and x_2 is MF_{22} and ... and x_j is MF_{j2} ,

then y is $f_2(x_1, x_2, \dots, x_j)$

⋮

if x_1 is MF_{1K} and x_2 is MF_{2K} and ... and x_j is MF_{jK} ,

then y is $f_K(x_1, x_2, \dots, x_j)$,

(1)

where x_j is MF_{jk} , $j = 1, 2, \dots, J$, and $k = 1, 2, \dots, K$, MF_{jk} denotes a Gaussian MF used for the fuzzification of the design attributes of new products, J is the number of design attributes of the new products, K is the number of clusters generated by the evolving clustering method, and $f_k(x_1, x_2, \dots, x_j)$ denotes the function of a first-order linear model for affective design.

The input space of each fuzzy rule is fuzzified based on the clustering results of the evolving clustering method. For each cluster, the cluster centre c_k and radius r_k are assigned to the Gaussian MF, which is defined as

$$\text{Gaussian MF}_{jk} = \alpha \cdot \exp \left[-\frac{(x_j - c_{jk})^2}{2r_{jk}^2} \right]. \quad (2)$$

For the rule consequent of each fuzzy rule, a first-order linear model is developed based on a weighted recursive least square, which uses a forgetting factor in the conventional DENFIS [23, 25]. To represent the satisfaction values of affective dimensions of the new product, the first-order linear model is represented by the following linear polynomial:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_j x_j. \quad (3)$$

For each rule, the u data instances $\{[x_1^i, x_2^i, \dots, x_j^i, \dots, x_j^i], y_i\}$ with $i = 1, 2, \dots, u$, belong to the same cluster. These instances are intended to be the initial training data for the recursive least squares. For the initial linear model, the regression coefficients of $\beta = [\beta_0, \beta_1, \beta_2, \beta_j, \beta_j]^T$ are calculated by applying the weighted least squares estimator using the following formulas:

$$\begin{aligned} \mathbf{P} &= (\mathbf{A}^T \mathbf{W} \mathbf{A})^{-1}, \\ \beta &= \mathbf{P} \mathbf{A}^T \mathbf{W} \mathbf{y}, \end{aligned} \quad (4)$$

where

$$\mathbf{A} = \begin{pmatrix} 1 & x_1^1 & x_2^1 & \dots & x_j^1 & \dots & x_j^1 \\ 1 & x_1^2 & x_2^2 & \dots & x_j^2 & \dots & x_j^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_1^i & x_2^i & \dots & x_j^i & \dots & x_j^i \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_1^u & x_2^u & \dots & x_j^u & \dots & x_j^u \end{pmatrix}, \quad (5)$$

$$\mathbf{y} = [y_1 \ y_2 \ \dots \ y_i \ \dots \ y_u]^T,$$

$$\mathbf{W} = \begin{pmatrix} \omega_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \omega_2 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \omega_i & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & \omega_u \end{pmatrix}$$

of which ω_i is $(1 - D_i)$ with $i = 1, 2, \dots, u$, and D_i is the distance between the i th survey data regarding affective values and the corresponding cluster centre.

The regression coefficient β and inverse matrix \mathbf{P} are used as the initial values for the future recursive calls, where β_u and \mathbf{P}_u represent the regression coefficient and the inverse matrix, respectively, at the u th iteration using the least squares. When new survey data regarding affective values is fed, β_{u+1} is updated based on the following equations:

$$\begin{aligned} \beta_{u+1} &= \beta_u + \omega_{u+1} \mathbf{P}_{u+1} a_{u+1} (y_{u+1} - a_{u+1}^T \beta_u), \\ \mathbf{P}_{u+1} &= \frac{1}{\lambda} \left(\mathbf{P}_u - \frac{\omega_{u+1} \mathbf{P}_u a_{u+1} a_{u+1}^T \mathbf{P}_u}{\lambda + a_{u+1}^T \mathbf{P}_u a_{u+1}} \right), \end{aligned} \quad (6)$$

where λ is a forgetting factor such that $0 < \lambda \leq 1$, and

$$a_{u+1}^T = [1 \ x_1^{u+1} \ x_2^{u+1} \ \dots \ x_j^{u+1} \ \dots \ x_j^{u+1}]. \quad (7)$$

Finally, a dynamic NF model can be created based on the set of fuzzy rule-based models that are generated using the evolving clustering method and the recursive least squares. As an illustration, Figure 1 shows a typical structure of a five-layer NF model with two design attributes for a new product, x_1 and x_2 , as well as one affective dimension for the new product, y . Subsequently, the back propagation (BP) algorithm can be used to further optimize the fuzzy MF of the dynamic NF model for affective design.

2.2. Modified DENFIS for Affective Design. It can be noted that the conventional DENFIS can only generate a dynamic NF model in the single-feed form, where only a single instance of survey data can be used to update the dynamic NF model. The influence of each data instance is typically decayed by applying the recursive least squares with a constant forgetting factor λ , as shown in Figure 2.

For affective design, some companies may conduct surveys several times over a period of time to obtain more

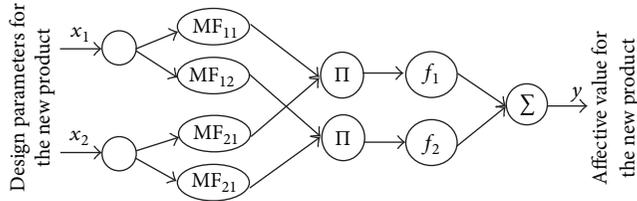


FIGURE 1: A five-layer structure of the dynamic NF model ($J = 2$, $K = 2$) for affective design.

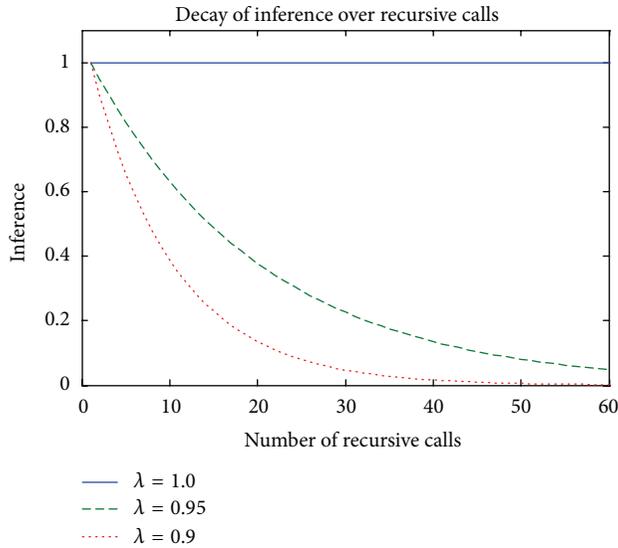


FIGURE 2: Decay of inference with a constant forgetting factor.

accurate data on customer affection towards the designs of products. The respondent evaluates the design profiles of N product samples. The design profile of the n th product sample can be represented as $X(n) = [x_1^n, x_2^n, \dots, x_j^n, \dots, x_j^n]$, where $n = 1, 2, \dots, N$. Subsequently, N customer affections are obtained from the survey which is participated in at time t . The affective data set collected at time t is given by

$$Y(t) = [y_1(t), y_2(t), \dots, y_N(t)], \quad (8)$$

where $y_n(t)$ denotes the affective rating acquired from the respondent towards the N product samples after conducting the survey at the t th period. Thus, the N survey data sets are available for updating the NF model for affective design.

A modified DENFIS is proposed to process a batch of N survey data sets for each update process, where the forgetting factor λ is a variable instead of a constant, such that the decay effect can be controlled by varying the value of λ . Typically, $\lambda < 1$ is used for the recursive least squares so the fresh data set can exert more influence than the previous data set during the recursive calls. By contrast, previous and current data sets are treated equally by the recursive least squares if $\lambda = 1$. The forgetting factor value is switched to update data sets from different periods of time when the modified DENFIS is performing the incremental learning process.

The modified DENFIS was developed to decrease the influence of survey data sets by batch over time. When

the survey data set $Y(t)$ is available to the model, its influence is greater than that of the previous data set $Y(t-1)$. During the incremental learning process, $y_1(t), y_2(t), \dots, y_N(t)$ are partitioned into clusters based on the evolving clustering method. For each cluster, the first-order model is updated by the data subset using the recursive least squares. When the first data instance of the subset, such as $y_1(t)$, is proceeded by new survey data, $\lambda < 1$ is set to exert the decay effect on previous data batches, which are compiled as the matrices β_{t-1} and P_{t-1} . β_{t-1} and P_{t-1} have been obtained from the previous recursive least squares in the previous data batch $Y(t-1)$. For the remaining data in the data subset, $\lambda = 1$ is set to suspend the decay during the training sequence from $y_1(t)$ to $y_N(t)$. Thus, the previous data batch fades out when the current data batch is processed with the same influence. This staircase decay is shown in Figure 3.

Figure 4 summarizes the architecture of the modified DENFIS for affective design, where the dynamic NF model enables faster incremental learning by applying the evolving clustering method and the recursive least squares. The modified DENFIS uses the recursive least squares with the variable forgetting factor to update the dynamic NF model by adapting the new data sets.

3. Case Study of Affective Design for Mobile Phones

A case study of the affective design for mobile phones is used to investigate the effectiveness of the proposed approach to modeling affective relationships. The case study mainly involves a survey and the implementation of the proposed approach for the affective design of mobile phones. The modified DENFIS was implemented using the MATLAB programming language.

The survey was conducted using questionnaires. The affective assessments of customers on 32 image samples of mobile phones were determined using four affective dimensions: "simplicity," "uniqueness," "high tech," and "handiness." The front and side views of the 32 mobile phone samples are presented in Figure 5. A total of 34 respondents filled out the questionnaires and indicated their feelings towards the product images of each sample using a five-point Likert scale, which is illustrated in Figure 6.

The morphological approach was adopted to define the design space of the product form for the mobile phones. The design composition and the possible design solutions with simple and graphical notations were feasibly depicted. Eight design attributes (from A_1 to A_8) were defined to describe the product forms of mobile phones, which included the top shape, bottom shape, function button shape, layout, length, width ratio, thickness, and border width. The first four design attributes are categorical, and the remaining four attributes are quantitative. The categorical attributes contain three to five options. The attributes and their options are listed in a design table (Table 1) for the product form of the mobile phones. Based on the design table, the design profile for each sample was identified, and the values of design attributes were measured.

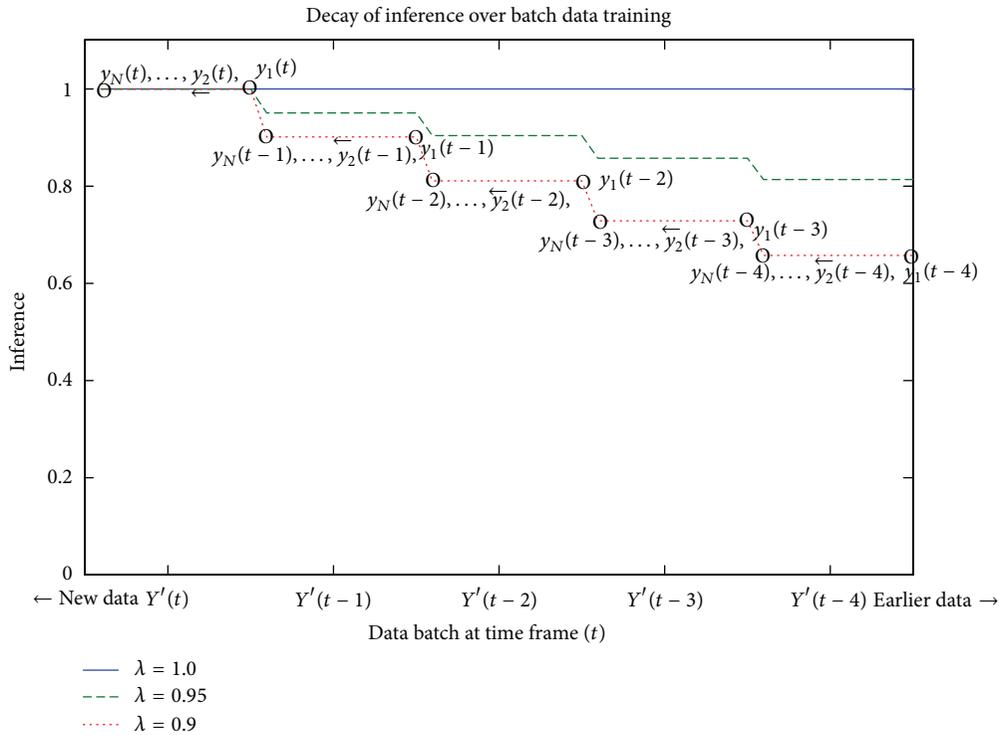


FIGURE 3: Decay of inference by the modified DENFIS.

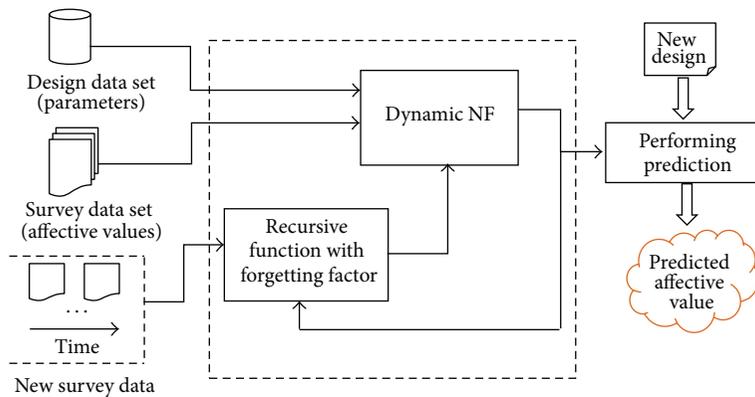


FIGURE 4: A dynamic NF model for affective design.

3.1. *Implementation of the Dynamic NF Model.* An initial dynamic NF model for modeling affective relationships was developed using the proposed DENFIS when the initial survey data sets were collected from the first seventeen respondents. The dynamic NF model was then sequentially updated using the survey data sets collected from the subsequent respondents. The forgetting factor $\lambda = 0.95$ was set to produce a slight decay of inference. Two dynamic NF models were developed; one involved BP training, whereas the other did not.

The structure of the dynamic NF model that was generated using the DENFIS is shown in Figure 7, where eight nodes in the input layer represent the eight design attributes of mobile phone design. Each node in the input layer links

with five MFs in the input layer, and the MFs are engaged to the five nodes in the rule layer. All the nodes in the three layers can be represented by a set of fuzzy rules that are formulated from the generated clusters of the evolving clustering method. Algorithm 1 presents these fuzzy rules, where each fuzzy rule can be considered as a local fuzzy model. Each local fuzzy model consists of fuzzy domains governed by fuzzy MFs for the design attributes, as shown in Figure 8. After inputting design attribute settings of mobile phones, the resulting local models were aggregated to determine satisfaction values of the affective dimensions for mobile phones.

3.2. *Prediction Performance of the Dynamic NF Models.* The leave-one-out cross-validation is performed to evaluate the

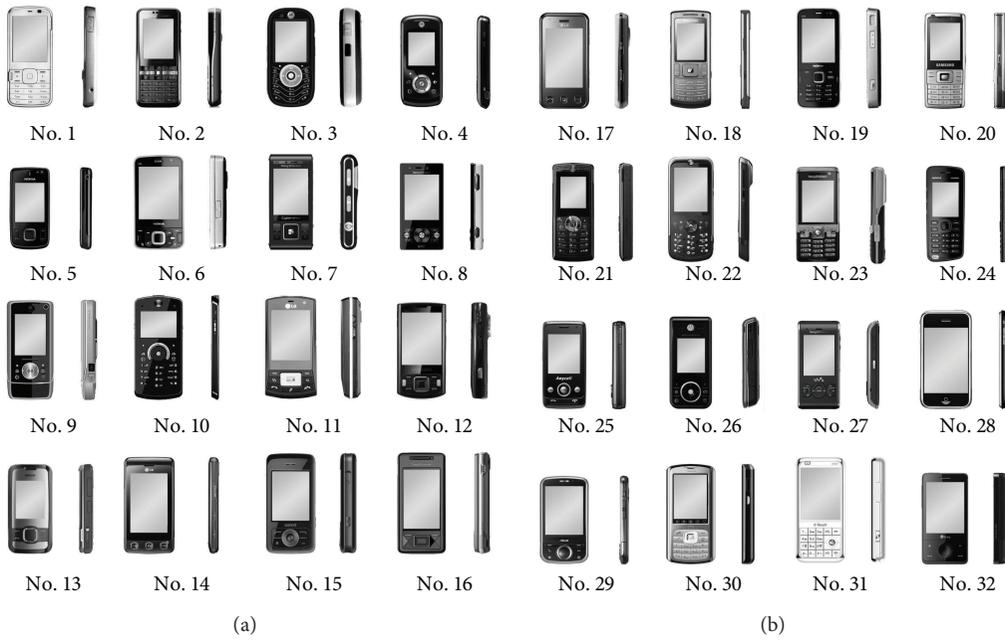


FIGURE 5: The 32 mobile phone image samples used in the case study.

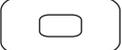
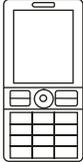
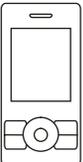


What does you feel about the shape of the mobile phone?

	1	2	3	4	5
(1) Simple or complex? (simple = 1, complex = 5)	<input type="checkbox"/>				
(2) Unique or general? (unique = 1, general = 5)	<input type="checkbox"/>				
(3) High tech or classic? (high tech = 1, classic = 5)	<input type="checkbox"/>				
(4) Handy or bulky? (handy = 1, bulky = 5)	<input type="checkbox"/>				

FIGURE 6: The questionnaire format for each mobile phone.

TABLE 1: Design table for the product form of the mobile phones.

Elements	Categorical attributes				
	Type 1	Type 2	Type 3	Type 4	Type 5
(1) Top shape (A_1)	 Line (x_{11})	 Arc (x_{12})	 Curve (x_{13})		
(2) Bottom shape (A_2)	 Line (x_{21})	 Arc (x_{22})	 Curve (x_{23})		
(3) Function button shape (A_3)	 Large round (x_{31})	 Small round (x_{32})	 Small squares (x_{33})	 Large squares (x_{34})	 Wide block (x_{35})
(4) Layout (A_4)	 Bar (x_{41})	 Slide (x_{42})	 Large screen (x_{43})		
	Quantitative attributes				
(5) Body length (A_5)					
(6) Body width (A_6)					
(7) Body thickness (A_7)					
(8) Border width (A_8)					

Fuzzy rule 1:
 If x_1 is MF₁₂ and x_2 is MF₂₁ and x_3 is MF₃₁ and x_4 is MF₄₁ and x_5 is MF₅₁ and x_6 is MF₆₃ and x_7 is MF₇₂ and x_8 is MF₈₃, Then $y_H = (0.7832 - 0.3457x_1 + 0.4648x_2 - 0.0777x_3, -0.1483x_4 - 0.5625x_5 + 0.1666x_6 + 0.1792x_7 - 0.4252x_8)$.

Fuzzy rule 2:
 If x_1 is MF₁₅ and x_2 is MF₂₄ and x_3 is MF₃₅ and x_4 is MF₄₂ and x_5 is MF₅₂ and x_6 is MF₆₁ and x_7 is MF₇₁ and x_8 is MF₈₂, Then $y_H = (0.6287 + 0.2538x_1 + 0.3483x_2 - 0.1493x_3 + 0.1085x_4 - 0.4197x_5 - 0.4555x_6 - 0.2626x_7 - 0.06333x_8)$.

Fuzzy rule 3:
 If x_1 is MF₁₁ and x_2 is MF₂₃ and x_3 is MF₃₄ and x_4 is MF₄₃ and x_5 is MF₅₄ and x_6 is MF₆₄ and x_7 is MF₇₄ and x_8 is MF₈₄, Then $y_H = (0.2227 + 0.009144x_1 - 0.1078x_2 + 0.03565x_3 + 0.1077x_4 + 0.1951x_5 + 0.1998x_6 + 0.08163x_7 + 0.004227x_8)$.

Fuzzy rule 4:
 If x_1 is MF₁₄ and x_2 is MF₂₅ and x_3 is MF₃₂ and x_4 is MF₄₄ and x_5 is MF₅₅ and x_6 is MF₆₂ and x_7 is MF₇₅ and x_8 is MF₈₁, Then $y_H = (0.2448 + 0.05979x_1 + 0.02159x_2, -0.04047x_3 + 0.2202x_4 + 0.07846x_5 - 0.01265x_6 - 0.01162x_7 + 0.1297x_8)$.

Fuzzy rule 5:
 If x_1 is MF₁₃ and x_2 is MF₂₂ and x_3 is MF₃₃ and x_4 is MF₄₅ and x_5 is MF₄₃ and x_6 is MF₆₅ and x_7 is MF₇₃ and x_8 is MF₈₅, Then $y_H = (0.3077 - 0.001267x_1 + 0.001715x_2 - 0.291x_3 - 0.1659x_4 - 0.3424x_5 + 0.1052x_6 + 0.496x_7 + 0.2649x_8)$.

Note: y_H is the predicted value with regard to the rating of handiness of mobile phone

ALGORITHM 1: A set of fuzzy rules engaged with the dynamic NF model for the affective design of mobile phones.

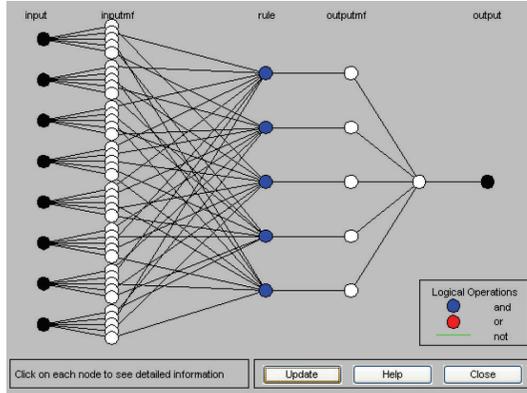


FIGURE 7: Structure of the developed dynamic NF model for the affective design of mobile phones.

prediction performance of the proposed dynamic NF model for the affective design of mobile phones. Data on the 32 mobile phone samples are randomly partitioned into training and test sets. The training set consists of 31 samples, while the test set contains only 1 sample. The cross-validation test involves all 32 folds, where each fold underwent the replacement of the training and test sets during the training process. Hence, each sample can be used as a test set only once.

The prediction performance of the two dynamic NF models that were generated by DENFIS (either with or without BP training) are compared with those generated by the common ANFIS approaches, namely, the subtractive clustering- (SC-) based ANFIS approach and the fuzzy c -means clustering- (FCM-) based ANFIS approach [22]. In this case study, a fully structured ANFIS was developed for the affective design. However, its training process could not be completely done because its structure generated by ANFIS approach contained more than 10935 hidden nodes, which was too complex and also required a large amount of computational memory for execution. When using the SC-based ANFIS approach and the FCM-based ANFIS approach, the hidden nodes are selectively built into the NF models based on the partitioning results of the SC and FCM clustering approaches, respectively. Therefore, a much less computational memory is required as compared with the fully structured ANFIS approach. Using the same training data sets, four NF models are generated based on the four approaches, namely, DENFIS with BP training, DENFIS without BP training, SC-based ANFIS, and FCM-based ANFIS. The generated NF models are then applied to estimate satisfaction values of the affective dimensions using the test data sets. Subsequently, the performances of the four NF models are compared based on the average root mean square error (RMSE) and the computational time, where the average RMSE is defined as follows:

$$\text{Average RMSE} = \frac{\sum \text{RMSE of test samples}}{\text{Total number of test samples in the cross-validation test}} \quad (9)$$

TABLE 2: Test performance of the NF models (simplicity).

Models	Average RMSE	Average computational time*/s
SC-based ANFIS	0.2874	0.6921
FCM-based ANFIS	0.166	0.6157
DENFIS without BP	0.147	0.4991
DENFIS with BP	0.1474	

* Average computational time per update of dynamic NF models is 0.01848 s.

TABLE 3: Test performance of the NF models (uniqueness).

Models	Average RMSE	Average computational time*/s
SC-based ANFIS	0.2859	0.692
FCM-based ANFIS	0.1086	0.606
DENFIS without BP	0.101	0.4876
DENFIS with BP	0.1143	

* Average computational time per update of dynamic NF models is 0.01806 s.

The results of the cross-validation tests for the four affective dimensions: “simplicity,” “uniqueness,” “high tech,” and “handiness,” are shown in Tables 2, 3, 4, and 5, respectively, whereas Figure 9 summarizes the results obtained by the four approaches. The test results for “simplicity,” “uniqueness,” and “high tech” reveal that the test errors when using the DENFIS without BP training are smaller than those obtained when using the FCM-based models. Moreover, the DENFIS with BP training outperforms the FCM-based ANFIS models in terms of its prediction performance. DENFIS with BP training significantly reduces the errors of the dynamic NF model for “high tech” and “handiness,” because BP training fine-tunes membership functions and improves the prediction accuracy of the dynamic NF models. The results shown in Tables 2 and 3 reveal that the errors of the NF models generated by DENFIS with BP training are slightly larger than those without BP training because overfitting may occur when BP training is used. However, the NF models that were generated by DENFIS with BP training obtain the best overall prediction performance as compared with the other three approaches.

Apart from the prediction performance, the test results show that the two DENFIS approaches are more computationally efficient than the other two ANFIS approaches. The computational time of the incremental training for the two DENFIS approaches is approximately 0.018 s, which is about 18% to 29% less than that of the two ANFIS approaches.

4. Conclusion

This paper proposes a dynamic evolving NF approach to modeling the relationships between design attributes and affective satisfaction which is based on collected customer survey data. The proposed approach is able to address the high nonlinearity and fuzziness of the modeling of the affective relationship, which cannot be addressed effectively by the commonly used statistical methods. The proposed model

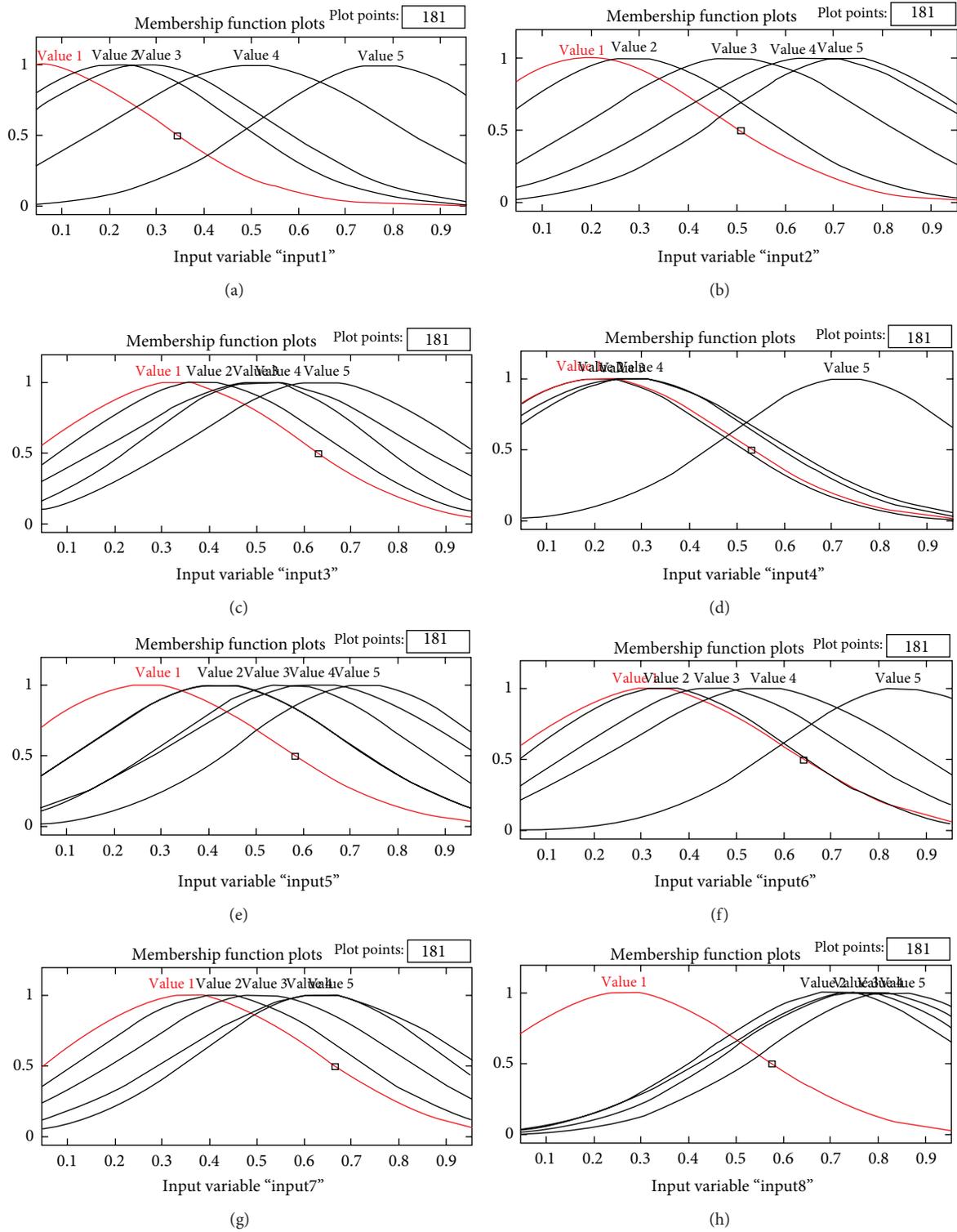


FIGURE 8: Fuzzy membership functions of the affections of mobile phone: (a) top shape, (b) bottom shape, (c) function button shape, (d) layout, (e) body length, (f) body thickness, and (g) border width.

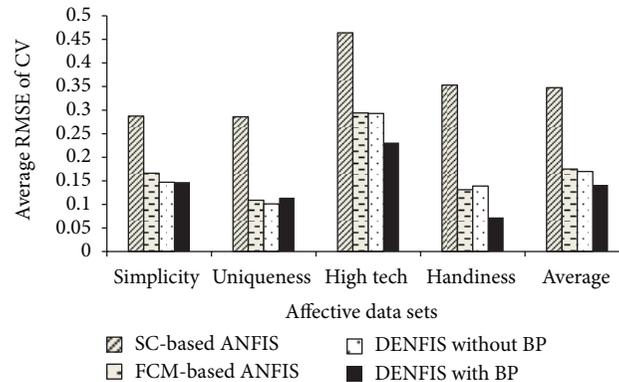


FIGURE 9: Histogram of the average RMSE of the NF models.

TABLE 4: Test performance of the NF models (high tech).

Models	Average RMSE	Average computational time*/s
SC-based ANFIS	0.464	0.6048
FCM-based ANFIS	0.2941	0.6887
DENFIS without BP	0.2927	0.493
DENFIS with BP	0.2312	

* Average computational time per update of dynamic NF models is 0.01826 s.

TABLE 5: Test performance of the NF models (handiness).

Models	Average RMSE	Average computational time*/s
SC-based ANFIS	0.3533	0.6859
FCM-based ANFIS	0.1312	0.6277
DENFIS without BP	0.1387	0.5047
DENFIS with BP	0.07244	

* Average computational time per update of dynamic NF models is 0.01869 s.

likewise addresses the two important aspects that cannot be handled by the existing NF models: (i) the existing NF models cannot be used to model the affective relationships which involve a large number of inputs; and (ii) the existing NF models cannot adapt to customers' affective preferences effectively that are obtained from newly collected survey data. The effectiveness of the proposed approach to modeling customer satisfaction for affective design is demonstrated using a case study of the affective design of the mobile phones. Results of the validation tests show that the dynamic NF model displays better prediction performance and shorter processing time as compared with the existing NF models including the SC-based and FCM-based ANFIS models. The performance of the dynamic NF models with BP training is the best among the NF models in the cross-validation tests. In addition, the dynamic NF models reduce the computational time by 18% to 29% as compared with the other two models. The fast incremental training process of the dynamic NF model enables long-term updates and maintenance of these models. Future work could study the generated models based

on the proposed approach to determine the optimal setting of the design attributes for affective design.

Acknowledgment

The work described in this paper was supported by a grant from The Hong Kong Polytechnic University (Project no. G-YK81 and A/C no. RU0H).

References

- [1] N. Crilly, J. Moultrie, and P. J. Clarkson, "Seeing things: consumer response to the visual domain in product design," *Design Studies*, vol. 25, no. 6, pp. 547–577, 2004.
- [2] Z. He and D. Wu, "A comparative study of ordinal probit and logistic regression for affective product design," *Advanced Materials Research*, vol. 452–453, pp. 642–647, 2012.
- [3] Y. Liu, "Engineering aesthetics and aesthetic ergonomics: theoretical foundations and a dual-process research methodology," *Ergonomics*, vol. 46, no. 13–14, pp. 1273–1292, 2003.
- [4] M. E. H. Creusen and J. P. L. Schoormans, "The different roles of product appearance in consumer choice," *Journal of Product Innovation Management*, vol. 22, no. 1, pp. 63–81, 2005.
- [5] C. H. Noble and M. Kumar, "Using product design strategically to create deeper consumer connections," *Business Horizons*, vol. 51, no. 5, pp. 441–450, 2008.
- [6] A. Turkyilmaz, A. Oztekin, S. Zaim, and O. F. Demirel, "Universal structure modeling approach to customer satisfaction index," *Industrial Management & Data Systems*, vol. 113, no. 7, pp. 932–949, 2013.
- [7] F. R. Camargo and B. Henson, "Measuring affective responses for human-oriented product design using the Rasch model," *Journal of Design Research*, vol. 9, no. 4, pp. 360–375, 2011.
- [8] M. Nagamachi, "Kansei engineering as a powerful consumer-oriented technology for product development," *Applied Ergonomics*, vol. 33, no. 3, pp. 289–294, 2002.
- [9] J. Jiao, Y. Zhang, and M. Helander, "A Kansei mining system for affective design," *Expert Systems with Applications*, vol. 30, no. 4, pp. 658–673, 2006.
- [10] H. M. Khalid and M. G. Helander, "A framework for affective customer needs in product design," *Theoretical Issues in Ergonomics Science*, vol. 5, no. 1, pp. 27–42, 2004.

- [11] M.-D. Shieh, C.-L. Huang, and C.-C. Yang, "A rough set approach to elicit customer preferences for fashion design," *Journal of Convergence Information Technology*, vol. 8, no. 3, pp. 338–349, 2013.
- [12] L.-K. Chan and M.-L. Wu, "Quality function deployment: a literature review," *European Journal of Operational Research*, vol. 143, no. 3, pp. 463–497, 2002.
- [13] H.-H. Lai, Y.-M. Chang, and H.-C. Chang, "A robust design approach for enhancing the feeling quality of a product: a car profile case study," *International Journal of Industrial Ergonomics*, vol. 35, no. 5, pp. 445–460, 2005.
- [14] P. Tarantino, *A Statistical Thinking Approach to Kansei Engineering for Product Innovation*, University of Naples Federico II, Naples, Italy, 2008.
- [15] E. Aktar Demirtas, A. S. Anagun, and G. Koksall, "Determination of optimal product styles by ordinal logistic regression versus conjoint analysis for kitchen faucets," *International Journal of Industrial Ergonomics*, vol. 39, no. 5, pp. 866–875, 2009.
- [16] H.-H. Lai, Y.-C. Lin, and C.-H. Yeh, "Form design of product image using grey relational analysis and neural network models," *Computers & Operations Research*, vol. 32, no. 10, pp. 2689–2711, 2005.
- [17] S.-W. Hsiao and H.-C. Tsai, "Applying a hybrid approach based on fuzzy neural network and genetic algorithm to product form design," *International Journal of Industrial Ergonomics*, vol. 35, no. 5, pp. 411–428, 2005.
- [18] X. Wan, J. Che, and L. Han, "Car styling perceptual modeling based on fuzzy rules," *Applied Mechanics and Materials*, vol. 201–202, pp. 794–797, 2012.
- [19] H.-C. Tsai, S.-W. Hsiao, and F.-K. Hung, "An image evaluation approach for parameter-based product form and color design," *Computer Aided Design*, vol. 38, no. 2, pp. 157–171, 2006.
- [20] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.
- [21] N. Kasabov, "Evolving neuro-fuzzy inference models," in *Evolving Connectionist Systems: The Knowledge Engineering Approach*, pp. 141–176, 2007.
- [22] L. A. Zadeh and C. Berkeley, *Fuzzy Logic Toolbox User's Guide Version 2*, MathWorks, 2001.
- [23] N. K. Kasabov and Q. Song, "DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154, 2002.
- [24] N. Kasabov, Q. Song, and T. Ma, "Fuzzy-neuro systems for local and personalized modelling," in *Forging New Frontiers: Fuzzy Pioneers II*, pp. 175–197, 2008.
- [25] N. Kasabov, "Evolving connectionist methods for unsupervised learning," in *Evolving Connectionist Systems*, pp. 53–82, 2007.

Research Article

From Nonlinear Optimization to Convex Optimization through Firefly Algorithm and Indirect Approach with Applications to CAD/CAM

Akemi Gálvez¹ and Andrés Iglesias^{1,2}

¹ Department of Applied Mathematics and Computational Sciences, E.T.S.I. Caminos, Canales y Puertos, University of Cantabria, Avenida de los Castros s/n, 39005 Santander, Spain

² Department of Information Science, Faculty of Sciences, Toho University, 2-2-1 Miyama, Funabashi 274-8510, Japan

Correspondence should be addressed to Andrés Iglesias; iglesias@unican.es

Received 16 August 2013; Accepted 29 September 2013

Academic Editors: Z. Cui and X. Yang

Copyright © 2013 A. Gálvez and A. Iglesias. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Fitting spline curves to data points is a very important issue in many applied fields. It is also challenging, because these curves typically depend on many continuous variables in a highly interrelated nonlinear way. In general, it is not possible to compute these parameters analytically, so the problem is formulated as a continuous nonlinear optimization problem, for which traditional optimization techniques usually fail. This paper presents a new bioinspired method to tackle this issue. In this method, optimization is performed through a combination of two techniques. Firstly, we apply the indirect approach to the knots, in which they are not initially the subject of optimization but precomputed with a coarse approximation scheme. Secondly, a powerful bioinspired metaheuristic technique, the firefly algorithm, is applied to optimization of data parameterization; then, the knot vector is refined by using De Boor's method, thus yielding a better approximation to the optimal knot vector. This scheme converts the original nonlinear continuous optimization problem into a convex optimization problem, solved by singular value decomposition. Our method is applied to some illustrative real-world examples from the CAD/CAM field. Our experimental results show that the proposed scheme can solve the original continuous nonlinear optimization problem very efficiently.

1. Introduction

Fitting spline curves to data points is a problem that appears very frequently in many scientific and engineering fields. Typical examples span from regression analysis in statistics [1, 2] to contour reconstruction in medical imaging [3]. They also encompass the computation of outlines in image processing [4], shape manipulation in geometric modeling and processing [5–7], and data approximation methods in numerical analysis [8, 9], to mention just a few examples. In this paper, our main motivation comes from the fields of computer-aided design and manufacturing (CAD/CAM) where spline curves are intensively used in many problems [10–14]. One of them is to fit data points obtained from metrology in CAD/CAM, a field consisting of the application of measurement technology to the quality control assessment

of designed or manufactured products in many manufacturing industries (automotive, aerospace, ship building, shoes, etc.).

In spite of its wide range of applications, the use of spline curves is still challenging because they typically depend on many different continuous variables (data parameters, knots, and spline coefficients) in a highly nonlinear way [15–20]. These sets of variables are also interrelated, meaning that changes in the values of a particular set of parameters affect the behavior of the others, and hence they cannot be manipulated independently [21–24]. For instance, the choice of knots depends on the curve parameterization, which in turn depends on the underlying structure of data points. Similarly, the computation of the spline coefficients depends on both the parameterization and the knots, and so on. From a mathematical standpoint, this implies that the fitting problem

cannot be partitioned into independent subproblems for the different sets of variables. As a consequence, it is not possible in general to compute all these parameters analytically [19, 25]. Instead, the typical formulation in the field is to treat this problem as a continuous nonlinear optimization problem [26, 27]. The bottom point is that traditional optimization techniques have also failed to provide satisfactory answer to this optimization problem. Among the alternatives suggested to solve this limitation, those based on artificial intelligence techniques captured the interest of the scientific community some years ago. The main line of research focused on the neural networks [28–30] and their extension, the functional networks [31–34]. However, the solutions reported were partial and applicable only to some particular problems. Consequently, there is a need for more efficient approaches to tackle this issue.

During the last few years, scientists and engineers have turned their attention to *bioinspired computation*, a field where the interplay between nature and computers has allowed us to model the living phenomena by using mathematics and computer science [35–37]. Simultaneously, the study of life has led to improved schemes to solve many problems in mathematics and computer science, including optimization problems [38–40]. Due to their good behavior for complex optimization problems involving ambiguous and noisy data, there has recently been an increasing interest in applying bioinspired optimization techniques to the spline fitting problem. However, there are still few works reported in the literature. Recent schemes in this area are described for particle swarm optimization [41–43], genetic algorithms [27, 44, 45], artificial immune systems [46, 47], estimation of distribution algorithms [48], and hybrid approaches [49–51].

Specially remarkable is the fact that some bioinspired methods have proved to be able to solve difficult optimization problems unsolvable with traditional optimization techniques. Being this our case, we turned our attention to a powerful bioinspired metaheuristic called firefly algorithm, recently introduced by Professor Xin-She Yang to solve difficult continuous optimization problems. The firefly algorithm is inspired in the flashing behavior of the fireflies and their social interaction in the natural environment (see Section 3 for details).

In this paper, we present a new bioinspired scheme for computing all parameters of a spline curve approximating a given set of data points. Our proposal is based on two fundamental techniques: the indirect approach and the firefly algorithm, which are combined in our method to perform the optimization of the knots and the data parameters, respectively. The indirect approach tries to overcome the fact that computing the knots requires a previous parameterization which, at its turn, requires a previous knot vector, leading in practice to a never-ending vicious circle. In the indirect approach, the knots are not initially the subject of optimization but precomputed with a coarse approximation scheme, which will be further improved at a later stage. This precomputed knot vector plays the role of an initial seed for the data parameterization step. An obvious risk of this indirect approach is that the whole method relies on this optimization stage. In this way, data parameterization becomes the most

critical step, since it carries out the most significant part of the optimization effort. Consequently, we need a powerful, reliable optimization method for this task. As it will be shown later on, the firefly algorithm is a good choice for this step. It is applied in the second step to perform optimization on data parameterization; then, the knot vector is refined by using De Boor's method, thus yielding a better approximation of the optimal knot vector. These two combined methods convert the original nonlinear continuous optimization problem into a convex optimization problem, which is solved by applying singular value decomposition. This scheme is applied to some illustrative real-world examples from the CAD/CAM field, including the side profile curve of a car body, the outline curves of a paint spray gun, and a 3D CAD/CAM workpiece from the automotive industry. Our experimental results show that the proposed scheme can solve the original continuous nonlinear optimization problem very efficiently.

The structure of this paper is as follows. Firstly, some basic concepts about parametric spline curves are given in Section 2. Then, Section 3 describes the firefly algorithm, the bioinspired metaheuristic used in this paper. The core of the paper is in Section 4, where our proposed method for spline curve fitting is reported in detail. Section 5 describes the experimental results of the application of our method to three illustrative real-world problems from the CAD/CAM field. The paper closes with the main conclusions and our plans for future work.

2. Parametric Spline Curves

In this section we describe the basic concepts needed in this paper about the parametric spline functions. The interested reader is referred to [6, 7, 52, 53] for a more detailed discussion about this subject. Note that in this paper vectors are denoted in bold.

Let $\Phi(\tau) = (\phi^1(\tau), \dots, \phi^n(\tau))$ be a parametric function defined on a finite interval $[\alpha, \beta]$. Consider now a strictly increasing sequence of real numbers $\mu_0 = \alpha < \mu_1 < \dots < \mu_v < \mu_{v+1} = \beta$ called knots. The function $\Phi(\tau)$ is a parametric polynomial spline of degree $\eta \geq 0$ with knots $\{\mu_k\}_k$ if the following two conditions are fulfilled for $i = 0, \dots, v$ and $j = 1, \dots, n$:

- (1) $\phi^j(\tau)$ is a polynomial spline of degree up to η on each interval $[\mu_i, \mu_{i+1}]$,
- (2) $\phi^j(\tau)$ and its derivatives up to order $\eta - 1$ are continuous on $[\mu_i, \mu_{i+1}]$.

Different basis functions can be used for polynomial splines. In this paper, we consider the B-spline basis functions of degree ν defined on $[\mu_i, \mu_{i+1}]$ according to the Cox-de-Boor recursive formula [52]:

$$\psi_{i,\nu+1}(\tau) = \varphi_{i,\nu}^+(\tau) \psi_{i,\nu}(\tau) + \varphi_{i+1,\nu}^-(\tau) \psi_{i+1,\nu}(\tau), \quad (1)$$

$$i = 0, \dots, v - \nu, \quad \nu > 1,$$

where $\varphi_{i,\nu}^+(\tau) = (\tau - \mu_i) / (\mu_{i+\nu} - \mu_i)$, $\varphi_{i,\nu}^-(\tau) = (\mu_{i+\nu} - \tau) / (\mu_{i+\nu} - \mu_i)$, and $\psi_{i,1}(\tau)$ is the unit function with support on

the interval $[\mu_i, \mu_{i+1}]$. The dimension of the vector space of functions satisfying conditions (1) and (2) is $v + \eta + 1$. The given knot vector $\{\mu_i\}_i$ yields $v - \eta + 1$ linearly independent basis functions of degree η . The remaining 2η basis functions are obtained by introducing the boundary knots $\mu_{-\eta} = \mu_{-\eta+1} = \dots = \mu_{-1} = \mu_0 = \alpha$ and $\mu_{v+1} = \mu_{v+2} = \dots = \mu_{v+\eta+1} = \beta$. With this choice of boundary knots all basis functions vanish outside the interval domain $[\alpha, \beta]$. Every parametric spline curve $\Phi(\tau)$ is represented by

$$\Phi(\tau) = \sum_{i=-\eta}^v \Xi_i \psi_{i,\eta+1}(\tau), \quad (2)$$

where $\{\Xi_i\}$ are the spline coefficients of the curve and $\psi_{i,\eta+1}(\tau)$ are the basis functions defined above. The k th derivative of $\Phi(\tau)$ is a spline of degree $\eta - k$ given by

$$\Phi^{(k)}(\tau) = \prod_{i=1}^k (\eta + 1 - i) \sum_{i=-\eta+k}^v \Xi_i^{(k)} \psi_{i,\eta+1-k}(\tau) \quad (3)$$

with $\Xi_i^{(j)} = (\Xi_i^{(j-1)} - \Xi_{i-1}^{(j-1)}) / (\mu_{i+\eta+1-j} - \mu_i)$ for $j > 0$ and $\Xi_i^{(0)} = \Xi_i$.

3. The Firefly Algorithm

The firefly algorithm (FFA) is a bioinspired metaheuristic algorithm introduced in 2008 by Yang to solve optimization problems [54–58]. The algorithm is based on the flashing behavior of the fireflies and their social interaction in the natural environment. The key ingredients of the method are the variation of light intensity and formulation of attractiveness. In general, the attractiveness of an individual is assumed to be proportional to their brightness, which in turn is associated with the encoded objective function. The reader is kindly referred to [40] for a comprehensive review of the firefly algorithm and other nature-inspired metaheuristic approaches. See also [59] for a gentle introduction to metaheuristic applications in engineering optimization.

In the firefly algorithm, there are three particular idealized rules, which are based on some of the major flashing characteristics of real fireflies [54] as follows

- (1) All fireflies are unisex, so that one firefly will be attracted to other fireflies regardless of their sex.
- (2) The degree of attractiveness of a firefly is proportional to its brightness, which decreases as the distance from the other firefly increases due to the fact that the air absorbs light. For any two flashing fireflies, the less bright one will move towards the brighter one. If there is not a brighter or more attractive firefly than a particular one, it will then move randomly.
- (3) The brightness or light intensity of a firefly is determined by the value of the objective function of a given problem. For instance, for maximization problems, the light intensity can simply be proportional to the value of the objective function.

The distance between any two fireflies i and j , at positions \mathbf{P}_i and \mathbf{P}_j , respectively, can be defined as a Cartesian or Euclidean distance as follows:

$$r_{ij} = \|\mathbf{P}_i - \mathbf{P}_j\|_2 = \sqrt{\sum_{k=1}^D (p_i^k - p_j^k)^2}, \quad (4)$$

where p_i^k is the k th component of the spatial coordinate \mathbf{P}_i of the i th firefly and D is the number of dimensions. In the firefly algorithm, as attractiveness function of a firefly j one should select any monotonically decreasing function of the distance to the chosen firefly, for example, the exponential function:

$$\hat{\beta} = \hat{\beta}_0 e^{-\hat{\gamma} r_{ij}^{\hat{\mu}}}, \quad \hat{\mu} \geq 1, \quad (5)$$

where r_{ij} is the distance defined as in (4), $\hat{\beta}_0$ is the initial attractiveness at $r = 0$, and $\hat{\gamma}$ is an absorption coefficient at the source which controls the decrease of the light intensity.

The movement of a firefly i which is attracted by a more attractive (i.e., brighter) firefly j is governed by the following evolution equation:

$$\mathbf{P}_i^{(t+1)} = \mathbf{P}_i^{(t)} + \hat{\beta}_0 e^{-\hat{\gamma} r_{ij}^{\hat{\mu}}} (\mathbf{P}_j^{(t)} - \mathbf{P}_i^{(t)}) + \hat{\alpha} \left(\hat{\sigma} - \frac{1}{2} \right), \quad (6)$$

where the superscripts between brackets in this expression are used to denote the corresponding generations. The first term on the right-hand side is the current position of the firefly at generation t , the second term is used for considering the attractiveness of the firefly to light intensity seen by adjacent fireflies, and the third term is used for the random movement of a firefly in case there are not any brighter ones. The coefficient $\hat{\alpha}$ is a randomization parameter determined by the problem of interest, while $\hat{\sigma}$ is a random number generator uniformly distributed in the space $[0, 1]$.

The method described in previous paragraphs corresponds to the original version of the firefly algorithm, as originally developed by its inventor. Since then, many different modifications and improvements on the original version have been developed, including the discrete FFA, multiobjective FFA, chaotic FFA, parallel FFA, elitist FFA, Lagrangian FFA, and many others, including its hybridization with other techniques. The interested reader is referred to the nice paper in [60] for a comprehensive, updated review and taxonomic classification of the firefly algorithms and all its variants and applications.

4. The Method

In this section our FFA-based method is fully explained. The section begins with the description of the optimization problem to be solved. Then, a general overview of the method and its flowchart are given. Then, each step of the method is discussed in detail. Finally, some details regarding the implementation issues are also given.

4.1. The Optimization Problem. Let us suppose that we are provided with a set of measured data points $\{\Theta_k\}_{k=1,\dots,\rho} \subset \mathbb{R}^D$

\mathbb{R}^n obtained by laser scanning, layout machine, or other digitizing methods, as it typically happens in many scientific and engineering problems. The goal consists of obtaining a parametric spline curve $\Phi(\tau)$ of degree η defined as above approximating the $\{\Theta_k\}_k$. Due to the conditions on the boundary knots, we can take $\Phi(\tau_1) = \Theta_1$ and $\Phi(\tau_\rho) = \Theta_\rho$ and perform approximation on the remaining parameters; that is,

$$\Theta_k \approx \Phi(\tau_k) = \sum_{j=-\eta}^v \Xi_j \psi_{j,\eta+1}(\tau_k) \quad (k = 2, \dots, \rho - 1). \quad (7)$$

Equation (7) can be written in matrix notation as

$$\Theta = \Psi \cdot \Xi, \quad (8)$$

where $\Theta = (\Theta_2, \dots, \Theta_{\rho-1})^T$, $\Xi = (\Xi_{-\eta}, \dots, \Xi_v)^T$, $\Psi = (\{\psi_{j,\eta+1}(\tau_k)\}_{j=2, \dots, \rho-1; k=-\eta, \dots, v})$ is the matrix of sampled spline basis functions, and $(\cdot)^T$ represents the transpose of a vector or matrix. The dimension of the search space D in (8) is given by $n(v + \eta - 1) + v + \rho - 2$, which could be of several thousands of variables for nontrivial shapes. Since the system (8) is overdetermined, the matrix of basis functions is not invertible and no direct solution can be obtained. Therefore, we consider the least-squares approximation of (7), defined as the minimization problem given by

$$\underset{\substack{\{\omega_k\}_k \\ \{\Xi_j\}_j \\ \{\tau_k\}_k}}{\text{minimize}} \left(\sum_{k=2}^{\rho-1} \omega_k \left\| \Theta_k - \sum_{j=-\eta}^v \Xi_j \psi_{j,\eta+1}(\tau_k) \right\|_{\ell_2} \right), \quad (9)$$

where ω_k are scalar weights and ℓ_2 represents the Euclidean norm (although any other norm might be used instead). Note that the parameters and knots are related by nonlinear basis functions, thus leading to a high-dimensional continuous nonlinear optimization problem. Assuming that a suitable data parameterization can be obtained, we have to solve a nonlinear continuous optimization problem involving both the spline coefficients and the knots as free variables of the problem. Unfortunately, this approach makes the optimization problem nonconvex, because $\Phi(\tau)$ is a nonconvex function of the knots [19, 26, 52]. To overcome this problem, we follow the so-called indirect approach, in which the knots are precomputed before the optimization process is executed and then refined for better fitting. With this strategy, the resulting problem is convex, so a global optimum can eventually be found. In order to apply the previous strategy, we need to obtain a suitable parameterization of data points, which thus becomes the most critical step of this approach. We solve this parameterization problem by applying the firefly algorithm, as it will be explained in next paragraphs.

4.2. Overview of the Method. The main steps of our method are summarized in Figure 1, showing the flowchart of our approach. Our initial input is given by the set of data points $\{\Theta_j\}_j$ and two parameters that are freely chosen by the user: the length of knot vector (determined by variable v), and the curve degree η (typical values for η are between 2 and 4, although any natural value can be used in our method).

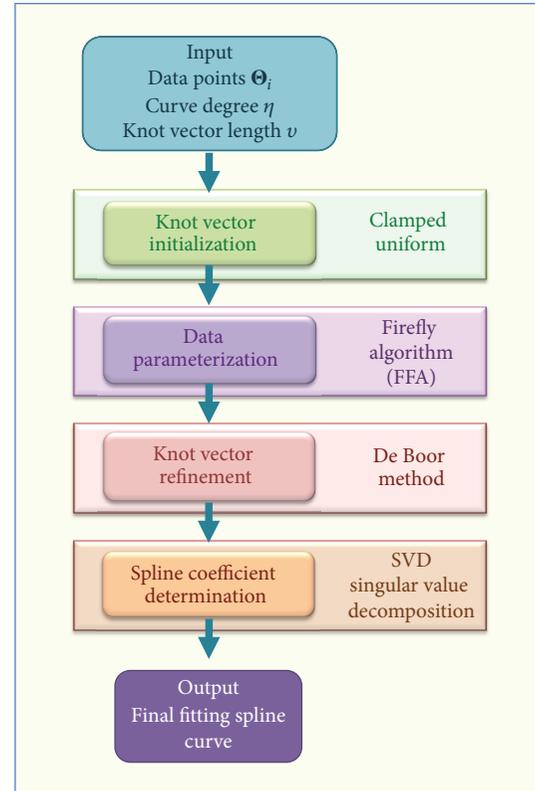


FIGURE 1: Graphical flowchart of the proposed method.

The first step of our approach consists of computing an initial knot vector. Then, we apply the firefly algorithm to perform data parameterization (Step 2). A new (refined) knot vector is computed based on the parameterization obtained in the previous step. Then, the convex optimization problem is solved by using singular value decomposition (Step 4). After this step, the best fitting spline curve to the data points for the given degree is finally obtained.

4.3. Main Steps of the Method. The proposed method consists of four main steps, analyzed in the next paragraphs.

Step 1 (knot vector initialization). The first step of the method computes an initial knot vector, which is required in order to evaluate the fitness functions during the optimization step for data parameterization. To this aim, we consider a clamped knot vector (this condition is a result of our choice of boundary knots) whereas the internal knots are uniformly distributed on the interval (α, β) ; that is, $\mu_i = \alpha + ((\beta - \alpha)i)/(v+1)$, for $i = 1, \dots, v$. This choice of knots is not optimal because it does not reflect the distribution of data points. This limitation will be overcome in Step 3, where this initial knot vector will be further refined.

Step 2 (data parameterization). In this step, the data points parameterization is carried out. As mentioned above, this is the most critical step of our method; the set of data parameters along with the knot vector, computed in the previous step and refined in the next one, allows us to convert the original

nonlinear nonconvex optimization problem (9) into a convex optimization problem. This task is accomplished by applying the firefly algorithm described in Section 3. To this purpose, a collection of n_f particles (fireflies) is considered. Each firefly corresponds to a vector of $\rho - 2$ real numbers on the interval (α, β) . The components of each firefly vector are always sorted in an increasing order to reflect the ordered structure of the data points. The fireflies are initialized by using a random function of uniform distribution on the interval domain (α, β) . We have used a collection of $n_f = 100$ fireflies for the examples reported in this paper. We also checked our results with larger populations by changing this parameter from 100 to 1000 fireflies with step-size 100 and do not notice significant variations in our results. However, a larger value could be required for very massive sets of data points ($\geq 10^5$ data points) exhibiting very complicated shapes.

Once the initial population of fireflies is generated, some parameters of the firefly algorithm have to be determined in order to apply them to our problem. Our choice of the parameters for the FFA is mostly empirical: we initially rely on standard values reported in the existing literature and then perform computer experiments to validate our parameter tuning. In this paper, we consider the following set of parameter values for the FFA method: $\hat{\beta}_0 = 1$, $\hat{\gamma} = 0.5$, and $\hat{\mu} = 2$. This set of parameter values has already been used in a previous paper by the authors for a Bézier surface parameterization problem with good results [61]. Our computer experiments also confirmed their good performance for the examples discussed in this paper and some others not reported here to keep the paper at manageable size. On the contrary, the number of iterations and the value of parameter $\hat{\alpha}$ required some improvement. We initially used a fixed number of iterations in our experiments as the stopping criterion, but this choice was found to be very inefficient for this problem. The main reasons are that the method can potentially be applied to sets of very different number of data points, meaning that the dimension of the search space can vary dramatically from one example to another, and that the initial knot vector used in our method is not optimal yet, making it difficult to determine in advance how many iterations are needed to achieve convergence. We then turned to a different termination condition, where the number of iterations is determined manually for each specific example, based on the observation of the convergence diagrams (as those shown in Figures 3 and 5). Although it is a tedious and time-consuming task, we found it to be more reliable in order to ensure convergence is properly achieved. Regarding the value for $\hat{\alpha}$, our initial choice $\hat{\alpha} = 0.3$ soon revealed to be too drastic, as the fireflies went out of range in just a few iterations (sometimes, even a single one was enough). We decreased its value gradually and carried out a lot of computer simulations. As a result, the best value was found at $\hat{\alpha} = 0.01$.

The last required component of the FFA is the fitness function. It corresponds to the evaluation of the least-squares function given by the operator to be minimized in (9); that is,

$$\Omega = \sum_{k=2}^{\rho-1} \omega_k \left\| \Theta_k - \sum_{j=-\eta}^v \Xi_j \psi_{j,\eta+1}(t_k) \right\|_{\ell_2}, \quad (10)$$

where the weights w_k are scalar numbers to express the degree of confidence of data points (larger weights are assigned to more reliable data). In this paper, we assume a constant confidence value for all data, so we take $w_k = 1, k = 2, \dots, \rho - 1$. After the selection of those parameters and the fitness function, the firefly algorithm is performed iteratively until the termination criterion is reached. To remove the stochastic effects of single executions, 20 independent executions have been carried out for each simulation trial. Then, the firefly with the best (i.e., minimum) fitness value is selected as the best solution to the problem. As a result, the best parameterization vector of data points is obtained.

Step 3 (knot vector refinement). Based on the parameterization obtained in the previous step, the knot vector is subsequently refined for better performance. To this purpose, the placement of knots should reflect the distribution of data parameters $\{\tau_k\}_k$. In this paper, the internal knots are computed by following a procedure firstly proposed by De Boor in [52]. The corresponding algorithm is described in Algorithm 1. It has been proved that this algorithm guarantees that every knot span contains at least one τ_k . At its turn, this condition ensures that the matrix $\Gamma = \Psi^T \Psi$ is positive definite and well conditioned. Furthermore, Γ has a semibandwidth less than $\eta - 1$ (see [52] for details). All these properties imply that the system of equations related to the convex problem can be solved by using efficient and reliable numerical methods [62], as explained in the next paragraphs.

Step 4 (spline coefficient determination). As we mentioned above, in this paper, we follow an indirect approach to compute the fitting spline curve to the data points. In this scheme, data parameters and knots are computed at earlier stages of the optimization process so that the approximation problem (9) becomes a convex problem. In that case, premultiplying by Ψ^T at both sides of (8), we get

$$\Gamma \cdot \Xi = \Sigma, \quad (11)$$

where $\Sigma = \Psi^T \cdot \Theta = (\{\sum_{k=2}^{\rho-1} \Theta_k \psi_{i,\eta+1}(\tau_k)\}_{i=-\eta, \dots, v})^T$. Note that $\Gamma = \Psi^T \Psi$ is a symmetric square matrix and positive semidefinite, so system (11) always has a solution. It can be solved numerically by Gaussian elimination, LU decomposition, or the singular value decomposition (SVD) (see [62] for details). In this paper, SVD has been used since it provides the best numerical answer in the sense of least squares for those cases in which the exact solution is not possible. To this aim, Γ is decomposed as the matrix product $\Gamma = \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T$, where \mathbf{U} is a column orthogonal matrix, \mathbf{W} is a diagonal matrix with positive or zero elements w_k called the singular values, and \mathbf{V} is a square orthogonal matrix. Furthermore, its inverse can readily be obtained as: $\Gamma^{-1} = \mathbf{V} \cdot [\text{diag}(1/w_k)] \cdot \mathbf{U}^T$. In addition, the knot vector obtained by the procedure described in Step 3 guarantees that Γ is positive, definite, and, therefore, nonsingular, so the problem can be solved by using this inverse matrix Γ^{-1} .

```

INPUT:       $\bar{\tau} = \{\tau_1, \dots, \tau_\rho\}$       /* Vector of data parameters from Step 2 */
             $v$                                 /* Number of internal knots */
OUTPUT:     $\bar{\mu} = \{\mu_1, \dots, \mu_v\}$       /* Vector of internal knots */
ALGORITHM: {Initialization}
             $\bar{\mu} \leftarrow \{\}$ 
             $\Delta \leftarrow \frac{\rho + 1}{v + 1}$ 
{Main loop}
  for  $j = 1$  to  $v$  do
     $i \leftarrow \text{int}(j\Delta)$  /*  $\text{int}(k)$ : returns the largest integer number  $\leq k$  */
     $\sigma \leftarrow j\Delta - i$ 
     $\mu_j \leftarrow (1 - \sigma)\tau_{i-1} + \sigma \tau_i$ 
     $\bar{\mu} \leftarrow \text{Append}(\mu_j)$ 
  end for

```

ALGORITHM 1: De Boor's knot vector refinement algorithm.

4.4. Implementation Issues. Regarding the implementation, all computations in this paper have been performed on a 2.6GHz Intel Core i7 processor with 8GB of RAM. The source code has been implemented by the authors in the native programming language of the popular scientific program *Matlab*, version 2012a. In our opinion, *Matlab* is a very suitable tool for this task: it is fast and provides reliable, well-tested routines for efficient matrix manipulations. It also contains a bulk of resources regarding the solving of systems of equations. For instance, *Matlab* provides us with the command `mldivide` to solve the equation $\mathbf{A} \cdot \mathbf{X} = \mathbf{B}$ for both squared and nonsquared systems (by using Gaussian elimination with partial pivoting and least-squares techniques, resp.). Depending on the general structure of matrix \mathbf{A} , this command applies specialized LAPACK and BLAS routines to get the best possible solution to this system. Also, *Matlab* provides us with a specialized command `svd` for computing the SVD of a matrix. This command carries out the matrix SVD decomposition automatically so it can be used in a rather black box-like way. Besides, *Matlab* provides excellent graphical options and optimized code for input/output interaction and high performance computations.

5. Experimental Results

Our method has been applied to several real-world examples of CAD/CAM shapes for the automotive industry. In this section we discuss three of them. The reported examples reflect the variety of cases our method can be applied to: they include both open and closed curves, as well as 2D and 3D shapes comprised of a single curve and multiple curves. We think we have provided enough examples to convince the reader of the broad applicability of our approach.

5.1. Side Profile Curve of a Car Body. The first example consists of the data fitting of a set of 695 data points from the *In* (+*y*-axis) side profile section of a model of a notchback three-box sedan car body. The data points were obtained

by a layout machine from the Spanish car maker provider Candemat years ago. This example has been primarily chosen because it includes areas of varying slope, ranging from the strong slope at both ends (upwards on the left, downwards on the right) to the soft slope in middle part, with a horizontal area between pillars A and C and in the cargo box area and a soft upward area in the car hood. In addition, it is a good example of a truly parametric curve that cannot be faithfully represented by simpler functions such as explicit functions and the like. Consequently, it is a very good candidate to check the performance of our approach.

The problem is also challenging because we are trying to represent the whole shape with a single curve. It is worthwhile mentioning here that the use of a single curve for a whole object is not common at all in industrial environments, where the shapes of final products such as a car body are usually represented by a very large set (several hundreds of thousands, even millions) of simpler curves. Therefore, even though we are using data from a real-world shape, this example must be understood as a purely academic example rather than a genuine real-world problem in the automotive industry. Yet, this example is very useful in this paper to analyze the performance of our approach.

Figure 2 shows our simulation results for a spline curve of degree $\eta = 3$. Top figure shows the original data points, represented by red cross symbols along with the reconstructed data points, displayed as empty circles in blue. The picture is intended to show the correspondence between the original and the reconstructed data points in a graphical way. As the reader can see, our method yields a very good matching between both sets of points. This observation is confirmed by Figure 2 (bottom), representing the same original data points along with the approximating spline curve, displayed as a solid line in blue. Figure 3 shows the evolution of the fitting error of the Ω operator for 1000 iterations. Since the diagram shows a similar general behavior for the 20 independent executions and larger number of iterations no longer improves the fitting error, we conclude that this number of iterations is enough for convergence. The corresponding fitting error in this example is 1.1248×10^{-2} .

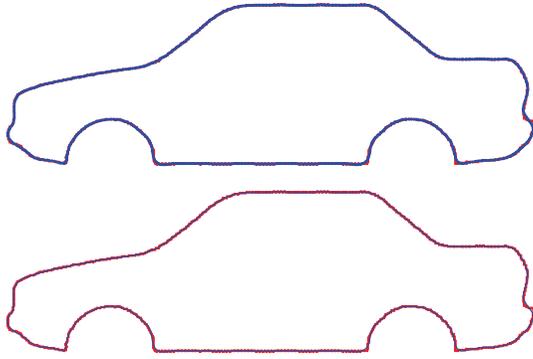


FIGURE 2: Adjusting data points (red cross symbols in both pictures) of a car body side profile with our method: (top) reconstructed data points (blue empty circle symbols); (bottom) approximating spline curve (blue solid line).

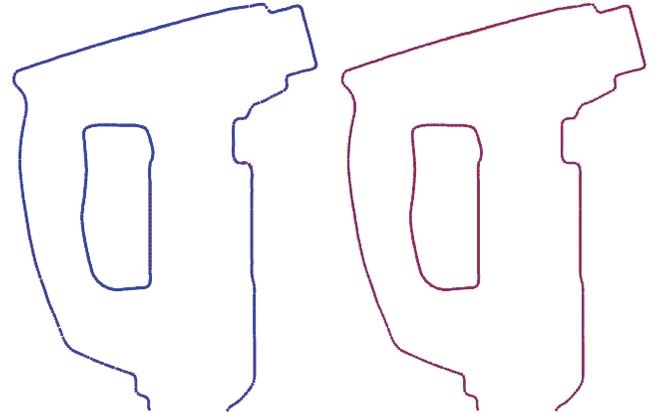


FIGURE 4: Adjusting data points (red cross symbols in both pictures) of two outlines curves of a paint spray gun with our method: (left) reconstructed data points (blue empty circle symbols); (right) approximating spline curve (blue solid line).

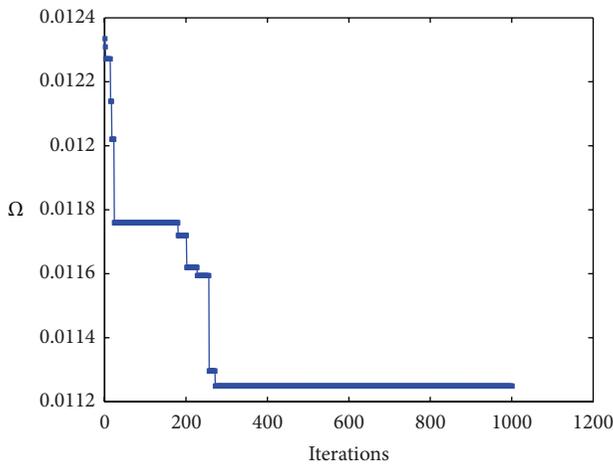


FIGURE 3: Fitting error evolution of the Ω operator for the car body example for 1000 iterations.

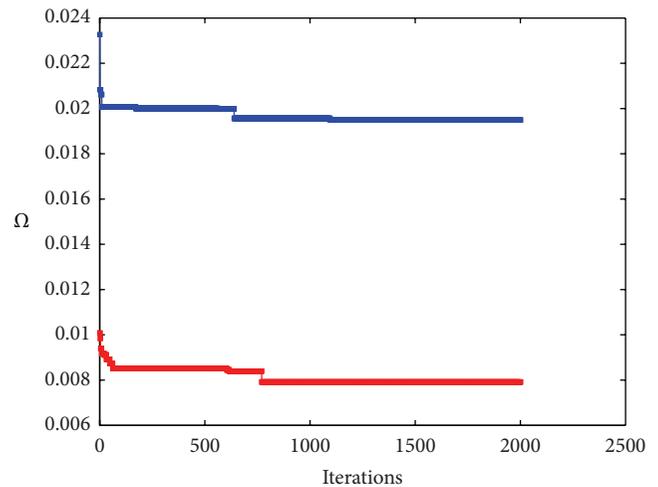


FIGURE 5: Fitting error evolution of the Ω operator for the outer curve (in blue) and the inner curve (in red) of the paint spray gun example for 2000 iterations.

This value gives only partial information because it does not consider the number of sampled points. This means that increasing the number of data points leads automatically to larger fitting errors even though each data point might be better fitted. To overcome this drawback, we also compute the root-mean square error (RMSE), which gives a better measure of the quality of the approximation. In this example we obtain an RMSE fitting error of 4.2669×10^{-4} . This value confirms the good performance of the proposed method for this problem. A close inspection of Figure 2 reveals, however, that some parts can still be further improved: this fact is visually noticeable in Figure 2, specially at the corners of the front and rear fenders and their linkings to the lower car body line, as well as at the lower parts of front and rear bumpers. This effect is attributable to our indirect approach, in which we do not compute the optimal knot vector but an approximation. We are currently working towards an improved approach to solve this limitation.

5.2. *Outline Curves of a Paint Spray Gun.* Figure 4 shows the results of our method for spline curves of degree $\eta = 3$ when

applied to a paint spray gun model. Left and right pictures of the figure have the same meaning as the top and bottom pictures of the previous figure, respectively. The spray gun model consists of two different curves for the outer and inner boundary lines with 542 and 276 data points, respectively. We applied our method to each curve independently. The fitting errors of the Ω operator for the outer and inner curves are 1.9493×10^{-2} and 7.9145×10^{-3} , respectively. The RMSE fitting errors are 8.3729×10^{-4} and 4.7639×10^{-4} , respectively. These values are reached for 2000 iterations, well within the convergence area as shown in Figure 5, which displays the evolution of the fitting error evolution of the Ω operator for 2000 iterations. Fitting errors for the outer and the inner curve in that figure are displayed in blue and red, respectively. Once again, the numerical errors and the visual appearance confirm the good performance of the method in these two cases as well. This example also shows that our approach has a great

flexibility, being able to deal with both open and closed curves such as the outer and inner curves of this model, respectively. Note also that even though the data points of the outer curve exhibit many changes of concavity, the method can approximate them very accurately with a single spline curve.

5.3. 3D CAD/CAM Workpiece. Figure 6 illustrates the application of our method to a complex CAD/CAM workpiece of a car body. This example aims at showing the ability of our method to perform well in a real industrial problem involving several geometric shapes. The figure shows two different views of a 3D automotive part comprised of 1610 curves stored in an industrial IGES file obtained from Candemat. As the reader can see, the shape consists of curves with very different topologies, ranging from simple regular shapes such as straight lines and conics to complicated irregular shapes. We therefore applied two different strategies for this example: simple regular shapes are reconstructed by using basic primitives (lines, circles, etc.) for which only some parameters have to be computed (e.g., using two data points for straight lines and three non-aligned data points for the center and radius of a circle), while the complicated shapes are reconstructed with our approach. A total of 243 curves have been reconstructed with our method in Figure 6 by using quadratic and cubic spline curves. The maximum, minimum, and average values of the RMSE fitting error are 3.1729×10^{-3} , 6.2547×10^{-4} , and 3.5842×10^{-6} , respectively. These error values confirm the good performance of our approach for a 3D real-world automotive part comprised of several (both open and closed) spline curves of different degrees.

6. Conclusions and Future Work

In this paper we introduce a new bioinspired method for computing a spline curve that approximates a given set of data points. This task involves many different variables which are interrelated with each other in a nonlinear way, leading to a continuous nonlinear optimization problem. Our approach solves this problem by combining two different procedures for the knots and the data parameters. For the former, an indirect approach that precomputes the knots instead of optimizing them is applied. This strategy leaves the most significant part of the optimization effort to the data parameterization. In our scheme, this task is addressed by a powerful bioinspired metaheuristic technique well suited for difficult continuous optimization problems, the firefly algorithm. Then, the knot vector is refined by using De Boor's method, thus yielding a better approximation to the optimal knot vector. The combination of the indirect approach and the firefly algorithm converts the original nonlinear continuous optimization problem into a convex optimization problem, solved by SVD. The proposed method has been applied to some illustrative real-world examples from the CAD/CAM field involving 2D and 3D open and closed curves. Our experimental results show that the proposed scheme can solve the original continuous nonlinear optimization problem very efficiently.

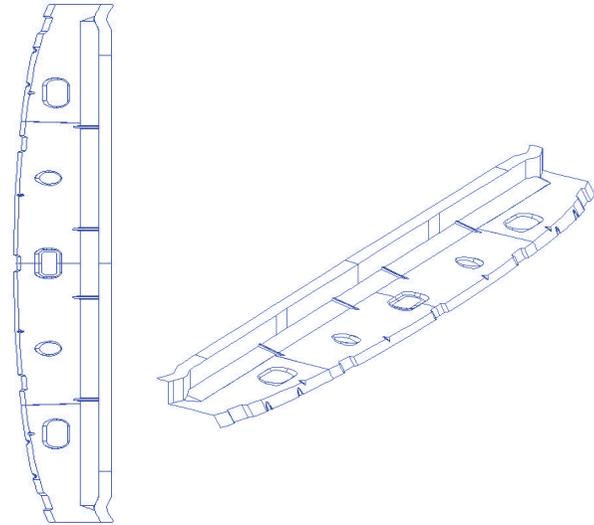


FIGURE 6: Two different views of a 3D CAD/CAM workpiece comprised of 1610 curves (courtesy of Candemat).

Regarding the future work, this method can be improved in several ways. As mentioned in Section 5, the indirect approach used in this paper implies that the knot vector is generally very good but not optimal, opening the door for further improvement. We think that a direct approach could lead to better results for the knot vector and, hence, for the overall method. Another interesting field of research is the use of some powerful modifications of the firefly algorithm which have been reported to return better results [63] or extend the capabilities of the standard version for continuous multiobjective optimization [64]. Also, the extension of this approach to other recently described bioinspired methods, such as the cuckoo search [65, 66] or the bat algorithm [67–69], might lead to further improvement of our results. They are all part of our plans for future work in the field.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper. Any commercial identity mentioned in this paper is cited solely for scientific purposes.

Acknowledgments

This research has been kindly supported by the Computer Science National Program of the Spanish Ministry of Economy and Competitiveness, Project Reference no. TIN2012-30768, Toho University (Funabashi, Japan), and the University of Cantabria (Santander, Spain). The authors are particularly grateful to the Department of Information Science of Toho University for all the facilities given to carry out this work.

References

- [1] N. R. Draper and H. Smith, *Applied Regression Analysis*, Wiley-Interscience, 3rd edition, 1998.
- [2] N. Molinari, J.-F. Durand, and R. Sabatier, "Bounded optimal knots for regression splines," *Computational Statistics and Data Analysis*, vol. 45, no. 2, pp. 159–178, 2004.
- [3] T. J. Jacobson and M. J. Murphy, "Optimized knot placement for B-splines in deformable image registration," *Medical Physics*, vol. 38, no. 8, pp. 4579–4582, 2011.
- [4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, 3rd edition, 2007.
- [5] R. E. Barnhill, *Geometric Processing for Design and Manufacturing*, SIAM, Philadelphia, Pa, USA, 1992.
- [6] G. Farin, *Curves and Surfaces for CAGD*, Morgan Kaufmann, San Francisco, Calif, USA, 5th edition, 2002.
- [7] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A. K. Peters, Wellesley, Mass, USA, 1993.
- [8] G. Dahlquist and A. Björck, *Numerical Methods*, Prentice Hall, 1974.
- [9] J. R. Rice, *The Approximation of Functions*, vol. 2, Addison-Wesley, Reading, Mass, USA, 1969.
- [10] M. Alhanaty and M. Bercovier, "Curve and surface fitting and design by optimal control methods," *CAD Computer Aided Design*, vol. 33, no. 2, pp. 167–182, 2001.
- [11] N. M. Patrikalakis and T. Maekawa, *Shape Interrogation for Computer Aided Design and Manufacturing*, Springer, 2002.
- [12] H. Pottmann, S. Leopoldseder, M. Hofer, T. Steiner, and W. Wang, "Industrial geometry: recent advances and applications in CAD," *CAD Computer Aided Design*, vol. 37, no. 7, pp. 751–766, 2005.
- [13] T. Várady, R. R. Martin, and J. Cox, "Reverse engineering of geometric models—an introduction," *CAD Computer Aided Design*, vol. 29, no. 4, pp. 255–268, 1997.
- [14] T. Várady and R. R. Martin, "Reverse engineering," in *Handbook of Computer Aided Geometric Design*, G. Farin, J. Hoschek, and M. Kim, Eds., Elsevier Science, 2002.
- [15] H. G. Burchard, "Splines (with optimal knots) are better," *Applicable Analysis*, vol. 3, no. 4, pp. 309–319, 1974.
- [16] C. A. de Boor and J. R. Rice, *Least Squares Cubic Spline Approximation—I: Fixed Knots*, CSD TR 20, Purdue University, Lafayette, Ind, USA, 1968.
- [17] C. A. de Boor and J. R. Rice, *Least Squares Cubic Spline Approximation—II: Variable Knots*, CSD TR 21, Purdue University, Lafayette, Ind, USA, 1968.
- [18] G. E. Hölzle, "Knot placement for piecewise polynomial approximation of curves," *Computer-Aided Design*, vol. 15, no. 5, pp. 295–296, 1983.
- [19] D. L. B. Jupp, "Approximation to data by splines with free knots," *SIAM Journal of Numerical Analysis*, vol. 15, pp. 328–343, 1978.
- [20] M. J. D. Powell, "Curve fitting by splines in one variable," in *Numerical Approximation to Functions and Data*, J. G. Hayes, Ed., Athlone Press, London, UK, 1970.
- [21] R. Goldenthal and M. Bercovier, "Spline curve approximation and design by optimal control over the knots," *Computing*, vol. 72, no. 1-2, pp. 53–64, 2004.
- [22] W. Li, S. Xu, G. Zhao, and L. P. Goh, "Adaptive knot placement in B-spline curve approximation," *CAD Computer Aided Design*, vol. 37, no. 8, pp. 791–797, 2005.
- [23] T. Lyche and K. Mørken, "Knot removal for parametric B-spline curves and surfaces," *Computer Aided Geometric Design*, vol. 4, no. 3, pp. 217–230, 1987.
- [24] W. Ma and J. Kruth, "Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces," *Computer-Aided Design*, vol. 27, no. 9, pp. 663–675, 1995.
- [25] T. Lyche and K. Mørken, "A data-reduction strategy for splines with applications to the approximation of functions and data," *IMA Journal of Numerical Analysis*, vol. 8, no. 2, pp. 185–208, 1988.
- [26] P. Dierckx, *Curve and Surface Fitting with Splines*, Oxford University Press, Oxford, UK, 1993.
- [27] F. Yoshimoto, T. Harada, and Y. Yoshimoto, "Data fitting with a spline using a real-coded genetic algorithm," *CAD Computer Aided Design*, vol. 35, no. 8, pp. 751–760, 2003.
- [28] P. Gu and X. Yan, "Neural network approach to the reconstruction of freeform surfaces for reverse engineering," *Computer-Aided Design*, vol. 27, no. 1, pp. 59–64, 1995.
- [29] M. Hoffmann, "Numerical control of kohonen neural network for scattered data approximation," *Numerical Algorithms*, vol. 39, no. 1-3, pp. 175–186, 2005.
- [30] G. K. Knopf and J. Kofman, "Free-form surface reconstruction using Bernstein basis function networks," in *Intelligent Engineering Systems through Artificial Neural Networks*, C. H. Dagli, Ed., vol. 9, pp. 797–802, ASME Press, 1999.
- [31] E. Castillo and A. Iglesias, "Some characterizations of families of surfaces using functional equations," *ACM Transactions on Graphics*, vol. 16, no. 3, pp. 296–318, 1997.
- [32] G. Echevarría, A. Iglesias, and A. Gálvez, "Extending neural networks for B-spline surface reconstruction," *Lectures Notes in Computer Science*, vol. 2330, pp. 305–314, 2002.
- [33] A. Iglesias, G. Echevarría, and A. Gálvez, "Functional networks for B-spline surface reconstruction," *Future Generation Computer Systems*, vol. 20, no. 8, pp. 1337–1353, 2004.
- [34] A. Iglesias and A. Gálvez, "A new artificial intelligence paradigm for computer aided geometric design," *Lectures Notes in Artificial Intelligence*, vol. 1930, pp. 200–213, 2001.
- [35] L. N. de Castro and F. J. Von Zuben, Eds., *Recent Developments in Biologically Inspired Computing*, Idea Group Publishing, Hershey, Pa, USA, 2005.
- [36] D. Floreano and C. Matthiussi, *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*, MIT Press, Cambridge, Mass, USA, 2008.
- [37] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.
- [38] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York, NY, USA, 1999.
- [39] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, John Wiley and Sons, Chichester, UK, 2005.
- [40] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, Frome, UK, 2nd edition, 2010.
- [41] A. Gálvez, A. Cobo, J. Puig-Pey, and A. Iglesias, "Particle swarm optimization for bézier surface reconstruction," *Lecture Notes in Computer Science*, vol. 5102, no. 2, pp. 116–125, 2008.
- [42] A. Gálvez and A. Iglesias, "Efficient particle swarm optimization approach for data fitting with free knot B-splines," *CAD Computer Aided Design*, vol. 43, no. 12, pp. 1683–1692, 2011.
- [43] A. Gálvez and A. Iglesias, "Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points," *Information Sciences*, vol. 192, pp. 174–192, 2012.

- [44] A. Gálvez, A. Iglesias, and J. Puig-Pey, "Iterative two-step genetic-algorithm-based method for efficient polynomial B-spline surface reconstruction," *Information Sciences*, vol. 182, no. 1, pp. 56–76, 2012.
- [45] F. Yoshimoto, M. Moriyama, and T. Harada, "Automatic knot placement by a genetic algorithm for data fitting with a spline," in *Proceedings of the Shape Modeling International*, pp. 162–169, IEEE Computer Society Press, 1999.
- [46] A. G. Gálvez, A. Iglesias, and A. Avila, "Immunological-based approach for accurate fitting of 3D noisy points with Bézier surfaces," in *Proceedings of the International Conference on Computer Science (ICCS '013)*, vol. 18, pp. 50–59, 2013.
- [47] E. Ülker and A. Arslan, "Automatic knot adjustment using an artificial immune system for B-spline curve approximation," *Information Sciences*, vol. 179, no. 10, pp. 1483–1494, 2009.
- [48] X. Zhao, C. Zhang, B. Yang, and P. Li, "Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation," *CAD Computer Aided Design*, vol. 43, no. 6, pp. 598–604, 2011.
- [49] A. Gálvez, A. Iglesias, A. Cobo, J. Puig-Pey, and J. Espinola, "Bézier curve and surface fitting of 3D point clouds through genetic algorithms, functional networks and least-squares approximation," *Lecture Notes in Computer Science*, vol. 4706, no. 2, pp. 680–693, 2007.
- [50] A. Gálvez and A. Iglesias, "A new iterative mutually-coupled hybrid GA-PSO approach for curve fitting in manufacturing," *Applied Soft Computing*, vol. 13, no. 3, pp. 1491–1504, 2013.
- [51] M. Sarfraz and S. A. Raza, "Capturing outline of fonts using genetic algorithms and splines," in *Proceedings of the 5th International Conference on Information Visualization (IV '01)*, pp. 738–743, IEEE Computer Society Press, 2001.
- [52] C. A. de Boor, *Practical Guide to Splines*, Springer, 2001.
- [53] L. Piegl and W. Tiller, *The NURBS Book*, Springer, Berlin, Germany, 1997.
- [54] X.-S. Yang, "Firefly algorithms for multimodal optimization," *Lecture Notes in Computer Science*, vol. 5792, pp. 169–178, 2009.
- [55] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.
- [56] X. S. Yang, "Review of meta-heuristics and generalised evolutionary walk algorithm," *International Journal of Bio-Inspired Computation*, vol. 3, no. 2, pp. 77–84, 2011.
- [57] X. S. Yang and S. Deb, "Eagle strategy using Lévy walk, and firey algorithms for stochastic optimization," in *Nature Inspired Cooperative Strategies for Optimization (NICSO)*, pp. 101–111, 2010.
- [58] X.-S. Yang, S. S. S. Hosseini, and A. H. Gandomi, "Firefly Algorithm for solving non-convex economic dispatch problems with valve loading effect," *Applied Soft Computing Journal*, vol. 12, no. 3, pp. 1180–1186, 2012.
- [59] X. S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, John Wiley & Sons, 2010.
- [60] I. Fister, I. Fister Jr, X. S. Yang, and J. Brest, "Memetic self-adaptive firefly algorithm," *Swarm and Evolutionary Computation*, 2013.
- [61] A. Gálvez and A. Iglesias, "Firey algorithm for polynomial Bézier surface parameterization," *Journal of Applied Mathematics*, vol. 2013, Article ID 237984, 9 pages, 2013.
- [62] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes*, Cambridge University Press, Cambridge, UK, 2nd edition, 1992.
- [63] I. Fister, X. S. Yang, J. Brest, and I. Fister Jr., "Memetic self-adaptive firefly algorithm," in *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, X. S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu, Eds., pp. 73–102, Elsevier, 2013.
- [64] X. S. Yang, "Multi-objective firey algorithm for continuous optimization," *Engineering with Computers*, vol. 29, pp. 175–184, 2013.
- [65] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.
- [66] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, December 2009.
- [67] X.-S. Yang, "A new metaheuristic Bat-inspired Algorithm," *Studies in Computational Intelligence*, vol. 284, pp. 65–74, 2010.
- [68] X. S. Yang, "Bat algorithm for multi-objective optimisation," *International Journal of BioInspired Computation*, vol. 3, pp. 267–274, 2011.
- [69] X. S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.

Research Article

An Effective Hybrid Firefly Algorithm with Harmony Search for Global Numerical Optimization

Lihong Guo,¹ Gai-Ge Wang,² Heqi Wang,¹ and Dinan Wang¹

¹ Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China

² School of Computer Science and Technology, Jiangsu Normal University, Xuzhou, Jiangsu 221116, China

Correspondence should be addressed to Gai-Ge Wang; gaigewang@163.com

Received 10 August 2013; Accepted 29 September 2013

Academic Editors: Z. Cui and X. Yang

Copyright © 2013 Lihong Guo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A hybrid metaheuristic approach by hybridizing harmony search (HS) and firefly algorithm (FA), namely, HS/FA, is proposed to solve function optimization. In HS/FA, the exploration of HS and the exploitation of FA are fully exerted, so HS/FA has a faster convergence speed than HS and FA. Also, top fireflies scheme is introduced to reduce running time, and HS is utilized to mutate between fireflies when updating fireflies. The HS/FA method is verified by various benchmarks. From the experiments, the implementation of HS/FA is better than the standard FA and other eight optimization methods.

1. Introduction

In engineering problems, optimization is to look for a vector that can maximize and minimize a function. Nowadays, stochastic method is generally utilized to cope with optimization problems [1]. Though there are many ways to classify them, a simple one is used to divide them into two groups according to their nature: deterministic and stochastic. Deterministic algorithms can get the same solutions if the initial conditions are unchanged, because they always follow the rigorous move. However, regardless of the initial values, stochastic ones are based on certain stochastic distribution; therefore they generally generate various solutions. In fact, both of them can find satisfactory solutions after some generations. Recently, nature-inspired algorithms are well capable of solving numerical optimization problems more efficiently.

These metaheuristic approaches are developed to solve complicated problems, like permutation flow shop scheduling [2], reliability [3, 4], high-dimensional function optimization [5], and other engineering problems [6, 7]. In the 1950s, nature evolution was idealized as an optimization technology and this made a new type of approach, namely, genetic algorithms (GAs) [8]. After that, many other metaheuristic methods have appeared, like evolutionary strategy

(ES) [9, 10], ant colony optimization (ACO) [11], probability-based incremental learning (PBIL) [12], big bang-big crunch algorithm [13–16], harmony search (HS) [17–19], charged system search (CSS) [20], artificial physics optimization [21], bat algorithm (BA) [22, 23], animal migration optimization (AMO) [24], krill herd (KH) [25–27], differential evolution (DE) [28–31], particle swarm optimization (PSO) [32–35], stud GA (SGA) [36], cuckoo search (CS) [37, 38], artificial plant optimization algorithm (APOA) [39], biogeography-based optimization (BBO) [40], and FA method [41, 42].

As a global optimization method, FA [42] is firstly proposed by Yang in 2008, and it is originated from the fireflies swarm. Recent researches demonstrate that the FA is quite powerful and relatively efficient [43]. Furthermore, the performance of FA can be improved with feasible promising results [44]. In addition, nonconvex problems can be solved by FA [45]. A summarization of swarm intelligence containing FA is given by Parpinelli and Lopes [46].

On the other hand, HS [17, 47] is a novel heuristic technique for optimization problems. In engineering optimization, the engineers make an effort to find an optimum that can be decided by an objective function. While, in the music improvisation process, musicians search for most satisfactory harmony as decided by aesthetician. HS method originates in the similarity between them [1].

In most cases, FA can find the optimal solution with its exploitation. However, the search used in FA is based on randomness, so it cannot always get the global best values. On the one hand, in order to improve diversity of fireflies, an improvement of adding HS is made to the FA, which can be treated as a mutation operator. By combining the principle of HS and FA, an enhanced FA is proposed to look for the best objective function value. On the other hand, FA needs much more time to search for the best solution and its performance significantly deteriorates with the increases in population size. In HS/FA, top fireflies scheme is introduced to reduce running time. This scheme is carried out by reduction of outer loop in FA. Through top fireflies scheme, the time complexity of HS/FA decreases from $O(NP^2)$ to $O(KEEP*NP)$, where KEEP is the number of top fireflies. The proposed approach is evaluated on various benchmarks. The results demonstrate that the HS/FA performs more effectively and accurately than FA and other intelligent algorithms.

The rest of this paper is structured below. To begin with, a brief background on the HS and FA is provided in Sections 2 and 3, respectively. Our proposed HS/FA is presented in Section 4. HS/FA is verified through various functions in Section 5, and Section 6 presents the general conclusions.

2. HS Method

As a relative optimization technique, there are four optimization operators in HS [17, 48, 49]:

- HM: the harmony memory, as shown in (1);
- HMS: the harmony memory size,
- HMCR: the harmony memory consideration rate,
- PAR: the pitch adjustment rate, and
- bw: the pitch adjustment bandwidth [1].

Consider

$$HM = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \cdots & x_D^1 & \text{fitness}(x^1) \\ x_1^2 & x_2^2 & \cdots & x_D^2 & \text{fitness}(x^2) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_D^{HMS} & \text{fitness}(x^{HMS}) \end{array} \right]. \quad (1)$$

The HS method can be explained according to the discussion of the player improvisation process. There are 3 feasible options for a player in the music improvisation process: (1) play several pitches that are the same with the HMCR; (2) play some pitches like a known piece; or (3) improvise new pitches [1]. These three options can be idealized into three components: use of HM, pitch adjusting, and randomization [1].

Similar to selecting the optimal ones in GA, the first part is important as it is [1]. This can guarantee that the optimal harmonies will not be destroyed in the HM. To make HS more powerful, the parameter HMCR should be properly set [1]. Through several experiments, in most cases, HMCR = 0.7~0.95.

TABLE 1: Benchmark functions.

No.	Name	No.	Name
F01	Beale	F19	Holzman 2 function
F02	Bohachevsky #1	F20	Levy
F03	Bohachevsky #2	F21	Pathological function
F04	Bohachevsky #3	F22	Penalty #1
F05	Booth	F23	Penalty #2
F06	Branin	F24	Powell
F07	Easom	F25	Quartic with noise
F08	Foxholes	F26	Rastrigin
F09	Freudenstein-Roth	F27	Rosenbrock
F10	Goldstein-Price	F28	Schwefel 2.26
F11	Hump	F29	Schwefel 1.2
F12	Matyas	F30	Schwefel 2.22
F13	Ackley	F31	Schwefel 2.21
F14	Alpine	F32	Sphere
F15	Brown	F33	Step
F16	Dixon and Price	F34	Sum function
F17	Fletcher-Powell	F35	Zakharov
F18	Griewank	F36	Wavy1

The pitch in the second part needs to be adjusted slightly; and hence a proper method is used to adjust the frequency [1]. If the new pitch x_{new} is updated by

$$x_{\text{new}} = x_{\text{old}} + bw(2\varepsilon - 1), \quad (2)$$

where ε is a random number in $[0, 1]$ and x_{old} is the current pitch. Here, bw is the bandwidth.

Parameter PAR should also be appropriately set. If PAR is very close to 1, then the solution is always updating and HS is hard to converge. If it is next to 0, then little change is made and HS may be premature. So, here we set PAR = 0.1~0.5 [1].

To improve the diversity, the randomization is necessary as shown in the third component. The usage of randomization allows the method to go a step further into promising area so as to find the optimal solution [1].

The HS can be presented in Algorithm 1. Where D is the number of decision variables. rand is a random real number in interval $(0, 1)$ drawn from uniform distribution.

3. FA Method

FA [42] is a metaheuristic approach for optimization problems. The search strategy in FA comes from the fireflies swarm behavior [50]. There are two significant issues in FA that are the formulation of attractiveness and variation of light intensity [42].

For simplicity, several characteristics of fireflies are idealized into three rules described in [51]. Based on these three rules, the FA can be described in Algorithm 2.

For two fireflies x_i and x_j , they can be updated as follows:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2} (x_i^t - x_j^t) + \alpha \varepsilon_i^t, \quad (3)$$

where α is the step size, β_0 is the attractiveness at $r = 0$, the second part is the attraction, while the third is randomization

```

Begin
  Step 1. Initialize the HM.
  Step 2. Evaluate the fitness.
  Step 3. while the halting criteria is not satisfied do
    for  $d = 1 : D$  do
      if  $\text{rand} < \text{HMCR}$  then // memory consideration
         $x_{\text{new}}(d) = x_a(d)$  where  $a \in (1, 2, \dots, \text{HMS})$ 
      if  $\text{rand} < \text{PAR}$  then // pitch adjustment
         $x_{\text{new}}(d) = x_{\text{old}}(d) + bw \times (2 \times \text{rand} - 1)$ 
      endif
    else // random selection
       $x_{\text{new}}(d) = x_{\text{min},d} + \text{rand} \times (x_{\text{max},d} - x_{\text{min},d})$ 
    endif
  endfor  $d$ 
  Update the HM as  $x_w = x_{\text{new}}$ , if  $f(x_{\text{new}}) < f(x_w)$  (minimization objective)
  Update the best harmony vector
  Step 4. end while
  Step 5. Output results.
End.

```

ALGORITHM 1: HS method.

```

Begin
  Step 1. Initialization. Set  $G = 1$ ; define  $\gamma$ ; set step size  $\alpha$  and  $\beta_0$  at  $r = 0$ .
  Step 2. Evaluate the light intensity  $I$  determined by  $f(x)$ 
  Step 3. While  $G < \text{MaxGeneration}$  do
    for  $i = 1 : \text{NP}$  (all NP fireflies) do
      for  $j = 1 : \text{NP}$  (NP fireflies) do
        if  $(I_j < I_i)$ ,
          move firefly  $i$  towards  $j$ ;
        end if
        Update attractiveness;
        Update light intensity;
      end for  $j$ 
    end for  $i$ 
     $G = G + 1$ ;
  Step 4. end while
  Step 5. Output the results.
End.

```

ALGORITHM 2: Firefly algorithm. FA method.

[50]. In our present work, we take $\beta_0 = 1$, $\alpha \in [0, 1]$, and $\gamma = 1$ [50].

4. HS/FA

Based on the introduction of HS and FA in the previous section, the combination of the two approaches is described and HS/FA is proposed, which updates the poor solutions to accelerate its convergence speed.

HS and FA are adept at exploring the search space and exploiting solution, respectively. Therefore, in the present work, a hybrid by inducing HS into FA method named HS/FA is utilized to deal with optimization problem, which can be considered as mutation operator. By this strategy, the mutation of the HS and FA can explore the new search

space and exploit the population, respectively. Therefore, it can overcome the lack of the exploration of the FA.

To combat the random walks used in FA, in the present work, the addition of mutation operator is introduced into the FA, including two detailed improvements.

The first one is the introduction of top fireflies scheme into FA to reduce running time that is analogous to the elitism scheme frequently used in other population-based optimization algorithms. In FA, due to dual loop, time complexity is $O(\text{NP}^2)$, whose performance significantly deteriorates with the increases in population size. This improvement is carried out by reduction of outer loop in FA. In HS/FA, we select the special firefly with optimal or near-optimal fitness (i.e., the brightest fireflies) to form top fireflies, and all the fireflies only move towards top fireflies. Through top fireflies scheme,

```

Begin
  Step 1. Initialization. Set  $t = 1$ ; define  $\gamma$ ; set  $\alpha, \beta_0$  at  $r = 0$ ; set HMCR and PAR; set the
  number of top fireflies KEEP.
  Step 2. Evaluate the light intensity  $I$ .
  Step 3. While  $t < \text{MaxGeneration}$  do
    Sort the fireflies by light intensity  $I$ ;
    for  $i = 1 : \text{KEEP}$  (all Top fireflies) do
      for  $j = 1 : \text{NP}$  (all fireflies) do
        if ( $I_j < I_i$ ) then
          Move firefly  $i$  towards  $j$ ;
        else
          for  $k = 1 : D$  (all elements) do // Mutate
            if ( $\text{rand} < \text{HMCR}$ ) then
               $r_1 = \lceil \text{NP} * \text{rand} \rceil$ 
               $x_{r_1}(k) = x_{r_1}(k)$ 
              if ( $\text{rand} < \text{PAR}$ ) then
                 $x_v(k) = x_v(k) + bw \times (2 \times \text{rand} - 1)$ 
              end if
            else
               $x_v(k) = x_{\min,k} + \text{rand} \times (x_{\max,k} - x_{\min,k})$ 
            end if
          end for  $k$ 
        end if
        Update attractiveness;
        Update light intensity;
      end for  $j$ 
    end for  $i$ 
    Evaluate the light intensity  $I$ .
    Sort the population by light intensity  $I$ ;
     $t = t + 1$ ;
  Step 4. end while
End.

```

ALGORITHM 3: HS/FA method.

the time complexity of HS/FA decreases from $O(\text{NP}^2)$ to $O(\text{KEEP} * \text{NP})$, where KEEP is the number of top fireflies. In general, KEEP is far smaller than NP, so the time used by HS/FA is much less than FA. Apparently, if $\text{KEEP} = \text{NP}$, the algorithm HS/FA is declined to the standard FA. If KEEP is too small, only few best fireflies are selected to form top fireflies and it converges too fast, moreover, may be premature for lack of diversity. If KEEP is extremely big (near NP), almost all the fireflies are used to form top fireflies, so all fireflies are explored well, leading to potentially optimal solutions, while the algorithm performs badly and converges too slowly. Therefore, we use $\text{KEEP} = 2$ in our study.

The second is the addition of HS serving as mutation operator striving to improve the population diversity to avoid the premature convergence. In standard FA, if firefly i is brighter than firefly j , firefly j will move towards firefly i , and then evaluate newly-generated fireflies and update light intensity. If not, firefly j does nothing. However, in HS/FA, if firefly i is not brighter than firefly j , firefly j is updated by mutation operation to improve the light intensity for firefly j . More concretely, for the global search part, with respect to HS/FA, we tune every element x_{k_j} ($k = 1, 2, \dots, D$) in x_j (the position of firefly j) using HS. When ξ_1 is not less

than HMCR, that is, $\xi_1 \geq \text{HMCR}$, the element x_{k_j} is updated randomly; whereas when $\xi_1 < \text{HMCR}$, we update the element x_{k_j} in accordance with x_{r_1} . Under this circumstance, pitch adjustment operation in HS is applied to update the element x_{k_j} if $\xi_2 < \text{PAR}$ to increase population diversity, as shown in (2), where ξ_1 and ξ_2 are two uniformly distributed random numbers in $[0, 1]$, r_1 is the integer number in $[1, \text{NP}]$, and NP is population size.

In sum, the detailed presentation of HS/FA can be given in Algorithm 3.

5. The Results

The HS/FA method is tested on optimization problems through several simulations conducted in test problems. To make a fair comparison between different methods, all the experiments were conducted on the same conditions described in [1].

In this section, HS/FA is compared on optimization problems with other nine methods, which are ACO [11], BBO [40], DE [28–30], ES [9, 10], FA [41, 42], GA [8], HS [17–19], PSO [32, 52], and SGA [36]. Here, for HS, FA, and HS/FA, the parameters are set as follows: absorption coefficient $\gamma =$

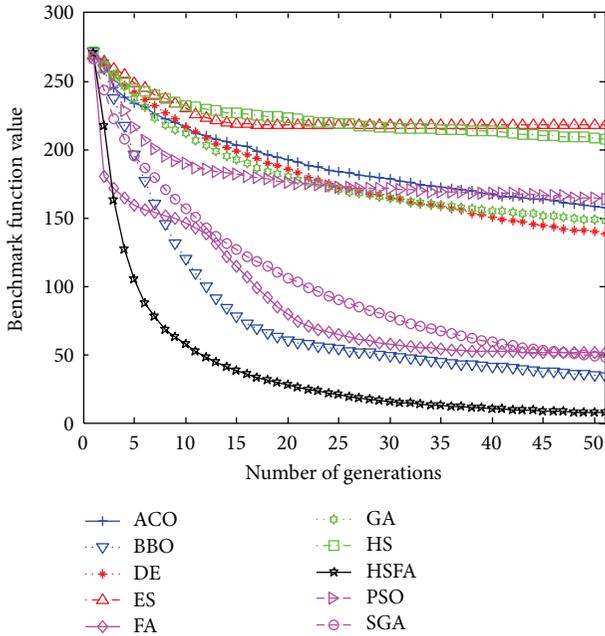


FIGURE 1: Performance comparison for the F26 Rastrigin function.

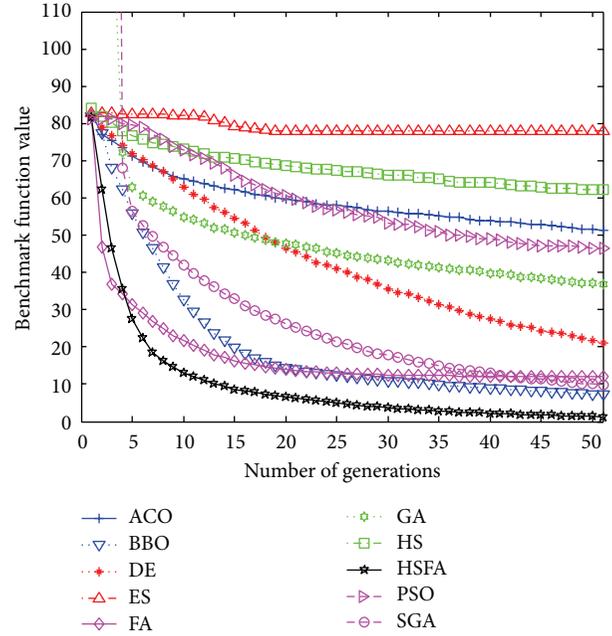


FIGURE 3: Performance comparison for the F30 Schwefel 2.22 function.

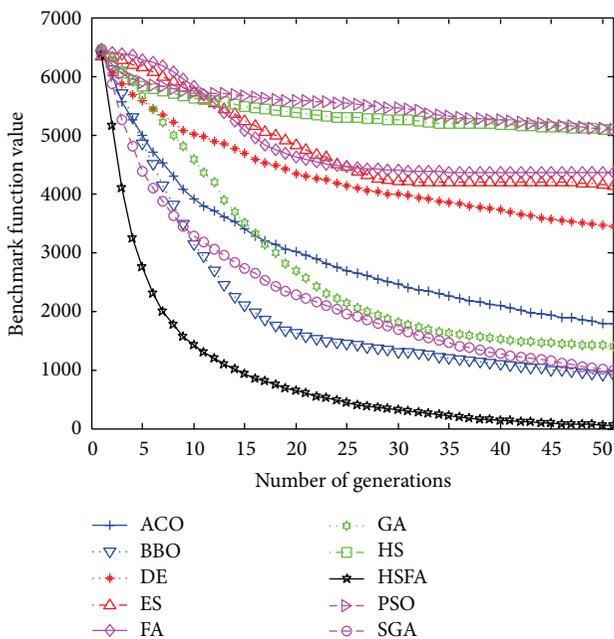


FIGURE 2: Performance comparison for the F28 Schwefel 2.26 function.

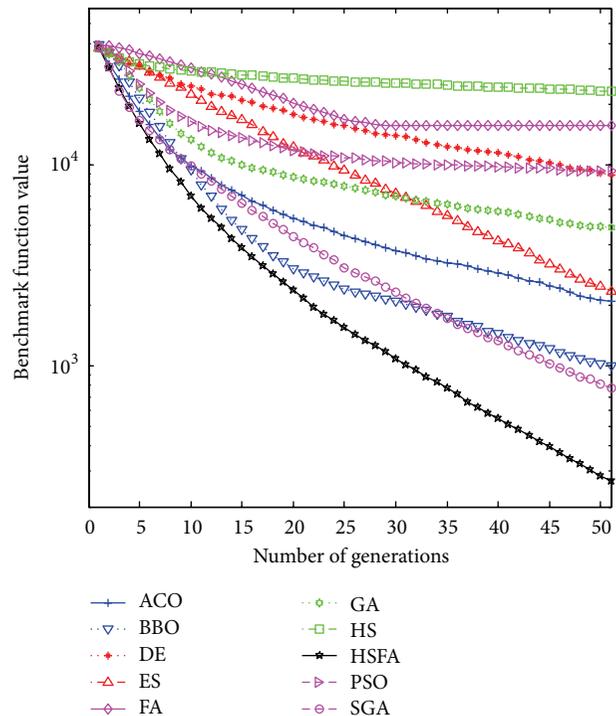


FIGURE 4: Performance comparison for the F33 step function.

1.0, the HMCR = 0.9, and the PAR = 0.1. For parameters used in other methods, they can be referred to as in [48, 53]. Thirty-six functions are utilized to verify our HS/FA method, which can be shown in Table 1. More knowledge of all the benchmarks can be found in [54].

Because all the intelligent algorithms always have some randomness, in order to get representative statistical features,

we did 500 implementations of each method on each problem. Tables 2 and 3 illustrate the average and best results found by each algorithm, respectively. Note that we have used two different scales to normalize the values in the tables, and its detailed process can be found in [54]. The dimension of each function is set to 30.

TABLE 2: Mean normalized optimization results.

	ACO	BBO	DE	ES	FA	GA	HS	HSFA	PSO	SGA
F01	1.01	1.01	1.00	1.02	1.08	1.04	1.07	1.00	1.01	1.25
F02	1.43	2.71	1.00	2.55	1.00	1.39	16.66	1.00	3.13	1.22
F03	1.25	1.84	1.00	2.28	1.00	1.17	11.77	1.01	3.50	1.26
F04	3.9E4	1.7E5	49.66	2.8E5	1.00	4.0E4	2.0E6	1.5E3	3.7E5	2.5E5
F05	1.01	1.02	1.00	1.11	1.00	1.01	1.15	1.00	1.05	1.19
F06	1.03	1.02	1.00	1.09	1.00	1.02	1.03	1.00	1.03	3.01
F07	2.40	2.48	2.27	2.35	2.23	1.83	1.88	1.00	1.71	2.99
F08	1.72	1.72	1.72	1.72	1.72	1.72	1.72	1.72	1.72	1.00
F09	1.03	1.01	1.00	2.05	17.29	1.00	6.55	1.00	1.04	1.24
F10	2.40	2.40	2.40	3.06	2.40	2.40	3.09	2.40	2.70	1.00
F11	1.00	1.00	1.00	1.03	1.00	1.00	1.03	1.00	1.02	1.25
F12	1.00	1.00	1.00	1.01	1.00	1.00	1.02	1.00	1.00	1.14
F13	4.32	2.56	3.68	5.56	1.39	4.98	5.70	1.00	4.83	2.63
F14	36.17	7.98	43.16	73.74	11.68	33.13	70.32	1.00	53.62	8.48
F15	570.98	14.02	27.90	1.1E3	141.86	99.02	652.65	1.00	485.92	12.10
F16	1.6E3	75.20	31708	1.2E4	7.35	942.08	1.1E4	1.00	1.6E3	26.84
F17	21.98	2.35	7.67	21.63	5.30	7.33	19.01	1.00	15.46	2.26
F18	8.49	5.40	14.18	66.70	2.31	28.49	139.02	1.00	52.77	5.69
F19	2.8E3	167.15	544.18	1.9E4	25.71	1.3E3	1.9E4	1.00	2.7E3	39.88
F20	93.79	13.59	68.16	276.60	20.05	92.42	282.65	1.00	173.17	9.33
F21	3.20	2.49	1.74	1.00	3.69	2.65	3.88	1.78	2.55	2.42
F22	1.2E8	9.7E3	2.8E5	5.1E7	6.64	5.8E5	7.8E7	1.00	7.9E6	9.81
F23	2.2E7	2.9E4	3.1E5	1.4E7	7.36	6.7E5	2.2E7	1.00	3.5E6	5.0E3
F24	112.59	8.00	48.08	188.98	1.04	25.76	133.99	1.00	52.91	2.92
F25	1.2E3	103.34	63738	1.8E4	17.91	1.4E3	1.8E4	1.00	4.1E3	62.77
F26	24.37	4.58	21.06	32.51	7.75	20.84	29.89	1.00	23.03	7.29
F27	37.23	2.38	5.34	49.70	1.00	10.38	34.12	1.04	12.06	2.00
F28	37.76	18.45	73.51	92.92	93.82	31.93	109.20	1.00	112.28	21.21
F29	4.79	2.52	6.72	7.41	1.00	5.40	7.13	1.93	4.81	4.31
F30	42.08	6.33	16.76	63.65	9.36	30.16	53.57	1.00	35.27	8.32
F31	2.98	3.13	3.87	4.56	1.00	3.93	4.78	1.15	3.97	2.79
F32	205.80	13.56	37.41	382.80	1.87	131.27	361.49	1.00	151.62	14.65
F33	40.44	20.26	53.05	312.01	5.14	111.20	471.25	1.00	194.10	15.72
F34	274.21	27.02	46.57	546.26	6.17	138.85	550.25	1.00	188.96	25.75
F35	1.2E5	1.46	3.32	3.57	1.18	3.12	3.34	1.00	3.12	2.64
F36	9.82	5.26	14.12	29.95	10.67	16.90	35.57	1.00	23.95	5.37

The bold data are the best function value among different methods for the specified function.

From Table 2, on average, HS/FA is well capable of finding function minimum on twenty-eight of the thirty-six functions. FA performs the second best on ten of the thirty-six functions. Table 3 shows that HS/FA and FA perform the same and best on twenty-two of the thirty-six and seventeen functions, respectively. ACO, DE, and GA perform the best on eight benchmarks. From the above tables, we can see that, for low-dimensional functions, both FA and HS/FA perform well, and their performance has little difference between each other.

Further, convergence graphs of ten methods for most representative functions are illustrated in Figures 1, 2, 3, and 4 which indicate the optimization process. The values here are the real mean function values from above experiments.

F26 is a complicated multimodal function and it has a single global value 0 and several local optima. Figure 1 shows that HS/FA converges to global value 0 with the fastest speed. Here FA converges a little faster initially, but it is likely to be trapped into subminima as the function value decreases slightly.

F28 is also a multimodal problem and it has only a global value 0. For this problem, HS/FA is superior to the other nine methods and finds the optimal value earliest.

For this function, the figure illustrates that HS/FA significantly outperforms all others in the optimization process. At last, HS/FA converges to the best solution superiorly to others. BBO is only inferior to HS/FA and performs the second best for this case.

TABLE 3: Best normalized optimization results.

	ACO	BBO	DE	ES	FA	GA	HS	HSFA	PSO	SGA
F01	1.00									
F02	1.00	1.71	1.00	1.53	1.00	1.00	1.49	1.00	1.72	1.00
F03	1.00	1.28	1.00	1.12	1.00	1.00	1.28	1.00	1.34	1.13
F04	2.0E14	2.0E14	3.2E10	2.3E15	2.5E8	1.00	1.1E15	3.8E11	1.0E15	4.5E15
F05	1.00	1.00	1.00	1.02	1.00	1.00	1.00	1.00	1.00	1.00
F06	1.01	1.01	1.00	1.01	1.00	1.01	1.00	1.00	1.00	2.51
F07	2.5E6	3.3E6	7.2E5	1.6E6	1.00	1.7E4	2.1E5	2.88	3.7E5	3.3E6
F08	1.99	1.99	1.99	1.99	1.99	1.99	1.99	1.99	1.99	1.00
F09	1.00	1.00	1.00	1.01	1.00	1.00	1.00	1.00	1.00	1.00
F10	2.65	2.65	2.65	2.74	2.65	2.65	2.65	2.65	2.65	1.00
F11	1.00	1.03								
F12	1.00									
F13	8.34	3.93	7.05	11.02	1.00	8.71	11.56	1.54	9.51	3.98
F14	63.46	11.42	88.82	159.53	12.22	40.65	144.04	1.00	102.48	10.66
F15	218.30	9.54	47.47	591.91	33.50	128.64	792.73	1.00	358.64	9.44
F16	4.7E3	109.24	1.3E3	4.0E4	1.00	315.53	6.1E4	2.31	5.9E3	43.61
F17	42.58	3.24	11.83	41.06	1.20	9.50	43.49	1.00	29.07	3.02
F18	7.99	3.62	13.85	63.26	1.00	14.42	160.01	1.08	37.87	2.32
F19	3.1E3	135.16	893.82	3.9E4	1.56	566.65	5.4E4	1.00	6.1E3	3.67
F20	251.29	32.94	142.30	720.64	7.96	131.48	863.43	1.00	483.70	23.66
F21	3.96	2.89	1.65	1.00	4.60	2.91	4.87	1.90	2.72	2.37
F22	31.83	55.55	1.5E5	1.3E8	8.26	89.30	3.0E8	1.00	5.8E6	15.15
F23	1.00	4.4E3	1.5E6	8.8E7	27.10	3.3E4	1.6E8	4.89	2.8E7	29.22
F24	2.2E3	88.70	1.1E3	4.7E3	1.60	215.01	3.4E3	1.00	940.82	34.50
F25	3.0E3	380.11	2.9E3	1.0E5	1.00	3.1E3	1.3E5	2.01	2.4E4	54.67
F26	39.66	5.65	30.04	58.39	6.03	27.36	42.32	1.00	38.57	8.91
F27	54.77	2.13	12.53	87.36	1.27	10.85	58.14	1.00	21.11	2.77
F28	164.45	67.82	335.75	447.01	430.29	85.82	596.00	1.00	551.44	68.85
F29	8.78	4.00	16.50	15.31	1.00	10.85	18.42	3.27	6.75	7.29
F30	63.53	10.38	27.71	105.79	7.15	47.04	88.91	1.00	58.02	12.83
F31	3.80	5.63	7.23	9.39	1.00	7.31	10.20	1.93	7.56	4.53
F32	740.24	30.59	184.72	1.8E3	1.00	322.80	1.8E3	2.87	725.62	31.00
F33	149.29	66.43	224.57	1.4E3	3.86	255.71	2.4E3	1.00	1.0E3	42.71
F34	491.51	35.62	100.26	1.1E3	1.64	113.66	1.1E3	1.00	400.61	27.55
F35	3.44	2.50	5.89	6.67	1.00	4.28	5.48	1.46	3.83	3.74
F36	11.05	6.01	18.56	40.10	9.07	18.47	43.18	1.00	31.18	4.64

The bold data are the best function value among different methods for the specified function.

HS/FA significantly outperforms all others in the optimization process. Furthermore, Figure 4 indicates that, at the early stage of the optimization process, FA converges faster than HS/FA, while HS/FA is well capable of improving its solution steadily in the long run. Here FA shows faster converges initially (within 20 iterations), however it seems to be trapped into subminima as the function value decreases slightly (after 20 iterations), and it is outperformed by HS/FA after 30 iterations.

From Figures 1–4, our HS/FA’s performance is far better than the others. In general, BBO and FA, especially FA, are only inferior to the HS/FA. Note that, in [40], BBO is compared with seven EAs and an engineering problem. The

experiments proved the excellent performance of BBO. It is also indirectly proven that our HS/FA is a more effective optimization method than others.

6. Conclusions

In the present work, a hybrid HS/FA was proposed for optimization problems. FA is enhanced by the combination of the basic HS method. In HS/FA, top fireflies scheme is introduced to reduce running time; the other is used to mutate between fireflies when updating fireflies. The new harmony vector takes the place of the new firefly only if it is better than before, which generally outperforms HS

and FA. The HS/FA strive to exploit merits of the FA and HS so as to escape all fireflies being trapped into local optima. Benchmark evaluation on the test problems is used to investigate the HS/FA and the other nine approaches. The results demonstrated that HS/FA is able to make use of the useful knowledge more efficiently to find much better values compared with the other optimization algorithms.

References

- [1] G. Wang and L. Guo, "A novel hybrid bat algorithm with harmony search for global numerical optimization," *Journal of Applied Mathematics*, vol. 2013, Article ID 696491, 21 pages, 2013.
- [2] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Advances in Engineering Software*, vol. 55, pp. 10–31, 2013.
- [3] D. Zou, L. Gao, S. Li, and J. Wu, "An effective global harmony search algorithm for reliability problems," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4642–4648, 2011.
- [4] D. Zou, L. Gao, J. Wu, S. Li, and Y. Li, "A novel global harmony search algorithm for reliability problems," *Computers and Industrial Engineering*, vol. 58, no. 2, pp. 307–316, 2010.
- [5] X.-S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karanoglu, *Swarm Intelligence and Bio-Inspired Computation*, Elsevier, Waltham, Mass, USA, 2013.
- [6] A. H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, *Metaheuristic Applications in Structures and Infrastructures*, Elsevier, Waltham, Mass, USA, 2013.
- [7] X. S. Yang, A. H. Gandomi, S. Talatahari, and A. H. Alavi, *Metaheuristics in Water, Geotechnical and Transport Engineering*, Elsevier, Waltham, Mass, USA, 2013.
- [8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston, Mass, USA, 1989.
- [9] T. Back, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, Oxford, UK, 1996.
- [10] H. Beyer, *The Theory of Evolution Strategies*, Springer, New York, NY, USA, 2001.
- [11] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MIT Press, Cambridge, UK, 2004.
- [12] B. Shumeet, "Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning," Carnegie Mellon University CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pa, USA, 1994.
- [13] O. K. Erol and I. Eksin, "A new optimization method: big bang-big crunch," *Advances in Engineering Software*, vol. 37, no. 2, pp. 106–111, 2006.
- [14] A. Kaveh and S. Talatahari, "Size optimization of space trusses using big bang-big crunch algorithm," *Computers and Structures*, vol. 87, no. 17-18, pp. 1129–1140, 2009.
- [15] A. Kaveh and S. Talatahari, "Optimal design of schwedler and ribbed domes via hybrid big bang-big crunch algorithm," *Journal of Constructional Steel Research*, vol. 66, no. 3, pp. 412–419, 2010.
- [16] A. Kaveh and S. Talatahari, "A discrete big bang-big crunch algorithm for optimal design of skeletal structures," *Asian Journal of Civil Engineering*, vol. 11, no. 1, pp. 103–122, 2010.
- [17] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.
- [18] P. Yadav, R. Kumar, S. K. Panda, and C. S. Chang, "An intelligent tuned harmony search algorithm for optimisation," *Information Sciences*, vol. 196, pp. 47–72, 2012.
- [19] S. Gholizadeh and A. Barzegar, "Shape optimization of structures for frequency constraints by sequential harmony search algorithm," *Engineering Optimization*, vol. 45, no. 6, pp. 627–646, 2013.
- [20] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," *Acta Mechanica*, vol. 213, no. 3-4, pp. 267–289, 2010.
- [21] L. Xie, J. Zeng, and R. A. Formato, "Selection strategies for gravitational constant G in artificial physics optimisation based on analysis of convergence properties," *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 380–391, 2012.
- [22] A. H. Gandomi, X.-S. Yang, A. H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing & Applications*, vol. 22, no. 6, pp. 1239–1255, 2013.
- [23] X. S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.
- [24] X. Li, J. Zhang, and M. Yin, "Animal migration optimization: an optimization algorithm inspired by animal migration behavior," *Neural Computing and Applications*, 2013.
- [25] A. H. Gandomi and A. H. Alavi, "Krill herd: a new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 4831–4845, 2012.
- [26] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "Stud krill herd algorithm," *Neurocomputing*, 2013.
- [27] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "An effective krill herd algorithm with migration operator in biogeography-based optimization," *Applied Mathematical Modelling*, 2013.
- [28] R. Storn and K. Price, "Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep. 1075-4946, International Computer Science Institute, Berkeley, Calif, USA, 1995.
- [29] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
- [30] X. Li and M. Yin, "Application of differential evolution algorithm on self-potential data," *PLoS One*, vol. 7, no. 12, Article ID e51199, 2012.
- [31] G. G. Wang, A. H. Gandomi, A. H. Alavi, and G. S. Hao, "Hybrid krill herd algorithm with differential evolution for global numerical optimization," *Neural Computing & Applications*, 2013.
- [32] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, December 1995.
- [33] R. J. Kuo, Y. J. Syu, Z.-Y. Chen, and F. C. Tien, "Integration of particle swarm optimization and genetic algorithm for dynamic clustering," *Information Sciences*, vol. 195, pp. 124–140, 2012.
- [34] S. Talatahari, M. Kheirollahi, C. Farahmandpour, and A. H. Gandomi, "A multi-stage particle swarm for optimum design of truss structures," *Neural Computing & Applications*, vol. 23, no. 5, pp. 1297–1309, 2013.
- [35] K. Y. Huang, "A hybrid particle swarm optimization approach for clustering and classification of datasets," *Knowledge-Based Systems*, vol. 24, no. 3, pp. 420–426, 2011.
- [36] W. Khatib and P. Fleming, "The stud GA: a mini revolution?" *Parallel Problem Solving from Nature*, pp. 683–691, 1998.

- [37] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, Coimbatore, India, December 2009.
- [38] A. H. Gandomi, S. Talatahari, X. S. Yang, and S. Deb, "Design optimization of truss structures using cuckoo search algorithm," *The Structural Design of Tall and Special Buildings*, vol. 22, no. 17, pp. 1330–1349, 2013.
- [39] X. Cai, S. Fan, and Y. Tan, "Light responsive curve selection for photosynthesis operator of APOA," *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 373–379, 2012.
- [40] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.
- [41] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Mixed variable structural optimization using firefly algorithm," *Computers & Structures*, vol. 89, no. 23–24, pp. 2325–2336, 2011.
- [42] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver, Frome, UK, 2008.
- [43] X. S. Yang, "Firefly algorithms for multimodal optimization," in *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications*, pp. 169–178, Springer, Sapporo, Japan, 2009.
- [44] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.
- [45] X.-S. Yang, S. S. S. Hosseini, and A. H. Gandomi, "Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect," *Applied Soft Computing Journal*, vol. 12, no. 3, pp. 1180–1186, 2012.
- [46] R. Parpinelli and H. Lopes, "New inspirations in swarm intelligence: a survey," *International Journal of Bio-Inspired Computation*, vol. 3, no. 1, pp. 1–16, 2011.
- [47] D. Zou, L. Gao, J. Wu, and S. Li, "Novel global harmony search algorithm for unconstrained problems," *Neurocomputing*, vol. 73, no. 16–18, pp. 3308–3318, 2010.
- [48] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, and J. Li, "Incorporating mutation scheme into krill herd algorithm for global numerical optimization," *Neural Computing and Applications*, 2012.
- [49] S. Z. Zhao, P. N. Suganthan, Q.-K. Pan, and M. Fatih Tasgetiren, "Dynamic multi-swarm particle swarm optimizer with harmony search," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3735–3742, 2011.
- [50] G. Wang, L. Guo, H. Duan, L. Liu, and H. Wang, "A modified firefly algorithm for UCAV path planning," *International Journal of Hybrid Information Technology*, vol. 5, no. 3, pp. 123–144, 2012.
- [51] A. H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, "Firefly algorithm with chaos," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 1, pp. 89–98, 2013.
- [52] Y. Zhang, D. Huang, M. Ji, and F. Xie, "Image segmentation using PSO and PCM with Mahalanobis distance," *Expert Systems with Applications*, vol. 38, no. 7, pp. 9036–9040, 2011.
- [53] G. G. Wang, L. Guo, A. H. Gandomi, A. H. Alavi, and H. Duan, "Simulated annealing-based krill herd algorithm for global optimization," *Abstract and Applied Analysis*, vol. 2013, Article ID 213853, 11 pages, 2013.
- [54] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

Research Article

Noise-Assisted Concurrent Multipath Traffic Distribution in Ad Hoc Networks

Narun Asvarujanon,¹ Kenji Leibnitz,² Naoki Wakamiya,^{1,2} and Masayuki Murata^{1,2}

¹ Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan

² Center for Information and Neural Networks (CiNet), National Institute of Information and Communications Technology (NICT) and Osaka University, 1-4 Yamadaoka, Suita, Osaka 565-0871, Japan

Correspondence should be addressed to Narun Asvarujanon; narun-a@ist.osaka-u.ac.jp

Received 5 August 2013; Accepted 3 September 2013

Academic Editors: Z. Cui and X. Yang

Copyright © 2013 Narun Asvarujanon et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The concept of biologically inspired networking has been introduced to tackle unpredictable and unstable situations in computer networks, especially in wireless ad hoc networks where network conditions are continuously changing, resulting in the need of robustness and adaptability of control methods. Unfortunately, existing methods often rely heavily on the detailed knowledge of each network component and the preconfigured, that is, fine-tuned, parameters. In this paper, we utilize a new concept, called attractor perturbation (AP), which enables controlling the network performance using only end-to-end information. Based on AP, we propose a concurrent multipath traffic distribution method, which aims at lowering the average end-to-end delay by only adjusting the transmission rate on each path. We demonstrate through simulations that, by utilizing the attractor perturbation relationship, the proposed method achieves a lower average end-to-end delay compared to other methods which do not take fluctuations into account.

1. Introduction

In wireless ad hoc networks, it is known that transmissions over wireless channels suffer from radio propagation loss, shadowing, fading, radio interference, and limited bandwidth. Moreover, there are also effects from traffic patterns which can degrade certain links if the network control is not traffic aware. Therefore, a lot of research attempts have been made in every layer and even across layers to improve the performance of communications in ad hoc networks. However, most improvements consider only the existing problems and lack the flexibility towards emerging problems, especially the highly focused cross-layer optimization becomes less extensible and difficult to maintain [1].

Traditional network control mechanisms often rely on a certain set of predefined rules and fine-tuned parameters for known situations. However, computer network architectures and their protocols have become increasingly sophisticated over time through addition of many features to support new applications, where different applications may require different settings of protocol parameters. Since the total

number of possible situations occurring in the real world is too numerous to be handled by preprogrammed sets of definitions, it is necessary that new networking mechanisms are designed in a flexible and adaptive manner to cater for any changes in the environment.

In an attempt to design new adaptive networking methods, concepts based on biological mechanisms have been proposed [2, 3] for self-organized control since they are able to provide greater robustness and adaptability to external influences. The core idea is to derive a protocol that is based on the model of a natural phenomenon. For example, swarm intelligence is a concept where individual agents mimic the behavior of foraging ants or bees in insect swarms and it has been successfully applied to routing problems in the past [4]. Firefly groups perform a distributed synchronization of their flashing behavior and this is applied to synchronization in sensor networks [5]. Reaction-diffusion describes the chemical dynamics of morphogens in the development of stripes or spots on animal furs. Based on the reaction-diffusion dynamics, the coding rate for camera sensor networks can be controlled [6].

Since biological systems are often described as dynamic systems, they rely on a mathematical formulation given as differential equations. In dynamic systems, attractors describe the states to which the system evolves over time. In the past, we studied the concept of attractor selection, which is based on the dynamics found in gene expression [7] and has been previously also applied to tackle problems in communication networks [8, 9]. In this paper, we apply a similar biological mechanism called attractor perturbation (AP), which is derived from the fluctuation-response relationship observed in an experiment on the evolution of functional proteins in a cell [10]. A previous application of AP to computer networks can be found in [11, 12].

In this paper, we focus on bandwidth improvement and end-to-end delay minimization in ad hoc networks. In terms of bandwidth improvement, one of the most common approaches is to use multiple paths in the same or across different media (multihoming). To enable the ability to utilize multiple paths concurrently, there is some existing work in both wired, for example, opportunistic multipath scheduling (OMS) [13], and wireless networks, for example, concurrent multipath transfer (CMT) [14, 15] and adaptive load balancing algorithm (ALBAM) [16]. However, most existing control methods require a full knowledge of the current network status, for example, queue length on each node, which is difficult to obtain or requires frequent probing causing bandwidth degradation. Therefore, we apply AP to concurrent multipath traffic distribution to improve the available bandwidth while utilizing the AP relationship to predict the outcome of the traffic adjustment and also minimize the end-to-end delay at the same time.

The contributions of this paper are as follows. First is the end-to-end characteristics of the AP-based proposal, which allows easy deployment in existing networks without the need of modifying all intermediate nodes. Second is the usage of statistical information, which consumes less bandwidth to obtain than using probing results. Third is the ability to provide a simplified view of the network as a black box with only the end-to-end observed variables while maintaining the ability to influence the network performance. Last is the expectable adaptability of the proposal since it does not rely only on the predefined parameters but also takes into account the fluctuation as a source of robustness and adaptability in a similar manner to other biological mechanisms.

The rest of this paper is organized as follows. We first explain the biologically inspired attractor perturbation model which is the basis of this study in Section 2. Next, we describe our proposal, that is, a concurrent multipath traffic distribution method, in Section 3. Then, the evaluation results from simulation are presented and discussed in Section 4. Finally, we conclude this paper and describe future work.

2. Attractor Perturbation

The attractor perturbation model is derived from observations of fluctuation and response in biological systems, in particular, an experiment on the evolution of functional proteins in clone bacteria cells.

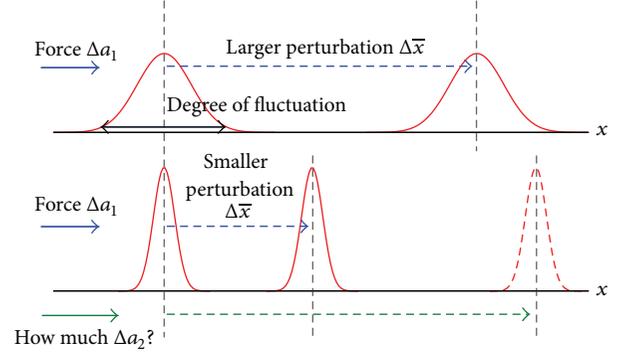


FIGURE 1: Dynamics of attractor perturbation.

2.1. Mathematical Model. In [10], it was found that the fluctuation, which is expressed by the variance of the fluorescence of a bacterial protein, and its response, which is the average change in this fluorescence against the applied force, have a linear relationship modeled as follows:

$$\bar{x}_{a+\Delta a} - \bar{x}_a = b\Delta a\sigma_a^2, \quad (1)$$

where b is a scalar constant, x is a time dependent measurable variable in the system with mean \bar{x} and variance σ_a^2 , and a is the amount of force applied to the system. The attractor perturbation model is very similar to the fluctuation-dissipation theorem in physics.

There are two major assumptions underlying the model formulation of AP. First, the variable x must have a Gaussian-like distribution which is often observed in biology. Second, the variable x and the parameter a are closely associated. In other words, a change in the parameter a would strongly affect the distribution of the variable x .

2.2. Applications of AP Model. Equation (1) reveals that the difference in the average of the variable x before and after applying a change to the parameter a is linearly proportional to the amount of change in a , which is the force Δa , and the variance of the variable x prior to the change. Therefore, one can predict the response to the applied force from the fluctuation of the targeted system. Since the amount of change in a can be seen as controllable, it is possible to adjust the difference in average of x , called *perturbation*, by taking the current variance of x into consideration. Obviously, using the same amount of force Δa to perturb the average of x when the variance σ_a^2 is large will also lead to a larger perturbation, as shown in Figure 1. Based on this relationship, we use this model to estimate the amount of force required to achieve the desired amount of perturbation.

To confirm the applicability of the AP model in our proposal, we first observed the delay distribution in ad hoc networks and discovered that it resembles a Gaussian distribution. Second, a related study has already shown the applicability of AP to traffic rate control for achieving a target delay in wired networks [12]. Since AP allows simplifying the underlying system as a black box by observing only the end-to-end variables, it is a promising way to reuse the concept of

rate control in ad hoc networks. Hence, we decided to use AP for concurrent multipath traffic distribution by performing traffic rate control on each path to achieve overall higher bandwidth and lower end-to-end delay as explained in the next section.

3. Concurrent Multipath Traffic Distribution

The advantage of using multiple paths is that if one path breaks due to failures at intermediate links or nodes, at least one other path can still be maintained. Furthermore, using multiple paths permits a better load balancing by distributing traffic more evenly in the network. Particularly, if nodes in an ad hoc scenario are operated by batteries, this may lead to reduced energy consumption of intermediate nodes. Finally, using multiple paths concurrently can improve the total available bandwidth in the network.

In today's wireless networks, it becomes common that participating devices can connect to more than one radio access technologies (RATs) and even within the same RAT, there are possibly multiple separated channels to use. Therefore, the concept of multipath can now be extended to multichannel and multihoming in heterogeneous wireless networks. Even though our current work focuses on ad hoc networks, our concept of path is still applicable to traffic allocation over multichannel and multihoming scenarios. The allocation granularity, which describes the unit of information allocated to each path, is also of great importance [17]. Coarse granularities, such as per connection or per flow, tend to reduce the management overhead but are not as flexible as small granularities, for example, per packet, since these permit a better distribution of traffic. However, per-packet granularity may require reordering at the destination, if the latencies differ too much among paths.

3.1. System Model. In this study, we consider a situation where a source node is connected to the destination node via multiple paths and each path i does not cause interference with another as illustrated in Figure 2. This network model covers both ad hoc (or mesh) networks with multiple radio channels and also multihoming systems. For the sake of simplicity, we consider only $n = 2$ in this paper, but the proposed method can be easily extended to $n > 2$ cases as shown in the appendix.

The notations of variables on each path i are as follows:

- (i) observed end-to-end delay (measurable variable): x_i ,
- (ii) current traffic rate (controllable variable or force): a_i ,
- (iii) amount of traffic rate adjustment: Δa_i ,
- (iv) average end-to-end delay prior to applying Δa_i : \bar{x}_i ,
- (v) average end-to-end delay after applying Δa_i : \bar{x}'_i ,
- (vi) delivered packet count: n_i .

3.2. Problem Definition. Our proposal aims at minimizing the average end-to-end delay of all packets. Using AP, we attempt to minimize the *total delay sum*, which directly corresponds to the average delay of all packets on both paths.

The delay sum can be estimated through the product of the expected delay and the adjusted traffic rate on each path.

According to the AP concept, in case of two paths, we have the expected average delay \bar{x}'_i :

$$\begin{aligned}\bar{x}'_1 &= \bar{x}_1 + b_1 \Delta a_1 \sigma_1^2, \\ \bar{x}'_2 &= \bar{x}_2 + b_2 \Delta a_2 \sigma_2^2.\end{aligned}\tag{2}$$

Therefore, we can define a function $f(\Delta a_1, \Delta a_2)$ as an estimation of the average delay after applying traffic rate adjustment Δa_i as follows:

$$\begin{aligned}f(\Delta a_1, \Delta a_2) &= (a_1 + \Delta a_1) \bar{x}'_1 + (a_2 + \Delta a_2) \bar{x}'_2 \\ &= (a_1 \bar{x}_1 + a_2 \bar{x}_2) + (\bar{x}_1 + a_1 b_1 \sigma_1^2) \Delta a_1 \\ &\quad + (\bar{x}_2 + a_2 b_2 \sigma_2^2) \Delta a_2 + b_1 \sigma_1^2 \Delta a_1^2 + b_2 \sigma_2^2 \Delta a_2^2.\end{aligned}\tag{3}$$

Given that $c' = (a_1 \bar{x}_1 + a_2 \bar{x}_2)$, $c_1 = (\bar{x}_1 + a_1 b_1 \sigma_1^2)$, $c_2 = (\bar{x}_2 + a_2 b_2 \sigma_2^2)$, $k_1 = b_1 \sigma_1^2$, and $k_2 = b_2 \sigma_2^2$, (3) can be formulated as a constrained optimization (minimization) problem as follows:

$$\begin{aligned}\text{Minimize } f(\Delta a_1, \Delta a_2) &= c' + c_1 \Delta a_1 + c_2 \Delta a_2 \\ &\quad + k_1 \Delta a_1^2 + k_2 \Delta a_2^2 \\ \text{subject to } \Delta a_1 + \Delta a_2 &= 0.\end{aligned}\tag{4}$$

The solution of the minimization problem in (4) is the amount of the adjustment in traffic rate to be applied to each path in order to achieve minimal average end-to-end delay of all packets. The *subject to* condition is required since the total amount of traffic prior to and after the adjustment has to remain the same.

3.3. Lagrangian Optimization. The minimization problem which has the form as in (4) can be solved using *Lagrangian Optimization*.

The Lagrangian has the general form of

$$L(x^*, \lambda^*) = f(x) - \sum_i [\lambda_i (g_i(x) - b_i)],\tag{5}$$

where x^* is the optimal solution of x and λ^* is the penalizing Lagrangian multiplier.

The associated Lagrangian of (4) is:

$$\begin{aligned}L(\Delta a_1^*, \Delta a_2^*, \lambda^*) &= c' + c_1 \Delta a_1^* + c_2 \Delta a_2^* + k_1 \Delta a_1^{*2} \\ &\quad + k_2 \Delta a_2^{*2} - \lambda^* (\Delta a_1^* + \Delta a_2^*),\end{aligned}\tag{6}$$

$$\frac{\partial L}{\partial \Delta a_1^*} = c_1 + 2k_1 \Delta a_1^* - \lambda = 0,\tag{7}$$

$$\frac{\partial L}{\partial \Delta a_2^*} = c_2 + 2k_2 \Delta a_2^* - \lambda = 0,\tag{8}$$

$$\frac{\partial L}{\partial \lambda^*} = -(\Delta a_1^* + \Delta a_2^*) = 0.\tag{9}$$

In the three equations (6)–(8), there are three unknown variables Δa_1^* , Δa_2^* , and λ^* . Therefore, this optimization

```

(1) procedure AdjTRAFFIC ( $\bar{x}_1, \sigma_1^2, a_1, n_1, \bar{x}_2, \sigma_2^2, a_2, n_2$ )
(2)                                      $\triangleright$  Only AP+Com uses  $n_1, n_2$ 
(3)   for all  $i$  do
(4)      $\bar{x}_i \leftarrow (\rho(a_i - n_i) + \bar{x}_i n_i) / \rho a_i$ 
(5)                                      $\triangleright$  Delay compensation (AP+Com only)
(6)   end for
(7)    $(\Delta a_1^*, \Delta a_2^*) \leftarrow \text{SolveMinimization}(\bar{x}_1, \sigma_1^2, a_1, \bar{x}_2, \sigma_2^2, a_2)$ 
(8)   if  $|\Delta a_1^*| > \alpha_{\max} \times (a_1 + a_2)$  then
(9)      $\Delta a_1^* \leftarrow \alpha_{\max} \times (a_1 + a_2) \times \Delta a_1^* / |\Delta a_1^*|$ 
(10)     $\Delta a_2^* \leftarrow -\Delta a_1^* \triangleright$  Maximum rate adjustment ratio  $\alpha_{\max}$ 
(11)   end if
(12)    $a_1 \leftarrow a_1 + \Delta a_1^*$ 
(13)    $a_2 \leftarrow a_2 + \Delta a_2^*$ 
(14) end procedure

```

ALGORITHM 1: AP-based traffic distribution.

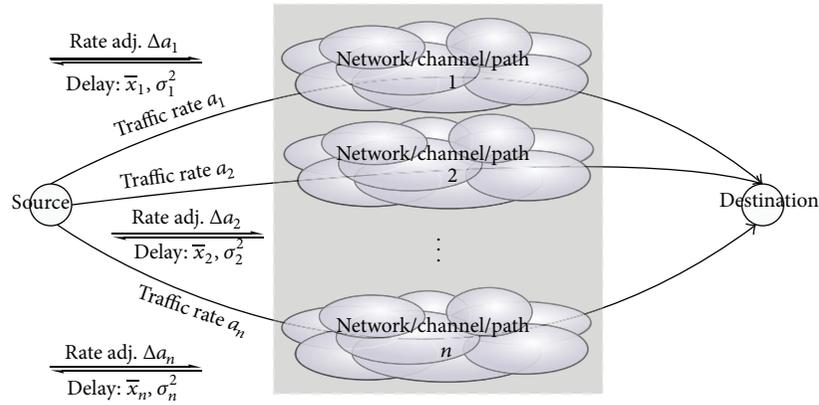


FIGURE 2: Overall system model.

problem can be solved and we obtain the optimal amount of traffic rate adjustment Δa_i for each path i to minimize the sum of average delays.

According to the steps taken previously, the optimal solution in case of two paths is as follows:

$$\begin{aligned}
 \Delta a_1^* &= \frac{c_2 - c_1}{2(k_1 + k_2)} \\
 &= \frac{(\bar{x}_2 + a_2 b_2 \sigma_2^2) - (\bar{x}_1 + a_1 b_1 \sigma_1^2)}{2(b_1 \sigma_1^2 + b_2 \sigma_2^2)}, \quad (10) \\
 \Delta a_2^* &= -\Delta a_1^*.
 \end{aligned}$$

3.4. Traffic Distributing Steps. The optimal solution Δa_i^* from (10) is used in Algorithm 1, executed at the source every interval of duration ρ ($=5$ s in our simulation experiments).

In every iteration of the algorithm, our main AP-based traffic control method (AP-Com) uses the measured average \bar{x}_i , the variance σ_i^2 , and the current traffic rate a_i to solve the minimization problem. The optimal solution is applied to the current traffic rate gradually, which is limited by the maximum rate adjustment ratio α_{\max} .

In addition to our main proposal AP-Com, a variation of AP-based method with delay compensation (AP+Com) is also proposed here. The delay compensation process serves to maintain throughput in our mechanism at the cost of using more information of the delivered packet count n_i on each path i for calculating the number of lost packets. Without delay compensation, AP-Com behaves in favor of lower delay regardless of the delivery performance on each path. In AP+Com, we compensate for packet loss assuming that each lost packet has the end-to-end delay equal to ρ . We will show results of both AP-based methods with and without delay compensation in the evaluation section.

The coefficient b_i is required to solve the minimization problem. This value is crucial for estimating the average delay after applying the traffic rate adjustment and is determined in every iteration based on the current iteration's average delay, the previous iteration's average delay and variance, and the amount of rate adjustment applied in the previous iteration using (1). In case that there is no traffic rate adjustment in the previous iteration, the default b given in the configuration is used in that iteration and the accurate b_i can be calculated on the subsequent iteration. Note that another study on AP in wired networks [12] has found that the AP-based control does not require a fine tuning of coefficient b . We will show

later in Section 4.4 that the same concept also holds in our wireless case.

4. Evaluation

To demonstrate the validity of the AP-based traffic distribution, we performed simulations using the QualNet network simulator. We divided the evaluation into two parts, throughput in a static scenario and end-to-end delay improvement in mobile scenarios.

4.1. Comparison Target. In the static scenario evaluation, we compare the performance of the AP-based proposal to two existing multipath transport layer control protocols: concurrent multipath transfer (CMT) [14] and multipath real-time transport protocol (MPRTP) [18]. CMT utilizes the SCTP [19] protocol to send data concurrently to the destination, which has multiple networking interfaces, while MPRTP is an extended version of the RTP protocol to allow scheduling of RTP traffic over multiple paths concurrently.

Our proposal is similar to the recently proposed MPRTP protocol because both are implemented over UDP. The scheduling algorithm of MPRTP uses loss rate, packet sizes, bytes sent, and interval information from RTP's receiver reports (RRs) to estimate the bandwidth of the current path. Using packet loss information, paths are categorized as congested, mildly congested, and non-congested conditions. The scheduler then continuously assigns a portion of traffic to each path, more if it is noncongested and less if it is congested, while keeping the same total rate. We have implemented Algorithm 1 of MPRTP from [18] in QualNet, assuming a perfect knowledge of end-to-end information instead of using real RR packets, and compare its performance with our proposal.

As already mentioned previously, CMT is implemented over SCTP, which is supported by the IETF alongside with TCP and UDP as a general purpose reliable transport protocol with connection-oriented, reliable data transfer, window-based, congestion control, and flow control features, similar to TCP. One important feature of SCTP is its built-in multihoming capability where a connection can be established between a set of IP addresses. However, standard SCTP uses only a pair of primary IP addresses at a time which does not allow concurrent transmissions. CMT is a modified version of SCTP that allows concurrent transmissions and includes few improvements on fast retransmission, congestion window update, and delayed acknowledgment algorithms. It was found in [15] that the receiving buffer, referred to as *rBuf* in the original paper, can be a performance bottleneck of CMT. Therefore, we only compare the best CMT results without such constraint, called *CMT Unlimited*, as a reference in this section.

We did not implement our proposal over TCP or SCTP because in the TCP scheme, due to various control mechanisms, for example, rate control and congestion avoidance control, end-to-end delays do not generally follow a Gaussian distribution. There are a few special cases when TCP traffic does follow a Gaussian distribution [20, 21]; however, we leave the investigation of those cases as a future work.

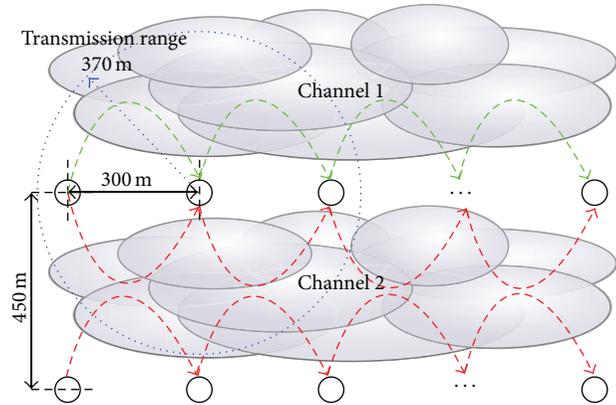


FIGURE 3: Simulation scenario from [15].

Currently, to study the pure behavior of our proposal, we assume that the end-to-end information is known to the source node without an actual measurement. However, a feedback mechanism can be easily implemented to deliver this information to the source node. Since the statistical information is needed only once every execution interval, the overhead can be considered negligible and the actual results should be similar to the simulation results shown in this paper. Moreover, to have a fair comparison, our comparison targeting MPRTP also uses the same assumptions.

4.2. Static Scenario. We set up the simulation scenario exactly the same as described in [15]; see Figure 3. There are two chains of nodes where the distance between nodes on the same chain is 300 m and the distance between chains is 450 m. The transmission range of each node is approximately 370 m where the carrier sensing range and the interference range span farther under the two-ray path loss model without fading. The default transmission range in QualNet 5.2 is only 300 m and we matched the transmission range to [15] by slightly increasing the TX power.

In this scenario, one chain serves as the main concurrent multipath sessions for bandwidth evaluation and the other chain serves as interfering background traffic. On the main chain, each node is equipped with two IEEE 802.11b interfaces connected to two noninterfering channels. On the background traffic chain, each node is equipped with only one interface connected to the second channel which is used in the main chain. The data rate for IEEE 802.11b is 2 Mbps and the RTS/CTS mechanism is enabled. Static routes are used in this simulation to eliminate complications due to the effect of the routing protocol.

The number of nodes varies from 10 to 34 (4, 8, and 16 hops on each chain). The traffic used in this evaluation is CBR with 1000 bytes per packet. We performed the simulations using several different traffic rates on the main chain and have selected the one with the highest obtained throughput shown in Figure 4. The main total traffic rates for the 4, 8, and 16 hop cases are 65.1, 48.8, and 48.8 KBps, respectively, which are decided based on the number of hops to the destination and the ratio of capacity explained in [22]. The main traffic is sent

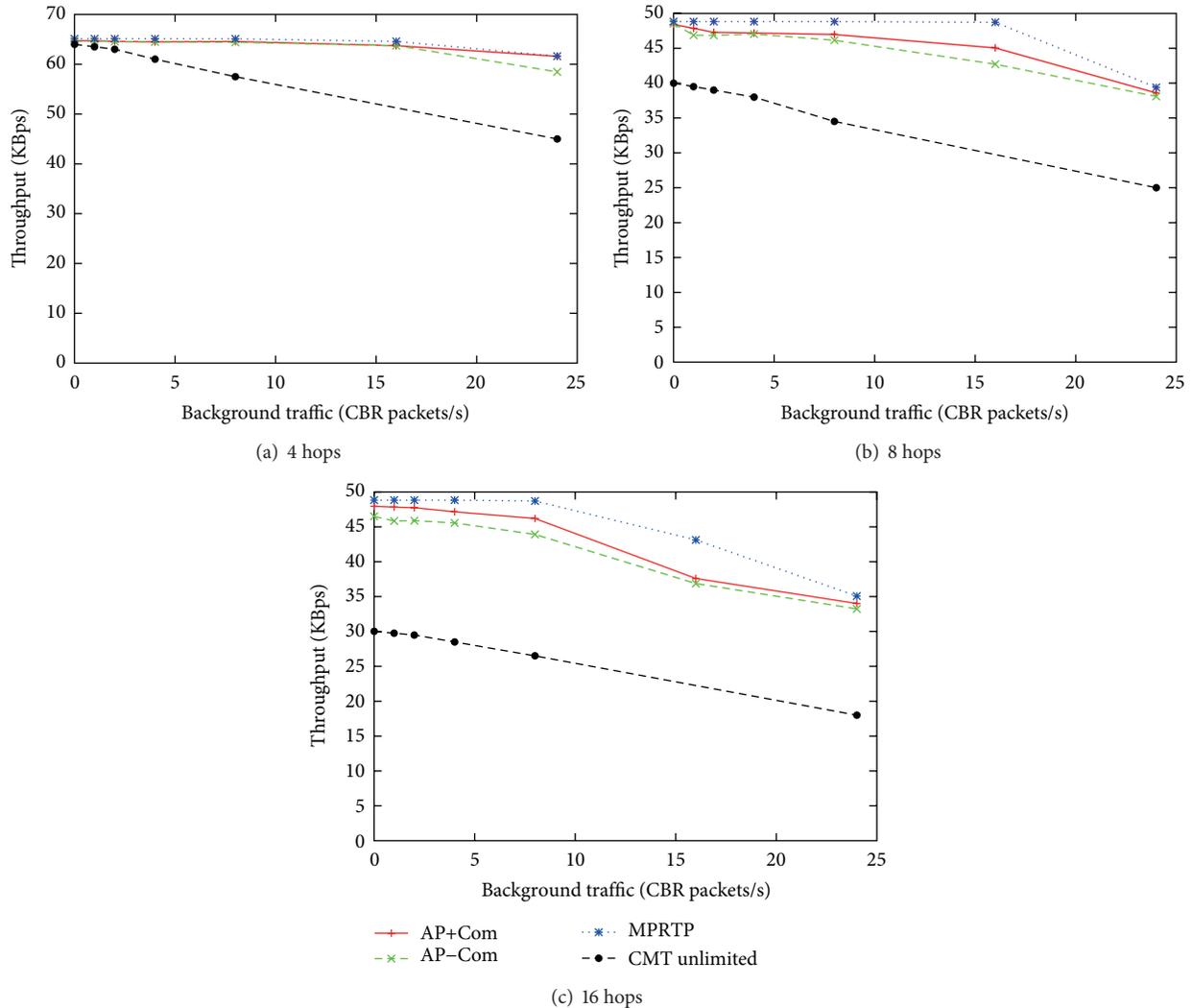


FIGURE 4: Throughput comparison between AP-based proposals, MPRTT, and CMT unlimited $rBuf$ from [15].

from the source during 60–360 seconds in a 420-second long simulation. The amount of background traffic varies from 0 to 24 packets per second. The results of our protocol shown in Figure 4 are the average of 30 runs, which is the same number of runs performed in [15].

From Figure 4, it can be clearly seen that in comparison to CMT, MPRTT, and AP-based methods (with $b = 0.1$ and $\alpha_{\max} = 0.1$) can achieve much higher throughput and are less susceptible to the interference from background traffic. Even though now the implementations of both AP and MPRTT methods do not fully use feedback packets to gather the statistical information, a single feedback packet per decision interval ρ ($=5$ s in this study) can hardly affect the higher bandwidth shown here. Therefore, we can claim here that the AP-based method and MPRTT are viable alternatives to CMT, which can provide better bandwidth improvement when an application can tolerate or handle packet loss.

Among UDP-based proposals, MPRTT could achieve higher bandwidth due to its accurate rule-based bandwidth prediction in cases of low interference and background traffic

load. However, when congestion occurs and more packet loss is observed, the bandwidth difference becomes smaller. Since MPRTT relies heavily on the information accuracy, the smaller difference is most likely due to the lower accuracy of rule-based bandwidth prediction of MPRTT.

A similar behavior can be observed between AP+Com, which estimates delay compensation using packet loss, and AP-Com, which does not use delay compensation. With the delay compensation process added in AP+Com, the performance of the AP-based method is slightly better than in AP-Com because the compensated delay reflects the actual network conditions better and enhances the accuracy of AP in estimating delay after adjusting the traffic rate. However, the performance difference becomes smaller in the same manner to MPRTT when the load is high.

It is important to emphasize that while using much less information, that is, only delay information without delivered packet count nor lost packet count in comparison to AP+Com and MPRTT, AP-Com can achieve comparable throughput to other protocols. This is a piece of evidence of

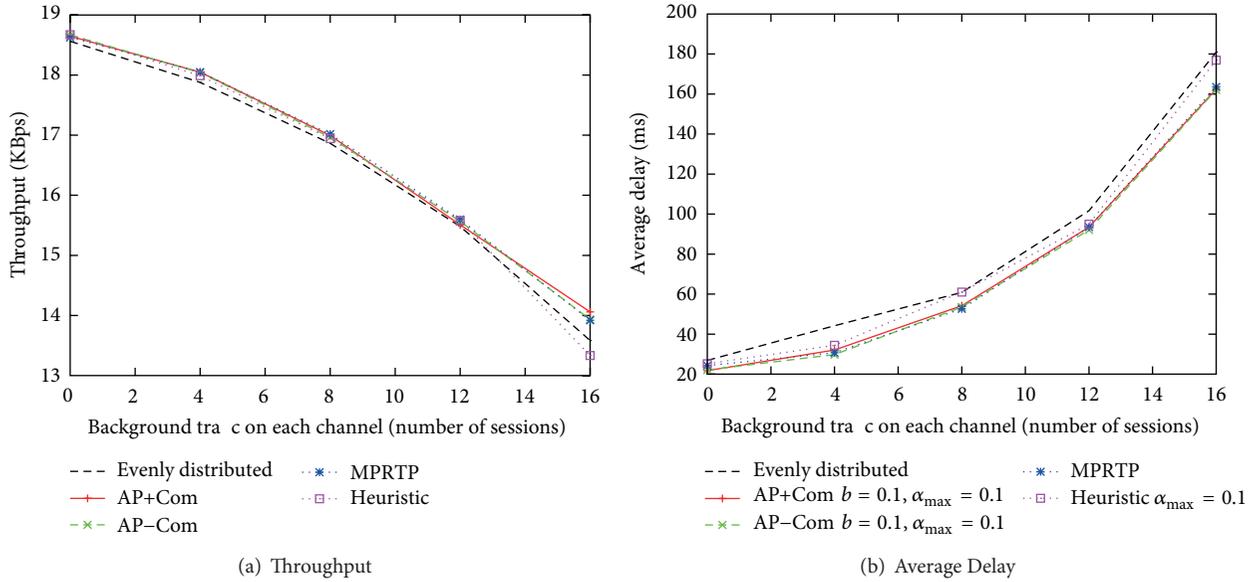


FIGURE 5: Performance comparison under mobility scenario.

the adaptability of the AP-based methods, which uses delay fluctuations, and further supportive results will be shown in the next subsection.

4.3. Mobile Scenario. In this section, we evaluate the average delay of the AP-based proposals with $b = 0.1$ and $\alpha_{max} = 0.1$ in mobile scenarios. In such scenarios, an adaptive traffic distribution method is required since a traffic pattern on a certain path is affected by changes in other paths due to rerouting, topology changes, and so forth. Most concurrent multipath traffic distribution methods do not support/consider mobile scenarios. Therefore, in addition to the baseline strategy where the traffic is split evenly on both paths (*evenly distributed*) and the MPRTP approach, we developed another comparison method, called *heuristic* method, which operates based on the end-to-end average delay in a similar manner to our AP-based method. The main differences are that the heuristic method

- (i) adjusts the traffic with the fixed ratio of the total traffic rate $\alpha_{max} = 0.1$ (the AP-based method calculates the optimal solution in the range of $[-\alpha_{max}, \alpha_{max}]$),
- (ii) cannot estimate the delay after applying the traffic rate adjustment, and
- (iii) makes the decision to transfer the traffic purely from the path with higher average delay or the path with higher loss rate (in case of no delivered packet) to the path with lower one.

We expect that the evaluation against the *heuristic* method will reveal the importance of taking the fluctuation into account when performing traffic distribution.

The scenario settings are as follows. 100 mobile nodes are distributed randomly in a $1500 \times 1500 \text{ m}^2$ area. The random waypoint model is used with a minimum speed of 2 m/s,

a maximum speed of 10 m/s, and a pause time of 30 s. Each node is equipped with two 802.11b interfaces with the data rate of 2 Mbps, connected to two noninterfering radio channels. There is one main multipath traffic session with total traffic rate of 20 packets/s and the packet size of 1000 bytes, which is the same as the previous scenario. The number of background CBR traffic sessions varies from 0, 4, 8, and 12 to 16 sessions per channel. Every background traffic session has the traffic rate of 1 packet/s. We chose a relatively low bit rate of background traffic to only increase interference, while ensuring sufficient bandwidth for the main session to avoid overloading conditions, in which we cannot evaluate the performance of traffic distribution methods.

The average results from 100 runs are shown in Figure 5. Figure 5 shows the throughput and average delay against the amount of background traffic. Since the differences between each curve in Figure 5(b) cannot be clearly seen, more details of average delay on each run is shown in Figure 6 using box-and-whisker diagram where the box reflects the lower quartile (Q1), median (Q2), and upper quartile (Q3). The bars show the range of ± 1.5 IQR and the dots show the data that are outside the range.

It can be observed from Figure 5 that the throughput of each approach is quite similar. However, there is a difference in average delay as shown in Figure 5(b). It is out of question that the baseline approach without traffic redistribution (*evenly distributed*) has the worst average delay. Our AP proposals can achieve the same level of average delay as MPRTP by using only end-to-end delay statistics. The newly proposed comparison method (*heuristic*), which uses only average end-to-end delay, performs much worse than the AP proposals because using only the average delay cannot provide a good estimate of the path quality, that is, congestion level.

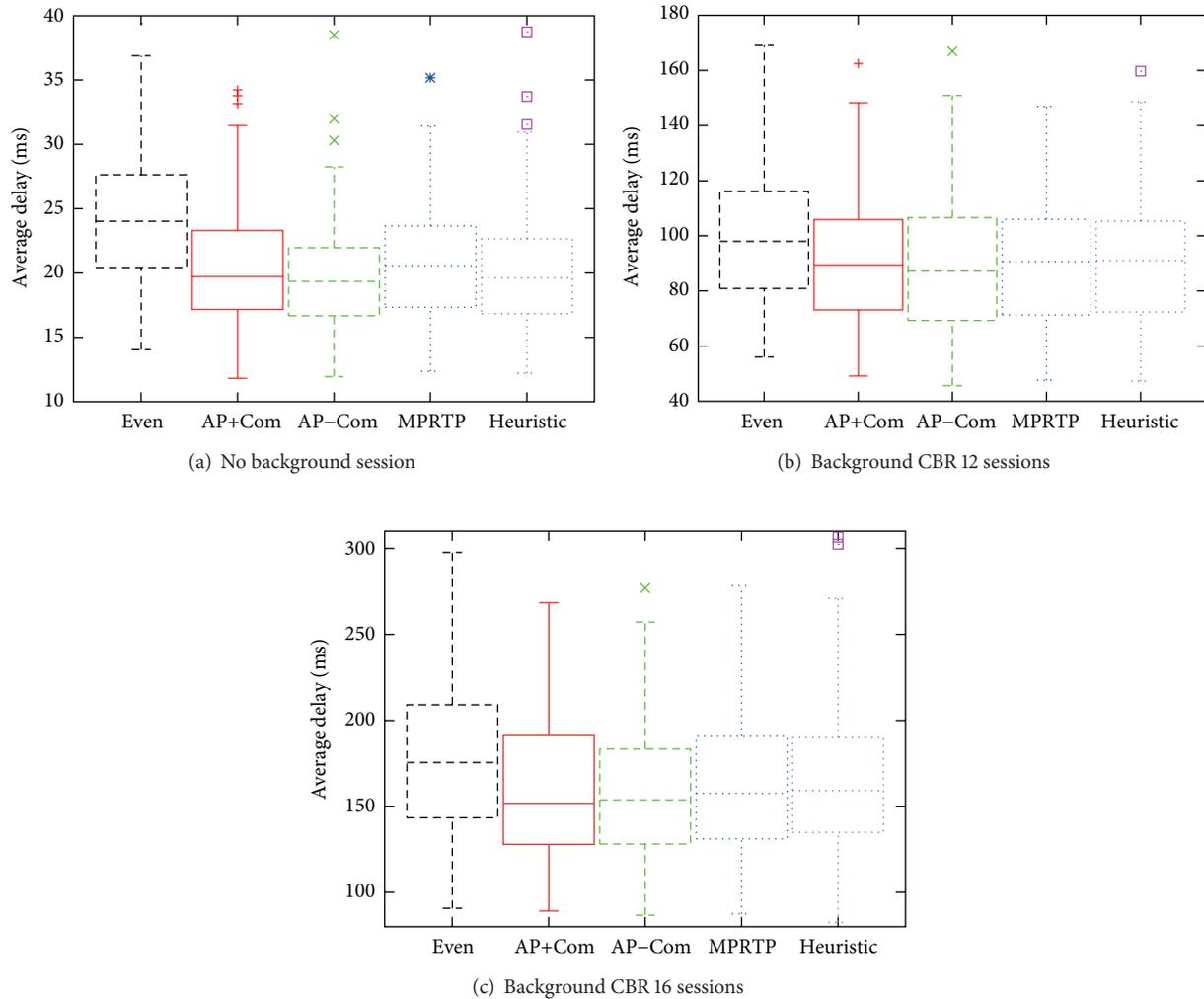


FIGURE 6: Average delay comparison under mobility scenario (y-axis is capped for visibility).

Moreover, Figure 6 indicates that the median of all methods generally follow the same tendency of the average, except the heuristic one. This is an effect from cases where the average delay is very high (capped and cannot be seen in the figure). Those cases are caused by the inappropriate traffic distribution that induced high congestion, which consequently causes failure in routing, hence, a much higher end-to-end delay.

According to these results, it can be understood that AP-based methods, which use both average and variance, can perform better than methods using only the average, like heuristic. Therefore, it is safe to claim that considering not only the average delay in the current interval, but also the fluctuation is important for improving the performance of the traffic distribution method.

Additionally, by using only the statistical information on delay, AP-Com can achieve comparable throughput and end-to-end delay to MPRTP, which requires more information of delivered bytes and loss rate. Hence, it is confirmed that the AP-based method does not need the details of the system

under its control, which is preferable from an implementation viewpoint because a high processing overhead, energy consumption, and errors from actual measurements can be avoided.

4.4. Discussion on Bio-Inspired Adaptability. From Figures 5 and 6, it can be seen that AP-Com is the best among all approaches. Even though the throughput results of AP-Com in the static ad hoc network scenario were slightly lower than the other approaches, it can adapt well to scenarios with higher dynamics. This result conforms with our previous assumption regarding the rule-based bandwidth prediction of MPRTP and the delay compensation of AP+Com and shows that a bio-inspired method indeed reveals better adaptability to different scenarios without the need of fine-tuning parameters.

To further support this claim, we also added the results from bandwidth improvement scenario with different coefficients b in Figure 7. It can also be seen that even with

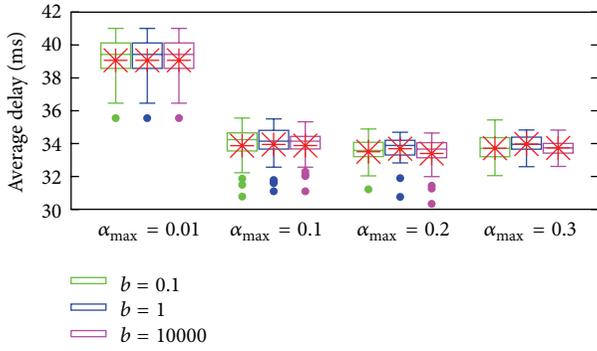


FIGURE 7: Results of AP+Com with different values of b .

inaccurate b for a specific scenario, the AP-based method can adapt to that situation and perform considerably well, due to its core bio-inspired model.

5. Conclusion

We presented a novel biologically inspired concurrent multipath traffic distribution method based on attractor perturbation. In AP, the whole underlying system is regarded as a black box and its control is based on the observed average and variance of the time series of the considered performance metric. Therefore, our proposal requires only end-to-end statistical information to perform traffic distribution. From simulation results, we have shown that our main proposal (AP-Com) can achieve lower average end-to-end delay without sacrificing throughput when compared to the heuristic method and evenly distributed traffic on all paths. Moreover, it can even achieve similar average end-to-end delay as MP RTP, which uses delivered bytes and loss rate in addition to delay information. It is natural that a mechanism using more information achieves better performance, but it suffers from inaccuracy of obtained information and also requires parameters fine tuning. An evaluation of cases with information errors remains future work.

In addition to the performance aspect, our proposal does not require any careful parameter fine tuning due to its bio-inspired nature. The usage of fluctuation, or noise, within the core AP model gives it a flexibility to handle frequent changes in the network. It is also expected that with this adaptability, our proposal should be able to handle emerging problems better than traditional methods.

Appendix

Minimization Problem: n -Path Case

According to the AP concept, in case of n paths, we have

$$\begin{aligned} \bar{x}'_1 &= \bar{x}_1 + b_1 \Delta a_1 \sigma_1^2 \\ &\vdots \\ \bar{x}'_n &= \bar{x}_n + b_n \Delta a_n \sigma_n^2. \end{aligned} \tag{A.1}$$

Total delay sum of n -path case can be calculated as follows:

$$\begin{aligned} f(\Delta a_1, \Delta a_2, \dots, \Delta a_n) &= (a_1 + \Delta a_1) \bar{x}'_1 + \dots + (a_n + \Delta a_n) \bar{x}'_n \\ &= (a_1 + \Delta a_1) (\bar{x}_1 + b_1 \Delta a_1 \sigma_1^2) + \dots \\ &\quad + (a_n + \Delta a_n) (\bar{x}_n + b_n \Delta a_n \sigma_n^2) \\ &= (a_1 \bar{x}_1 + \dots + a_n \bar{x}_n) + (\bar{x}_1 + a_1 b_1 \sigma_1^2) \Delta a_1 + \dots \\ &\quad + (\bar{x}_n + a_n b_n \sigma_n^2) \Delta a_n + b_1 \sigma_1^2 \Delta a_1^2 + \dots + b_n \sigma_n^2 \Delta a_n^2 \\ &= \sum_i^n (a_i \bar{x}_i + (\bar{x}_i + a_i b_i \sigma_i^2) \Delta a_i + (b_i \sigma_i^2) \Delta a_i^2). \end{aligned} \tag{A.2}$$

The minimization problem can be formulated similarly to the 2-path case

$$\begin{aligned} &\text{Minimize } f(\Delta a_1, \Delta a_2, \dots, \Delta a_n) \\ &\text{subject to } \sum_i^n \Delta a_i = 0. \end{aligned} \tag{A.3}$$

The associated Lagrangian of (A.3) is

$$\begin{aligned} L(\Delta a_1^*, \dots, \Delta a_n^*, \lambda^*) &= \sum_i^n (a_i \bar{x}_i + (\bar{x}_i + a_i b_i \sigma_i^2) \Delta a_i^* \\ &\quad + (b_i \sigma_i^2) \Delta a_i^{*2}) \\ &\quad - \lambda^* \sum_i^n \Delta a_i^*, \\ \frac{\partial L}{\partial \Delta a_1^*} &= a_1 b_1 \sigma_1^2 + 2b_1 \sigma_1^2 \Delta a_1^* - \lambda = 0 \\ &\vdots \\ \frac{\partial L}{\partial \Delta a_n^*} &= a_n b_n \sigma_n^2 + 2b_n \sigma_n^2 \Delta a_n^* - \lambda = 0, \\ \frac{\partial L}{\partial \lambda^*} &= -\sum_i^n \Delta a_i^* = 0. \end{aligned} \tag{A.4}$$

From (A.4), we can form an augmented matrix as follows:

$$\left[\begin{array}{cccccc|c} 2b_1 \sigma_1^2 & 0 & 0 & \dots & 0 & -1 & -2b_1 \sigma_1^2 \\ 0 & 2b_2 \sigma_2^2 & 0 & \dots & 0 & -1 & -2b_2 \sigma_2^2 \\ \vdots & \vdots & \ddots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 2b_{n-1} \sigma_{n-1}^2 & 0 & -1 & -2b_{n-1} \sigma_{n-1}^2 \\ 0 & 0 & \dots & 0 & 2b_n \sigma_n^2 & -1 & -2b_n \sigma_n^2 \\ 1 & 1 & \dots & 1 & 1 & 0 & 0 \end{array} \right]. \tag{A.5}$$

This augmented matrix can be solved using row elimination.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross-layer design," *IEEE Wireless Communications*, vol. 12, no. 1, pp. 3–11, 2005.
- [2] M. Meisel, V. Pappas, and L. Zhang, "A taxonomy of biologically inspired research in computer networking," *Computer Networks*, vol. 54, no. 6, pp. 901–916, 2010.
- [3] F. Dressler and O. B. Akan, "Bio-inspired networking: from theory to practice," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 176–183, 2010.
- [4] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, Oxford, UK, 1999.
- [5] A. Tyrrell and G. Auer, "Imposing a reference timing onto firefly synchronization in wireless networks," in *Proceedings of the IEEE 65th Vehicular Technology Conference (VTC '07)-Spring*, pp. 222–226, Dublin, Ireland, April 2007.
- [6] H. Yamamoto, K. Hyodo, N. Wakamiya, and M. Murata, "A reaction-diffusion-based coding rate control mechanism for camera sensor networks," *Sensors*, vol. 10, no. 8, pp. 7651–7673, 2010.
- [7] A. Kashiwagi, I. Urabe, K. Kaneko, and T. Yomo, "Adaptive response of a gene network to environmental changes by fitness-induced attractor selection," *PLoS ONE*, vol. 1, no. 1, article e49, 2006.
- [8] K. Leibnitz, N. Wakamiya, and M. Murata, "Biologically inspired self-adaptive multi-path routing in overlay networks," *Communications of the ACM*, vol. 49, no. 3, pp. 63–67, 2006.
- [9] N. Asvarujanon, K. Leibnitz, N. Wakamiya, and M. Murata, "Robust and adaptive mobile ad hoc routing with attractor selection," in *Proceedings of the 4th International Workshop on Adaptive and Dependable Mobile Ubiquitous Systems (ADAMUS '10)*, July 2010.
- [10] K. Sato, Y. Ito, T. Yomo, and K. Kaneko, "On the relation between fluctuation and response in biological systems," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 24, pp. 14086–14090, 2003.
- [11] K. Leibnitz and M. Murata, "Attractor selection and perturbation for robust networks in fluctuating environments," *IEEE Network*, vol. 24, no. 3, pp. 14–18, 2010.
- [12] M. Waki, N. Wakamiya, and M. Murata, "Proposal and evaluation of attractor perturbation-based rate control for stable end-to-end delay," in *Proceedings of the 7th International Conference on Bio-Inspired Models of Network, Information, and Computing Systems (BIONETICS '12)*, pp. 1–15, December 2012.
- [13] C. Cetinkaya, "Improving the efficiency of multipath traffic via opportunistic traffic scheduling," *Computer Networks*, vol. 51, no. 8, pp. 2181–2197, 2007.
- [14] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, 2006.
- [15] I. Aydin and C. C. Shen, "Performance evaluation of concurrent multipath transfer using SCTP multihoming in multihop wireless networks," in *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications (NCA '09)*, pp. 234–241, Pittsburgh, Pa, USA, July 2009.
- [16] F. Zhong, C. K. Yeo, and B. S. Lee, "Adaptive load balancing algorithm for multi-homing mobile nodes in local domain," in *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC '11)*, pp. 482–486, Las Vegas, Nev, USA, January 2011.
- [17] S. Mueller, R. P. Tsang, and D. Ghosal, "Multipath routing in mobile ad hoc networks: issues and challenges," *Performance Tools and Applications to Networked Systems*, vol. 2965, pp. 209–234, 2004.
- [18] V. Singh, S. Ahsan, and J. Ott, "MPRTP: multipath considerations for real-time media," in *Proceedings of the 4th ACM Multimedia Systems Conference (MMSys '13)*, pp. 190–201, New York, NY, USA, 2013.
- [19] R. Stewart and C. Metz, "SCTP: new transport protocol for TCP/IP," *IEEE Internet Computing*, vol. 5, no. 6, pp. 64–69, 2001.
- [20] J. Kilpi and I. Norros, "Testing the Gaussian approximation of aggregate traffic," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement (IMW '02)*, pp. 49–61, Marseille, France, November 2002.
- [21] R. van de Meent, M. Mandjes, and A. Pras, "Gaussian traffic everywhere?" in *Proceedings of the IEEE International Conference on Communications (ICC '06)*, pp. 573–578, Istanbul, Turkey, July 2006.
- [22] J. Li, C. Blake, D. S. J. de Couto, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom '01)*, pp. 61–69, Rome, Italy, July 2001.