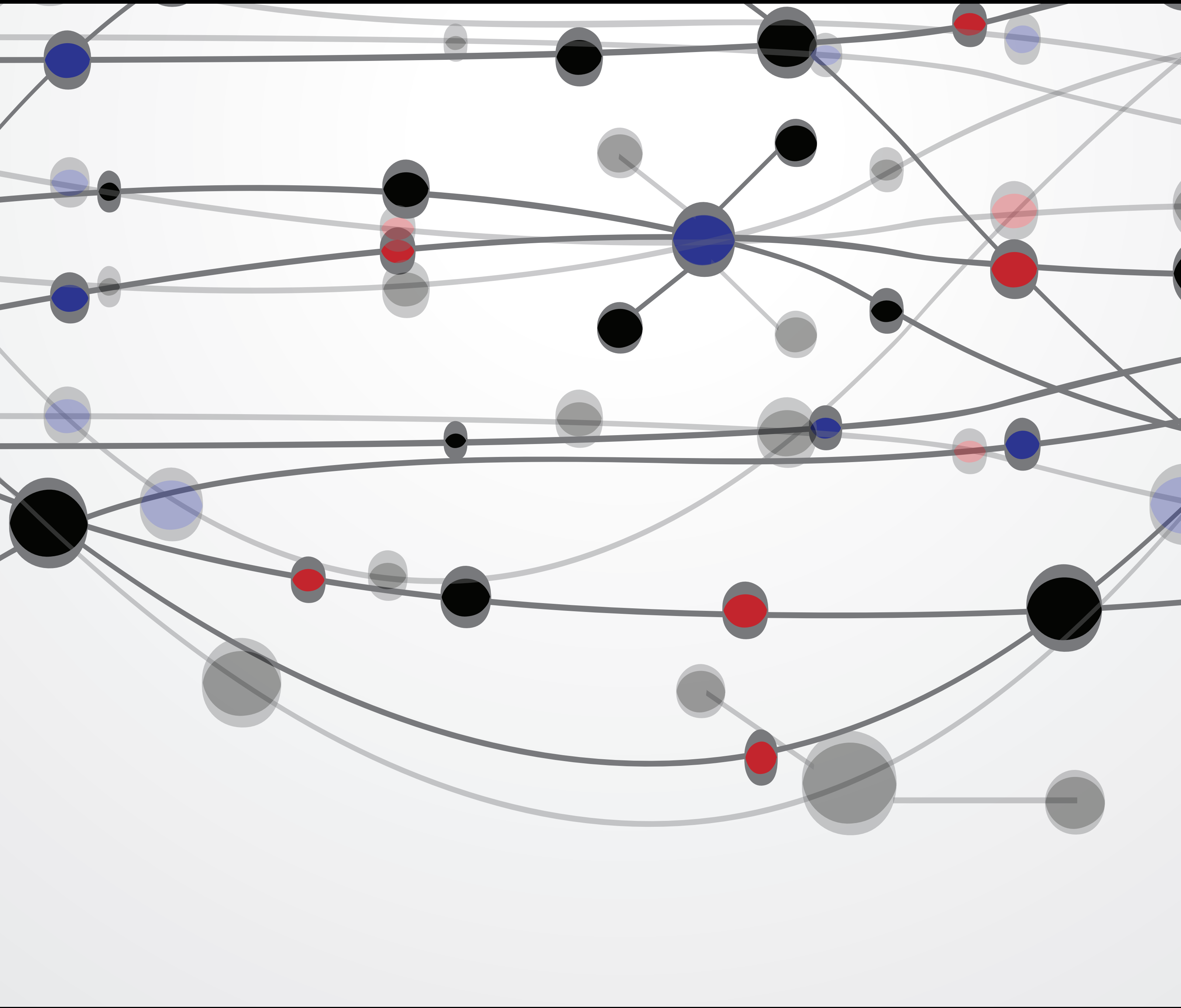
The Scientific World Journal

Swarm Intelligence and Its Applications 2014

Guest Editors: Yudong Zhang, Praveen Agarwal, Vishal Bhatnagar, Saeed Balochian, and Xuewu Zhang



Swarm Intelligence and Its Applications 2014

The Scientific World Journal

Swarm Intelligence and Its Applications 2014

Guest Editors: Yudong Zhang, Praveen Agarwal, Vishal Bhatnagar, Saeed Balochian, and Xuewu Zhang

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

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 204294, 4 pages
http://dx.doi.org/10.1155/2014/204294

Editorial

Swarm Intelligence and Its Applications 2014

Yudong Zhang,[1] Praveen Agarwal,[2] Vishal Bhatnagar,[3] Saeed Balochian,[4] and Xuewu Zhang[5]

[1] *School of Computer Science and Technology, Nanjing Normal University, Nanjing, Jiangsu 210023, China*
[2] *Department of Mathematics, Anand International College of Engineering, Agra Road, Near Bassi, Jaipur, Rajasthan 303012, India*
[3] *Ambedkar Institute of Advanced Communication Technologies and Research, Government of NCT of Delhi, Geeta Colony, Delhi 110031, India*
[4] *Department of Electrical Engineering, Gonabad Branch, Islamic Azad University, Gonabad, Khorasan-e-Razavi 96916-29, Iran*
[5] *MRI Lab, Columbia University, New York, NY 10032, USA*

Correspondence should be addressed to Yudong Zhang; zhangyudongnuaa@gmail.com

Received 11 June 2014; Accepted 11 June 2014; Published 23 June 2014

Copyright © 2014 Yudong Zhang 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.

Swarm intelligence (SI) represents the collective behavior of decentralized, self-organized systems. SI systems consist typically of a population of simple agents that interact locally with one another and with their environment. The inspiration of SI originates from biological systems. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local, and to a certain degree random, interactions between such agents lead to the emergence of intelligence, unknown to the individual agents. Natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling. Besides the applications to conventional optimization problems, SI is employed in various fields such as library materials acquisition, communications, medical dataset classification, dynamic control, heating system planning, moving objects tracking, pattern recognition, and statistical prediction.

The main objective of this special issue is to provide the readers with a collection of high quality research articles that address the broad challenges in application aspects of swarm intelligence and reflect the emerging trends in state-of-the-art algorithms.

The paper authored by Z.-C. Wang and X.-B. Wu (Tongji University) investigates the applicability and performance of biogeography-based optimization (BBO) for integer programming. They find that the original BBO algorithm does not perform well on a set of benchmark problems of integer programming. Hence, they modify the mutation operator and/or the neighborhood structure of the algorithm, resulting in three new BBO-based methods, named BlendBBO, BBO_DE, and LBBO_LDE, respectively. Computational experiments show that these methods are competitive approaches to solve integer-programming problems, and the LBBO_LDE shows the best performance on the benchmark problems.

In the paper by J. Wang et al. (North China Electric Power University), they model the complex process-planning problem as a combinatorial optimization problem with constraints. An ant colony optimization (ACO) approach is developed to deal with process planning problem by simultaneously considering activities such as sequencing operations, selecting manufacturing resources, and determining setup plans to achieve the optimal process plan. A weighted directed graph is conducted to describe the operations, precedence constraints between operations, and the possible visited path between operation nodes. A representation of process plan is described based on the weighted directed graph. Ant colony goes through the necessary nodes on the graph to achieve the optimal solution with the objective of minimizing total production costs. Two cases are carried out to study the influence of various parameters of ACO on the system performance. Extensive comparative experiments are conducted to demonstrate the feasibility and efficiency of the proposed approach.

R. Kalatehjari et al. (Universiti Teknologi Malaysia) apply particle swarm optimization (PSO) in three-dimensional (3D) slope stability problem to determine the critical slip surface (CSS) of soil slopes. A detailed description of adopted PSO is presented to provide a good basis for more contribution of this technique to the field of 3D slope stability problems. A general rotating ellipsoid shape is introduced as the specific particle for 3D slope stability analysis. A detailed sensitivity analysis is designed and performed to find the optimum values of parameters of PSO. Example problems are used to evaluate the applicability of PSO in determining the CSS of 3D slopes. The first example presents a comparison between the results of PSO and PLAXI-3D finite element software. The second example compares the ability of PSO to determine the CSS of 3D slopes with other optimization methods from the literature. The results demonstrate the efficiency and effectiveness of PSO in determining the CSS of 3D soil slopes.

Another paper is by Y. Sun et al. (Beijing University of Posts and Telecommunications, Columbia University). It focuses on how to outsource computation task to the cloud securely and proposes a secure outsourcing multiparty computation protocol on lattice-based encrypted data in two-cloud-server scenario. The main idea is to transform the outsourced data, respectively, encrypted by different users' public keys to the ones encrypted by the same two private keys of the two assisted servers, so that it is feasible to operate on the transformed cipher-texts to compute an encrypted result following the function to be computed. In order to keep the privacy of the result, the two servers cooperatively produce a custom-made result for each user that is authorized to get the result, so that all authorized users can recover the desired result while other unauthorized ones including the two servers cannot. Compared with previous research, the protocol is completely noninteractive between any users. Both of the computation and the communication complexities of each user in their solution are independent of the computing function.

In their paper, M.-Y. Ju et al. (National University of Tainan) propose a hybrid evolutionary algorithm using scalable encoding method for path planning problems. The scalable representation is based on binary tree structure encoding. To solve the problem of hybrid genetic algorithm and particle swarm optimization, the "dummy node" is added to the binary trees to deal with the different lengths of representations. The experimental results show that the proposed hybrid method uses fewer turning points than traditional evolutionary algorithms and generate shorter collision-free paths for mobile robot navigation.

The paper by H. Mo et al. (Harbin Engineering University, Harbin University of Commerce, University of Pretoria, and Shaoxing University) proposes a novel constrained multiobjective biogeography optimization algorithm (CMBOA). It is the first biogeography optimization algorithm for constrained multiobjective optimization. In CMBOA, a disturbance migration operator is designed to generate diverse feasible individuals, in order to promote the diversity of individuals on Pareto front. Infeasible individuals nearby feasible region are evolved to feasibility by recombining with their nearest nondominated feasible individuals. The convergence of CMBOA is proved by using probability theory. The performance of CMBOA is evaluated on a set of 6 benchmark problems. The experimental results show that the CMBOA performs better than or similar to the classical NSGA-II and IS-MOEA.

The paper authored by I. C. Obagbuwa and A. O. Adewumi (University of KwaZulu-Natal) introduces the hunger component to the existing cockroach swarm optimization (CSO) algorithm, to improve its searching ability and population diversity. The original CSO is modelled with three components: chase-swarming, dispersion, and ruthlessness; additional hunger component modelled using partial differential equation (PDE) method is included. The performance of the proposed algorithm is tested on well-known benchmarks and compared with the existing CSO, modified cockroach swarm optimization (MCSO), roach infestation optimization RIO, and hungry roach infestation optimization (HRIO). The comparison results show clearly that the proposed algorithm outperforms the existing algorithms.

In the paper by L. Liu et al. (Harbin Engineering University), they propose a distribution model of ant colony foraging, through analysis of the relationship between the position distribution and food source in the process of ant colony foraging. They design a continuous domain optimization algorithm based on the model. They give the form of solution for the algorithm, the distribution model of pheromone, the update rules of ant colony position, and the processing method of constraint condition. The algorithm is tested against a set of test trials by unconstrained optimization test functions and a set of optimization test functions. The results of other algorithms are compared and analyzed, to verify the correctness and effectiveness of the proposed algorithm.

A. Shabri and R. Samsudin (Universiti Teknologi Malaysia) propose a hybrid model integrating wavelet and multiple linear regressions (MLR) for crude oil price forecasting. In this model, Mallat wavelet transform is first selected to decompose an original time series into several subseries with different scale. Then, the principal component analysis (PCA) is used in processing subseries data in MLR for crude oil price forecasting. The particle swarm optimization (PSO) is used to adopt the optimal parameters of the MLR model. To assess the effectiveness of this model, daily crude oil market and West Texas Intermediate (WTI) are used as the case study. Time-series prediction capability performance of the WMLR model is compared with the MLR, ARIMA, and GARCH models using various statistics measures. The experimental results show that the proposed model outperforms the individual models in forecasting of the crude oil prices series.

In their paper, F. A. Ahmad et al. (Universiti Putra Malaysia) propose a new approach based on integrated intelligent system inspired by foraging of honeybees applied to multimobile robot scenario. This integrated approach caters for both working and foraging stages for known/unknown power station locations. Swarm mobile robot inspired by honeybee is simulated to explore and identify the power station for battery recharging. The mobile robots will share the location information of the power station with each other.

The results show that mobile robots consume less energy and less time when they are cooperating with each other for foraging process. The optimizing of foraging behavior will result in the mobile robots spending more time to do real work.

The paper by J.-Q. Li et al. (Northeastern University and Liaocheng University) proposes a hybrid algorithm that combines particle swarm optimization (PSO) and iterated local search (ILS) for solving the hybrid flow-shop scheduling (HFS) problem with preventive maintenance (PM) activities. In the proposed algorithm, different crossover operators and mutation operators are investigated. In addition, an efficient multiple insert mutation operator is developed for enhancing the searching ability of the algorithm. Furthermore, an ILS-based local search procedure is embedded in the algorithm to improve the exploitation ability of the proposed algorithm. The detailed experimental parameter for the canonical PSO is tuning. The proposed algorithm is tested on the variation of 77 Carlier and Néron's benchmark problems. Detailed comparisons with the present efficient algorithms, including hGA, ILS, PSO, and IG, verify the efficiency and effectiveness of the proposed algorithm.

The paper authored by Q. Xu et al. (Shandong University) proposes a fast elitism Gaussian estimation of distribution algorithm (FEGEDA). The Gaussian probability model is used to model the solution distribution. The parameters of Gaussian come from the statistical information of the best individuals by fast learning rule, which enhances the efficiency of the algorithm. An elitism strategy is used to maintain the convergent performance. The performances of the algorithm are examined based upon several benchmarks. In the simulations, a one-dimensional benchmark is used to visualize the optimization process and probability model learning process during the evolution, and several two-dimensional and higher dimensional benchmarks are used to testify the performance of FEGEDA. The experimental results indicate the capability of FEGEDA, especially in the higher dimensional problems, and the FEGEDA exhibits a better performance than some other algorithms and EDAs. Finally, FEGEDA is used in PID controller optimization of PMSM and is compared with the classical PID and GA.

In the paper by K. S. Lim et al. (Universiti Teknologi Malaysia, Universiti Malaysia Pahang, Universiti Malaya, and Hanbat National University), their research incorporates the concept of multiple nondominated leaders to further improve the vector evaluated particle swarm optimization (VEPSO) algorithm. Multiple nondominated solutions that are best at a respective objective function are used to guide particles in finding optimal solutions. The improved VEPSO is measured by the number of nondominated solutions found, generational distance, spread, and hypervolume. The results from the conducted experiments show that the proposed VEPSO significantly improves the existing VEPSO algorithms.

Z. Yin et al. (Harbin Institute of Technology) focus on multiuser detection in tracking and data relay satellite (TDRS) system forward link. Minimum mean square error (MMSE) is a low complexity multiuser detection method, but MMSE detector cannot achieve satisfactory bit error ratio and near-far resistance, whereas artificial fish swarm algorithm (AFSA) is expert in optimization and it can realize the global convergence efficiently. Therefore, a hybrid multiuser detector based on MMSE and AFSA (MMSE-AFSA) is proposed. The result of MMSE and its modified formations are used as the initial values of artificial fishes to accelerate the speed of global convergence and reduce the iteration times for AFSA. The simulation results show that the bit error ratio and near-far resistance performances of the proposed detector are much better, compared with MF, DEC, and MMSE, and are quite close to OMD. Furthermore, the proposed MMSE-AFSA detector also has a large system capacity.

In their paper, S. Molla-Alizadeh-Zavardehi et al. (Islamic Azad University and University of Tehran) deal with a problem of minimizing total weighted tardiness of jobs in a real-world single batch-processing machine (SBPM) scheduling in the presence of fuzzy due date. First, a fuzzy mixed integer linear programming model is developed. Then, due to the complexity of the problem that is NP hard, they design two hybrid metaheuristics called GA-VNS and VNS-SA applying the advantages of genetic algorithm (GA), variable neighborhood search (VNS), and simulated annealing (SA) frameworks. Besides, they propose three fuzzy earliest due date heuristics to solve the given problem. Through computational experiments with several random test problems, a robust calibration is applied on the parameters. Finally, computational results on different-scale test problems are presented to compare the proposed algorithms.

The paper by H. Liu et al. (Beijing Institute of Technology, University of Science and Technology Liaoning, and Nanchang University) presents a human behavior-based PSO, which is called HPSO. There are two remarkable differences between PSO and HPSO. First, the global worst particle is introduced into the velocity equation of PSO, which is endowed with random weight that obeys the standard normal distribution; this strategy is conducive to trade off exploration and exploitation ability of PSO. Second, the two acceleration coefficients c_1 and c_2 in the standard PSO (SPSO) are eliminated to reduce the parameters sensitivity of solved problems. Experimental results on 28 benchmark functions, which consist of unimodal, multimodal, rotated, and shifted high-dimensional functions, demonstrate the high performance of the proposed algorithm in terms of convergence accuracy and speed with lower computation cost.

The paper authored by B. Crawford et al. (Pontificia Universidad Católica de Valparaíso, Universidad Finis Terrae, Universidad Autónoma de Chile, and Universidad Diego Portales) presents a novel application of the artificial bee colony algorithm to solve the nonunicost set covering problem. The artificial bee colony algorithm is a recent swarm metaheuristic technique based on the intelligent foraging behavior of honey bees. Experimental results show that the artificial bee colony algorithm is competitive in terms of solution quality with other recent metaheuristic approaches for the set covering problem.

In the paper by J.-S. Wang et al. (University of Science & Technology Liaoning), they propose an echo state network (ESN) based fusion soft-sensor model optimized by the improved glowworm swarm optimization (GSO) algorithm, for predicting the key technology indicators (concentrate grade and tailings recovery rate) of flotation process. Firstly, the color feature (saturation and brightness) and texture features (angular second moment, sum entropy, inertia moment, etc.) based on grey-level cooccurrence matrix (GLCM) are adopted to describe the visual characteristics of the flotation froth image. Then, the kernel principal component analysis (KPCA) method is used to reduce the dimensionality of the high-dimensional input vector composed by the flotation froth image characteristics and process datum and extracts the nonlinear principal components in order to reduce the ESN dimension and network complex. The ESN soft-sensor model of flotation process is optimized by the GSO algorithm with congestion factor. Simulation results show that the model has better generalization and prediction accuracy to meet the online soft-sensor requirements of the real-time control in the flotation process.

B. Li et al. (Shandong University and Qilu University of Technology) propose a novel KELM learning algorithm using the PSO approach to optimize the parameters of kernel functions of neural networks, which is called the AKELM learning algorithm, for improving the prediction accuracy of robot execution failures. The simulation results with the robot execution failures datasets show that, by optimizing the kernel parameters, the proposed algorithm has good generalization performance and outperforms KELM and the other approaches in terms of classification accuracy. Other benchmark problems simulation results also show the efficiency and effectiveness of the proposed algorithm.

In the paper by T. S. Kiong et al. (Universiti Tenaga Nasional, Universiti Kebangsaan Malaysia), their research considers the adaptive beamforming technique used to cancel interfering signals (placing nulls) and produce or steer a strong beam toward the target signal according to the calculated weight vectors. Minimum variance distortion response (MVDR) beamforming is capable of determining the weight vectors for beam steering; however, its nulling level on the interference sources remains unsatisfactory. Beamforming can be considered as an optimization problem, such that optimal weight vector should be obtained through computation. Hence, in their paper, a new dynamic mutated artificial immune system (DM-AIS) is proposed to enhance MVDR beamforming for controlling the null steering of interference and increase the signal to interference-noise ratio (SINR) for wanted signals.

Finally, F. Zou et al. (Xi'an University of Technology, Huaibei Normal University) present a new teaching-learning-based optimization (TLBO) variant called barebones teaching-learning-based optimization (BBTLBO), to solve the global optimization problems. In their method, each learner of teacher phase employs an interactive learning strategy, which is the hybridization of the learning strategy of teacher phase in the standard TLBO and Gaussian sampling learning based on neighborhood search, and each learner of learner phase employs the learning strategy of learner phase in the standard TLBO or the new neighborhood search strategy. To verify the performance of their approaches, 20 benchmark functions and 2 real-world problems are utilized. Conducted experiments can be observed that the BBTLBO performs significantly better than, or at least comparable to, TLBO and some existing bare-bone algorithms. The results indicate that the proposed algorithm is competitive to some other optimization algorithms. We expect that this special issue offers a comprehensive and timely view of the area of applications of SI and that it will offer stimulation for further research.

Acknowledgments

We would like to express our gratitude to all of the authors for their contributions and the reviewers for their effort providing valuable comments and feedback.

Yudong Zhang
Praveen Agarwal
Vishal Bhatnagar
Saeed Balochian
Xuewu Zhang

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 136920, 17 pages
http://dx.doi.org/10.1155/2014/136920

Research Article

Bare-Bones Teaching-Learning-Based Optimization

Feng Zou,[1,2] Lei Wang,[1] Xinhong Hei,[1] Debao Chen,[2] Qiaoyong Jiang,[1] and Hongye Li[1]

[1] *School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China*
[2] *School of Physics and Electronic Information, Huaibei Normal University, Huaibei 235000, China*

Correspondence should be addressed to Lei Wang; wangleeei@163.com

Received 20 February 2014; Accepted 7 April 2014; Published 10 June 2014

Academic Editors: S. Balochian and Y. Zhang

Copyright © 2014 Feng Zou 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.

Teaching-learning-based optimization (TLBO) algorithm which simulates the teaching-learning process of the class room is one of the recently proposed swarm intelligent (SI) algorithms. In this paper, a new TLBO variant called bare-bones teaching-learning-based optimization (BBTLBO) is presented to solve the global optimization problems. In this method, each learner of teacher phase employs an interactive learning strategy, which is the hybridization of the learning strategy of teacher phase in the standard TLBO and Gaussian sampling learning based on neighborhood search, and each learner of learner phase employs the learning strategy of learner phase in the standard TLBO or the new neighborhood search strategy. To verify the performance of our approaches, 20 benchmark functions and two real-world problems are utilized. Conducted experiments can been observed that the BBTLBO performs significantly better than, or at least comparable to, TLBO and some existing bare-bones algorithms. The results indicate that the proposed algorithm is competitive to some other optimization algorithms.

1. Introduction

Many real-life optimization problems are becoming more and more complex and difficult with the development of scientific technology. So how to resolve these complex problems in an exact manner within a reasonable time cost is very important. The traditional optimization algorithms are difficult to solve these complex nonlinear problems. In recent years, nature-inspired optimization algorithms which simulate natural phenomena and have different design philosophies and characteristics, such as evolutionary algorithms [1–3] and swarm intelligence algorithms [4–7], are a research field which simulates different natural phenomena to solve a wide range of problems. In these algorithms the convergence rate of the algorithm is given prime importance for solving real-world optimization problems. The ability of the algorithms to obtain the global optima value is one aspect and the faster convergence is the other aspect.

As a stochastic search scheme, TLBO [8, 9] is a newly population-based algorithm based on swarm intelligence and has characters of simple computation and rapid convergence; it has been extended to the function optimization, engineering optimization, multiobjective optimization, clustering, and so forth [9–17]. TLBO is a parameter-free evolutionary technique and is also gaining popularity due to its ability to achieve better results in comparatively faster convergence time to genetic algorithms (GA) [1], particle swarm optimizer (PSO) [5], and artificial bee colony algorithm (ABC) [6]. However, in evolutionary computation research there have been always attempts to improve any given findings further and further. This work is an attempt to improve the convergence characteristics of TLBO further without sacrificing the accuracies obtained in TLBO and in some occasions trying to even better the accuracies. The aims of this paper are of threefold. First, authors propose an improved version of TLBO, namely, BBTLBO. Next, the proposed technique is validated on unimodal and multimodal functions based on different performance indicators. The result of BBTLBO is compared with other algorithms. Results of both the algorithms are also compared using statistical paired t-test. Thirdly, it is applied to solve the real-world optimization problem.

The remainder of this paper is organized as follows. The TLBO algorithm is introduced in Section 2. Section 3 presents a brief overview of some recently proposed

```
(1) Begin
(2)     Initialize $N$ (number of learners) and $D$ (number of dimensions)
(3)     Initialize learners $X$ and evaluate all learners $X$
(4)     Donate the best learner as Teacher and the mean of all learners $X$ as Mean
(5)     while (stopping condition not met)
(6)         for each learner $X_i$ of the class % Teaching phase
(7)             $\mathrm{TF} = \mathrm{round}(1 + \mathrm{rand}(0,1))$
(8)             for $j = 1:D$
(9)                 $newX_{ij} = X_{ij} + \mathrm{rand}(0,1) * (Teacher(j) - \mathrm{TF} * Mean(j))$
(10)            endfor
(11)            Accept $newX_i$ if $f(newX_i)$ is better than $f(X_i)$
(12)        endfor
(13)        for each learner $X_i$ of the class % Learning phase
(14)            Randomly select one learner $X_k$, such that $i \neq k$
(15)            if $f(X_i)$ better $f(X_k)$
(16)                for $j = 1:D$
(17)                    $newX_{ij} = X_{ij} + \mathrm{rand}(0,1) * (X_{ij} - X_{kj})$
(18)                endfor
(19)            else
(20)                for $j = 1:D$
(21)                    $newX_{ij} = X_{ij} + \mathrm{rand}(0,1) * (X_{kj} - X_{ij})$
(22)                endfor
(23)            endif
(24)            Accept $newX_i$ if $f(newX_i)$ is better than $f(X_i)$
(25)        endfor
(26)        Update the Teacher and the Mean
(27)    endwhile
(28) end
```

Algorithm 1: TLBO( ).

bare-bones algorithms. Section 4 describes the improved teaching-learning-based optimization algorithm using neighborhood search (BBTLBO). Section 5 presents the tests on several benchmark functions and the experiments are conducted along with statistical tests. The applications for training artificial neural network are shown in Section 6. Conclusions are given in Section 7.

2. Teaching-Learning-Based Optimization

Rao et al. [8, 9] first proposed a novel teaching-learning-based optimization (TLBO) inspired from the philosophy of teaching and learning. The TLBO algorithm is based on the effect of the influence of a teacher on the output of learners in a class which is considered in terms of results or grades. The process of working of TLBO is divided into two parts. The first part consists of "teacher phase" and the second part consists of "learner phase." The "teacher phase" means learning from the teacher and the "learner phase" means learning through the interaction between learners.

A good teacher is one who brings his or her learners up to his or her level in terms of knowledge. But in practice this is not possible and a teacher can only move the mean of a class up to some extent depending on the capability of the class. This follows a random process depending on many factors. Let M be the mean and let T be the teacher at any iteration. T will try to move mean M toward its own level, so now the new mean will be T designated as M_{new}. The solution is updated according to the difference between the existing and the new mean according to the following expression:

$$newX = X + r \times \left(M_{\text{new}} - \mathrm{TF} \times M\right), \tag{1}$$

where TF is a teaching factor that decides the value of mean to be changed and r is a random vector in which each element is a random number in the range $[0, 1]$. The value of TF can be either 1 or 2, which is again a heuristic step and decided randomly with equal probability as

$$\mathrm{TF} = \mathrm{round}\,[1 + \mathrm{rand}\,(0, 1)]\,. \tag{2}$$

Learners increase their knowledge by two different means: one through input from the teacher and the other through interaction between themselves. A learner interacts randomly with other learners with the help of group discussions, presentations, formal communications, and so forth. A learner learns something new if the other learner has more knowledge than him or her. Learner modification is expressed as

$$newX_i = \begin{cases} X_i + r * \left(X_i - X_j\right) & \text{if } f\left(X_i\right) < f\left(X_j\right) \\ X_i + r * \left(X_j - X_i\right) & \text{otherwise.} \end{cases} \tag{3}$$

As explained above, the pseudocode for the implementation of TLBO is summarized in Algorithm 1.

3. Bare-Bones Algorithm

In this section, we only presented a brief overview of some recently proposed bare-bones algorithms.

3.1. BBPSO and BBExp. PSO is a swarm intelligence-based algorithm, which is inspired by the behavior of birds flocking [5]. In PSO, each particle is attracted by its personal best position (p_{best}) and the global best position (g_{best}) found so far. Theoretical studies [18, 19] proved that each particle converges to the weighted average of p_{best} and g_{best}:

$$\lim_{t\to\infty} X_i(t) = \frac{c_1 \cdot g_{\text{best}} + c_2 \cdot p_{\text{best}}}{c_1 + c_2}, \tag{4}$$

where c_1 and c_2 are two leaning factors in PSO.

Based on the convergence characteristic of PSO, Kennedy [20] proposed a new PSO variant called bare-bones PSO (BBPSO). Bare-bones PSO retains the standard PSO social communication but replaces dynamical particle update with sampling from a probability distribution based on g_{best} and p_{best_i} as follows:

$$x_{i,j}(t+1) = N\left(\frac{g_{\text{best}} + p_{\text{best}_{i,j}}(t)}{2}, \left|g_{\text{best}} - p_{\text{best}_{i,j}}(t)\right|\right), \tag{5}$$

where $x_{i,j}(t+1)$ is the jth dimension of the ith particle in the population and N represents a Gaussian distribution with mean $(g_{\text{best}} + p_{\text{best}_{i,j}}(t))/2$ and standard deviation $|g_{\text{best}} - p_{\text{best}_{i,j}}(t)|$.

Kennedy [20] proposed also an alternative version of the BBPSO, denoted by BBExp, where (5) is replaced by

$$x_{i,j}(t+1) = \begin{cases} N\left(\frac{g_{\text{best}} + p_{\text{best}_{i,j}}(t)}{2}, \left|g_{\text{best}} - p_{\text{best}_{i,j}}(t)\right|\right) & \text{rand}(0,1) > 0.5 \\ p_{\text{best}_{i,j}}(t) & \text{otherwise,} \end{cases} \tag{6}$$

where rand (0,1) is a random value within $[0, 1]$ for the jth dimension. For the alternative mechanism, there is a 50% chance that the search process is focusing on the previous best positions.

3.2. BBDE, GBDE, and MGBDE. Inspired by the BBPSO and DE, Omran et al. [21] proposed a new and efficient DE variant, called bare-bones differential evolution (BBDE). The BBDE is a new, almost parameter-free optimization algorithm that is a hybrid of the bare-bones particle swarm optimizer and differential evolution. Differential evolution is used to mutate, for each particle, the attractor associated with that particle, defined as a weighted average of its personal and neighborhood best positions. For the BBDE, the individual is updated as follows:

$$x_{i,j}(t+1) = \begin{cases} p_{i_3,j}(t) + r_2 \cdot \left(x_{i_1,j}(t) - x_{i_2,j}(t)\right) & \text{rand}(0,1) > \text{CR} \\ p_{\text{best}_{i3,j}}(t) & \text{otherwise,} \end{cases} \tag{7}$$

where i_1, i_2, and i_3 are three indices chosen from the set $\{1, 2, \ldots, \text{NP}\}$ with $i_1 \neq i_2 \neq i$, rand (0, 1) is a random value within [0, 1] for the jth dimension, and $p_{i,j}(t)$ is defined by

$$p_{i,j}(t+1) = r_{1,j} \cdot p_{\text{best}_{i,j}}(t) + \left(1 - r_{2,j}\right) g_{\text{best}_i}(t), \tag{8}$$

where p_{best} and g_{best} are personal best position and the global best position, $r_{1,j}$, is a random value within [0, 1] for the jth dimension.

Based on the idea that the Gaussian sampling is a fine tuning procedure which starts during exploration and is continued to exploitation, Wang et al. [22] proposed a new parameter-free DE algorithm, called GBDE. In the GBDE, the mutation strategy uses a Gaussian sampling method which is defined by

$$v_{i,j}(t+1) = \begin{cases} N\left(\frac{X_{\text{best},j}(t) + x_{i,j}(t)}{2}, \left|X_{\text{best},j}(t) - x_{i,j}(t)\right|\right) & \text{rand}(0,1) \le \text{CR} \vee j = j_{\text{rand}} \\ x_{i,j}(t) & \text{otherwise,} \end{cases} \tag{9}$$

where N represents a Gaussian distribution with mean $(X_{\text{best},j}(t)+x_{i,j}(t))/2$ and standard deviation $|X_{\text{best},j}(t)-x_{i,j}(t)|$ and CR is the probability of crossover.

To balance the global search ability and convergence rate, Wang et al. [22] proposed a modified GBDE (called MGBDE). The mutation strategy uses a hybridization of GBDE and DE/best/1 as follows:

$$v_{i,j}(t+1) = \begin{cases} X_{\text{best},j}(t) + F \cdot \left(x_{i_1,j}(t) - x_{i_2,j}(t)\right) & \text{rand}(0,1) \le 0.5 \\ N\left(\frac{X_{\text{best},j}(t)+x_{i,j}(t)}{2}, \left|X_{\text{best},j}(t)-x_{i,j}(t)\right|\right) & \text{otherwise.} \end{cases} \tag{10}$$

4. Proposed Algorithm: BBTLBO

The bare-bones PSO utilizes this information by sampling candidate solutions, normally distributed around the formally derived attractor point. That is, the new position is generated by a Gaussian distribution for sampling the search space based on the g_{best} and the p_{best} at the current iteration. As a result, the new position will be centered around the weighted average of p_{best} and g_{best}. Generally speaking, at the initial evolutionary stages, the search process focuses on exploration due to the large deviation. With an increasing number of generations, the deviation becomes smaller, and the search process will focus on exploitation. From the search behavior of BBPSO, the Gaussian sampling is a fine tuning procedure which starts during exploration and is continued to exploitation. This can be beneficial for the search of many evolutionary optimization algorithms. Additionally, the bare-bones PSO has no parameters to be tuned.

Based on a previous explanation, a new bare-bones TLBO (BBTLBO) with neighborhood search is proposed in this

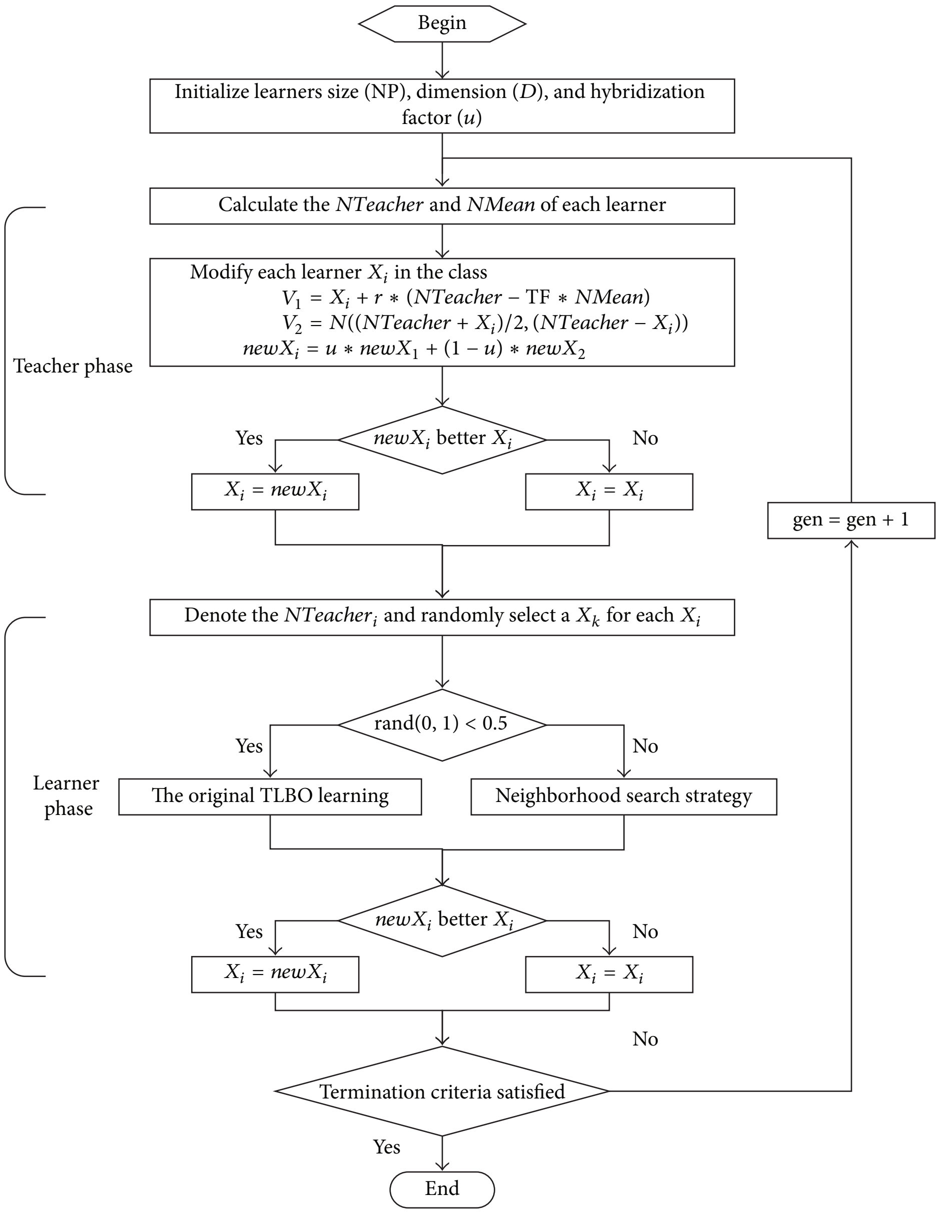

Begin
Initialize learners size (NP), dimension (D), and hybridization factor (u)
Calculate the $NTeacher$ and $NMean$ of each learner
Modify each learner X_i in the class
$V_1 = X_i + r * (NTeacher - \mathrm{TF} * NMean)$
$V_2 = N((NTeacher + X_i)/2, (NTeacher - X_i))$
$newX_i = u * newX_1 + (1 - u) * newX_2$
Teacher phase
$newX_i$ better X_i
Yes
No
$X_i = newX_i$
$X_i = X_i$
gen = gen + 1
Denote the $NTeacher_i$ and randomly select a X_k for each X_i
rand(0, 1) < 0.5
Learner phase
The original TLBO learning
Neighborhood search strategy
Termination criteria satisfied
End

Figure 1: Flow chart showing the working of BBTLBO algorithm.

paper. In fact, for TLBO, if the new learner has a better function value than that of the old learner, it is replaced with the old one in the memory. Otherwise, the old one is retained in the memory. In other words, a greedy selection mechanism is employed as the selection operation between the old and the candidate one. Hence, the new teacher and the new learner are the global best (g_{best}) and learner's personal best (p_{best}) found so far, respectively. The complete flowchart of the BBTLBO algorithm is shown in Figure 1.

4.1. Neighborhood Search. It is known that birds of a feather flock together and people of a mind fall into the same group. Just like evolutionary algorithms themselves, the notion of neighborhood is inspired by nature. Neighborhood technique is an efficient method to maintain diversity of the solutions. It plays an important role in evolutionary algorithms and is often introduced by researchers in order to allow maintenance of a population of diverse individuals and improve the exploration capability of population-based heuristic algorithms [23–26]. In fact, learners with similar interests form different learning groups. Because of his or her favor characteristic, the learner maybe learns from the excellent individual in the learning group.

For the implementation of grouping, various types of connected distances may be used. Here we have used a ring topology [27] based on the indexes of learners for the sake of simplicity. In a ring topology, the first individual is the neighbor of the last individual and vice versa. Based on the ring topology, a k-neighborhood radius is defined, where k is a predefined integer number. For each individual,

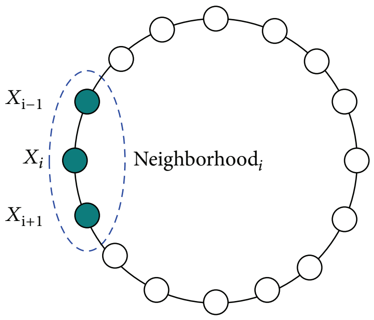


FIGURE 2: Ring neighborhood topology with three members.

its k-neighborhood radius consists of $2k + 1$ individuals (including oneself), which are $X_{i-k}, \ldots, X_i, \ldots, X_{i+k}$. That is, the neighborhood size is $2k + 1$ for a k-neighborhood. For simplicity, k is set to 1 (Figure 2) in our algorithm. This means that there are 3 individuals in each learning group. Once groups are constructed, we can utilize them for updating the learners of the corresponding group.

4.2. Teacher Phase. To balance the global and local search ability, a modified interactive learning strategy is proposed in teacher phase. In this learning phase, each learner employs an interactive learning strategy (the hybridization of the learning strategy of teacher phase in the standard TLBO and Gaussian sampling learning) based on neighborhood search.

In BBTLBO, the updating formula of the learning for a learner X_i in teacher phase is proposed by the hybridization of the learning strategy of teacher phase and the Gaussian sampling learning as follows:

$$\begin{aligned} V_{1,j}(t+1) &= X_{i,j}(t) + \operatorname{rand}(0,1) \\ &\quad \cdot \left(NTeacher_{i,j}(t) - \mathrm{TF} \cdot NMean_{i,j}(t)\right), \\ V_{2,j}(t+1) &= N\left(\frac{NTeacher_{i,j}(t) + NMean_{i,j}(t)}{2},\right. \\ &\quad \left. \left| NTeacher_{i,j}(t) - NMean_{i,j}(t) \right| \right), \\ newX_{i,j}(t+1) &= u \cdot V_{1,j}(t+1) + (1-u) \cdot V_{2,j}(t+1), \end{aligned} \tag{11}$$

where u called the hybridization factor is a random number in the range $[0, 1]$ for the jth dimension, *NTeacher* and *NMean* are the existing neighborhood best solution and the neighborhood mean solution of each learner, and TF is a teaching factor which can be either 1 or 2 randomly.

In the BBTLBO, there is a $(u * 100)\%$ chance that the jth dimension of the ith learner in the population follows the behavior of the learning strategy of teacher phase, while the remaining $(100 - u * 100)\%$ follow the search behavior of the Gaussian sampling in teacher phase. This will be helpful to balance the advantages of fast convergence rate (the attraction of the learning strategy of teacher phase) and exploration (the Gaussian sampling) in BBTLBO.

4.3. Learner Phase. At the same time, in the learner phase, a learner interacts randomly with other learners for enhancing his or her knowledge in the class. This learning method can be treated as the global search strategy (shown in (3)).

In this paper, we introduce a new learning strategy in which each learner learns from the neighborhood teacher and the other learner selected randomly of his or her corresponding neighborhood in learner phase. This learning method can be treated as the neighborhood search strategy. Let $newX_i$ represent the interactive learning result of the learner X_i. This neighborhood search strategy can be expressed as follows:

$$\begin{aligned} newX_{i,j} &= X_{i,j} + r_1 * \left(NTeacher_{i,j} - X_{i,j}\right) \\ &\quad + r_2 * \left(X_{i,j} - X_{k,j}\right), \end{aligned} \tag{12}$$

where r_1 and r_2 are random vectors in which each element is a random number in the range $[0, 1]$, *NTeacher* is the teacher of the learner X_i's corresponding neighborhood, and the learner X_k is selected randomly from the learner's corresponding neighborhood.

In BBTLBO, each learner is probabilistically learning by means of the global search strategy or the neighborhood search strategy in learner phase. That is, about 50% of learners in the population execute the learning strategy of learner phase in the standard TLBO (shown in (3)), while the remaining 50% execute neighborhood search strategy (shown in (12)). This will be helpful to balance the global search and local search in learner phase.

Moreover, compared to the original TLBO, BBTLBO only modifies the learning strategies. Therefore, both the original TLBO and BBTLBO have the same time complexity $O(\mathrm{NP} \cdot D \cdot \mathrm{Gen}_{\max})$, where NP is the number of the population, D is the number of dimensions, and $\mathrm{Gen}_{\max}$ is the maximum number of generations.

As explained above, the pseudocode for the implementation of BBTLBO is summarized in Algorithm 2.

5. Functions Optimization

In this section, to illustrate the effectiveness of the proposed method, 20 benchmark functions are used to test the efficiency of BBTLBO. To compare the search performance of BBTLBO with some other methods, other different algorithms are also simulated in the paper.

5.1. Benchmark Functions. The details of 20 benchmark functions are shown in Table 1. Among 20 benchmark functions, F_1 to F_9 are unimodal functions, and F_{10} to F_{20} are multimodal functions. The searching range and theory optima for all functions are also shown in Table 1.

5.2. Parameter Settings. All the experiments are carried out on the same machine with a Celoron 2.26 GHz CPU, 2 GB memory, and Windows XP operating system with Matlab 7.9.

```
(1) Begin
(2)     Initialize $N$ (number of learners), $D$ (number of dimensions) and hybridization factor $u$
(3)     Initialize learners $X$ and evaluate all learners $X$
(4)     while (stopping condition not met)
(5)         for each learner $X_i$ of the class % Teaching phase
(6)             TF = round(1 + rand(0, 1))
(7)             Donate the $N$ Teacher and the $N$ Mean in its neighborhood for each learner
(8)             Updating each learner according (11)
(9)             Accept $newX_i$ if $f(newX_i)$ is better than $f(X_i)$
(10)        endfor
(11)        for each learner $X_i$ of the class % Learning phase
(12)            Randomly select one learner $X_k$, such that $i \neq k$
(13)            if rand(0, 1) < 0.5
(14)                Updating each learner according (3)
(15)            else
(16)                Donate the $N$ Teacher in its neighborhood for each learner
(17)                Updating each learner according (12)
(18)            endif
(19)            Accept $newX_i$ if $f(newXi)$ is better than $f(X_i)$
(20)        endfor
(21)    endwhile
(22) end
```

Algorithm 2: BBTLBO( ).

For the purpose of reducing statistical errors, each algorithm is independently simulated 50 times. For all algorithms, the population size was set to 20. Population-based stochastic algorithms use the same stopping criterion, that is, reaching a certain number of function evaluations (FEs).

5.3. Effect of Variation in Parameter u. The hybridization factor u is set to {0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0}. Comparative tests have been performed using different u. In our experiment, the maximal FEs are used as ended condition of algorithm, namely, 40,000 for all test functions. Table 2 shows the mean optimum solutions and the standard deviation of the solutions obtained using different hybridization factor u in the 50 independent runs. The best results among the algorithms are shown in bold. Figure 3 presents the representative convergence graphs of different benchmark functions in terms of the mean fitness values achieved by using different hybridization factor u on all test functions. Due to the tight space limitation, some sample graphs are illustrated.

The comparisons in Table 2 and Figure 3 show that when the hybridization factor u is set to 0.9, BBTLBO offers the best performance on 20 test functions. Hence, the hybridization factor u is set to 0.9 in the following experiments.

5.4. Comparison of BBTLBO with Some Similar Bare-Bones Algorithms. In this section, we compare BBTLBO with five other recently proposed three bare-bones DE variants and two bare-bones PSO algorithms. Our experiment includes two series of comparisons in terms of the solution accuracy and the solution convergence (convergence speed and success rate). We compared the performance of BBTLBO with other similar bare-bones algorithms, including BBPSO [20], BBExp [20], BBDE [21], GBDE [22], and MGBDE [22].

5.4.1. Comparisons on the Solution Accuracy. In our experiment, the maximal FEs are used as ended condition of algorithm, namely, 40,000 for all test functions. The results are shown in Table 3 in terms of the mean optimum solution and the standard deviation of the solutions obtained in the 50 independent runs by each algorithm on 20 test functions. The best results among the algorithms are shown in bold. Figure 4 presents the convergence graphs of different benchmark functions in terms of the mean fitness values achieved by 7 algorithms for 50 independent runs. Due to the tight space limitation, some sample graphs are illustrated.

From Table 3 it can be observed that the mean optimum solution and the standard deviation of all algorithms perform well for the functions F_{15} and F_{17}. Although BBExp performs better than BBTLBO on function F_9 and MGBDE performs better than BBTLBO on function F_{20}, our approach BBTLBO achieves better results than other algorithms on the rest of test functions. Table 3 and Figure 4 conclude that the BBTLBO has a good performance of the solution accuracy for test functions in this paper.

5.4.2. Comparison of the Convergence Speed and SR. In order to compare the convergence speed and successful rate (SR) of different algorithms, we select a threshold value of the objective function for each test function. For other functions, the threshold values are listed in Table 4. In our experiment, the stopping criterion is that each algorithm is terminated when the best fitness value so far is below the predefined threshold value (T Value) or the number of FEs reaches to

Table 1: Details of numerical benchmarks used.

Function	Formula	D	Range	Optima
Sphere	$F_1(x)=\sum_{i=1}^{D}x_i^2$	30	[−100, 100]	0
Sum square	$F_2(x)=\sum_{i=1}^{D}ix_i^2$	30	[−100, 100]	0
Quadric	$F_3(x)=\sum_{i=1}^{D}ix_i^4+\text{random}(0,1)$	30	[−1.28, 1.28]	0
Step	$F_4(x)=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2$	30	[−100, 100]	0
Schwefel 1.2	$F_5(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	30	[−100, 100]	0
Schwefel 2.21	$F_6(x)=\max\{\lvert x_i\rvert, 1\le i\le D\}$	30	[−100, 100]	0
Schwefel 2.22	$F_7(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	30	[−10, 10]	0
Zakharov	$F_8(x)=\sum_{i=1}^{D}x_i^2+\left(\sum_{i=1}^{D}0.5ix_i\right)^2+\left(\sum_{i=1}^{D}0.5ix_i\right)^4$	30	[−100, 100]	0
Rosenbrock	$F_9(x)=\sum_{i=1}^{D-1}\left[100\left(x_i^2-x_{i+1}\right)^2+(x_i-1)^2\right]$	30	[−2.048, 2.048]	0
Ackley	$F_{10}(x)=20-20\exp\left(\left(-\frac{1}{5}\right)\sqrt{\left(\frac{1}{D}\right)\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\left(\frac{1}{D}\right)\sum_{i=1}^{D}\cos(2\pi x_i)\right)+e$	30	[−32, 32]	0
Rastrigin	$F_{11}(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	30	[−5.12, 5.12]	0
Weierstrass	$F_{12}(x)=\sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k(x_i+0.5)\right)\right]\right)-D\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\times 0.5\right)\right]$ $a=0.5\quad b=3\quad k\max=20$	30	[−0.5, 0.5]	0
Griewank	$F_{13}(x)=\sum_{i=1}^{D}\left(\frac{x_i^2}{4000}\right)-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[−600, 600]	0
Schwefel	$F_{14}(x)=418.9829D+\sum_{i=1}^{D}(-x_i\sin\sqrt{\text{abs}(x_i)})$	30	[−500, 500]	0
Bohachevsky1	$F_{15}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$	2	[−100, 100]	0
Bohachevsky2	$F_{16}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)*\cos(4\pi x_2)+0.3$	2	[−100, 100]	0
Bohachevsky3	$F_{17}(x)=x_1^2+2x_2^2-0.3\cos((3\pi x_1)+(4\pi x_2))+0.3$	2	[−100, 100]	0
Shekel5	$F_{18}(x)=-\sum_{i=1}^{5}\left[(x-a_i)(x-a_i)^T+c_i\right]^{-1}$	4	[0, 10]	−10.1532
Shekel7	$F_{19}(x)=-\sum_{i=1}^{7}\left[(x-a_i)(x-a_i)^T+c_i\right]^{-1}$	4	[0, 10]	−10.4029
Shekel10	$F_{20}(x)=-\sum_{i=1}^{10}\left[(x-a_i)(x-a_i)^T+c_i\right]^{-1}$	4	[0, 10]	−10.5364

the maximal FEs 40,000. The results are shown in Table 4 in terms of the mean number of FEs (MFEs) required to converge to the threshold and successful rate (SR) in the 50 independent runs. “NaN” represents that no runs of the corresponding algorithm converged below the predefined threshold before meeting the maximum number of FEs. The best results among the six algorithms are shown in boldface.

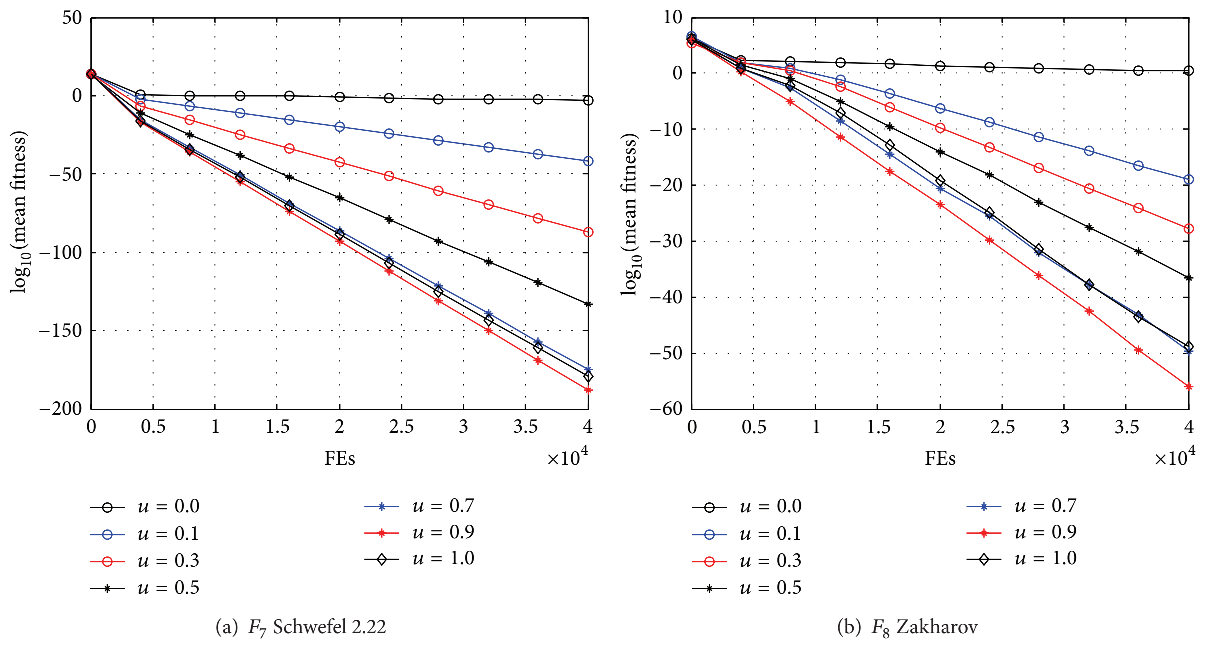
From Table 5 it can be observed that all algorithms hardly converge to the threshold for unimodal functions F_3, F_5, F_6, and F_8 and multimodal functions F_{11}, F_{12}, and F_{14}. BBTLBO converges to the threshold except for functions F_3, F_9, and F_{14}. From the results of total average FEs, BBTLBO converges faster than other algorithms on all unimodal functions and the majority of multimodal functions except for functions F_{15}, F_{16}, F_{19}, and F_{20}. The acceleration rates between BBTLBO and other algorithms are mostly 10 for functions F_1, F_2, F_4, F_7, F_9, F_{10}, and F_{13}. From the results of total average SR, BBTLBO achieves the highest SR for those test functions of which BBTLBO successfully converges to the threshold value. It can be concluded that the BBTLBO has a good performance of convergence speed and successful rate (SR) of the solutions for test functions in this paper.

Table 2: Comparisons mean ± std of the solutions using different u.

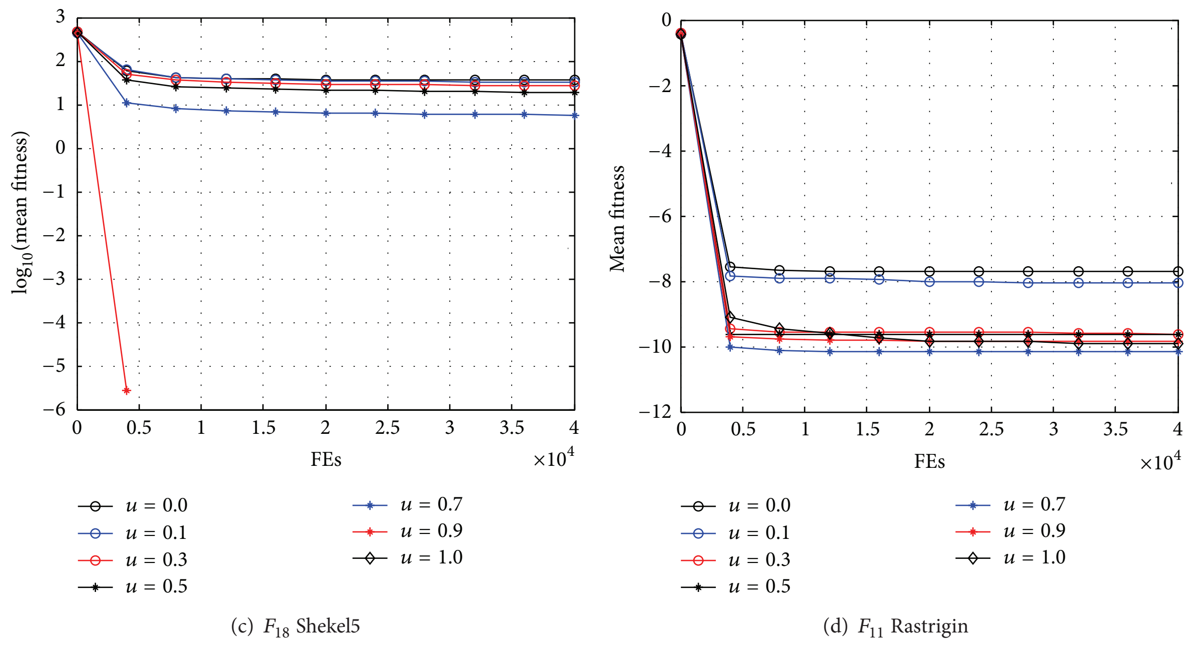
Fun	BBTLBO ($u = 0.0$)	BBTLBO ($u = 0.1$)	BBTLBO ($u = 0.3$)	BBTLBO ($u = 0.5$)	BBTLBO ($u = 0.7$)	BBTLBO ($u = 0.9$)	BBTLBO ($u = 1.0$)
F_1	1.75e − 001 ± 1.21e + 000	6.89e − 071 ± 1.01e − 070	1.23e − 163 ± 00	1.21e − 256 ± 00	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_2	8.98e − 005 ± 5.73e − 004	5.62e − 069 ± 2.72e − 068	2.20e − 161 ± 1.12e − 160	2.43e − 254 ± 00	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_3	1.20e − 001 ± 6.34e − 002	5.91e − 003 ± 1.44e − 003	1.01e − 003 ± 3.48e − 004	4.35e − 004 ± 1.97e − 004	2.35e − 004 ± 1.30e − 004	2.27e − 004 ± 1.26e − 004	**1.99e − 004 ± 1.13e − 004**
F_4	7.65e + 002 ± 5.83e + 002	4.80e − 001 ± 8.86e − 001	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_5	5.58e + 002 ± 6.53e + 002	1.87e − 028 ± 5.73e − 028	3.53e − 054 ± 1.86e − 053	3.69e − 073 ± 2.27e − 072	9.53e − 096 ± 6.74e − 095	**2.16e − 115 ± 1.10e − 114**	2.56e − 100 ± 1.30e − 099
F_6	2.51e + 001 ± 5.34e + 000	6.67e − 021 ± 8.81e − 021	2.81e − 061 ± 6.36e − 061	8.22e − 100 ± 1.80e − 099	8.18e − 137 ± 1.41e − 136	**3.63e − 154 ± 1.34e − 153**	8.86e − 147 ± 3.22e − 146
F_7	1.37e − 003 ± 9.54e − 003	8.72e − 043 ± 1.52e − 042	5.68e − 088 ± 8.76e − 088	1.01e − 133 ± 2.38e − 133	2.60e − 175 ± 00	**1.16e − 188 ± 00**	8.33e − 180 ± 00
F_8	2.41e + 000 ± 3.07e + 000	1.32e − 019 ± 2.98e − 019	2.13e − 028 ± 7.69e − 028	3.44e − 037 ± 1.24e − 036	2.20e − 050 ± 9.12e − 050	**1.07e − 056 ± 4.39e − 056**	2.03e − 049 ± 8.94e − 049
F_9	**2.66e + 001 ± 1.79e + 000**	2.72e + 001 ± 3.17e − 001	2.77e + 001 ± 3.18e − 001	2.83e + 001 ± 2.78e − 001	2.84e + 001 ± 2.67e − 001	2.83e + 001 ± 3.41e − 001	2.80e + 001 ± 3.87e − 001
F_{10}	8.30e + 000 ± 1.76e + 000	1.77e − 001 ± 6.10e − 001	5.90e − 015 ± 1.70e − 015	**3.55e − 015 ± 00**	**3.55e − 015 ± 00**	**3.55e − 015 ± 00**	**3.55e − 015 ± 00**
F_{11}	3.74e + 001 ± 9.05e + 000	3.33e + 001 ± 1.18e + 001	2.71e + 001 ± 8.00e + 000	1.89e + 001 ± 1.14e + 001	5.73e + 000 ± 1.06e + 001	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{12}	8.15e + 000 ± 1.93e + 000	3.38e − 001 ± 1.16e + 000	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{13}	5.06e − 001 ± 8.08e − 001	6.52e − 003 ± 8.86e − 003	1.78e − 003 ± 3.68e − 003	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{14}	**4.33e + 003 ± 6.79e + 002**	4.67e + 003 ± 6.10e + 002	5.17e + 003 ± 6.68e + 002	5.59e + 003 ± 6.85e + 002	5.53e + 003 ± 7.10e + 002	5.58e + 003 ± 7.80e + 002	5.40e + 003 ± 6.53e + 002
F_{15}	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{16}	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{17}	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{18}	−7.71e + 000 ± 3.47e + 000	−8.06e + 000 ± 3.39e + 000	−9.64e + 000 ± 1.81e + 000	−9.65e + 000 ± 1.76e + 000	**−1.02e + 001 ± 6.77e − 003**	−9.85e + 000 ± 1.22e + 000	−9.93e + 000 ± 1.12e + 000
F_{19}	−7.69e + 000 ± 3.52e + 000	−8.13e + 000 ± 3.36e + 000	−9.87e + 000 ± 1.83e + 000	**−1.03e + 001 ± 9.45e − 001**	−9.76e + 000 ± 1.95e + 000	−9.82e + 000 ± 1.78e + 000	−9.61e + 000 ± 1.99e + 000
F_{20}	−8.12e + 000 ± 3.53e + 000	−9.38e + 000 ± 2.69e + 000	−1.01e + 001 ± 1.65e + 000	**−1.01e + 001 ± 1.61e + 000**	−9.70e + 000 ± 2.28e + 000	−9.41e + 000 ± 2.43e + 000	−1.00e + 001 ± 1.69e + 000

Table 3: Comparisons mean ± std of the solutions using different algorithms.

Fun	BBPSO	BBExp	BBDE	GBDE	MGBDE	BBTLBO
F_1	$5.44e - 027 \pm 1.87e - 026$	$2.62e - 024 \pm 5.00e - 024$	$3.90e - 035 \pm 2.00e - 034$	$4.35e - 022 \pm 1.13e - 021$	$3.35e - 035 \pm 2.11e - 034$	$\mathbf{0.0 \pm 0.0}$
F_2	$13800 \pm 2.11e + 004$	$1000 \pm 4.63e + 003$	$6.20e - 021 \pm 4.38e - 020$	$1400 \pm 4.52e + 003$	$1.28e - 032 \pm 8.37e - 032$	$\mathbf{0.0 \pm 0.0}$
F_3	$1.32e + 000 \pm 3.18e + 000$	$2.22e - 002 \pm 7.55e - 003$	$1.64e - 002 \pm 9.57e - 003$	$2.49e - 002 \pm 9.88e - 003$	$1.16e - 002 \pm 5.26e - 003$	$\mathbf{2.27e - 004 \pm 1.26e - 004}$
F_4	$5.60e + 000 \pm 9.28e + 000$	$9.60e - 001 \pm 4.27e + 000$	$7.89e + 001 \pm 3.05e + 002$	$8.40e - 001 \pm 9.12e - 001$	$1.08e + 000 \pm 1.28e + 000$	$\mathbf{0.0 \pm 0.0}$
F_5	$1.24e + 004 \pm 6.66e + 003$	$4.41e + 003 \pm 3.37e + 003$	$2.09e + 000 \pm 4.00e + 000$	$5.36e + 003 \pm 3.26e + 003$	$7.57e + 002 \pm 1.16e + 003$	$\mathbf{2.16e - 115 \pm 1.10e - 114}$
F_6	$1.67e + 001 \pm 9.19e + 000$	$1.20e + 000 \pm 5.22e - 001$	$1.39e + 001 \pm 4.47e + 000$	$3.60e - 001 \pm 1.95e - 001$	$1.10e + 000 \pm 2.94e + 000$	$\mathbf{3.63e - 154 \pm 1.34e - 153}$
F_7	$2.34e + 001 \pm 1.32e + 001$	$1.00e + 000 \pm 3.03e + 000$	$4.06e - 019 \pm 2.15e - 018$	$6.00e - 001 \pm 2.40e + 000$	$2.00e - 001 \pm 1.41e + 000$	$\mathbf{1.16e - 188 \pm 00}$
F_8	$1.87e + 002 \pm 1.34e + 002$	$1.58e + 002 \pm 7.00e + 001$	$1.16e - 001 \pm 2.35e - 001$	$1.72e + 002 \pm 6.67e + 001$	$2.49e + 001 \pm 1.99e + 001$	$\mathbf{1.07e - 056 \pm 4.39e - 056}$
F_9	$7.07e + 001 \pm 1.48e + 002$	$\mathbf{3.57e + 001 \pm 2.50e + 001}$	$2.76e + 001 \pm 1.06e + 001$	$3.17e + 001 \pm 2.07e + 001$	$2.76e + 001 \pm 1.46e + 001$	$2.83e + 001 \pm 3.41e - 001$
F_{10}	$1.06e + 001 \pm 9.29e + 000$	$1.52e + 000 \pm 5.11e + 000$	$1.34e + 000 \pm 1.15e + 000$	$2.59e + 000 \pm 6.45e + 000$	$5.54e - 001 \pm 2.79e + 000$	$\mathbf{3.55e - 015 \pm 00}$
F_{11}	$1.16e + 002 \pm 3.53e + 001$	$1.81e + 001 \pm 7.28e + 000$	$6.76e + 001 \pm 3.89e + 001$	$1.55e + 001 \pm 5.96e + 000$	$2.03e + 001 \pm 9.23e + 000$	$\mathbf{0.0 \pm 0.0}$
F_{12}	$2.73e + 000 \pm 2.11e + 000$	$1.20e - 001 \pm 4.42e - 001$	$1.73e + 000 \pm 1.32e + 000$	$1.21e - 001 \pm 3.37e - 001$	$5.17e - 001 \pm 8.67e - 001$	$\mathbf{0.0 \pm 0.0}$
F_{13}	$2.14e - 002 \pm 4.11e - 002$	$2.30e - 003 \pm 4.29e - 003$	$4.07e - 002 \pm 4.89e - 002$	$3.08e - 003 \pm 7.42e - 003$	$4.63e - 003 \pm 7.16e - 003$	$\mathbf{0.0 \pm 0.0}$
F_{14}	$3.64e + 003 \pm 6.28e + 002$	$2.58e + 003 \pm 5.51e + 002$	$2.30e + 003 \pm 4.09e + 002$	$2.49e + 003 \pm 5.41e + 002$	$2.60e + 003 \pm 5.05e + 002$	$\mathbf{5.58e + 003 \pm 7.80e + 002}$
F_{15}	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$
F_{16}	$4.37e - 003 \pm 3.09e - 002$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$
F_{17}	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$
F_{18}	$-5.60e + 000 \pm 3.41e + 000$	$-7.90e + 000 \pm 2.74e + 000$	$-7.09e + 000 \pm 3.33e + 000$	$-7.63e + 000 \pm 2.86e + 000$	$-8.01e + 000 \pm 3.00e + 000$	$\mathbf{-9.85e + 000 \pm 1.22e + 000}$
F_{19}	$-5.97e + 000 \pm 3.31e + 000$	$-7.87e + 000 \pm 3.03e + 000$	$-6.21e + 000 \pm 3.66e + 000$	$-8.60e + 000 \pm 2.68e + 000$	$-8.37e + 000 \pm 2.90e + 000$	$\mathbf{-9.82e + 000 \pm 1.78e + 000}$
F_{20}	$-5.81e + 000 \pm 3.65e + 000$	$-9.40e + 000 \pm 2.42e + 000$	$-6.02e + 000 \pm 3.77e + 000$	$\mathbf{-9.46e + 000 \pm 2.24e + 000}$	$-9.38e + 000 \pm 2.51e + 000$	$-9.41e + 000 \pm 2.43e + 000$



(a) F_7 Schwefel 2.22 (b) F_8 Zakharov

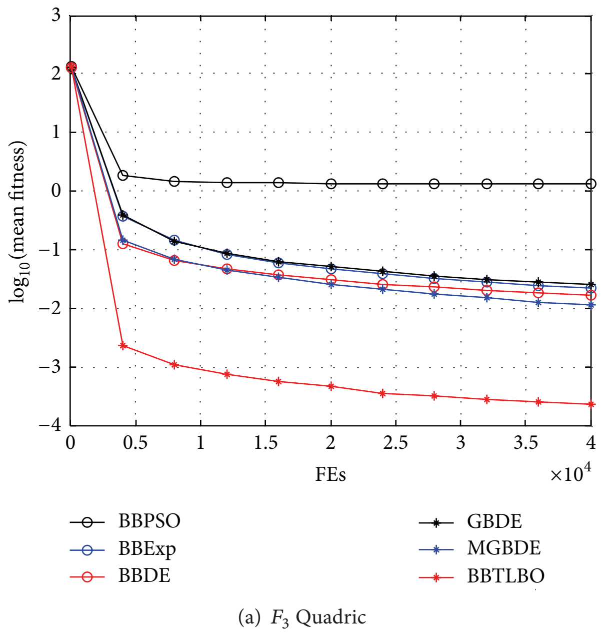


(c) F_{18} Shekel5 (d) F_{11} Rastrigin

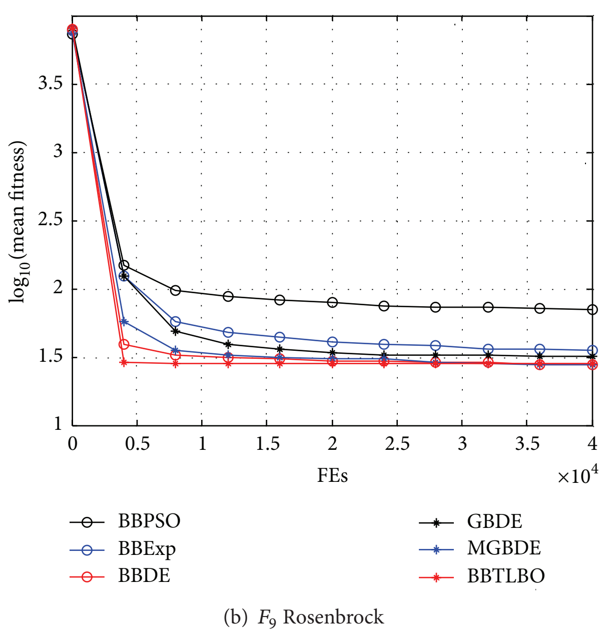
Figure 3: Comparison of the performance curves using different u.

5.5. Comparison of BBTLBO with DE Variants, PSO Variants, and Some TLBO Variants. In this section, we compared the performance of BBTLBO with other optimization algorithms, including jDE [28], SaDE [29], PSOcfLocal [27], PSOwFIPS [30], and TLBO [8, 9]. In our experiment, the maximal FEs are used as the stopping criterion of all algorithms, namely, 40,000 for all test functions. The results are shown in Table 5 in terms of the mean optimum solution and the standard deviation of the solutions obtained in the 50 independent runs by each algorithm on 20 test functions, where "$w/t/l$" summarizes the competition results among BBTLBO and other algorithms. The best results among the algorithms are shown in boldface.

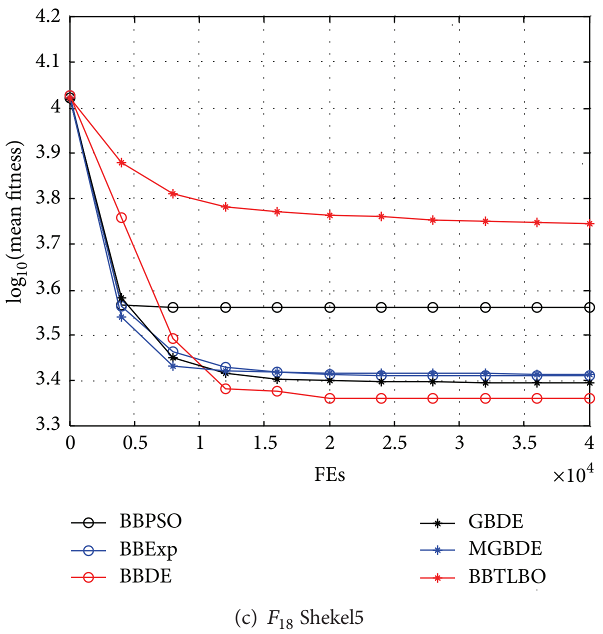
The comparisons in Table 5 show that that all algorithms perform well for F_{15}, F_{16}, and F_{17}. Although SaDE outperforms BBTLBO on F_{14}, PSOcfLocal outperforms BBTLBO on F_9 and PSOwFIPS outperforms BBTLBO on F_{19} and F_{20}, and BBTLBO offers the highest accuracy on functions F_3, F_4, F_5, F_7, F_8, F_{10}, F_{11}, and F_{18}. "$w/t/l$" shows that BBTLBO offers well accuracy for the majority of test functions in this paper.



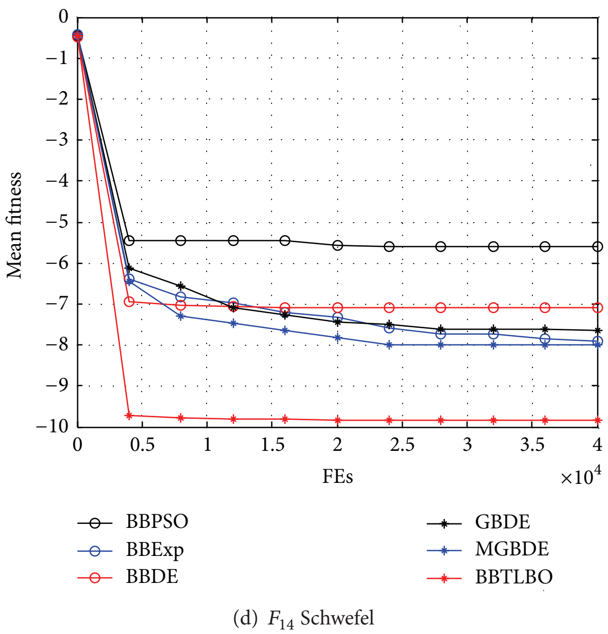
(a) F_3 Quadric



(b) F_9 Rosenbrock



(c) F_{18} Shekel5



(d) F_{14} Schwefel

Figure 4: Comparison of the performance curves using different algorithms.

Table 5 concludes that BBTLBO has a good performance of the solution accuracy for all unimodal optimization problems and most complex multimodal optimization problems.

6. Two Real-World Optimization Problems

In this section, to show the effectiveness of the proposed method, the proposed BBTLBO algorithm is applied to estimate parameters of two real-world problems.

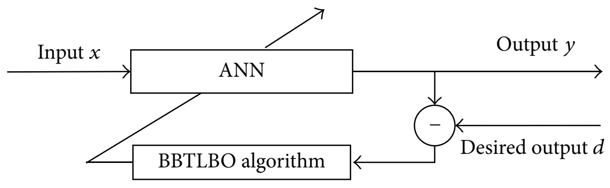


Figure 5: BBTLBO-based ANN.

6.1. Nonlinear Function Approximation. The artificial neural network trained by our BBTLBO algorithm is a three-layer feed-forward network and the basic structure of the proposed scheme is depicted in Figure 5. The inputs are connected to all the hidden units, which in turn all connected to all

Table 4: The mean number of FEs and SR with acceptable solutions using different algorithms.

Fun	t value	BBPSO		BBExp		BBDE		GBDE		MGBDE		BBTLBO	
		MFEs	SR	MFEs	SR	MFEs	SR	MFEs	SR	MFEs	SR	MFEs	SR
F_1	$1E-8$	15922	**100**	17727	**100**	11042	**100**	19214	**100**	11440	**100**	**1390**	**100**
F_2	$1E-8$	17515	54	19179	94	12243	**100**	20592	90	12634	**100**	**1500**	**100**
F_3	$1E-8$	NaN	0	NaN	0	NaN	0	NaN	0	NaN	0	NaN	**0**
F_4	$1E-8$	11710	24	8120	84	3634	6	7343	40	4704	34	**525**	**100**
F_5	$1E-8$	NaN	0	NaN	0	NaN	0	NaN	0	NaN	0	**4100**	**100**
F_6	$1E-8$	NaN	0	NaN	0	NaN	0	NaN	0	NaN	0	**2603**	**100**
F_7	$1E-8$	17540	6	21191	90	17314	**100**	22684	94	15322	98	**2144**	**100**
F_8	$1E-8$	NaN	0	NaN	0	NaN	0	NaN	0	NaN	0	**9286**	**100**
F_9	$1E-2$	17073	62	18404	42	14029	24	18182	52	17200	80	NaN	**0**
F_{10}	$1E-8$	24647	26	27598	90	18273	26	29172	82	18320	84	**2110**	**100**
F_{11}	$1E-8$	NaN	0	NaN	0	NaN	0	NaN	0	NaN	0	**2073**	**100**
F_{12}	$1E-8$	NaN	0	25465	50	NaN	0	27317	64	19704	24	**2471**	**100**
F_{13}	$1E-8$	16318	32	21523	58	11048	16	22951	64	14786	58	**1470**	**100**
F_{14}	$1E-8$	NaN	0	NaN	0	NaN	0	NaN	0	NaN	0	NaN	**0**
F_{15}	$1E-8$	**658**	**100**	1176	100	1274	**100**	1251	**100**	1206	**100**	799	**100**
F_{16}	$1E-8$	**657**	98	1251	**100**	1294	**100**	1343	**100**	1308	**100**	813	**100**
F_{17}	$1E-8$	995	**100**	2626	**100**	1487	**100**	2759	**100**	1921	**100**	**973**	**100**
F_{18}	-10.15	1752	34	6720	44	2007	52	4377	32	8113	64	**1684**	**94**
F_{19}	-10.40	2839	34	8585	48	**1333**	42	6724	50	3056	66	2215	**90**
F_{20}	-10.53	1190	36	8928	74	**1115**	40	6548	76	5441	80	2822	**82**

the outputs. The variables consist of neural network weights and biases. Suppose a three-layer forward neural network architecture with M input units, N hidden units, and K output units, and the number of the variables is shown as follows:

$$L = (M+1) * N + (N+1) * K. \tag{13}$$

For neural network training, the aim is to find a set of weights with the smallest error measure. Here the objective function is the mean sum of squared errors (MSE) over all training patterns which is shown as follows:

$$\text{MSE} = \frac{1}{Q * K} \sum_{i=1}^{Q} \sum_{j}^{K} \frac{1}{2} \left(d_{ij} - y_{ij}\right)^2, \tag{14}$$

where Q is the number of training data set, K is the number of output units, d_{ij} is desired output, and y_{ij} is output inferred from neural network.

In this example, a three-layer feed-forward ANN with one input unit, five hidden units, and one output unit is constructed to model the curve of a nonlinear function which is described by the following equation [31]:

$$y = \sin(2x) \exp(-2x). \tag{15}$$

In this case, activation function used in the output layer is the sigma function and activation function used in the output layer is linear. The number (dimension) of the variables is 16 for BBTLBO-based ANN. In order to train the ANN, 200 pairs of data are chosen from the real model. For each algorithm, 50 runs are performed. The other parameters are the same as those of the previous investigations. The results are shown in Table 6 in terms of the mean MSE and the standard deviation obtained in the 50 independent runs for three methods. Figure 6 shows the predicted time series for training and test using different algorithms. It can conclude that the approximation achieved by BBTLBO has good performance.

6.2. Tuning of PID Controller. The continuous form of a discrete-type PID controller with a small sampling period Δt is described as follows [32]:

$$u[k] = K_P \cdot e(k) + K_I \cdot \sum_{i=1}^{k} e[i] \cdot \Delta t + K_D \cdot \frac{e[k] - e[k-1]}{\Delta t}, \tag{16}$$

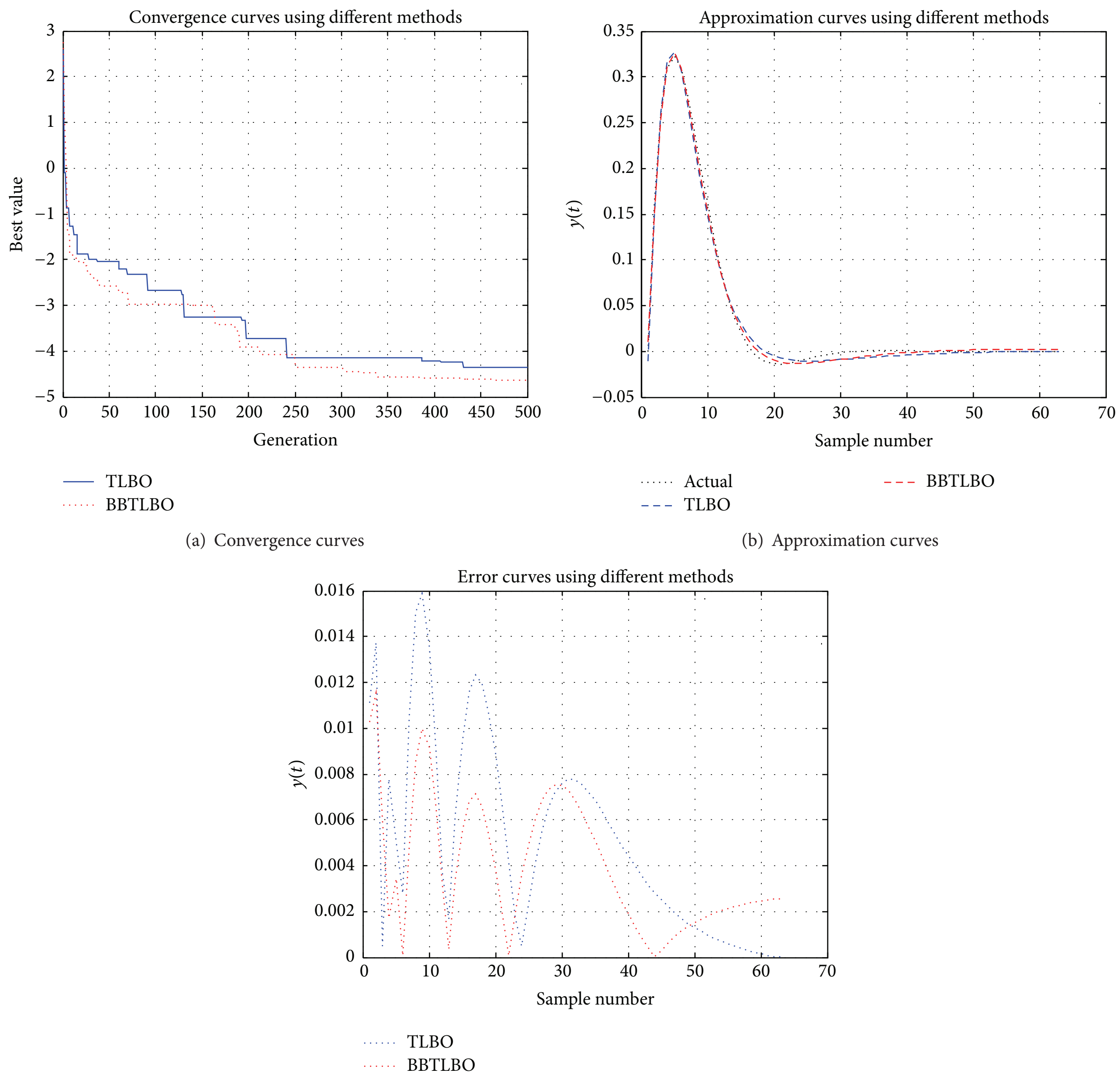
where $u[k]$ is the controlled output, respectively. $e[k] = r[k] - y[k]$ is the error signal, $r[k]$ and $y[k]$ are the reference signal and the system output, and K_P, K_I, and K_D represent the proportional, integral and derivate gains, respectively.

For an unknown plant, the goal of this problem is to minimize the integral absolute error (IAE), which is given as follow [32, 33]:

$$f(t) = \int_0^\infty \left(\omega_1 |e(t)| + \omega_2 u^2(t)\right) dt + \omega_3 t_r, \tag{17}$$

Table 5: Comparisons mean ± std of the solutions using different algorithms.

Fun	jDE	SaDE	PSOcfLocal	PSOwFIPS	TLBO	BBTLBO
F_1	3.63*e* − 025 ± 1.85*e* − 024	7.65*e* − 025 ± 3.34*e* − 024	9.23*e* − 018 ± 3.03*e* − 017	1.01*e* − 002 ± 5.48*e* − 003	3.05*e* − 189 ± 00	**0.0 ± 0.0**
F_2	1.49*e* − 023 ± 6.69*e* − 023	2.75*e* − 025 ± 1.08*e* − 024	3.68*e* − 017 ± 5.37*e* − 017	1.08*e* − 001 ± 5.05*e* − 002	1.29*e* − 185 ± 00	**0.0 ± 0.0**
F_3	3.22*e* − 002 ± 2.83*e* − 002	2.08*e* − 002 ± 1.18*e* − 002	1.28*e* − 002 ± 5.50*e* − 003	1.86*e* − 002 ± 4.39*e* − 003	5.70*e* − 004 ± 2.37*e* − 004	**2.27e − 004 ± 1.26e − 004**
F_4	2.11*e* + 001 ± 6.74*e* + 001	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_5	1.22*e* + 002 ± 1.37*e* + 002	4.28*e* + 001 ± 2.59*e* + 001	1.17*e* + 001 ± 9.30*e* + 000	2.60*e* + 003 ± 6.79*e* + 002	9.45*e* − 043 ± 6.47*e* − 042	**2.16e − 115 ± 1.10e − 114**
F_6	3.06*e* + 001 ± 8.50*e* + 000	2.45*e* + 000 ± 2.60*e* + 000	4.67*e* − 001 ± 2.82*e* − 001	2.66*e* + 000 ± 5.58*e* − 001	2.08*e* − 078 ± 4.30*e* − 078	**3.63e − 154 ± 1.34e − 153**
F_7	8.28*e* − 019 ± 3.49*e* − 018	5.40*e* − 016 ± 3.81*e* − 015	1.34*e* − 011 ± 1.27*e* − 011	1.70*e* − 002 ± 2.85*e* − 003	3.84*e* − 096 ± 5.53*e* − 096	**1.16e − 188 ± 00**
F_8	2.16*e* + 000 ± 4.16*e* + 000	4.88*e* − 001 ± 5.82*e* − 001	9.60*e* − 002 ± 6.99*e* − 002	5.86*e* + 001 ± 1.70*e* + 001	7.09*e* − 022 ± 4.99*e* − 021	**1.07e − 056 ± 4.39e − 056**
F_9	2.49*e* + 001 ± 1.05*e* + 001	2.61*e* + 001 ± 1.07*e* + 000	**2.40e + 001 ± 1.52e + 000**	2.65*e* + 001 ± 3.54*e* − 001	2.55*e* + 001 ± 5.01*e* − 001	2.83*e* + 001 ± 3.41*e* − 001
F_{10}	5.05*e* − 001 ± 7.06*e* − 001	2.07*e* − 001 ± 4.58*e* − 001	1.94*e* − 001 ± 4.56*e* − 001	2.16*e* − 002 ± 4.37*e* − 003	3.62*e* − 015 ± 5.02*e* − 016	**3.55e − 015 ± 00**
F_{11}	2.03*e* + 000 ± 1.94*e* + 000	3.86*e* + 000 ± 1.97*e* + 000	4.26*e* + 001 ± 1.06*e* + 001	1.15*e* + 002 ± 1.54*e* + 001	1.55*e* + 001 ± 8.09*e* + 000	**0.0 ± 0.0**
F_{12}	2.88*e* − 002 ± 1.45*e* − 001	6.50*e* − 002 ± 1.87*e* − 001	7.89*e* − 001 ± 1.03*e* + 000	1.36*e* + 000 ± 7.41*e* − 001	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{13}	1.87*e* − 002 ± 3.58*e* − 002	1.18*e* − 002 ± 1.75*e* − 002	1.16*e* − 002 ± 1.58*e* − 002	1.06*e* − 001 ± 9.93*e* − 002	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{14}	1.93*e* + 002 ± 1.42*e* + 002	**1.35e + 002 ± 1.26e + 002**	4.49*e* + 003 ± 8.25*e* + 002	3.96*e* + 003 ± 8.40*e* + 002	4.82*e* + 003 ± 6.86*e* + 002	5.58*e* + 003 ± 7.80*e* + 002
F_{15}	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{16}	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{17}	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
F_{18}	−9.40*e* + 000 ± 2.10*e* + 000	−9.25*e* + 000 ± 2.30*e* + 000	−7.76*e* + 000 ± 3.42*e* + 000	−9.79*e* + 000 ± 1.44*e* + 000	−9.72*e* + 000 ± 1.42*e* + 000	**−9.85e + 000 ± 1.22e + 000**
F_{19}	−9.85*e* + 000 ± 1.90*e* + 000	−9.87*e* + 000 ± 1.83*e* + 000	−9.24*e* + 000 ± 2.70*e* + 000	**−1.04e + 001 ± 4.23e − 009**	−9.22*e* + 000 ± 2.41*e* + 000	−9.82*e* + 000 ± 1.78*e* + 000
F_{20}	−9.65*e* + 000 ± 2.23*e* + 000	−1.01*e* + 001 ± 1.59*e* + 000	−9.63*e* + 000 ± 2.50*e* + 000	**−1.05e + 001 ± 1.01e − 004**	−9.65*e* + 000 ± 2.23*e* + 000	−9.41*e* + 000 ± 2.43*e* + 000
$w/t/l$	13/3/4	12/4/4	13/4/3	12/4/4	11/6/3	—



(a) Convergence curves
(b) Approximation curves
(c) Error curves

FIGURE 6: Comparison of the performance curves using different algorithms.

TABLE 6: Comparisons between BBTLBO and other algorithms on MSE.

Algorithm	Training error		Testing error	
	Mean	Std	Mean	Std
TLBO	$9.85e-004$	$9.26e-004$	$9.43e-004$	$9.18e-004$
BBTLBO	$3.45e-004$	$2.02e-004$	$2.76e-004$	$1.82e-004$

where $e(t)$ and $u(t)$ are used to represent the system error and the control output at time t, t_r is the rising time, and ω_i ($i = 1, 2, 3$) are weight coefficients.

To avoid overshooting, a penalty value is adopted in the cost function. That is, once overshooting occurs, the value of overshooting is added to the cost function, and the cost function is given as follows [32, 33]:

$$\begin{aligned}
&\text{if } dy(t) < 0\\
&\quad f(t) = \int_0^{\infty} \left(\omega_1 |e(t)| + \omega_2 u^2(t) + \omega_4 |dy(t)|\right) dt + \omega_3 t_r\\
&\text{else}\\
&\quad f(t) = \int_0^{\infty} \left(\omega_1 |e(t)| + \omega_2 u^2(t)\right) dt + \omega_3 t_r\\
&\text{end,}
\end{aligned} \tag{18}$$

TABLE 7: Comparisons of parameters of PID controllers using different algorithms.

Algorithm	K_P	K_I	K_D	Overshoot (%)	Peak time (s)	Rise time (s)	Cost function	CPU time (s)
GA	0.11257	0.02710	0.28792	2.90585	1.65000	1.05000	16.34555	7.05900
PSO	0.11772	0.01756	0.27737	1.04808	1.65000	0.65000	11.60773	6.91000
BBTLBO	0.11605	0.01661	0.25803	0.34261	1.80000	0.70000	11.34300	7.04500

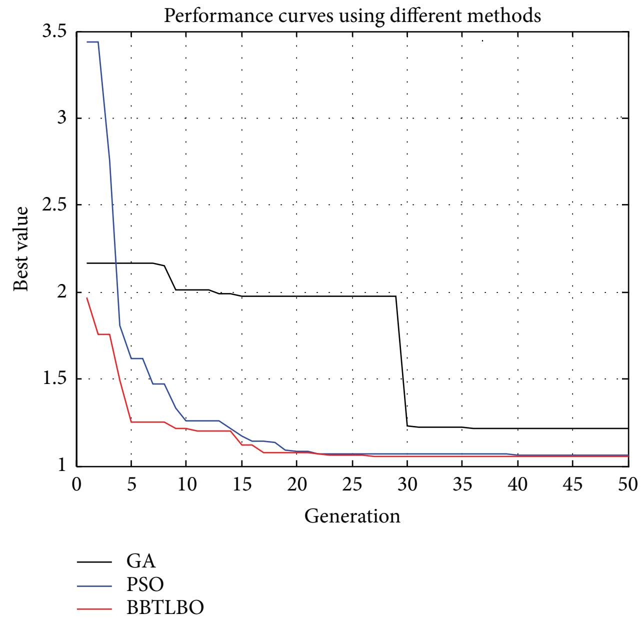


FIGURE 7: Performance curves using different methods.

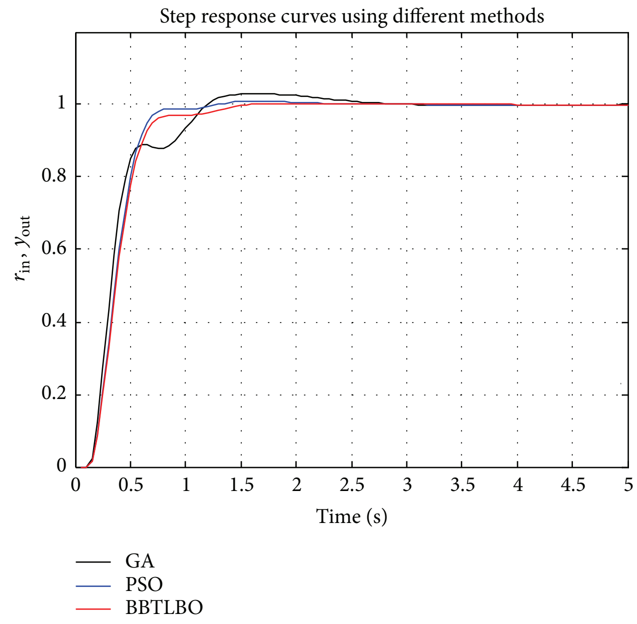


FIGURE 8: Step response curves using different methods.

where ω_4 is a coefficient and $\omega_4 \gg \omega_1$, $dy(t) = y(t) - y(t-1)$, and $y(t)$ is the output of the controlled objective.

In our simulation, the formulas for the plant examined are given as follows [34]:

$$G(s) = \frac{1958}{s^3 + 17.89s^2 + 103.3s + 190.8}. \quad (19)$$

The system sampling time is $\Delta t = 0.05$ second and the control value u is limited in the range of $[-10, 10]$. Other relevant system variables are $K_P \in [0, 1]$, $K_I \in [0, 1]$, and $K_D \in [0, 1]$. The weight coefficients of the cost function are set as $\omega_1 = 0.999$, $\omega_2 = 0.001$ $\omega_3 = 2$, and $\omega = 100$ in this example.

In the simulations, the step response of PID control system tuned by the proposed BBTLBO is compared with that tuned by the standard genetic algorithm (GA) and the standard PSO (PSO). The population sizes of GA, PSO, and BBTLBO are 50, and the corresponding maximum numbers of iterations are 50, 50, and 50, respectively. In addition, the crossover rate is set as 0.90 and the mutation rate is 0.10 for GA.

The optimal parameters and the corresponding performance values of the PID controllers are listed in Table 7 and the corresponding performance curves and step responses curves are given in Figures 7 and 8. It can be seen from Figure 7 and Table 7 that the PID controller tuned by BBTLBO has the minimum cost function and CPU time. Although PID controllers tuned by PSO have a smaller peak time and rise time, their maximum overshoots are much larger than the overshoot tuned by BBTLBO. It concludes that the PID controller tuned by the BBTLBO could perform the best control performance in the simulations.

7. Conclusion

In this paper, TLBO has been extended to BBTLBO which uses the hybridization of the learning strategy in the standard TLBO and Gaussian sampling learning to balance the exploration and the exploitation in teacher phase and uses a modified mutation operation so as to eliminate the duplicate learners in learner phase. The proposed BBTLBO algorithm is utilized to optimize 20 benchmark functions and two real-world optimization problems. From the analysis and experiments, the BBTLBO algorithm significantly improves the performance of the original TLBO, although it needs to spend more CPU time than the standard TLBO algorithm in each generation. From the results compared with other algorithms on the 20 chosen test problems, it can be observed that the BBTLBO algorithm has good performance by using neighborhood search more effectively to generate better quality solutions, although the BBTLBO algorithm does not always have the best performance in all experiments cases of this paper. It can be also observed that the BBTLBO algorithm gives the best performance on two real-world optimization problems compared with other algorithms in the paper.

Further work includes research into neighborhood search based on different topological structures. Moreover, the algorithm may be further applied to constrained, dynamic, and noisy single-objective and multiobjective optimization domain. It is expected that BBTLBO will be used to more real-world optimization problems.

Conflict of Interests

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

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (61100173, 61100009, 61272283, and 61304082). This work is partially supported by the Natural Science Foundation of Anhui Province, China (Grant no. 1308085MF82), and the Doctoral Innovation Foundation of Xi'an University of Technology (207-002J1305).

References

[1] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[2] L. C. Jiao and L. Wang, "A novel genetic algorithm based on immunity," *IEEE Transactions on Systems, Man, and Cybernetics A: Systems and Humans*, vol. 30, no. 5, pp. 552–561, 2000.

[3] 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.

[4] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MIT Press, 2004.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[6] 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.

[7] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[8] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems," *CAD Computer Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.

[9] 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.

[10] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization algorithm for unconstrained and constrained real-parameter optimization problems," *Engineering Optimization*, vol. 44, no. 12, pp. 1447–1462, 2011.

[11] V. Toğan, "Design of planar steel frames using teaching-learning based optimization," *Engineering Structures*, vol. 34, pp. 225–232, 2012.

[12] 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.

[13] S. O. Degertekin and M. S. Hayalioglu, "Sizing truss structures using teaching-learning-based optimization," *Computers and Structures*, vol. 119, pp. 177–188, 2013.

[14] 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.

[15] R. V. Rao and V. Patel, "Multi-objective optimization of combined Brayton and inverse Brayton cycles using advanced optimization algorithms," *Engineering Optimization*, vol. 44, no. 8, pp. 965–983, 2011.

[16] T. Niknam, F. Golestaneh, and M. S. Sadeghi, "Theta-multi-objective teaching-learning-based optimization for dynamic economic emission dispatch," *IEEE Systems Journal*, vol. 6, no. 2, pp. 341–352, 2012.

[17] 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.

[18] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[19] F. van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information Sciences*, vol. 176, no. 8, pp. 937–971, 2006.

[20] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the Swarm Intelligence Symposium (SIS '03)*, pp. 80–87, 2003.

[21] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *European Journal of Operational Research*, vol. 196, no. 1, pp. 128–139, 2009.

[22] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 634–647, 2013.

[23] X. H. Hu and R. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 677–1681, 2002.

[24] M. G. Omran, A. P. Engelbrecht, and A. Salman, "Using the ring neighborhood topology with self-adaptive differential evolution," in *Advances in Natural Computation*, pp. 976–979, Springer, Berlin, Germany, 2006.

[25] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 1, pp. 150–169, 2010.

[26] I. Maruta, T. H. Kim, D. Song, and T. Sugie, "Synthesis of fixed-structure robust controllers using a constrained particle swarm optimizer with cyclic neighborhood topology," *Expert Systems with Applications*, vol. 40, no. 9, pp. 3595–3605, 2013.

[27] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the International Conference on Evolutionary Computation*, pp. 1671–1676, Honolulu, Hawaii, USA, 2002.

[28] 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.

[29] 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.

[30] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[31] F. Herrera and M. Lozano, "Gradual distributed real-coded genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 1, pp. 43–62, 2000.

[32] J. Liu, *Advanced PID Control and MATLAB Simulation*, Electronic Industry Press, 2003.

[33] J. Zhang, J. Zhuang, H. Du, and S. Wang, "Self-organizing genetic algorithm based tuning of PID controllers," *Information Sciences*, vol. 179, no. 7, pp. 1007–1017, 2009.

[34] R. Haber-Haber, R. Haber, M. Schmittdiel, and R. M. del Toro, "A classic solution for the control of a high-performance drilling process," *International Journal of Machine Tools and Manufacture*, vol. 47, no. 15, pp. 2290–2297, 2007.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 164053, 9 pages
http://dx.doi.org/10.1155/2014/164053

Research Article

Minimum Variance Distortionless Response Beamformer with Enhanced Nulling Level Control via Dynamic Mutated Artificial Immune System

Tiong Sieh Kiong,[1] S. Balasem Salem,[2] Johnny Koh Siaw Paw,[2] K. Prajindra Sankar,[2] and Soodabeh Darzi[3]

[1] *Power Engineering Center, College of Engineering, Universiti Tenaga Nasional, Kajang, Selangor, Malaysia*
[2] *Center of System and Machine Intelligence, College of Engineering, Universiti Tenaga Nasional, Kajang, Selangor, Malaysia*
[3] *Department of Electrical, Electronic & Systems Engineering, Universiti Kebangsaan Malaysia, Bangi, Selangor, Malaysia*

Correspondence should be addressed to Soodabeh Darzi; sdtutn@yahoo.com

Received 18 February 2014; Accepted 23 March 2014; Published 5 June 2014

Academic Editors: V. Bhatnagar and Y. Zhang

Copyright © 2014 Tiong Sieh Kiong 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 smart antenna applications, the adaptive beamforming technique is used to cancel interfering signals (placing nulls) and produce or steer a strong beam toward the target signal according to the calculated weight vectors. Minimum variance distortionless response (MVDR) beamforming is capable of determining the weight vectors for beam steering; however, its nulling level on the interference sources remains unsatisfactory. Beamforming can be considered as an optimization problem, such that optimal weight vector should be obtained through computation. Hence, in this paper, a new dynamic mutated artificial immune system (DM-AIS) is proposed to enhance MVDR beamforming for controlling the null steering of interference and increase the signal to interference noise ratio (SINR) for wanted signals.

1. Introduction

The evolution of adaptive beamforming was initiated in military and aerospace applications through the employment of electronically steered antennas based on phased-array technology. Typical applications include long-range surveillance radar, active jammer rejection, and multibeam antennas for space communications [1]. The same antenna array techniques were then assumed suitable for mobile radio communication to solve multipath fading and the cochannel interference problem. A partially adaptive antenna array technology, known as the intermediate frequency side lobe canceller (SLC), was invented by Howells in the late 1950s [2]. This technology incorporates the capability of automatic interference nulling. However, SLC was not fully adaptive because the main beam has a fixed pattern, and the auxiliary array contains only a few controlled elements. This simple adaptive antenna facilitated the development of fully adaptive array by Howells' coworker, Applebaum, in 1965. The algorithm, commonly known as the Howells-Applebaum algorithm, was developed to maximize the signal-to-noise ratio (SNR) at the output of the beamformer. At the same time, in 1960, another independent research group led by Windrow invented another adaptive array approach based on linear covariance minimum variance (LCMV) [3]. This algorithm, later known as the Windrow-Hoff LMS algorithm, was developed based on the minimum mean square error (MMSE) criterion for the automatic adjustment of array weights. This algorithm is well known for its simplicity but only achieves satisfactory performance under specific operational conditions. The major drawback of this algorithm is its low convergence rate, which refers to the speed by which the mean of the estimated weights approaches the optimal value, thereby making it unsuitable for certain applications.

In 1969, Capon introduced another adaptive beamformer known as minimum variance distortionless response (MVDR), which is a technique capable of resolving signals separated by a fraction of antenna beam width. This optimum beamformer, also known as the LCMV beamformer, requires only the knowledge of the desired signal direction of arrival (DOA) to maximize the SNR. Another important contribution was by Reed, Mallett, and Brennen in 1974. They introduced a fast convergence algorithm known as the sample matrix inversion (SMI) technique, which overcame the problem of slow convergence faced by the LMS algorithm. One of the most important factors in smart antenna processes is beamforming, which refers to the allocation of signals in particular positions and phase angles for each antenna for the corresponding angle of the system [4].

Beamforming technology is a key technique in nulling antenna. By performing some processes with the received array signals, such as weighting and summation, beamforming can help the antenna realize many advanced functions, such as beam shaping, beam scanning, and beam nulling [5]. Reception beamforming is independently achieved at each receiver; however, the transmitter in transmit beamforming has to consider all receivers to optimize the beamformer output [6, 7].

One of the beamforming algorithm used in smart antenna is MVDR beamforming, which can calculate the weight vector to determine the desired signal from the interference. Moreover, MDVR maximizes the sensitivity in one direction only [8]. The MVDR beamformer, also known as Capon beamformer, minimizes the output power of the beamformer under a single linear constraint on the response of the array toward the desired signal. The idea of combining multiple inputs in a statistically optimum manner under the constraint of no signal distortion can be attributed to Darlington. Several researchers developed beamformers, in which additional linear constraints are imposed (e.g., Er and Cantoni) [9, 10].

Many approaches proposed the use of a mathematical model to improve the robustness of the MVDR beamformer, as presented in [11, 12]. Research on the artificial immune system (AIS) and its application has become increasingly important in the field of intelligent information systems [13–15]. A new optimization technique was presented for the design of linear antenna arrays. The proposed technique was based on a novel variant of particle swarm optimization (PSO) called Boolean PSO with adaptive velocity mutation. The antenna arrays were optimized based on the requirements for maximizing the power gain at a desired direction and minimizing the side lobe level of the radiation pattern [16]. A complex-valued genetic algorithm for the optimization of beamforming in linear array antennas was proposed. The algorithm was proven to enhance searching efficiency significantly while effectively avoiding premature convergence. Numerical results were presented to illustrate the advantages of the proposed technique over conventional pattern synthesis methods [17].

In this paper, the main goal is to design a beamforming method based on MVDR in corporation with new dynamic mutated artificial immune system (DM-AIS) algorithm in order to enhance the null level at interference sources. By using this method, finding a new mathematical model, changing the filter hardware for signal processing, or changing the design of antenna based on the increased number of elements is no longer necessary. Another reason for proposing the new method is the difficulty in obtaining the optimum value using any normal algorithm without intensification. In this investigation, DM-AIS have been applied in beamforming with uniform linear antenna arrays of 0.5 λ spacing between adjacent elements and radiating at a frequency of 2.3 GHz. The rest of this paper is organized as follows. Section 2 introduces the basics of adaptive beamforming. Sections 3 and 4 summarized a basis to describe the conventional MVDR beamforming and AIS, respectively. Section 5 shows the incorporation of MVDR with DM-AIS. Simulation results of one user with two interferences and comparison of conventional MVDR with mp-QP MVDR and DM-AIS are reported in Section 6. And finally Section 7 concludes this investigation.

2. Background of Adaptive Beamforming

Adaptive beamforming is a technique for receiving a signal of interest (SOI) from specific directions while suppressing the interfering signals adaptively in other directions using an array of sensors. This technique can automatically optimize the array pattern by adjusting the elemental control weights until a prescribed objective function is satisfied. This technique provides a means for separating a desired signal from interfering signals.

Beamforming has numerous applications in radar, sonar, seismology, microphone array speech processing, and, more recently, wireless communications. In particular, the use of antenna arrays, in combination with signal processing algorithms at the base station, offers the possibility of exploiting the spatial dimension to separate multiple cochannel users. This approach provided increased channel capacity and wider area coverage. Array beamforming methods in such systems use the spatial dimension to combat interference, noise, and multipath fading of the desired signal [11]. The outputs of the individual sensors were linearly combined after being scaled with the corresponding weights. This process optimizes the antenna array to achieve maximum gain in the direction of the desired signal and nulls in the direction of interferers. For a beamformer, the output at any time n, $y(n)$ is given by a linear combination of the data at M antennas, with $x(n)$ being the input vector and $w(n)$ being the weight vector, as shown in Figure 1:

$$y(n) = w^{H}(n)^{*}(n). \tag{1}$$

Weight vector $W(n)$ can be defined as follows:

$$\begin{aligned} w(n) &= \sum_{N=0}^{M-1} w_n, \\ x(n) &= \sum_{n=0}^{M-1} X_n. \end{aligned} \tag{2}$$

For any algorithm that evades the matrix inverse operation and uses the immediate gradient vector $\nabla J(n)$ for weight

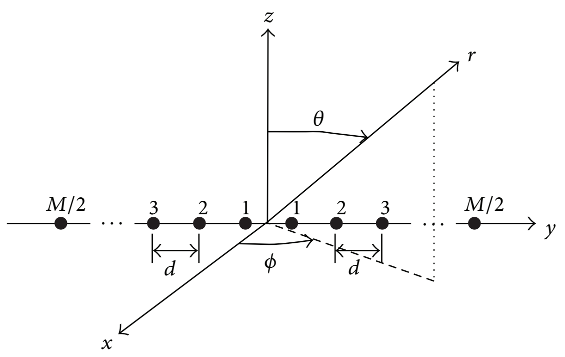


Figure 1: Linear array with elements along the y-axis.

vector upgrading, the weight vector at time $n + 1$ can be written as follows:

$$W(n+1) = W(n) + \frac{1}{2}\mu [\nabla J(n)], \tag{3}$$

where μ is the step size parameter, which controls the speed of convergence and lies between 0 and 1. The minimum values of μ facilitate the sluggish concurrence and high-quality estimation of the cost function. Comparatively, the huge values of μ may direct to a rapid union. However, the constancy over the least value may disappear:

$$0 < \mu < \frac{1}{\lambda}. \tag{4}$$

An exact calculation of instantaneous gradient vector $\nabla J(n)$ is not possible because prior information on covariance matrix R and cross-correlation vector p is needed. Thus, an instantaneous estimate of gradient vector is given by

$$\begin{aligned} \nabla J(n) &= -2p(n) + 2R(n)W(n) \\ R(n) &= X(n)X^H(n), \\ P(n) &= d(n)^* X(n). \end{aligned} \tag{5}$$

By substituting values from (5) into (3), the weight vector is derived as follows:

$$\begin{aligned} W(n+1) &= W(n) + \mu [p(n) - R(n)W(n)] \\ &= W(n) + \mu X(n) \lfloor d^*(n) - X(n)W(n) \rfloor \\ &= W(n) + \mu X e^*(n). \end{aligned} \tag{6}$$

The desired signal can be defined by the following three equations:

$$\begin{aligned} y(n) &= w^H(n)x(n) \\ e(n) &= d(n) \cdot y(n)W(n+1) \\ &= W(n) + \mu X(n)e^*(n). \end{aligned} \tag{7}$$

Numerous algorithms were introduced for the design of an adaptive beamformer [8]. One of the most popular approaches for adaptive beamforming was proposed by Capon [4]. His algorithm leads to an adaptive beamformer with an MVDR. Some constraints, such as the antenna gain being constant in the desired direction, are used to ensure that the desired signals are not filtered out along with the unwanted signals. The MVDR beamformer not only minimizes the array output power but also maintains a distortionless main lobe response toward the desired signal. However, the MVDR beamformer may have an unacceptably low nulling level, which may significantly degrade the performance in the case of unexpected interfering signals. In particular, the performance of MVDR degrades in rapidly moving jammer environments. This degradation occurs because of jammer motion, which may bring jammers out of the sharp notches of the adapted pattern. To achieve high interference suppression and SOI enhancement, an adaptive array must introduce deep and widened nulls in the DOAs of strong interferences while keeping the desired signal distortionless. Thus, the issue of nulling level control is especially important for both deterministic and adaptive arrays [18].

3. Conventional MVDR Beamforming

When a beamformer has a constant response in the direction of a useful signal, the LCMV algorithm becomes an MVDR algorithm [19]. The MVDR algorithm is capable of suppressing the interference, but with high value in SNR and low noise. At the same time, the MVDR algorithm depends on the steering vectors, which in turn depend on the incident angle of the received signal from the element of the array antenna. The direction of useful signal must be known and the output power subject to a unity gain constraint in the direction of desired signal must be minimized. The array output is given by

$$y = w^H x. \tag{8}$$

The output power is as follows:

$$p = \{E|y|^2\} = E\{w^H x x^H w\} = w^H E\{x x^H w\} = w^H R, \tag{9}$$

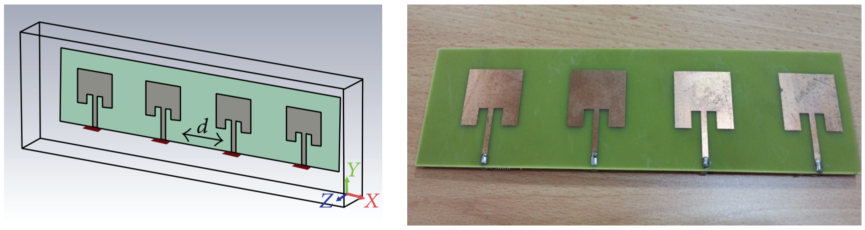
where the R covariance matrix should be $(M, 1)$ for the received signal x and H is the hermitian transpose.

The optimum weights are selected to minimize the array output power P_{MVDR} while maintaining unity gain in the look direction $a(\theta)$, which is the steering vector of the desired signal. The MVDR adaptive algorithm can be written as follows:

$$\min_w \{w^H R w\} \quad \text{subject to } w^H a(\theta) = 1. \tag{10}$$

The steering vector $a(\theta)$ is given by

$$a(\theta) = \begin{bmatrix} 1 \\ \exp\left\{j\frac{2\pi}{\lambda}(\sin\theta_i)d\right\} \\ \\ \exp\left\{j\frac{2\pi}{\lambda}(\sin\theta_i)(m-1)d\right\} \end{bmatrix}, \tag{11}$$



(a) CST program (b) Fabricated

Figure 2: (a, b) Four elements of the linear array.

where d is the space between the elements of the antenna, θ_i is the desired angle, and m is the number of elements, as shown in Figure 2.

The optimization weight vectors can then be acquired using the following formula [20]:

$$W_{\mathrm{MVDR}} = \frac{R^{-1}a(\theta)}{a^H(\theta)R^{-1}a(\theta)}. \tag{12}$$

These weights are the solution of the optimization problem mentioned at (10) and with the use of four elements of array antenna, four weights as below will be obtained:

$$w_{\mathrm{MVDR}} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}. \tag{13}$$

Subsequently, the beamformer weights are selected based on minimum mean value of output power according to the number of users inside the coverage area while maintaining unity response in the desired direction. Nevertheless, the restraint ensures that the signal passes through the beamformer undistorted. Consequently, the output signal power is similar to the look direction source power. The total noise, including interferences and uncorrelated noise, is then reduced by the minimization process. Notably, the minimization of the total output noise, while constantly maintaining the output signal, is the same as maximizing the output SINR. However, for the optimal beamformer to perform as described above and to maximize the SINR by cancelling interferences, the number of interferences must be less than or equal to $M-2$ because an array with M elements has $M-1$ degrees of freedom and has been utilized by the constraint in the look direction. Given that the MVDR beamformer maximizes sensitivity in one direction only, this beamformer is unsuitable for multipath environments, where the desired signal spreads in all directions [21]. The multipath occurs in non-line-of-sight environments such as populated urban areas, where numeorus scatterers are close to the users and the base station. Thus, the MVDR beamformer may have an unacceptably low nulling level, which may significantly degrade performance in the case of unexpected interfering signals. As a result, the beamforming optimization problem is formulated as a multiparametric quadratic programming (mp-QP) [18].

4. Artificial Immune System

The proposed usage of artificial intelligent system as the enhancement method for the adaptive beamforming technique is based on a framework that is built around the concept of reactive artificial immune system (AIS). AIS, which is inspired by theoretical immunology and observed immune functions, is a branch of the metaheuristic algorithm with promising results in the field of optimization. While AIS resembles some other metaheuristics algorithms such as genetic algorithm (GA), except the recombination operator, the former has code simplicity and is low in computational cost. This AIS system is indeed based upon the normal human immune system in the way that it is reactive towards foreign elements. The immune system is highly robust, adaptive, inherently parallel, and self-organized. It has powerful learning and memory capabilities and presents an evolutionary type of response to infectious foreign elements [22, 23].

The main agents of the adaptive immune system are lymphocytes that are are called the B cells which produce antibodies to attack the enemy. Some B cells become "memory cells" which keep molecular records of past invader and minimize the body's response time to an infection. The clonal selection principle is the whole process of antigen recognition, cell proliferation, and differentiation into a memory cell.

The clonal-selection theory proposes that as an antigen enters the immune system certain B cells are selected based on their reaction to this antigen to undergo rapid cloning and expansion. This reaction is often termed the affinity of that B cell (or antibody) for the given antigen. Those B cells are selected, based on their affinity to an antigen, to produce a number of clones to attack or neutralize the invading antigen. Cells that have a higher degree of affinity are allowed to produce more clones. The clonal production develops immune cells that are more adept at recognizing and reacting to the antigen through mutation. B cell offsprings undergo mutation based on an inverse proportionality to their affinity values. Through this process, the affinity of subsequent generations of B cells will have greater reaction to the antigen, and more diversity will also have been added to the system through the wider exploration afforded by the high mutation rates of the cells with lower affinity measures.

The process of a standard clonal selection algorithm can be summarized as follows [24, 25].

(1) Generate a random initial population of antibody Ab, given by

$$P(0) := \{Ab_1(0), \ldots, Ab_n(0)\}. \tag{14}$$

(2) Compute the fitness of each Ab

$$P(0) : \{f(x_1(0), \ldots, x_n(0))\}. \tag{15}$$

(3) Generate clones by cloning all cells in the Ab population. The amount of clone is given by

$$N_C = \sum_{i=1}^{n} \text{round}\left(\frac{\beta \bullet N}{i}\right), \tag{16}$$

where N_C is the total clones generated for each Ab, β is the multiplying factor, and N is the number of Ab.

(4) Mutate the clone population to produce a mature clone population with δ number of children. The new Ab is composed of

$$P' := \{Ab'_1, \ldots, Ab'_\delta\}. \tag{17}$$

(5) Evaluating affinity values of the clones' population is

$$P' : \{f(x'_1, \ldots, x'_\delta)\}. \tag{18}$$

The next generation of Ab is obtained using $G = \varepsilon + \delta$ selection by choosing the best ε individuals out of the G population.

(6) Select the best Ab to compose the new Ab population by

$$P_{\text{new}} := s_G P'. \tag{19}$$

(7) Steps 3 to 6 are repeated until a predefined stopping condition is reached.

5. MVDR Beamforming Incorporation with New Dynamic Mutated Artificial Immune System

In this paper, AIS with dynamic mutation (DM) was utilized to enhance the null level of the MVDR beamforming technique. In analogy, the adaptive antenna represents the body of an organism, whereas the interference and noise signal sources represent external harmful attacks toward the organism. Hence, the adaptive antenna system will organize its antibody to protect the body of organism from external antigen attacks. The adaptive antenna system will try to optimize through its AIS iteration process to develop deep null at the DOA of the interference sources to achieve the maximum SINR.

In this AIS algorithm, the weight vector w will be generated as the system antibody. The algorithm will initiate by generating a population of N antibodies, which is represented by weight vectors W_N. The number of generated weight vectors depends on the population size P_{size}. For the first iteration, the first set of weight vectors W_1 is obtained from the computation of the conventional MVDR weight vector. The weight vectors in every antibody contain an M number of weight vectors, depending on the number of sensors or antenna elements used, and can be expressed as follows:

$$\begin{aligned}
w_{1M} &= \text{Real}\{w_{\text{mvdr}}\}_M + \text{Imag}\{w_{\text{mvdr}}\}_M \\
w_{2M} &= \text{Real}\{w_{1M}{}^{*}\text{rand}(M, P_{\text{Size}} - 1)\} \\
&\quad + \text{Imag}\{w_{1M}{}^{*}\text{rand}(M, P_{\text{Size}} - 1)\} \\
&\ \vdots \\
w_{NM} &= \text{Real}\{w_{(N-1)M}{}^{*}\text{rand}(M, P_{\text{Size}} - 1)\} \\
&\quad + \text{Imag}\{w_{(N-1)M}{}^{*}\ \text{rand}(M, P_{\text{Size}} - 1)\}.
\end{aligned} \tag{20}$$

In matrix format, the weight vectors in the population of any iteration can be represented by

$$W_{NM} = \begin{bmatrix} w_{\text{mvdr1}} & w_{\text{mvdr2}} & w_{\text{mvdr3}} & w_{\text{mvdr4}} \\ w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ w_{n1} & w_{n2} & w_{n3} & w_{n4} \end{bmatrix}, \tag{21}$$

where

W_{NM} is weight vectors of total population N with M sensors in each antenna;

w_{mvdr} is weight vectors from MVDR beamformer;

M is number of sensor. In this study, antenna sensor of (4, 1) is used;

P_{size} is population size;

M is number of sensor.

Each set of antibodies W has amplitude and phase ($A\forall\theta$) to steer the radiation beam toward its target user and place the deep null toward the interference sources to achieve the optimum SINR. The best weight vector is determined according to the fitness value obtained from fitness function as shown below:

$$\text{Fitness\_Function (FF)} = \frac{P_{\text{User}}}{\sum_{n=1}^{N} P_{\text{Inter\_}n} + \text{Noise}}, \tag{22}$$

where P_{User} = power of target user, P_{Inter} = power of interference, and N = number of interference sources.

The new candidate population of antibodies (weight vectors) based on the mutation rate depends on the fitness value of w in (22), which is given by

$$M_{\text{rate}} = e^{-(\sqrt[5]{(Fw_{\text{best}})^2})}, \tag{23}$$

where M_{rate} = mutation rate and Fw_{best} = fitness of best population weight.

The mutation rate of clones is inversely proportional to their antigenic affinity [24, 26]. A higher affinity denotes a smaller mutation rate.

The mutations and clones needed to create the new weight are given by

$$W = w_{\text{best}} + M_{\text{rate}}{}^{*}\left(\text{Real}\left\{\text{rand}\left(M, P_{\text{Size}} - 1\right)\right\} - \text{Imag}\left\{\text{rand}\left(M, P_{\text{Size}} - 1\right)\right\}\right). \tag{24}$$

The n antibodies generate Nc clones proportional to their affinities. The number of cloned antibodies Nc can be computed by

$$N_C = \sum_{i=1}^{n} \text{round}\frac{B \cdot w}{i}, \tag{25}$$

where B is a multiplying factor with a value of 1 and w is the weight vector.

Each term of this sum corresponds to the clone size of each selected antibody. The clones are then subjected to hyper mutation and receptor editing. To enable the algorithm to find the best solution, a deep null must be achieved to obtain the maximum SINR. Therefore, the DM was introduced to identify the best solution. The selection of the 10 best solutions depends on the maximum SINR value. Thus, from these 10 best solutions, DM-AIS can select three random values from the 10 best weights to achieve the SINR size. When the scaling factor is (0, 1, 2), the best value of the deep null is derived:

$$w_{DM} = w_{1M} + \text{sf}^{*}\left(w_{2M} - w_{3M}\right), \tag{26}$$

where W_1, W_2, W_3 = random weight select from the best three solutions and sf = scaling factor from (0, 1, 2).

The next step is the clonal operation to obtain the best solution.

Affinity is applied to achieve the maximum power for target and deep null for interference. If this value is the best, it will store and yield the final weight value to stop the calculation. The DM-AIS steps in an adaptive antenna are as follows.

(1) A random initial population of W is generated, which is given by

$$W_N(i) := \{W_1(i), \ldots, W_n(i)\}. \tag{27}$$

(2) The fitness of each W is computed:

$$P(i) : \{f(w_1(i), \ldots, w_n(i))\}. \tag{28}$$

(3) Clones are generated by cloning all the cells in the W population. The amount of clone is obtained using (23).

(4) The clone population is mutated to produce a mature clone population with i number of weight. The rate of mutation is given by

$$M_{\text{rate}} = e^{-\left(\sqrt[5]{(Fw_{\text{best}})^2}\right)}. \tag{29}$$

Thus, the new W is composed of $W'_N(i) = \{W'_1, \ldots, W'_i\}$.

(5) The affinity values of the clone population are expressed as follows:

$$W'_N(i) : \left\{f\left(W'_1, \ldots, W'_i\right)\right\}. \tag{30}$$

The next generation of W is obtained using (20). The best W_{best} is selected to compose the new W_{new} population as follows:

$$W_{\text{new}} = W_{\text{best}}\ W'_N(i). \tag{31}$$

(6) The 10 best weight vectors are selected depending on the selection assumption (P_{size}) to identify the best number of the weight.

(7) The vectors are mutated by selecting three random weights from the 10 best weights using (26).

(8) Steps 3 to 6 are repeated until a predefined stopping condition is reached.

Remarks. Similar to weight vector optimization techniques based on the support vector regression (SVR-AS) [27] and mp-QP MVDR [18], the proposed DM-AIS MVDR technique is also effective in finding the optimal weight vector for an antenna array under specific conditions. The differences among the methodologies are as follows.

(1) The proposed DM-AIS MVDR determines the optimal weight vector of an antenna array through its cloning and mutation process to fine-tune the weight vector toward a radiation pattern that can generate deep null toward the interference source, while maintaining high gain at the target users' directions for an instantaneous scenario.

(2) The mp-AP MVDR produces the optimal weight vector of an antenna array for the optimal problem of the receiving beam based on a multiparametric approach [18].

(3) The SVR-AS requires the radiation pattern as the train data to find the optimal weight vector of an antenna array [27].

The benefits of the proposed approach are as follows.

(1) The nulling extent can be deepened.

(2) This approach can guarantee that the nulling levels in the specified areas are better than conventional MVDR.

(3) This approach does not require pretrain data and does not require complex mathematical computation to identify the optimal weight vector. Instead, merely the DOAs of the target user and interference sources are required for DM-AIS MVDR to determine its optimal weight vector for best radiation beamforming.

6. Simulation and Results

6.1. One User with Two Interferences. In this section, simulations were conducted to validate the proposed approach.

Table 1: Parameters of DM-AIS.

Dynamic mutated parameters	Value
Number of population	20
Allocated number of best population	10
Clone size	4
Number of max. iteration	20
Scaling factor	(0, 1, 2)

Table 2: Weight vector after 10 and 20 iterations for one user at 40° and interference at 50° and 30°.

Sensor number	Weight	10 iterations	20 iterations
1	W_1	$-4.5333 - 0.6902j$	$-4.6171 - 0.7212j$
2	W_2	$-2.4985 + 3.8447j$	$-2.5978 + 3.9071j$
3	W_3	$-1.3543 - 1.9258j$	$-1.3966 - 1.9127j$
4	W_4	$-3.7544 + 0.6641j$	$-3.7899 + 0.6865j$

The uniform linear array (ULA) consists of four elements ($M = 4$) equispaced by half-wavelength. The desired signal and two interference signals are plane waves impinging on the ULA from the directions 50°, 30°, and 40°, respectively. In this simulation, the SNR from 5 dB to 30 dB was used for the desired signal and the two interference signals. The beam pattern nulling level should be below −70 dB. The complex vector of beamformer weights calculated by the aforementioned (27)–(31) is presented in Table 2, whereas the beam patterns generated are plotted in Figure 3. All the beam patterns have nulls at the DOAs of the interference signals and maintain a distortionless response for the SOI. However, DM-AIS place deep nulls (with nulling level equal to −80 dB) at the DOAs of two interference signal sources. The MVDR after 10 iterations reduces nulling levels compared with the MVDR after 20 iterations (Table 1).

Figure 3 gives the ratio of SINR 67.9479 dB in 20 iterations while at 10 iterations the SINR is 40.5387 dB of the aforementioned beamformer for the previous scenario. It can be seen that the DM-AIS at 20 iterations shows better ratio which is 122.49% of improvement. If we compare the improvement in SINR between normal AIs and DM-AIS for the same scenario in 10 iterations we obtain the percentage 131.48%.

But an insignificant increase in the SINR of 0.2 is obtained for an increase in the number of iterations. The iterations were taken from 30 till 50, as shown in Figure 4. This implies that there is no need to increase the time required for simulations. The 20 iterations will give the best results by the short time. These results should be saved. The simulated results demonstrate that the DM-AIS with dynamic mutation rate give a good result effectively at a faster speed. The simulated results are dynamically adapting its value when the SINR changes. In comparison, the dynamic mutation rate has shown better results with finer resolutions.

We can then discriminate the difference in results after using the new method with DM-AIS. To justify the results, the new method must be proven to be more effective than any previously suggested method. The new method should have more applications by increasing the number of users and reducing interference. The results shown in Table 3 and Figure 5 provide details for two users at angles 0° and 10° with interference at angles 50°, −50°, 30°, and −30°.

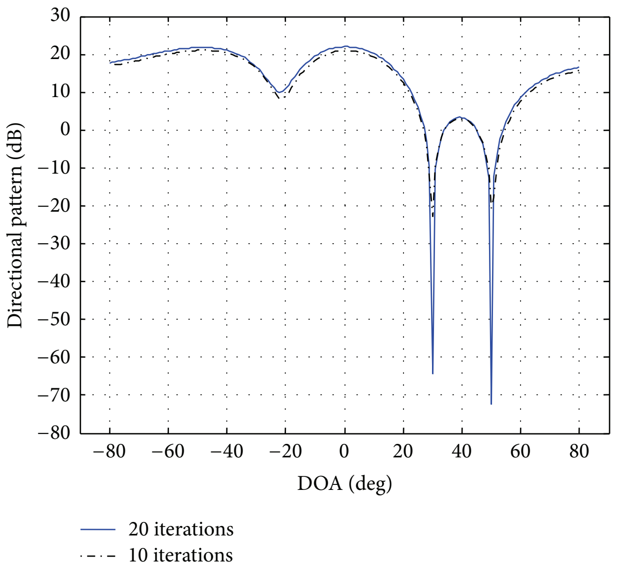


Figure 3: Power response DM-AIS with 10 and 20 iterations for user at 40° with interference at 30° and 50°.

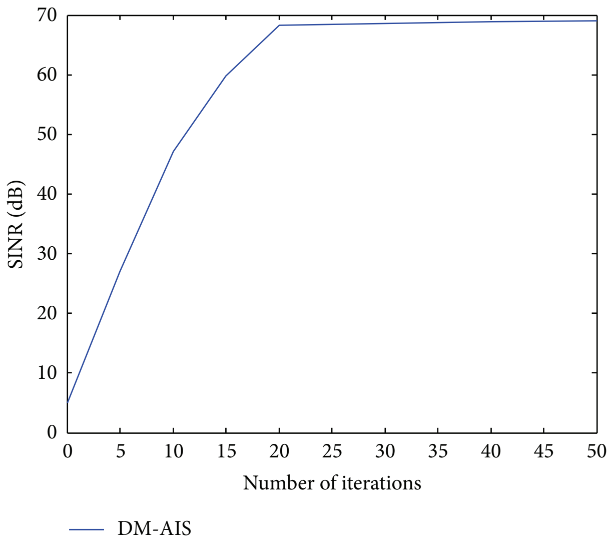


Figure 4: Number of iterations with SINR of DM-AIS for one user at 40° and interference at 50° and 30° by using 4 elements.

Table 3: Weight vector for two users with four interferences using DM-AIS.

Sensor number	Weight	10 iterations	20 iterations
1	W_1	$0.3330 - 0.2789j$	$0.3652 - 0.2806j$
2	W_2	$0.4506 - 0.0414j$	$0.4622 - 0.0401j$
3	W_3	$0.4590 + 0.0487j$	$0.5500 + 0.0474j$
4	W_4	$0.3348 + 0.2723j$	$0.3292 + 0.2748j$

Table 4: Weight vector values of conventional MVDR, mp-QP MVDR, and DM-AIS.

Sensor number	Conventional MVDR	mp-QP MVDR	DM-AIS MVDR
1	$0.1626 - 0.0118j$	$0.1077 + 0.0469j$	$0.1594 - 0.0123j$
2	$0.1249 + 0.0093j$	$0.1269 + 0.0072j$	$0.1224 + 0.0123j$
3	$0.0630 - 0.0026j$	$0.0808 + 0.0264j$	$0.0626 - 0.0020j$
4	$0.0143 + 0.0059j$	$0.0441 - 0.0007j$	$0.0185 + 0.0055j$
5	$0.1264 + 0.0058j$	$0.1106 - 0.0188j$	$0.1196 + 0.0110j$
6	$0.1107 - 0.0291j$	$0.1043 - 0.0196j$	$0.1232 - 0.0308j$
7	$0.0212 - 0.0075j$	$0.1015 - 0.0267j$	$0.0180 - 0.0080j$
8	$0.0794 - 0.0279j$	$0.0846 - 0.0012j$	$0.0747 - 0.0209j$
9	$0.1337 + 0.0280j$	$0.1293 + 0.0069j$	$0.1274 + 0.0230j$
10	$0.1639 + 0.0306j$	$0.1102 - 0.0204j$	$0.1743 + 0.0223j$

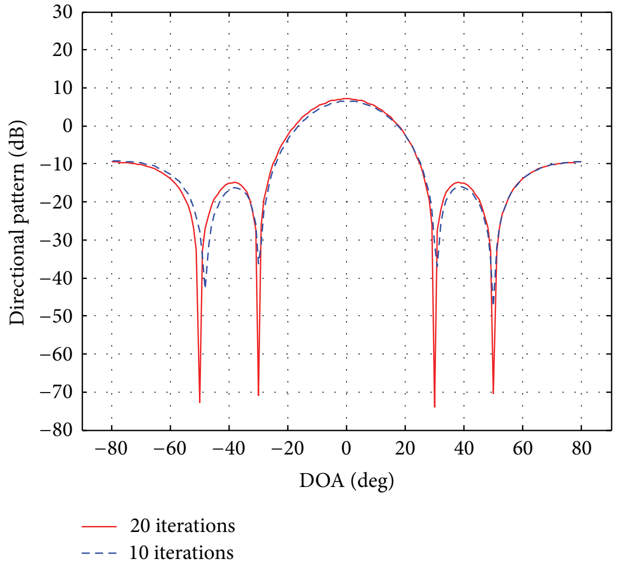


Figure 5: Power response for two users at 0 and 10 interference sources at 50°, −50°, 30°, and −30° with DM-AIS after 10 and 20 iterations.

6.2. Comparison of Conventional MVDR with MP-QP MVDR and DM-AIS. To prove the importance of this project, the results of this work were those of previous work to enhance the MVDR in smart antennas. From the studied literature, no researcher has discussed or addressed beamforming by using four elements in the smart antenna or by applying DM-AIS in the smart antenna. Therefore, the results of this project are compared with robust MVDR beamforming for nulling level control via multiparametric quadratic programming results using 10 elements and with the direction of user at 0° with interference at 40° and −40° [8]. Figure 6 and Table 4 show the results.

Figure 6 shows that all beam patterns have nulls in the interference signals and maintain a distortionless response for the SOI. However, DM-AIS MVDR places deep nulls (with nulling level equal to −90 dB) at the two interference signal sources, whereas mp-QP MVDR obtained a null level equal

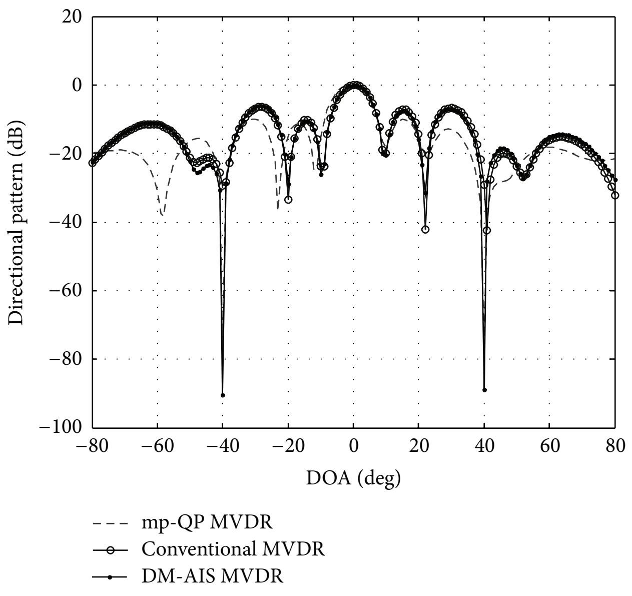


Figure 6: Comparison among DM-IS, mp-QP MVDR, and MVDR of one user at 0° and interference 40° and −40°.

Table 5: Illustration of the SINR for DM-AIS, mp-QP MVDR, and MVDR.

MVDR SINR (dB)	mp-QP MVDR SINR (dB)	DM-AIS MVDR SINR (dB)
25	76.9897	86.9897

to −80 dB. Therefore, the DM-AIS with MVDR response presents lower nulling levels compared with the new mp-QP MVDR. The maximum SINRs calculation for each technique is compared with one another. The maximum value obtained from DM-AIS is shown in Table 5.

This result shows that the new beamforming using artificial intelligence to determine the desired signal of user is effective and that mathematical equations or filter hardware signal processing are unnecessary. For the proposed approach, the weights value vectors make it difficult to obtain the optimum value by using any other algorithm without intensification.

7. Conclusion

A new DM-AIS was presented and applied in adaptive beamforming with four elements of linear antenna arrays. The proposed DM-AIS was able to enhance the MVDR technique through further optimization of weight vector, which aimed to control the nulling level of interference and the directionality of the desired signal. Very low levels of interference with good accuracy were achieved even in the case of multiple users or multiple interferences. The results of the proposed approach were compared with those of the mp-QP MVDR and conventional MVDR, and the effectiveness of the proposed approach in minimizing the power of interference and increasing SINR was observed. Finally, the DM-AIS can be useful to antenna engineers for the pattern synthesis of antenna arrays because of its good accuracy and the lack of a requirement for complicated mathematical functions.

Conflict of Interests

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

Acknowledgment

This work was supported in part by MOSTI (Ministry of Science, Technology and Innovation, Malaysia) with Project no. 01-02-03-SF0202.

References

[1] M. Barrett and R. Arnott, "Adaptive antennas for mobile communications," *Electronics and Communication Engineering Journal*, vol. 6, no. 4, pp. 203–214, 1994.

[2] J. Litva and T. K. Lo, *Digital Beamforming in Wireless Communications*, A Tech House, 1996.

[3] M. Souden, J. Benesty, and S. Affes, "A study of the LCMV and MVDR noise reduction filters," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4925–4935, 2010.

[4] T. S. Ghouse Basha, P. V. Sridevi, and M. N. Giri Prasad, "Enhancement in gain and interference of smart antennas using two stage genetic algorithm by implementing it on beam forming," *International Journal of Electronics Engineering*, vol. 3, no. 2, pp. 265–269, 2011.

[5] Z. Xing-Wei, L. Yuan, L. Hui-jie, and L. Xu-wen, "The difference of nulling antenna technology between geo and leo satellites," *Energy Procedia*, vol. 13, pp. 559–567, 2011.

[6] C. A. Balanisand and P. I. Ioannides, *Introduction of Smart Antennas*, Morgan and Claypool Publication, San Rafael, Calif, USA, 2007.

[7] H. Dahrouj and W. Yu, "Coordinated beamforming for the multicell multi-antenna wireless system," *IEEE Transactions on Wireless Communications*, vol. 9, no. 5, pp. 1748–1759, 2010.

[8] B. Salem, S. K. Tiong, and S. P. Koh, "Beamforming algorithms technique by using MVDR and LCMV," *World Applied Programming*, vol. 2, no. 5, pp. 315–324, 2012.

[9] E. A. P. Habets, J. Benesty, I. Cohen, S. Gannot, and J. Dmochowski, "New insights into the MVDR beamformer in room acoustics," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 1, pp. 158–170, 2010.

[10] T. S. Kiong, M. Ismail, and A. Hassan, "WCDMA forward link capacity improvement by using adaptive antenna with genetic algorithm assisted MDPC beamforming technique," *Journal of Applied Sciences*, vol. 6, no. 8, pp. 1766–1773, 2006.

[11] D. Li, Q. Yin, P. Mu, and W. Guo, "Robust MVDR beamforming using the DOA matrix decomposition," in *Proceedings of the 1st International Symposium on Access Spaces (ISAS '11)*, pp. 105–110, June 2011.

[12] R. Michael and I. Psaromiligkos, "Robust minimum variance beamforming with dual response constraints," *IEEE Transaction on Signal Processing*, vol. 60, 2010.

[13] J. Ye, W. Pang, Y. Wang, D. Lv, and Y. Chen, "Side lobe reduction in thinned array synthesis using immune algorithm," in *Proceedings of the International Conference on Microwave and Millimeter Wave Technology (ICMMT '08)*, pp. 1131–1133, April 2008.

[14] L. N. de Castro, *Fundamentals of Natural Computing*, Chapman & Hall/CRC, 2006.

[15] F. M. Burnet, *The Clonal Selection Theory of Acquired Immunity*, University Press Cambridge, 1959.

[16] Z. D. Zaharis and T. V. Yioultsis, "A novel adaptive beamforming technique applied on linear antenna arrays using adaptive mutated Boolean PSO," *Progress in Electromagnetics Research*, vol. 117, pp. 165–179, 2011.

[17] Y. Wang, S. Gao, H. Yu, and Z. Tang, "Synthesis of antenna array by complex-valued genetic algorithm," *International Journal of Computer Science and Network Security*, vol. 11, no. 1, pp. 91–96, 2011.

[18] F. L. Liu, J. K. Wang, C. Y. Sun, and R. Y. Du, "Robust MVDR beamformer for nulling level control via multi-parametric quadratic programming," *Progress In Electromagnetics Research C*, vol. 20, pp. 239–254, 2011.

[19] J. R. Al-Enezi, M. F. Abbod, and S. Alsharhan, "Artificial immune system models, algorithm and applications," *International Journal of Research and Reviews in Applied Sciences*, vol. 3, no. 2, pp. 118–131, 2010.

[20] G. Renzhou, "Suppressing radio frequency interferences with adaptive beamformer based on weight iterative algorithm," in *Proceedings of the IET Conference Wireless, Mobile and Sensor Networks (CCWMSN '07)*, pp. 648–651, 2007.

[21] A. Ali, Anum, R. L. Ali, and A. ur-Rehman, "An improved gain vector to enhance convergence characteristics of recursive least squares algorithm," *International Journal of Hybrid Information Technology*, vol. 4, pp. 99–107, 2011.

[22] H. Bersini and F. Varela, "Hints for adaptive problem solving gleaned from immune networks," in *Parallel Problem Solving from Nature*, vol. 496 of *Lecture Notes in Computer Science*, pp. 343–354, Springer, Berlin, Germany, 1991.

[23] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," *Physica D: Nonlinear Phenomena*, vol. 22, no. 1–3, pp. 187–204, 1986.

[24] L. N. de Castro and F. J. von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239–251, 2002.

[25] L. Lin, X. Guanghua, L. Dan, and Z. Shuanfeng, "Immune clonal selection optimization method with mixed mutation strategies," in *Proceedings of the 2nd International Conference on Bio-Inspired Computing: Theories and Applications (BICTA '07)*, pp. 37–41, September 2007.

[26] L. N. de Castro and F. J. von Zuben, "The clonal selection algorithm with engineering applications," in *Proceedings of GECCO*, pp. 36–39, 2000.

[27] R. González Ayestarán, M. F. Campillo, and F. Las-Heras, "Multiple support vector regression for antenna array characterization and synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 9, pp. 2495–2501, 2007.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 672983, 9 pages
http://dx.doi.org/10.1155/2014/672983

Research Article

Hybrid Biogeography-Based Optimization for Integer Programming

Zhi-Cheng Wang and Xiao-Bei Wu

College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China

Correspondence should be addressed to Xiao-Bei Wu; xb.wu@ymail.com

Received 15 January 2014; Accepted 17 March 2014; Published 3 June 2014

Academic Editors: S. Balochian and Y. Zhang

Copyright © 2014 Z.-C. Wang and X.-B. Wu. 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.

Biogeography-based optimization (BBO) is a relatively new bioinspired heuristic for global optimization based on the mathematical models of biogeography. By investigating the applicability and performance of BBO for integer programming, we find that the original BBO algorithm does not perform well on a set of benchmark integer programming problems. Thus we modify the mutation operator and/or the neighborhood structure of the algorithm, resulting in three new BBO-based methods, named BlendBBO, BBO_DE, and LBBO_LDE, respectively. Computational experiments show that these methods are competitive approaches to solve integer programming problems, and the LBBO_LDE shows the best performance on the benchmark problems.

1. Introduction

An integer programming problem is a discrete optimization problem where the decision variables are restricted to integer values. In computer science and operations research, a remarkably wide range of problems, such as project scheduling, capital budgeting, goods distribution, and machine scheduling, can be expressed as integer programming problems [1–6]. Integer programming also has applications in bioinspired computational models such as artificial neural networks [7, 8].

The general form of an integer programming model can be stated as

$$\begin{aligned} \min \quad & f(\vec{x}) \\ \text{s.t.} \quad & \vec{x} \in \vec{S} \subseteq \mathbb{Z}^D, \end{aligned} \tag{1}$$

where $\vec{x}$ is a D-dimensional integer vector, $\mathbb{Z}^D$ is a D-dimensional discrete space of integers, and $\vec{S}$ is a feasible region that is not necessarily bounded. Any maximization version of integer programming problems can be easily transformed to a minimization problem.

One of the most well-known deterministic approaches for solving integer programming problems is the branch-and-bound algorithm [9]. It uses a "divide-and-conquer" strategy to split the feasible region into subregions, obtaining for each subregion a lower bound by ignoring the integrality constraints and checking whether the corresponding solution is a feasible one; if so, the current solution is optimum to the original problem; otherwise recursively split and tackle the subregions until all the variables are fixed to integers.

However, integer programming is known to be *NP*-hard [10], and thus the computational cost of deterministic algorithms increases very rapidly with problem size. In recent years, evolutionary algorithms (EA), which are stochastic search methods inspired by the principles of natural biological evolution, have attracted great attention and have been successfully applied to a wide range of computationally difficult problems. These heuristic algorithms do not guarantee finding the exact optimal solution in a single simulation run, but in most cases they are capable of finding acceptable solutions in a reasonable computational time.

Genetic algorithms (GA) are one of the most popular EA, but the encoding of the integer search space with fixed length binary strings as used in standard GA is not feasible for integer problems [11]. Many other heuristics, such as evolutionary strategy (ES) [12], particle swarm optimization (PSO) [13], and differential evolution (DE) [14], are initially proposed for continuous optimization problems. However, they can be adapted to integer programming by embedding the integer space into the real space and truncating or rounding real values to integers, and the applicability and performance of such approach are demonstrated by experimental studies.

Kelahan and Gaddy [15] conducted an early study that performs random search in integer spaces in the spirit of a $(1 + 1)$-ES; that is, at each iteration a child solution vector is generated by adding a random vector to the parent vector, and the better one between the parent and the child is kept for the next generation. Rudolph [11] developed a $(\mu + \lambda)$-ES based algorithm, which uses the principle of maximum entropy to guide the construction of a mutation distribution for arbitrary search spaces.

Laskari et al. [16] studied the ability of PSO for solving integer programming problems. On their test problems, PSO outperforms the branch-and-bound method in terms of number of function evaluations (NFE), and PSO exhibits high success rates even in cases where the branch-and-bound algorithm fails. Improved versions of PSO, including the quantum-behaved PSO which is based on the principle of state superposition and uncertainty [17] and barebones PSO which is based on samples from a normal distribution and requires no parameter tuning [18], have also been applied and shown to be efficient alternatives to integer programming problems.

Omran and Engelbrecht [19] investigated the performance of DE in integer programming. They tested three versions of DE and found that the self-adaptive DE (SDE) requiring no parameter tuning is the most efficient and performs better than PSO.

In this paper, we propose three algorithms for integer programming based on a relatively new bioinspired method, namely, biogeography-based optimization (BBO). We modify the mutation operator of the original BBO to enhance its exploration or global search ability and adopt a local neighborhood structure to avoid premature convergence. Experimental results show that our methods are competitive approaches to solving integer programming problems.

2. Biogeography-Based Optimization

Biogeography is the science of the geographical distribution of biological organisms over space and time. MacArthur and Wilson [20] established the mathematical models of island biogeography, which show that the species richness of an island can be predicted in terms of such factors as habitat area, immigration rate, and extinction rate. Inspired by this, Simon [21] developed the BBO algorithm, where a solution vector is analogous to a habitat, the solution components are analogous to a set of suitability index variables (SIVs), and the solution fitness is analogous to the species richness or habitat suitability index (HSI) of the habitat. Central to the algorithm is the equilibrium theory of island biogeography, which indicates that high HSI habitats have a high species

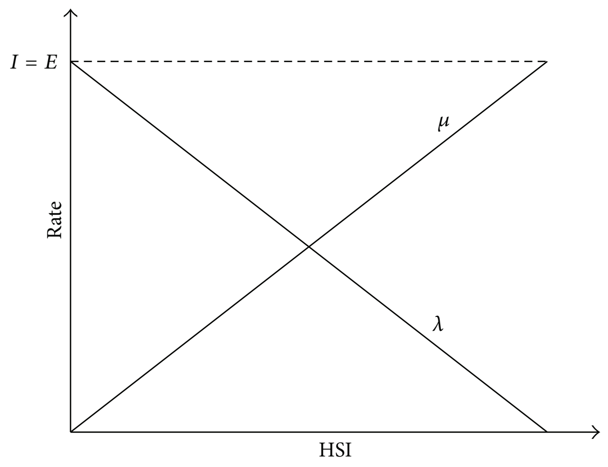


FIGURE 1: A linear model of emigration and immigration rates of a habitat.

emigration rate and low HSI habitats have a high species immigration rate. For example, in a linear model of species richness (as illustrated in Figure 1), a habitat H_i's immigration rate λ_i and emigration rate μ_i are calculated based on its fitness f_i as follows:

$$\lambda_i = I\left(\frac{f_{\max} - f_i}{f_{\max} - f_{\min}}\right)$$
$$\mu_i = E\left(\frac{f_i - f_{\min}}{f_{\max} - f_{\min}}\right), \tag{2}$$

where $f_{\max}$ and $f_{\min}$ are, respectively, the maximum and minimum fitness values among the population and I and E are, respectively, the maximum possible immigration rate and emigration rate. However, there are other nonlinear mathematical models of biogeography that can be used for calculating the migration rates [22, 23].

Migration is used to modify habitats by mixing features within the population. BBO also has a mutation operator for changing SIV within a habitat itself and thus probably increasing diversity of the population. For each habitat H_i, a species count probability P_i computed from λ_i and μ_i indicates the likelihood that the habitat was expected a priori to exist as a solution for the problem. In this context, very high HSI habitats and very low HSI habitats are both equally improbable, and medium HSI habitats are relatively probable. The mutation rate of habitat H_i is inversely proportional to its probability:

$$\pi_i = \pi_{\max}\left(1 - \frac{P_i}{P_{\max}}\right), \tag{3}$$

where $\pi_{\max}$ is a control parameter and $P_{\max}$ is the maximum habitat probability in the population.

Algorithm 1 describes the general framework of BBO for a D-dimensional global numerical optimization problem (where l_d and u_d are the lower and upper bounds of the dth dimension, respectively, and rand is a function that generates a random value uniformly distributed in $[0, 1]$).

```
(1) Randomly initialize a population of $n$ solutions (habitats);
(2) while stop criterion is not satisfied do
(3)     for $i = 1$ to $n$ do
(4)         Calculate $\lambda_i$, $\mu_i$, and $\pi_i$ according to $f_i$;
(5)     for $i = 1$ to $n$ do
(6)         for $d = 1$ to $D$ do
(7)             if rand() < $\lambda_i$ then //migration
(8)                 Select a habitat $H_j$ with probability $\propto \mu_j$;
(9)                 $H_{i,d} \leftarrow H_{j,d}$;
(10)    for $i = 1$ to $n$ do
(11)        for $d = 1$ to $D$ do
(12)            if rand() < $\pi_i$ then //mutation
(13)                $H_{i,d} \leftarrow l_d + \text{rand}() \times (u_d - l_d)$;
(14)    Evaluate the fitness values of the habitats;
(15)    Update $f_{\max}$, $P_{\max}$, and the best known solution;
(16) return the best known solution.
```

Algorithm 1: The original BBO algorithm.

Typically, in Line 8 we can use a roulette wheel method for selection, the time complexity of which is $O(n)$. It is not difficult to see that the complexity of each iteration of the algorithm is $O(n^2D + nO(f))$, where $O(f)$ is the time complexity for computing the fitness function f.

3. Biogeography-Based Heuristics for Integer Programming

In BBO, the migration operator provides good exploitation ability, while the broader exploration of the search space is mainly based on the mutation operator. Simon [21] suggested that $\pi_{\max}$ should be set to a small value (about 0.01), which results in low mutation rates. However, when being applied to integer programming, we need to use higher mutation rates to improve the exploration of search space. According to our experimental studies, when $\pi_{\max}$ is set to about 0.25~0.3, the BBO algorithm exhibits the best performance on integer programming problems.

Note that the migration operator does not violate the integer constraints, and the rounding of real values to integers is required only after mutations (Line 13 of Algorithm 1). Nevertheless, even using a higher mutation rate, the performance of BBO is far from satisfactory for integer programming. This is mainly because random mutation operator does not utilize any information of the population to guide the exploration of search space. In this work, we introduce two other mutation operators to BBO, which results in three variants of BBO for integer programming.

3.1. A Blended Mutation Operator.

In the first variant, namely, BlendBBO, we use a blended mutation operator, which is motivated by the blended crossover operator used by Mühlenbein and Schlierkamp-Voosen [24] in GA and by Ma and Simon [25] in constrained optimization. In our approach, if a component of vector H_i is subject to mutate, we first select another vector H_j with probability $\propto \mu_j$ and then use the following equation to work out the new value of the component:

$$H_{i,d} = \text{round}\left(\alpha H_{i,d} + (1-\alpha) H_{j,d}\right), \tag{4}$$

where α is a random value uniformly distributed in $[0,1]$. Note that if the dth dimension of the search space has a bound, (4) will never result in a value outside the bound.

Moreover, we employ an elitism mechanism in solution update (as used in ES [12, 26]): the migration operator always generates a new vector H_i' for each existing vector H_i (rather than directly changing H_i); if H_i' is better than H_i, no mutation will be applied and H_i' directly enters to the next generation; otherwise the mutation operator is applied to H_i. This not only decreases the required NFE but also increases the convergence speed of the algorithm. The algorithm flow of BBO with the blended mutation is presented in Algorithm 2.

3.2. DE Mutation Operator.

The second variant, namely, BBO_DE, replaces the random mutation operator with the mutation operator of DE, which mutates a vector component by adding the weighted difference between the corresponding components of two randomly selected vectors to a third one:

$$H_{i,d} = \text{round}\left(H_{r_1,d} + F\left(H_{r_2,d} - H_{r_3,d}\right)\right), \tag{5}$$

where r_1, r_2, and r_3 are three unique randomly selected habitat indices that are different to i, and F is a constant scaling coefficient.

DE is well known for its good exploration ability, and the combination of BBO migration and DE mutation achieves a good balance between exploitation and exploration. BBO_DE also uses our new solution update mechanism described above. Therefore, the algorithm flow of BBO_DE simply replaces Lines 15 and 16 of Algorithm 2 with the DE mutation operation described by (5).

```
(1) Randomly initialize a population of $n$ solutions (habitats);
(2) while stop criterion is not satisfied do
(3)     for $i = 1$ to $n$ do
(4)         Calculate $\lambda_i$, $\mu_i$, and $\pi_i$ according to $f_i$;
(5)     for $i = 1$ to $n$ do
(6)         Let $H_i' = H_i$;
(7)         for $d = 1$ to $D$ do
(8)             if rand() < $\lambda_i$ then //migration
(9)                 Select a habitat $H_j$ with probability $\propto \mu_j$;
(10)                $H_{i,d}' \leftarrow H_{j,d}$;
(11)        if $f(H_i) < f(H_i')$ then $H_i \leftarrow H_i'$;
(12)        else
(13)          for $d = 1$ to $D$ do
(14)            if rand() < $\pi_i$ then //mutation
(15)                Let $\alpha$ = rand() and select a habitat $H_j$ with probability $\propto \mu_j$;
(16)                $H_{i,d} \leftarrow \text{round}(\alpha H_{i,d} + (1-\alpha)H_{j,d})$;
(17)    Evaluate the fitness values of the habitats;
(18)    Update $f_{\max}$, $P_{\max}$, and the best known solution;
(19) return the best known solution.
```

Algorithm 2: The BBO with the blended mutation for integer programming.

3.3. Local Topology of the Population. The original BBO uses a fully connected topology; that is, all the individual solutions are directly connected in the population and can migrate with each other. But such a global topology is computationally intensive and is prone to premature convergence. To overcome this problem, our third variant replaces the global topology with a local one. One of the simplest local topologies is the ring topology, where each individual is directly connected to two other individuals [27, 28]. But here we employ a more generalized local topology, the random topology, where each individual has K immediate neighbors that are randomly selected from the population and K is a control parameter [28].

In consequence, whenever an individual vector H_i is to be immigrated, the emigrating vector is chosen from its neighbors rather than the whole population, based on the migration rates. The neighborhood structure can be saved in an $n \times n$ matrix L: if two habitats H_i and H_j are directly connected then $L(i, j) = 1$; otherwise $L(i, j) = 0$. It is easy to see that the complexity of each iteration of the algorithm is $O(nKD + nO(f))$.

Storn and Price [14] have proposed several different strategies on DE mutation. The scheme of (5) is denoted as DE/rand/1. Another scheme is named DE/best/1, which always chooses the best individual of the population as H_{r_1} in (5). Omran et al. [29] extended it to the DE/lbest/1 scheme, which uses a ring topology and always chooses the better neighbor of the vector to be mutated.

In our approach, BBO migration and DE mutation share the same local random topology. That is, each H_i individual has K neighbors, and at each time an H_j is chosen from the neighbors with probability $\propto \mu_j$ to participate in the mutation such that

$$H_{i,d} = \text{round}\left(H_{j,d} + F\left(H_{r_2,d} - H_{r_3,d}\right)\right). \qquad (6)$$

Moreover, if the current best solution has not been improved after every n_p generation (where n_p is a predefined constant), we reset the neighborhood structure randomly.

The third variant is named LBBO_LDE, and it also uses the same solution update mechanism as the previous two variants.

4. Computational Experiments

We test the three variants of BBO on a set of integer programming benchmark problems, which are taken from [16, 30, 31] and frequently encountered in the relevant literature. The details of the benchmark problems are described in the Appendix. For comparison, we also implement the basic BBO, DE, and SDE [19] for integer programming. The branch-and-bound method is not included for comparison, because it has shown that DE outperforms branch-and-bound on most test problems [16, 19].

For all the six algorithms, we use the same population size $n = 50$ and run them on each problem for 40 times with different random seeds. The migration control parameters are set as $I = E = 1$ for BBO, BlendBBO, BBO_DE, and LBBO_LDE, and the mutation control parameter $\pi_{\max}$ is set to 0.01 for BBO and 0.25 for BlendBBO (BBO_DE and LBBO_LDE do not use this parameter). Empirically, the neighborhood size K and the threshold of nonimprovement generations n_p are both set to 3 for LBBO_LDE. The other parameters with regard to DE and SDE are set as suggested in [14, 32].

The first two problems F_1 and F_2 are high-dimensional problems. For F_1, we, respectively, consider it in 10 and 30 dimensions. Table 1 presents the success rates (SR) and required NFE of the algorithms to achieve the optimum in 10 dimensions, and Figure 2 presents the corresponding

TABLE 1: SR and required NFE of the algorithms on F_1 in 10 dimensions.

Method	SR	Best	Worst	Mean	Std
BBO	0%	NA	NA	NA	NA
BlendBBO	20%	1946	2215	2080.50	190.21
DE	100%	3200	4100	3777.50	239.78
SDE	100%	3050	4050	3762.40	265.23
BBO_DE	100%	3102	3964	3674.80	269.07
LBBO_LDE	100%	2061	2694	2493.75	145.59

TABLE 2: SR and required NFE of the algorithms on F_1 in 30 dimensions.

Method	SR	Best	Worst	Mean	Std
BBO	0%	NA	NA	NA	NA
BlendBBO	0%	NA	NA	NA	NA
DE	0%	NA	NA	NA	NA
SDE	85%	7850	11350	9301.00	1130.10
BBO_DE	90%	6765	9752	8204.25	920.39
LBBO_LDE	100%	5923	7071	6471.60	272.29

convergence curves of the algorithms. As we can see, the original BBO fails to solve the problem, and the SR of BlendBBO is only 20%. The four algorithms utilizing the DE mutation operator can guarantee the optimal result on the 10-dimensional problem, among which LBBO_LDE shows the best performance, and the other three algorithms have similar performance, but the result of BBO_DE is slightly better than DE and SDE.

Table 2 and Figure 3, respectively, present the results and the convergence curves of the algorithms on F_1 in 30 dimensions. On this high-dimensional problem, BBO, BlendBBO, and DE all fail to obtain the optimum, SDE and BBO_DE, respectively, have SR of 85% and 90% for obtaining the optimum, and only our LBBO_LDE can always guarantee the optimum.

From the convergence curves we can also find that the BBO algorithm converges very fast at the early stage, but thereafter its performance deteriorates because it is ineffective to explore other potentially promising areas of the search space. By combining with the DE mutation operator, our hybrid BBO methods inherit the fast convergence speed of BBO, at the same time taking advantage of the exploration ability of DE.

For F_2, we, respectively, consider it in 5 and 15 dimensions, the experimental results of which are, respectively, presented in Tables 3 and 4 and the convergence curves of which are presented in Figures 4 and 5. The results are similar to those of F_1: for the low dimensional problem, SDE, BBO_DE, and LBBO_LDE are efficient; for the high-dimensional problem, only LBBO_LDE can guarantee the optimum; the performance LBBO_LDE is the best while that of BBO is the worst; SDE performs better than DE and BBO_DE performs slightly better than SDE, and BlendBBO outperforms BBO but is worse than the algorithms with the DE mutation operator.

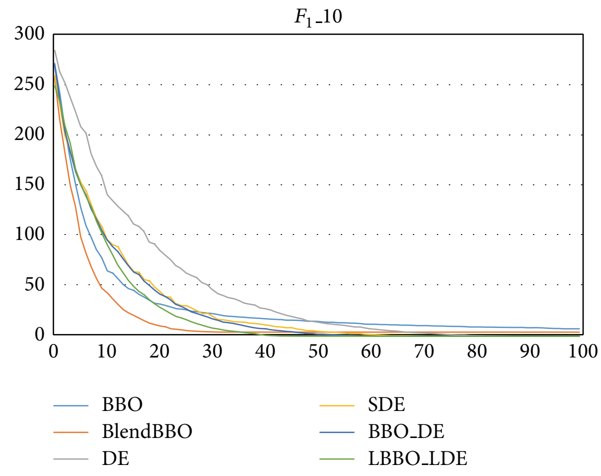


FIGURE 2: The convergence curves of the algorithms on F_1 in 10 dimensions, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

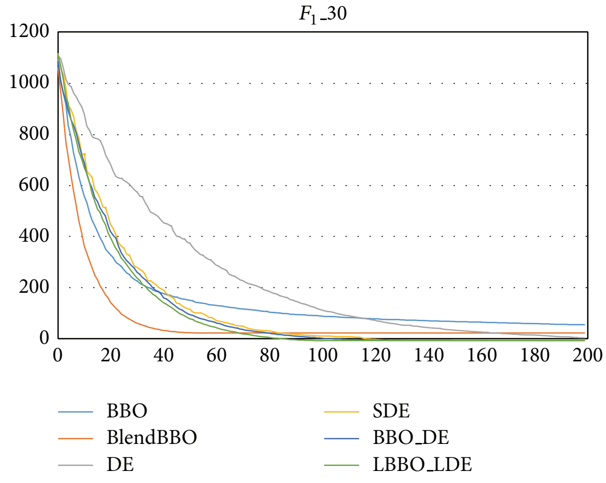


FIGURE 3: The convergence curves of the algorithms on F_1 in 30 dimensions, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

TABLE 3: SR and required NFE of the algorithms on F_2 in 5 dimensions.

Method	SR	Best	Worst	Mean	Std
BBO	10%	3663	5168	4415.50	1064.20
BlendBBO	55%	1154	1545	1363.36	121.94
DE	95%	1500	2100	1797.37	188.91
SDE	100%	1450	2600	2022.35	368.17
BBO_DE	100%	1773	2669	2336.65	258.22
LBBO_LDE	100%	1058	1898	1451.20	214.48

F_3 is a 5-dimensional problem more difficult than F_2. As we can see from the results shown in Table 5, BBO and BlendBBO always fail on the problem, and DE, SDE, and LBBO_LDE can guarantee the optimum. The required

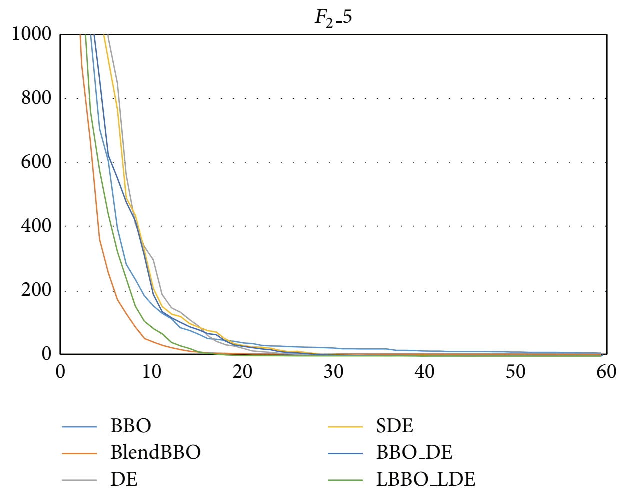


Figure 4: The convergence curves of the algorithms on F_2 in 5 dimensions, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

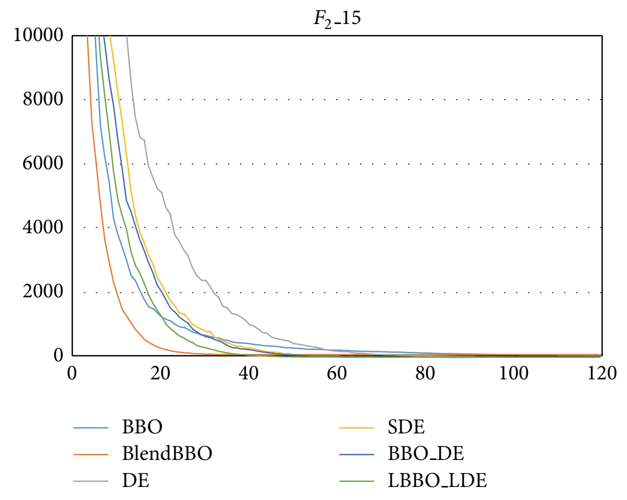


Figure 5: The convergence curves of the algorithms on F_2 in 15 dimensions, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

Table 4: SR and required NFE of the algorithms on F_2 in 15 dimensions.

Method	SR	Best	Worst	Mean	Std
BBO	0%	NA	NA	NA	NA
BlendBBO	2.5%	4815	4815	4815	NA
DE	95%	5300	6650	5978.95	335.95
SDE	95%	5000	6400	5698.32	389.24
BBO_DE	97.5%	5030	6210	5639.11	362.69
LBBO_LDE	100%	3488	4528	4188.30	300.77

NFE of DE is slightly better than SDE and LBBO_LDE, but LBBO_LDE converges faster than DE, as shown in Figure 6.

F_4 is a relatively easy problem, on which even the worst BBO has an SR of 75%, and all the other algorithms can

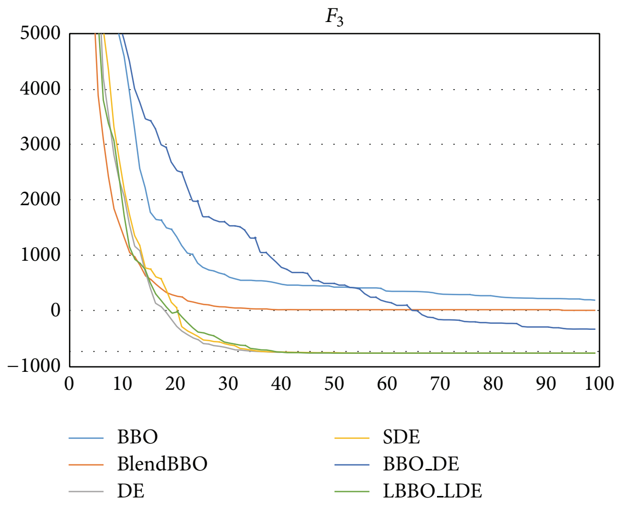


Figure 6: The convergence curves of the algorithms on F_3, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

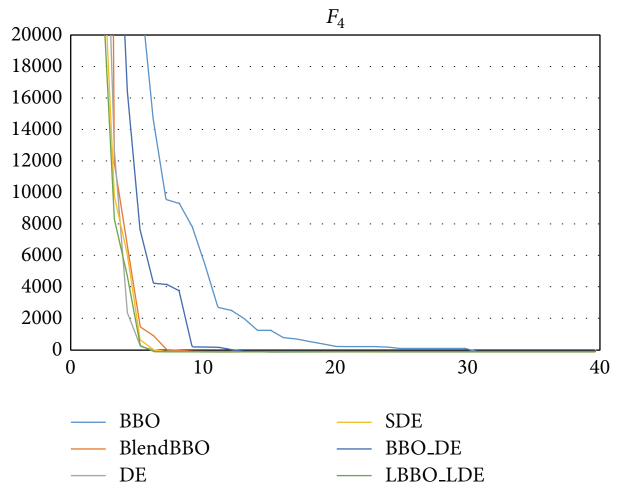


Figure 7: The convergence curves of the algorithms on F_4, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

guarantee the optimum. LBBO_LDE is the best one in terms of both NFE and convergence speed, as shown in Table 6 and Figure 7.

The remaining three test problems are also relatively easy. The experimental results are presented in Tables 7, 8, and 9, and the convergence curves are shown in Figures 8, 9, and 10, respectively. As we can clearly see, the four algorithms with the DE mutation operator can always obtain the optima on these problems, and LBBO_LDE shows the best performance.

In summary, our LBBO_LDE outperforms the other algorithms on all of the test problems. Generally speaking, the original BBO converges fast at first, but it is easy to be trapped by the local optima. BlendBBO alleviates the dilemma to a certain degree, but the DE mutation operator is more effective than the blended mutation operator, as

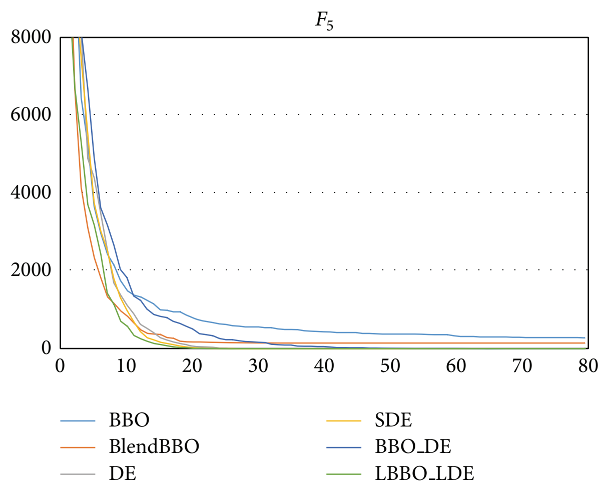


Figure 8: The convergence curves of the algorithms on F_5, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

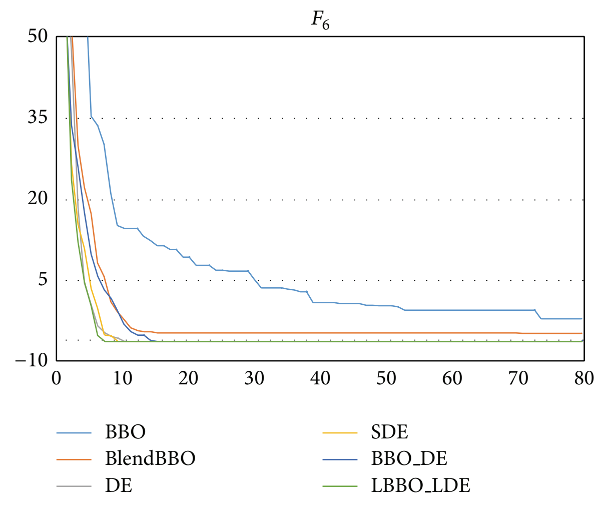


Figure 9: The convergence curves of the algorithms on F_6, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

Table 5: SR and required NFE of the algorithms on F_3.

Method	SR	Best	Worst	Mean	Std
BBO	0%	NA	NA	NA	NA
BlendBBO	0%	NA	NA	NA	NA
DE	100%	2050	2950	2490.00	262.38
SDE	100%	2500	4050	3260.00	638.05
BBO_DE	10%	4810	5246	5028.00	308.30
LBBO_LDE	100%	1758	4181	2958.85	849.24

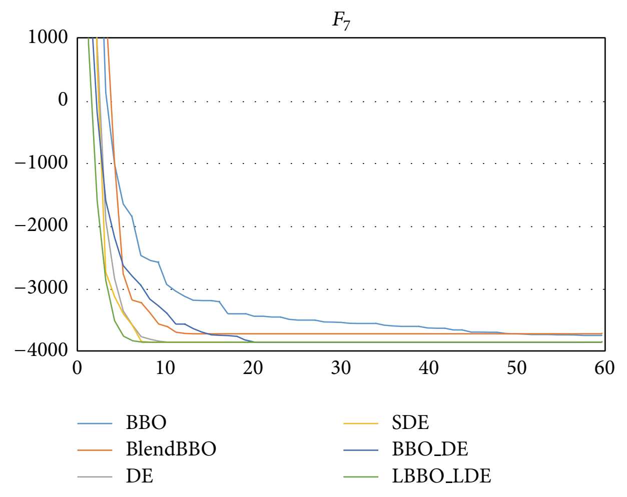


Figure 10: The convergence curves of the algorithms on F_7, where the vertical axis represents the objective value and the horizontal axis represents the number of generations.

Table 6: SR and required NFE of the algorithms on F_4.

Method	SR	Best	Worst	Mean	Std
BBO	75%	477	3429	1726.87	1020.19
BlendBBO	100%	258	552	424.50	85.34
DE	100%	300	650	420.00	93.75
SDE	100%	250	600	520.00	129.65
BBO_DE	10%	59	1058	632.30	277.77
LBBO_LDE	100%	236	525	400.60	72.20

Table 7: SR and required NFE of the algorithms on F_5.

Method	SR	Best	Worst	Mean	Std
BBO	0%	NA	NA	NA	NA
BlendBBO	25%	1223	2676	1842	692.53
DE	100%	1200	1700	1445.00	140.39
SDE	100%	1100	1750	1550.50	196.45
BBO_DE	100%	2473	4498	3681.25	620.36
LBBO_LDE	100%	1012	1874	1532.35	249.46

Table 8: SR and required NFE of the algorithms on F_6.

Method	SR	Best	Worst	Mean	Std
BBO	45%	140	1989	1150.22	678.16
BlendBBO	80%	178	635	455.81	128.81
DE	100%	200	550	392.50	140.39
SDE	100%	200	500	405.20	103.72
BBO_DE	100%	196	995	708.50	198.38
LBBO_LDE	100%	183	511	410.05	86.37

demonstrated by our experimental results. By combining BBO and DE, the BBO_DE algorithm provides an efficient alternative to popular methods such as SDE. The local topology used in LBBO_LDE further improves the search ability and suppresses the premature convergence, especially on high-dimensional problems where the performance of DE and SDE deteriorates quickly. Therefore, LBBO_LDE is a very competitive heuristic method for solving integer programming problem.

Table 9: SR and required NFE of the algorithms on F_7.

Method	SR	Best	Worst	Mean	Std
BBO	60%	167	3403	1714.75	986.18
BlendBBO	70%	262	721	459.50	149.00
DE	100%	350	600	480.00	76.78
SDE	100%	300	650	451.00	89.33
BBO_DE	100%	479	1327	978.40	311.11
LBBO_LDE	100%	249	614	389.45	96.08

5. Conclusion

In this paper we develop three algorithms for integer programming based on the BBO heuristic. The BlendBBO uses a blended mutation operator, BBO_DE integrates the DE mutation operator, and LBBO_LDE further uses a local neighborhood structure for selecting individuals for migration and mutation. Experimental results show that LBBO_LDE has the best performance on a set of benchmark integer programming problem.

In general, the LBBO_LDE algorithm with local neighborhood size K of 3~5 is efficient on the test problem, but none of the values can provide the best performance on all the problems. Currently we are studying a mechanism that dynamically adjusts the neighborhood size as well as other control parameters according to the search state [33]. Moreover, the test problems considered in the paper only have bounds for decision variables but do not include other constraints, and we are extending the proposed approach to solve more complex constrained optimization problems, including multiobjective ones [34–36]. We also believe that our approach can be adapted to effectively handle other kinds of combinatorial optimization problems, such as 0-1 integer programming and permutation-based optimization.

Appendix

Integer Programming Benchmark Problems

Consider

$$\begin{aligned}
&F_1(\vec{x}) = \sum_{i=1}^{D} x_i, \quad \vec{x}^* = (0)^D, \quad F_1(\vec{x}^*) = 0,\\
&F_2(\vec{x}) = \sum_{i=1}^{D} x_i^2, \quad \vec{x}^* = (0)^D, \quad F_2(\vec{x}^*) = 0,\\
&F_3(\vec{x}) = -(15, 27, 36, 18, 12)\,\vec{x}^\top\\
&\qquad + \vec{x}\begin{pmatrix} 35 & -20 & -10 & 32 & -10\\ -20 & 40 & -6 & -31 & 32\\ -10 & -6 & 11 & -6 & -10\\ 32 & -31 & -6 & 38 & -20\\ -10 & 32 & -10 & -20 & 32 \end{pmatrix}\vec{x}^\top,\\
&\vec{x}^* = (0, 11, 22, 16, 6) \quad \text{or } \vec{x}^* = (0, 12, 23, 17, 6),\\
&F_3(\vec{x}^*) = -737,\\
&F_4(\vec{x}) = \left(9x_1^2 + 2x_2^2 - 11\right)^2 + \left(3x_1 + 4x_2^2 - 7\right)^2,\\
&\vec{x}^* = (1, 1), \quad F_4(\vec{x}^*) = 0,\\
&F_5(\vec{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2\\
&\qquad + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4,\\
&\vec{x}^* = (0)^4, \quad F_5(\vec{x}^*) = 0,\\
&F_6(\vec{x}) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2,\\
&\vec{x}^* = (2, -1), \quad F_6(\vec{x}^*) = -6,\\
&F_7(\vec{x}) = -3803.84 - 138.08x_1 - 232.93x_2\\
&\qquad + 123.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2,\\
&\vec{x}^* = (0, 1), \quad F_7(\vec{x}^*) = 3833.12.
\end{aligned} \tag{A.1}$$

In the above problems, the ranges of variables are all set as $\vec{x} \in [-100, 100]^D$.

Conflict of Interests

The authors declare that they have no conflict of interests regarding the publication of this paper.

References

[1] S. Sofianopoulou, “The process allocation problem: a survey of theapplication of graph-theoretic and integer programming approaches,” *Journal of the Operational Research Society*, vol. 43, no. 5, pp. 407–413, 1992.

[2] J. M. Whitacre, “Recent trends indicate rapid growth of nature-inspired optimization in academia and industry,” *Computing*, vol. 93, no. 2-4, pp. 121–133, 2011.

[3] Y. Zheng and J. Xue, “A problem reduction based approach to discrete optimization algorithm design,” *Computing*, vol. 88, no. 1-2, pp. 31–54, 2010.

[4] Y. J. Zheng, S. Y. Chen, Y. Lin, and W. L. Wang, “Bio-inspired optimizationof sustainable energy systems: a review,” *Mathematical Problemsin Engineering*, vol. 2013, Article ID 354523, 12 pages, 2013.

[5] Y. J. Zheng, H. F. Ling, H. H. Shi, H. S. Chen, and S. Y. Chen, “Emergencyrailway wagon scheduling by hybrid biogeography-based optimization,” *Computers & Operations Research*, vol. 43, no. 3, pp. 1–8, 2014.

[6] Y. J. Zheng, H. F. Ling, and J. Y. Xue, “Ecogeography-based optimization: enhancing biogeography-based optimization with ecogeographic barriers and differentiations,” *Computers & Operations Research*, vol. 50, pp. 115–127, 2014.

[7] V. Dua, “A mixed-integer programming approach for optimal configuration of artificial neural networks,” *Chemical Engineering Research and Design*, vol. 88, no. 1, pp. 55–60, 2010.

[8] Y. Tan, J. Wang, and J. M. Zurada, “Nonlinear blind source separation using a radial basis function network,” *IEEE Transactions on Neural Networks*, vol. 12, no. 1, pp. 124–134, 2001.

[9] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, 1988.

[10] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[11] G. Rudolph, "An evolutionary algorithm for integer programming," in *Parallel Problem Solving from Nature*, Y. Davidor, H. P. Schwefel, and R. Männer, Eds., vol. 866 of *Lecture Notes in Computer Science*, pp. 139–148, Springer, Berlin, 1994.

[12] H. G. Beyer and H. P. Schwefel, "Evolution strategies—a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[13] 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.

[14] 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.

[15] R. Kelahan and J. Gaddy, "Application of the adaptive random searchto discrete and mixed integer optimization," *International Journal for Numerical Methods in Engineering*, vol. 12, pp. 289–298, 1978.

[16] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis, "Particle swarm optimizationfor integer programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1582–1587, 2002.

[17] J. Liu, J. Sun, and W. Xu, "Quantum-behaved particle swarm optimizationfor integer programming," in *Neural Information Processing*, I. King, J. Wang, L. W. Chan, and D. Wang, Eds., vol. 4233 of *Lecture Notes in Computer Science*, pp. 1042–1050, Springer, Berlin, Germany, 2006.

[18] M. G. H. Omran, A. Engelbrecht, and A. Salman, "Barebones particle swarm for integer programming problems," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 170–175, April 2007.

[19] M. G. H. Omran and A. P. Engelbrecht, "Differential evolution for integer programming problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 2237–2242, September 2007.

[20] R. MacArthur and E. Wilson, *The Theory of Biogeography*, Princeton University Press, Princeton, NJ, USA, 1967.

[21] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[22] H. Ma, "An analysis of the equilibrium of migration models for biogeography-based optimization," *Information Sciences*, vol. 180, no. 18, pp. 3444–3464, 2010.

[23] H. Ma and D. Simon, "Analysis of migration models of biogeography-based optimization using Markov theory," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 6, pp. 1052–1060, 2011.

[24] H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive models for thebreeder genetic algorithm I. Continuous parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 25–49, 1993.

[25] H. Ma and D. Simon, "Blended biogeography-based optimization for constrained optimization," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 3, pp. 517–525, 2011.

[26] D. Du, D. Simon, and M. Ergezer, "Biogeography-based optimization combined with evolutionary strategy and immigration refusal," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '09)*, pp. 997–1002, October 2009.

[27] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 1, pp. 150–169, 2010.

[28] Y. J. Zheng, H. F. Ling, X. B. Wu, and J. Y. Xue, "Localized biogeographybasedoptimization," *Soft Computing*, 2014.

[29] M. G. Omran, A. P. Engelbrecht, and A. Salman, "Using neighborhood topologies with differential evolution," in *Proceedings of the Workshop of International Conference on Computational Intelligence and Security*, pp. 35–40, 2005.

[30] A. Glankwahmdee, J. S. Liebman, and G. L. Hogg, "Unconstrained discrete nonlinear programming," *Engineering Optimization*, vol. 4, no. 2, pp. 95–107, 1979.

[31] S. Rao, *Engineering Optimization—Theory and Practice*, Wiley Eastern, New Delhi, India, 1996.

[32] M. Omran, A. Salman, and A. Engelbrecht, "Self-adaptive differential evolution," in *Computational Intelligence and Security*, Y. Hao, J. Liu, Y. Wang et al., Eds., vol. 3801 of *Lecture Notes in Computer Science*, pp. 192–199, Springer, Berlin, Germany, 2005.

[33] Y. J. Zheng, H. F. Ling, and Q. Guan, "Adaptive parameters for a modifiedcomprehensive learning particle swarm optimizer," *Mathematical Problemsin Engineering*, vol. 2012, Article ID 207318, 11 pages, 2012.

[34] Y. J. Zheng, Q. Song, and S. Y. Chen, "Multiobjective fireworks optimizationfor variable-rate fertilization in oil crop production," *Applied Soft Computing*, vol. 13, no. 11, pp. 4253–4263, 2013.

[35] Y. J. Zheng and S. Y. Chen, "Cooperative particle swarm optimization formultiobjective transportation planning," *Applied Intelligence*, vol. 39, no. 1, pp. 202–216, 2013.

[36] Y. Zheng, H. Ling, J. Xue, and S. Chen, "Population classificationin fire evacuation: a multiobjective particle swarm optimization approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 70–81, 2014.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 271895, 11 pages
http://dx.doi.org/10.1155/2014/271895

Research Article

A Graph-Based Ant Colony Optimization Approach for Process Planning

JinFeng Wang, XiaoLiang Fan, and Shuting Wan

School of Energy, Power and Mechanical Engineering, North China Electric Power University, Baoding 071003, China

Correspondence should be addressed to JinFeng Wang; wm803@sohu.com

Received 21 February 2014; Accepted 30 April 2014; Published 3 June 2014

Academic Editors: F. Di Martino, D. Simson, and Y. Zhang

Copyright © 2014 JinFeng 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.

The complex process planning problem is modeled as a combinatorial optimization problem with constraints in this paper. An ant colony optimization (ACO) approach has been developed to deal with process planning problem by simultaneously considering activities such as sequencing operations, selecting manufacturing resources, and determining setup plans to achieve the optimal process plan. A weighted directed graph is conducted to describe the operations, precedence constraints between operations, and the possible visited path between operation nodes. A representation of process plan is described based on the weighted directed graph. Ant colony goes through the necessary nodes on the graph to achieve the optimal solution with the objective of minimizing total production costs (TPC). Two cases have been carried out to study the influence of various parameters of ACO on the system performance. Extensive comparative experiments have been conducted to demonstrate the feasibility and efficiency of the proposed approach.

1. Introduction

Process planning is the function which translates the design requirements into the detailed technologically feasible instructions, which involves selecting machining operations, sequencing these operations, choosing manufacturing resources, determining setup plans, machining parameters, and so forth. These activities must be carried out simultaneously to achieve an optimal process plan. But, due to the complexity of part structures and variability of machining environment, process planning is well known as a complicated decision-making process. Computer-aided process planning (CAPP) system will assist human planners in completing the process planning, which is an essential component for linking the various models and processes in a computer-integrated manufacturing system (CIMS) [1].

With the development of computer technologies, CAPP has received much attention during the last three decades and played an increasingly important role in a CIMS [2]. The initial "variant" CAPP systems are based on the group technology (GT) coding and classification system. A baseline process plan for a part family has been defined in such systems. According to the part code, approximately 90% of the process plans can be yielded automatically while the remaining 10% is achieved through modifying the process plans manually. The application of artificial intelligence in CAPP system accelerates the generation of a complete process plan, namely, from the variant CAPP system to the generative CAPP system. A generative CAPP system consists of three main consecutive activities: (1) identifying manufacturing features, (2) determining feasible machining operation and available machining resources, and (3) selecting machining operation and machining resources and sequencing machining operations [3, 4]. This paper focuses on the solution of the third activity and presents an ACO approach to solve the process planning problem.

The rest of this paper is organized as follows. Section 2 introduces previous related work. Process planning problem is described in Section 3. The proposed ACO approach for process planning is given in Section 4. In Section 5, simulation experiments are made and the results are discussed compared with other approaches. Finally, Section 6 concludes the present study and outlines the future study.

2. Previous Related Works

In the past three decades, many optimization approaches have been developed and widely applied for solving process planning problem, such as knowledge-based reasoning approach [5, 6], graph manipulation [7, 8], the genetic algorithm (GA) [9–11], tabu search approach (TS) [4, 12], simulated annealing (SA) algorithm [3, 13], particle swarm optimization (PSO) [14, 15], artificial neural networks [16], ant colony optimization (ACO) [17, 18], and artificial immune system (AIS) [19].

Usher and Sharma [5] proposed an approach of intelligent reasoning based on the feathers of part. Many constraints and criteria were present in operation planning, which were analyzed intelligently to generate the potential operation plans. Usher and Bowden [1] apply an improved operation sequence coding of genetic algorithms for process planning problem, which can determine optimal operation sequences for parts of varying complexity. Zhang et al. [10] proposed a GA for a novel computer-aided process planning (CAPP) model in a job shop manufacturing environment. GA is used to select machining resources and sequence operations simultaneously. The dynamic status of machining resources in the job shop and alternative optimal plans are not taken into account. Li et al. [4] consider the process planning problem as a constraint-based optimization problem and propose a tabu search-based approach to solve it. In the proposed optimization approach, precedence constraints between features and their related operations are defined and classified according to their effects on the plan feasibility and processing quality. Ma et al. [13] modeled the constraints of process planning problems in a concurrent manner. Precedence relationships among all the operations are used to generate the entire solution space with multiple planning tasks. Based on the proposed model, they used an algorithm based on simulated annealing (SA) to search for the optimal solution. Guo et al. [14] proposed a PSO approach to operation planning problem. The initial process plans randomly generated are encoded into particles of the PSO algorithm. To avoid falling into local optimal and improve the particles' movements, several new operators have been developed. Penalty strategy is used considering the evaluation of infeasible particles. Krishna and Mallikarjuna Rao [17] proposed a novel approach to apply the ant colony algorithm as a global search technique for process planning problem by considering various feasibility constraints. Chan et al. model the machine tool selection and operation allocation of flexible manufacturing systems and solve process problem by a fuzzy goal—programming approach based on artificial immune systems.

Recently, to improve the quality of results and efficiency of the search, many hybrid approaches are developed for process planning problem, for example, GA + SA [3], graph manipulation + GA [8], and local search algorithm + PSO [20]. Li et al. [3] developed a hybrid genetic algorithm and a simulated annealing approach for optimizing process plans for prismatic parts. They modeled the process planning as a combinatorial optimization problem with constraints. The evaluation criterion was the combination of machine costs, cutting tool costs, machine change costs, tool change, and setup costs. Ding et al. [20] proposed a hybrid approach to incorporate a genetic algorithm, neural network, and analytical hierarchical process (AHP) for process planning problem. A globally optimized fitness function is defined including the evaluation of manufacturing rules using AHP, calculation of cost and time, and determination of relative weights using neural network techniques. Huang et al. [8] model the process planning problem as a combinatorial optimization problem with constraints and developed a hybrid graph and genetic algorithm (GA) approach. In the approach, graph theory accompanied with matrix theory is embedded into the main frame of GA. The precedence constraints between operations are formulated in an operation precedence graph (OPG). An improved GA was applied to solve process planning problem based on the operation precedence graph (OPG). Wang et al. [21] proposed an optimization approach based on particle swarm optimization (PSO) to solve the process planning problem and introduced a novel solution representation scheme for the application of PSO. In the hybrid approach, two kinds of local search algorithms are incorporated and interweaved with PSO evolution to improve the best solution in each generation.

Although significant improvements have been achieved for process planning problem, there still remains potential for further improvement [22]. For example, optimization approach needs to be improved to be more efficient, and a more reasonable constraint modeling and handling mechanism needs to be developed; also, some practical manufacturing environment should be considered, and the approach should provide the multiple alternative optimal plans.

3. Process Planning Problem Description

3.1. Process Plan Representation. In CAD systems, a part is generally described by features with specific machining meanings, such as planes, chamfers, holes, slots, and steps. Given a part and a set of manufacturing resources, the process planning problem of CAPP can be described as follows.

The CAD information of part is read before process planning. Then, the machining method of each feature is selected according to the attributes of different features, which can be expressed by the various operations eventually. So, it is necessary to determine one or several operations for each feature in advance. The operations consist of machines, cutting tools, and tool approach directions (TAD). A TAD is defined as a direction from which a cutting tool can access a feature [7, 10]. For each feature of part, the selection of machines, cutting tools, and TADs is based on the feature geometry and available machining resources. For a part with m feathers, the relationships between part, feather, and operation are shown in Figure 1.

An example part is shown in Figure 2. The part includes six feathers: F1 (a step), F2 (two holes arranged in a replicated feature), F3 (a through hole), F4 (a slot), F5 (a chamfer), and F6 (two blind holes arranged in a replicated feature). Some feathers may have more than one machining method. Each machining method has different selection of machines,

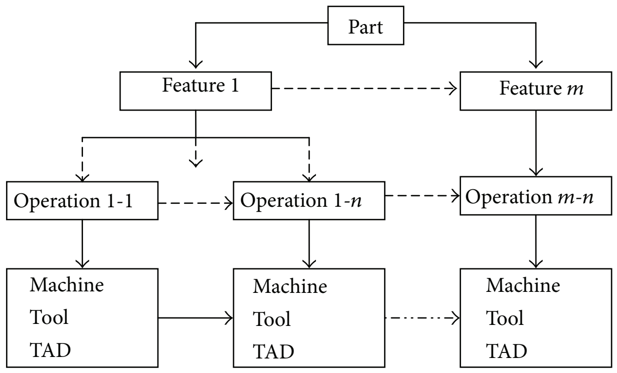


FIGURE 1: The representation of process plan for a part with *m* feathers.

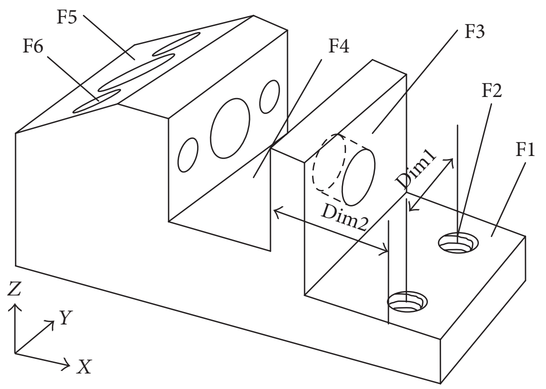


FIGURE 2: An example part.

cutting tools, and TADs. For the example part, the feather of F3 may have two different machining methods of drilling → reaming and drilling → grinding. However, it is possible to have different combination of machines, cutting tools, and TAD even though the selection is overlapped [11]. For any part, TAD includes six directions, that is, $+X$, $-X$, $+Y$, $-Y$, $+Z$, and $-Z$. However, it is common that some TAD will be likely to be discarded for the interference between feathers. For example, the drilling of F6 has two possible TADs, that is, $-X$ and $+X$, because the tool cannot access the hole from the direction of $-X$, and the TAD of $-X$ will be discarded. The features and their valid TADs can be recognized using a geometric reasoning approach [23, 24]. The final result of operation selection for the example part is shown in Table 1. "Op" represents operation; for example, "Op_1" represents operation 1. There is only 1 operation for the feathers of F1, F4, and F5 and 2 operations for the feathers of F2, F3, and F6.

3.2. Precedence Constraints. Process planning involves determining in what order to perform a set of selected operations such that the resulting order satisfies the precedence constraints. These constraints are established by considering both a large number of geometrical interactions and technological requirements between the various factors [1, 3, 8, 25], which cause process planning to become more complicated. The constraints can be divided into the feasibility constraints and optimality constraints [1]. A feasible process plan is deemed to be one which does not violate any of the feasibility constraints. The optimality constraints affect the quality, cost, and efficiency of a feasible process plan, which may be violated at certain times in cases of contradictions to some feasibility constraints. Faheem et al. indicate constraint affecting the generation of process plans which can be classified as "hard" or "soft" constraints [25]. Hard constraints affect the manufacturing feasibility and a process plan should be consistent with these constraints. Soft constraints only affect the quality, cost, or efficiency of a feasible process plan. Many constraints and rules have been proposed and summarized [1, 4, 9, 10]. These precedence constraints are summarized as follows [18].

Rule 1. Primary surfaces prior to secondary surface.

Rule 2. Planes prior to its associated features.

Rule 3. Rough machining operation prior to finishing machining operation.

Rule 4. Datum surfaces prior to its associated features.

Rule 5. Some good manufacturing practice. For example, features related to thin wall should be machined first; features that caused tool damage and failure of clamping potentially should be machined before or later, and feathers that affect the cost or the quality of machining should be machined first.

These constraints between machining operations can be used to constrain the search in a smaller space and enhance search efficiency. Some examples of the above precedence constraints for the example part in Figure 2 are illustrated in Table 2.

3.3. Process Plan Evaluation Criterion. The most common evaluation criteria for process plan include minimum number of setups, shortest process time, and minimum machining cost. Váncza and Márkus used number of setups, number of tool changes, and total cost of individual operations as evaluation criteria [9]. Usher and Sharma used number of setups, continuity of motion, and loose precedence as evaluation criteria [5]. Zhang et al. used machine costs, cutting tool costs, number of machine changes, number of tool changes, and number of setups as evaluation criteria [10]. Many evaluation criteria have been proposed, which include process time, number of setups, number of tool changes, number of machine changes, continuity of motion, and total cost of individual operations. Because the detailed information on machining parameters is not available at this stage, the total machining time cannot be used for plan evaluation. In this paper, five cost evaluation criteria are adopted and are similar to the criteria in paper [3, 4].

(1) Total machine cost (TMC) is

$$\text{TMC} = \sum_{i=1}^{n} \text{MC}_i, \tag{1}$$

Table 1: Operation selection for the example part.

Feathers	Operations	Machines	Tools	TADs
F1	Milling (Op_1)	Vertical milling machine (M1)	Milling cutter (T1)	$+X$, $+Z$
F2	Drilling (Op_2)	Vertical milling machine (M1)	Drill (T2)	$-Z$
	Tapping (Op_3)	Drilling press (M2)	Tapping tool (T3)	
F3	Drilling (Op_4)	Vertical milling machine (M1)	Drill (T4)	$-X$
	Reaming (Op_5)	Drilling press (M2)	Reamer (T5)	
F4	Milling (Op_6)	Vertical milling machine (M1)	Slot cutter (T6)	$+Z$
F5	Milling (Op_7)	Vertical milling machine (M1)	Chamfer cutter (T7)	$-Z$, $+Y$
F6	Drilling (Op_8)	Vertical milling machine (M1)	Drill (T8)	$+X$
	Reaming (Op_9)	Drilling press (M2)	Reamer (T9)	

Table 2: Precedence constraints between operations.

Features	Operations	Precedence constraints description
F1	Op_1	Op_1 is prior to Op_2 and Op_3 for *Rule 2*.
		Op_1 is prior to Op_4 and Op_5 for *Rule 5*.
F2	Op_2	Op_2 is prior to Op_3 for *Rule 3*.
F3	Op_4	Op_4 is prior to Op_5 for *Rule 3*.
	Op_4, Op_5	Op_4 and Op_5 are prior to Op_6 for *Rule 5*.
		Op_4 and Op_5 are prior to Op_7 for *Rule 5*.
F4	Op_6	Op_6 is prior to Op_2 and Op_3 for *Rule 4*.
F6	Op_8	Op_8 is prior to Op_9 for *Rule 3*.
	Op_8, Op_9	Op_8 and Op_8 are prior to Op_7 for *Rule 5*.

Table 3: Definition of a tool change.

Conditions of machining two consecutive operations	Tool change
Same tool and same machine	No
Same tool and different machines	Yes
Different tools and same machine	Yes
Different tools and different machines	Yes

Table 4: Definition of a setup change.

Conditions of machining two consecutive operations	Setup change
Same TAD and same machine	No
Same TAD and different machines	Yes
Different TADs and same machine	Yes
Different TADs and different machines	Yes

where n is the total number of operations and MC_i is the machine cost of the ith machine for an operation, a constant for a specific machine.

(2) Total tool cost (TTC) is

$$TTC = \sum_{i=1}^{n} TC_i, \tag{2}$$

where TC_i is the tool cost of the ith tool for an operation, a constant for a specific tool.

(3) Total machine change cost (TMCC): a machine change is needed when two adjacent operations are executed on different machines

$$TMCC = MCC * NMC, \tag{3}$$

where MCC is the machine change cost and NMC is the number of machine changes, which can be calculated by

$$NMC = \sum_{i=1}^{n-1} \Omega_1 \left(M_{i+1}, M_i\right), \tag{4}$$

where M_i is the machine for the ith operation and $\Omega_1(x, y)$ is a judging function:

$$\Omega_1 (x, y) = \begin{cases} 1 & x \neq y, \\ 0 & x = y. \end{cases} \tag{5}$$

(4) Total tool change cost (TTCC): a tool change is defined in Table 3 [3]

$$TTCC = TCC * NTC, \tag{6}$$

where TCC is the tool change cost and NTC is the number of tool changes, which can be calculated by

$$NTC = \sum_{i=1}^{n-1} \Omega_2 \left(\Omega_1 \left(M_{i+1}, M_i\right), \Omega_1 \left(T_{i+1}, T_i\right)\right), \tag{7}$$

where T_i is the ith tool. $\Omega_2(x, y)$ is a judging function:

$$\Omega_2 (x, y) = \begin{cases} 0 & x = y = 0, \\ 1 & \text{otherwise}. \end{cases} \tag{8}$$

(5) Total setup cost (TSCC): a setup change is defined in Table 4 [3]

$$TSCC = SCC * NSC, \tag{9}$$

where SCC is the setup cost and NSC is the number of setups, which can be calculated by

$$NSC = \sum_{i=1}^{n-1} \Omega_2 \left(\Omega_1 \left(M_{i+1}, M_i\right), \Omega_1 \left(TAD_{i+1}, TAD_{ii}\right)\right) + 1, \tag{10}$$

where TAD_i is the ith TAD.

Table 5: Cost indexes for the example part in Figure 2.

MC		TC									MCC	TCC	SCC
M1	M2	T1	T2	T3	T4	T5	T6	T7	T8	T9			
40	10	10	3	7	3	8	10	10	3	8	300	60	20

Table 6: An optimal process plan for the example part in Figure 2.

Operation	Op_1	Op_8	Op_9	Op_4	Op_5	Op_6	Op_7	Op_2	Op_3
Machine	M1	M1	M1	M1	M1	M1	M1	M1	M1
Tool	T1	T8	T9	T4	T5	T6	T7	T2	T3
TAD	$+X$	$+X$	$+X$	$-X$	$-X$	$+Z$	$-Z$	$-Z$	$-Z$
NMC = 0, NCC = 8, NSC = 4. TMC = 360, TTC = 62, TMCC = 0, TTCC = 480, TSCC = 80, TPC = 982.									

(6) Total production cost (TPC) is

$$\mathrm{TPC} = w_1 * \mathrm{TMC} + w_2 * \mathrm{TTC} + w_3 * \mathrm{TMCC} + w_4 * \mathrm{TTCC} + w_5 * \mathrm{TSCC}. \tag{11}$$

In (11), TPC is total production cost. w_1, w_2, w_3, w_4, and w_5 are weights of TMC, TTC, TMCC, TTCC, andTSCC, respectively. These weights can be assigned referring to the active situations, which provide the flexibility to customize the optimization objective function according to various situations. The different values of w_1, w_2, w_3, w_4, and w_5 constitute the flexible combination to meet the requirement of process planning in different manufacturing environment. The detailed method of setting these parameters is given in the subsequent sections.

4. The Proposed ACO Algorithm

4.1. Graph-Based Representation of Process Plan. The proposed ACO algorithm basically generates solutions by standard ACO procedures [26]. To construct a feasible process plan with the ACO approach, the process planning problem has to be visualized and represented by a weighted directed graph [27].

The weighted graph is denoted by $D = (O, A, B)$, where O is a set of nodes, A is a set of directed arcs, and B is a set of undirected arcs. The nodes of O stand for all of the operations Op_i, and A corresponds to the precedence constraints between the operations of the parts. B represents the set of arcs connecting all possible combination of the nodes. Both A and B represent possible paths for ants travelling from one node to another. The ants are basically free to travel along the paths unless there is a precedence constraint specified by A. Figure 3 is the weighted graph for the example in Figure 2.

The approach in this paper applying the ACO algorithm for process planning is to search for a path in a weighted graph (Figure 3), where all necessary nodes have to be visited to complete the process plan to minimize TPC. The characteristic of this approach is to construct process plans from an autocatalytic process, in which artificial ants favor the process plan with smaller TPC and they will deposit more pheromones on the visited paths so that there is a higher probability for the following ants to continue choosing the better paths.

4.2. Initialization. Before starting the ACO for process planning, the ant colony was placed on the initial node. The selection of the initial node determines which features can be machined firstly, which affects the result of process planning and the performance of ACO. Only these operations attached to the features with no precedent features may be selected as the initial node. For the example part in Figure 2, only F1 has no precedent features, so Op_1 will be allowed to be the first visited node. In fact it is difficult to select the initial node from many operation nodes, because the initial node is not unique in most of the process planning. In this paper a dummy node Op_d, acting as the initial node, is added to the weighted graph to connect the first feasible operations of the parts, as shown in Figure 3. In addition, the undirected arc is added from the initial node to the possibly first visited operation nodes. The number of ants (K) in the colony is arbitrary, and it can be set as a parameter, which is allowed to be adjusted in accordance with the scale of the problem and the performance of the algorithm.

4.3. Iteration. For the ant k, a path will be achieved after traversing all the nodes in a weighted graph, which represents the one of feasible process plans. To choose the next visiting node, the ant k is guided by the heuristic information η_{uv} on the node and the pheromone amount τ_{uv} on the arc linking the source node u and possible destination node v. The heuristic information η_{uv} can reflect the attractiveness of the next visiting node for the ant k. When minimizing TPC is used to be objective function for process planning, MC and TC of the operation node will be treated to calculate η_{uv}. The heuristic information η_{uv} can be given as follows:

$$\eta_{uv} = \frac{E}{\mathrm{PC}}, \tag{12}$$

where E is a positive constant, and it can be set by trial and error. PC is the processing cost of the selected node operation and it is calculated as follows:

$$\mathrm{PC} = w_1 * \mathrm{MC} + w_2 * \mathrm{TC}. \tag{13}$$

Table 7: Features, operations, and machining information of the sample part.

Features	Feature descriptions	Operations	TADs	Machines	Tools
F1	Planar surface	Milling (Op_1)	$+Z$	M2, M3	T6, T7, T8
F2	Planar surface	Milling (Op_2)	$-Z$	M2, M3	T6, T7, T8
F3	Two pockets arranged as a replicated feature	Milling (Op_3)	$+X$	M2, M3	T6, T7, T8
F4	Four holes arranged as a replicated feature	Drilling (Op_4)	$+Z$, $-Z$	M1, M2, M3	T2
F5	A step	Milling (Op_5)	$+X$, $-Z$	M2, M3	T6, T7
F6	A protrusion (rib)	Milling (Op_6)	$+Y$, $-Z$	M2, M3	T7, T8
F7	A boss	Milling (Op_7)	$-a$	M2, M3	T7, T8
F8	A compound hole	Drilling (Op_8)	$-a$	M1, M2, M3	T2, T3, T4
		Reaming (Op_9)		M1, M2, M3	T9
		Boring (Op_{10})		M2, M3	T10
F9	A protrusion (rib)	Milling (Op_{11})	$-Y$, $-Z$	M2, M3	T7, T8
F10	A compound hole	Drilling (Op_{12})	$-Z$	M1, M2, M3	T2, T3, T4
		Reaming (Op_{13})		M1, M2, M3	T9
		Boring (Op_{14})		M3, M4	T10
F11	Nine holes arranged	Drilling (Op_{15})	$-Z$	M1, M2, M3	T1
		Tapping (Op_{16})		M1, M2, M3	T5
F12	A pocket	Milling (Op_{17})	$-X$	M2, M3	T7, T8
F13	A step	Milling (Op_{18})	$-X$, $-Z$	M2, M3	T6, T7
F14	A compound hole	Teaming (Op_{19})	$+Z$	M1, M2, M3	T9
		Boring (Op_{20})		M3, M4	T10

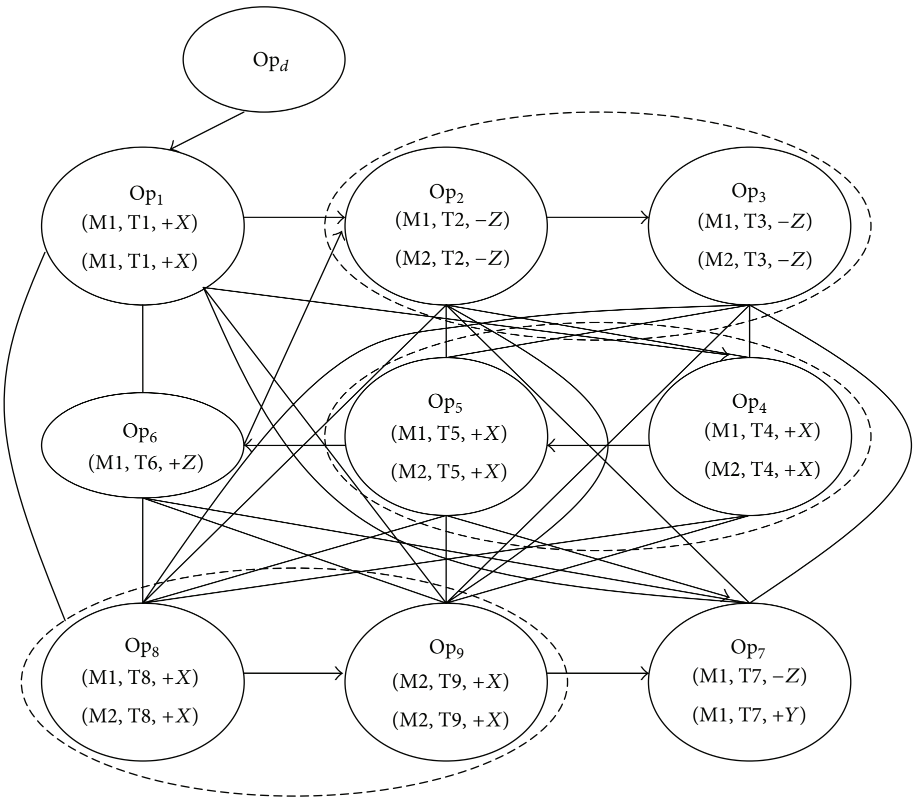

Op_d
Op_1 (M1, T1, $+X$) (M1, T1, $+X$)
Op_2 (M1, T2, $-Z$) (M2, T2, $-Z$)
Op_3 (M1, T3, $-Z$) (M2, T3, $-Z$)
Op_6 (M1, T6, $+Z$)
Op_5 (M1, T5, $+X$) (M2, T5, $+X$)
Op_4 (M1, T4, $+X$) (M2, T4, $+X$)
Op_8 (M1, T8, $+X$) (M2, T8, $+X$)
Op_9 (M2, T9, $+X$) (M2, T9, $+X$)
Op_7 (M1, T7, $-Z$) (M1, T7, $+Y$)

Figure 3: A disjunctive weighted directed graph for the example part.

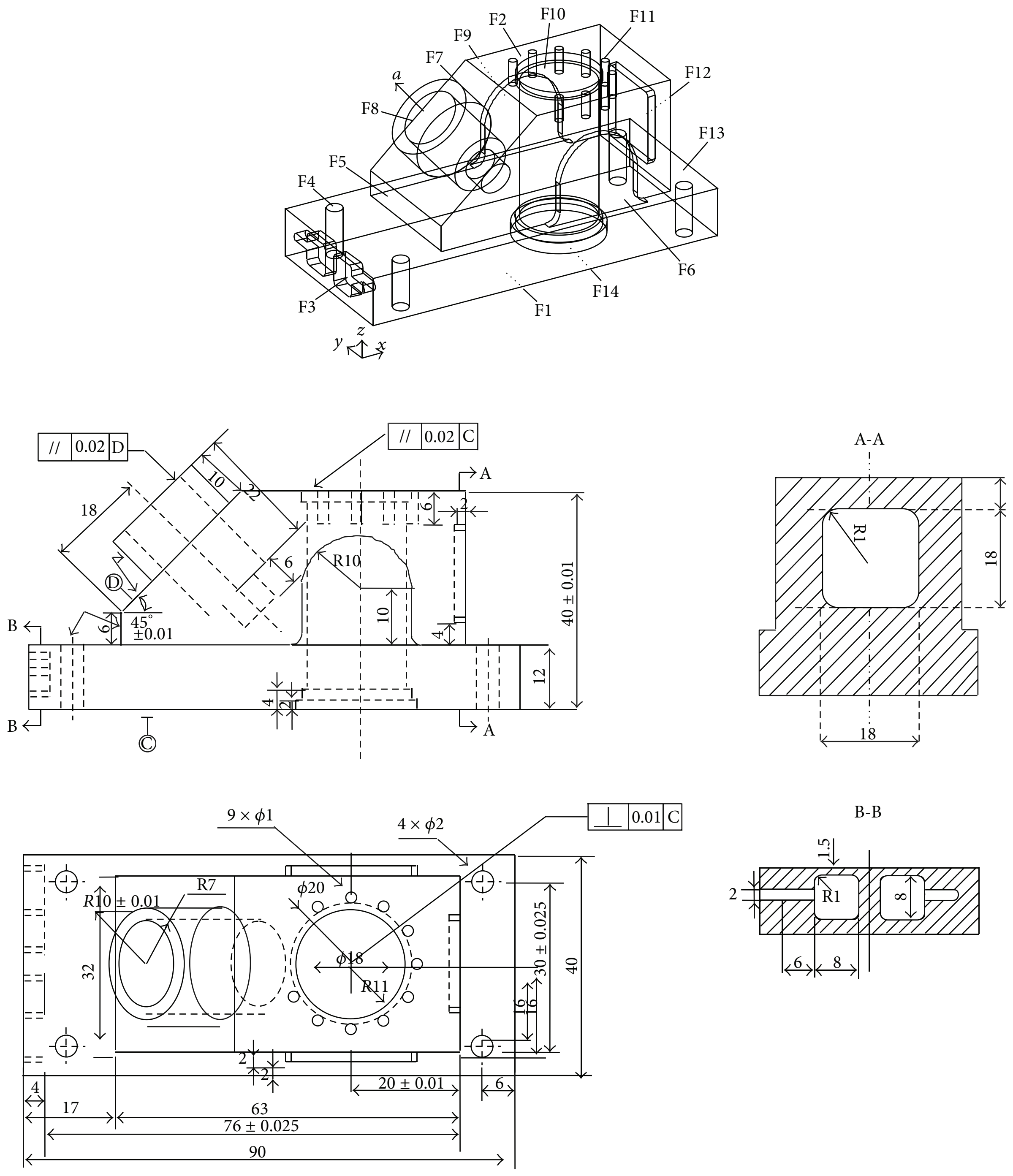


Figure 4: A sample part.

Equation (12) shows that the nodes with the smaller processing cost have the higher heuristic information amount and these nodes have more attraction for the ant k.

The pheromone amount τ_{uv} can reflect the attractiveness of the arc accessing to the destination node from the current node, which specifies how good the previous process plans are for the following ants. It will be updated according to the value of TPC of the process plan achieved by the ant k. The pheromone amount τ_{uv} can be given as follows:

$$\tau_{uv}^{k} = (1-\rho) * \tau_{uv}^{k} + \Delta\tau_{uv}^{k}, \qquad (14)$$

where ρ is an evaporation coefficient of the pheromone on the arc linking the source node u and possible destination

Table 8: Available machining resources and costs in a workshop environment.

Number	Types	MC
Machines		
M1	Drilling press	10
M2	Three-axis vertical milling machine	40
M3	CNC 3-axis vertical milling machine	100
M4	Boring machine	60
Number	Types	TC
Tools		
T1	Drill 1	7
T2	Drill 2	5
T3	Drill 3	3
T4	Drill 4	8
T5	Tapping tool	7
T6	Mill 1	10
T7	Mill 2	15
T8	Mill 2	30
T9	Ream	15
T10	Boring tool	20
MCC = 160, SCC = 100, TCC = 20		

node v. $\Delta\tau_{uv}^{k}$ is the quantity of the pheromone trail on the arc(u, v) generated by the ant k after each iteration. Also, it can be given as

$$\Delta\tau_{uv}^{k} = \begin{cases} \dfrac{Q}{\text{TPC}} & \text{if ant } k \text{ passes the arc } (u, v), \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

where Q is a positive constant. Before ant colony begins the iteration, the pheromone amount on every arc is set to be τ_0 initially.

The heuristic information and the pheromone amount constructed a probability of moving from a node to another node for an ant. The more the pheromone amount on the arc and the heuristic information on the node, the higher the selective probability. For the ant k, the selective probability p_{uv}^{k} from the source node u to the destination node v can be given as follows:

$$p_{uv}^{k} = \begin{cases} \dfrac{[\tau_{uv}]^{\alpha}[\eta_{uv}^{k}]^{\beta}}{\sum_{w \in S_k}[\tau_{uw}]^{\alpha}[\eta_{uw}^{k}]^{\beta}} & v \in S_k, \\ 0 & v \notin S_k, \end{cases} \tag{16}$$

where α and β denote the weighting parameters controlling the relative importance of the pheromone amount and the heuristic information, respectively. S_k represents the set of nodes allowed to be visited at the next step for the ant k.

4.4. Termination. If all of the ants almost constructed the same process plans repeatedly at the early stage of the ACO algorithm, the algorithm would fall into the local convergence, which leads to failure in the exploration of new paths for the subsequent iteration. Once the algorithm falls into the local convergence, the output of process planning would not be the optimal result, even far from the optimal results. To void the local convergence, the parameter of M_{rpt} controlling the repeated number of the same process plan is set in advance. When the adjacent two-process plan is completely the same, the variable of S_{rpt} will increase by 1; otherwise S_{rpt} will be reset to be 0. When S_{rpt} reaches to M_{rpt}, it means that no improvement on the solutions is made in the recent iterations. The ants may have converged to local optimal results. In addition, the local convergence occurs at the early stage of the ACO algorithm. To prevent the quick convergence, the maximum iteration M_{ite} is set in advance. Obviously, with the number of iterations S_{ite} increasing, even approaching to the M_{ite}, the M_{rpt} will increase and can be calculated as follows:

$$M_{rpt} = S_{ite} * q * \frac{S_{ite}}{M_{ite}}, \tag{17}$$

where q is random number, $q \in (0, 1)$.

If the two events of $S_{rpt} = M_{rpt}$ and $S_{ite} < M_{ite}$ are satisfied simultaneously, it is considered that the local convergence occurs and the algorithm will be restarted. If the only event of $S_{ite} = M_{ite}$ is satisfied, the resulting process plan will be output and algorithm will be terminated.

5. Experiments and Results

5.1. Walkthrough Example. When ACO is applied in process planning, those parameters including K, ρ, α, β, E, Q, τ_0 have to be adjusted according to the situation to achieve the optimal process plan. The example part in Figure 2 is used to illustrate the proposed ACO approach. All the cost indexes are shown in Table 5 and it is assumed that all the machines and tools are available; namely, w_1–w_5 in (11) and (13) are set as 1.

A lot of preliminary experiments are dominated to test the effect of various parameters. In each experiment, one parameter is changed and the other parameters were fixed, and the effect of the changed parameter on the algorithm properties was analyzed at different levels. The resulting process plan is shown in Table 6 by the proposed ACO approach at the value of $K = 5$, $\rho = 0.8$, $\alpha = 2$, $\beta = 1$, $E = 45$, $Q = 1000$, $\tau_0 = 1$, $M_{ite} = 50$.

5.2. Simulation Experiments. More complex process planning problems are considered in extensive simulation experiments. A sample part taken from the work of Li et al. [3, 4] is used to test the proposed ACO approach (Figure 4). The part consists of 14 defined manufacturing features, including planes, holes, and pockets. The detailed information of features, operations, manufacturing resources, and precedence relationship of the part is given in Tables 7, 8, and 9.

The above simulation experiment for the example part in Figure 2 shows that the selection of parameters is very important to the quality of the results. For the sample example in Figure 4, the method of determining those parameters is more complex, due to the enlargement of the problem size. It is assumed that all the machines and tools are available; namely, w_1–w_5 in (11) and (13) are set as 1.

Table 9: Precedence relationship between features and operations.

Features	Operation	Precedence constraints description
F1	Milling (Op_1)	F1 (Op_1) is the datum face for the part; hence, it is machined before all features
F2	Milling (Op_2)	F2 (Op_2) is before F10 (Op_{12}, Op_{13}, Op_{14}) and F11 (Op_{15}, Op_{16}) for *Rule 2*
F3	Milling (Op_3)	
F4	Drilling (Op_4)	
F5	Milling (Op_5)	F5 (Op_5) is before F4 (Op_4) and F7 (Op_7) for *Rule 4*
F6	Milling (Op_6)	F6 (Op_6) is before F10 (Op_{12}, Op_{13}, Op_{14}) for *Rule 4*
F7	Milling (Op_7)	F7 (Op_7) is before F8 (Op_8, Op_9, Op_{10}) for *Rule 4*
F8	Drilling (Op_8) Reaming (Op_9) Boring (Op_{10})	Op_8 is before (Op_9 and Op_{10}); Op_9 is before Op_{10} for *Rule 3*
F9	Milling (Op_{11})	F9 (Op_{11}) is before F10 (Op_{12}, Op_{13}, Op_{14}) for *Rule 4*
F10	Drilling (Op_{12}) Reaming (Op_{13}) Boring (Op_{14})	Op_{12} is before Op_{13} and Op_{14}; Op_{13} is before Op_{14} for *Rule 3*; F10 (Op_{12}, Op_{13}, Op_{14}) is before F11 (Op_{15}, Op_{16}) for *Rule 4*;Op_{12} of F10 is before F14 (Op_{19}, Op_{20})
F11	Drilling (Op_{15}) Tapping (Op_{16})	Op_{15} is before Op_{16} for *Rule 3*
F12	Milling (Op_{17})	
F13	Milling (Op_{18})	F13 (Op_{18}) is before Op_4 and Op_{17} for *Rule 2* and *Rule 1,* respectively
F14	Reaming (Op_{19}) Boring (Op_{20})	Op_{19} is before Op_{20} for *Rule 3*

Table 10: Four of the fifty process plans.

Process plan 1																				
Operation	1	2	18	11	6	12	13	19	17	3	5	7	8	9	10	20	14	4	15	16
Machine	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4	4	1	1	1
Tool	7	7	7	7	7	3	9	9	7	7	7	7	3	9	10	10	10	2	1	5
TAD	$+Z$	$-Z$	$-Z$	$-Z$	$-Z$	$-Z$	$-Z$	$+Z$	$-X$	$+X$	$+X$	$-a$	$-a$	$-a$	$-a$	$+Z$	$-Z$	$-Z$	$-Z$	$-Z$
NMC = 2, NTC = 10, NSC = 9, TMCC = 320, TTCC = 200, TSCC = 900, TMC = 750, TTC = 265, TPC = 2435																				
Process plan 2																				
Operation	1	11	6	2	12	18	13	19	17	3	5	7	8	9	10	20	14	15	16	4
Machine	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4	4	1	1	1
Tool	7	7	7	7	3	6	9	9	7	7	7	7	3	9	10	10	10	1	5	2
TAD	$+Z$	$-Z$	$-Z$	$-Z$	$-Z$	$-Z$	$-Z$	$+Z$	$-X$	$+X$	$+X$	$-a$	$-a$	$-a$	$-a$	$+Z$	$-Z$	$-Z$	$-Z$	$-Z$
NMC = 2, NTC = 11, NSC = 9, TMCC = 320, TTCC = 220, TSCC = 900, TMC = 750, TTC = 260, TPC = 2450																				
Process plan 3																				
Operation	1	5	3	18	6	2	11	12	13	17	7	8	9	19	14	20	10	4	15	16
Machine	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4	4	4	1	1	1
Tool	6	6	6	6	6	6	7	3	9	7	7	2	9	9	10	10	10	2	1	5
TAD	$+Z$	$+X$	$+X$	$-Z$	$-Z$	$-Z$	$-Z$	$-Z$	$-Z$	$-X$	$-a$	$-a$	$-a$	$+Z$	$-Z$	$+Z$	$-a$	$-Z$	$-Z$	$-Z$
NMC = 2, NTC = 9, NSC = 10, TMCC = 320, TTCC = 200, TSCC = 1000, TMC = 770, TTC = 237, TPC =2527																				
Process plan 4																				
Operation	1	3	5	6	2	18	11	12	13	17	7	8	9	19	14	20	10	4	15	16
Machine	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4	4	4	1	1	1
Tool	6	6	6	6	6	6	7	3	9	7	7	2	9	9	10	10	10	2	1	5
TAD	$+Z$	$+X$	$+X$	$-Z$	$-Z$	$-Z$	$-Z$	$-Z$	$-Z$	$-X$	$-a$	$-a$	$-a$	$+Z$	$-Z$	$+Z$	$-a$	$-Z$	$-Z$	$-Z$
NMC = 2, NTC = 9, NSC = 10, TMCC = 320, TTCC = 200, TSCC = 1000, TMC = 770, TTC = 237, TPC = 2527																				

Table 11: Average results of simulation experiment.

Type	Mean	Maximum	Minimum	Standard deviation
TMC	754.2	800	750	9.82
TTC	261.88	267	237	7.63
TMCC	320	320	320	320
TTCC	202	220	180	10.77
TSCC	918	1000	900	38.42
TPC	2456.1	2527.0	2435.0	37.98

Table 12: Results compared to other algorithms for the sample part in Figure 4.

Condition	Proposed approach	ACO	TS	SA	GA
(1)					
Mean	2456.1	2490.0	2609.6	2668.5	2796.0
Maximum	2527.0	2500.0	2690.0	2829.0	2885.0
Minimum	2435.0	2450.0	2527.0	2535.0	2667.0
(2)					
Mean	2115.4	2117.0	2208.0	2287.0	2370.0
Maximum	2380.0	2120.0	2390.0	2380.0	2580.0
Minimum	2090.0	2090.0	2120.0	2120.0	2220.0
(3)					
Mean	2600	2600.0	2630.0	2630.0	2705.0
Maximum	2740.0	2600.0	2740.0	2740.0	2840.0
Minimum	2580.0	2600.0	2580.0	2590.0	2600.0

The sample example is solved by the ACO approach with the varied values of $K \in \{5, 10, 20, 40\}$, $\rho \in \{0.05, 0.1, 0.25, 0.5, 0.8\}$, $\alpha \in \{0.5, 1, 5, 10\}$, $\beta \in \{0.5, 1, 5, 10\}$, $E \in \{50, 55, 65, 80\}$, $Q \in \{1500, 2000, 2500, 3000\}$, and $M_{ite} \in \{100, 2000, 300, 400\}$ and with the fixed value of $\tau_0 = 1$. 50 trials were separately conducted to evaluate the performance of the proposed approach. Experimental observation has shown that $K = 10$, $\rho = 0.8$, $\alpha = 1$, $\beta = 1$, $E = 80$, $Q = 3000$, $\tau_0 = 1$, and $M_{ite} = 200$ are the best choices of these parameters. Four of the process plans generated are listed in Table 10. The best process plan (minimal TPC) is shown as process plan 1 in Table 10. The average result of 50 trials is shown in Table 11.

5.3. Comparative Tests. Three conditions are used to test the proposed algorithm for the sample parts [3, 4].

(1) All machines and tools are available, and w_1–w_5 in (11) and (13) are set as 1.

(2) All machines and tools are available, and $w_2 = w_5 = 0$, $w_1 = w_3 = w_4 = 1$.

(3) Machine M2 and tool T7 are down, $w_2 = w_5 = 0$, $w_1 = w_3 = w_4 = 1$.

In Table 12, the TPC generated by the proposed ACO is compared with those of GA and SA approaches by Li et al. [3] and TS by Li et al. [4], as well as the ACO by Liu et al. [18].

Under condition (1), a lower TPC (2435.0) has been found using the proposed ACO approach, and the mean TPC (2456.1) is better than the costs of the other four algorithms. Under condition (2), the minimum TPC (2090) is the same as the ACO [6]. Under condition (3), the minimum TPC (2580) is the same as the TS [4]. The mean TPC generated by the proposed approach is better than the other four algorithms under the three conditions.

6. Conclusions

A graph-based ACO approach is developed to solve the process planning optimization problem against process constraints for prismatic parts, which considers the selection of machine resources, determining process operation, and sequencing operation according to machine cost. The approach is characterized by the following aspects.

(1) A graph-based representation of process plan is proposed. A weighted directed graph is used to represent process planning problem. The graph includes nodes set, directed arcs set, and undirected arcs set, which stand for operations, precedence constraints between the operations, and possible visited path connecting the nodes, respectively.

(2) A lower TPC is found by the proposed approach for the sample part, which means that the optimal process plan is generated by now under the same conditions. Comparing with the other algorithms, the proposed approach has generated the better process plan results under the three conditions.

In the further study, a deep discussion of selecting the ACO approach parameters is conducted. In addition, the multiobjective optimization will be incorporated into the ACO approach for handling the multiobjective process planning problem.

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 Fundamental Research Funds for the Central Universities (13MS100 and 14ZD37) and the National Natural Science Foundation of China (51177046).

References

[1] J. M. Usher and R. O. Bowden, "The application of genetic algorithms to operation sequencing for use in computer-aided process planning," *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 999–1013, 1996.

[2] F. Cay and C. Chassapis, "An IT view on perspectives of computer aided process planning research," *Computers in Industry*, vol. 34, no. 3, pp. 307–337, 1997.

[3] W. D. Li, S. K. Ong, and A. Y. C. Nee, "Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts," *International Journal of Production Research*, vol. 40, no. 8, pp. 1899–1922, 2002.

[4] W. D. Li, S. K. Ong, and A. Y. C. Nee, "Optimization of process plans using a constraint-based tabu search approach," *International Journal of Production Research*, vol. 42, no. 10, pp. 1955–1985, 2004.

[5] J. M. Usher and G. C. Sharma, "Intelligent reasoning in the generation of alternative sequences for feature-based process planning," *Intelligent Automation and Soft Computing*, vol. 3, no. 3, pp. 207–220, 1997.

[6] Y. Tseng and C. Liu, "Concurrent analysis of machining sequences and fixturing set-ups for minimizing set-up changes for machining mill-turn parts," *International Journal of Production Research*, vol. 39, no. 18, pp. 4197–4214, 2001.

[7] H. Zhang, "A hybrid-graph approach for automated setup planning in CAPP," *Robotics and Computer-Integrated Manufacturing*, vol. 15, no. 1, pp. 89–100, 1999.

[8] W. Huang, Y. Hu, and L. Cai, "An effective hybrid graph and genetic algorithm approach to process planning optimization for prismatic parts," *International Journal of Advanced Manufacturing Technology*, vol. 62, no. 9–12, pp. 1219–1232, 2012.

[9] J. Váncza and A. Márkus, "Genetic algorithms in process planning," *Computers in Industry*, vol. 17, no. 2-3, pp. 181–194, 1991.

[10] F. Zhang, Y. F. Zhang, and A. Y. C. Nee, "Using genetic algorithms in process planning for job shop machining," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 4, pp. 278–289, 1997.

[11] Z. W. Bo, L. Z. Hua, and Z. G. Yu, "Optimization of process route by genetic algorithms," *Robotics and Computer-Integrated Manufacturing*, vol. 22, no. 2, pp. 180–188, 2006.

[12] K. Lian, C. Zhang, X. Shao, and Y. Zeng, "A multi-dimensional tabu search algorithm for the optimization of process planning," *Science China Technological Sciences*, vol. 54, no. 12, pp. 3211–3219, 2011.

[13] G. H. Ma, Y. F. Zhang, and A. Y. C. Nee, "A simulated annealing-based optimization algorithm for process planning," *International Journal of Production Research*, vol. 38, no. 12, pp. 2671–2687, 2000.

[14] Y. W. Guo, A. R. Mileham, G. W. Owen, and W. D. Li, "Operation sequencing optimization using a particle swarm optimization approach," *Proceedings of the Institution of Mechanical Engineers B: Journal of Engineering Manufacture*, vol. 220, no. 12, pp. 1945–1958, 2006.

[15] X. Li, L. Gao, and X. Wen, "Application of an efficient modified particle swarm optimization algorithm for process planning," *International Journal of Advanced Manufacturing Technology*, vol. 67, no. 5–8, pp. 1355–1369, 2013.

[16] S. Sette, L. Boullart, and L. van Langenhove, "Optimising a production process by a neural network/genetic algorithm approach," *Engineering Applications of Artificial Intelligence*, vol. 9, no. 6, pp. 681–689, 1996.

[17] A. G. Krishna and K. Mallikarjuna Rao, "Optimisation of operations sequence in CAPP using an ant colony algorithm," *International Journal of Advanced Manufacturing Technology*, vol. 29, no. 1-2, pp. 159–164, 2006.

[18] X. Liu, H. Yi, and Z. Ni, "Application of ant colony optimization algorithm in process planning optimization," *Journal of Intelligent Manufacturing*, vol. 24, no. 1, pp. 1–13, 2013.

[19] F. T. S. Chan, R. Swarnkar, and M. K. Tiwari, "Fuzzy goal-programming model with an artificial immune system (AIS) approach for a machine tool selection and operation allocation problem in a flexible manufacturing system," *International Journal of Production Research*, vol. 43, no. 19, pp. 4147–4163, 2005.

[20] L. Ding, Y. Yue, K. Ahmet, M. Jackson, and R. Parkin, "Global optimization of a feature-based process sequence using GA and ANN techniques," *International Journal of Production Research*, vol. 43, no. 15, pp. 3247–3272, 2005.

[21] Y. F. Wang, Y. F. Zhang, and J. Y. H. Fuh, "A hybrid particle swarm based method for process planning optimisation," *International Journal of Production Research*, vol. 50, no. 1, pp. 277–292, 2012.

[22] X. Xu, L. Wang, and S. T. Newman, "Computer-aided process planning: a critical review of recent developments and future trends," *International Journal of Computer-Integrated Manufacturing*, vol. 24, no. 1, pp. 1–31, 2011.

[23] J. M. Usher, "An object-oriented approach to product modeling for manufacturing systems," *Computers and Industrial Engineering*, vol. 25, no. 1–4, pp. 557–560, 1993.

[24] S. K. Ong, W. D. Li, and A. Y. C. Nee, "STEP-based integration of feature recognition and design-by-feature for manufacturing applications in a concurrent engineering environment," *International Journal of Computer Applications in Technology*, vol. 18, no. 1–4, pp. 78–92, 2003.

[25] W. Faheem, J. F. Castano, C. C. Hayes, and D. M. Gaines, "What is a manufacturing interaction?" in *Proceedings of the 1998 ASME Design Engineering Technical Conferences*, pp. 1–6, Atlanta, Ga, USA, 1998.

[26] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[27] Y. K. Kim, K. Park, and J. Ko, "A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling," *Computers and Operations Research*, vol. 30, no. 8, pp. 1151–1171, 2003.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 973093, 12 pages
http://dx.doi.org/10.1155/2014/973093

Research Article

The Contribution of Particle Swarm Optimization to Three-Dimensional Slope Stability Analysis

Roohollah Kalatehjari,[1] Ahmad Safuan A Rashid,[1] Nazri Ali,[1] and Mohsen Hajihassani[2]

[1] *Department of Geotechnics and Transportation, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia*
[2] *Construction Research Alliance, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia*

Correspondence should be addressed to Ahmad Safuan A Rashid; ahmadsafuanutm@gmail.com

Received 21 February 2014; Accepted 8 April 2014; Published 1 June 2014

Academic Editors: V. Bhatnagar and Y. Zhang

Copyright © 2014 Roohollah Kalatehjari 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.

Over the last few years, particle swarm optimization (PSO) has been extensively applied in various geotechnical engineering including slope stability analysis. However, this contribution was limited to two-dimensional (2D) slope stability analysis. This paper applied PSO in three-dimensional (3D) slope stability problem to determine the critical slip surface (CSS) of soil slopes. A detailed description of adopted PSO was presented to provide a good basis for more contribution of this technique to the field of 3D slope stability problems. A general rotating ellipsoid shape was introduced as the specific particle for 3D slope stability analysis. A detailed sensitivity analysis was designed and performed to find the optimum values of parameters of PSO. Example problems were used to evaluate the applicability of PSO in determining the CSS of 3D slopes. The first example presented a comparison between the results of PSO and PLAXI-3D finite element software and the second example compared the ability of PSO to determine the CSS of 3D slopes with other optimization methods from the literature. The results demonstrated the efficiency and effectiveness of PSO in determining the CSS of 3D soil slopes.

1. Introduction

Slope stability analysis is a major concern in projects related to man-made or natural slopes. Several techniques are applied to analyze the stability state of a slope, of which limit equilibrium method (LEM) is the most popular [1]. This method undertakes the static behavior of the slope at the verge of failure and develops equilibriums of the soil body in static condition. Consequently, no stress-strain relationship is considered and corresponding deformation within the soil body is not studied [2]. As a result, the shape of each potential slip surface which defines the lower boundary of sliding body has to be assumed. A numerical ratio as factor of safety (FOS) is used to determine the critical slip surface (CSS) as the least stable slip surface among all potentials. FOS compares the available shear strength of the soil with the existing shear stress (mobilized shear strength) on the assumed slip surface as follows [3]:

$$\mathrm{FOS} = \frac{S}{T}, \tag{1}$$

where S is mobilized shear strength force (kN) and T is available shear strength of the soil (kN). The mobilized shear strength force is defined as

$$S = \left(\frac{\left[c' + (\sigma_n - u_w)\tan\phi'\right]}{\mathrm{FOS}} \right), \tag{2}$$

where FOS is factor of safety, S is mobilized shear strength force (kN), c' is cohesion of the soil in terms of effective stress (kN/m^2), ϕ' is angle of internal friction of soil in terms of effective stress (kN/m^2), and σ_n is normal stress on the slip surface (kN/m^2), and u_w is pore water pressure on the slip surface (kN/m^2). In general, the following principles are required to analyze the stability of a slope within LEM [2].

(1) A kinematically admissible slip surface is assumed to define the mechanism of failure.

(2) Two static principles as the assumption of plastic behavior for soil mass and validity of Mohr-coulomb failure criterion are employed to determine the shearing strength along the assumed slip surface.

(3) Equation of FOS is developed for the assumed slip surface by dividing the available shear strength at the surface by the required shear resistance to bring the equilibrium into limiting condition.

(4) An iterative process is used to find the satisfying value of FOS.

(5) By using the steps above, a search technique is employed to find the CSS among all assumed slip surfaces.

Although all the LEMs have mutual principles, they differ in utilizing static equilibrium, assumptions, and simplifications. They can be considered as two-dimensional (2D) and three-dimensional (3D) methods. 2D methods simplify the geometry of slopes by transforming the problem into an assumed 2D form. Consequently, some internal and external forces are simplified or ignored in this process. Such simplifications in 2D methods may result in different outcomes form the results of 3D methods. Although the assumptions of 3D methods are mostly derived from the related 2D basics, some new definitions are only available in 3D methods due to plus one dimension that 3D methods have. Ability to consider 3D shapes of slip surface, asymmetric and complex slopes, sliding direction, and intercolumn forces are some of the privileges of 3D methods. However, 3D methods might consider, simplify, or ignore any of these aspects.

Determining the CSS, despite of utilized 2D or 3D method, needs a massive search among possible slip surface. Searching problem is usually defined as optimization problem in engineering. This problem is framed to find appropriate solution among the candidates by minimizing or maximizing an objective function. If more than one solution exists among candidates of a problem, it turns to global optimization. Global optimization methods try to find the global solution, while avoiding local solutions.

Particle swarm optimization (PSO) was initially introduced by Kennedy and Eberhart [4] as a global optimization technique. PSO simulates the birds flock activities when they randomly search for food in their path. Since PSO has been released, its successful application in various engineering problems has begun. The popularity of PSO is mainly due to its comprehensible performance as well as its simple operation [5]. Many researchers applied PSO to solve their problems in the fields of structural [6–8], environmental [9–11], hydrological [12, 13], and geotechnical [14–16] engineering.

Cheng et al. [17] tried to determine the CSS of seven slopes by using PSO as one of the first applications of PSO in slope stability analysis and came to the conclusion that PSO produces appropriate and reasonable results. Furthermore, in a comparison with pattern method, they [17] reported that PSO is capable of finding the global minimum FOS and its related CSS in different slopes. Ever since, PSO has been used progressively as an effective technique to deal with the problem of determining the CSS, to name a few, Cheng et al. [17], Cheng et al. [18], Zhao et al. [19], Tian et al. [20], Li et al. [21], Kalatehjari et al. [22, 23], and W. Chen and P. Chen [24]. However, the contribution of PSO was limited to 2D slope stability problem. In fact, only a few researchers published their results in determining the CSS in 3D slope stability problems and none of them applied PSO [25–30].

Based on the successful performance of PSO in 2D slope stability analysis as well as other problems of geotechnical engineering, it is believed that it can contribute well to determining the CSS of 3D slopes. This paper applies PSO in 3D slope stability problem to determine the CSS of soil slopes. A detailed description of adopted PSO is presented to provide a good basis for more contribution of this technique to the field of 3D slope stability problems.

2. Overview of Particle Swarm Optimization

Kennedy and Eberhart [4] initialized PSO by simulating the behavior of a birds swarm with defined instructions for individual behaviors as well as intercommunications. These instructions help in decision making process of individuals which is based on the following items [4]:

(i) experience of individual as its best results so far;

(ii) outlay of experience of swarm as the best result among all individuals.

Swarm intelligence as the ability of each individual to use the experience of others guides the swarm toward its optimum goal. Three principals of the swarm behavior in PSO were similar to what described by Reynolds [31].

(i) Individuals are collision-proof.

(ii) Individuals travel toward swarm objective.

(iii) Individuals travel to the center of swarm.

The standard flowchart of PSO is shown in Figure 1. This process starts by randomly generating a certain number of individuals, namely particles, where each represents a possible solution for the problem [4, 17]. The structure of a particle may contain three sections that separately record its current position, best position so far, and velocity, respectively, as coordinates of current position, coordinates of best position so far, and velocity vectors in a D-dimensional space, where D starts from one [32]. Consequently, a $3 \times D$-dimensional particle is fitting for a particle in D-dimensional space.

PSO reaches its goal if meets the termination criteria. These criteria are set to guarantee the ending of iterative search process. Appropriate termination criteria are necessary to accomplish a successful search by avoiding premature or late convergence [33]. The commonly used termination criteria are set as follows:

(i) reaching a maximum number of iterations;

(ii) finding a satisfactory solution;

(iii) achieving a constant fitness for a certain number of iterations.

The closeness of each particle to the best possible solution is defined by the objective function which is aimed to be minimized or maximized by PSO. A fitness function related to the objective function is usually set to calculate fitness value of each particle by assessing its current position. The

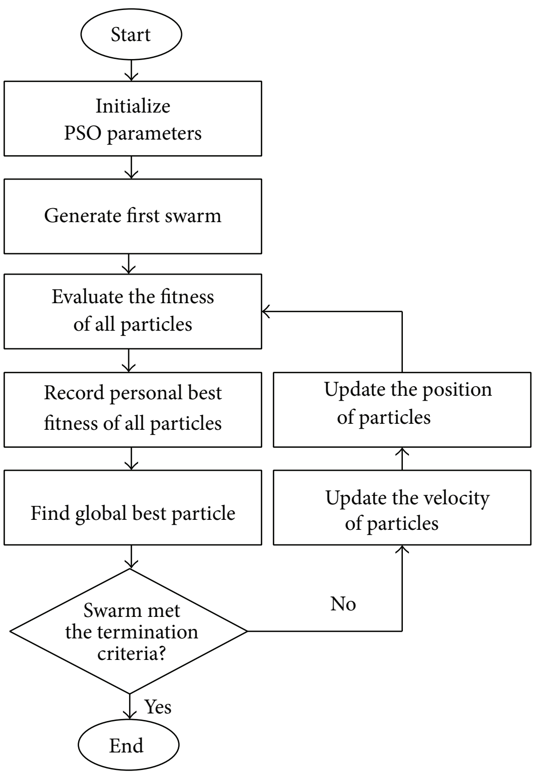

Start
Initialize PSO parameters
Generate first swarm
Evaluate the fitness of all particles
Record personal best fitness of all particles
Find global best particle
Swarm met the termination criteria?
No
Yes
End
Update the position of particles
Update the velocity of particles

Figure 1: Standard flowchart of PSO.

velocity of particles is determined by (3) based on their best position and global best position in the swarm. To continue the search, (4) updates the position of all particles based on their current position and the obtained velocity. Through an iterative process, the improvement of fitness of particles continues until PSO meets the termination criteria. The global solution is then achieved by the current position of the best particle in the last iteration:

$$\begin{aligned} v_{n(i)} = v_{n(i-1)} &+ u(0,\vartheta_1)\left(bp_{n(i)} - x_{n(i)}\right) \\ &+ u(0,\vartheta_2)\left(bg_{n(i)} - x_{n(i)}\right), \end{aligned} \quad (3)$$

$$x_{n(i+1)} = x_{n(i)} + v_{n(i)}, \quad (4)$$

where $v_{n(i-1)}$ and $v_{n(i)}$ are, respectively, the velocity of nth particle in past and current iterations, $u(0,\vartheta_1)$ and $u(0,\vartheta_2)$ are the vectors of random numbers of nth particle uniformly distributed, respectively, in $[0,\vartheta_1]$ and $[0,\vartheta_2]$, $bp_{n(i)}$ is the best position of nth particle so far, $bg_{n(i)}$ is the position of the best particle of the swarm so far, and $x_{n(i-1)}$ and $x_{n(i)}$ are the positions of nth particle, respectively, in the current and the next iterations.

Initial, cognitive, and social parts are three components of velocity equation. The values of ϑ_1 and ϑ_2 in this equation control the exploration and exploitation behaviours of the swarm. While equal values of 2 are commonly used for these parameters in early search, greater values of ϑ_1 and ϑ_2, respectively, provide faster convergence to the solution and enhance discovering the searching space. The velocity of particles may increase surprisingly by adjusting these parameters, so a limiting bound of velocity as $[-v_{\max}, v_{\max}]$ is attached to PSO as constriction coefficients [34]. Shi and Eberhart [35] modified the original equation of velocity to reduce the role of constriction coefficient and introduced (5) by introducing ω as the inertia weight of particles. Later on, Clerc and Kennedy [36] demonstrated that inertia weights of greater than one may cause converge problems in PSO and proposed (6) by introducing ξ as the constant multiplier in (7). This modification prevents the swarm to explode, guarantees the mature converge, and almost eliminates the need of constriction coefficient. Principally, interior parameters (inertia weight and velocity coefficient) and exterior parameters (swarm size and topology) of PSO should be carefully adjusted to provide the best results:

$$\begin{aligned} v_{n(i)} = \omega v_{n(i-1)} &+ u(0,\vartheta_1)\left(bp_{n(i)} - x_{n(i)}\right) \\ &+ u(0,\vartheta_2)\left(bg_{n(i)} - x_{n(i)}\right), \end{aligned} \quad (5)$$

$$\begin{aligned} v_{n(i)} = \xi \Big[v_{n(i-1)} &+ u(0,\vartheta_1)\left(bp_{n(i)} - x_{n(i)}\right) \\ &+ u(0,\vartheta_2)\left(bg_{n(i)} - x_{n(i)}\right)\Big], \end{aligned} \quad (6)$$

$$\xi = \frac{2}{(\vartheta_1 + \vartheta_1) - 2 + \sqrt{(\vartheta_1 + \vartheta_1)^2 - 4(\vartheta_1 + \vartheta_1)}}, \quad (\vartheta_1 + \vartheta_1) > 4. \quad (7)$$

Topology in the method of intercommunication between particles controls the convergence of a swarm. Topology is divided into static and dynamic categories. In static topologies, the number of connected neighbors to a particle is constant throughout the optimization process. However, this number increases by the progress of optimization process in dynamic topologies to enhance the searching abilities [37].

The original PSO aided a conical static topology based on intercommunication of all particles with the global best particle. However, Eberhart and Kennedy [38] proposed another static topology by introducing intercommunication between individuals and local best particles. In this model, each particle was connected to K number of its neighbors in the swarm array. The main advantage of this method was the ability of subconvergence in different regions of the search space. Although the convergence of this method was slower than the conical method, it was able to better escape from local optima. For each problem, the appropriate topology can be defined by performing sensitivity analysis on convergence and execution time of PSO. Figure 2 illustrates conical and local ($K = 2$) topologies for randomly generated 100 particles in a 2D search space.

The size of swarm is defined as the number of its particles. While a small swarm may fail to converge over a global solution, a large swarm may have late convergence. The size of swarm commonly varies from 20 to 50, but the optimum number is usually determined through sensitivity analysis on the convergence parameter of the swarm [36].

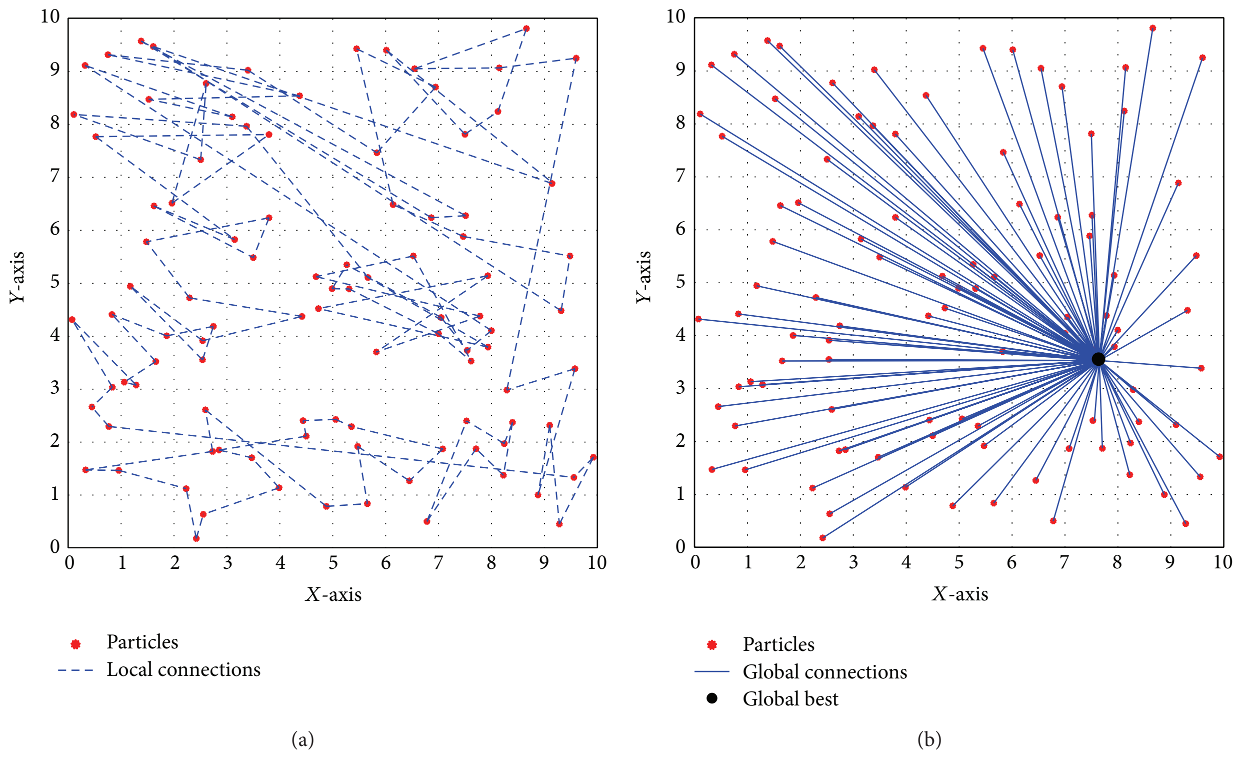


Figure 2: (a) Local and (b) global topologies in 2D search space.

3. Application of PSO in Slope Stability Analysis

PSO is mainly applied in stability analysis of soil slopes within the framework of LEM [1]. This analysis involves two consequent steps, that is, calculating FOS of candidate slip surfaces and determining the CSS among all candidates [39]. PSO commonly contributes to the second step to determine the shape and location of the CSS which are generally unknown in soil slopes [40].

PSO can be applied in both 2D and 3D slope stability analyses [32]. In 2D analysis, it can be employed to determine the shape of the CSS in a predefined 2D section of the slope. Different shapes are possible for slip surfaces in 2D analysis, such as circular, ellipse, spiral, and polygonal or arbitrary surfaces [22, 41, 42]. In contrast, 3D slip surfaces such as spherical, ellipsoidal, and Nonuniform Rational B-Splines (NURBS) are commonly assumed in 3D analysis [23, 43, 43, 44].

Figure 3 shows flowchart of PSO to determine the CSS in slope stability analysis. The optimization procedure is started by setting initial parameters of PSO. Then, a certain number of particles (N) is generated in a random pattern over the search space. Since the improvement of swarm has just begun, personal bests of all particles in initial swarm are identical to the particles themselves. Based on the same reason, the velocity of all initial particles is set to zero. After setting up the initial values, the first particle is arranged as its corresponding slip surface. This surface is qualified if it can satisfy the conditions of the problem. Otherwise, it is disqualified. A predefined minimum fitness value is given to disqualify slip surfaces. This value represents a predefined maximum FOS. For a qualified surface, FOS is calculated and the corresponding fitness values of particle are assigned by the fitness function of PSO. This process is repeated for all particles of the swarm.

Current positions of particles that improved their fitness values are recorded to update their personal bests, while previous personal bests are used for other particles. The global best particle is defined by the greatest fitness value in the current swarm. Through an iterative process, subsequent swarms are generated by updating velocities and positions of former particles. The optimization process is terminated by meeting the termination criteria. Eventually, the global best particle of the last swarm represents the CSS of the slope.

3.1. Coding of the Particles.

The structure of particles followed the standard PSO particles involving three sections as current position, previous best position, and the velocity. A rotating ellipsoid was selected as the general 3D shape of slip surfaces. This ellipsoid can rotate on x-y plane ($0 \leq \theta_{xy} \leq \pi$) to provide various slip surfaces (Figure 4).

In order to achieve the equation of rotated ellipsoid, the parametric equation of general ellipsoid was transferred into new axes by the rotation angle, θ_{xy}. The result presents the rotated ellipsoid in (8). It should be noted that this ellipsoid

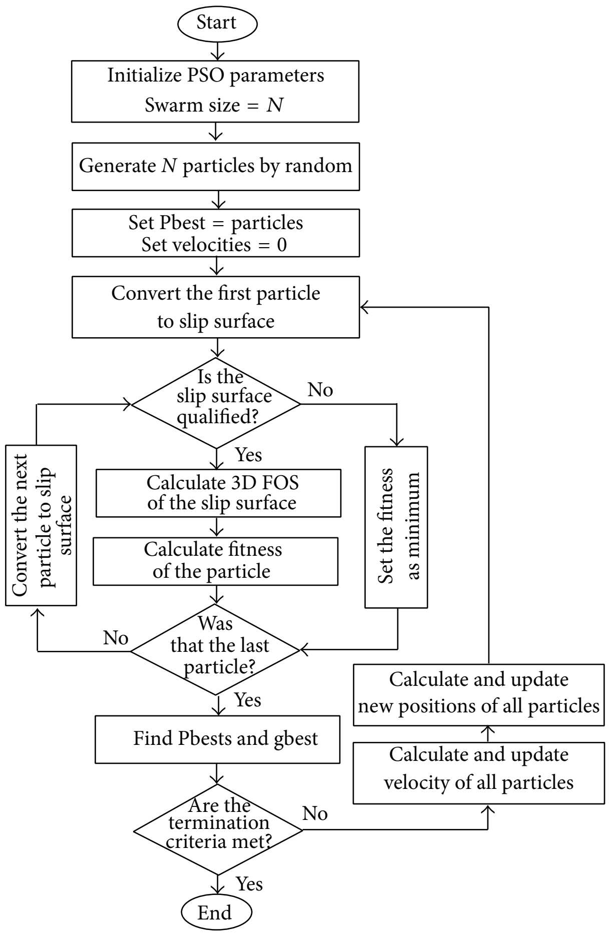

Start
Initialize PSO parameters
Swarm size = N
Generate N particles by random
Set Pbest = particles
Set velocities = 0
Convert the first particle to slip surface
Is the slip surface qualified?
No
Yes
Calculate 3D FOS of the slip surface
Calculate fitness of the particle
Set the fitness as minimum
Convert the next particle to slip surface
Was that the last particle?
No
Yes
Find Pbests and gbest
Calculate and update new positions of all particles
Calculate and update velocity of all particles
Are the termination criteria met?
No
Yes
End

Figure 3: Flowchart of PSO to determine the CSS in slope stability analysis.

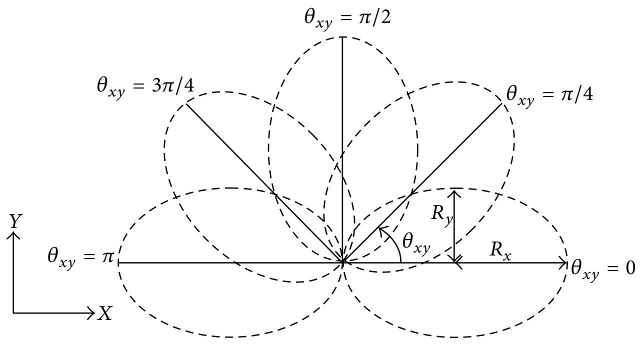


Figure 4: Projection of rotating ellipsoid on x-y plane.

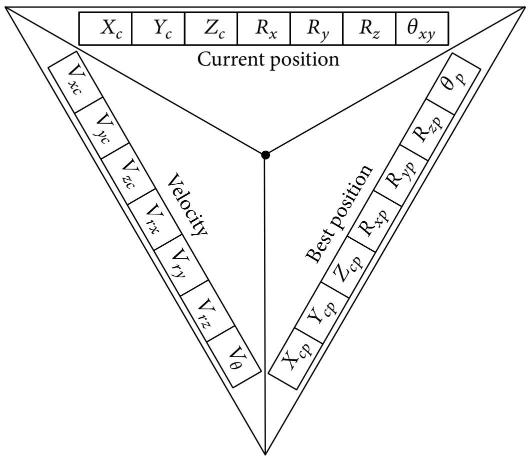

X_c Y_c Z_c R_x R_y R_z θ_{xy}
Current position
V_{xc} V_{yc} V_{zc} V_{rx} V_{ry} V_{rz} V_θ
Velocity
X_{cp} Y_{cp} Z_{cp} R_{xp} R_{yp} R_{zp} θ_p
Best position

Figure 5: Schematic structures of PSO particles.

can be easily transformed to spherical and cylindrical slip surface by, respectively, setting equal three and two semiradiuses:

$$\frac{\left(\cos\theta_{xy}\left(x-X_c\right)-\sin\theta_{xy}\left(y-Y_c\right)\right)^2}{R_x^2} - \frac{\left(\cos\theta_{xy}\left(y-Y_c\right)+\sin\theta_{xy}\left(x-X_c\right)\right)^2}{R_y^2} + \frac{\left(z-Z_c\right)^2}{R_z^2} = 1, \tag{8}$$

where θ_{xy} is rotation angle of the ellipsoid in x-y plane, X_c, Y_c, and Z_c are coordinates of center of ellipsoid in x-, y-, and z-directions, R_x, R_y, and R_z are semiradiuses of ellipsoid in x-, y-, and z-directions, and x, y, and z are coordinates of an arbitrary point on the surface of ellipsoid.

Based on the parameters of the rotating ellipsoid, Figure 5 shows schematic structure of a PSO particle. The section of current position records the coordinates of center of ellipsoid, its semiradiuses, and its rotation angle. Considering best position and velocity sections, PSO has seven-dimensional search space and twenty-one-cell particles.

3.2. Fitness Function. The quality of particles can be calculated by the fitness function. This function is related to the objective function and provides quantitative tracking of improvement of particles. Consequently, it makes it possible to compare and rank particles in the swarm, where maximum fitness shows the best particle and minimum fitness identifies the worst particle of the swarm. PSO attempts to increase the maximum fitness of swarms during its iterations. Since the objective function of 3D slope stability analysis is equation of FOS, PSO attempts to find the CSS with the minimum FOS by maximizing the fitness value in

$$\text{Fitness}_{n(i)} = \frac{1}{\text{FOS}\left(x_{n(i)}\right)}, \tag{9}$$

where $\text{Fitness}_{n(i)}$ is fitness value of the nth particle in ith iteration and $\text{FOS}(x_{n(i)})$ is FOS of 3D slip surface described by $x_{n(i)}$.

3.3. Sensitivity Analysis on PSO Parameters. The best performance of PSO is guaranteed by initializing appropriate parameters for it. A sensitivity analysis can help to do so. The optimum values PSO parameters in 3D slope stability analysis were defined by designing and performing several independent tests on swarm size, coefficients of velocity, and inertia weight of the swarm. In addition, the convergence

Table 1: Properties of slope in sensitivity analysis.

	Parameters		
	γ (kN/m^3)	c' (kN/m^2)	ϕ' (°)
Layer 1	19.78	12	17
Layer 2	17.64	24.5	20

Table 2: Results of sensitivity tests on swarm size.

	Test number						
	1	2	3	4	5	6	7
Swarm size	5	15	25	35	45	55	65
Total CPU time (s)	365	625	549	1726	2792	2241	10101

behavior of PSO as the average fitness of swarms was observed during the tests. A 3D soil slope was designed with complex geometry, layers of soil, and piezometric line (Figure 6). It should be noted that coding of the study was done by the authors in MATLAB software (Licensed by Universiti Teknologi Malaysia). The overall shape of the slope shows two imbalanced hills with steep sides makes it difficult to find the CSS for conventional slope stability analyses. This specific shape was selected to verify the effectiveness of PSO in complex 3D slopes. The properties of soil layers are described in Table 1.

The size of the swarm is defined based on the condition of search space, dimension of particles, and/or other specifications of the problem. The most common population sizes are 20 to 50 [25]. However, Clerc and Kennedy [36] proposed a relationship to determine the optimum value of swarm size as follows:

$$N_s = 10 + \left[\sqrt{D_s}\right], \tag{10}$$

where N_s is swarm size, D_s is dimension of the particles, and [ ] is calculator of integer part. Since the dimension of particles in the present problem is seven, the proposed optimum swarm size by this equation is 12. Considering the most common range of the swarm size and the result of equation, an interval of swarm sizes was prepared for sensitivity test. It should be noted that the first swarm of all tests was produced by the same random pattern and the maximum iteration number was set to 100 for all tests. Table 2 shows the results of tests. The phrase "CPU time" in this table means the exact amount of time that CPU spent on each test. CPU time was used to produce fair comparisons, since some factors including the operating system and available memory can affect the overall duration of the tests.

Figure 7 illustrates the convergence behavior of PSO in corresponding swarm sizes of Table 2. Three different convergence behaviors as good, late, and failure can describe these trends. Swarm sizes 5, 15, 25, 35, and 55 provided good convergence over maximum iterations, while swarm sizes 45 and 65, respectively, delivered delay and failure in convergence. Among all the tests, the best convergence was obtained by swarm size of 35 that provided the best convergence with the highest average fitness.

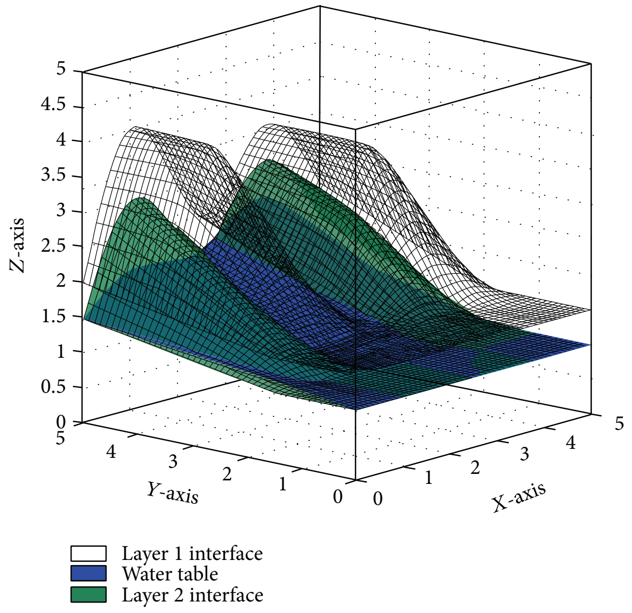


Figure 6: Generated slope model for sensitivity analysis.

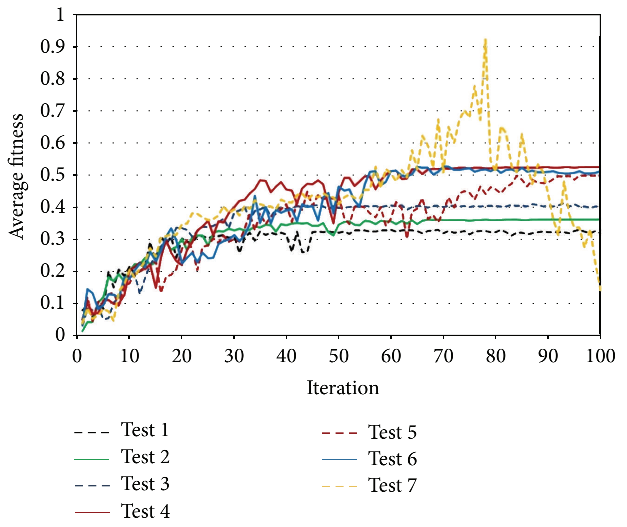
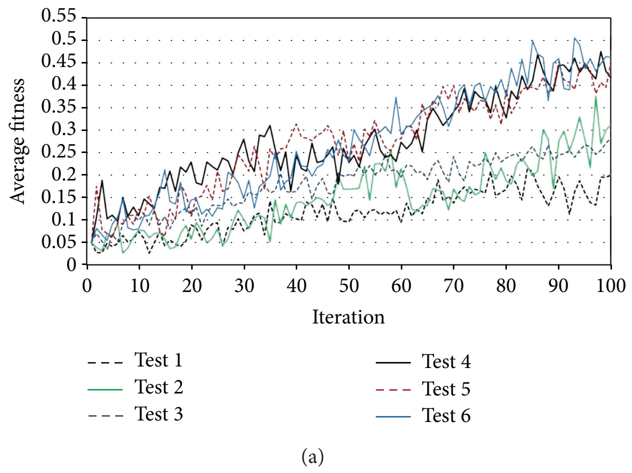


Figure 7: Convergence behavior of PSO with different swarm sizes.

The next tests were performed to find the optimum values of coefficients ϑ_1 and ϑ_2 of velocity equation. Based on the original coefficients of Kennedy and Eberhart [4] and the modified coefficients of Clerc and Kennedy [36], a series of combinations were established as shown in Table 3. All tests were performed by the same initial swarm with the size of 35 (previously obtained as optimum) and the maximum iterations of 100.

The results can be presented in two separated groups including unequal and equal coefficients. Figure 8 illustrate the results of the tests. The first group failed to converge over the maximum iteration period, but the second group showed different performances. Overall, the best convergence and the greatest average fitness belonged to equal coefficients of 1.75 that makes it the optimum coefficient of velocity equation.



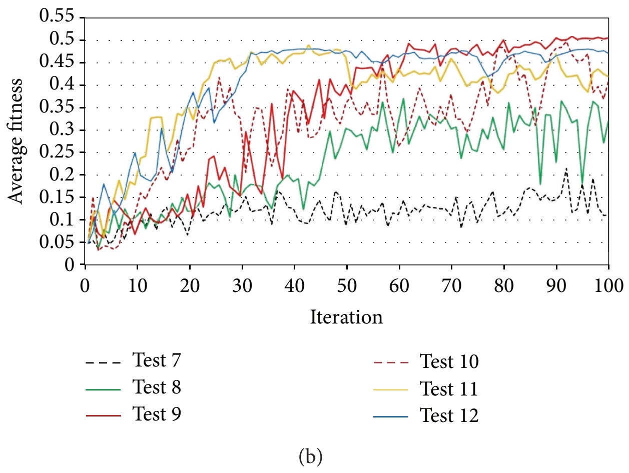


FIGURE 8: Results of sensitivity tests on (a) unequal and (b) equal coefficients.

TABLE 3: Combinations of velocity equation coefficients in different tests.

Test number	Relationship	ϑ_1	ϑ_2	$\vartheta_1 + \vartheta_2$
1	$\vartheta_1 = 0.25\vartheta_2$	0.800	3.200	4
2	$\vartheta_1 = 0.50\vartheta_2$	1.333	2.667	4
3	$\vartheta_1 = 0.75\vartheta_2$	1.714	2.286	4
4	$\vartheta_2 = 0.25\vartheta_1$	3.200	0.800	4
5	$\vartheta_2 = 0.50\vartheta_1$	2.667	1.333	4
6	$\vartheta_2 = 0.75\vartheta_1$	2.286	1.714	4
7	$\vartheta_1 = \vartheta_2$	2.500	2.500	5
8	$\vartheta_1 = \vartheta_2$	2.000	2.000	4
9	$\vartheta_1 = \vartheta_2$	1.750	1.750	3.5
10	$\vartheta_1 = \vartheta_2$	1.500	1.500	3
11	$\vartheta_1 = \vartheta_2$	1.000	1.000	2
12	$\vartheta_1 = \vartheta_2$	0.500	0.500	1

The last sensitivity tests were performed to find the optimum inertia weight (ω) of velocity equation. The same initial swarm with size of 35 and equal coefficients of velocity equation as 1.75 (previously defined as optimum values) were applied for all the tests. Five tests with inertia weights of 0, 0.25, 0.5, 0.75, and 1, respectively, were performed based on the proposed values of Shi and Eberhart [35] and Clerc and Kennedy [36]. Figure 9 shows the convergence behavior of PSO in the tests. The results showed successful convergence in all tests, except test 3. It should be noted that test 5 was identical with the original PSO, where no inertia weight was present in velocity equation. Immature convergence has occurred for tests 1 and 2. Although fast convergence appears an advantage at first, it is a sign of trapping a swarm in local solutions. Test 3 failed to converge, test 4 had instable convergence, and test 5 failed to improve its average fitness over the maximum iterations. Consequently, it was not possible to introduce an optimum inertia weight to guarantee

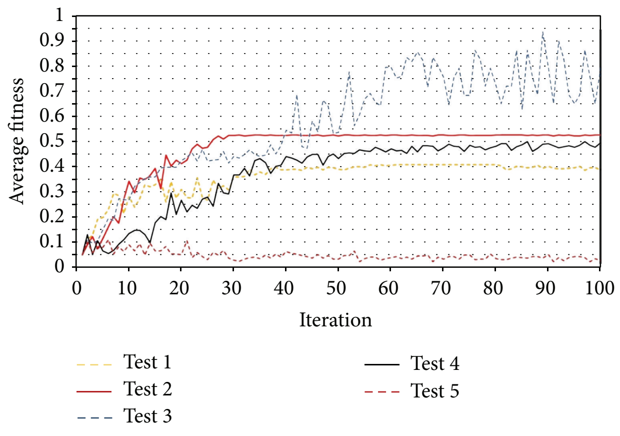


FIGURE 9: Results of sensitivity tests on inertia weight.

the convergence of PSO and improvement of average fitness over the maximum iteration number simultaneously.

A dynamic inertia weight was utilized by the present study to overcome the convergence problem of PSO. The proposed strategy started with the most anticonvergence inertia weight (0.5), continued with the normal convergence inertia weight (0.75), and ended up with the most stable convergence (0.25). The switching levels of inertia weights were defined as one-third and two-thirds of maximum iterations. Figure 10 shows the results of sensitivity tests on dynamic inertia weight with different maximum iterations from 50 to 300 by steps of 50. All tests performed well to converge and improve the average fitness over their maximum iterations, so dynamic inertia weight was adopted for PSO.

4. Example Problems

Two example problems were analyzed to verify the performance of PSO in determining the CSS. The properties of the slope materials are shown in Table 4. Example problem 1 was performed to verify the performance of PSO in determining

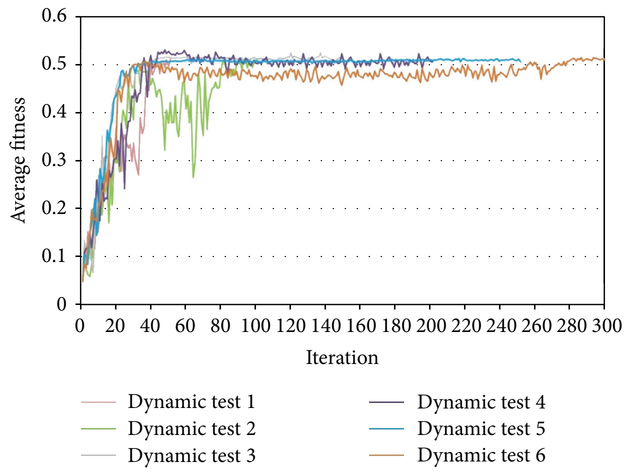


FIGURE 10: Results of sensitivity tests on dynamic inertia weight.

TABLE 4: Properties of slopes in example problems.

Properties	c' (kN/m^2)	ϕ' (degree)	γ (kN/m^3)	υ	E (kN/m^2)
Problem 1	15	20	17	0.3	$1E+6$
Problem 2	10	10	18	—	—

the CSS in comparison with PLAXIS-3D finite element software (License by Universiti Teknologi Malaysia). Alkasawneh et al. [44] applied different search techniques to determine the CSS in 2D slope stability analysis. Figure 11 illustrates the geometry of the slope. A 3D model was developed based on this 2D section in which the third dimension was extended by 100 meters. Figure 12 shows the generated 3D models of the slope by the present study and PLAXIS-3D. In both methods, cylindrical slip surface was employed to determine the CSS of the slope.

PSO improved average and best fitness of the swarm as shown in Figure 13. Figure 14 shows the minimum FOS versus iterations. PSO provided continuous reduction of FOS to find the CSS. The present study obtained FOS of 1.78 versus the minimum FOS of 1.77 of the PLAXIS-3D. Figure 15 shows the CSS obtained by the present study and the result of PLAXIS-3D. The present study and PLAXIS-3D obtained similar FOS for the CSS with a small difference of 0.3%. This result demonstrates the ability of PSO to determine the CSS with the minimum FOS in 3D slope stability analysis.

Example problem 2 was performed to verify the ability of PSO to determine the CSS with general ellipsoid shape in a comparison with previous studies from the literature. This example was initially analyzed by Yamagami et al. [45] and was reanalyzed by Yamagami and Jiang [25] Yamagami et al. [45] used random generation of surfaces to determine the CSS of this slope, while Yamagami and Jiang [25] employed a combination of dynamic programming and random number generation to do so. It should be noted that the same equation of FOS as previous studies was used to make fair comparison of the results. The example involved a homogeneous slope with gradient of 2 : 1 subjected to a square load of 50 kPa on the top. The uniform load was applied on a square surface

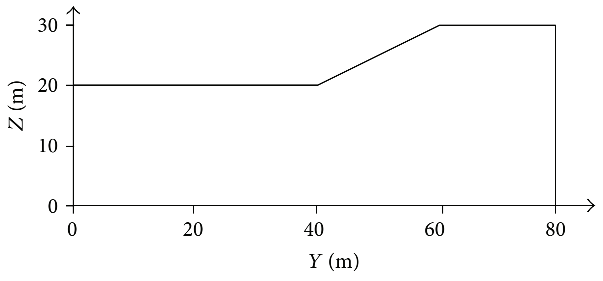
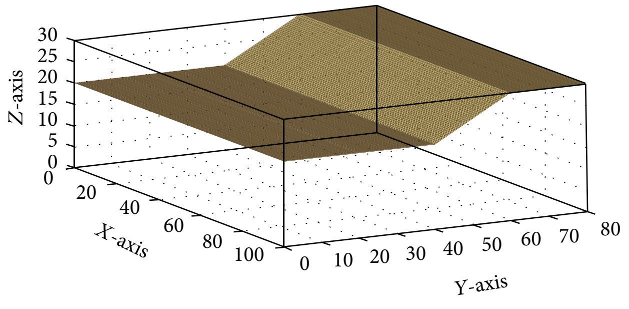
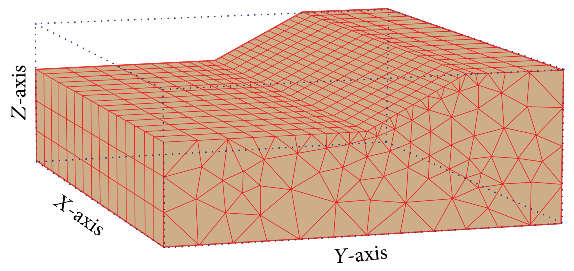


FIGURE 11: Geometry of 2D section of example problem 1.



(a)



(b)

FIGURE 12: Generated 3D models of example problem 1 by (a) the present study and (b) PLAXIS-3D.

with 8 meters sides at the top center of the slope. Figure 16 illustrates the geometry of example problem 1.

Figure 17 shows the generated 3D model by the present study for example problem 2. Figure 18 plots the process of PSO to improve fitness of the swarm. The trend of average fitness value of the swarm experienced some instability during the process. The main reason of this behavior is the presence of disqualified slip surfaces in the swarm that dramatically decreases the average fitness value. These surfaces were rarely presented in the previous example due to adopted simpler cylindrical shape compared with more complicated ellipsoid shape in this example. The trend of minimum FOS versus iterations is shown in Figure 19. Continuous decrement of FOS by PSO leads to determining the CSS of the slope.

In spite of similar equation of FOS, the present study found the CSS with a smaller FOS than other methods which is the best result so far. The minimum FOS obtained by PSO was 0.95 compared with 1.14 and 1.03 of random generation of surfaces [45] and DP with RNG [25], respectively. This result demonstrates the ability of PSO to accurately determine the ellipsoid CSS in 3D slope stability analysis. Figure 20

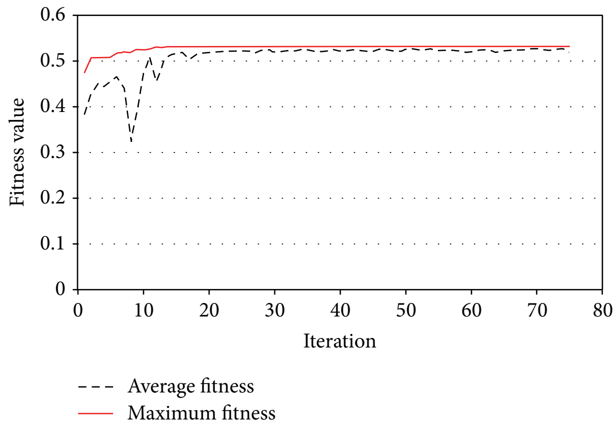


Figure 13: Fitness values versus iterations in example problem 1.

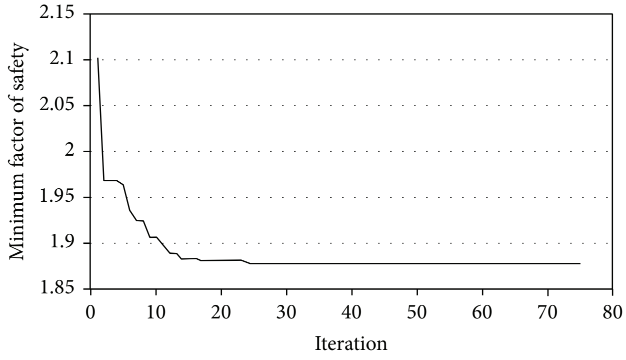
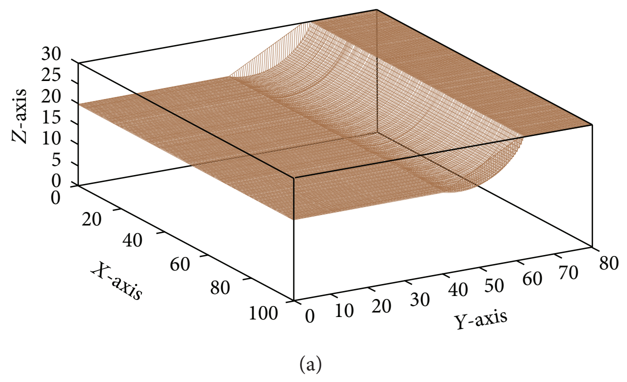


Figure 14: Minimum FOS versus iterations in example problem 1.



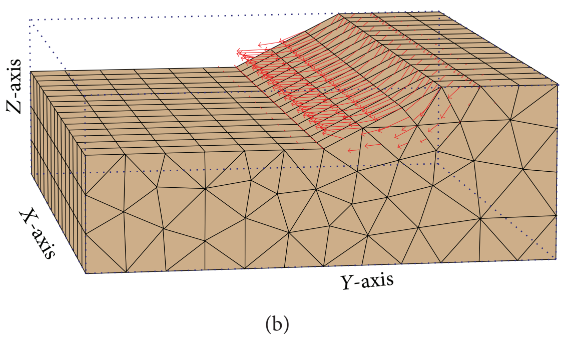


Figure 15: (a) The CSS obtained by the present study and (b) exaggerated displacement vectors of PLAXIS-3D in example problem 1.

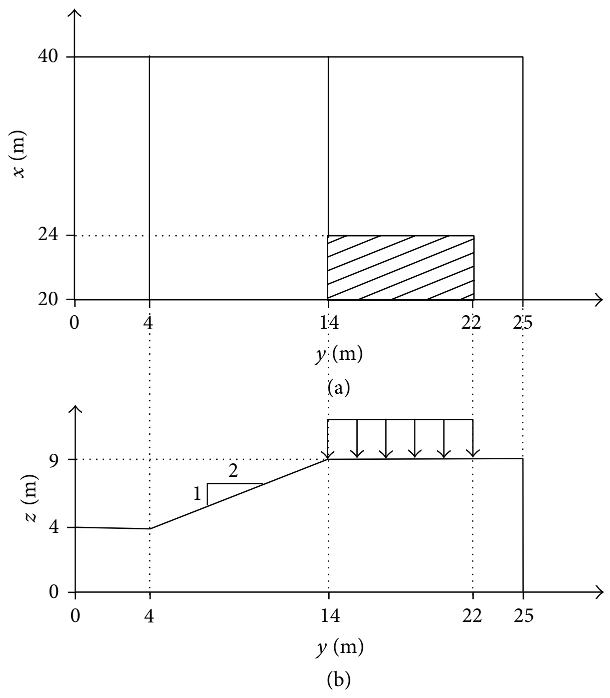


Figure 16: (a) Half-plan view and (b) central cross-section of slope in example problem 2.

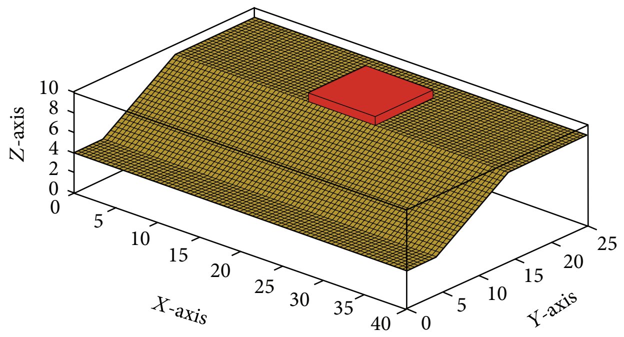


Figure 17: Generated 3D models of example problem 2 by the present study.

illustrates the CSS obtained by the present study in example problem 2.

5. Conclusion

Determining the critical slip surface of a soil slope is a traditional problem in geotechnical engineering which is still challenging for researchers. This problem needs a massive searching process. Although classical searching methods work for relatively simple problems, they are surrounded by local minima. Moreover, their processes become particularly slow by increasing the number of possible solutions. To eliminate these limitations, PSO has been applied in slope stability analysis based on its successful results in advanced engineering problems. However, this contribution was limited to 2D slope stability analysis. This paper applied PSO in

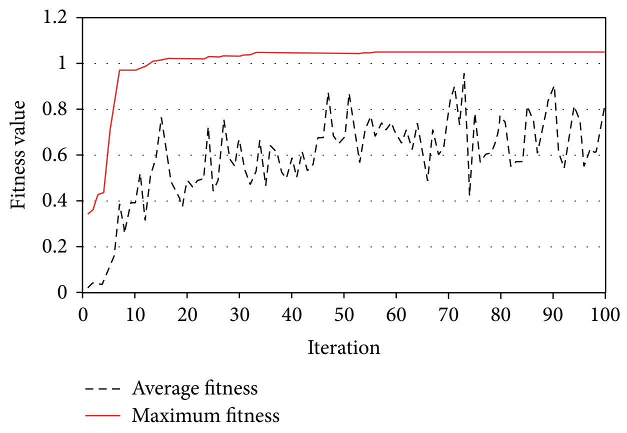


Figure 18: Fitness values versus iterations in example problem 2.

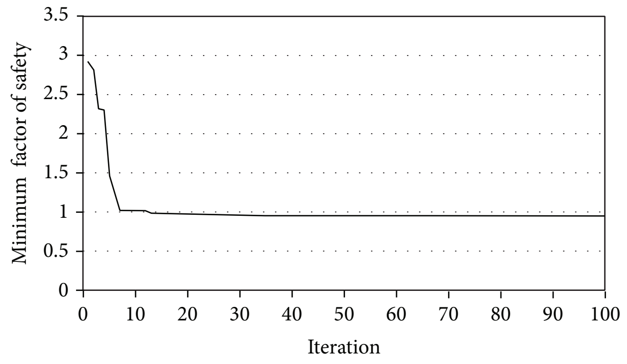


Figure 19: Minimum FOS versus iterations in example problem 2.

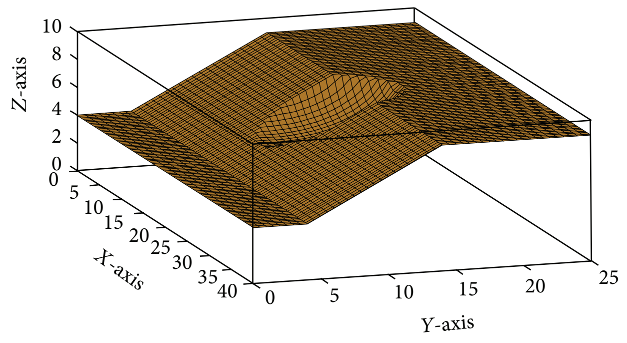


Figure 20: The CSS obtained by the present study in example problem 2.

3D slope stability problem to determine the CSS of soil slopes. A detailed description of adopted PSO was presented to provide a good basis for more contribution of this technique to the field of 3D slope stability problems.

The application of PSO in slope stability analysis was described by presenting a general flowchart. A general rotating ellipsoid shape was introduced as the specific particle for 3D slope stability analysis. In order to find the optimum values of parameters of PSO, a sensitivity analysis was designed and performed. The related codes were prepared by the authors in MATLAB. A 3D model with complex geometry, soil layers, and piezometric line was used in the analysis. This analysis included three steps to find the optimum swarm size, coefficients, and inertia weight of the velocity equation, respectively. Moreover, the performance of PSO to converge over a global optimum solution was verified during the tests. Based on the obtained values of parameters, PSO was prepared for 3D slope stability analysis.

The applicability of PSO in determining the CSS of 3D slopes was evaluated by analyzing two example problems. The first example presented a comparison between the results of PSO and PLAXI-3D finite element software. The second example compared the ability of PSO to determine the CSS of 3D slopes with other optimization methods from the literature. Both of the example problems demonstrated the efficiency and effectiveness of PSO in determining the CSS of 3D soil slopes. Based on the results, it is believed that PSO is highly capable of contributing to the field of 3D slope stability analysis.

Conflict of Interests

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

Acknowledgments

The presented study was undertaken with support from research grant. The authors are pleased to acknowledge the Ministry of Education of Malaysia for the Prototype Research Grant Scheme (R.J130000.7822.4L620) and Universiti Teknologi Malaysia for providing the research funds.

References

[1] R. Kalatehjari and N. Ali, "A review of three-dimensional slope stability analyses based on limit equilibrium method," *Electronic Journal of Geotechnical Engineering*, vol. 18, pp. 119–134, 2013.

[2] D. G. Fredlund, "Analytical methods for slope stability analysis," in *Proceedings of the 4th International Symposium on Landslides, State-of-the-Art*, pp. 229–250, Toronto, Canada, September 1984.

[3] A. W. Bishop, "The use of the slip circle in the stability analysis of earth slope," *Geotechnique*, vol. 5, no. 1, pp. 7–17, 1955.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[5] Y. Zhang, P. Agarwal, V. Bhatnagar, S. Balochian, and J. Yan, "Swarm intelligence and its applications," *The Scientific World Journal*, vol. 2013, Article ID 528069, 3 pages, 2013.

[6] G. Poitras, G. Lefrançois, and G. Cormier, "Optimization of steel floor systems using particle swarm optimization," *Journal of Constructional Steel Research*, vol. 67, no. 8, pp. 1225–1231, 2011.

[7] S. Gholizadeh and E. Salajegheh, "Optimal design of structures subjected to time history loading by swarm intelligence and an advanced metamodel," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, no. 37–40, pp. 2936–2949, 2009.

[8] S. S. Gilan, H. B. Jovein, and A. A. Ramezanianpour, "Hybrid support vector regression—particle swarm optimization for

prediction of compressive strength and RCPT of concretes containing metakaolin," *Construction and Building Materials*, vol. 34, pp. 321–329, 2012.

[9] L. Yunkai, T. Yingjie, O. Zhiyun et al., "Analysis of soil erosion characteristics in small watersheds with particle swarm optimization, support vector machine, and artificial neuronal networks," *Environmental Earth Sciences*, vol. 60, no. 7, pp. 1559–1568, 2010.

[10] S. Wan, "Entropy-based particle swarm optimization with clustering analysis on landslide susceptibility mapping," *Environmental Earth Sciences*, vol. 68, no. 5, pp. 1349–1366, 2013.

[11] M. R. Nikoo, R. Kerachian, A. Karimi, A. A. Azadnia, and K. Jafarzadegan, "Optimal water and waste load allocation in reservoir-river systems: a case study," *Environmental Earth Sciences*, vol. 71, no. 9, pp. 4127–4142, 2014.

[12] K. K. Kuok, S. Harun, and S. M. Shamsuddin, "Particle swarm optimization feedforward neural network for hourly rainfall-runoff modeling in Bedup Basin, Malaysia," *International Journal of Civil & Environmental Engineering*, vol. 9, no. 10, pp. 9–18, 2010.

[13] K. K. Kuok, S. Harun, and S. M. Shamsuddin, "Particle swarm optimization feedforward neural network for modeling runoff," *International Journal of Environmental Science and Technology*, vol. 7, no. 1, pp. 67–78, 2010.

[14] J. S. Yazdi, F. Kalantary, and H. S. Yazdi, "Calibration of soil model parameters using particle swarm optimization," *International Journal of Geomechanics*, vol. 12, no. 3, pp. 229–238, 2012.

[15] T. V. Bharat, P. V. Sivapullaiah, and M. M. Allam, "Swarm intelligence-based solver for parameter estimation of laboratory through-diffusion transport of contaminants," *Computers and Geotechnics*, vol. 36, no. 6, pp. 984–992, 2009.

[16] B. R. Chen and X. T. Feng, "CSV-PSO and its application in geotechnical engineering," in *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, F. T. S. Chan and M. K. Tiwari, Eds., chapter 15, pp. 263–288, 2007.

[17] Y. M. Cheng, L. Li, and S. C. Chi, "Performance studies on six heuristic global optimization methods in the location of critical slip surface," *Computers and Geotechnics*, vol. 34, no. 6, pp. 462–484, 2007.

[18] Y. M. Cheng, L. Li, Y. J. Sun, and S. K. Au, "A coupled particle swarm and harmony search optimization algorithm for difficult geotechnical problems," *Structural and Multidisciplinary Optimization*, vol. 45, no. 4, pp. 489–501, 2012.

[19] H. B. Zhao, Z. S. Zou, and Z. L. Ru, "Chaotic particle swarm optimization for non-circular critical slip surface identification in slope stability analysis," in *Proceedings of the International Young Scholars' Symposium on Rock Mechanics: Boundaries of Rock Mechanics Recent Advances and Challenges for the 21st Century*, pp. 585–588, Beijing, China, May 2008.

[20] D. Tian, L. Xu, and S. Wang, "The application of particle swarm optimization on the search of critical slip surface," in *Proceedings of the International Conference on Information Engineering and Computer Science (ICIECS '09)*, pp. 1–4, Wuhan, China, December 2009.

[21] L. Li, G. Yu, X. Chu, and S. Lu, "The harmony search algorithm in combination with particle swarm optimization and its application in the slope stability analysis," in *Proceedings of the International Conference on Computational Intelligence and Security (CIS '09)*, vol. 2, pp. 133–136, Beijing, China, December 2009.

[22] R. Kalatehjari, N. Ali, M. Kholghifard, and M. Hajihassani, "The effects of method of generating circular slip surfaces on determining the critical slip surface by particle swarm optimization," *Arabian Journal of Geosciences*, vol. 7, no. 4, pp. 1529–1539, 2014.

[23] R. Kalatehjari, N. Ali, M. Hajihassani, and M. K. Fard, "The application of particle swarm optimization in slope stability analysis of homogeneous soil slopes," *International Review on Modelling and Simulations*, vol. 5, no. 1, pp. 458–465, 2012.

[24] W. Chen and P. Chen, "PSOslope: a stand-alone windows application for graphic analysis of slope stability," in *Advances in Swarm Optimization*, Y. Tan, Y. Shi, Y. Chai, and G. Wang, Eds., vol. 6728 of *Lecture Notes in Computer Science*, pp. 56–63, 2011.

[25] T. Yamagami and J. C. Jiang, "A search for the critical slip surface in three-dimensional slope stability analysis," *Soils and Foundations*, vol. 37, no. 3, pp. 1–16, 1997.

[26] J. C. Jiang, T. Yamagami, and R. Baker, "Three-dimensional slope stability analysis based on nonlinear failure envelope," *Chinese Journal of Rock Mechanics and Engineering*, vol. 22, no. 6, pp. 1017–1023, 2003.

[27] X. J. E. Mowen, "A simple Monte Carlo method for locating the three-dimensional critical slip surface of a slope," *Acta Geologica Sinica*, vol. 78, no. 6, pp. 1258–1266, 2004.

[28] X. J. E. Mowen, W. Zengfu, L. Xiangyu, and X. Bo, "Three-dimensional critical slip surface locating and slope stability assessment for lava lobe of Unzen Volcano," *Journal of Rock Mechanics and Geotechnical Engineering*, vol. 3, no. 1, pp. 82–89, 2011.

[29] Y. M. Cheng, H. T. Liu, W. B. Wei, and S. K. Au, "Location of critical three-dimensional non-spherical failure surface by NURBS functions and ellipsoid with applications to highway slopes," *Computers and Geotechnics*, vol. 32, no. 6, pp. 387–399, 2005.

[30] M. Toufigh, A. Ahangarasr, and A. Ouria, "Using non-linear programming techniques in determination of the most probable slip surface in 3D slopes," *World Academy of Science, Engineering and Technology*, vol. 17, pp. 30–35, 2006.

[31] C. W. Reynolds, "Flocks, herbs, and schools: a distributed behavorial model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[32] R. Kalatehjari, *An improvised three-dimensional slope stability analysis based on limit equilibrium method by using particle swarm optimization [Ph.D. dissertation]*, Faculty of Civil Engineering, Universiti Teknologi Malaysia, 2013.

[33] A. P. Engelbrecht, *Computational intelligence: an introduction*, John Wiley & Sons, 2007.

[34] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[35] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, Piscataway, NJ, USA, May 1998.

[36] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[37] R. Mendes, P. Cortez, M. Rocha, and J. Neves, "Particle swarms for feedforward neural network training," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '02)*, pp. 1895–1899, Honolulu, Hawaii, USA, May 2002.

[38] 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, IEEE, Nagoya, Japan, October 1995.

[39] R. Baker, "Determination of the critical slip surface in slope stability computations," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 4, no. 4, pp. 333–359, 1980.

[40] H. P. J. Bolton, G. Heymann, and A. A. Groenwold, "Global search for critical failure surface in slope stability analysis," *Engineering Optimization*, vol. 35, no. 1, pp. 51–65, 2003.

[41] Y. Chen, X. Wei, and Y. Li, "Locating non-circular critical slip surfaces by particle swarm optimization algorithm," *Chinese Journal of Rock Mechanics and Engineering*, vol. 25, no. 7, pp. 1443–1449, 2006.

[42] L. Li, S.-C. Chi, R.-M. Zheng, G. Lin, and X.-S. Chu, "Mixed search algorithm of non-circular critical slip surface of soil slop," *China Journal of Highway and Transport*, vol. 20, no. 6, pp. 1–6, 2007.

[43] L. Li, S.-C. Chi, and Y.-M. Zheng, "Three-dimensional slope stability analysis based on ellipsoidal sliding body and simplified JANBU method," *Rock and Soil Mechanics*, vol. 29, no. 9, pp. 2439–2445, 2008.

[44] W. Alkasawneh, A. I. H. Malkawi, J. H. Nusairat, and N. Albataineh, "A comparative study of various commercially available programs in slope stability analysis," *Computers and Geotechnics*, vol. 35, no. 3, pp. 428–435, 2008.

[45] T. Yamagami, K. Kojima, and M. Taki, "A search for the three-dimensional critical slip surface with random generation of surface," in *Proceedings of the 36th Symposium on Slope Stability Analyses and Stabilizing Construction Methods*, pp. 27–34, 1991.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 413265, 7 pages
http://dx.doi.org/10.1155/2014/413265

Research Article

Two-Cloud-Servers-Assisted Secure Outsourcing Multiparty Computation

Yi Sun,[1] Qiaoyan Wen,[1] Yudong Zhang,[2] Hua Zhang,[1] Zhengping Jin,[1] and Wenmin Li[1]

[1] *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2] *Brain Image Processing, Columbia University, New York, NY 10032, USA*

Correspondence should be addressed to Yi Sun; sybupt@bupt.edu.cn

Received 19 February 2014; Accepted 13 April 2014; Published 28 May 2014

Academic Editors: A. Miné and B. Sun

Copyright © 2014 Yi Sun 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.

We focus on how to securely outsource computation task to the cloud and propose a secure outsourcing multiparty computation protocol on lattice-based encrypted data in two-cloud-servers scenario. Our main idea is to transform the outsourced data respectively encrypted by different users' public keys to the ones that are encrypted by the same two private keys of the two assisted servers so that it is feasible to operate on the transformed ciphertexts to compute an encrypted result following the function to be computed. In order to keep the privacy of the result, the two servers cooperatively produce a custom-made result for each user that is authorized to get the result so that all authorized users can recover the desired result while other unauthorized ones including the two servers cannot. Compared with previous research, our protocol is completely noninteractive between any users, and both of the computation and the communication complexities of each user in our solution are independent of the computing function.

1. Introduction

Secure multiparty computation (SMC) [1–7] is dedicated to computing a certain function among a set of mutually distrusted participants on their private inputs without revealing private information. Informally speaking, assuming that there are m participants, $P_1, P_2, \ldots, P_m$, each of them has a private number, respectively, $x_1, x_2, \ldots, x_m$. They want to cooperate to compute the function $y = f(x_1, x_2, \ldots, x_m)$ without revealing x_i of P_i to other parties $P_j, j \neq i, i, j \in \{1, \ldots, m\}$, as well as guaranteeing that any unauthorized ones cannot get the result y. In the past, researchers mainly focused on designing the style of secure multiparty computation protocols by which users themselves cooperatively accomplish the function evaluation through their internal interactions [1, 3, 8–11]. The computation and communication complexities always depend polynomially on the complexity of the function to be computed. Therefore, users suffer from the heavy overload of these protocols.

The emergence of the cloud [12, 13] inspires users to apply the powerful computing ability of the cloud to help them to conduct complicated computations, that is, secure outsourcing computation [14–18] to the cloud. They expect that the cloud can independently complete any function computation on their outsourced data although the data has been encrypted by their own keys for security. Moreover, the final result should be kept private to the cloud even though it is the cloud that conducts all of the computations about the computing function. In this way, users only need to encrypt their data and decrypt the returned message to get the desired result. All computations about the computing function are in the charge of the cloud. There are no interactions between any users, and the computation and communication complexities of each user are independent of the computing function. However, this expectation is proven to be impossible in the single cloud server setting due to the impossibility of program obfuscation [19]. Therefore, in this paper, we try to realize it by introducing one more cloud server to the original model described above. More precisely, we consider the following scenario.

There are $m + 2$ distrusted parties including m users and two cloud servers in our system. We assume that all of them act semihonestly. The m users $P_1, P_2, \ldots, P_m$, with each having a private input, respectively, x_i, as well as a pair of public-private keys (pk_i, sk_i), $i = 1, \ldots, m$, encrypt their respective private inputs by their own public keys and then upload the ciphertexts of the inputs to a cloud server. They want to obtain the value $y = f(x_1, x_2, \ldots, x_m)$ even if they may not be aware of what the computing function $f(\cdot)$ is by applying two cloud servers to operate on the outsourced encrypted data without revealing $x_1, x_2, \ldots, x_m$ and the result y.

In this paper, we study the outsourcing computation problem in multiple users-two-cloud-servers scenario and propose a two-cloud-servers-assisted secure outsourcing multiparty computation protocol to compute any function on lattice-based encrypted data under multiple keys of the users. Herein, we apply one cloud server called the storing-cloud (SC) to store the outsourced data encrypted by users and make a midtransformation to these ciphertexts once some function begins to be computed. We call the other cloud server the computing-cloud (CC). It is responsible for transforming the midtransformed ciphertexts by SC to the ones that are blinded by the same two private keys of the two assisted cloud servers so that CC can further compute y following the function on the ciphertexts. Finally, in order to protect the result, the two servers cooperatively produce a custom-made result for each user. Compared with previous solutions, our protocol has the following three advantages.

(1) Our protocol is completely noninteractive between any users.

(2) The cloud is to do all of the computations related to the computing function, while users would do nothing except for encrypting their private inputs and decrypting the returned result.

(3) The computation and communication complexities of each user in our solution are independent of the computing function.

Organization. The rest of this paper is organized as follows. In Section 2, we briefly give an overview of some recent related works. Herein, we consider the problems in secure outsourcing computation from the point of view of the users and the cloud servers, respectively, and then rationally construct our protocol in the multiple users-two-cloud-servers setting. In Section 3, we briefly introduce a lattice-based encryption scheme and the security model and then present our protocol in Section 4 in detail. In Section 5, we analyze the proposed protocol in detail and give a strict proof based on real-ideal simulation paradigm. Finally, we summarize our work of this paper in the last section.

2. Related Works

According to previous research, there are many problems to be considered when outsourcing private data for function computation to the cloud. We discuss the difficulties in secure outsourcing computation to the cloud from the following two aspects.

(1) To Users: Privacy of the Inputs and Results. In secure outsourcing computation, users have to contribute their private data as the inputs of the function while not participating in the computation process. Moreover, all parties of the protocol including all users and cloud servers are mutually distrusted. Therefore, users would not like to submit their private data to the cloud. Allowing for security, a usual solution is to encrypt the private data before outsourcing them to the cloud. And there are some basic encryption models according to the encryption keys that users used.

In 2009, Gentry [20] presented a model where all users use a joint public key to encrypt their own private inputs while sharing the private key. Therein, the cloud cannot obtain the inputs or the result because they are protected by the encryption scheme, while the cloud does not have the private key. However, users have to participate in another interactive protocol to firstly recover the private key and then achieve the desired result. The processes, producing a joint public key, sharing the private key, and jointly recovering the result by their shared private key, bring large number of additional interactions among users, which is contrary to our expectation that we want to design a secure protocol with the least communications. Encrypting private data by the joint public key is not so satisfactory either. In cloud outsourcing scenario, it means that there are no interactions among the users whatsoever and the least two rounds of inevitable interactions between the user and the cloud server, sending out the inputs and receiving the result. Therefore, we look forward to a protocol with the least communications as well as low computations and high security. A recent work by Asharov et al. [21] proposes a scheme where users utilize their own public keys to encrypt their inputs, respectively, and guarantee that the cloud can succeed in computing the function on their private inputs by computing on the ciphertexts of the inputs encrypted under different keys. Although users still have to interact to obtain the result in the last step, encrypting respective input by the public key of each user is the best encryption model so far.

As to the privacy of the result, in 2011, Halevi et al. [22] proposed a noninteractive protocol to securely realize outsourcing computation. Therein, the server is entitled to learn the result. However, the computing result may be the vital information to the users in some scenario and so it cannot be revealed to others. Hence, besides the security of the inputs discussed above, users must consider the security of the result when constructing protocols. It should guarantee that any unauthorized users are not able to get the result although they may contribute their inputs and the cloud servers are not able to get the result although the result is computed by them. To this aspect, [20] has already protected the result by a joint public key of the users. However, this method is still not satisfactory since each authorized user is also not able to get it individually.

(2) To Cloud Servers: Feasibility of Operating on Encrypted Inputs. As discussed above, users would like to upload the encrypted inputs under their respective public keys to the cloud server rather than the original inputs. Therefore, the cloud servers, whose task is to compute a function on users' private inputs, would only obtain the ciphertexts of the inputs. That means that the cloud has to compute the function

(i) KeyGen(1^k): sample a ring element vector $a \leftarrow R_q^n$ and a ring element s from the distribution χ, denoted as $s \leftarrow \chi$, a ring element vector x from the distribution χ^n, denoted as $x \leftarrow \chi^n$. Then, the private key is $sk = s$; the public key is $p = as + 2x \in R_q^n$.
(ii) Enc(pk, m): sample $e \leftarrow \chi^n$ and compute $c_0 := \langle p, e \rangle + m \in R_q$ and $c_1 := \langle a, e \rangle \in R_q$. Output the ciphertext $c := (c_0, c_1) \in R_q^2$.
(iii) Dec(sk, c): compute $\mu = c_0 - c_1 s \in R_q$ and output $m' := \mu(\text{mod}2)$.

Algorithm 1

on users' private inputs through performing corresponding computations on the ciphertexts of the inputs encrypted by different public keys of users. As we know, fully homomorphic encryption (FHE) [20, 23] can operate on the ciphertexts of the inputs to compute the desired result produced by the inputs. But the usual FHE schemes are single-key schemes in the sense that they only can perform computations on ciphertexts encrypted under the same key. It is not feasible to conduct computations on the ciphertexts encrypted under different keys. In order to solve this problem, López-Alt et al. [24] propose a new FHE called multikey fully homomorphic encryption (MFHE) which has applied the techniques of bootstrapping, modulus reduction, and relinearization to operate on the ciphertexts of the inputs encrypted by multiple, unrelated keys. When outsourcing private data to the cloud, user can firstly encrypt it by its own key by applying MFHE. It is indeed the optimal solution from the point of view of the feasibility of ciphertexts and the privacy of inputs. However, as we mentioned before, it is still not satisfactory because users need to evaluate the decryption key and then use it to recover the result interactively by participating in another SMC protocol.

In fact, according to [19], it is proved that it is indeed impossible to construct a completely noninteractive protocol in the single server setting due to the impossibility of program obfuscation. Hence, if we want to obtain a secure protocol with complete noninteraction of users in outsourcing computation, we need at least two cloud servers.

In brief, allowing for the privacy of inputs and results from the perspective of users as well as the feasibility of operating on the outsourced encrypted data from the perspective of the cloud servers, if we want to construct a completely noninteractive secure outsourcing multiparty computation protocol where the computation and communication complexities of each user are independent of the computing function, we have the following conclusions.

(1) All private data should be encrypted by the owners themselves using their respective public keys before outsourcing to the cloud servers.

(2) The returned messages for each user should be different so that all authorized users can recover the final result by their respective private key but the unauthorized ones cannot.

(3) It is reasonable to consider it in two-cloud-servers scenario.

3. Preliminaries

3.1. Lattice-Based Encryption. Since the privacy of the inputs and the computation complexity of each user depend on the encryption algorithm that the user used, an encryption scheme outstanding in both security and efficiency is the right one that users want to adopt. Hence, lattice-based encryption, which is against quantum attacks and is much more efficient than RSA and even the elliptic curve cryptosystem, becomes the first choice of rational users. Herein, we will show how the two cloud servers deal with the outsourced data encrypted by the lattice-based public key encryption scheme proposed in [24, 25] (denoted as LE scheme in this paper). Specifically, we recall it as follows.

Notations. Let k be the security parameter. Then, the LE scheme is parameterized by a prime $q = q(k)$, a degree n polynomial $f(x) \in \mathbb{Z}[x]$, and an error distribution χ over the ring $R_q = \mathbb{Z}_q[x]/\langle f(x) \rangle$. The parameters n, f, q, and χ are public. It assumes that, given the security parameter k, there are polynomial-time algorithms that output f and q and a sample from the error distribution χ.

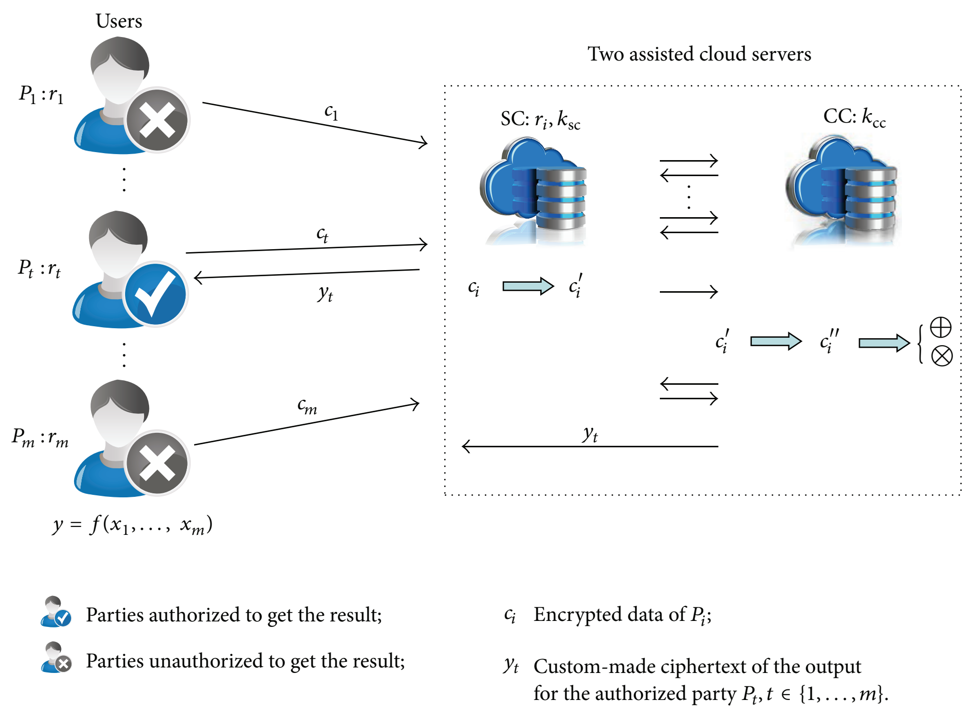
The LE encryption scheme consists of the following three algorithms: KeyGen($\cdot$), Enc($\cdot$), and Dec($\cdot$) (Algorithm 1).

In [24], they apply the techniques of bootstrapping, modulus reduction, and relinearization to realize and to operate on the ciphertexts of the inputs encrypted by multiple, unrelated keys. Therein, they have obtained a secure outsourcing multiparty computation protocol on lattice-based encrypted data under multiple keys of users in one server scenario. However, it is not satisfactory because the interaction in the decryption stage is still inevitable.

In this paper, based on this encryption scheme, we consider the outsourcing problem in two-cloud-servers scenario and succeed to construct a secure noninteractive outsourcing protocol that achieves the least computation and communication complexities for users.

3.2. Security Model. In this paper, we will discuss our protocol in the semihonest model and analyze its security using the real-ideal paradigm [5].

Firstly, in the ideal world, the computation of the functionality $\mathcal{F}$ on users' private inputs is conducted by an additional trusted party that receives x_i from user P_i, $i = 1, 2, \ldots, m$, and returns the result $f(x_1, x_2, \ldots, x_m)$ to the authorized users P_t, while other unauthorized parties do not get any output. Hence, in the ideal world, all users' private



Users
$P_1 : r_1$
$P_t : r_t$
$P_m : r_m$
c_1
c_t
y_t
c_m
$y = f(x_1, \ldots, x_m)$
Two assisted cloud servers
SC: r_i, k_{sc}
CC: k_{cc}
c_i
c_i'
c_i''
$\oplus$
$\otimes$
y_t
Parties authorized to get the result;
Parties unauthorized to get the result;
c_i Encrypted data of P_i;
y_t Custom-made ciphertext of the output for the authorized party $P_t, t \in \{1, \ldots, m\}$.

FIGURE 1: Framework of our construction.

inputs are well protected, and only authorized users are able to learn about the result. However, there is no trusted party in the real world, and so all parties have to run a protocol Π to get the desired result. During executing the protocol Π, all parties act semihonestly following the protocol but make effort to gain more information about other parties' inputs, intermediate results, or overall outputs by the transcripts of the protocol. An adversary can corrupt a party to receive all messages directed to it and control the messages to be sent out from it.

Herein, we denote the joint output of the ideal world adversary $\mathcal{S}$ and the outputs of the remaining parties in an ideal execution for computing the functionality $\mathcal{F}$ with inputs $\vec{x} = (x_1, x_2, \ldots, x_m)$ as $\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\vec{x})$, the joint output of the real world adversary $\mathcal{A}$, and the outputs of the remaining parties in an execution of protocol Π with inputs $\vec{x} = (x_1, x_2, \ldots, x_m)$ as $\text{REAL}_{\Pi,\mathcal{A}}(\vec{x})$. Then, we say that protocol Π securely realizes functionality $\mathcal{F}$ if, for every real adversary $\mathcal{A}$ corrupting any parties and possibly the cloud servers, there exists an ideal world adversary $\mathcal{S}$ with black-box access to $\mathcal{A}$ such that, for all input vectors $\vec{x}$, $\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\vec{x}) \overset{c}{\approx} \text{REAL}_{\Pi,\mathcal{A}}(\vec{x})$.

4. Our Result

We consider the secure outsourcing computation problem in the multiple users-two-cloud-servers scenario described as follows. There are $m + 2$ parties including m users and 2 noncolluding cloud servers: one is called the storing-cloud (SC), and the other is called the computing-cloud (CC). Each user P_i has a private input denoted as x_i and a pair of public-private keys (pk_i, sk_i) while sharing a private random r_i with SC that has a private number k_{sc}. CC has a private number k_{cc}. Users want to outsource the task of computing function $f(\cdot)$ on users' private inputs to the two cloud servers. They only provide the ciphertexts of the private data encrypted by a lattice-based encryption scheme under their different public keys and require the cloud servers to give the authorized users the result while keeping the security of the inputs. What is more, users wish that the cloud servers take charge of all of the computations related to the function $f(\cdot)$ and that there is noninteraction of users whatsoever so that the computation and communication complexities of each user are independent of the function to be computed. Herein, we deem that the two rounds of inevitable communications and a request from a user to the cloud servers for computing function $f(\cdot)$ are the three basic rounds of communication in this paper. Then, for each user, they expect that there are no other interactions at all between any user-to-user or user-to-server except the three basic rounds of communication. Furthermore, the computation complexity of each user depends on the encryption scheme it has used. The framework of our construction can be illustrated in Figure 1.

In this following section, we formally propose our solution denoted as protocol Π for convenience in detail and then analyze its security using the real-ideal paradigm in the semihonest model.

Without loss of generality, we represent the function $f(\cdot)$ to be computed by means of arithmetic circuit $\mathcal{C}_f$ consisting of any number of addition gates and l multiplication gates where each gate has two input wires and one output wire. Then, any functionality can be reduced to the two basic operations, addition and multiplication, over two inputs. Our construction can be summarized in Algorithm 2.

Protocol π: Two cloud servers-assisted secure outsourcing computation protocol
Setup.
For $i = 1, 2, \ldots, m$, sample a ring element vector $a_i \leftarrow R_q^n$, a ring element $s_i \leftarrow \chi$, and a ring element vector $x_i \leftarrow \chi^n$. Then, the private key of P_i is $sk_i := s_i$; the public key of P_i is $p_i := a_i s_i + 2x_i \in R_q^n$. And then, P_i shares a private random r_i with SC that has a private number k_{sc}. CC has a private number k_{cc}. As a preparation, user P_i firstly sends $r_i \cdot s_i$ to CC; CC computes $k_{\text{cc}} \cdot r_i \cdot s_i$ and then sends it back to SC. Then, SC can obtain $k_{\text{cc}} \cdot s_i$ by removing r_i.

Upload.
For $i = 1, 2, \ldots, m$, each user P_i encrypts its own private input x_i by the LE scheme. Firstly, P_i samples $e_i \leftarrow \chi^n$ and computes $c_0^i := \langle p_i, e_i \rangle + x_i \in R_q$ and $c_1^i := \langle a_i, e_i \rangle \in R_q$. Then, it outputs the ciphertext $c_i := (c_0^i, c_1^i) \in R_q^2$.

Outsourcing Computation.
After receiving all ciphertexts of the private inputs from users, SC stores all ciphertexts and executes a midtransformation to the outsourced data when computing some function $f(\cdot)$. After that, CC further transforms the midtransformed ciphertexts and then computes the function $f(\cdot)$ following the circuit $\mathscr{C}_f$ that consisted of addition gates and multiplication gates.

(1) Midtransforming.
Firstly, SC midtransforms the ciphertexts encrypted by users' own keys as $c_i \rightarrow c_i{}'$, where $c_i{}' = (c_0^{i}{}', c_1^{i}{}') = (k_{\text{sc}} \cdot c_0^i, k_{\text{sc}} \cdot (k_{\text{cc}} \cdot s_i) \cdot c_1^i)$, and sends $c_i{}'$ to CC.

(2) Computing.
After receiving $c_i{}'$, CC further transforms $c_i{}'$ to $c_i{}'' = (k_{\text{cc}} \cdot k_{\text{sc}} \cdot c_0^i, k_{\text{sc}} \cdot (k_{\text{cc}} \cdot s_i) \cdot c_1^i)$. Denote $k = k_{\text{sc}} \cdot k_{\text{cc}}$; then, $c_i{}'' = (c_0^{i}{}'', c_1^{i}{}'') = (k \cdot c_0^i, k \cdot s_i \cdot c_1^i)$. CC then computes the ciphertext of the result by the transformed ciphertexts of users' private inputs.
Add. For each addition gate, $c_i{}'' \oplus c_j{}'' = (c_1^{i}{}'' - c_0^{i}{}'') \oplus (c_1^{j}{}'' - c_0^{j}{}'') = k \cdot (x_i + x_j)$.
Mul. For each multiplication gate, $c_i{}'' \otimes c_j{}'' = (c_1^{i}{}'' - c_0^{i}{}'') \otimes (c_1^{j}{}'' - c_0^{j}{}'') = k^2 \cdot (x_i \times x_j)$.

(3) Producing Custom-Made Result.
After computing gate by gate following the circuit $\mathscr{C}_f$, CC obtains the intermediate result encrypted by the private numbers of the two assisted cloud servers SC and CC; that is, $y' = k^{l+1} \cdot y = k_{\text{sc}}{}^{l+1} \cdot k_{\text{cc}}{}^{l+1} \cdot y$, where $y = f(x_1, x_2, \ldots, x_m)$ and l is the number of the multiplication gates of $\mathscr{C}_f$. To produce a custom-made result for each user, CC firstly sends y' to SC. SC removes k_{sc}^{l+1} and adds r_t to compute $y_t{}' = r_t \cdot k_{\text{cc}}^{l+1} \cdot y$ and then sends $y_t{}'$ back to CC. CC finally removes k_{cc}^{l+1} to produce the custom-made ciphertext $y_t = r_t \cdot y$ and sends it to the authorized party P_t, $t \in \{1, 2, \ldots, m\}$.

Output
For each authorized party P_t, $t \in \{1, 2, \ldots, m\}$, it obtains the result y by removing r_t.

Algorithm 2

In setup, each user P_i invokes KeyGen(1^k) to compute its public-private keys (pk_i, sk_i). At the same time, each P_i selects a random r_i and sends it to SC via secure channels, while SC and CC, respectively, choose private numbers k_{sc} and k_{cc}. Assuming that all users' private data x_i, $i = 1, 2, \ldots, m$, are the real inputs of function $f(\cdot)$, then each P_i sends $r_i \cdot s_i$ to CC. CC further computes $k_{\text{cc}} \cdot r_i \cdot s_i$ and sends it to SC. After that, P_i submits the ciphertext c_i of the private input x_i encrypted by its own public key p_i to SC in the upload process.

In computation process, SC firstly midtransforms the outsourced data which are encrypted by different keys of users, and CC further transforms the midtransformed ciphertexts to the ones that are blinded by the same private numbers of the two servers so that CC can operate on the ciphertexts to compute $f(\cdot)$. Specifically, for addition/multiplication gate, CC can easily get the result by $(c_1^{i\prime\prime} - c_0^{i\prime\prime}) \oplus (c_1^{j\prime\prime} - c_0^{j\prime\prime}) = k \cdot (x_i + x_j)$ and $(c_1^{i\prime\prime} - c_0^{i\prime\prime}) \otimes (c_1^{j\prime\prime} - c_0^{j\prime\prime}) = k^2 \cdot (x_i \times x_j)$. Computing gate by gate following the circuit $\mathscr{C}_f$, CC can obtain the intermediate result $y' = k^{l+1} \cdot y = k_{\text{sc}}^{l+1} \cdot k_{\text{cc}}^{l+1} \cdot y$.

In order to guarantee that only the authorized user can get the final result, SC and CC cooperatively produce a custom-made result for each authorized user as follows. (Herein, we assume that P_t, $t \in \{1, 2, \ldots, m\}$, is authorized to get the result.) Firstly, CC sends y' to SC. SC removes k_{sc}^{l+1} and adds r_t to compute $y_t' = r_t \cdot k_{\text{cc}}^{l+1} \cdot y$ and then sends y_t' back to CC. CC finally removes k_{cc}^{l+1} to obtain the custom-made ciphertext $y_t = r_t \cdot y$ and sends it to P_t.

In the last process, the authorized user P_t obtains the result y by removing r_t from y_t.

5. Analysis

From the protocol described above, the correctness is obvious due to the homomorphic properties of the transformed ciphertexts. We will have a detailed discussion on its security. Note that, before the actual computations which are performed by SC and CC, there are setup and upload processes. We will individually illustrate their security at first. Afterwards, we will prove the security of the core of our protocol, that is, the outsourcing computation process, in the real-ideal framework. Finally, from the composition theorem [5], we can conclude that our protocol is secure.

Theorem 1. *Protocol Π is secure as long as the LE scheme is secure and SC and CC are noncolluding.*

Proof. Firstly, we look at the setup and upload processes individually.

In setup, each user, respectively, encrypts its private input by its own public key which is produced by invoking a semantically secure LE scheme. The security of this process is obvious. Afterwards, P_i sends $r_i \cdot s_i$ to CC and CC sends $k_{cc} \cdot r_i \cdot s_i$ to SC. Herein, P_i's private key s_i is protected by the blinding factors: r_i which is private to P_i and SC and k_{cc} which is private to CC. Therefore, the private keys of users will not be revealed in this process.

In upload, users outsource the encrypted data to SC. Since the LE scheme is semantically secure, given two ciphertexts $c_i(m_1)$, $c_i(m_2)$ of the two plaintexts m_1, m_2 uploaded by P_i, it is computationally infeasible for SC to distinguish the two ciphertexts. Hence, users can store their encrypted data in SC securely.

In outsourcing computation process, SC firstly midtransforms c_i to c_i' and sends c_i' to CC. It is obvious that it is secure since SC blinds the midtransformed ciphertext c_i' by the private number k_{sc}, which is secret to CC. As to the core of the computation process, we will discuss the security in the real-ideal framework. From the security definition, we say that protocol Π is secure if all adversarial behavior in the real world can be simulated in the ideal model where there exists an additional trusted party to perform all computations related to the function $f(\cdot)$ to be computed. We assume that there is a simulator $\mathcal{S}$ in the ideal world and then prove that it can simulate the semihonest adversary $\mathcal{A}$ that exists in the real execution. Since CC is able to independently complete addition and multiplication operations, we only need to prove that Add and Mul are secure against the semihonest adversary $\mathcal{A}$ corrupting CC. We prove this as follows.

Simulator $\mathcal{S}$. Run $\mathcal{A}$ on input$\{c_{\mathcal{S}}(m_1), c_{\mathcal{S}}(m_2)\}$.

Firstly, $\mathcal{S}$ computes

$$\begin{gathered} c_{\mathcal{S}}(m_1) = \operatorname{Enc}(pk_{\mathcal{S}}, 1); \\ c_{\mathcal{S}}(m_2) = \operatorname{Enc}(pk_{\mathcal{S}}, 1) \end{gathered} \tag{1}$$

and sends $c_{\mathcal{S}}(m_1)$, $c_{\mathcal{S}}(m_2)$ to $\mathcal{A}$.

Secondly, $\mathcal{A}$ sends two ciphertexts $c_{\mathcal{S}}(m_1^*)$, $c_{\mathcal{S}}(m_2^*)$ to $\mathcal{S}$. Then, $\mathcal{S}$ computes

$$\begin{gathered} c_{\mathcal{S}}(m_1^* + m_2^*) = c_{\mathcal{S}}(m_1^*) \oplus c_{\mathcal{S}}(m_2^*); \\ c_{\mathcal{S}}(m_1^* \times m_2^*) = c_{\mathcal{S}}(m_1^*) \otimes c_{\mathcal{S}}(m_2^*) \end{gathered} \tag{2}$$

and returns $c_{\mathcal{S}}(m_1^* + m_2^*)$, $c_{\mathcal{S}}(m_1^* \times m_2^*)$ to $\mathcal{A}$.

Finally, $\mathcal{S}$ outputs what $\mathcal{A}$ outputs.

Now, we can prove the security of Add and Mul algorithms by contradiction. Firstly, we assume that the view of the adversary $\mathcal{A}$ in the real world is distinguishable from the view simulated by the simulator $\mathcal{S}$. Then, we could find an algorithm to distinguish the ciphertexts encrypted by the LE encryption scheme, which is contrary to our assumption that the LE is semantically secure. Hence, the view of the adversary $\mathcal{A}$ in the real world is indistinguishable from the view simulated by the simulator $\mathcal{S}$. That is,

$$\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}}(c_{\mathcal{S}}(m_i)) \overset{c}{\approx} \mathrm{REAL}_{\Pi,\mathcal{A}}(c_{\mathcal{S}}(m_i)), \quad i = 1, 2. \tag{3}$$

Therefore, the two algorithms Add and Mul are secure. Furthermore, from the composition theorem [5], we can conclude that our protocol is secure as long as the LE scheme is secure and SC and CC are noncolluding in semihonest scenario. □

6. Conclusion

Only contributing the encrypted forms of their private inputs under their own public keys to gain the desired result of some function on the private inputs via powerful cloud with minimal computations and communications is the optimal method especially when users want to compute some complex function. In this paper, we introduce two noncolluding cloud servers to construct a secure outsourcing multiparty computation protocol on lattice-based encrypted cloud data under multiple keys in semihonest scenario. All computations related to the computing function are in the charge of the two cloud servers. Therefore, the computation complexity of each user only depends on the encryption scheme it has used. What is more, the communication complexity of each user is also independent of the function to be computed and there is no interaction of users whatsoever any more.

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 NSFC (Grants nos. 61300181, 61272057, 61202434, 61170270, 61100203, and 61121061) and the Fundamental Research Funds for the Central Universities (Grants nos. 2012RC0612 and 2011YB01).

References

[1] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 160–164, Chicago, Ill, USA, 1982.

[2] Y. Lindell and B. Pinkas, "A proof of security of yao's protocol for two-party computation," *Journal of Cryptology*, vol. 22, no. 2, pp. 161–188, 2009.

[3] O. S. Goldreich, S. Mical, and A. Wigderson, "How to play any mental game," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing (STOC '87)*, pp. 218–229, New York, NY, USA, 1987.

[4] O. S. Goldreich, "Secure multiparty computation," Manuscript, Preliminary version, 1998.

[5] O. S. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*, Cambridge University press, 2004.

[6] M. M. Prabhakaran and A. Sahai, Eds., *Secure Multiparty Computation*, IOS Press, 2013.

[7] R. Fagin, M. Naor, and P. Winkler, "Comparing information without leaking it," *Communications of the ACM*, vol. 39, pp. 77–85, 1996.

[8] D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols," in *Proceedings of the 20th annual ACM symposium on Theory of computing (STOC '88)*, pp. 11–19, 1988.

[9] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology—CRYPTO 2012*, vol. 7417 of *Lecture Notes in Computer Science*, pp. 643–662, Springer, 2012.

[10] Y. Lindell and B. Pinkas, "An efficient protocol for secure two-party computation in the presence of malicious adversaries," in *Advances in Cryptology—EUROCRYPT 2007*, vol. 4515 of *Lecture Notes in Computer Science*, pp. 52–78, Springer, 2007.

[11] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two-party computation is practical," in *Advances in Cryptology—ASIACRYPT 2009*, vol. 5912 of *Lecture Notes in Computer Science*, pp. 250–267, Springer, 2009.

[12] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[13] T. Velte, A. Velte, and R. Elsenpeter, *Cloud Computing, a Practical Approach*, McGraw-Hill, 2009.

[14] J. Loftus and N. P. Smart, "Secure outsourced computation," in *Progress in Cryptology—AFRICACRYPT 2011*, pp. 1–20, Springer, Berlin, Germany, 2011.

[15] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communication Security (ASIACCS '10)*, pp. 48–59, April 2010.

[16] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-Party computation," *IACR Cryptology ePrint Archive*, vol. 2011, article 272, 2011.

[17] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," *IACR Cryptology ePrint Archive*, vol. 2013, article 13, 2013.

[18] B. Wang, M. Li, M. Chow et al., "Computing encrypted cloud data efficiently under multiple keys," in *Proceedings of the IEEE Conference on Communications and Network Security (CNS '13)*, pp. 504–513, 2013.

[19] M. Van Dijk and A. Juels, "On the impossibility of cryptography alone for privacy-preserving cloud computing," in *Proceedings of the 5th USENIX conference on Hot topics in security (HotSec '10)*, pp. 1–8, 2010.

[20] C. Gentry, *A fully homomorphic encryption scheme [Doctoral dissertation]*, Stanford University, 2009.

[21] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, "Multiparty computation with low communication, computation and interaction via threshold fhe," in *Advances in Cryptology—EUROCRYPT 2012*, Lecture Notes in Computer Science, pp. 483–501, Springer, Berlin, Germany, 2012.

[22] S. Halevi, Y. Lindell, and B. Pinkas, "Secure computation on the web: computing without simultaneous interaction," in *Advances in Cryptology—CRYPTO 2011*, pp. 132–150, Springer, 2011.

[23] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS '11)*, pp. 97–106, 2011.

[24] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "Cloud-assisted multiparty computation from fully homomorphic encryption," *IACR Cryptology ePrint Archive*, vol. 2011, article 663, 2011.

[25] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Advances in Cryptology—CRYPTO 2011*, vol. 6841 of *Lecture Notes in Computer Science*, pp. 505–524, 2011.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 746260, 8 pages
http://dx.doi.org/10.1155/2014/746260

Research Article

Path Planning Using a Hybrid Evolutionary Algorithm Based on Tree Structure Encoding

Ming-Yi Ju, Siao-En Wang, and Jian-Horn Guo

Department of Computer Science and Information Engineering, National University of Tainan, Tainan 70005, Taiwan

Correspondence should be addressed to Ming-Yi Ju; myju@mail.nutn.edu.tw

Received 21 February 2014; Accepted 8 April 2014; Published 28 May 2014

Academic Editors: V. Bhatnagar and Y. Zhang

Copyright © 2014 Ming-Yi Ju 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 evolutionary algorithm using scalable encoding method for path planning is proposed in this paper. The scalable representation is based on binary tree structure encoding. To solve the problem of hybrid genetic algorithm and particle swarm optimization, the "dummy node" is added into the binary trees to deal with the different lengths of representations. The experimental results show that the proposed hybrid method demonstrates using fewer turning points than traditional evolutionary algorithms to generate shorter collision-free paths for mobile robot navigation.

1. Introduction

Path planning is one of the important research issues in mobile robot. When executing a task, the robot is supposed to plan optimal or feasible paths to avoid obstacles in its way and to minimize cost such as time, energy, and distance. Accordingly, path planning can be regarded as an issue in optimization.

A general solution to optimization problem is the use of evolutionary algorithm because of the model-free characteristic. When dealing with an optimization problem, evolutionary algorithm uses some mechanisms inspired by biological evolution to find the approximate solutions. The most famous evolutionary algorithm is genetic algorithm (GA) [1]. Another technology for solving the optimization problem is swarm intelligence. Similarly, the swarm intelligence is inspired from artificial life research. The most famous swarm intelligence is particle swarm optimization (PSO) [2–4]. One of the advantages of GA is its mechanism of probabilistic and useful exchange of information among chromosomes to find an optimal solution, but it is a time-consuming process. In contrast to GA, the advantage of PSO is that each particle's movement is influenced by its local best known position but is also guided towards the best known positions in the search-space to accelerate the convergence. However, PSO does not guarantee that an optimal solution is ever found. Therefore, the idea of integrating two techniques to solve the path planning problem is feasible. Although a number of studies have been made on hybrid GA and PSO, the problem of hybrid algorithms still needs to be improved on the encoding method. Many studies use the fixed number of turning points to represent the path without considering the environmental complexity and then lead to poor path quality. The main reason is due to the solution dimension of PSO which must be fixed. However, the number of turning points used to encode a path should depend on the complexity of the environment. More turning points are needed to accomplish the plan of making a collision-free path in cluttered environments.

This paper adopts a scalable encoding method and develops a novel approach to solve the problem of hybrid GA and PSO. The scalable representation is based on binary tree structure encoding. Each binary tree represents a path. The path can be acquired by tracing the binary tree using the in-order traversal. In the binary tree, each node represents a turn in the path. With more obstacles in the workspace, the binary tree has more nodes. With no obstacles in the workspace, the binary tree only has the starting point and the end point. The number of the nodes depends on the complexity of the environment. The proposed binary tree structure also can solve the learning problem in PSO.

The remainder of this paper is organized as follows. Section 2 introduces the related work to hybrid evolutionary algorithm. Section 3 elaborates the proposed hybrid GA-PSO approach as well as a brief explanation of how the binary tree is built. Section 4 illustrates the experimental results for comparing the proposed hybrid approach with traditional GA and PSO. Concluding remark and future work are presented in Section 5.

2. Related Work

The methods for hybrid GA and PSO can be categorized to parallel hybrid [5, 6] and series hybrid [7]. In parallel hybrid strategy, GA and PSO are two independent subsystems. After the evolutionary process starts, the two subsystems execute simultaneously. The main system will check whether the best solution in the two subsystems satisfies the termination criterion or not. If the generations can be divided by the designated iterative times, the main system will perform the hybrid process. In the hybrid process, the main system selects a fixed number of individuals from both subsystems randomly. In contrast, series hybrid strategy uses a series of connections to cascade GA and PSO. In each generation, every individual is sorted according to its fitness value. Only the top-half best-performing individuals, called elites, in each generation are used as parents to generate offspring. The learning mechanism of PSO is adopted to enhance elites. The crossover operator and the mutation operator are then applied to the elites to produce new individuals for the next generation.

We can easily tell the principal differences between these two categories of hybrid methods. Parallel hybrid method generates two different subpopulations by GA and PSO individually for solving a problem. The communication between GA and PSO only exchanges the individuals. In contrast to parallel hybrid, series hybrid uses the same population to search for a better candidate solution iteratively. Series hybrid method applies all the GA and PSO operations on the same population. However, the coding way of all hybrid methods still uses fixed number of turning points to represent a candidate solution for path planning. The required number of turning points in a path should depend on the complexity of the environment. The encoding way using fixed number of turning points is not appropriate. In [8, 9], the traditional grid-based encoding way is used to represent the candidate paths. In contrast to the grid-based encoding, [10] demonstrates a new approach to represent the path based on binary tree structure. Although there are many studies on combining GA and PSO, the main solution to scalable encoding way still remains unsatisfactory.

3. Methodology

The flowchart of the proposed hybrid evolutionary approach is shown in Figure 1. First, the initial paths and the corresponding binary trees are created. Then, every path is evaluated according to its fitness value. After getting the fitness value, the system sorts the paths from the best to the worst. The top half paths, called elites, are saved and the worst half paths are deleted. The elite paths are enhanced by the PSO operator. Because worst half paths are deleted, the crossover operator is applied to create new paths. When applying the crossover operator, the parents are selected from the elite paths. And each new created path has probability to apply the mutation operator of GA. Finally, the system will check each path to find whether it meets the criteria or not. If no path meets the criteria, the system returns to the evaluating step.

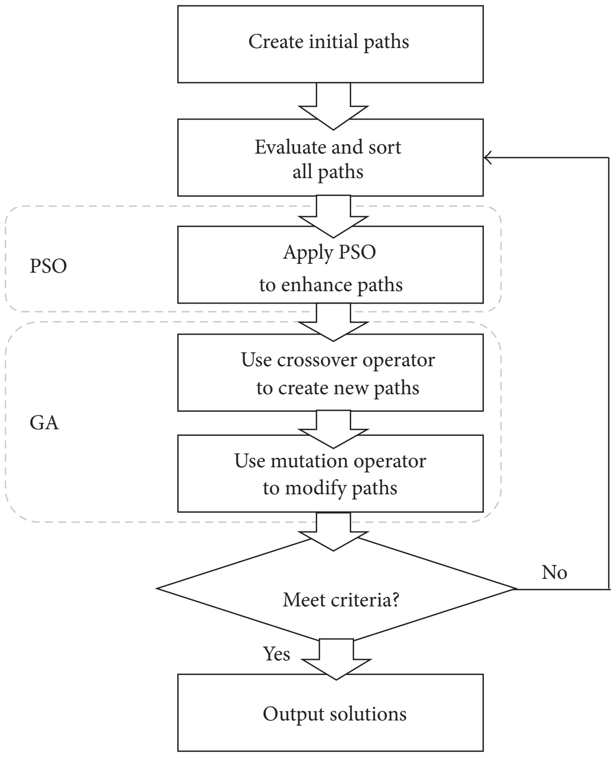

Create initial paths
Evaluate and sort all paths
PSO
Apply PSO to enhance paths
GA
Use crossover operator to create new paths
Use mutation operator to modify paths
Meet criteria?
No
Yes
Output solutions

Figure 1: The flowchart of hybrid evolutionary algorithm.

In order to make the hybrid evolutionary algorithm work properly, a good encoding method to combine two kinds of evolutionary algorithms is needed. The proposed hybrid method applies the binary tree structure and adds some dummy nodes to solve the hybrid problem of GA and PSO.

3.1. The Creation of Initial Paths. This part describes how to generate a binary tree to represent a path. As shown in Figure 2(a), there is an obstacle between the starting point (S) and the end point (G). First of all, a direct line would be used to connect the starting point and the end point. If there is no obstacle on the line, this line will be the shortest path. If there are obstacles on the line, a new turning point with a random offset distance d perpendicular to the line segment would be generated from the middle of the line. An obstacle-free path will be created as shown in Figure 2(b). There are two directions for a line to create a new node, as shown in Figure 3. Every time a new turning point is created, the system will check whether every line segment is obstacle-free or not. If the path is still not obstacle-free, a new turning point will be created until the path can avoid every obstacle. If the number of turning points exceeds the

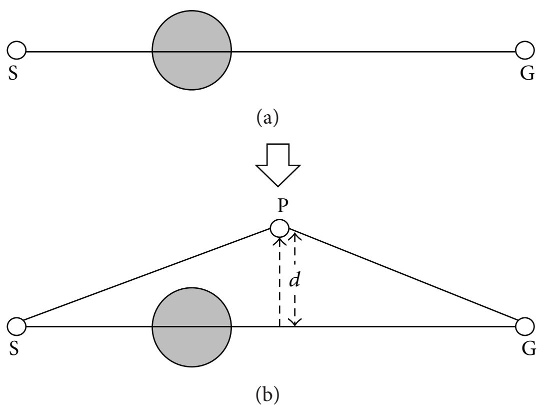


Figure 2: A simple example of creating a collision-free path.

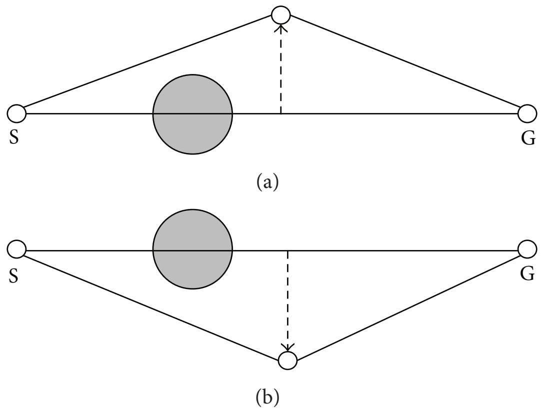


Figure 3: Two directions for a path to create a new turning point.

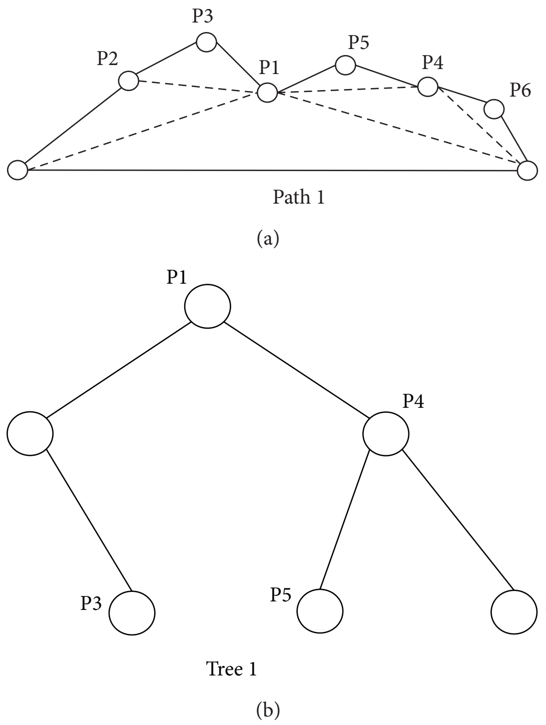


Figure 4: A path with 6 turning points and its corresponding binary tree.

predefined threshold, new turning points will not be created anymore. A path with the maximum number of turning points is not a good candidate solution and will be deleted after the sorting process. As shown in Figure 4, all the turning points of a path are encoded to a binary tree for the process of evolution [10].

3.2. Operator of Particle Swarm Optimization. Since PSO does not allow solutions with different dimensions, a novel method is proposed to solve the hybrid problem based on the scalable binary tree structure. Figures 4 and 5 illustrate two different paths and their binary tree representation, respectively. Figures 4(a) and 5(a) show the original paths, and Figures 4(b) and 5(b) represent the corresponding binary trees. Two binary trees have different node numbers and different attitudes. The updating mechanism of PSO cannot apply to two paths with different node numbers. The concept of the proposed hybrid approach is the use of "dummy nodes." The dummy nodes are created and added when two binary trees have different node numbers in the updating step.

In Figure 6, tree 1 has 6 nodes and tree 2 has 9 nodes. Tree 1 will generate four dummy nodes and tree 2 will generate one dummy node, as shown in Figure 6. At this step, tree 1 and tree 2 have the same attitude. The dummy node is different from the common node because the dummy node is a "null" node and the offset distance d of dummy node is zero. In the corresponding path of tree 1 and tree 2 as shown in Figure 7, the gray nodes are dummy nodes. According to the updating mechanism of PSO, the path with lower fitness will be influenced by its local best known position but at the same time is guided towards the best known positions in the search space as well. Therefore, the attitude of the path will be similar to the path from which it learns.

Consider that path 1 is the global best solution. As shown in Figure 7, S → P2 segment in path 1 and S → P4 → P3 → P5 → P2 segment in path 2; the P4, P3, and P5 in path 2 learn from dummy nodes in path 1 so their offset distances will approach zero and the path length can be reduced. After the updating mechanism step, those dummy nodes will be deleted.

3.3. Operators of Genetic Algorithm

3.3.1. Crossover Operator. Crossover is a way to produce chromosomes diversity in genetic algorithm. Two individuals can exchange information by applying crossover. In binary tree structure encoding, we exchange the subtree or node to accomplish the crossover operator.

(a) Swap Subtree Crossover Operator. First, select two paths randomly. Then, select a node randomly from two paths, respectively. Secondly, exchange two subtrees. The coordinates will be computed again after exchange, in order to update their current correct location. As shown in Figure 8, both tree 1 and tree 2 randomly select subtrees to exchange.

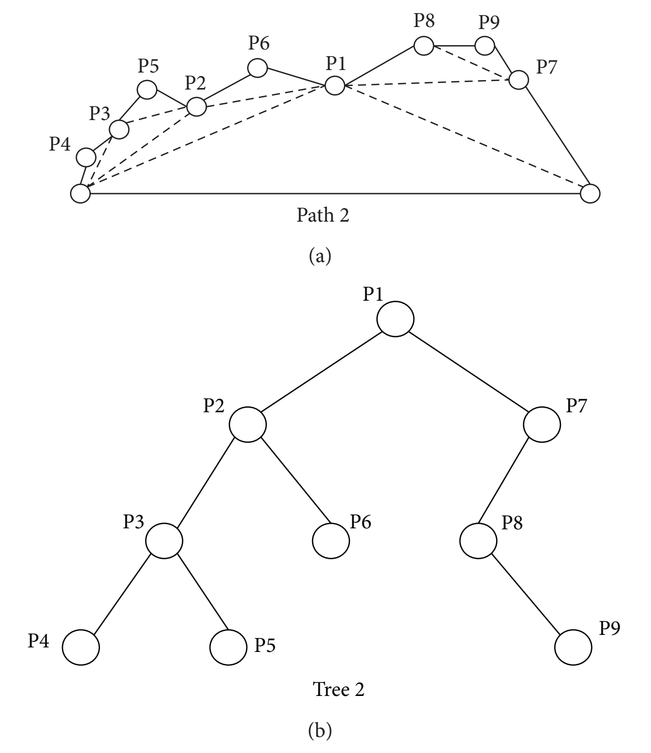


Figure 5: A path with 9 turning points and its corresponding binary tree.

The corresponding path applying swap subtree crossover operator is shown in Figure 9.

(b) Generate New Individuals Crossover Operator. The process is the same as swap subtree crossover operator. First, two paths (tree 1 and tree 2) will be selected randomly. Then, select a node in two trees, respectively. Second, the front part of tree 1 will be combined with the subtree of tree 2 to generate a new tree 3, as shown in Figure 10. And the corresponding path is shown in Figure 11. This operator will produce a new path. Therefore, this operator is used to make up the number of generations.

(c) One Point Swap Crossover Operator. The difference between swap subtree crossover operator and one point crossover operator is that the latter only exchanges one node. A simple example is shown in Figure 12 and the corresponding path is shown in Figure 13.

3.3.2. Mutation Operator. Mutation is another way to generate chromosomes diversity in GA. This operator will be applied only when a new path is generated. Mutation changes the contents of a single chromosome so that the path may have a greater variability. Therefore, the system may generate higher probability to find a better solution. Of course, this variation also may make an individual worse. Each node has the possibility to mutate except at start point and end point.

(a) Perturb Mutation Operator. The function of perturb mutation is to change the offset distance d of a tree node.

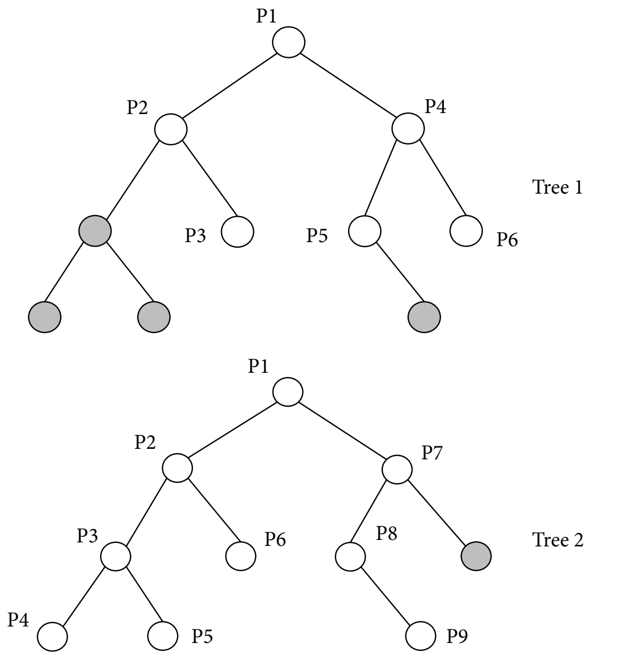


Figure 6: After adding the dummy nodes, symbolized as the grey circles, the two different binary trees have the same attitude.

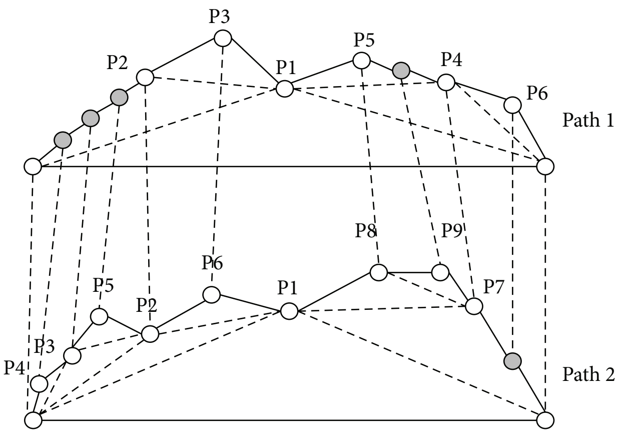


Figure 7: The corresponding path of Figure 6.

The new offset distance d is generated randomly. A simple example is shown in Figure 13. The gray node becomes farther from the original path than before but the direction does not change.

(b) Flip Mutation Operator. Flip mutation is similar to the perturb mutation but changes the direction of the offset distance d. A simple example is shown in Figure 14. The gray node only change the direction but the distance d does not change.

4. Experimental Results and Analysis

In order to evaluate the performance of the proposed hybrid approach, three different types of test environments are used

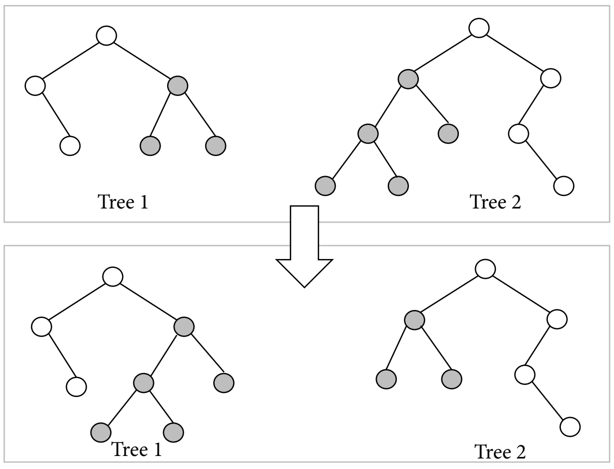


Figure 8: Apply swap subtree crossover operator to generate the offspring from the given parents.

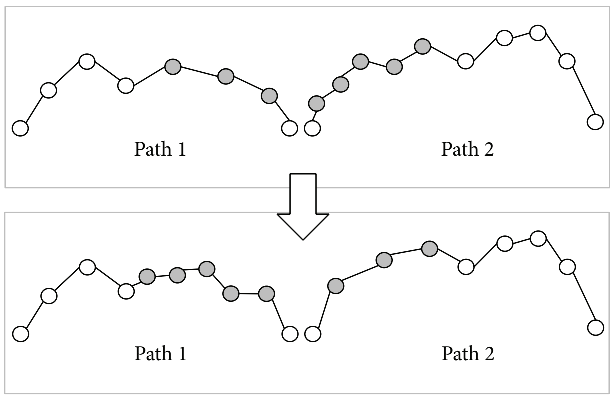


Figure 9: The corresponding paths of Figure 8.

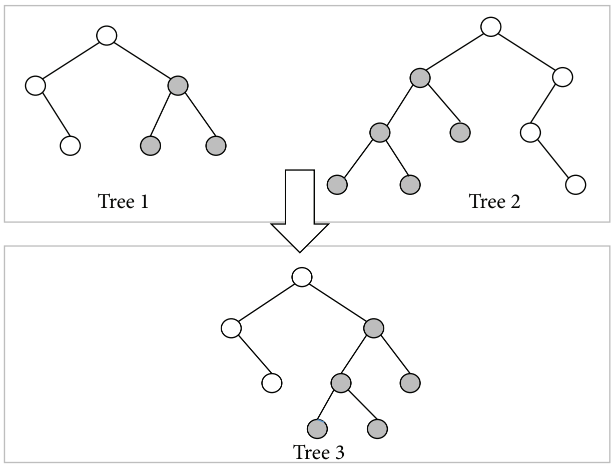


Figure 10: Apply individual crossover operator to generate a new offspring from the given parents.

in our experiments. Table 1 illustrates the control parameters used in those experiments. The number of population size is 30, which represents 30 individuals to perform the evolution. The maximum number of generations is set to 2,000. If the evolution reaches 2,000 generations and does not satisfy the termination threshold, the evolution will be stopped. Mutation probability is set as 0.1%. The inertia weight and learning factors of the PSO are given as 0.5 and 1.3, respectively. The fitness value is defined as the length of a path plus the penalty for collision condition. The penalty is set to 1,000 as collision occurrence; otherwise, the penalty is given as zero. According to different experimental environments, we set the different starting points and ending points.

Figure 15 displays a workspace with 20 overlapped obstacles. In this experiment, the starting point is set at (0, 0) and the ending point at (200, 200). To emphasize genetic length variability of the proposed hybrid method, we use two different tree sizes, 7 and 15 nodes, to perform comparison. Having more nodes can mean having more opportunities to avoid obstacles, but more nodes may also cause unnecessary detour. Table 2 shows the results for the GA and PSO with 7 tree nodes. According to the results, the proposed hybrid method gets a large advantage on effective path, although the proposed hybrid algorithm loses in the shortest path length of PSO. However, the result indicates that the proposed method is more stable to find a valid path. As the results of GA and PSO with 15 nodes shown in Table 3, the proposed hybrid method and GA won PSO in the valid number of paths.

The second test environment, shown in Figure 16, for the path planning problem is one that is commonly used for performance evaluation. In this form of obstacle distribution, the proposed hybrid method can produce more numbers of effective paths than the other two algorithms. Experimental results are illustrated in Tables 4 and 5. It is noticeable that the number of effective paths generated by the proposed hybrid method is substantially better than GA and PSO.

The difference between the third experiment environment, shown in Figure 17, and the previous ones is that the

Table 1: Experiment parameters.

Name	Value	Comment
Population size	30	
Maximum generation allowed	2000	Termination criterion I
Fitness threshold	10	Termination criterion II: if the change of best fitness is smaller than the predefined threshold by 10 times, terminate the evolutionary process
Crossover probability	1	For GA
Mutation probability	0.001	For GA
Inertia weight	0.5	For PSO
Learning factors	1.3	For PSO

obstacles do not overlap in the test environment, and so the obstacles take up more proportion on the map. As shown in Table 6, whether in the mean length of effective paths or

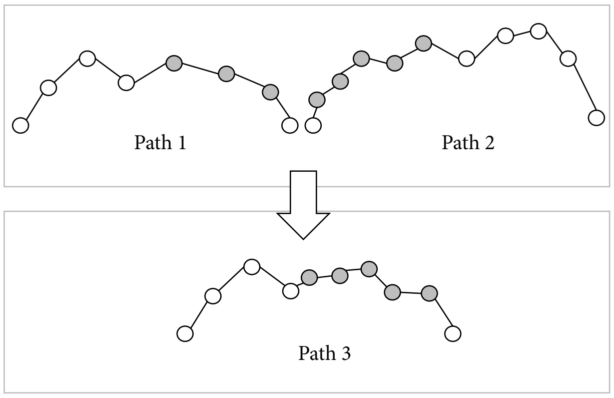


Figure 11: The corresponding paths of Figure 10.

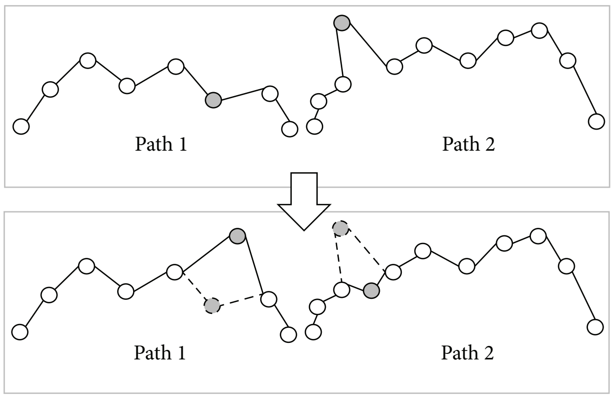


Figure 12: Apply one point swap crossover operator to generate offspring.

Table 2: Comparison in performance among the proposed hybrid approach, PSO, and GA for test environment I.

	Hybrid	PSO	GA
Initial tree size (number of nodes)	≤7	7	7
Number of valid paths	2979	2730	2934
Mean length of valid paths	311.9088	306.2684	314.4024
Standard deviation of valid paths	21.0225	13.9147	21.4342
Shortest valid path	295.5128	295.2754	295.5218
Average solution tree size (average number of nodes)	3.3283	7	7

the number of valid paths, the results of the proposed hybrid method are better than the other two algorithms.

5. Conclusion and Future Work

This study describes a method using binary tree structure encoding to hybrid genetic algorithm and particle swarm optimization for solving the path planning problems. Dummy nodes are added to the binary trees to deal with the different lengths of solution/chromosome problem for

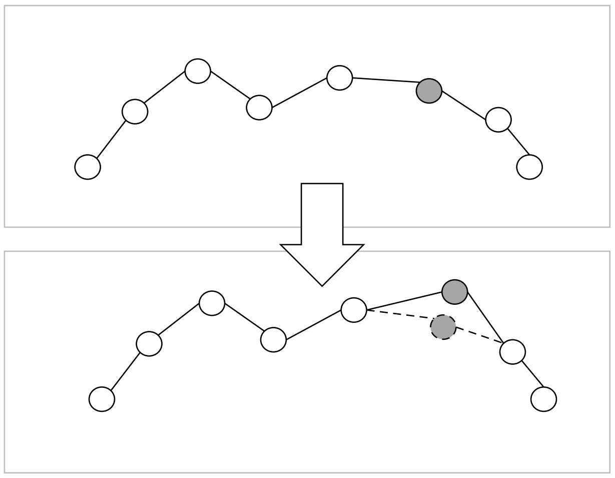


Figure 13: Apply perturb mutation operator to generate offspring.

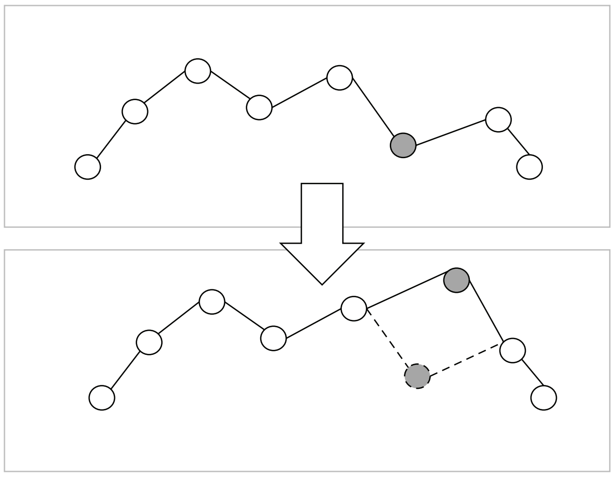


Figure 14: Apply flip mutation operator to generate offspring.

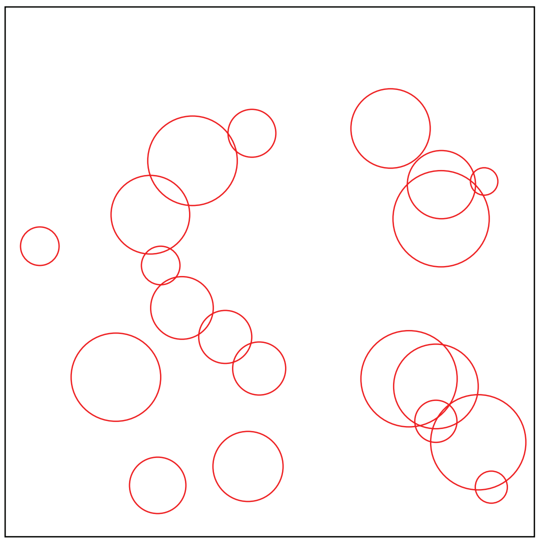


Figure 15: Test environment I.

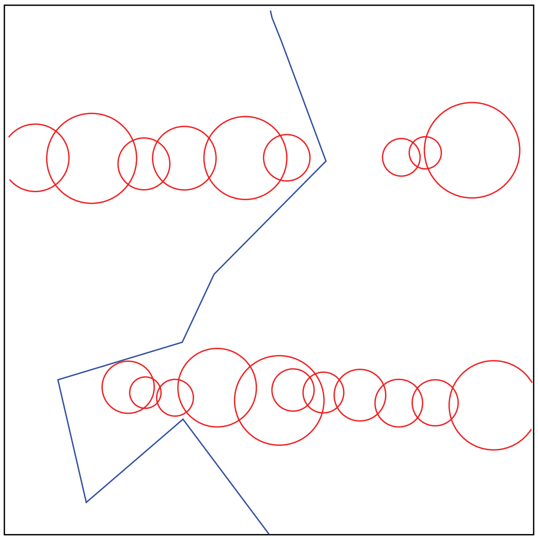


Figure 16: Test environment II.

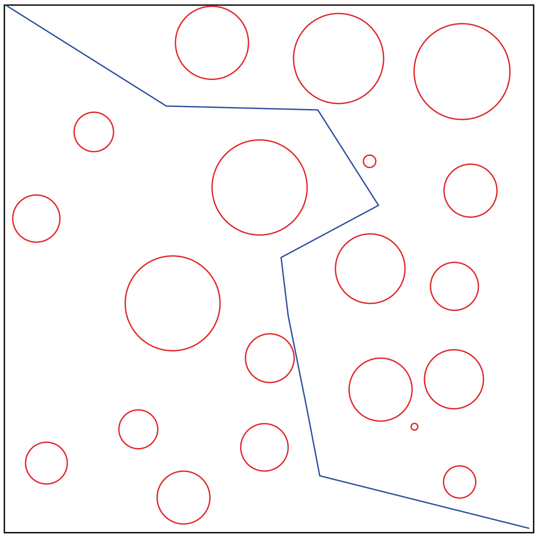


Figure 17: Test environment III.

Table 3: Comparison in performance among the proposed hybrid approach, PSO, and GA for test environment I.

	Hybrid	PSO	GA
Initial tree size (number of nodes)	≤15	15	15
Number of valid paths	2996	2931	2996
Mean length of valid paths	314.3656	311.4129	315.1283
Standard deviation of valid paths	21.8718	19.0713	21.4997
Shortest valid path	295.3884	294.8781	295.6429
Average solution tree size (average number of nodes)	3.2774	15	15

PSO. The experimental results show that the proposed hybrid method uses fewer turning points than the traditional grid based methods and the number of obstacle-free paths is generated more than traditional evolutionary algorithms. In the future, we will improve this method to generate a curve path so that the method can be applied to the system.

Table 4: Comparison in performance among the proposed hybrid approach, PSO, and GA for test environment II.

	Hybrid	PSO	GA
Initial tree size (number of nodes)	≤7	7	7
Number of valid paths	2032	307	211
Mean length of valid paths	326.1601	300.8695	316.1638
Standard deviation of valid paths	31.3879	14.8971	17.1472
Shortest valid path	274.7688	283.5653	289.0466
Average solution tree size (average number of nodes)	5.9975	7	7

Table 5: Comparison in performance among the proposed hybrid approach, PSO, and GA for test environment II.

	Hybrid	PSO	GA
Initial tree size (number of nodes)	≤15	15	15
Number of valid paths	2620	1410	952
Mean length of valid paths	370.1507	304.9307	320.2028
Standard deviation of valid paths	84.7093	21.1082	20.7493
Shortest valid path	275.2839	270.8929	279.0292
Average solution tree size (average number of nodes)	7.3294	15	15

Table 6: Comparison in performance among the proposed hybrid approach, PSO, and GA for test environment III.

	Hybrid	PSO	GA
Initial tree size (number of nodes)	≤15	15	15
Number of valid paths	3000	2982	2966
Mean length of valid paths	317.2784	317.6262	323.0512
Standard deviation of valid paths	22.9993	23.6171	24.3007
Shortest valid path	284.6030	285.6138	286.2925
Average solution tree size (average number of nodes)	4.52167	15	15

Conflict of Interests

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

References

[1] J. H. Holland, *Adaptation in Natural and Artifical Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.

[3] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 1995 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, October 1995.

[4] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the Congress on Evolutionary Computation 2001*, vol. 1, pp. 81–86, May 2001.

[5] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and L. M. Wang, "An improved GA and a novel PSO-GA-based hybrid algorithm," *Information Processing Letters*, vol. 93, no. 5, pp. 255–261, 2005.

[6] Y.-T. Kao and E. Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions," *Applied Soft Computing Journal*, vol. 8, no. 2, pp. 849–857, 2008.

[7] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 34, no. 2, pp. 997–1006, 2004.

[8] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, vol. 5, pp. 4350–4355, May 2004.

[9] J. Tu and S. X. Yang, "Genetic algorithm based path planning for a mobile robot," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 1221–1226, September 2003.

[10] C. Hocaoĝlu and A. C. Sanderson, "Planning multiple paths with evolutionary speciation," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 3, pp. 169–191, 2001.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 232714, 12 pages
http://dx.doi.org/10.1155/2014/232714

Research Article

Constrained Multiobjective Biogeography Optimization Algorithm

Hongwei Mo,[1] Zhidan Xu,[2] Lifang Xu,[3] Zhou Wu,[4] and Haiping Ma[5]

[1] *Automation College, Harbin Engineering University, Harbin 150001, China*
[2] *Institute of Basic Science, Harbin University of Commerce, Harbin 150028, China*
[3] *Engineering Training Center, Harbin Engineering University, Harbin 150001, China*
[4] *Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Gauteng 0028, South Africa*
[5] *Department of Electrical Engineering, Shaoxing University, Shaoxing, Zhejiang 312000, China*

Correspondence should be addressed to Hongwei Mo; honwei2004@126.com

Received 28 December 2013; Accepted 9 March 2014; Published 26 May 2014

Academic Editors: V. Bhatnagar and Y. Zhang

Copyright © 2014 Hongwei Mo 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.

Multiobjective optimization involves minimizing or maximizing multiple objective functions subject to a set of constraints. In this study, a novel constrained multiobjective biogeography optimization algorithm (CMBOA) is proposed. It is the first biogeography optimization algorithm for constrained multiobjective optimization. In CMBOA, a disturbance migration operator is designed to generate diverse feasible individuals in order to promote the diversity of individuals on Pareto front. Infeasible individuals nearby feasible region are evolved to feasibility by recombining with their nearest nondominated feasible individuals. The convergence of CMBOA is proved by using probability theory. The performance of CMBOA is evaluated on a set of 6 benchmark problems and experimental results show that the CMBOA performs better than or similar to the classical NSGA-II and IS-MOEA.

1. Introduction

In the domain of science and engineering, most of the problems are attributed to constrained multiobjective optimization problems (CMOPs), which need to optimize multiple conflicting objectives subject to various inequality and equality constraints. So the algorithms of solving CMOPs have to search the set of nondominated feasible solutions fulfilling all constraints. It is desirable that those gained solutions can approximate the true Pareto front with better diversity and even distribution. Evolutionary algorithms (EAs) are population-based search algorithms and can find multiple optimal solutions in one single run, and they are suitable to solve multiobjective problems (MOPs). But for the specific application of solving CMOPs, we find that most of the existing constrained multiobjective EAs (MOEAs) cannot effectively exploit the population because their obtained convergence and diversity are not acceptable.

Biogeography-based optimization (BBO) algorithm is a population-based search algorithm [1, 2], which had been applied to solve single objective optimization problems (SOPs) and some engineering problems [3–5]. In the aspect of MOPs, Ma et al. decomposed multiobjective optimization problems into several related subproblems and used parallel BBO to optimize each subproblem [6]. We successfully improved BBO for MOPs, which had proved that the migration strategy of BBO is effective for solving MOPs [7, 8]. In view of good population exploiting ability of BBO, in this study, we propose a novel constrained multiobjective biogeography optimization algorithm (CMBOA) for the first time and analyze its convergence by the probability theory.

The proposed CMBOA includes the following features. First, the individuals are classified into the feasible and infeasible ones based on their constraint violation. Second, feasible individuals are evaluated by combining their objective functions value and crowded distance. Infeasible individuals are evaluated by combining their constraint violation and Euclidean distance from the nearest nondominated feasible individual. Third, a new migration operator with additional disturbance is designed to generate diverse feasible solutions. And infeasible solutions nearby feasible regions are recombined with their nearest nondominated feasible solutions to evolve towards feasibility.

In rest of the paper, reviews of multiobjective evolutionary algorithms (MOEAs) for CMOPs are given in Section 2, and basic conception of CMOPs, the review of CMOPs, and the BBO are briefly introduced. The CMBOA is proposed in Section 3. In Section 4, compared with the classical algorithms on benchmark CMOPs, simulation results on CMBOA are analyzed and discussed. At last, conclusions are drawn in Section 5.

2. Related Background

2.1. Problem Statement. The aim of the constrained multi-objective optimization problems (constrained MOPs) is to find multiple nondominated solutions under constraints. If these nondominated solutions are uniformly distributed and widely spread along the Pareto front, their quality is better. Without the loss of generality, we consider the minimization of CMOPs, which can be defined as follows:

$$\begin{aligned} \min \quad & \mathbf{y} = F(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})] \\ \text{s.t} \quad & g_i(\mathbf{x}) \le 0, \quad i = 1, 2, \dots, q \\ & h_j(\mathbf{x}) = 0, \quad j = q+1, q+2, \dots, l \\ & \mathbf{x} = (x_1, x_2, \dots, x_n) \\ & x_i^{\min} \le x_i \le x_i^{\max}, \quad i = 1, 2, \dots, n, \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \dots, x_n) \in R^n$ is a decision vector with n decision variables. $\mathbf{y} = (f_1, f_2, \dots, f_m) \in R^m$ is an objective vector with m objects. Each dimension variable of the decision space is bounded by its upper bound $x_i^{\max}$ and lower bound $x_i^{\min}$. $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ are inequality constraints and equality constraint, respectively. The equality constraints generally should be transformed into inequality form and combined with other inequality constraints as follows:

$$G_j(\mathbf{x}) = \begin{cases} \max\{g_i(\mathbf{x}), 0\}, \\ \max\{| h_j(\mathbf{x}) - \delta, 0\}, \end{cases} \tag{2}$$

where $i = 1, 2, \dots, q$, $j = q+1, q+2, \dots, l$, and δ is a tolerance parameter for the equality constraint. In this paper, only CMOPs with inequality constraints are considered. Constraint violation function of a solution x is defined as follows:

$$v(x) = \sum_{i=1}^{n} (G_i(x))^2, \tag{3}$$

where $v(x) \ge 0$. If $v(x) > 0$, then x is an infeasible solution; otherwise, it is a feasible solution. By the degree of constraint violation, infeasible solutions can be compared with another one. For feasible solutions, Pareto domination is defined as follows, which is applied to evaluate their fitness.

Definition 1 (Pareto domination solution). Let $x, y \in R^n$, and a solution vector x is said to dominate a solution y and is denoted by $x \prec y$ if

$$\begin{gathered} \forall i \in \{1, 2, \dots, m\} : f_i(x) \le f_i(y), \\ \exists j \in \{1, 2, \dots, m\} : f_j(x) \le f_j(y). \end{gathered} \tag{4}$$

2.2. Reviews of CMOEA. Most of MOEAs are proposed for solving unconstrained multiobjective optimization [9]. According to different constraint handling methods adopted in MOEAs, the existing constrained multiobjective evolutionary algorithms (CMOEAs) can be categorized into five main groups.

The first group adopts the constraint handling techniques applied for single objective constraint optimization [10–12]. Geng et al. proposed a constrained evolutionary multiobjective optimization with infeasible elitists and stochastic ranking selection (IS-MOEA) [10]. The algorithm conserves infeasible elitists that acts as bridges connecting disconnected feasible regions, and stochastic ranking is adopted to balance objectives and constraints. IS-MOEA especially obtains improvement on the problems with two or more disconnected feasible regions.

The second group uses the basic mechanism of MOEAs and handles constraints by optimizing them as additional objectives. Mezura-Montes and Coello put forward a naïve method to solve CMOPs by ignoring infeasible solutions [13]. The algorithm is easy to implement, but when feasible regions are small and surrounded by infeasible solutions, it is difficult to find feasible solutions.

The third group is based on ranking of priority of the feasible and infeasible solutions [14–16]. Fonseca and Fleming proposed a unified approach for multiobjective optimization and multiple constraint handling [14]. Their algorithm handled constraints by assigning high priority to constraints and low priority to objective functions, when focusing on search of feasible solutions. Srinivas and Deb proposed a constrained multiobjective algorithm, in which constrained dominating relation of individuals is defined [16]. In this algorithm, all feasible solutions dominate all infeasible ones. Feasible solutions are sorted by their Pareto dominating relations and infeasible solutions are sorted based on their constraint violation. The algorithm can gain better performance but unfortunately it ignored the contribution of infeasible solutions to the Pareto front.

The forth group uses repair scheme to reproduce feasible solutions or less violated solutions from the original infeasible solutions [17–19]. Jimenez et al. proposed the evolutionary algorithm of nondominated sorting with radial slots (ENORA) [17], which employs the min.-max. formulation for constraint handling. Feasible individuals evolve toward optimality, while infeasible individuals evolve toward feasibility. Harada et al. proposed Pareto descent repair (PDR) operator that searches feasible solutions out of infeasible individuals in the constraint function space [19].

The fifth group designs new mechanisms to evolve feasible solutions towards Pareto front and evolve the infeasible solutions towards feasible regions [20–23]. Ray et al.

suggested using three different nondominated rankings of the population [20]. The first ranking is performed by using the objective function values; the second is performed by using different constraints; and the last ranking is based on the combination of all objective functions and constraints. Depending on these rankings, the algorithm performs according to the predefined rules. Chafekar et al. proposed two novel approaches for solving constrained multiobjective optimization problems [21]. One method called objective exchange genetic algorithm of design optimization (OEGADO) runs several GAs concurrently with each GA optimizing one objective and exchanging information about its objective with others. Another called objective switching genetic algorithm for design optimization (OSGADO) runs each objective sequentially with a common population for all objectives. Deb proposed GA's population-based approach that does not require any penalty parameter. Once sufficient feasible solutions are found, a niching method (along with a controlled mutation operator) is used to maintain diversity among feasible solutions [23].

2.3. Biogeography-Based Optimization (BBO). Biogeography is the science of the geographical distribution of biological organisms. In BBO, each problem solution is considered as a "habitat" with habitat survival index (HSI), which is similar to the fitness of EAs to evaluate an individual. High HSI habitats share their features with low HSI habitats. The process of sharing good features among solutions is denoted as migration. BBO adopts the migration strategy to share information among solutions. Good individuals' information can be conserved during the evolutionary process to ensure the population convergence. A mutation operator is used to generate diverse solutions to promote the diversity of the population. The detailed operations are described as follows.

Suppose that the species number of each individual i is S_i, and then its immigration rate λ_i and emigration rate μ_i can be calculated as follows [1]:

$$\begin{aligned}\lambda_i &= I\left(1-\frac{S_i}{S_{\max}}\right),\\ \mu_i &= \frac{ES_i}{S_{\max}},\end{aligned} \tag{5}$$

where $S_{\max}$ is the most species number of all habitats. I and E represent the maximization of immigration rate and emigration rate, respectively. In migration operator, the individuals' immigration rate and emigration rate are used to decide whether a solution should share its feature value with the other solutions. A better solution has a higher immigration rate and a lower emigration rate. By the migration, the solutions with high emigration rate tend to share their information with those with high immigration rate. Solutions with high immigration rate accept a lot of features from solutions with high emigration rate. With the aid of migration, BBO shows good exploitation ability in the search space.

Consider that species number change with species migrating; the probability P_s that the habitat contains exactly S species can be calculated using the following differential equation:

$$\dot{P}_s = \begin{cases} -\left(\lambda_s+\mu_s\right)P_s + \mu_{s+1}P_{s+1}, & S=0,\\ -\left(\lambda_s+\mu_s\right)P_s + \lambda_{s-1}P_{s-1} + \mu_{s+1}P_{s+1}, & 1\le S < S_{\max}-1\\ -\left(\lambda_s+\mu_s\right)P_s + \lambda_{s-1}P_{s-1}, & S=S_{\max}. \end{cases} \tag{6}$$

Then the mutation rate m_i is defined as [1]

$$m_i = P_{\text{mute}}\left(1-\frac{P_i}{P_{\max}}\right), \tag{7}$$

where P_{mute} is a predefined parameter, P_i is calculated according to (6), and $P_{\max} = \max_{1\le i\le N}\{P_i\}$. The mutation operator is implemented based on m_i. A solution with low probability P_i is likely to mutate other solutions. Conversely, some solutions with high P_i have very little chance to mutate. By the mutation operator, the diverse solutions are produced. The detailed operator on migration and mutation can refer to [1].

3. The Proposed Constrained Multiobjective Biogeography-Based Optimization Algorithm

3.1. CMBOA Description. In CMBOA, infeasible solutions recombine with nondominated feasible individuals and evolve towards feasibility. Firstly, the initial population is produced stochastically, and then the population is classified into the feasible and infeasible ones based on each individual's constraint violation. Secondly, depending on whether the feasible population is empty or not, infeasible population will adopt two types of operators. If feasible population is empty, infeasible population will implement differential evolution operator until feasible individuals present; otherwise, infeasible solutions nearby feasible regions recombine with their nearest nondominated feasible solutions to obtain feasibility. Diverse nondominated feasible solutions are generated from feasible individuals by applying the novel migration operator. With the increasing of nondominated feasible solutions, update operator is used to limit their number and ensure their even distribution. Both the feasible and infeasible solutions are combined in an external archive. The proposed CMBOA is described as Algorithm 1.

The procedure of CMBOA is described as follows.

Step 1 (initialization). Initialize the iterative number $t = 1$; the size of feasible elitist and infeasible elitist archive are N_1 and N_2, respectively. Generate randomly the initial population $A(t)$ with $N(t)$ individuals; that is $A(t) = \{a_1(t), a_2(t), \ldots, a_{N(t)}(t)\}$, the external archive $M(t) = \Phi$.

Step 2. Update the external archive.

Step 2.1. Divide the combination populations $A(t) \cup M(t)$ into the feasible and infeasible ones.

```
Step 1: Parameter setting: population size $N$, the size of feasible elitist archive $N_1$, the size of
        infeasible elitist archive $N_2$, maximum generation $g_{\max}$. Generate an initial population $A(t)$,
        set the iterative generation $t = 1$, and archive $M(t) = \Phi$,
Step 2: Update of the archive $M(t)$
    Step 2.1: Divide the population $A(t) \cup M(t)$ into the feasible set $P(t)$ and the infeasible set
              $Q(t)$ based on their constraint violation.
    Step 2.2: Select $N_1$ individuals with small fitness from $P(t)$ by individuals' fitness sorting to
              update feasible archive $P(t)$, and select $N_2$ individuals with small constraint violation
              from $Q(t)$ to update infeasible archive $Q(t)$.
    Step 2.3: Combine $P(t)$ and $Q(t)$ to gain archive set $M(t+1)$, $M(t+1) = P(t) \cup Q(t)$;
              If $t \geq g_{\max}$ is satisfied, output $P(t)$ and the algorithm stops; otherwise go the next step.
Step 3: Generate the offspring population
    Step 3.1: If $P(t) = \Phi$, then $P(t+1) = \Phi$, and perform the differential evolution operator on
              infeasible population $Q(t)$ to obtain $Q(t+1)$. Otherwise, go to Step 3.2
    Step 3.2: Implement selection operation on $P(t)$ to gain the breeding pool $D(t)$, and then
              execute migration operation on $D(t)$ to generate $P(t+1)$;
    Step 3.3: Implement the crossover and mutation on infeasible population $Q(t)$ to generate $Q(t+1)$.
    Step 3.4: Combine $P(t+1)$ and $Q(t+1)$ to obtain the offspring population $A(t+1)$,
              $A(t+1) = P(t+1) \cup Q(t+1)$
Step 4: let $t = t + 1$ and return Step 2.
```

Algorithm 1: CMBOA.

Computing the constraint violation of the individuals in $A(t) \cup M(t)$ according to (3), we have

$$v(A(t) \cup M(t)) = \left\{v(a_1(t)), v(a_2(t)), \ldots, v(a_{N(t)}(t))\right\}. \quad (8)$$

Depending on whether the value of $v(a_i(t))$ is zero or not, the population $A(t) \cup M(t)$ is divided into the feasible subpopulation $P(t)$:

$$P(t) = \left\{p_1(t), p_2(t), \ldots, p_{N_f(t)}(t)\right\} \quad (9)$$

and the infeasible subpopulation $Q(t)$:

$$Q(t) = \left\{q_1(t), q_2(t), \ldots, q_{N_{\text{if}}(t)}(t)\right\}. \quad (10)$$

Note that $N_f(t) + N_{\text{if}}(t) = N(t)$.

Step 2.2 (elitist feasible and infeasible archive). According to Definition 1, identify nondominated individuals of $P(t)$ to form the temporary set $P'(t)$:

$$P'(t) = \left\{p'_1(t), p'_2(t), \ldots, p'_{\text{non}(t)}(t)\right\}. \quad (11)$$

If the size of $P'(t)$ is smaller than the predefined size N_1, let $P(t) = P'(t)$. Otherwise, the crowding distance $I_{i.\text{cd}}(p'_i(t))$ of individual $p'_i(t)$, $1 \leq i \leq$ non is computed as follows [24]:

$$\begin{aligned} I_{\text{cd}}\left(p'_i(t)\right) = &\left(f_1\left(p'_i(t)\right) - f_1\left(p'_{i-1}(t)\right)\right) \\ &+ \left(f_2\left(p'_i(t)\right) - f_2\left(p'_{i-1}(t)\right)\right), \end{aligned} \quad (12)$$

where $f_k(p'_i(t))$ denotes the kth objective function value of individual $p'_i(t)$, $1 \leq i \leq$ non. According to the sequencing of crowding distance, select N_1 largest crowding distance individuals from $P'(t)$ to form the elitist feasible archive $P(t)$:

$$\begin{aligned} P(t) &= T_d\left(P'(t)\right) \\ &= T_d\left\{p'_1(t), p'_2(t), \ldots, p'_{\text{non}(t)}(t)\right\} \\ &= \left\{p_1(t), p_2(t), \ldots, p_{N_1}(t)\right\}. \end{aligned} \quad (13)$$

For the infeasible population $Q(t)$, if its size is smaller than the predefined size N_2, then $Q(t)$ keeps invariable. Otherwise, the fitness of its individual $q_i(t)$ is calculated as follows:

$$\text{fit}\left(q_i(t)\right) = \begin{cases} (1-\gamma)\, v\left(q_i(t)\right) + \gamma d\left(q_i(t)\right), & \gamma > 0, \\ v\left(q_i(t)\right), & \gamma = 0, \end{cases} \quad (14)$$

where γ is the proportion of nondominated feasible individuals in current population, $v(q_i(t))$ is constraint violation of the individual $q_i(t)$, and $d(q_i(t))$ denotes its Euclidean distance away from the nearest nondominated feasible solution. And then the proceeding N_2 individuals with small fitness from $Q(t)$ are conserved in elitist infeasible archive $Q(t)$.

Step 2.3 (formation of the archive). Combine elitist feasible archive $P(t)$ and elitist infeasible archive $Q(t)$ to gain the archive $M(t)$:

$$\begin{aligned} M(t) &= P(t) \cup Q(t) \\ &= \left\{p_1(t), p_2(t), \ldots, p_{N_1}(t), q_1(t), q_2(t), \ldots, q_{N_2}(t)\right\} \\ &= \left\{m_1(t), m_2(t), \ldots, m_{N_1}(t), \ldots, m_{N_1+N_2}(t)\right\}. \end{aligned} \quad (15)$$

If $t \geq g_{\max}$ is satisfied, export $P(t)$ as the output of the algorithm and the algorithm stops; otherwise, $t = t + 1$ and go to Step 3.

```
For $i = 1$ to $N_1$
    Randomly select two individuals $d_{s1}$, $d_{s2}$ from the population $D(t)$
    Select individual $d_s$ based on its immigration rate $\lambda_S$
    For $j = 1$ to $n$
        If rand < $\lambda_i$ then
            $p_{i,j}(t+1) = d_{s,j}(t) + \omega(t)(d_{s1,j}(t) - d_{s2,j}(t))$
        Else
            $p_{i,j}(t+1) = d_{i,j}(t)$
        End if
    End for
End for
```

Algorithm 2: Disturbance migration operator T_i.

Step 3. Generate the offspring population.
Step 3.1 (operation on feasible solutions). In CMBOA, when there are no feasible solutions in the current population, that is, $p(t) = \Phi$, we use the mutation operator of differential evolution to produce feasible individuals [25]. That is, three individuals $q_{r1}(t)$, $q_{r2}(t)$, and $q_{r3}(t)$ are selected randomly from $Q(t)$ and the mutation operator is performed as (16) until feasible solutions set $p(t)$ is not empty:

$$p_{i,j}(t) = q_{r1,j}(t) + \eta\left(q_{r2,j}(t) - q_{r3,j}(t)\right), \tag{16}$$

where η is a mutation constant and is a random number in the region $(0, 1)$. Otherwise, go to the next step.

Step 3.2 (selection operation). For feasible population $P(t)$, in order to ensure its convergence and even distribution, we define the fitness value of each individual by combining the nondominated rank and crowed distance of each individual $p_i(t)$, $1 \le i \le N_1$ as

$$\text{fit}\left(p_i(t)\right) = \frac{(1-\gamma)}{c_{ik}\left(p_i(t)\right)} + \gamma I_{\text{cd}}\left(p_i(t)\right), \tag{17}$$

where $I_{\text{cd}}(p_i(t))$ and $c_{ik}(p_i(t))$ denote the individual $p_i(t)$'s crowed distance and nondominated rank, respectively, and γ is defined in (14). By this fitness, when the number of nondominated feasible solutions is small, individuals with lower ranks have high fitness so that they have more chance to be selected. With the number of nondominated feasible solutions increasing, more individuals with large crowded distance are selected with high probability.

Perform tournament selection operator T_S on $P(t)$ to form the breeding pool $D(t)$:

$$\begin{aligned} D(t) &= T_S(P(t)) \\ &= T_S\left\{p_1(t), p_2(t), \dots, p_{N_1}(t)\right\} \\ &= \left\{d_1(t), d_2(t), \dots, d_{N_1}(t)\right\}. \end{aligned} \tag{18}$$

Step 3.3 (migration operation). The original migration operator of BBO has good exploitation ability of the population, but it is designed for the integer encoded individuals and single optimization problem. For continuous MOPs, the migration operator cannot ensure to produce the diverse solutions. So we propose a new migration operator. During the process of species migration, an individual is often affected by the other individuals. So we introduce a disturbance term in the migration operation to promote the diversity of the population. The detail process is shown in Algorithm 2.

In Algorithm 2, the disturbance factor $\omega(t)$ is defined as

$$\omega(t) = \frac{4}{5}\left(1 - \frac{1}{1 + e^{-0.1(t - g_{\max}/2)}}\right), \tag{19}$$

where $d_{i,j}(t)$ is the jth variable of the individual $d_i(t)$, $g_{\max}$ denotes the maximum iteration number, and t is the number of iteration at current generation. The amplitude of disturbance factor $\omega(t)$ decreases constantly with the increasing of generation t. At the beginning, large disturbance makes the population explore a wide region in decision space. Diverse solutions will be generated to promote the diversity of population because of difference of $\omega(t)$. At the end, a small disturbance is used to exploit effectively the local regions to guarantee its convergence.

The migration operator T_i on the population $D(t)$ is defined as

$$\begin{aligned} P(t+1) &= T_i(D(t)) \\ &= \left\{T_i(d_1(t)), T_i(d_2(t)), \dots, T_i\left(d_{N_1}(t)\right)\right\} \\ &= \left\{p_1(t), p_2(t), \dots, p_{N_1}(t)\right\}. \end{aligned} \tag{20}$$

Step 3.4 (crossover and mutation operation on the infeasible population). It had been noticed that infeasible solutions can contribute to the diversity of solutions on the Pareto front. When feasible solutions exist in the current population, an individual $q_{r1}(t)$ is selected randomly from $Q(t)$ and recombined with the nearest individual $d_{r2}(t)$ of $D(t)$. The crossover operator is described as

$$q_{i,j}(t+1) = \lambda q_{r1,j}(t) + (1-\lambda)\, d_{r2,j}(t), \tag{21}$$

where λ is a recombination parameter in the region $(0, 1)$. By the operator, infeasible individuals nearby the feasible region will approximate the feasibility.

The above crossover operation T_c on $Q(t)$ is

$$\begin{aligned} Q(t+1) &= T_c(Q(t)) \\ &= \left\{T_c(q_1(t)), T_c(q_2(t)), \dots, T_c\left(q_{N_2}(t)\right)\right\} \\ &= \left\{q_1(t+1), q_2(t+1), \dots, q_{N_2}(t+1)\right\}. \end{aligned} \tag{22}$$

Step 3.5. Combine $P(t+1)$ and $Q(t+1)$ to obtain the offspring population $A(t+1)$; namely, $A(t+1) = P(t+1) \cup Q(t+1)$.

Step 4. If the stopping criteria is not satisfied, $t = t+1$ and return to Step 2.

3.2. Time Complexity Analysis of CMBOA. The objectives of optimization problem are m, the size of population is N, the size of feasible archive is N_1, the size of infeasible archive is N_2, and the maximum of iterative times is $g_{\max}$. Time complexity for computation of constraint violation is $O(N)$. For migration and mutation operators on feasible individuals, its time complexity is $O(N_1)$, while time complexity for crossover operator on infeasible individuals is $O(N_2)$; time complexity for updating of feasible archive is $O(m(2N_1+N_2)^2)$, updating of infeasible archive is $O(m(N_1+2N_2)^2)$, and then the worst time complexity of CMBOA is

$$\begin{aligned} &O(N) + O(N_1) + O(N_2) + O\left(m(2N_1+N_2)^2\right) \\ &\quad + O\left(m(N_1+2N_2)^2\right) = O\left(m(2N_1+N_2)^2\right). \end{aligned} \tag{23}$$

3.3. Convergence Analysis of CMBOA. According to the description of CMBOA, it can be considered as an evolution Markov chain:

$$A(t) \xrightarrow{T_d} P(t) \xrightarrow{T_s} D(t) \xrightarrow{T_i} P(t+1) \longrightarrow A(t+1). \tag{24}$$

Let S be feasible solution space, $S \le N$ represents a state space composed of populations whose size is not more than N, and s_i denotes the ith state in state space. A_t^i denotes that the population $A(t)$ is in the state s_i, and $p(P_t^j \mid A_t^i)$ means the transformation probability from A_t^i to P_t^j. According to the description of CMBOA, we know that the series $\{A_t\}_{t\ge1}$ is an inhomogeneous Markov chain [26]. By using probability theory, the convergence of CMBOA is analyzed as follows.

Lemma 2. *There exists* $0 < \delta_1 < 1$, *s.t.* $p(P_{t+1}^e \mid D_t^d) \ge \delta_1$.

Proof.

$$\begin{aligned} p\left(P_{t+1}^e \mid D_t^d\right) &= \prod_{x\in D_t, y\in P_{t+1}} p(x,y) \\ &= \prod_{x\in D_t, y\in P_{t+1}} \lambda_x^{k(x,y)}(1-\lambda_x)^{n-k(x,y)} \\ &\ge \min\left(\lambda_{\min}, 1-\lambda_{\max}\right)^{N_1} \\ &= \delta_1, \end{aligned} \tag{25}$$

where $\alpha = \min_{x\in D_t}\{\lambda_x\}$, $\beta = \max_{x\in D_t}\{\lambda_x\}$, and $k(x,y) = \sum_{x_i \ne y_i, 1\le i\le n} \operatorname{sgn}|x_i - y_i|$. □

Lemma 3. *There exists* $0 < \delta_2 < 1$, *s.t.* $p(A_{t+1}^j \mid P_{t+1}^e) \ge \delta_2$.

Proof.

$$\begin{aligned} p\left(A_{t+1}^j \mid P_{t+1}^e\right) &= \frac{\left|M_f(P_{t+1}\cup Q_{t+1}\cup M_{t+1}, \prec) \cap M_f(S,\prec)\right|}{\left|P_{t+1}\cup Q_{t+1}\cup M_{t+1}\right|} \\ &\ge \left(\frac{M^*}{2(N_1+N_2)}\right)^{2N_1} \\ &= \delta_2, \end{aligned} \tag{26}$$

where $M^* = |M_f(P_{t+1}\cup Q_{t+1}\cup M_{t+1}, \prec) \cap M_f(S,\prec)|$ and $M_f(P_{t+1}\cup Q_{t+1}\cup M_{t+1}, \prec)$ denotes the nondominated feasible solutions of the population $P_{t+1}\cup Q_{t+1}\cup M_{t+1}$. □

Lemma 4. *Let* $M_f(s,\prec)$ *be the nondominated feasible solutions set; if* $s_i \cap M_f(s,\prec) = \Phi$ *and* $s_j \cap M_f(s,\prec) \ne \Phi$, *then there exists* $0 < \delta < 1$, *s.t.* $p(A_{t+1}^j \mid A_t^i) \ge \delta$.

Proof. Using K-C equation, we can obtain

$$\begin{aligned} p\left(A_{t+1}^j \mid A_t^i\right) &= \sum_{s_d} \sum_{,s_e \in S^{\le 2(N_1+N_2)}} p\left(D_t^d \mid A_t^i\right) p\left(P_{t+1}^e \mid D_t^d\right) \\ &\quad \times p\left(A_{t+1}^j \mid P_{t+1}^e\right). \end{aligned} \tag{27}$$

Note that $p(D_t^d \mid A_t^i) = 1$; based on Lemmas 2 and 3, we derive

$$p\left(A_{t+1}^j \mid A_t^i\right) \ge \delta_1\delta_2 \equiv \delta. \tag{28}$$

□

Theorem 5. *CMBOA is weakly convergent for any initial population distribution; that is,*

$$\lim_{t\to\infty} p\left(A_{t+1} \cap M_f(s,\prec) = \Phi\right) = 0. \tag{29}$$

Proof. If $s_i \cap M_f(s,\prec) = \Phi$, then

$$p\left(A_{t+1} \cap M_f(s,\prec) = \Phi \mid A_t^i\right) \le 1 - \delta. \tag{30}$$

If $s_i \cap M_f(s,\prec) \ne \Phi$, the archive $M(t)$ is applied to conserve the elitist individuals; then

$$p\left(A_{t+1} \cap M_f(s,\prec) = \Phi \mid A_t^i\right) = 0. \tag{31}$$

By (27) and (28), we can obtain

$$\begin{aligned} & p\left(A_{t+1} \cap M_f(s,\prec) = \Phi\right) \\ & \quad = \sum_{s_i \cap M_f(s,\prec)=\Phi} p\left(A_{t+1} \cap M_f(s,\prec) = \Phi \mid A_t^i\right) p\left(A_t^i\right) \\ & \qquad + \sum_{s_i \cap M_f(s,\prec) \neq \Phi} p\left(A_{t+1} \cap M_f(s,\prec) = \Phi \mid A_t^i\right) p\left(A_t^i\right) \\ & \quad \leq (1-\delta)^t, \\ & \qquad \lim_{t\to\infty} p\left(A_{t+1} \cap M_f(s,\prec) = \Phi\right) = 0. \end{aligned} \tag{32}$$

Hence, CMBOA is weakly convergent for any initial population distribution. □

4. Simulation Results

4.1. Experimental Setup. In order to test the validity of the proposed CMBOA, several benchmark functions with multiple features are selected including OSY [27], TNK [26], CONSTR [27], CTP1-CTP5 [28], CF1, CF2, CF4, and CF6 [29]. For OSY, its Pareto front is a concatenation of five regions and every region lies on the intersection of certain constraints; for TNK, its Pareto optimal solutions lie on a non-linear constraint surface; for CONSTR, its Pareto optimal set is concatenation of the constraint boundary and some parts of unconstrained Pareto optimal; for CTP serious functions, their Pareto optimal set is a collection of a number of discrete regions and most of solutions lie on non-linear constraint boundary. OSY, CONSTR, CTP1, CF4 and CF6 have continuous Pareto fronts, while the remaining ones have disjoint Pareto fronts (TNK, CTP2-CTP5, CF1, and CF2).

4.2. Performance Metrics. In this experiment, two performance metrics are selected to do quantitatively comparison.

Cover Metric C [30]. Suppose that U and V are two approximate Pareto optimal sets obtained by Algorithms 1 and 2:

$$C(U,V) = \frac{|\{u \in U; \exists v \in V : u \leq v\}|}{|V|}, \tag{33}$$

where $\leq$ denotes the dominated or equal relation. The value $C(U,V) = 1$ represents that all individuals in V are weakly dominated by individuals in U. $C(U,V) = 0$ denotes no individuals in V which is weakly dominated by U. Note that $C(U,V) \neq 1 - C(V,U)$; hence two directions must be considered simultaneously.

Hyper Volume HV [30]. The indicator calculates the volume covered by all nondominated solutions in the objective space. For each solution X_i, a hypercube hv_i is constructed with a predefined reference point and the solution X_i as the diagonal corners of the hypercube hv_i. All hypercubes are found and HV is calculated as follows:

$$\mathrm{HV} = \bigcup_{i=1}^{|x|} hv_i. \tag{34}$$

The indicator is related to the approximation and diversity of the nondominated solution set. The higher the value of HV is, the better the diversity and approximation of solution set obtained is.

4.3. Performance of CMBOA on Benchmark Functions

4.3.1. Test on Benchmark Functions. To demonstrate the effectiveness of the proposed CMBOA for CMOPs, 12 benchmark functions are chosen to show its validity. In the experiment, each individual is described as a real vector. The parameters of CMBOA are set as follows: population size = 100, feasible elitist maximum size $N_1 = 100$, infeasible elitist maximum size $N_2 = 20$, the maximum immigration rate and migration rate $E = I = 1$, the termination generation = 100, F is a random in the interval $(0.2, 0.8)$, and $CR = 0.5$.

For all benchmark function, the Pareto fronts obtained by CMBOA are shown in Figure 1. From this figure, it can be seen that the Pareto optimal solutions obtained by CMBOA are very close to the true Pareto front for all benchmark functions. For most of benchmark functions, the solutions generated by the proposed algorithm can be distributed evenly on the true Pareto front except CF2, CF4, and CF6 because they have variable linkages.

4.3.2. Comparison with Original Migration Operator. In order to demonstrate the effectiveness of the novel migration operation, OSY and CTP4 are selected. The Pareto fronts gained by the algorithms with original and novel migration operator are shown in Figure 2, where "$*$" denotes the Pareto front gained by CMBOA with the novel migration and "o" is the ones gained by the algorithm with original migration operator. From Figure 2, it can be seen that the algorithm with original migration cannot converge to the true Pareto front for OSY and CTP4, and only few solutions are produced for OSY. However, CMBOA with the novel migration operator obtains good convergence and diversity for OSY and CTP4, which demonstrates that the novel migration operator is superior to the original migration operator for OSY and CTP4.

4.3.3. Parameter Sensitivity Analysis. The disturbance parameter $F(t)$ is not tuned elaborately but is set as (20). In this section, to investigate the robustness of the disturbance parameter $F(t)$, simulations with different settings $F(t) = 0.2, 0.4, 0.6, 0.8$ are performed. Benchmark functions OSY and CTP4 are selected to test the sensitivity of $F(t)$. The Pareto fronts gained under different $F(t)$ settings are shown in Figure 3. From the results, it is observed that the algorithms with different $F(t)$ settings can all converge to the true Pareto front for OSY and CTP4, which illustrates that the disturbance parameter $F(t)$ is capable to perform consistently and effectively for OSY and CTP4. So the disturbance parameter $F(t)$ is reliable and robust to produce better solutions.

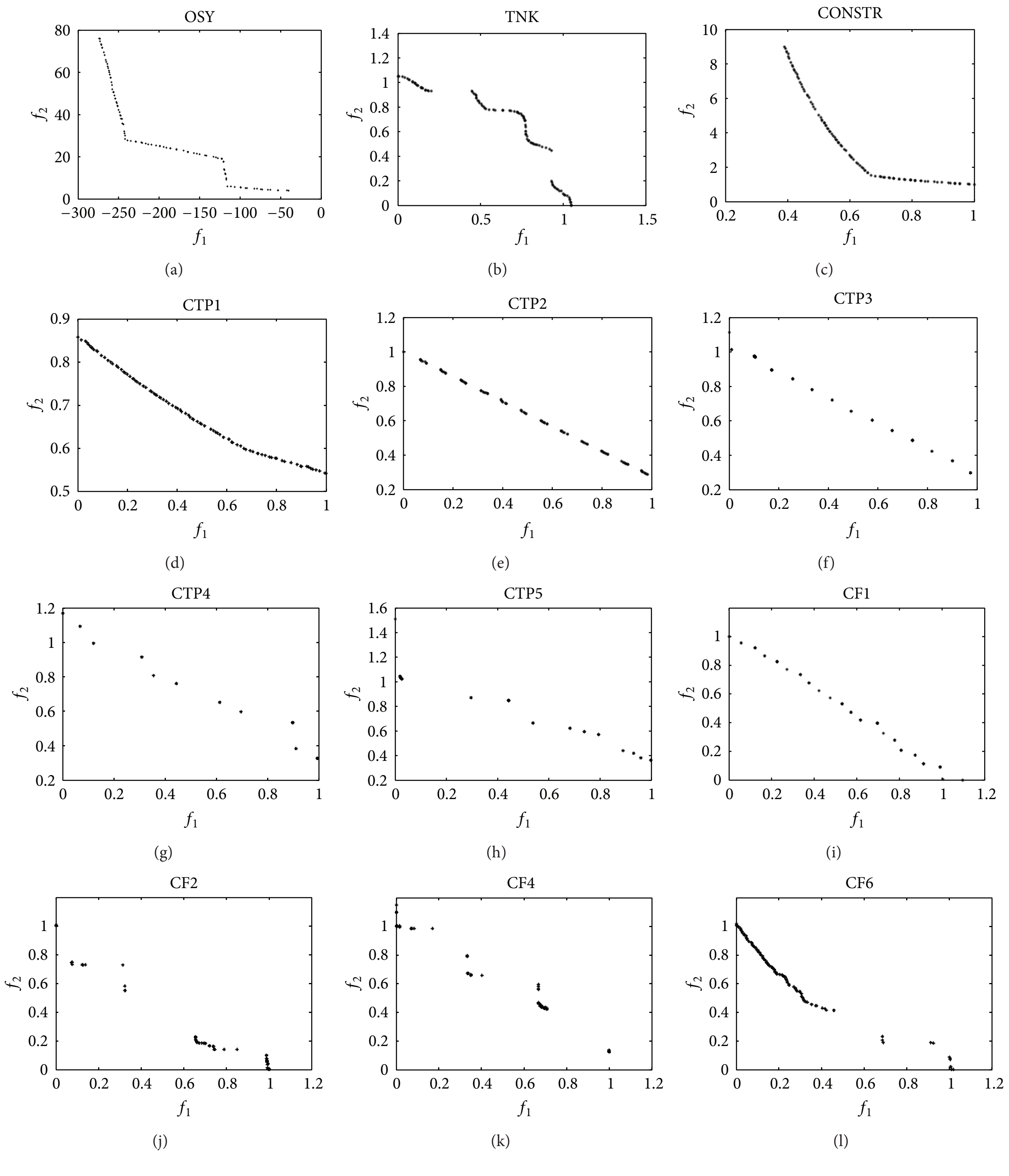


Figure 1: Final Pareto front for all test functions by the proposed algorithm CMBOA.

4.4. Comparison with Other Algorithms. To show the superior performance of the proposed algorithm, it is compared with the most popular multiobjective algorithms including NSGA-II [24] and IS-MOEA [1]. For NSGAII, the parameters are set as population size = 100, crossover probability = 0.9, mutation probability = $1/n$, SBX crossover parameter = 20, polynomial mutation parameter = 20, and the termination generation = 100. For IS-MOEA, the parameters are set as population size = 100, crossover probability = 0.9, mutation probability = $1/n$, SBX crossover parameter = 20, polynomial mutation parameter = 20 comparison probability = 0.45, penalty parameters $\omega_j = 1$, $\beta = 1$, and the termination

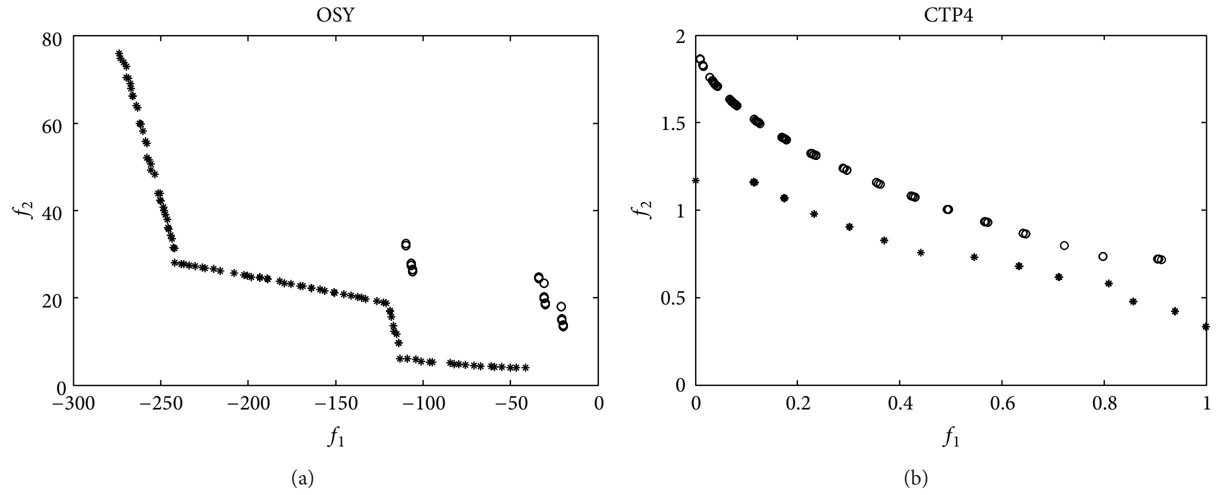


FIGURE 2: Final Pareto front for OSY and CTP4 under original and novel migration operator.

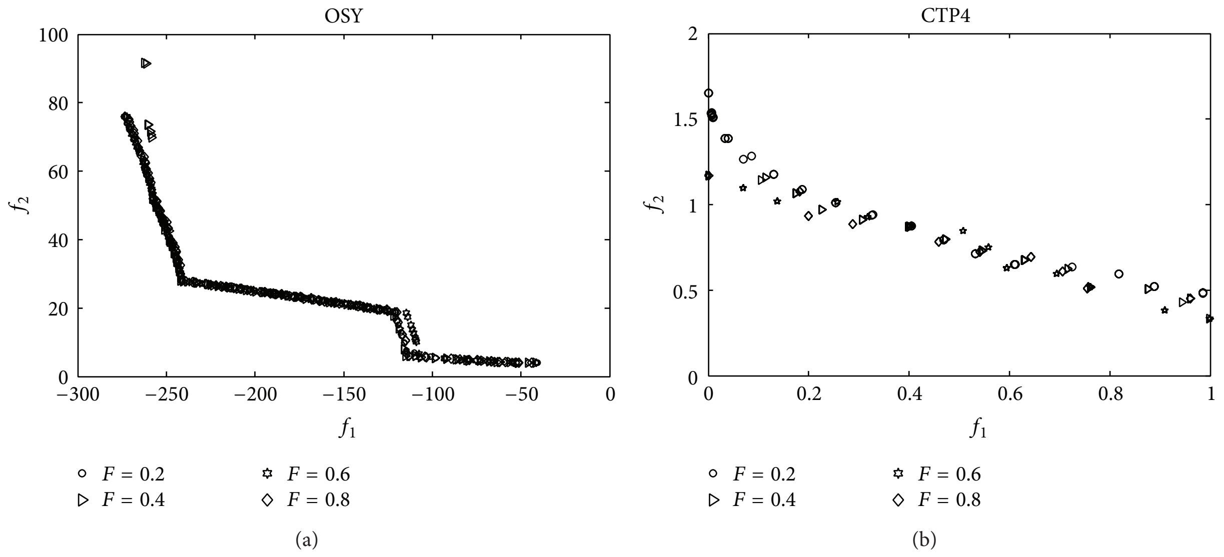


FIGURE 3: Final Pareto front for OSY and CTP4 under $F(t) = 0.2, 0.4, 0.6, 0.8$.

generation = 100. For the proposed CMBOA, the parameters are set the same as the previous section. For all algorithms, 30 independent runs are conducted on each of the benchmark functions to get the statistical results in cover metric C and hypervolume HV. Their distribution of simulation results is shown in Tables 1–6.

In Table 1, if the value of C (CMBOA, NSGA-II) is larger than that of C (NSGA-II,CMBOA), it indicates that the proposed CMBOA has better convergence than NSGA-II; otherwise, it indicates that the proposed CMBOA is inferior to NSGA-II in term of convergence. From Table 1, it can be seen that for CONSTR, CF2, CF4, and CF6, and NSGA-II is better than CMBOA in term of convergence. However, for the other eight test functions, CMBOA has better convergence than NSGA-II. Wilcoxon rank-sum test is used to examine their difference [31] and the results are shown in Table 2. The alternative hypothesis is $p \leq \alpha$ and $\alpha = 0.05$. If $p \leq \alpha$ is met, the algorithms have significant difference; otherwise, they have no difference. From Table 2, we can see that, compared with NSGA-II, CMBOA is significantly superior on OSY, CTP1, CTP3, CTP4, CTP5, CF1, and CF6 in converging close to Pareto front. In order to analyze their difference in convergence, the distribution of their cover metric values is studied by Wilcoxon rank-sum test which is summarized in Table 3. In Table 3, the CMBOA is significantly superior to IS-MOEA in most benchmark functions except TNK, CTP1, and CTP2 on convergence. In Table 4, we can see that, except TNK, CMBOA is superior to IS-MOEA in convergence, while IS-MOEA is better than CMBOA in TNK.

In order to evaluate the convergence and the diversity of solutions obtained by the proposed CMBOA, statistical results of hypervolume metric are summarized in Table 5. In this table, higher hypervolume value indicates that the algorithm has better diversity. From Table 5, it can be seen

Table 1: Mean and variance (Var.) of the cover metric on CMBOA and NSGA-II.

Algorithm	Benchmark functions					
	OSY	TNK	CONSTR	CTP1	CTP2	CTP3
C(CMBOA, NSGA-II)						
Mean	0.3003	0.2007	0.1320	0.2643	0.2690	0.7543
Var.	0.0324	0.0019	0.0012	0.0023	0.0039	0.0319
C(NSGA-II, CMBOA)						
Mean	0.1683	0.1940	0.1517	0.1270	0.2397	0.2567
Var.	0.0160	0.0011	0.0016	0.0026	0.0043	0.0262
Algorithm	Benchmark functions					
	CTP4	CTP5	CF1	CF2	CF4	CF6
C(CMBOA, NSGA-II)						
Mean	0.7507	0.6863	0.8403	0.1433	0.0860	0.2250
Var.	0.0540	0.0675	0.0138	0.0152	0.0209	0.0100
C(NSGA-II, CMBOA)						
Mean	0.1793	0.3150	0.1550	0.1683	0.2500	0.4093
Var.	0.0274	0.0296	0.0120	0.0080	0.0961	0.0135

Table 2: Wilcoxon rank-sum test on *C* value of CMBOA and NSGA-II.

	OSY	TNK	CONSTR	CTP1	CTP2	CTP3
(*C*(CMBOA, NSGA-II), *C*(NSGA-II, CMBOA))	**0.0020**	0.3937	0.0738	**5.6395e − 010**	0.1096	**4.3641e − 010**
	CTP4	CTP5	CF1	CF2	CF4	CF6
(*C*(CMBOA, NSGA-II), *C*(NSGA-II, CMBOA))	**6.3039e − 010**	**1.7176e − 006**	**3.1436e − 011**	0.1408	0.7492	**4.9194e − 007**

Table 3: Wilcoxon rank-sum test on *C* value of CMBOA and IS-MOEA.

	OSY	TNK	CONSTR	CTP1	CTP2	CTP3
(*C*(CMBOA, IS-MOEA), *C*(CMBOA, IS-MOEA))	**4.4824e − 012**	**2.8394e − 011**	**0.0010**	0.0685	**5.7712e − 009**	**0.0058**
	CTP4	CTP5	CF1	CF2	CF4	CF6
(*C*(CMBOA, IS-MOEA), *C*(CMBOA, IS-MOEA))	**3.2785e − 011**	**9.1907e − 009**	**1.8685e − 011**	**2.3115e − 010**	**1.1817e − 008**	**9.7713e − 012**

Table 4: Mean and variance (Var.) of the cover metric on CMBOA and IS-MOEA.

Algorithm	Benchmark functions					
	OSY	TNK	CONSTR	CTP1	CTP2	CTP3
C(CMBOA, IS-MOEA)						
Mean	0.9044	0.0997	0.1670	0.1823	0.1773	0.5976
Var.	0.0456	0.0010	0.0010	0.0048	0.0037	0.0318
C(IS-MOEA, CMBOA)						
Mean	0.0517	0.3093	0.1367	0.1523	0.3167	0.4710
Var.	0.0140	0.0029	0.0012	0.0017	0.0045	0.0249
Algorithm	Benchmark functions					
	CTP4	CTP5	CF1	CF2	CF4	CF6
C(CMBOA, IS-MOEA)						
Mean	0.7990	0.6701	0.9667	0.6591	0.6190	0.8472
Var.	0.0241	0.0334	0.0020	0.0680	0.0957	0.0305
C(IS-MOEA, CMBOA)						
Mean	0.1767	0.3223	0.0520	0.0787	0.0713	0.0143
Var.	0.0173	0.0133	0.0045	0.0109	0.0210	0.0011

Table 5: Mean and variance (Var.) of the hypervolume (HV) metric.

Algorithm	HV	Benchmark functions					
		OSY	TNK	CONST	CTP1	CTP2	CTP3
CMBOA	Mean	0.9835	0.998	0.9992	0.9995	0.9992	0.9949
	Var.	0.0032	3.0145*e* – 007	2.2481*e* – 007	9.3282*e* – 008	1.5011*e* – 007	1.3630*e* – 005
NSGA-II	Mean	0.9107	0.9965	0.9993	0.9944	0.9983	0.9763
	Var.	0.0162	5.7807*e* – 006	3.1099*e* – 007	4.1408*e* – 005	2.8101*e* – 006	0.0029
IS-MOEA	Mean	0.7576	0.9965	0.9992	0.9757	0.9579	0.9609
	Var.	0.0257	1.5731*e* – 004	2.8652*e* – 007	0.0021	0.0069	0.0011
Algorithm	HV	Benchmark Functions					
		CTP4	CTP5	CF1	CF2	CF4	CF6
CMBOA	Mean	0.9289	0.9190	0.9956	0.9549	0.8820	0.9535
	Var.	0.0015	0.0014	1.4497*e* – 005	0.0016	0.0077	8.3474*e* – 004
NSGA-II	Mean	0.8581	0.8180	0.9901	0.8971	0.9319	0.9445
	Var.	0.0185	0.0181	4.6140*e* – 005	0.0037	0.0026	0.0017
IS-MOEA	Mean	0.8716	0.8518	0.9670	0.7838	0.7444	0.8472
	Var.	0.0115	0.0088	3.899*e* – 004	0.0208	0.0496	0.0127

Table 6: Distribution of HV value using Wilcoxon rank-sum test.

	OSY	TNK	CONSTR	CTP1	CTP2	CTP3
HV(CMBOA, NSGA-II)	2.5721*e* – 007	3.0199*e* – 011	**0.0555 >0.05**	4.6159*e* – 010	**0.0824 >0.05**	8.9934*e* – 011
HV(CMBOA, IS-MOEA)	1.4110*e* – 009	3.0199*e* – 011	2.8314*e* – 008	3.3242*e* – 006	**0.3255 >0.05**	9.8329*e* – 008
	CTP4	CTP5	CF1	CF2	CF4	CF6
HV(CMBOA, NSGA-II)	4.0772*e* – 011	1.0937*e* – 010	2.6099*e* – 010	0.0044	**0.8534 >0.05**	5.9706*e* – 005
HV(CMBOA, IS-MOEA)	3.3384*e* – 011	4.5726*e* – 009	3.0199*e* – 011	1.0937*e* – 010	4.1825*e* – 009	3.0199*e* – 011

that CMBOA has better diversity than the other two algorithms for almost all test functions except CONSTR and CF4. From the variance of metric HV, we can see that CMBOA has the smallest variance which indicates that it is more reliable and robust than NSGA-II and IS-MOEA in producing better solutions. In order to analyze the distribution of HV value in further, its Wilcoxon rank-sum test value is summarized in Table 6. From Table 6, we can conclude that CMBOA is superior to NSGA-II and IS-MOEA in terms of the distribution and diversity of solutions except CONSTR, CTP2, and CF4. Experiment results above show that the CMBOA has competitive performance with NSGA-II and IS-MOEA in the terms of convergence and diversity.

5. Conclusions

In this paper, we propose a new constrained multiobjective biogeography-based optimization algorithm. A new disturbance migration operator is designed to generate diverse feasible solutions. Infeasible solutions nearby the feasible region are recombined with their nearest feasible ones to change the feasibility. Theoretically, the weak convergence of CMBOA is proved by the probability theory and its time complexity is analyzed. Experimentally, CMBOA is tested on several typical benchmark functions and compared with classical NSGA-II and IS-MOEA. The statistical results show that the proposed CMBOA is highly competitive in terms of convergence and diversity. In future work, we may improve CMBOA to obtain better performance on variable linkage problems.

Conflict of Interests

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

Acknowledgments

The work is supported by National Nature Science Foundation of China (no. 61075113), Excellent Youth Foundation of Heilongjiang Province (no. JC12012), the Fundamental Research Funds for the Central Universities (no. HEUCFZ1209), and Harbin Excellent Discipline Leader (no. 2012RFXXG073).

References

[1] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[2] D. Simon, "A probabilistic analysis of a simplified biogeography-based optimization algorithm," *Evolutionary Computation*, vol. 19, no. 2, pp. 167–185, 2011.

[3] A. Sinha, S. Das, and B. K. Panigrahi, "A linear state-space analysis of the migration model in an island biogeography system," *IEEE Transactions on Systems, Man, and Cybernetics A: Systems and Humans*, vol. 41, no. 2, pp. 331–337, 2010.

[4] D. Simon, A. Shah, and C. Scheidegger, "Distributed learning with biogeography- based optimization: Markov modeling and robot control," *Swarm and Evolutionary Computation*, vol. 10, pp. 12–24, 2013.

[5] S. Rajasomashekar and P. Aravindhababu, "Biogeography based optimization technique for best compromise solution of economic emission dispatch," *Swarm and Evolutionary Computation*, vol. 7, pp. 47–57, 2012.

[6] H.-P. Ma, X.-Y. Ruan, and Z.-X. Pan, "Handling multiple objectives with biogeography-based optimization," *International Journal of Automation and Computing*, vol. 9, no. 1, pp. 30–36, 2012.

[7] H. Mo and Z. Xu, "Research of biogeography-based multi-objective evolutionary algorithm," *Journal of Information Technology Research*, vol. 4, no. 2, pp. 70–80, 2011.

[8] Z. D. Xu and H. W. Mo, "Improvement for migration operator in biogeography-based optimization algorithm," *Pattern Recognition and Artificial Intelligence*, vol. 25, no. 3, pp. 544–548, 2012.

[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] H. Geng, M. Zhang, L. Huang, and X. Wang, "Infeasible elitists and stochastic ranking selection in constrained evolutionary multi-objective optimization," in *Proceedings of the 6th International Conference on Simulated Evolution and Learning (SEAL '06)*, pp. 336–344, Hefei, China, 2006.

[11] R. Mallipeddi, S. Jeyadevi, P. N. Suganthan, and S. Baskar, "Efficient constraint handling for optimal reactive power dispatch problems," *Swarm and Evolutionary Computation*, vol. 5, pp. 28–36, 2012.

[12] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 80–92, 2008.

[13] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[14] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms—part I: a unified formulation," *IEEE Transactions on Systems, Man, and Cybernetics A: Systems and Humans*, vol. 28, no. 1, pp. 26–37, 1998.

[15] T. Bäck, F. Hoffmeister, and H. Schwefel, "A survey of evolution strategies," in *Proceedings of the International Conference on Genetic Algorithms*, pp. 2–9, San Diego, Calif, USA, 1991.

[16] N. Srinivas and K. Deb, "Multi-objective function optimization using non-dominated sorting genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[17] F. Jimenez, A. F. Gomez-Skarmeta, G. Sanchez, and K. Deb, "An evolutionary algorithm for constrained multi-objective optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '02)*, pp. 1133–1138, Honolulu, Hawaii, USA, 2002.

[18] J. A. Joines and C. R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pp. 579–584, Orlando, Fla, USA, June 1994.

[19] K. Harada, J. Sakuma, I. Ono, and S. Kobayashi, "Constraint-handling method for multi- objective function optimization: pareto descent repair operator," in *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization (EMO '07)*, pp. 156–170, Matshushima, Japan, 2007.

[20] T. Ray, K. Tai, and K. C. Seow, "Multiobjective design optimization by an evolutionary algorithm," *Engineering Optimization*, vol. 33, no. 4, pp. 399–424, 2001.

[21] D. Chafekar, J. Xuan, and K. Rasheed, "Constrained multi-objective optimization using steady state genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computational Conference (GECCO '03)*, pp. 813–824, Chicago, Ill, USA, 2003.

[22] R. Farmani and J. A. Wright, "Self-adaptive fitness formulation for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 445–455, 2003.

[23] 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.

[24] 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.

[25] 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.

[26] M. Tanaka, "GA-based decision support system for multi-criteria optimization," in *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization (EMO '95)*, pp. 1556–1561, Guanajuato, Mexico, 1995.

[27] A. Osyczka and S. Kundu, "A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm," *Structural Optimization*, vol. 10, no. 2, pp. 94–99, 1995.

[28] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[29] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multi-objective optimization test instances for the CEC 2009 special session and competition," Tech. Rep., University of Essex, Colchester, UK; Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, 2008.

[30] J. D. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," TIK-Report 214, 2005.

[31] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 375358, 13 pages
http://dx.doi.org/10.1155/2014/375358

Research Article

An Improved Cockroach Swarm Optimization

I. C. Obagbuwa and A. O. Adewumi

School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Westville, Durban 4000, South Africa

Correspondence should be addressed to I. C. Obagbuwa; ibidunobagbuwa@yahoo.com

Received 6 December 2013; Accepted 18 March 2014; Published 14 May 2014

Academic Editors: P. Agarwal, S. Balochian, V. Bhatnagar, and Y. Zhang

Copyright © 2014 I. C. Obagbuwa and A. O. Adewumi. 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.

Hunger component is introduced to the existing cockroach swarm optimization (CSO) algorithm to improve its searching ability and population diversity. The original CSO was modelled with three components: chase-swarming, dispersion, and ruthless; additional hunger component which is modelled using partial differential equation (PDE) method is included in this paper. An improved cockroach swarm optimization (ICSO) is proposed in this paper. The performance of the proposed algorithm is tested on well known benchmarks and compared with the existing CSO, modified cockroach swarm optimization (MCSO), roach infestation optimization RIO, and hungry roach infestation optimization (HRIO). The comparison results show clearly that the proposed algorithm outperforms the existing algorithms.

1. Introduction

Swarm intelligence (SI) is a method of computing whereby simple decentralized agents get information by interacting locally with one another and their environment [1]. The local information received is not controlled centrally; local interaction of agents results in amazing and emergent global patterns which can be adopted for solving problems [1].

SI algorithms draw inspiration from insects and animals social behaviour and have been proven in literature to be efficient in solving global optimization problems. Examples of existing SI algorithms include particle swarm optimization (PSO), ant colony optimization (ACO), and bee colony optimization (BCO). PSO based on bird social behaviour, introduced by Kennedy and Eberhart [2], has been applied to several problems, including power and management processes [3, 4] and combinatorial optimization problem in [5]. ACO based on ant social behaviour, introduced by Dorigo [6], has been applied to problems such as vehicle routing problem [7] and network routing problem [8]. BCO based on bees social behaviour, introduced by Pham et al. [9], has been applied to real world problems by Karaboga and his research group [10–12].

One of the recent developments in SI is cockroach optimization [13–16]. Cockroach belongs to Insecta Blattodea, abodes in warm, dark, and moist shelters, and exhibits habits which include chasing, swarming, dispersing, being ruthless and omnivorous, and food searching. Cockroaches interact with peers and respond to their immediate environment and make decisions based on their interaction such as selecting shelter, searching for food sources and friends, dispersing when danger is noticed, and eating one another when food is scarce.

The original cockroach swarm optimization (CSO) algorithm, introduced by Zhaohui and Haiyan [14], was modified by ZhaoHui with the introduction of inertial weight [15]. CSO algorithms [14, 15] mimic chase swarming, dispersion, and ruthless social behaviour of cockroaches.

Global optimization problems are considered as very hard problems, ever increasing in complexity. It became necessary to design better optimization algorithms; this necessitated the design of a better cockroach algorithm. This paper extends MCSO with the introduction of another social behaviour called hunger behaviour. Hunger behaviour prevents local optimum and enhances diversity of population. An improved cockroach swarm optimization (ICSO) is presented in this paper.

The organization of this paper is as follows: Section 2 presents CSO, MCSO, and ICSO models with algorithmic steps; Section 3 shows the experiments carried out and results obtained; the paper is summarised in Section 4.

2. Cockroach Swarm Optimization

CSO algorithm is a population based global optimization algorithm which has been applied to problems in literature including [17–19]. CSO [14] models are given as follows.

(1) Chase-Swarming Behaviour.

$$x_i = \begin{cases} x_i + \text{step} \cdot \text{rand} \cdot (p_i - x_i), & x_i \neq p_i \\ x_i + \text{step} \cdot \text{rand} \cdot (p_g - x_i), & x_i = p_i, \end{cases} \quad (1)$$

where x_i is the cockroach position, step is a fixed value, rand is a random number within $[0, 1]$, p_i is the personal best position, and p_g is the global best position. Consider

$$p_i = \text{Opt}_j \left\{ x_j, \left| x_i - x_j \right| \leq \text{visual} \right\}, \quad (2)$$

where perception distance visual is a constant, $j = 1, 2, \ldots, N$, $i = 1, 2, \ldots, N$. Consider

$$p_g = \text{Opt}_i \{x_i\}. \quad (3)$$

(2) Dispersion Behaviour.

$$x_i = x_i + \text{rand}(1, D), \quad i = 1, 2, \ldots, N, \quad (4)$$

where $\text{rand}(1, D)$ is a D-dimensional random vector that can be set within a certain range.

(3) Ruthless Behaviour.

$$x_k = p_g, \quad (5)$$

where k is a random integer within $[1, N]$ and p_g is the global best position.

2.1. Modified Cockroach Swarm Optimization. ZhaoHui presented a modified cockroach swarm optimization (MCSO) [15] with the introduction of inertial weight to chase swarming component of original CSO as shown below. Other models remain as in original CSO.

Chase-swarming behaviour is as follows:

$$x_i = \begin{cases} w \cdot x_i + \text{step} \cdot \text{rand} \cdot (p_i - x_i), & x_i \neq p_i \\ w \cdot x_i + \text{step} \cdot \text{rand} \cdot (p_g - x_i), & x_i = p_i, \end{cases} \quad (6)$$

where w is an inertial weight which is a constant.

2.2. Improved Cockroach Swarm Optimization. In this paper, MCSO is extended with additional component called hunger behaviour.

2.2.1. Hunger Behaviour. At interval of time, when cockroach is hungry, it migrates from its comfortable shelter and friends company to look for food [13, 20]. Hunger behaviour is modelled using partial differential equation (PDE) migration techniques [21]. Cockroach migrates from its shelter to any available food source x_{food} within the search space. A threshold hunger is defined, when cockroach is hungry and threshold hunger is reached; it migrates to food source. Hunger behaviour prevents local optimum and enhances diversity of population.

PDE migration equation is described by Kerckhove [21]:

$$\frac{\partial u}{\partial t} = -c \frac{\partial u}{\partial x} \quad (7)$$

with $u(0, x) = u_0(x)$.

Parameter c is the controlling speed of the migration. u is the population size, t is time, and x is location or position. $u(t, x)$ is the population size at time t in location x with $u(0, x) = u_0(x)$ being the initial population distribution. Consider

$$\begin{aligned} \frac{\partial u}{\partial t} &= -c \frac{\partial u}{\partial x}, \\ \frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} &= 0. \end{aligned} \quad (8)$$

The characteristic equations are

$$\begin{aligned} \frac{dt}{1} &= \frac{dx}{c} = \frac{du}{0}, \\ dx - c\,dt &= 0. \end{aligned} \quad (9)$$

By integration, we have

$$\begin{aligned} x - ct &= \alpha, \\ u &= u(\alpha), \\ u &= u(x - ct), \\ u[t, x] &= u_0[-ct + x]. \end{aligned} \quad (10)$$

Consider displacement = speed × time.

In $u_0(x - ct)$, $u_0(x)$ displaces ct.

$u_0(x - ct)$ satisfies migration equation at any initial population distribution $u_0(x)$ [21].

Hunger behaviour is modelled as follows:

If (hunger == t_{hunger})

$$x_i = x_i + (x_i - ct) + x_{\text{food}}, \quad (11)$$

where x_i denotes cockroach position, $(x_i - ct)$ denotes cockroach migration from its present position, c is a constant which controls migration speed at time t, x_{food} denotes food location, t_{hunger} denotes hunger threshold, and hunger is a random number $[0, 1]$.

2.2.2. Improved Cockroach Swarm Optimization Models

(1) Chase-Swarming Behaviour.

$$x_i = \begin{cases} w \cdot x_i + \text{step} \cdot \text{rand} \cdot (p_i - x_i), & x_i \neq p_i, \\ w \cdot x_i + \text{step} \cdot \text{rand} \cdot (p_g - x_i), & x_i = p_i, \end{cases} \quad (12)$$

```
INPUT: Fitness function: $f(x), x \in R^D$
set parameters and generate an initial population of cockroach
set $p_g = x_1$
for $i = 2$ to $N$ do
  if $f(x_i) < f(p_g)$ then
    $p_g = x_i$
  end if
end for
for $t = 1$ to $T_{\max}$ do
  for $i = 1$ to $N$ do
    for $j = 1$ to $N$ do
      if abs$(x_i - x_j) <$ visual; $f(x_j) < f(x_i)$ then
        $p_i = x_i$
      end if
    end for
    if $p_i == x_i$ then
      $x_i = w \cdot x_i + \text{step} \cdot \text{rand} \cdot (p_g - x_i)$
    else
      $x_i = w \cdot x_i + \text{step} \cdot \text{rand} \cdot (p_i - x_i)$
    end if
    if $f(x_i) < f(p_g)$ then
      $p_g = x_i$
    end if
  end for
  if Hunger == $t_{\text{hunger}}$ then
    $x_i = x_i + (x_i - ct) + x_{\text{fd}})$
    hunger$_i$ = 0
    Increment hunger$_i$ counters
  end if
  for $i = 1$ to $N$ do
    $x_i = x_i + \text{rand}(1, D)$
    if $f(x_i) < f(p_g)$ then
      $p_g = x_i$
    end if
  end for
  $k = \text{randint}([1, N])$
  $x_k = p_g$;
end for
Check termination condition
```

Algorithm 1: An improved cockroach swarm optimization algorithm.

where w is an inertial weight which is a constant, step is a fixed value, rand is a random number within $[0, 1]$, p_i is the personal best position, and p_g is the global best position. Consider

$$p_i = \text{Opt}_j \left\{x_j, \left|x_i - x_j\right| \le \text{visual}\right\}, \tag{13}$$

where perception distance visual is a constant, $j = 1, 2, \ldots, N$, $i = 1, 2, \ldots, N$. Consider

$$p_g = \text{Opt}_i \{x_i\}. \tag{14}$$

(2) Hunger Behaviour. If hunger == t_{hunger},

$$x_i = x_i + (x_i - ct) + x_{\text{food}}, \tag{15}$$

where x_i denotes cockroach position, $(x_i - ct)$ denotes cockroach migration from its present position, c is a constant which controls migration speed at time t, x_{food} denotes food location, t_{hunger} denotes hunger threshold, and hunger is a random number within $[0, 1]$.

(3) Dispersion Behaviour.

$$x_i = x_i + \text{rand}(1, D), \quad i = 1, 2, \ldots, N, \tag{16}$$

where rand$(1, D)$ is a D-dimensional random vector that can be set within a certain range.

(4) Ruthless Behaviour.

$$x_k = p_g, \tag{17}$$

where k is a random integer within $[1, N]$ and p_g is the global best position.

The algorithm for ICSO is illustrated in Algorithm 1 and its computational steps given as follows.

(1) Initialise cockroach swarm with uniform distributed random numbers and set all parameters with values.

(2) Find p_i and p_g using (12) and (13).

(3) Perform chase-swarming using (11).

(4) Perform hunger behaviour using (14)

(5) Perform dispersion behaviour using (15).

(6) Perform ruthless behaviour using (16).

(7) Repeat the loop until stopping criterion is reached.

Series of experiments are conducted in Section 3 using established global optimization problems to test ICSO performance. The performance of ICSO is compared with that of existing algorithms RIO, HRIO, CSO, and MCSO.

3. Simulation Studies

The speed, accuracy, robustness, stability, and searching capabilities of ICSO are evaluated in this section with 23 benchmark test functions. The test functions were adopted from [22–24]; any further information about the test functions can be found in these references. The test functions are of different characteristics such as unimodal (U), multimodal (M), separable (S), and nonseparable (N). Table 1 of this paper shows the test functions used, whose problem ranges from 2 to 30 in dimension as in [22–24].

All algorithms were implemented in MATLAB 7.14 (R2012a) and run on a computer with 2.30 GHz processor with 4.00 GB of RAM. Experimental setting of [13–15] is used for the experiments of this paper; experiment runs 20 times with maximum iteration 1000, perception distance visual $=$ 5, the largest step was step $=$ 2, and inertia weight was $w = 0.618$; we defined hunger threshold $t_{\text{hunger}} = 0.5$ and hunger as a randomly generated number $[0, 1]$ in each iteration for ICSO. Cockroach parameters [13] are used for RIO and HRIO; $c_0 = 0.7$ and $c_{\max} = 1.43$, hunger threshold $t_{\text{hunger}} = 100$, and hunger as randomly generated number $[0, (t_{\text{hunger}} - 1)]$. Cockroach population size $N = 50$ is used in this paper for all the algorithms. Further details about RIO, HRIO, CSO, and MSCO can be found in [13–15].

ICSO along with similar algorithms, that is, CSO, MSCO, RIO, and HRIO, was implemented with several simulation experiments conducted and reported. Success rate, average and best fitness, standard deviation (STD), and execution time in seconds are used as performance measure for comparative purpose (see Tables 2, 3, and 4 of this paper).

ICSO locates minimum values for the tested benchmark problems such as Bohachevsky, Rastrigin, Easom, Schaffer, Step, and Storn's Tchebychev problems as shown in Tables 2, 3, and 4. The comparison of the average performance of ICSO with that of RIO, HRIO, CSO, and MCSO is shown in Table 5; the comparison result clearly shows that ICSO outperforms other algorithms. Similarly, the best performance of ICSO with that of RIO, HRIO, CSO, and MCSO is shown in Table 6; ICSO has better performance than others.

ICSO algorithm has consistent performance in each iteration. This is proved by very low standard deviation of the average optimal recoded during experiments. The ICSO average optimal STD is compared with the STD of RIO, HRIO, CSO, and MCSO in Table 7. ICSO has better minimum STD than others.

ICSO locates good solutions in each experiment; this is proved by the success rate of the algorithm. Table 8 shows the comparison of the success rate of the proposed algorithm with the existing algorithms RIO, HRIO, CSO, and MCSO. ICSO has 100% success rate in all test functions except Rosenbrock.

ICSO utilizes minimum time in executing the selected test function. Table 9 shows the comparison of the execution time of ICSO and that of RIO, HRIO, CSO, and MCSO; ICSO is shown to have utilized minimum time.

To determine the significant difference between the performance of the proposed algorithm and the existing algorithms, test statistic of Jonckheere-Terpstra (J-T) test was conducted using the statistical package for the social science (SPSS). The Null hypothesis test for J-T test is that there is no difference among several independent groups. As the usual practice in most literature, P value threshold value for hypothesis test was set to 0.05. If P value is less than 0.05, the Null is rejected which means there is significant difference between the groups. Otherwise the Null hypothesis is accepted. Table 10 shows the result of J-T test; P value (Asymp. Sig.) was computed to be 0.001. The P value is less than the threshold value 0.05; therefore, there is significant difference in performance of ICSO and that of RIO, HRIO, CSO, and MCSO for benchmarks evaluated.

Effect size of the significant difference is the measure of the magnitude of the observed effect. The effect size r, $(1 > r < 0)$ of the significant difference of J-T test, was calculated as

$$r = \frac{Z}{\sqrt{N}}, \tag{18}$$

where Z is the standard data of J-T statistic as shown in Table 10, N is the total number of samples, and $N = 114$. Consider

$$Z = \frac{x - \mu}{\sigma}, \tag{19}$$

where x denotes observed J-T statistic, μ denotes the mean J-T statistic, and σ denoted the standard deviation of J-T statistic. Consider

$$\begin{aligned} Z &= \frac{1952 - 2599}{199.355} = -3.245, \\ r &= \frac{-3.245}{\sqrt{114}} = -0.3. \end{aligned} \tag{20}$$

The distance between the observed data and the mean in units of standard deviation is absolute value of $|Z|$ (Z is negative when observed data is below the mean and positive when above). The effect size 0.3 is of medium size, using Cohen's guideline on effect size [25, 26]. The statistics of 0.3 effect size shows that there is significant difference of medium magnitude between proposed algorithm and existing algorithms.

Table 1: Benchmark test functions.

Number	Range	D	C	Functions	Description
1	[−100, 100]	30	US	Step	$f(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$
2	[−100, 100]	30	US	Sphere	$f(x)=\sum_{i=1}^{n}x_i^2$
3	[−10, 10]	30	US	Sumsquares	$f(x)=\sum_{i=1}^{n}ix_i^2$
4	[−100, 100]	2	MS	Bohachevsky1	$f(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$
5	[−100, 100]	2	MN	Bohachevsky2	$f(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)(4\pi x_2)+0.3$
6	[−100, 100]	2	MN	Bohachevsky3	$f(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1+4\pi x_2)+0.3$
7	[0, 180]	20	UN	Sinusoidal20	$f(x)=-\left[A\prod_{i=1}^{n}\sin(x_i-z)+\prod_{i=1}^{n}\sin(B(x_i-z))\right]$ $A=2.5, B=5, z=30$
8	[−100, 100]	30	UN	Quadric	$f(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$
9	[−100, 100]	2	UN	Easom	$f(x)=-\cos x_1\cos x_2\cdot\exp\left(-(x_1-\pi)^2\right)\exp(-\left(x_2-\pi)^2\right)$
10	[−10, 10]	2	UN	Matyas	$f(x)=0.26(x_1+x_2)-0.48x_1x_2$
11	[−5, 10]	10	UN	Zakharov	$f(x)=\sum_{i=1}^{n}(x_i)^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^4$
12	[−10, 10]	24	UN	Powell	$f(x)=\sum_{i=1}^{n/k}(x_{4i-3}+10x_{4i-2})^2+5(x_{4i-1}-x_{4i})^2+(x_{4i-2}-x_{4i-1})^4+10(x_{4i-3}-x_{4i})^4$
13	[−10, 10]	30	UN	Schwefel2.22	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$
14	[−30, 30]	30	UN	Rosenbrock	$f(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$
15	[−5.12, 5.12]	30	MS	Rastrigin	$f(x)=\sum_{i=1}^{n}x_i^2-10\cos(2\pi x_i)+10$

TABLE 1: Continued.

Number	Range	D	C	Functions	Description
16	[−100, 100]	2	MN	Schaffer1	$f(x) = 0.5 + \dfrac{\sin^2 \sqrt{x_1^2 + x_2^2}^{\,2} - 0.5}{\left[1 + 0.001\left(x_1^2 + x_2^2\right)\right]^2}$
17	[−100, 100]	30	MN	Schaffer2	$f(x) = \left(x_1^2 + x_2^2\right)^{0.25}\left(\sin^2\left(50(x_1^2 + x_2^2)^{0.1}\right) + 1\right)$
18	[−600, 600]	30	MN	Griewangk	$f(x) = \dfrac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\dfrac{x_i}{\sqrt{i}}\right) + 1$
19	[−32, 32]	30	MN	Ackley	$f(x) = -20\exp^{\left(-0.2\sqrt{\sum_{i=1}^{n}(x_i^2/n)}\right)} - \exp^{\left(\sum_{i=1}^{n}\cos(2\pi x_i/n)\right)} + 20 + e$
20	[−5, 5]	2	MN	Three hump camel back	$f(x) = 2x_1^2 - 1.05x_1^4 + \dfrac{1}{6}x_1^6 + x_1x_2 + x_2^2$
21	[−5, 5]	2	MN	Six hump camel back	$f(x) = 4x_1^2 - 2.1x_1^4 + \dfrac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
22	$[-128, 128]^n$	9	UN	Storn's Tchebychev	$f(x) = p_1 + p_2 + p_3,$
23	$[-32768, 32768]^n$	17		Storn's Tchebychev	where $p_1 = \begin{cases}(u-d)^2 & \text{if } u < d\\ 0 & \text{if } u \geq d\end{cases}\quad u = \sum_{i=1}^{n}(1.2)^{n-i}x_i$; $p_2 = \begin{cases}(v-d)^2 & \text{if } v < d\\ 0 & \text{if } v \geq d\end{cases}\quad v = \sum_{i=1}^{n}(-1.2)^{n-i}x_i$; $p_3 = \sum_{j=0}^{m}\begin{cases}(w_j-1)^2 & \text{if } w_j > 1\\ (w_j+1)^2 & \text{if } w_j < -1\\ 0 & \text{if } -1 \leq w_j \leq 1\end{cases}\quad w_j = \sum_{i=1}^{n}\left(\dfrac{2j}{m} - 1\right)^{n-i}x_i,$ for $n = 9$: $d = 72.661$, and $m = 60$ for $n = 17$: $d = 10558.145$, and $m = 100$.

D : dimension; C: characteristic; U: unimodal; S: seperable; N: non-separable.

Table 2: Simulation results of RIO, HRIO, CSO, MCSO, and ICSO.

SN	Fn.	Dim.	Opt.		RIO	HRIO	CSO	MCSO	ICSO
1	Boha1	2	0	Ave.	$3.4405E-05$	$3.2877E-04$	$2.9893E02$	$3.5153E-09$	0.0000
				STD	$2.5963E-05$	$3.0334E-04$	$5.0332E02$	$1.4392E-08$	0.0000
				Best	$1.3520E-07$	$5.2651E-06$	$2.0651E-05$	0.0000	0.0000
				Success	20/20	20/20	5/20	20/20	20/20
				Time	1.137525	0.886356	23.913237	0.075212	0.097187
2	Boha2	2	0	Ave.	$4.2829E-05$	$4.6703E-04$	$9.0941E02$	$8.4459E-12$	0.0000
				STD	$3.0070E-05$	$3.4047E-04$	$1.7794E03$	$2.9240E-11$	0.0000
				Best	$2.2910E-06$	$9.374E-06$	$1.3775E-05$	0.0000	0.0000
				Success	20/20	20/20	4/20	20/20	20/20
				Time	0.998178	0.946887	26.492095	0.072021	0.074106
3	Boha3	2	0	Ave.	$5.3479E-05$	$4.7575E-04$	$7.4284E02$	$2.1388E-14$	0.0000
				STD	$2.9141E-05$	$2.3273E-04$	$1.6739E03$	$4.8670E-14$	0.0000
				Best	$3.1200E-06$	$4.6981E-05$	$2.3093E-07$	0.0000	0.0000
				Success	20/20	20/20	3/20	20/20	20/20
				Time	1.089920	0.885252	25.028054	0.080908	0.068189
4	3camel	2	0	Ave.	$1.4962E-02$	$4.3021E-04$	$5.003E09$	$7.098E-11$	$5.9853E-31$
				STD	$6.6769E-02$	$2.8371E-04$	$1.7137E10$	$3.0201E-10$	$2.5457E-30$
				Best	$1.1739E-06$	$2.2449E-05$	$1.7642E-05$	$3.1395E-19$	$2.2320E-53$
				Success	19/20	20/20	12/20	20/20	20/20
				Time	4.231533	0.794983	18.281683	0.104132	0.078845
5	6camel	2	−1.03163	Ave.	$-4.3522E-01$	$-4.7652E-01$	$1.5763E05$	$-1.0263E-08$	$-2.9798E-25$
				STD	$3.3322E-01$	$3.1284E-01$	$7.0503E05$	$4.4391E-08$	$1.3325E-24$
				Best	−1.0215	−1.0034	$-9.4052E-01$	$-1.9879E-07$	$-5.9589E-24$
				Success	20/20	20/20	19/20	20/20	20/20
				Time	0.406355	0.330198	5.723039	0.0945856	0.086637
6	Easom	2	−1	Ave.	−1	−1	$-4.3165E-01$	−1	−1
				STD	$3.7518E-02$	$2.1031E-02$	$3.4470E-01$	$1.4897E-08$	$4.4116E-17$
				Best	−1	−1	−1	−1	−1
				Success	20/20	20/20	20/20	20/20	20/20
				Time	0.124022	0.107303	0.106738	0.077179	0.092393
7	Matyax	2	0	Ave.	$4.9470E-05$	$3.2297E-04$	7.5712	$2.6876E-13$	$4.0732E-35$
				STD	$3.0244E-05$	$2.6018E-04$	$1.1247E01$	$8.9347E-13$	$1.8125E-34$
				Best	$6.2897E-06$	$1.2684E-05$	$8.8777E-06$	$6.6695E-21$	$1.1292E-55$
				Success	20/20	20/20	11/20	20/20	20/20
				Time	0.973322	0.711734	13.559576	0.88536	0.076693
8	Schaffer1	2	−1	Ave.	−1.9069	−1.6211	$-2.9174E-01$	−1	−1
				STD	$7.0381E-01$	$5.9214E-01$	$7.5142E-01$	$5.9575E-07$	$4.1325E-15$
				Best	−2.7458	−2.7164	−2.7438	−1	−1
				Success	20/20	20/20	20/20	20/20	20/20
				Time	0.109048	0.086433	0.119076	0.072400	0.081599
9	Schaffer2	2	0	Ave.	$2.0179E-03$	$1.6566E-03$	7.1618	$3.3168E-04$	$2.2149E-09$
				STD	$2.6407E-03$	$1.4451E-03$	5.3095	$3.0328E-04$	$2.9483E-09$
				Best	$6.2423E-05$	$4.1422E-04$	$2.8354E-01$	$1.5810E-05$	$1.9383E-14$
				Success	2/20	13/20	0/20	20/20	20/20
				Time	62.567654	31.415836	29.194283	0.084127	0.082320

Dim. denotes dimension. Opt. denotes optimum value. Boha1 denotes Bohachevsky1. Boha2 denotes Bohachevsky2. Boha3 denotes Bohachevsky3. 3camel denotes three hump camel back. 6camel denotes six hump camel back.

Table 3: Simulation results of RIO, HRIO, CSO, MCSO, and ICSO.

SN	Fn.	Dim.	Opt.		RIO	HRIO	CSO	MCSO	ICSO
10	Sphere	30	0	Ave.	$2.2168E-05$	$1.6676E-04$	$1.8123E02$	$1.5201E-12$	$3.3448E-34$
				STD	$2.4528E-05$	$2.4018E-04$	$8.1048E02$	$6.7224E-12$	$1.3324E-33$
				Best	$5.7627E-09$	$5.5635E-08$	$4.9195E-07$	$2.9978E-24$	$2.8205E-54$
				Success	20/20	20/20	19/20	20/20	20/20
				Time	0.617544	0.557871	25.378161	0.82512	0.199373
11	Rastrigin	30	0	Ave.	$3.8135E-05$	$3.2150E-04$	$3.6022E03$	$9.1994E-11$	0.0000
				STD	$3.4436E-05$	$3.0003E-04$	$5.5728E03$	$3.9456E-10$	0.0000
				Best	$2.7098E-07$	$2.1450E-07$	$3.1340E-04$	0.0000	0.0000
				Success	20/20	20/20	5/20	20/20	20/20
				Time	0.956329	0.826770	71.811170	0.175563	0.369987
12	Rosenbrock	30	0	Ave.	$2.5281E06$	$3.3571E06$	$9.5067E11$	$2.9000E01$	$2.9000E01$
				STD	$4.0528E06$	$7.1150E06$	$2.2713E12$	0.0000	0.0000
				Best	$1.6773E04$	$3.7562E04$	$4.4068E01$	$2.9000E01$	$2.9000E01$
				Success	0/20	0/20	0/20	0/20	0/20
				Time	126.618734	127.469638	81.361663	76.084929	78.572185
13	Ackley	30	0	Ave.	$2.0001E01$	$2.0005E01$	$1.9222E01$	$5.1593E-06$	$1.0651E-15$
				STD	$3.0455E-03$	$1.5671E-02$	5.8258	$1.9149E-05$	$7.9441E-16$
				Best	$2.0001E01$	$1.9998E01$	$2.0133E01$	$6.4623E-09$	$8.1818E-16$
				Success	0/20	0/20	0/20	20/20	20/20
				Time	122.216187	117.635854	82.227210	0.235012	0.192339
14	Quadric	30	0	Ave.	$2.4498E-05$	$2.2711E-04$	$3.4991E-04$	$4.4754E-13$	$7.2183E-28$
				STD	$2.7957E-05$	$2.3635E-04$	$3.3725E-04$	$1.9751E-12$	$3.2218E-27$
				Best	$1.1360E-08$	$5.8230E-07$	$4.1551E-08$	$5.6309E-23$	$5.910E-52$
				Success	20/20	20/20	20/20	20/20	20/20
				Time	0.718785	0.512242	31.075809	0.247456	0.227244
15	Schwefel2.22	30	0	Ave.	$2.3131E02$	$2.4395E02$	$2.9013E54$	$6.3587E-06$	$6.0407E-16$
				STD	$1.3193E02$	$1.2341E02$	$1.2971E55$	$1.1936E-05$	$1.2203E-15$
				Best	$6.7400E01$	$1.7354E01$	$3.6854E01$	$5.9410E-08$	$5.1670E-24$
				Success	0/20	0/20	0/20	20/20	20/20
				Time	128.445013	127.084387	79.924516	0.217104	0.219296
16	Griewangk	30	0	Ave.	$7.9510E-01$	$7.7746E-01$	$2.6148E01$	$3.3151E-11$	0.0000
				STD	$3.7583E-01$	$2.5454E-01$	$3.6626E01$	$1.4672E-10$	0.0000
				Best	$2.9324E-01$	$3.2031E-01$	$6.3912E-05$	0.0000	0.0000
				Success	0/20	0/20	5/20	20/20	20/20
				Time	126.872461	126.210153	70.852376	0.211351	0.210934
17	Sumsquare	30	0	Ave.	$1.9818E03$	$4.6771E03$	$9.0499E05$	$4.2446E-11$	$1.5600E-24$
				STD	$2.8370E03$	$6.7104E03$	$1.0253E06$	$1.2930E-10$	$6.9785E-24$
				Best	$1.6463E01$	$2.0516E02$	$1.8730E02$	$1.49990E-16$	$1.3765E-47$
				Success	0/20	0/20	0/20	20/20	20/20
				Time	122.748646	125.154349	78.809270	0.273780	0.236129
18	Sinusoidal	30	-3.5	Ave.	$-4.2587E-01$	$-3.7898E-01$	-2.449	-3.1030	-3.1030
				STD	$2.6632E-01$	$1.9791E-01$	1.0203	$5.0473E-05$	$1.9436E-14$
				Best	-1.1922	$-8.3111E-01$	-3.3087	-3.1032	-3.1030
				Success	20/20	20/20	20/20	20/20	20/20
				Time	0.204559	0.240200	0.234205	0.200361	0.217635

Dim. denotes dimension. Opt. denotes optimum value.

Table 4: Simulation results of RIO, HRIO, CSO, MCSO, and ICSO.

SN	Function	Dim.	Opt.		RIO	HRIO	CSO	MCSO	ICSO
				Ave.	1.0167*E*04	1.0216*E*04	6.3663*E*18	2.3878*E* − 09	4.1579*E* − 26
				STD	3.8643*E*03	5.1012*E*03	2.2732*E*19	8.8529*E* − 09	1.8549*E* − 25
19	Zakharov	30	0	Best	2.6634*E*03	2.3151*E*03	1.3578*E*09	2.0954*E* − 15	6.3965*E* − 57
				Success	0/20	0/20	0/20	20/20	20/20
				Time	115.192226	114.691827	79.926232	0.205280	0.259202
				Ave.	0.0000	0.0000	2.0004*E*04	0.0000	0.0000
				STD	0.0000	0.0000	8.4815*E*04	0.0000	0.0000
20	Step	30	0	Best	0.0000	0.0000	0.0000	0.0000	0.0000
				Success	20/20	20/20	16/20	20/20	20/20
				Time	0.686403	0.633264	39.136696	0.239525	0.225102
				Ave.	1.8348*E* − 03	3.7434*E* − 03	1.0840*E*08	2.6031*E* − 12	1.8207*E* − 24
				STD	1.6248*E* − 03	6.1711*E* − 03	4.1180*E*08	6.9959*E* − 12	5.6824*E* − 24
21	Powell	24	0	Best	9.6693*E* − 05	6.8033*E* − 04	5.2392*E*01	1.2287*E* − 19	1.2265*E* − 54
				Success	2/20	12/20	0/20	20/20	20/20
				Time	122.796991	92.876086	74.794730	1.527170	0.853751
				Ave.	0.0000	0.0000	0.0000	0.0000	0.0000
				STD	0.0000	0.0000	0.0000	0.0000	0.0000
22	ST	9	0	Best	0.0000	0.0000	0.0000	0.0000	0.0000
				Success	20/20	20/20	20/20	20/20	20/20
				Time	0.435911	0.426320	0.437944	0.431122	0.436741
				Ave.	0.0000	0.0000	0.0000	0.0000	0.0000
				STD	0.0000	0.0000	0.0000	0.0000	0.0000
23	ST	17	0	Best	0.0000	0.0000	0.0000	0.0000	0.0000
				Success	20/20	20/20	20/20	20/20	20/20
				Time	1.066161	1.052169	1.159830	1.089657	1.147114

Dim. denotes dimension. Opt. denotes optimum value.

Table 5: Comparison of average performance of RIO, HRIO, CSO, MCSO, and ICSO.

SN	Function	RIO	HRIO	CSO	MCSO	ICSO	Optimum
1	Bohachevsky1	3.4405*E* − 05	3.2877*E* − 04	2.9893*E*02	3.5153*E* − 09	**0.0000**	0
2	Bohachevsky2	4.2829*E* − 05	4.6703*E* − 04	9.0941*E*02	8.4459*E* − 12	**0.0000**	0
3	Bohachevsky3	5.3479*E* − 05	4.7575*E* − 04	7.4284*E*02	2.1388*E* − 14	**0.0000**	0
4	3 Hump camel back	1.4962*E* − 02	4.3021*E* − 04	5.003*E*09	7.098*E* − 11	**5.9853E − 31**	0
5	6 Hump camel back	−4.3522*E* − 01	−4.7652*E* − 01	1.5763*E*05	−1.0263*E* − 08	**−2.9798E − 25**	−1.03163
6	Easom	**−1**	**−1**	−4.3165*E* − 01	**−1**	**−1**	−1
7	Matyax	4.9470*E* − 05	3.2297*E* − 04	7.5712	2.6876*E* − 13	**4.0732E − 35**	0
8	Schaffer1	−1.9069	−1.6211	−2.9174*E* − 01	**−1**	**−1**	−1
9	Schaffer2	2.0179*E* − 03	1.6566*E* − 03	7.1618	3.3168*E* − 04	**2.2149E − 09**	0
10	Sphere	2.2168*E* − 05	1.6676*E* − 04	1.8123*E*02	1.5201*E* − 12	**3.3448E − 34**	0
11	Rastrigin	3.8135*E* − 05	3.2150*E* − 04	3.6022*E*03	9.1994*E* − 11	**0.0000**	0
12	Rosenbrock	2.5281*E*06	3.3571*E*06	9.5067*E*11	**2.9000E01**	**2.9000E01**	0
13	Ackley	2.0001*E*01	2.0005*E*01	1.9222*E*01	5.1593*E* − 06	**1.0651E − 15**	0
14	Quadric	2.4498*E* − 05	2.2711*E* − 04	3.4991*E* − 04	4.4754*E* − 13	**7.2183E − 28**	0
15	Schwefel2.22	2.3131*E*02	2.4395*E*02	2.9013*E*54	6.3587*E* − 06	**6.0407E − 16**	0
16	Griewangk	7.9510*E* − 01	7.7746*E* − 01	2.6148*E*01	3.3151*E* − 11	**0.0000**	0
17	Sumsquare	1.9818*E*03	4.6771*E*03	9.0499*E*05	4.2446*E* − 11	**1.5600E − 24**	0
18	Sinusoidal	−4.2587*E* − 01	−3.7898*E* − 01	−2.449	**−3.1030**	**−3.1030**	−3.5
19	Zakharov	1.0167*E*04	1.0216*E*04	6.3663*E*18	2.3878*E* − 09	**4.1579E − 26**	0
20	Step	**0.0000**	**0.0000**	2.0004*E*04	**0.0000**	**0.0000**	0
21	Powell	1.8348*E* − 03	3.7434*E* − 03	1.0840*E*08	2.6031*E* − 12	**1.8207E − 24**	0
22	ST9	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0
23	ST17	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0
Number of good optimums		4	4	2	7	23	

ST9 denotes Storn's Tchebychev 9. ST17 denotes Storn's Tchebychev 17.

Table 6: Comparison of best performance of RIO, HRIO, CSO, MCSO, and ICSO.

SN	Function	RIO	HRIO	CSO	MCSO	ICSO	Optimum
1	Bohachevsky1	$1.3520E-07$	$5.2651E-06$	$2.0651E-05$	**0.0000**	**0.0000**	0
2	Bohachevsky2	$2.2910E-06$	$9.374E-06$	$1.3775E-05$	**0.0000**	**0.0000**	0
3	Bohachevsky3	$3.1200E-06$	$4.6981E-05$	$2.3093E-07$	**0.0000**	**0.0000**	0
4	3 hump camel back	$1.1739E-06$	$2.2449E-05$	$1.7642E-05$	$3.1395E-19$	**2.2320E − 53**	0
5	6 hump camel back	−1.0215	−1.0034	$-9.4052E-01$	$-1.9879E-07$	**5.9589E − 24**	−1.03163
6	Easom	**−1**	**−1**	**−1**	**−1**	**−1**	−1
7	Matyax	$6.2897E-06$	$1.2684E-05$	$8.8777E-06$	$6.6695E-21$	**1.1292E − 55**	0
8	Schaffer1	−2.7458	−2.7164	−2.7438	**−1**	**−1**	−1
9	Schaffer2	$6.2423E-05$	$4.1422E-04$	$2.8354E-01$	$1.5810E-05$	**1.9383E − 14**	0
10	Sphere	$5.7627E-09$	$5.5635E-08$	$4.9195E-07$	$2.9978E-24$	**2.8205E − 54**	0
12	Rosenbrock	$1.6773E04$	$3.7562E04$	$4.4068E01$	**2.9000E01**	**2.9000E01**	0
14	Quadric	$1.1360E-08$	$5.8230E-07$	$4.1551E-08$	$5.6309E-23$	**5.910E − 52**	0
15	Schwefel2.22	$6.7400E01$	$1.7354E01$	$3.6854E01$	$5.9410E-08$	**5.1670E − 24**	0
16	Griewangk	$2.9324E-01$	$3.2031E-01$	$6.3912E-05$	**0.0000**	**0.0000**	0
17	Sumsquare	$1.6463E01$	$2.0516E02$	$1.8730E02$	$1.49990E-16$	**1.3765E − 47**	0
18	Sinusoidal	−1.1922	$-8.3111E-01$	**−3.3087**	−3.1032	−3.1030	−3.5
19	Zakharov	$2.6634E03$	$2.3151E03$	$1.3578E09$	$2.0954E-15$	**6.3965E − 57**	0
20	Step	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0
21	Powell	$9.6693E-05$	$6.8033E-04$	$5.2392E01$	$1.2287E-19$	**1.2265E − 54**	0
22	ST9	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0
23	ST17	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0
Number of good optimums		4	4	5	11	22	

ST9 denotes Storn's Tchebychev 9. ST17 denotes Storn's Tchebychev 17.

Table 7: Comparison of standard deviation of mean global optimum values of RIO, HRIO, CSO, MCSO, and ICSO.

SN	Function	RIO	HRIO	CSO	MCSO	ICSO
1	Bohachevsky1	$2.5963E-05$	$3.0334E-04$	$5.0332E02$	$1.4392E-08$	**0.0000**
2	Bohachevsky2	$3.0070E-05$	$3.4047E-04$	$1.7794E03$	$2.9240E-11$	**0.0000**
3	Bohachevsky3	$2.9141E-05$	$2.3273E-04$	$1.6739E03$	$4.8670E-14$	**0.0000**
4	3 hump camel back	$6.6769E-02$	$2.8371E-04$	$1.7137E10$	$3.0201E-10$	**2.5457E − 30**
5	6 hump camel back	$3.3322E-01$	$3.1284E-01$	$7.0503E05$	$4.4391E-08$	**1.3325E − 24**
6	Easom	$3.7518E-02$	$2.1031E-02$	$3.4470E-01$	$1.4897E-08$	**4.4116E − 17**
7	Matyax	$3.0244E-05$	$2.6018E-04$	$1.1247E01$	$8.9347E-13$	**1.8125E − 34**
8	Schaffer1	$7.0381E-01$	$5.9214E-01$	$7.5142E-01$	$5.9575E-07$	**4.1325E − 15**
9	Schaffer12	$2.6407E-03$	$1.4451E-03$	5.3095	$3.0328E-04$	**2.9483E − 09**
10	Sphere	$2.4528E-05$	$2.4018E-04$	$8.1048E02$	$6.7224E-12$	**1.3324E − 33**
11	Rastrigin	$3.4436E-05$	$3.0003E-04$	$5.5728E03$	$3.9456E-10$	**0.0000**
12	Rosenbrock	$4.0528E06$	$7.1150E06$	$2.2713E12$	**0.0000**	**0.0000**
13	Ackley	$3.0455E-03$	$1.5671E-02$	5.8258	$1.9149E-05$	**7.9441E − 16**
14	Quadric	$2.7957E-05$	$2.3635E-04$	$3.3725E-04$	$1.9751E-12$	**3.2218E − 27**
15	Schwefel2.22	$1.3193E02$	$1.2341E02$	$1.2971E55$	$1.1936E-05$	**1.2203E − 15**
16	Griewangk	$3.7583E-01$	$2.5454E-01$	$3.6626E01$	$1.4672E-10$	**0.0000**
17	Sumsquare	$2.8370E03$	$6.7104E03$	$1.0253E06$	$1.2930E-10$	**6.9785E − 24**
18	Sinusoidal	$2.6632E-01$	$1.9791E-01$	1.0203	$5.0473E-05$	**1.9436E − 14**
19	Zakharov	$3.8643E03$	$5.1012E03$	$2.2732E19$	$8.8529E-09$	**1.8549E − 25**
20	Step	0.0000	0.0000	$8.4815E04$	**0.0000**	**0.0000**
21	Powell	$1.6248E-03$	$6.1711E-03$	$4.1180E08$	$6.9959E-12$	**5.6824E − 24**
22	ST9	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
23	ST17	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Number of good STD		2	2	2	4	23

ST9 denotes Storn's Tchebychev 9. ST17 denotes Storn's Tchebychev 17.

Table 8: Comparison of success performance of RIO, HRIO, CSO, MCSO, and ICSO.

SN	Function	RIO	HRIO	CSO	MCSO	ICSO
1	Bohachevsky1	1	1	2.5	1	1
2	Bohachevsky2	1	1	0.2	1	1
3	Bohachevsky3	1	1	0.15	1	1
4	3 hump camel back	0.95	1	0.6	1	1
5	6 hump camel back	1	1	0.95	1	1
6	Easom	1	1	1	1	1
7	Matyax	1	1	0.55	1	1
8	Schaffer1	1	1	1	1	1
9	Schaffer2	0.1	0.65	0	1	1
10	Sphere	1	1	0.95	1	1
11	Rastrigin	1	1	0.25	1	1
12	Rosenbrock	0	0	0	0	0
13	Ackley	0	0	0	1	1
14	Quadric	1	1	1	1	1
15	Schwefel2.22	0	0	0	1	1
16	Griewangk	0	0	0.25	1	1
17	Sumsquare	0	0	0	1	1
18	Sinusoidal	1	1	1	1	1
19	Zakharov	0	0	0	1	1
20	Step	1	1	0.8	1	1
21	Powell	0.1	0.6	0	1	1
22	ST9	1	1	1	1	1
23	ST17	1	1	1	1	1
Number of 100% success rates		14	15	6	22	22

ST9 denotes Storn's Tchebychev 9. ST17 denotes Storn's Tchebychev 17.

Table 9: Comparison of exec1ution time of RIO, HRIO, CSO, MCSO, and ICSO.

SN	Function	RIO	HRIO	CSO	MCSO	ICSO
1	Bohachevsky1	1.137525	0.886356	23.913237	**0.075212**	0.097187
2	Bohachevsky2	0.998178	0.946887	26.492095	**0.072021**	0.074106
3	Bohachevsky3	1.089920	0.885252	25.028054	0.080908	**0.068189**
4	3 hump camel back	4.231533	0.794983	18.281683	0.104132	**0.078845**
5	6 hump camel back	0.406355	0.330198	5.723039	0.0945856	**0.086637**
6	Easom	0.124022	0.107303	0.106738	**0.077179**	0.092393
7	Matyax	0.973322	0.711734	13.559576	0.88536	**0.076693**
8	Schaffer1	0.109048	0.086433	0.119076	**0.072400**	0.081599
9	Schaffer2	62.567654	31.415836	29.194283	0.084127	**0.082320**
10	Sphere	0.617544	0.557871	25.378161	0.82512	**0.199373**
11	Rastrigin	0.956329	0.826770	71.811170	**0.175563**	0.369987
12	Rosenbrock	126.618734	127.469638	81.361663	**76.084929**	78.572185
13	Ackley	122.216187	117.635854	82.227210	0.235012	**0.192339**
14	Quadric	0.718785	0.512242	31.075809	0.247456	**0.227244**
15	Schwefel2.22	128.445013	127.084387	79.924516	**0.217104**	0.219296
16	Griewangk	126.872461	126.210153	70.852376	0.211351	**0.210934**
17	Sumsquare	122.748646	125.154349	78.809270	0.273780	**0.236129**
18	Sinusoidal	0.204559	0.240200	0.234205	**0.200361**	0.217635
19	Zakharov	115.192226	114.691827	79.926232	**0.205280**	0.259202
20	Step	0.686403	0.633264	39.136696	0.239525	**0.225102**
21	Powell	122.796991	92.876086	74.794730	1.527170	**0.853751**
22	ST9	0.435911	**0.426320**	0.437944	0.431122	0.436741
23	ST17	1.066161	**1.052169**	1.159830	1.089657	1.147114
Number of minimum execution times		—	2	—	9	12

ST9 denotes Storn's Tchebychev 9. ST17 denotes Storn's Tchebychev 17.

Table 10: Jonckheere-Terpstra test statistics[a].

	Fitness
Number of levels in algorithm	5
N	114
Observed J-T statistic	1952.000
Mean J-T statistic	2599.500
STD of J-T statistic	199.355
Standard data of J-T statistic	−3.245
Asymp. Sig. (2-tailed)	0.001

[a]Grouping variable: algorithm.

4. Conclusion

Cockroach swarm optimization algorithm is extended in this paper with a new component called hunger component. Hunger component enhances the algorithm diversity and searching capability. An improved cockroach swarm optimization algorithm is proposed. The efficiency of the proposed algorithm is shown through empirical studies where its performance was compared with that of existing algorithms, that is, CSO, MSCO, RIO, and HRIO. Results show its outstanding performance compared to the existing algorithms. Application of the algorithm to real life problems can be considered in further studies.

Conflict of Interests

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

Acknowledgment

The authors thank College of Agriculture, Engineering and Science, University of KwaZulu-Natal, for supporting this paper through bursary award.

References

[1] S. Camazine, J. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, *Self-Organization in Biological Systems*, Princeton University Press, Princeton, NJ, USA, 2001.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *IEEE Neural Networks Proceedings*, vol. 4, pp. 1942–1948, 1995.

[3] F. Yoshikazu, N. Hideyuki, and T. Yuji, "Particle swarm optimization for the optimal operational planning of energy plants," in *Innovations in Swarm Intelligence Studies in Computational Intelligence*, C. P. Lim, L. C. Jain, and S. Dehuri, Eds., pp. 159–174, Springer, Berlin, Germany, 2009.

[4] P. Chen, "Particle swarm optimization for power dispatch with pumped hydro," in *Particle Swarm Optimization*, A. Lazinica, Ed., pp. 131–144, In-Tech, Zagreb, Croatia, 2009.

[5] I. Omar and B. Cees, "A particle optimization approach to graph permutation," in *Particle Swarm Optimization*, A. Lazinica, Ed., pp. 291–312, In-Tech, Zagreb, Croatia, 2009.

[6] M. Dorigo, *Optimization, learning and natural algorithms [Ph.D. thesis]*, Politecnico di Milano, Milan, Italy, 1992.

[7] A. Rizzoli A, R. Montemanni, E. Lucibello, and L. Gambardella, *Ant Colony Optimization for Real-World Vehicle Routing Problems: from Theory to Applications*, Springer, 2007.

[8] A. A. A. Radwan, T. M. Mahmoud, and E. H. Hussein, "AntNet-RSLR: a proposed Ant routing protocol for MANETs," in *Proceedings of the IEEE Saudi International Electronics, Communications and Photonics Conference (SIECPC '11)*, April 2011.

[9] D. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "The bees algorithm," Tech. Rep., Manufacturing Engineering Centre, Cardiff University, Cardiff, UK, 2005.

[10] 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.

[11] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, vol. 31, no. 1–4, pp. 61–85, 2009.

[12] B. Basturk and D. Karaboga, "An Artificail Bees Colony (ABC) algorithm for numeric computation," in *Proceedings of the IEEE Swarm Intellience Symposium*, Indianapolis, Ind, USA, 2006.

[13] T. C. Havens, C. J. Spain, N. G. Salmon, and J. M. Keller, "Roach infestation optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '08)*, September 2008.

[14] C. ZhaoHui and T. HaiYan, "Cockroach swarm optimization," in *Proceedings of the 2nd International Conference on Computer Engineering and Technology (ICCET '10)*, vol. 6, pp. 652–655, April 2010.

[15] C. ZhaoHui, "A modified cockroach swarm optimization," *Energy Procedia*, vol. 11, p. 49, 2011.

[16] I. C. Obagbuwa, A. O. Adewumi, and A. A. Adebiyi, "A dynamic step-size adaptation roach infestation optimization," in *Proceedings of the IEEE International Conference on Advance Computing (IACC '14)*, Gurgaon, Indian, 2014.

[17] L. Cheng, Z. Wang, S. Yanhong, and A. Guo, "Cockroach swarm optimization algorithm for TSP," *Advanced Engineering Forum*, vol. 1, pp. 226–229, 2011.

[18] C. ZhaoHui and T. HaiYan, "Cockroach swarm optimization for vehicle routing problems," *Energy Procedia*, vol. 13, pp. 30–35, 2011.

[19] I. C. Obagbuwa, A. O. Adewumi, and A. A. Adebiyi, "Stochastic constriction cockroach swarm optimization for multidimensional space function problems," *Mathematical Problems in Engineering*, vol. 2014, Article ID 430949, 12 pages, 2014.

[20] J. B. Williams, M. Louis, R. Christine, and A. Nalepal, *Cock-Roaches Ecology, Behaviour and Natural History*, Johns Hopkins University Press, Baltimore, Md, USA, 2007.

[21] M. Kerckhove, "From population dynamics to partial differential equations," *The Mathematica Journal*, vol. 14, 2012.

[22] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems," *Journal of Global Optimization*, vol. 31, no. 4, pp. 635–672, 2005.

[23] 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.

[24] P. Civicioglu and E. Besdok, "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artificial Intelligence Review*, vol. 39, pp. 1–32, 2013.

[25] J. Cohen, *Statistical Power Analysis for the Behavioural Science*, L. Erlbaum Associates, Hillsdale, NJ, USA, 2nd edition, 1988.

[26] J. Cohen, "Statistical power analysis for the behavioural science," *Current Direction in Psychological Science*, vol. 1, no. 3, pp. 98–101, 1992.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 428539, 9 pages
http://dx.doi.org/10.1155/2014/428539

Research Article

Ant Colony Optimization Algorithm for Continuous Domains Based on Position Distribution Model of Ant Colony Foraging

Liqiang Liu,[1] Yuntao Dai,[2] and Jinyu Gao[1]

[1] *College of Automation, Harbin Engineering University, 145 Nantong Street, Heilongjiang 150001, China*
[2] *College of Science, Harbin Engineering University, 145 Nantong Street, Heilongjiang 150001, China*

Correspondence should be addressed to Liqiang Liu; llq9842222@126.com

Received 20 February 2014; Accepted 30 March 2014; Published 11 May 2014

Academic Editors: S. Balochian and Y. Zhang

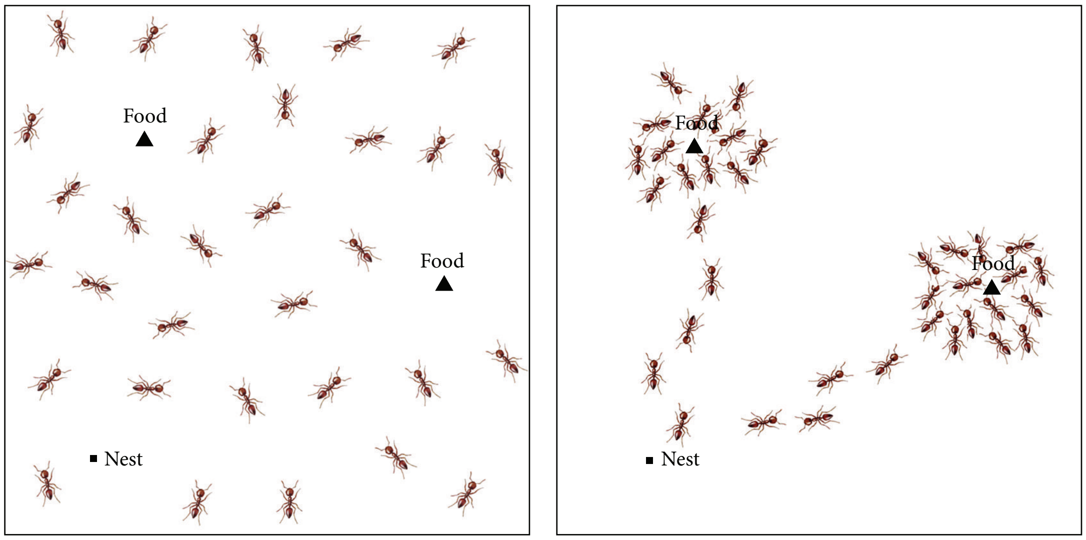
Copyright © 2014 Liqiang Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Ant colony optimization algorithm for continuous domains is a major research direction for ant colony optimization algorithm. In this paper, we propose a distribution model of ant colony foraging, through analysis of the relationship between the position distribution and food source in the process of ant colony foraging. We design a continuous domain optimization algorithm based on the model and give the form of solution for the algorithm, the distribution model of pheromone, the update rules of ant colony position, and the processing method of constraint condition. Algorithm performance against a set of test trials was unconstrained optimization test functions and a set of optimization test functions, and test results of other algorithms are compared and analyzed to verify the correctness and effectiveness of the proposed algorithm.

1. Introduction

Optimization is a kind of application technology using mathematical method to study how to search for the optimal solution for the problem in numerous solutions, as an important branch of science, which has been a widespread concern, and the rapid popularization and application in industrial production, economic and other fields. In the 1940s, with the increasingly widespread application of high-speed digital computers, optimization theory and algorithms developed rapidly and formed a new discipline. In recent years, swarm intelligence optimization theory has gradually developed into a new research direction of optimization techniques, typical algorithms with genetic algorithm [1], particle swarm optimization [2], ant colony optimization algorithm [3], artificial bee colony algorithm [4], firefly algorithm [5], and bat algorithm [6].

Inspired by the real ant colony foraging behavior in nature, early in the 1990s, the Italian scholars Dorigo et al. proposed ant colony optimization algorithm [3]. The algorithm adopts the distributed control, self-organizing, and positive feedback, and the optimization process does not depend on rigorous mathematical properties of optimization problem in itself and has the potential parallelism. Research on ant colony algorithm has shown that superiority of the algorithm for solving complex optimization problems. Because the ant colony optimization algorithm is essentially a kind of discrete optimization ideas, so the study of the optimization algorithm is mainly aimed at the problems of discrete domain optimization. But in real life, there are many optimization problems that are usually expressed as optimization problems of continuous domains. Therefore, how essentially discrete ant colony optimization algorithm would be applied to solve the optimization problems of continuous domains has become a new direction for research on ant colony optimization algorithm. In recent years, the studies of ant colony optimization algorithm for continuous domains have obtained some achievements and many scholars have proposed a variety of ant colony optimization algorithms for continuous domain [7–17]. Bilchev and Parmee first proposed a continuous ant colony optimization algorithm CACO [7], the algorithm for solving problems using genetic algorithms for global search of the solution space firstly and then using the ant colony optimization algorithm for local optimization to all the results. Dréo and Siarry proposed continuous interactive ant colony optimization algorithm CIAC



(a) Ant colony position distribution in the initial moment (b) Ant colony position distribution after a period of time

Figure 1: Process of ant colony foraging.

[8], the algorithm modify the way of pheromone update and rules of path-searching, and use two ways of pheromone communication to guide ant optimization. Monmarché et al. proposed API algorithm [9]; All the ants set out from the same starting point, and each ant uses a complementary strategy that carried out optimization independently. Socha and Dorigo, who proposed continuous domain ant colony optimization algorithm ACO_R [10], used a Gaussian kernel probability density function express as distribution model of pheromone and gave ACO_R metaheuristic framework.

This paper proposes position distribution model of ant colony foraging and designs ant colony optimization algorithm for continuous domains based on the model to solve the standard test functions to verify the correctness and effectiveness of the algorithm. This paper is organized as follows. The relationship between the position distribution and food source in the process of ant colony foraging is analyzed in Section 2, and the position distribution model of ant colony foraging is given. To solve the unconstrained optimization problems and constrained optimization problems for ant colony optimization algorithm of continuous domains is designed in Section 3. The algorithm performance test trials and comparative analysis are given in Section 4. The conclusion is given in Section 5.

2. Position Distribution Model of Ant Colony Foraging

In the process of real-world ant colony foraging, people find that ant colony have a built-in optimization capability: they always can find the shortest path from nest to food. By studying this phenomenon, people propose the ant colony optimization algorithm.

We can see the process of ant colony foraging from another perspective. As shown in Figure 1, an individual ant has no guidance of pheromone in the initial of foraging and searches for food sources blindly in the whole space; at this point, ant colony is distributed uniformly in the continuous space. As the process for feeding food, ants aggregated around the source will be increased, and the density of pheromone will increase in the vicinity, thus raising more ants to the food source. Also, the higher quality of the food source will attract a greater number of ants. Thus, in the process of ant colony foraging, the position distribution of ant colony and food source and quality is the same.

We can give such a model through the above process of analysis and expansion: assuming the food source is everywhere throughout in the continuous space, the quality of food source is different. At the initial moment, ants of ant colony distribute uniformly in the continuous space and release pheromones according to food sources of their position. The higher the quality of the food source, the more the pheromone ants released. The pheromone is distributed throughout the continuous space in a certain dispersed model, and ants perceive spatial concentration of pheromone intensity, moving to the position of a higher concentration of pheromone in a certain way and achieve the exploration of unknown regions during the move. The movement of the single ant will cause the change of the whole position distribution of ant colony, so that all the ants keep aggregating to the higher quality of food source and search the highest quality of food source in the continuous space eventually. This model is called position distribution model of ant colony foraging.

3. Ant Colony Algorithm of Continuous Domains Design

Below, we discuss the design process of ant colony optimization algorithm of continuous domain for solving unconstrained optimization problems and constrained optimization problems based on position distribution model of ant colony foraging.

3.1. Design of Algorithm for Solving Unconstrained Optimization Problem

3.1.1. Expression of Solution. Assuming the whole ant colony consists of m groups of substructure, each group contains n of ants. As shown in the following equation:

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ \text{ant}_{11} & \text{ant}_{12} & \cdots & \text{ant}_{1n} \\ \text{ant}_{21} & \text{ant}_{22} & \cdots & \text{ant}_{2n} \\ \vdots & \vdots & & \vdots \\ \text{ant}_{m1} & \text{ant}_{m2} & \cdots & \text{ant}_{mn} \end{bmatrix}, \tag{1}$$

the position ant_{ij} corresponding to the value x_j of the variable for j-ant in any subcolony i, the subcolony i of all the ants in the sequence of $\{\text{ant}_{i1}, \text{ant}_{i2}, \ldots, \text{ant}_{in}\}$ represents a solution of the optimization problem.

3.1.2. Distribution Model of Pheromones. In the position distribution model of ant colony foraging, each ant releases pheromone according to the quality of a food source of their position; pheromones are dispersed in the entire space, with increasing distance of the source and the concentration decreasing. Therefore, we need to choose a probability density function as distribution model of ant pheromone in optimization algorithm of continuous domains. Gaussian function is a common probability density function; we assume ants of the ant colony release pheromone externally on the function. At this point, j ant in any subcolony ant i corresponding to pheromone distribution model $\tau_{ij}(x)$ can be expressed as

$$\tau_{ij}(x) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-((x-\mu_{ij})^2/2\sigma_j^2)}, \qquad \sigma_j = \frac{(u_j - l_j)}{\psi \cdot (1 + \ln(n))}, \tag{2}$$

where μ_{ij} is the position ant_{ij} of ant j in the subcolony of ants i, namely, the distribution center, σ_j ($\sigma_j > 0$) means the width of the distribution function, u_j is the maximum allowable value of the variable x_j, l_j is the minimum allowable value of the variable x_j, n is the dimension of solution for the optimization problem, ψ ($\psi > 0$) is a parameter, and σ_j is used to adjust size.

3.1.3. Updating Position of Ant Colony. Before updating the position of ant colony, we need to choose a group as a parent from m subcolony. First, we use formula (3) to calculate each group of subcolony corresponding to the assessed value of solution. Consider the following:

$$\text{eval}_i = \frac{1}{\left(1 + e^{f(\text{ant}_{i1}, \text{ant}_{i2}, \ldots, \text{ant}_{in})/T}\right)}, \tag{3}$$

where $f(\text{ant}_{i1}, \text{ant}_{i2}, \ldots, \text{ant}_{in})$ is the assessment value of the subcolony ant i; T ($T > 0$) is the adjustment coefficient used to adjust the pressure of selection.

After assessment value for each group of subcolony is obtained, we calculate the selected probability for each group of subcolony according to

$$p_i = \frac{\text{eval}_i}{\sum_{j=1}^{m} \text{eval}_j}. \tag{4}$$

Finally, we select parent colony c according to formula (5)

$$c = \begin{cases} \arg \max\limits_{i=1,2,\ldots,m} (\text{eval}_i), & q \le q_0, \\ C & q > q_0, \end{cases} \tag{5}$$

where q_0 ($0 \le q_0 \le 1$) is a given parameter, q is a random variable which distributed in $[0, 1]$ uniformly. C is a random variable which is generated according to formula (4).

After getting the parent ant colony c, the ant pheromone distribution model function $\tau_{cj}(x)$ in the ant colony corresponding to random number generator for sampling, the k groups of children colony are generated. Then, according to the size of assessment value for each group of subcolony, we select the large assessment value of m group from $(m + k)$ group of subcolony in order to achieve position of ant colony update.

3.2. Algorithm of Solving Constrained Optimization Problem. First, we define a solution x of measure constrained optimization problem violate measure for the degree of constraint condition:

$$\text{viol}(x) = \sum_{j=1}^{r} c_j(x). \tag{6}$$

For inequality constraint $g_j(x) \le 0$, $c_j(x) = \max\{0, g_j(x)\}$. For equality constraint $h_j(x) = 0$,

$$c_j(x) = \begin{cases} |h_j(x)|, & |h_j(x)| > h_{j\,\text{Min}} \\ 0, & |h_j(x)| \le h_{j\,\text{Min}}, \end{cases} \tag{7}$$

where $h_{j\,\text{Min}}$ is a small positive number. $\text{viol}(x)$ is equal to zero represent x is feasible solution. $\text{viol}(x)$ which is greater than 0 represent x is infeasible solution.

When using this algorithm for solving constrained optimization problems, we allow infeasible solutions with probability $p\,\text{Max}$ ($0 \le p\,\text{Max} \le 1$) existing. Algorithm calculation process is consistent with Section 3.1, and we only adjust the update process of the position of ant colony to the following process.

(1) Calculate the number $e\text{Num} = (m+k) \times p\,\text{Max}$ of the maximum expected infeasible solutions in the group $(m + k)$ of subcolony.

(2) Calculate the number $r\text{Num}$ of real infeasible solutions based on the value $\text{viol}(x)$ in the group $(m + k)$ of ant colony.

(3) If $r\text{Num}$ is less than $e\text{Num}$, then reserve the maximum of the assessment for group m of ant colony

Table 1: Parameter values.

Parameter	m	k	q_0	ψ
Values	100	50	0.9	4

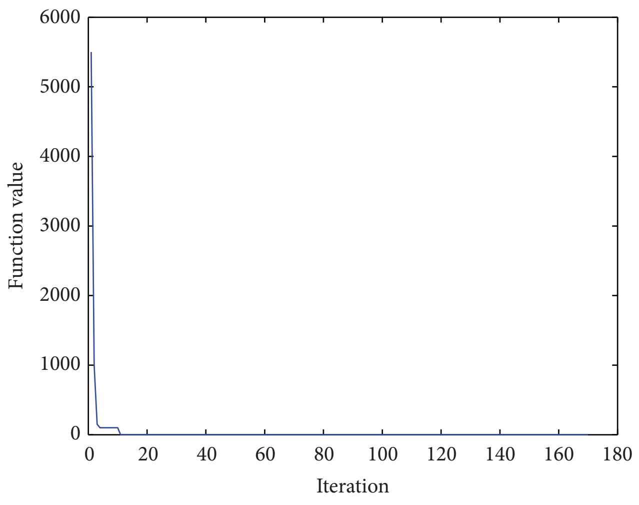


Figure 2: Curves of minimum function.

directly. If rNum is greater than eNum, the infeasible solutions are ranked by the value of viol(x), the greater number (rNum − eNum) of assessment value for the infeasible solutions from the value viol(x) is set to 0, and then reserve m group of the maximum fitness for $(m + k)$ group of ant colony according to the assessment value.

4. Testing and Analysis of Algorithm Performance

4.1. Solution of Unconstrained Optimization Problems. In the process of solving unconstrained optimization problems algorithm performance testing, we refer to [10] method; the entire test is divided into three groups; the use of this algorithm with a kind of probabilistic learning methods, a kind of continuous domains ant colony algorithm, and a kind of metaheuristic methods is compared. The operating parameters of the algorithm design in this paper are shown in Table 1.

4.1.1. Compare with a Kind of Learning of Probability Method. In this experiment, we use the algorithm in this paper to compare with [10, 21–24] which used a kind of probabilistic learning method for performance. In order to ensure the fairness of the results of comparison, the entire test method according to [10, 21] is given. The baseline function of this set of tests is given in Table 2. The stop condition of the algorithm is satisfied in

$$\left|f - f^*\right| < \varepsilon_{\min}, \quad (8)$$

where f is the optimal solution for algorithm, f^* is the known optimal solution, and $\varepsilon_{\min}$ needs to satisfy the accuracy, taken as 10^{-10}.

The comparative test results are shown in Table 3, where the results of other methods for solving are provided by [10, 21]. In the data of Table 3, the "1.0" represents the best algorithm for solving the extreme value of the basis functions. The actual median number of function evaluations is given in parentheses. Other algorithms corresponding evaluation data are the ratio of the evaluation number of the function and the best algorithms function when the stop condition is satisfied. "∞" represents the use of the algorithm that can not seek to satisfy the stop condition. The results marked "$*$" represent the use of the algorithm to get the corresponding extreme value of the basis functions, not to satisfy stop condition results are found every time.

By the test results, it can be found that the algorithm has better searching capability and faster speed of convergence. In the process of solving seven of the basis functions, four functions of solution have results significantly better than other probability learning algorithms.

4.1.2. Compare with a Kind of Ant Colony Algorithm of Continuous Domains. In this experiment, we use the algorithm in this paper and a kind of ant colony of continuous domains in [10] for performance comparison. The method of test is given according to [10]. The basis functions of this test are given in Table 4. The stop condition of algorithm is satisfied in

$$\left|f - f^*\right| < \varepsilon_1 \cdot f + \varepsilon_2, \quad (9)$$

where ε_1 is relative error, the value is 10^{-4}, ε_2 is the absolute error, and the value is 10^{-4}.

The results of comparative tests are shown in Table 5, where the percentage in brackets represents the minimum value of the independent use of the method for solving the corresponding basis functions 100 times, and ultimately the number of the stop conditions satisfied as a percentage of the total number of the algorithms is obtained. The symbol "—"represents that the algorithm is not used for solving the corresponding minimum of basis function; there is no data available for reference.

The results of this test prove that algorithm of this paper has better searching capability and faster speed of convergence. But we also find that the stability of the algorithm in this paper is relatively worse. In the process of solving the minimum of six basis functions, there are five solving functions which cannot guarantee that each stop solution condition satisfied the required accuracy.

4.1.3. Compare with a Kind of Metaheuristic Method. In this experiment, we use the algorithm in this paper and a kind of metaheuristic method in [10] for performance comparison. The test is carried out according to methods given in [10]. The basis functions of this set of tests are shown in Table 6. In this experiment, except for the three basis functions given in Table 6, the function also uses B_2 function, GP function, and R_2 and R_5 functions given in Table 4.

The results of comparative tests are shown in Table 7. We can find that algorithm of this paper has better searching capability and faster speed of convergence. But it also expose the instability of the algorithm.

Table 2: Basis functions of test 1 [10].

Function	Formula ($n = 10$)	Range	Optimum $f(x)$
Plane (PL)	$f_{\mathrm{PL}}(\vec{x}) = x_1$	$\vec{x} \in [0.5, 1.5]^n$	1.5
Diagonal Plane (DP)	$f_{\mathrm{DP}}(\vec{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i$	$\vec{x} \in [0.5, 1.5]^n$	1.5
Sphere (SP)	$f_{\mathrm{SP}}(\vec{x}) = \sum_{i=1}^{n} x_i^2$	$\vec{x} \in [-3, 7]^n$	0
Ellipsoid (EL)	$f_{\mathrm{EL}}(\vec{x}) = \sum_{i=1}^{n}\left(100^{(i-1)/(n-1)} x_i\right)^2$	$\vec{x} \in [-3, 7]^n$	0
Cigar (CG)	$f_{\mathrm{CG}}(\vec{x}) = x_1^2 + 10^4\sum_{i=2}^{n} x_i^2$	$\vec{x} \in [-3, 7]^n$	0
Tablet (TB)	$f_{\mathrm{TB}}(\vec{x}) = 10^4 x_1^2 + \sum_{i=2}^{n} x_i^2$	$\vec{x} \in [-3, 7]^n$	0
Rosenbrock (Rn)	$f_{\mathrm{Rn}}(\vec{x}) = \sum_{i=1}^{n-1} 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2$	$\vec{x} \in [-5, 5]^n$	0

Table 3: Results of test 1.

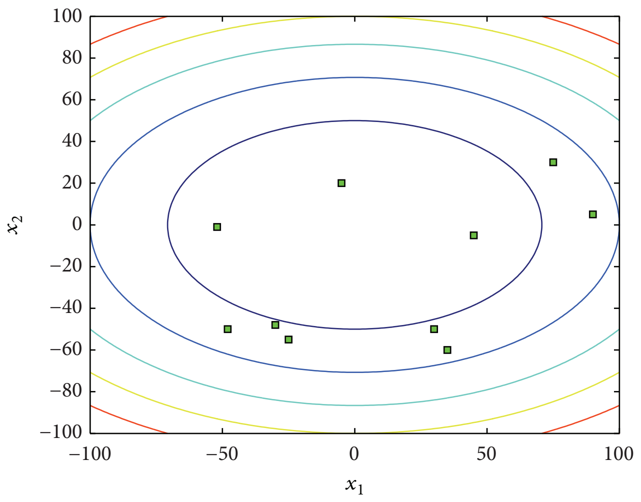
Function	This paper	(1 + 1) ES	CSA-ES	CMA-ES	IDEA
PL	**1.0** (15)	52.5	84	75.5	∞
DP	**1.0** (58)	14.4	21.7	18.8	∞
SP	**1.0** (199)	6.9	11	8.9	34.4
EL	3.2	66	110	**1.0** (4450)	1.6
CG	60.1	610	80	**1.0** (3840)	4.6
TB	**1.0** (550)	214.7	303.4	7.9	13.5
Rn	4.7*	51*	180	**1.0** (7190)	210*

Table 4: Basis functions of test 2 [10].

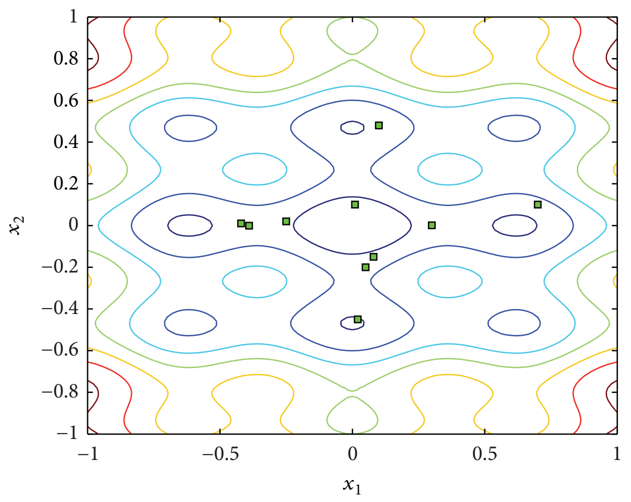
Function	Formula	Range	Optimum $f(x)$
B_2	$f_{B_2}(\vec{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	$\vec{x} \in [-100, 100]^n$ $n = 2$	0
Goldstein and Price	$f_{\mathrm{GP}}(\vec{x}) = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	$\vec{x} \in [0.5, 1.5]^n$ $n = 2$	3
Sphere model	$f_{\mathrm{SM}}(\vec{x}) = \sum_{i=1}^{n} x_i^2$	$\vec{x} \in [-5.12, 5.12]^n$ $n = 6$	0
Martin and Gaddy	$f_{\mathrm{MG}}(x) = (x_1 - x_2)^2 + \left(\frac{x_1 + x_2 - 10}{3}\right)^2$	$\vec{x} \in [-20, 20]^n$ $n = 2$	0
Rosenbrock	$f_{\mathrm{Rn}}(\vec{x}) = \sum_{i=1}^{n-1} 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2$	$\vec{x} \in [-5, 10]^n$ $n = 2.5$	0

Table 5: Results of test 2.

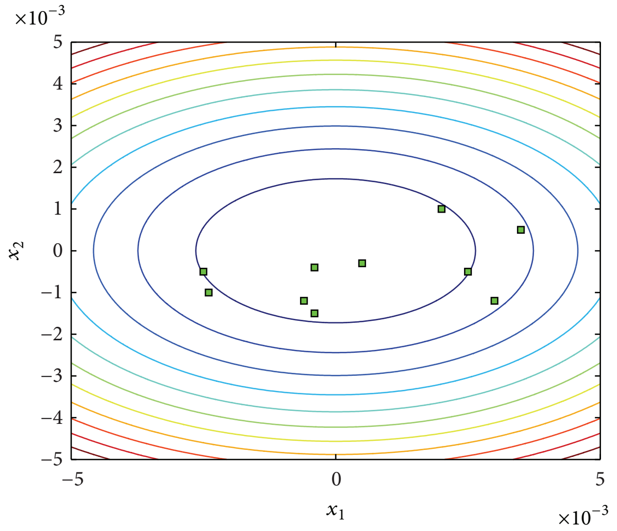
Function	This paper	$\mathrm{ACO_R}$	CACO	API	CIAC
R_2	**1.0** [95%] (62)	13.2	109.8	158.7	185.2
SM	**1.0** (69)	11.3	316.9	147.1	724.4
GP	**1.0** [97%] (54)	7.1	99.6	—	433.8 [56%]
MG	**1.0** [99%] (53)	6.5	32.5	—	221.3 [20%]
B_2	**1.0** [95%] (80)	6.8	—	—	149.6
R_5	1.4 [78%]	**1.0** [97%] (2487)	—	—	16 [90%]



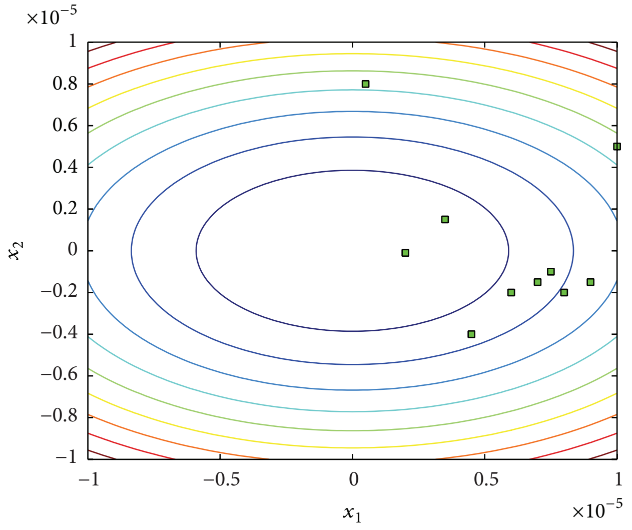
(a) Distribution of initial ant colony



(b) Distribution of the 50th iteration ant colony



(c) Distribution of the 100th iteration ant colony



(d) Distribution of the 158th iteration ant colony

FIGURE 3: Change of the distribution of ant colony.

TABLE 6: Basis functions of test 3 [10].

Function	Formula	Range	Optimum $f(x)$
Easom	$f_{\mathrm{ES}}(\vec{x}) = -\cos(x_1)\cos(x_2)e^{-((x_1-\pi)^2+(x_2-\pi)^2)}$	$\vec{x} \in [-100, 100]^n$ $n = 2$	-1
DeJong	$f_{\mathrm{DJ}}(\vec{x}) = x_1^2 + x_2^2 + x_3^2$	$\vec{x} \in [-5.12, 5.12]^n$ $n = 3$	0
Zakharov	$f_{\mathrm{Zn}}(\vec{x}) = \left(\sum_{i=1}^{n} x_i^2\right) + \left(\sum_{i=1}^{n} \frac{ix_i}{2}\right)^2 + \left(\sum_{i=1}^{n} \frac{ix_i}{2}\right)^4$	$\vec{x} \in [-5, 10]^n$ $n = 2.5$	0

When solving the minimum of the basis function Easom, stop condition to satisfy the accuracy requirements of the solution is found in the process of the algorithm independently running 100 times in only 43.

The algorithm for solving the convergence curve of the minimum of B_2 function is shown in Figure 2. After the 158th iteration in the algorithm of this paper, the values of function have been less than 10^{-10}; then the optimal solution has been found in the algorithm.

In the process of solving the function B_2 in Figure 3, each group of ant colony corresponding to the solution with the change of the distribution of algorithm iteration

Table 7: Results of test 3.

Function	This paper	CGA	ECTS	ESA	DE
B_2	**1.0** [95%] (80)	5.4	—	—	—
Easom	**1.0** (75) [43%]	19.6	—	—	—
GP	**1.0** [97%] (54)	7.7	4.3	14.5	—
R_2	**1.0** [95%] (62)	15.5	7.7	13.2	10.1
Z_2	**1.0** [98%] (48)	13	4.1	329.1	—
DJ	**1.0** (56)	13.3	—	—	7
R_5	1.7 [78%]	1.9	**1.0** (2142)	2.5	—
Z_5	**1.0** (81)	17	27.8	861.6	—

Table 8: Parameter value.

Parameter	m	k	q_0	ψ	pMax
Value	100	50	0.1	4	0.2

is shown. Where the initial distribution of ant colony is shown in Figure 3(a), all initial ant colony corresponding positions are distributed in $[-100, 100]$. With the operation of the algorithm, each group of ant colony rapidly approaches the optimal solution; in the 50th iteration, each ant of all the ant colony has been distributed in $[-1, 1]$. In the 158th step, the results of the algorithm for solving have satisfied stop conditions; then each ant colony is distributed in $[-10^{-5}, 10^{-5}]$, and there are two groups of ant colony of overlapping position.

4.2. Solution of Constrained Optimization Problems. In this set of test experiments, we use this algorithm for solving the basis function G01~G12 of constrained conditions, and [18–20] are compared. In order to guarantee the fairness of the test results, the method of test is consistent with the methods of [18–20] adopted. Using the algorithm for solving various basis functions 50 times independently, best results are compared. In the process of running each algorithm, if the optimal value function obtained by consecutive 150 times does not change, the running algorithm exits. Otherwise, the algorithm exited after iteration 30,000 times.

During solving the basis test function of constrained conditions, the parameter values of the algorithm in this paper are shown in Table 8.

The results of comparative tests are shown in Table 9.

We can see from the test results that effect of the algorithms in this paper for solving functions G01 and G02 was poor and solving the extreme values of function G03~G06, G08, G09, G11, and G12 gets minimum. It is evident that the algorithm in this paper for solving constrained optimization problems is effective. The algorithm in this paper for solving the maximum convergence curve of function G08 is shown in Figure 4.

In the process of solving, the feasible solution is found in the 17th iteration; after the 19th iteration, each group of subcolony corresponding solution is a feasible solution; the maximum value 0.095825 is founded by algorithm in the 157th iteration; all 10 groups of ant colony have found the maximum value in the 210th iteration.

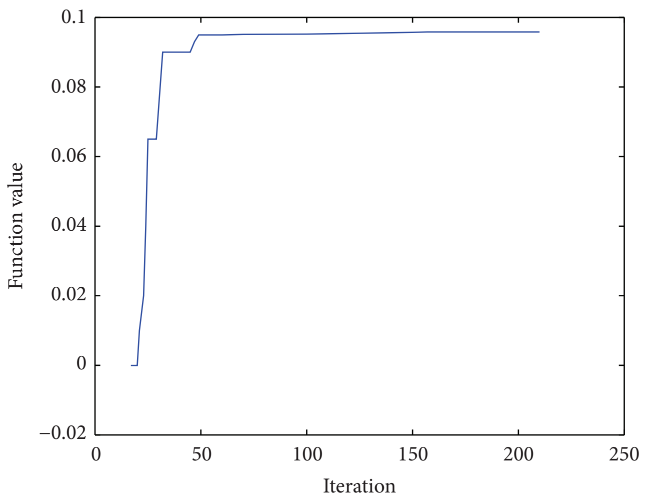


Figure 4: Maximum value curve of function.

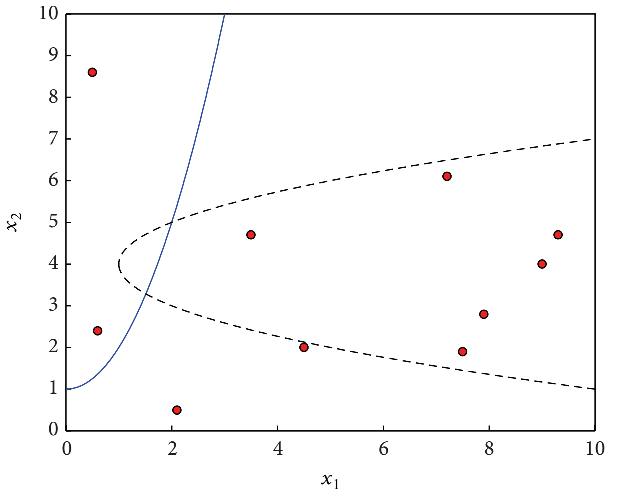
The process of the algorithm in this paper for solving function G08 shown in Figure 5. Distribution changes of each group of ant colony corresponding to the solution with algorithm iteration are shown. The initial distribution of ant colony is shown in Figure 5(a); position of all ant colony does not satisfy all constraint conditions. There have been 7 groups of ant colony corresponding to the position that satisfied the constraint condition in the 25th iteration (Figure 5(b)); all 10 groups of ant colony corresponding to the position are distributed in $x_1 \in [1.227968, 1.227973]$, $x_2 \in [4.245396, 4.245377]$, and there is a group of ant colony that had found the optimal solution in the 157th iteration (Figure 5(d)).

5. Conclusion

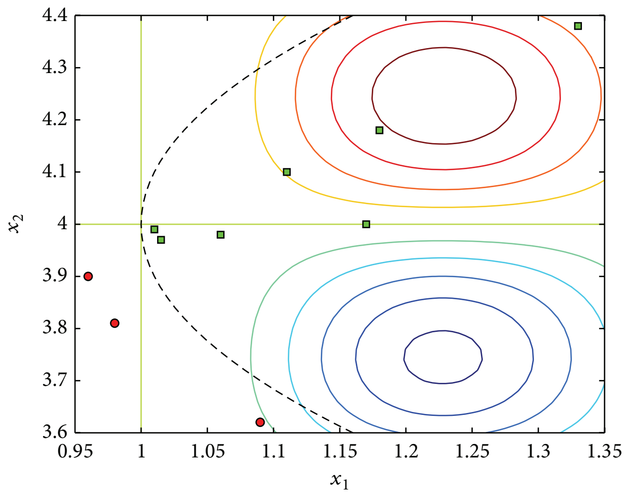
In this paper, in the process of position distribution relationship between food sources of ant colony foraging for analysis, a new position distribution model by ant foraging is proposed. Any point in the solution space could be seen as a food source in the model, using multiple groups of subcolony for optimization; each group of subcolony represented a solution of the problem. In every iteration step, a group of ant colony was chosen from all subcolonies as the parent ant colony firstly and then sampled from pheromone

TABLE 9: Results of test functions for solving constrained optimization problems [18–20].

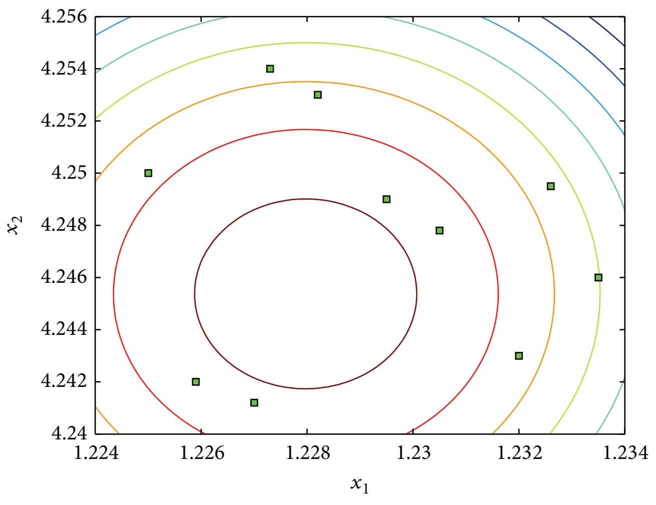
Function	Known optimal	This paper	ES_{SR}	KM	DP	PEPS_S
G01	−15.000	−13.934798	**−15.000**	−14.7864	**−15.000**	**−15.000**
G02	−0.803619	−0.781996	−0.803515	−0.79953	−0.803587	−0.803540
G03	−1.000	**−1.000**	**−1.000**	−0.9997	−0.583	**−1.000**
G04	−30665.539	**−30665.539**	**−30665.539**	−30664.5	−30365.488	−30665.538
G05	5126.498	5126.498	5126.497	—	—	5126.508
G06	−6961.814	**−6961.814**	**−6961.814**	−6952.1	−6911.247	**−6961.814**
G07	24.306	24.329	24.307	24.620	24.309	24.308
G08	−0.095825	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
G09	680.630	**680.630**	**680.630**	680.91	680.632	680.631
G10	7049.331	7078.146	7054.316	7147.9	—	7081.068
G11	0.750	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**
G12	−1.000000	**−1.000000**	**−1.000000**	−0.999999857	**−1.000000**	**−1.000000**



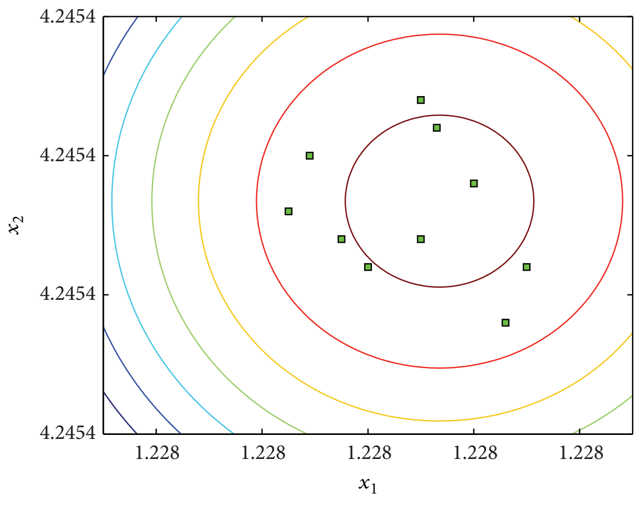
(a) Distribution of initial ant colony



(b) Distribution of the 25th iteration ant colony



(c) Distribution of the 100th iteration ant colony



(d) Distribution of the 157th iteration ant colony

FIGURE 5: Distribution changes of ant colony.

density function of the group, generated children colony, and finally updated position of ant colony, so that each group of subcolony continued moving towards the solution space of the higher fitness value, converging to the optimal solution eventually. By simulating the above process, we designed ant colony optimization algorithm of continuous domains; a set of test functions for unconstrained optimization problems and a set of test functions optimization comparison test were compared and gave the solving process of the B_2 test function and test function G08. Test results show that, in solving unconstrained optimization problems, the proposed algorithm has better searching capability and faster speed of convergence, but the stability of the algorithm is poor; when solving constrained optimization problems, the proposed algorithm has the basic optimization capability consistent with other algorithms.

Conflict of Interests

The authors declare that they have no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant nos. 51009036, 51109041, 51109045, and 51379049, Postdoctoral Foundation of Heilongjiang Province under Grant no. LBH-Z10217, Foundation of Central University HEUCF041216, and Foundation of Central University HEUCFX41302.

References

[1] D. Bunnag and M. Sun, "Genetic algorithm for constrained global optimization in continuous variables," *Applied Mathematics and Computation*, vol. 171, no. 1, pp. 604–636, 2005.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[3] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 26, no. 1, pp. 29–41, 1996.

[4] T. Thomas, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, New York, NY, USA, 1996.

[5] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2008.

[6] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds., vol. 284 of *Studies in Computational Intelligence*, pp. 65–74, Springer, Berlin, Germany, 2010.

[7] G. Bilchev and I. C. Parmee, "The ant colony metaphor for searching continuous design spaces," in *Evolutionary Computing*, T. C. Fogarty, Ed., vol. 993 of *Lecture Notes in Computer Science*, pp. 25–39, Springer, Berlin, Germany, 1995.

[8] J. Dréo and P. Siarry, "Continuous interacting ant colony algorithm based on dense heterarchy," *Future Generation Computer Systems*, vol. 20, no. 5, pp. 841–856, 2004.

[9] N. Monmarché, G. Venturini, and M. Slimane, "On how Pachycondyla apicalis ants suggest a new search algorithm," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 937–946, 2000.

[10] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[11] L. Chen, J. Shen, and L. Qin, "A method for solving optimization problem in continuous space by using ant colony algorithm," *Journal of Software*, vol. 13, no. 12, pp. 2317–2323, 2002.

[12] Y. Yang, X.-F. Song, J.-F. Wang, and S.-X. Hu, "Ant colony algorithm for continuous space optimization," *Control and Decision*, vol. 18, no. 5, pp. 573–576, 2003.

[13] Z.-G. Cheng, D.-Z. Chen, and X.-H. Wu, "Study of continuous ant colony optimization algorithm," *Journal of Zhejiang University (Engineering Science)*, vol. 39, no. 8, pp. 1147–1151, 2005.

[14] Z.-G. Cheng, D.-Z. Chen, and X.-H. Wu, "Continuous ant colony optimization system based on normal distribution model of pheromone," *Systems Engineering and Electronics*, vol. 28, no. 3, pp. 458–462, 2006.

[15] L. Wang and Q.-D. Wu, "Ant system algorithm in continuous space optimization," *Control and Decision*, vol. 18, no. 1, pp. 45–57, 2003.

[16] X.-L. Kou, S.-Y. Liu, and J.-K. Zhang, "Stochastic ant colony algorithm for continuous space optimization," *Systems Engineering and Electronics*, vol. 28, no. 12, pp. 1909–1911, 2006.

[17] H.-B. Duan, G.-J. Ma, D.-B. Wang, and X.-F. Yu, "Improved ant colony algorithm for solving continuous space optimization problems," *Journal of System Simulation*, vol. 19, no. 5, pp. 974–977, 2007.

[18] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000.

[19] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

[20] M. Yuchi, J.-H. Kim, and J. Jo, "A population ecology inspired parent selection strategy for numerical constrained optimization problems," *Applied Mathematics and Computation*, vol. 190, no. 1, pp. 292–304, 2007.

[21] S. Kern, S. D. Müller, N. Hansen, D. Büche, J. Očenášek, and P. Koumoutsakos, "Learning probability distributions in continuous evolutionary algorithms—a comparative review," *Natural Computing*, vol. 3, no. 1, pp. 77–112, 2004.

[22] A. Ostermeier, A. Gawelczyk, and N. Hansen, "Step-size adaptation based on non-local use of selection information," in *Parallel Problem Solving from Nature—PPSN III*, Y. Davidor, H. P. Schwefel, and R. Männer, Eds., vol. 866 of *Lecture Notes in Computer Science*, pp. 189–198, 1994.

[23] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[24] P. A. N. Bosman and D. Thierens, "Continuous iterated density estimation evolutionary algorithms within the IDEA framework," in *Proceedings of the Optimization by Building and Using Probabilistic Models Workshop at the Genetic and Evolutionary Computation Conference (GECCO '00)*, M. Pelikan, H. Mühlenbein, and A. O. Rodriguez, Eds., pp. 197–200, Morgan-Kaufmann Publishers, San Francisco, Calif, USA, 2000.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 854520, 8 pages
http://dx.doi.org/10.1155/2014/854520

Research Article

Crude Oil Price Forecasting Based on Hybridizing Wavelet Multiple Linear Regression Model, Particle Swarm Optimization Techniques, and Principal Component Analysis

Ani Shabri[1] and Ruhaidah Samsudin[2]

[1] *Department of Science Mathematic, Faculty of Science, Universiti Teknologi Malaysia, 81310 Johor, Malaysia*
[2] *Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia (UTM), 81310 Johor, Malaysia*

Correspondence should be addressed to Ani Shabri; ani_sabri@hotmail.com

Received 31 December 2013; Accepted 18 March 2014; Published 8 May 2014

Academic Editors: S. Balochian, V. Bhatnagar, and Y. Zhang

Copyright © 2014 A. Shabri and R. Samsudin. 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.

Crude oil prices do play significant role in the global economy and are a key input into option pricing formulas, portfolio allocation, and risk measurement. In this paper, a hybrid model integrating wavelet and multiple linear regressions (MLR) is proposed for crude oil price forecasting. In this model, Mallat wavelet transform is first selected to decompose an original time series into several subseries with different scale. Then, the principal component analysis (PCA) is used in processing subseries data in MLR for crude oil price forecasting. The particle swarm optimization (PSO) is used to adopt the optimal parameters of the MLR model. To assess the effectiveness of this model, daily crude oil market, West Texas Intermediate (WTI), has been used as the case study. Time series prediction capability performance of the WMLR model is compared with the MLR, ARIMA, and GARCH models using various statistics measures. The experimental results show that the proposed model outperforms the individual models in forecasting of the crude oil prices series.

1. Introduction

Crude oil prices do play significant role in the global economy and constitute an important factor affecting government's plans and commercial sectors. Forecasting crude oil price is among the most important issues facing energy economists. Therefore, proactive knowledge of its future fluctuations can lead to better decisions in several managerial levels.

The literature dealing with forecasting crude oil is substantial. The application of the classical time series models such as autoregressive moving average (ARMA) (Yu et al. [1], Mohammadi and Su [2], and Ahmad [3]) and econometric model such as generalized autoregressive conditional heteroscedasticity (GARCH) type models (Agnolucci [4], Wei et al. [5], Liu and Wan [6]) for crude oil forecasting has received much attention in the last decade. But because the crude oil price has the volatility, nonlinearity, and irregularity, the classical and econometric model can lead to the decrease of the accuracy.

Due to the limitations of the classical and econometric models, soft-computing models, such as neural fuzzy (Ghaffari and Zare [7]), artificial neural networks (Kaboudan [8], Mirmirani and Li [9], Shambora and Rossiter [10], and Yu et al. [11]), support vector machines (Xie et al. [12]), and genetic programming (GP), provide powerful solutions to nonlinear crude oil price prediction. Many experiments found that the soft-computing models often had some advantages over statistical-based models. However, these AI models also have their own shortcomings and disadvantages. For example, ANN often suffers from local minima and over-fitting, while other soft-computing models, such as SVM and GP, including ANN, are sensitive to parameter selection [1].

To remedy the above shortcomings, some hybrid methods have been used recently to predict crude oil price and obtain the best performances. In last year, wavelet transform has become a useful method for analyzing such as variations, periodicities, and trends in time series. The hybrid models with wavelet transform processes have been improved for forecasting. For example wavelet-neural network (Jammazi and Aloui [13], Qunli et al. [14], and Yousefi et al. [15]), wavelet-least square support vector machines (LSVM) (Bao et al. [16]), and wavelet-fuzzy neural network (Liu et al. [17]) have been employed recently on some studies in crude oil forecasting. They observed that the wavelet transform fairly improves forecasting accuracy.

A major drawback of wavelet transform for direction prediction is that the input variables lie in a high-dimensional feature space depends on the number of sub-time series components. Because the number of sub-time series components for wavelet is inadvisable to be too many, in this study principal component analysis (PCA) is proposed to reduce the dimensions of sub-time series components.

The multiple linear regressions (MLR) model that is much easier to interpret is considered as an alternative to ANN model. In this paper, a hybrid wavelet multiple linear regression (WMLR) model integrating wavelet and MLR is proposed for short-term daily crude oil price forecasting. The study applies particle swarm optimization (PSO) to adopt the optimal parameters to construct the MLR model. For verification purpose, the West Texas Intermediate (WTI) crude oil sport price is used to test the effectiveness of the proposed WMLR ensemble learning methodology. Finally to evaluate the model ability, the proposed model was compared with individual ARIMA and GARCH models.

2. Methodology

2.1. The ARIMA Model. The most comprehensive of all popular and widely known statistical methods used for time series forecasting are Box-Jenkins models (Box and Jenkins [18]). It has achieved great success in both academic research and industrial applications during the last three decades. The general form of ARIMA models can be expressed as

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i e_{t-i} + e_t, \tag{1}$$

where p is the order of the autoregressive, q is the order of the moving average, and e_t is the random error. The Box-Jenkins methodology is basically divided into four steps: identification, estimation, diagnostic checking, and forecasting.

2.2. The GARCH Model. GARCH models have found extensive application in the literature and the most popular volatility model is GARCH (1, 1) model proposed by Bollerslev [19]. The standard GARCH (1, 1) can be described as follows:

$$\begin{gathered} r_t = \mu_t + \varepsilon_t = \mu_t + h_t^{1/2}\eta_t \quad \eta_t \sim N(0,1), \\ h_t = \omega + \alpha\varepsilon_{t-1}^2 + \beta h_{t-1}, \end{gathered} \tag{2}$$

where μ_t denote the conditional mean and h_t is the conditional variances and η_t is a standardized error and $r_t = \ln(x_t/x_{t-1})$ is log return.

2.3. Multiple Linear Regressions. Multiple linear regressions (MLR) model is one of the modelling techniques to investigate the relationship between a dependent variable and several independent variables. Let the MLR have p independent variables with n observations. Thus the MLR can be written as

$$Y = w_0 + w_1 x_i + w_2 x_2 + \cdots + w_p x_p + \varepsilon_t, \tag{3}$$

where w are regression coefficients, Y is dependent variable, x_i are independent varaiables and ε_t is fitting errors. The method of least squares is generally used to estimate the coefficients model. In many applications, the results of a least squares fit are often unacceptable when the model is wrong or when the model is misspecified (Bozdogan and Howe [20]).

In this study, particle swarm optimization (PSO) method is presented to determine the optimal parameters of the MLR model. The PSO methods have proven to be very effective in solving a variety of difficult global optimization problems in forecasting (Chen and Kao [21] and Alwee et al. [22]), heat problem (Ma et al. [23] and Tyagi and Pandit [24]), and dynamic environments (Liu et al. [25]).

The classic solution of MLR model involves the minimization of the sum of the square errors between the model-predicted value and the corresponding data value:

$$\min f(w) = \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_i\right)^2, \tag{4}$$

where n is the number of training data samples, Y_i is the actual value, and $\widehat{Y}_i$ is the forecasted value of train data. The same methodology was used to solve this problem using PSO algorithms. The solution with a smaller fitness $f(w)$ of the training data set has a better chance of surviving in the successive generations.

2.4. Particle Swarm Optimization. Particle swarm optimization (PSO) is a population-based heuristic method inspired by the collective motion of biological organisms, such as bird flocking and fish schooling, to simulate the seeking behavior to a food source (Bratton and Kennedy [26]). The population of PSO is called a swarm and each individual in the population of PSO is called a particle. The PSO begins with a random population and searchers for fitness optimum just like genetic algorithm (GA). To find the optimum solution, each particle adjusts the direction through the best experience which it has found (p_{best}) and the best experience that has been found by all other members (g_{best}). Therefore, the particles fly around in a multidimensional space towards the better area over the search process.

Each particle consists of three vectors: the position for ith individual particle can be denoted as $X_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(D)})$, the best previous position p_{best} that the ith particle has searched is $P_i = (p_i^{(1)}, p_i^{(2)}, \ldots, p_i^{(D)})$, and the fly velocity of the ith is $V_i = (v_i^{(1)}, v_i^{(2)}, \ldots, v_i^{(D)})$. The performance of each particle is measured using a fitness function varying from problem in hand. During the iterative procedure, the ith particle at iteration t is updated by

$$\begin{aligned} v_i^d(t+1) &= \omega' \times v_i^d(t) + c_1 \times \varphi_1 \times \left[p_i^d(t) - x_i^d(t)\right] \\ &\quad + c_2 \times \varphi_2 \times \left[p_g^d(t) - x_i^d(t)\right], \\ x_i^d(t+1) &= x_i^d(t) + v_i^d(t+1), \end{aligned} \tag{5}$$

where ω is called inertia weight, c_1 and c_2 are acceleration constants, and φ_1 and φ_2 are stochastic value of $[0,1]$. In a PSO system, particles change their positions at each time step until a relatively unchanging position has been encountered or a maximum number of iterations have been met. In general, the performance of each particle is measured according to a fitness function, which is problem dependent. In MLR model, (4) is the fitness function under consideration. Figure 1 shows the flowchart of the developed PSO algorithm. For further details regarding PSO, please refer to Kennedy and Eberhart [27] and Bratton and Kennedy [26].

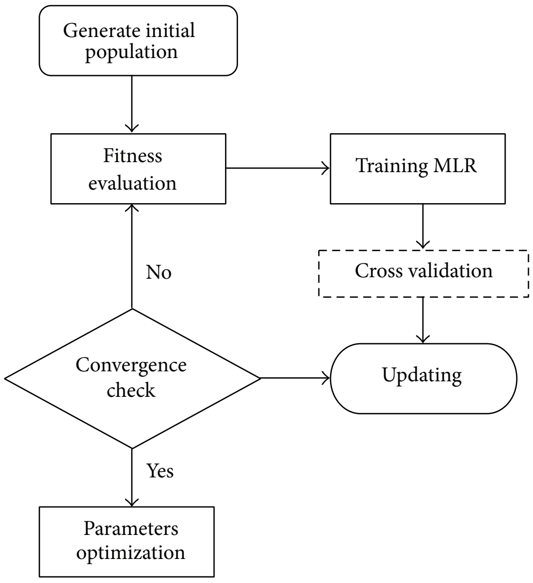


Figure 1: Flowchart of PSO algorithm.

2.5. Wavelet Analysis. Wavelet transformations provide useful decomposition of original time series by capturing useful information on various decomposition levels. Discrete wavelet transformation (DWT) is preferred in most of the forecasting problems because of its simplicity and ability to compute with less time. The DWT can be defined as

$$\psi_{m,n}\left(\frac{t-\tau}{s}\right) = \frac{1}{\sqrt{s_0^{m/2}}}\psi\left(\frac{t - n\tau_0 s_0^m}{s_0^m}\right), \tag{6}$$

where m and n are integers that control the scale and time. The most common choices for the parameters $s_0 = 2$ and $\tau_0 = 1$. $\psi(t)$ called the mother wavelet can be defined as $\int_{-\infty}^{\infty}\psi(t)dt = 0$.

For a discrete time series $x(t)$ where $x(t)$ occurs at discrete time t, the DWT becomes

$$W_{m,n} = 2^{-m/2}\sum_{t=0}^{N-1}\psi\left(2^{-m}t - n\right)x(t), \tag{7}$$

where $W_{m,n}$ is the wavelet coefficient for the discrete wavelet at scale $s = 2^m$ and $\tau = 2^m n$. According to Mallat's theory, the original discrete time series $x(t)$ can be decomposed into a series of linearity independent approximation and detail signals by using the inverse DWT. The inverse DWT is given by (Mallat [28])

$$x(t) = T + \sum_{m=1}^{M}\sum_{t=0}^{2^{M-m-1}} W_{m,n}2^{-m/2}\psi\left(2^{-m}t - n\right) \tag{8}$$

or in a simple format as

$$x(t) = A_M(t) + \sum_{m=1}^{M} D_m(t), \tag{9}$$

where $A_M(t)$ is called approximation subseries or residual term at levels M and $D_m(t)$ $(m = 1, 2, \ldots, M)$ are detail subseries which can capture small features of interpretational value in the data.

2.6. Principal Component Analysis. In an MLR, one of main tasks is to determine the model input variables that affect the output variables significantly. The choice of input variables is generally based on a priori knowledge of causal variables, inspections of time series plots, and statistical analysis of potential inputs and outputs. PCA is a technique widely used for reducing the number of input variables when we have huge volume of information and we want to have a better interpretation of variables (Çamdevýren et al. [29]).

The PCA approach introduces a few combinations for model input in comparison with the trial and error process. Given a set of centred input vectors $x_1, x_2, \ldots, x_m$ and $\sum_{t=1}^{m} x_t = 0$, usually $n < m$. Then the covariance matrix of vector is given by

$$C = \frac{1}{l}\sum_{t=1}^{l} x_t x_t^T. \tag{10}$$

The principal components (PCs) are computed by solving the eigenvalue problem of covariance matrix C,

$$\lambda_i u_i = C u_i, \quad i = 1, 2, \ldots, m, \tag{11}$$

where λ_i is one of the eigenvalues of C and u_i is the corresponding eigenvector. Based on the estimated u_i, the components of $z_t(i)$ are then calculated as the orthogonal transforms of x_t:

$$z_t(i) = u_i^T x_t, \quad i = 1, 2, \ldots, m. \tag{12}$$

The new components, $z_i(t)$, are called principal components. By using only the first several eigenvectors sorted in descending order of the eigenvalues, the number of principal components in z_t can be reduced. So PCA has the dimensional reduction characteristic. The principal components of PCA have the following properties: $z_t(i)$ are linear combinations of the original variables, uncorrelated and have sequentially maximum variances (Jolliffe [30]). The calculation variance contribution rate is

$$V_i = \frac{\lambda_i}{\sum_{i=1}^{m} \lambda_i} \times 100\%. \tag{13}$$

The cumulative variance contribution rate is

$$V(p) = \sum_{i=1}^{p} V_i. \tag{14}$$

The number of the selected principal components is based on the cumulative variance contribution rate, which as a rule is over 85~90.

3. Computer Simulation

3.1. An Application. In this study, the West Texas Intermediate (WTI) crude oil price series was chosen as experimental sample. The main reason of selecting the WTI crude oil is that these crude oil prices are the most famous benchmark prices, which are widely used as the basis of many crude oil price formulae. The daily data from January 1, 1986, to September 30, 2006, excluding public holidays, with a total of 5237 was employed as experimental data. For convenience of WMLR modeling, the data from January 1, 1986, to December 31, 2000, is used for the training set (3800 observations), and the remainder is used as the testing set (1437 observations). Figure 2 shows the daily crude oil prices from January 1, 1986, to September 30.

In practice, short-term forecasting results are more useful as they provide timely information for the correction of forecasting value. In this study, three main performance criteria are used to evaluate the accuracy of the models. These criteria are mean absolute error (MAE), root mean squared error (RMSE), and D_{stat}. The MAE and RMSE can be defined by

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(y_t - \hat{y}_t\right)^2},$$
$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}\left|y_t - \hat{y}_t\right|. \tag{15}$$

In crude oil price forecasting, improved decisions usually depend on correct forecasting of directions, of actual price, y_t and forecasted price, $\hat{y}_t$. The ability to predict movement direction can be measured by a directional statistic (D_{stat}) (Yu et al., [1]), which can be expressed as

$$D_{\text{stat}} = \frac{1}{N}\sum_{t=1}^{n} a_t \times 100\%,$$
$$a_t = \begin{cases} 1, & \text{if } \left(y_{t+1} - y_t\right)\left(\hat{y}_{t+1} - \hat{y}_t\right) \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

3.2. Application and Result. At first, the MLR model without data preprocessing was used to model daily oil prices. In

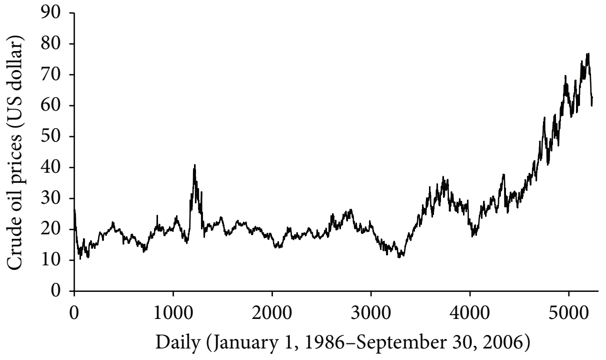


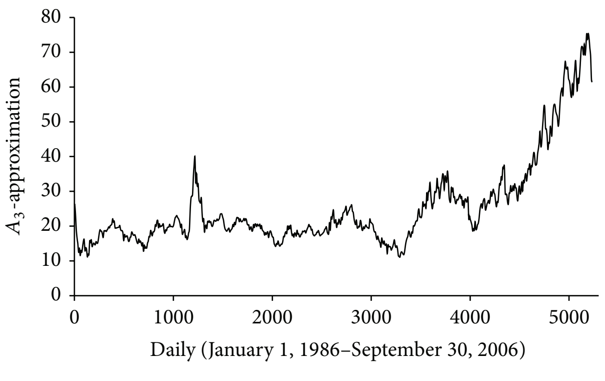
Figure 2: Daily crude oil prices from January 1, 198, to September 30, 2006.

the next step, the preprocessed data which uses subtime series components obtained using discrete wavelet transform (DWT) on original data were entered to the MLR model in order to improve the model accuracy. For the MLR model, the original log return time series are decomposed into a certain number of subtime series components. Deciding the optimal decomposition level of the time series data in wavelet analysis plays an important role in preserving the information and reducing the distortion of the datasets. However, there is no existing theory to tell how many decomposition levels are needed for any time series.

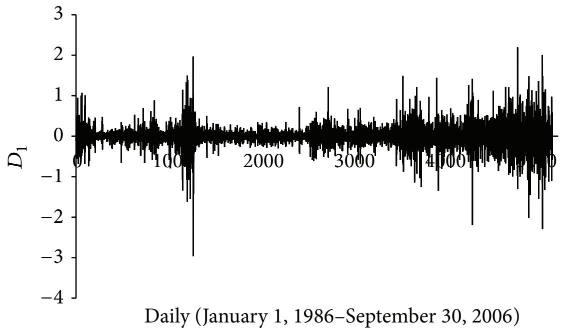
In the present study, the previous log return of daily oil price time series is decomposed into various subtime series (DWs) at different decomposition levels by using DWT to estimate current price value. Three decomposition levels (2, 4, and 8 months) were considered for this study. For the WTI series data, time series of 2-day mode (DW1), 4-day mode (DW2) and 8-day mode (DW3), and approximate mode are presented in Figure 3.

For the WTI series, six input combinations based on previous log return of daily oil prices are evaluated to estimate current prices value. The input combinations evaluated in the study are (i) r_{t-1}, (ii) r_{t-1}, r_{t-2}, (iii) $r_{t-1}, r_{t-2}, r_{t-3}$, (iv) $r_{t-1}, r_{t-2}, r_{t-3}, r_{t-4}$, (v) $r_{t-1}, r_{t-2}, r_{t-3}, r_{t-4}, r_{t-5}$, and (vi) $r_{t-1}, r_{t-2}, r_{t-3}, r_{t-4}, r_{t-5}, r_{t-6}$. In all cases, the output is the log return of current oil prices, r_t.

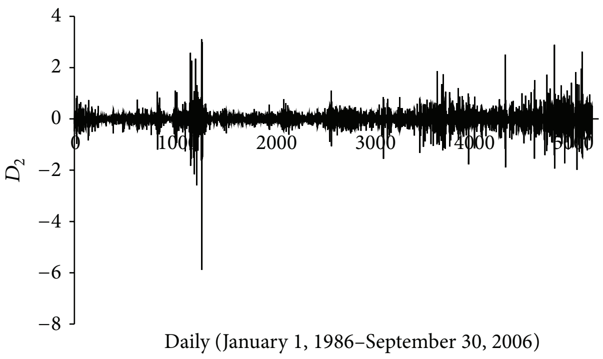
Each of DWs series plays distinct role in original time series and has different effects on the original prices oil series. The selection of dominant DWs as inputs of MLR model becomes important and effective on the output data and has positive effect excessively on model's ability. The model becomes exponentially more complex as the number of subtime series as input variables increases. Using a large number of input variables should be avoided to save time and calculation effort. Therefore, the effectiveness of new series obtained by PCA is used as input to the MLR model. The PCA approach helps us to reduce the number of original variables to a set of new variables. Generally, the objective of PCA is to identify a new set of variables such that each variable, called a principal component, is a linear combination of the original variables. The new set of variables accounts for 85%–90% of



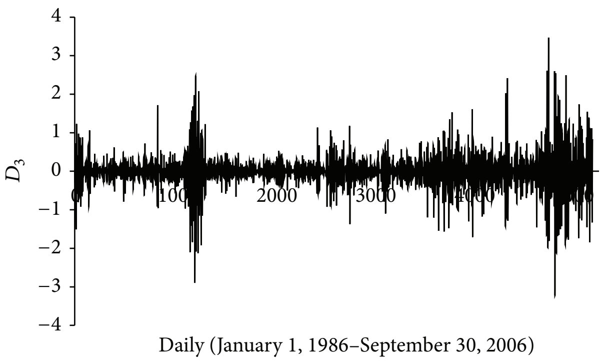
(a)



(b)

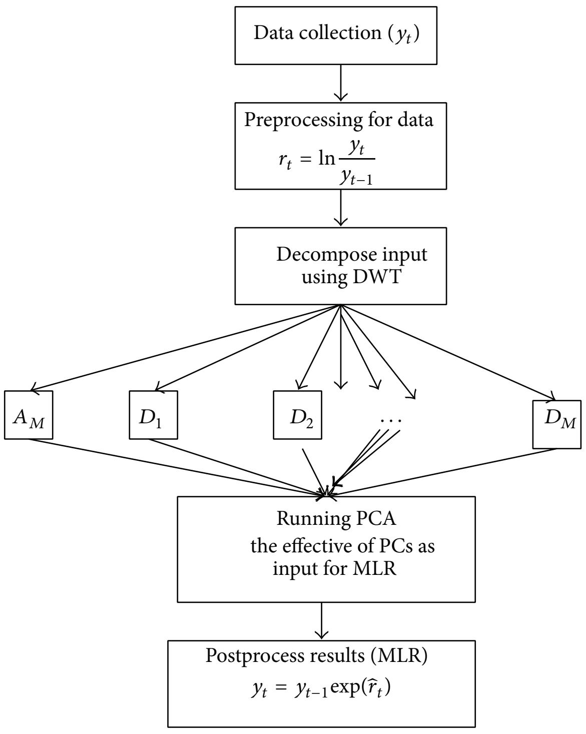


(c)



(d)

FIGURE 3: Decomposed wavelet subtime series components (Ds) of WTI crude oil price data.



Data collection (y_t)
Preprocessing for data $r_t = \ln \frac{y_t}{y_{t-1}}$
Decompose input using DWT
A_M
D_1
D_2
...
D_M
Running PCA the effective of PCs as input for MLR
Postprocess results (MLR) $y_t = y_{t-1}\exp(\hat{r}_t)$

FIGURE 4: The structure of the WMLR model.

TABLE 1: Eigen value and cumulative variance contribution rate of the 8 principal components.

PC	1	2	3	4	5	6	7	8
Eigen value	1.97	1.79	1.59	1.33	0.67	0.41	0.21	0.03
Cumulative Variance Rate	0.25	0.47	0.67	0.84	0.92	0.97	1.00	1.00

the total variation were considered as the number of new variables.

For example, taking two previous daily oil prices as a random variable. Every previous daily oil price time series are decomposed using DWT into three decomposition levels, respectively. Thus there were 8 subseries considered for the PCA analysis. The result of PCA analysis is shown in Table 1. Table 1 shows that the first four principle components can explain 84% variation of the data variation with the eigenvalues greater than 1 to be retained, in which all the 4 PCs were included in the MLR model. Thus the 8 original variables can be replaced by 4 new irrelevant variables. For training MLR, the PSO algorithm solving the recognition problem is implemented and the program code including wavelet toolbox was written in MATLAB language. The WMLR model structure developed in present study is shown in Figure 4.

The forecasting performances of the MLR and WMLR models in terms of the MAE, RMSE, and D_{stat} testing phase are compared and shown in Table 2. Table 2 shows MLR model; the M1 with 1 lag obtained the best MAE statistics of 0.6948 and the M6 with 6 lags obtained the best RMSE statistics of 0.9450, while the M1 with 5 lags obtained the best

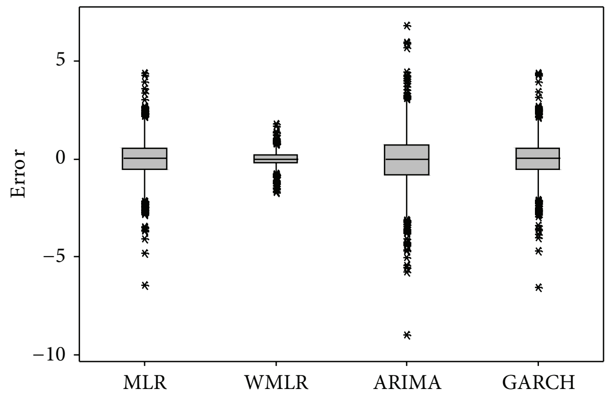


Figure 5: The errors of MLR, WMLR, ARIMA, and GARCH models for crude oil price forecasting.

Table 2: Forecasting performance indices of MLR and WLR.

Model Input	Lag	MLR			WMLR		
		MAE	RMSE	D_{stat}	MAE	RMSE	D_{stat}
M1	1	**0.6948**	0.9514	0.4788	0.6660	0.9001	0.5198
M2	1, 2	0.6972	0.9517	0.4781	0.6448	0.8842	0.5003
M3	1, 2, 3	0.6985	0.9545	0.4816	0.5345	0.7505	0.5797
M4	1, 2, 3, 4	0.6979	0.9550	0.4753	**0.4834**	**0.6572**	**0.6722**
M5	1, 2, 3, 4, 5	0.6976	0.9545	**0.4878**	0.5770	0.8046	0.5734
M6	1, 2, 3, 4, 5, 6	0.6969	**0.9450**	0.4850	0.5385	0.7389	0.6444

D_{stat} statistics of 0.4878. For WMLR, model M4 with 4 lags obtained the best MAE, RMSE, and D_{stat} statistics of 0.4834, 0.6572, and 0.6722, respectively. The equations of MLR with six input variables and WMLR with four input variables, respectively, are

$$\begin{aligned}\hat{y}_t &= -0.025y_{t-1} - 0.047y_{t-2} + 0.22y_{t-3} \\ &\quad - 0.082y_{t-4} - 0.060y_{t-5} + 0.0004y_{t-6}, \\ r_t &= 0.076z_1(t) - 0.221z_2(t) - 0.213z_3(t) \\ &\quad - 0.510z_4(t) + 0.416z_5(t) - 0.930z_6(t),\end{aligned} \tag{17}$$

where $z_i(t)$ are called principal components and $\hat{y}_t = y_{t-1}\exp(r_t)$.

For further analysis, the best performance of the LR, WMLR, ARIMA, and ARIMA-GARCH models was compared with the best results of ARIMA and forward neural network (FNN) studied by Yu et al. [1]. In Table 3, it shows that WMLR outperform MLR, ARIMA, GARCH, Yu' ARIMA and Yu' FNN models in terms of RMSE statistics. This results show that the new series (DWT) have significant extremely positive effect on MLR model results.

Figure 5 shows the Box-plot for the ARIMA, ARIMA-GARCH, MLR, and WMLR models for testing period. It can be seen that the errors of WMLR model are quite close to the zero. Overall, it can be concluded that the WMLR model provided more accurate forecasting results than the other models for crude oil forecasting.

Table 3: The RMSE and MAE comparisons for different models.

Model	RMSE	MAE
ARIMA (2, 1, 5)	1.3835	1.0207
GARCH (1, 1)	0.9513	0.6947
MLR	0.9450	0.6969
WMLR	**0.6572**	**0.4834**
Yu' ARIMA (Yu et al., [1])	2.0350	—
Yu' FNN (Yu et al., [1])	0.8410	—

4. Conclusions

The accuracy of the wavelet multiple linear regression (WMLR) technique in the forecasting daily crude oil has been investigated in this study. The PCA is used to choose the principle component scores of the selected inputs which were used as independent variables in the MLR model and the particle swarm optimization (PSO) is used to adopt the optimal parameters of the MLR model. The performance of the proposed WMLR model was compared to regular LR, ARIMA, and GARCH model for crude oil forecasting. Comparison results indicated that the WMLR model was substantially more accurate than the other models. The study concludes that the forecasting abilities of the MLR model are found to be improved when the wavelet transformation technique is adopted for the data preprocessing. The decomposed periodic components obtained from the DWT technique are found to be most effective in yielding accurate forecast when used as inputs in the MLR model. The accurate forecasting results indicate that WMLR model provides a superior alternative to other models and a potentially very useful new method for crude oil forecasting. The WMLR model presented in this study is a simple explicit mathematical formulation. The WMLR model is much simpler in contrast to ANN model and can be successfully used in modeling short-term crude oil price. In the present study, three resolution levels were employed for decomposing crude oil time series. If more resolution levels were used, the results from WMLR model may turn out better. This may be a subject of another study.

Conflict of Interests

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

Acknowledgments

The authors thankfully acknowledged the financial support that was afforded by Universiti Teknologi Malaysia under GUP Grant (VOT 06J13). Besides that, the authors would like to thank the Department of Irrigation and Drainage, Ministry of natural Resources and Environment, Malaysia, in helping us to provide the data.

References

[1] L. Yu, S. Wang, and K. K. Lai, "Forecasting crude oil price with an EMD-based neural network ensemble learning paradigm," *Energy Economics*, vol. 30, no. 5, pp. 2623–2635, 2008.

[2] H. Mohammadi and L. Su, "International evidence on crude oil price dynamics: applications of ARIMA-GARCH models," *Energy Economics*, vol. 32, no. 5, pp. 1001–1008, 2010.

[3] M. I. Ahmad, "Modelling and forecasting Oman Crude Oil Prices using Box-Jenkins techniques," *International Journal of Trade and Global Markets*, vol. 5, pp. 24–30, 2012.

[4] P. Agnolucci, "Volatility in crude oil futures: a comparison of the predictive ability of GARCH and implied volatility models," *Energy Economics*, vol. 31, no. 2, pp. 316–321, 2009.

[5] Y. Wei, Y. Wang, and D. Huang, "Forecasting crude oil market volatility: further evidence using GARCH-class models," *Energy Economics*, vol. 32, no. 6, pp. 1477–1484, 2010.

[6] L. Liu and J. Wan, "A Study of Shangai fuel oil futures price volatility based on high frequency data: long-range dependence, modeling and forecasting," *Economic Modelling*, vol. 29, pp. 2245–2253, 2012.

[7] A. Ghaffari and S. Zare, "A novel algorithm for prediction of crude oil price variation based on soft computing," *Energy Economics*, vol. 31, no. 4, pp. 531–536, 2009.

[8] M. A. Kaboudan, "Compumetric forecasting of crude oil prices," in *The Proceedings of IEEE Congress on Evolutionary Computation*, pp. 283–287.

[9] S. Mirmirani and H. C. Li, "A comparison of VAR and neural networks with genetic algorithm in forecasting price of oil," *Advances in Econometrics*, vol. 19, pp. 203–223, 2004.

[10] W. E. Shambora and R. Rossiter, "Are there exploitable inefficiencies in the futures market for oil?" *Energy Economics*, vol. 29, no. 1, pp. 18–27, 2007.

[11] L. Yu, S. Wang, and K. K. Lai, "A novel nonlinear ensemble forecasting model incorporating GLAR and ANN for foreign exchange rates," *Computers and Operations Research*, vol. 32, no. 10, pp. 2523–2541, 2005.

[12] W. Xie, L. Yu, S. Y. Xu, and S. Y. Wang, "A new method for crude oil price forecasting based on support vector machines," *Lecture Notes in Computer Science*, vol. 3994, pp. 441–451, 2006.

[13] R. Jammazi and C. Aloui, "Crude oil price forecasting: experimental evidence from wavelet decomposition and neural network modeling," *Energy Economics*, vol. 34, no. 3, pp. 828–841, 2012.

[14] W. Qunli, H. Ge, and C. Xiaodong, "Crude oil price forecasting with an improved model based on wavelet transform and RBF neural network," in *Proceedings of the International Forum on Information Technology and Applications (IFITA '09)*, pp. 231–234, May 2009.

[15] S. Yousefi, I. Weinreich, and D. Reinarz, "Wavelet-based prediction of oil prices," *Chaos, Solitons and Fractals*, vol. 25, no. 2, pp. 265–275, 2005.

[16] Y. Bao, X. Zhang, L. Yu, K. K. Lai, and S. Wang, "Hybridizing wavelet and least squares support vector machines for crude oil price forecasting," in *Proceedings of the 2nd International Workshop on Intelligent Finance*, Chengdu, China, 2007.

[17] J. Liu, Y. Bai, and B. Li, "A new approach to forecast crude oil price based on fuzzy neural network," in *Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD '07)*, pp. 273–277, August 2007.

[18] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, Holden-Day, San Francisco, Calif, USA, 1976.

[19] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[20] H. Bozdogan and J. A. Howe, "Misspecified multivariate regression models using the genetic algorithm and information complexity as the fitness function," *European Journal of Pure and Applied Mathematics*, vol. 5, no. 2, pp. 211–249, 2012.

[21] S. M. Chen and P. Y. Kao, "TAEIX forecasting based on fuzzy time series, partical swarm optimization techniques and support vector machines," *Information Sciences*, vol. 247, pp. 62–71, 2013.

[22] R. Alwee, S. M. Shamsuddin, and R. Sallehuddin, "Hybrid support vector regression and autoregressive integrated moving average models improved by particle swarm optimization for property crime rates forecasting with economic indicators," *The Scientific World Journal*, vol. 2013, Article ID 951475, 11 pages, 2013.

[23] 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.

[24] G. Tyagi and M. Pandit, "Combined heat and power dispatch using time varying acceleration coefficient particle swarm optimazation," *International Journal of Engineering and Innovative Technology*, vol. 1, no. 4, pp. 234–240, 2012.

[25] L. Liu, S. Yang, and D. Wang, "Particle swarm optimization with composite particles in dynamic environments," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 40, no. 6, pp. 1634–1648, 2010.

[26] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 120–127, IEEE Service Center, Piscataway, NJ, USA, April 2007.

[27] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[28] S. G. Mallat, “Theory for multiresolution signal decomposition: the wavelet representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[29] H. Çamdevýren, N. Demýr, A. Kanik, and S. Keskýn, “Use of principal component scores in multiple linear regression models for prediction of Chlorophyll-a in reservoirs,” *Ecological Modelling*, vol. 181, no. 4, pp. 581–589, 2005.

[30] I. T. Jolliffe, *Principal Component Analysis*, Springer, New York, NY, USA, 1986.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 153162, 12 pages
http://dx.doi.org/10.1155/2014/153162

Research Article

Optimization of Power Utilization in Multimobile Robot Foraging Behavior Inspired by Honeybees System

Faisul Arif Ahmad, Abd Rahman Ramli, Khairulmizam Samsudin, and Shaiful Jahari Hashim

Department of Computer and Communication Systems Engineering, Faculty of Engineering, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

Correspondence should be addressed to Faisul Arif Ahmad; faisul@upm.edu.my

Received 6 February 2014; Accepted 9 March 2014; Published 7 May 2014

Academic Editors: P. Agarwal, S. Balochian, V. Bhatnagar, and Y. Zhang

Copyright © 2014 Faisul Arif Ahmad 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.

Deploying large numbers of mobile robots which can interact with each other produces swarm intelligent behavior. However, mobile robots are normally running with finite energy resource, supplied from finite battery. The limitation of energy resource required human intervention for recharging the batteries. The sharing information among the mobile robots would be one of the potentials to overcome the limitation on previously recharging system. A new approach is proposed based on integrated intelligent system inspired by foraging of honeybees applied to multimobile robot scenario. This integrated approach caters for both working and foraging stages for known/unknown power station locations. Swarm mobile robot inspired by honeybee is simulated to explore and identify the power station for battery recharging. The mobile robots will share the location information of the power station with each other. The result showed that mobile robots consume less energy and less time when they are cooperating with each other for foraging process. The optimizing of foraging behavior would result in the mobile robots spending more time to do real work.

1. Introduction

Today, research on multimobile robots using large numbers with biologically inspired system is increasing [1–3]. Multimobile robots consisting of more than one mobile robot are used in different environments and different types of mobile robot. The aim of multimobile robot application is applied to environment that is difficult or high risk work for human, especially in hazardous area and disaster area such as search and rescue in collapsed building. Currently, mobile robots are widely used in hazardous environment such as investigation in dangerous environment and exploration of nonhuman living environment such as space exploration and deep sea exploration. One of the examples was in Fukushima Daiichi nuclear power plant. The nuclear plant has been hit by disaster of earthquake and tsunami in east coast of Japan in year 2011 and caused the nuclear disaster to the nearest area. Mobile robots were used to explore the area of disaster in semiautonomous system [4].

In early day, several mobile robots with big size are used [5–8]. Nowadays, small size of body and simple mobile robots are very popular in research and education [1, 8–12]. Multimobile robot systems inspired by biological system such as ants, termites, honeybees, and fishes are defined as a swarm robot. Intelligence algorithm for control system which is inspired by biological system is defined as swarm intelligent system [13]. Swarm intelligence has been applied to homogeneous robotic system [11, 14] which is simple robots. But in [9, 15, 16], they used a group of various types of mobile robots which is defined as heterogeneous swarm robotic system. Most of the research on swarm mobile robotic system is

(i) focusing on communication systems among mobile robots [11, 15, 16] in order to share information,

(ii) focusing on developed autonomous battery recharging [17–19] to have fully autonomous swarm robots without human intervention,

(iii) focusing on developed navigation system [3] in order to run mobile robot without loss,

(iv) focusing on developed management and optimization of energy system based on adaptation from biologically swarm behaviors [12, 19, 20],

(v) focusing on development of physical robot (small robot or macrorobot) [1, 11, 21] or simulated tools for swarm robotics [22–24].

Mobile robot is designed to be operated with finite energy resource, which is supplied by batteries. The energy of battery decreased with time due to robot processing, control system, and so forth. The battery power will run out if no action is taken to recharge it when it is running low. Researchers have developed an autonomous charging platform system for autonomous mobile robot systems. The first autonomous charging system on mobile robot is developed by Grey Walter in 1950, known as tortoise robot. Silverman et al. [18] had designed a docking platform for mobile robots recharging system. They defined threshold of battery voltage when robot needed to be recharged. A pan-tilt-zoom (PTZ) camera mounted on the mobile robot is used as a visual tool for searching orange colored piece of paper that is labeled above docking source. Cassinis et al. [25] developed a docking system for autonomous mobile robot charging system. Processing for docking operation system is implemented by creating a marker for reference by mobile robot during docking operation. The marker is identified by a vision system that is mounted on Pioneer 2DX robot. Ngo and Schiøler [26] developed a recharging system by exchanging the batteries. If a robot has low power battery during operation it will notify the coordinator robot (host) through the radio communication. The host will identify the nearest charger source or nearest robot that has high energy to the robot that requires energy. Then, the host will command the robot to go to nearest power station or command another robot that has high energy to go to the robot and exchange the battery. The message in communication is embedded with current position and status of the robot energy.

To operate mobile robot continuously, it needs to be recharged before its power resource is exhausted. The process of recharging battery needs to be fully autonomous; mobile robots need a capability of self-maintained or self-recharging battery system. In the previous paragraph, the researchers developed power station and function for autonomous recharging battery, but they did not identify how to handle the charging behavior with other behaviors such as robot task and robot interactions. Couture-Beil and Vaughan [27] applied the effect of charger location to evaluate system performance in limited environment. There were two environments that they examined which are location along robot's working path and nearby robot's working path. A minimum threshold of energy has been identified to drive the robot to the charging station. As the researcher limits the path environment of mobile robot with two ways of traveling, the time is increasing whenever numbers of mobile robots increased. Liu et al. [20] developed an autonomous ratio adjustment for several types of behaviors based on division labor from honeybees to maximize the net energy income to mobile robots. They used the assumption to identify the crowdedness of the foraging by assuming the frequent intersection of mobile robots. They did not study the relationship between number of power source and the number of intersection of mobile robot.

Several algorithms based on honeybee intelligent have been developed as optimization tools, such as artificial bee colony (ABC) [28] and honeybee mating optimization (HBMO) [29]. In the mobile robot application, both algorithms currently have been applied in the path-planning navigation [30, 31]. For example, in the ABC algorithm, four parameters have been divided, which are employed bee, onlooker bee, scout bee, and food source position. The division of bees in the environment will reduce the number of working robots in time. Based on that, the ABC algorithm is found not suitable to be applied to the mobile robots with working and foraging behavior that need to do task faster. Whereas in HBMO, the algorithm is based on finding the best result among the random execution of bee to find the optimal path. Another intelligent system from honeybee, which is based on foraging and working behavior, is more suitable to be applied to mobile robots having working and foraging power station as the main behavior.

In this paper, the work inspired by biological honeybees system is designed to optimize the working energy in the working behavior of mobile robots. In order to do that, the time utilization, power consumption, and traveling distance of the foraging behavior in the foraging area need to be decreased. Based on knowledge sharing and static threshold behavior, the system is applied to honeybees inspired environment and local communication.

A group of homogeneous mobile robots, known as AMiR [11], were used in simulation environment which is divided into two areas, which are home area and foraging area. This paper is divided to several sections. Section 2 will discuss the background of autonomous recharging battery in mobile robots. In Section 3, the methodology of the development which is a swarm robotic system based on foraging behavior of honeybees is discussed and then continues with the experimental method. The result and discussion will be written in Section 4 and the conclusion in Section 5, respectively.

2. Background

In biological system, a swarm of honeybees shows that swarm intelligence and self-organisation task are done without centralized control decision [32], where each honeybee is capable of receiving inputs, makes its own decision, and then executes the decision by itself. In a beehive, honeybees worker is categorized based on different stages of age. The stage of honeybees has been identify as the honeybees worked from the first day of their life which is cleaning blood cells, tending larvae, hive construction, guarding the hive and foraging food [32, 33] (see Table 1).

In biological honeybees, the information of the food source is shared with others through local communication. The information embedded with distance and direction of the food sources (I, II, III) is referring to beehive and the sun as shown in Figure 1. Two types of dances have been identified by von Frisch [34], which are waggle dance for longer distance and round dance for near distance.

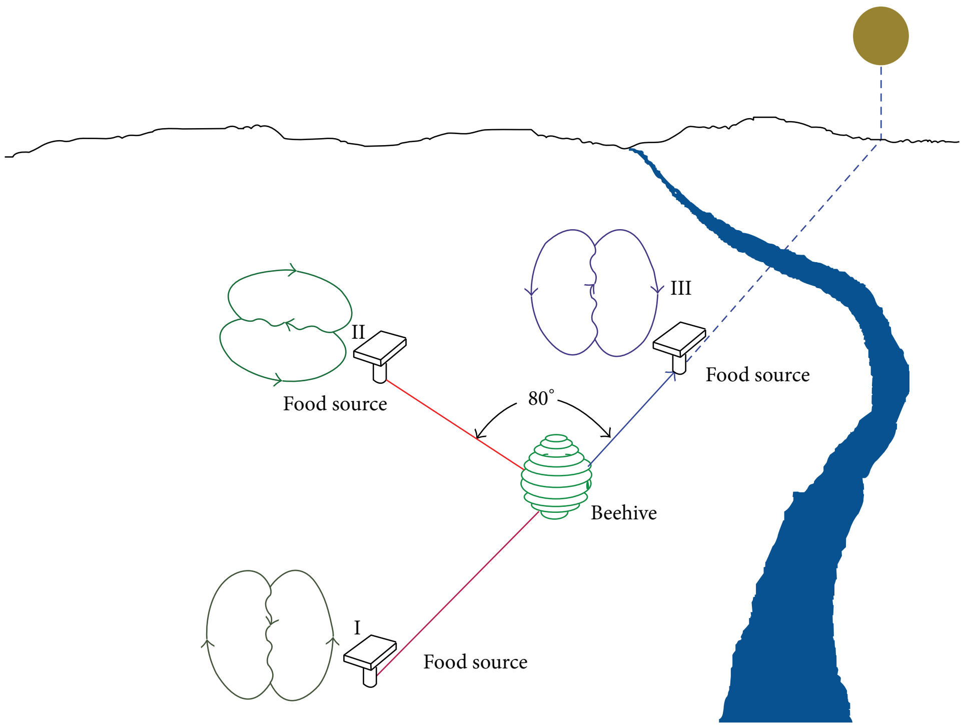


Figure 1: Honeybees communication through dances [32].

Table 1: Division of labor for honeybees [32, 33].

Period	Day	Stage
First	1-2	Cleaning blood cells
Second	3–9	Tending larvae
Third	10–16	Construction
Fourth	17–20	Guarding the hive
Fifth	21 and above	Foraging food

Table 2: Threshold energy of behavior mobile robot.

Remaining energy	Behaviors
$E_w \geqslant E_{\text{th}}$	Robot's worker
$E_f < E_{\text{th}}$	Robot's forager

Application of swarm intelligent system to mobile robots has been widely used with small or macromobile robots, such as swarm bot [14], autonomous miniature robot (AMiR) [11], EPUCK [21], JASMINE [35, 36], and ALICE [37]. Even though the size and the structure of mobile robot are small and simple, robotic system which is composed of large numbers can give better performance and robust compared to a complex and single mobile robot [38]. This type of mobile robot can be used in operation in small area such as a jet turbine and the complex engineering structure [10]. Another advantage is that the small size of robot can be easily and economically developed and replicated for being applied to bigger size of swarm robot. As the size is small, the processing power is also limited. But with large number of unit mobile robots that succeed by cooperation, it can overcome these disadvantages.

3. Methodology

Mobile robots behavior was divided in two different behaviors which were working mobile robot for doing task in home area and foraging mobile robot for forage power station in foraging area. This behavior is inspired by the honeybees foraging and working system. For working behavior, mobile robots are walking around the home area in random way until their battery decreased to the threshold of remaining energy by percentage. For foraging mobile robot, it forages for power station randomly without remembering the path (Table 2). The following subsection describes the simulation platform and environment used in this work.

3.1. Simulation Environment. Inspired by behaviors of honeybees from their working stage, 10 mobile robots of AMiR are designed on the simulation platform which is known as Player/Stage [22]. Figure 2 shows the experimental environment with two separated areas, a working area (left side) and a foraging area (right side). The experimental environment imitated the environment of honeybees which work in a beehive and forage for food (honey) outside their home.

The foods (normally come from flower) of honeybees are allocated randomly outside area of their beehive. In the foraging area of mobile robot, three power stations which are power station A, power station B, and power station C are randomly placed in different locations (see Figure 2). All

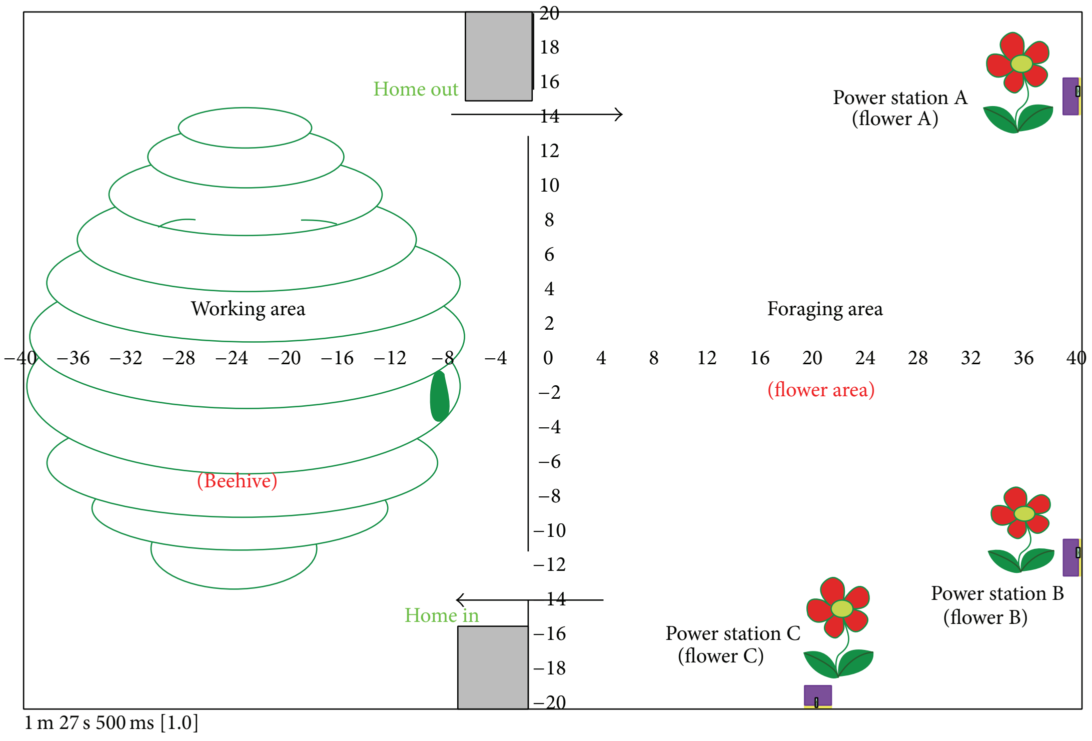


Figure 2: Environment for experiment of multimobile robots.

mobile robots have no information about the location of all power stations as also honeybees. Referring to Figure 2, working area (home area) and foraging area are separated by wall with doors as work area for biological honeybees is also separated from foraging area. The top door is provided for robot moving from working area to foraging area which is identified as home out (H(o)). The bottom door is used for mobile robot moving to working area from foraging area, identified as home in (H(i)). Mobile robots will be running randomly in home area and also searching for power station in foraging area when needed. Mobile robots started in the experiment with carrying energy in different capacities. Each of mobile robots has knowledge on exit and entry door for home area as honeybees have knowledge on their beehive's exit and entry. In the real environment, this could be implemented on office and house which normally have partition of rooms.

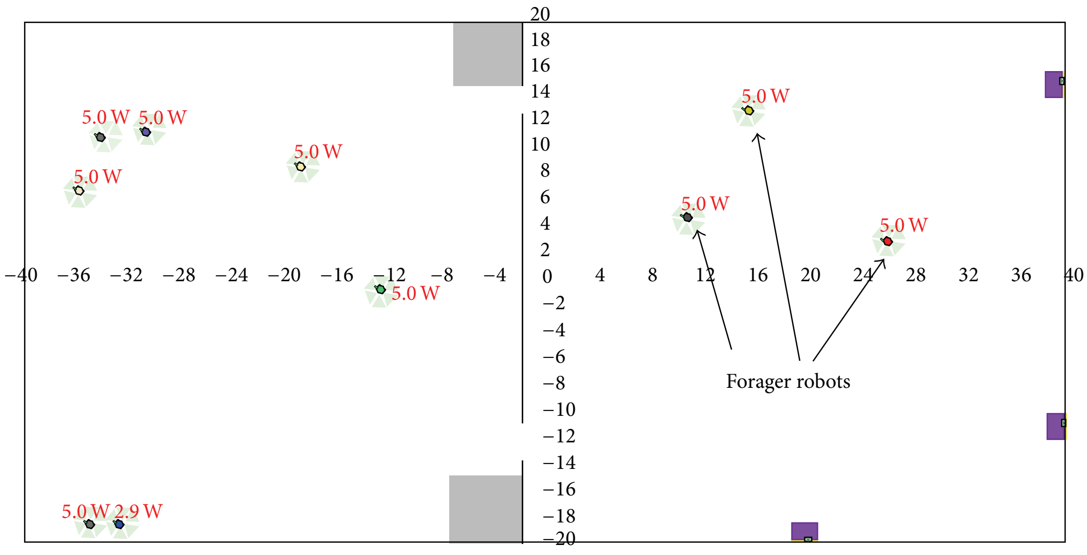
Figure 3 depicted foraging behaviors of multimobile robot in Player/Stage platform. Forager robot is moving randomly in foraging area (Figure 3(a)) looking for power station (A, B, or C). Whenever mobile robot finds a power station, it will identify the position of the location, remember it, and then charge until its battery is full (Figures 3(b) and 4).

After the full charging, the mobile robot returns home through the entry door. The entry location is known by mobile robot as honeybees know the entry area of the beehive. The mobile robot used coordinate information as its guidance in the movement, as honeybees used their movement based on the reference of the beehive and the sun location (Figure 5).

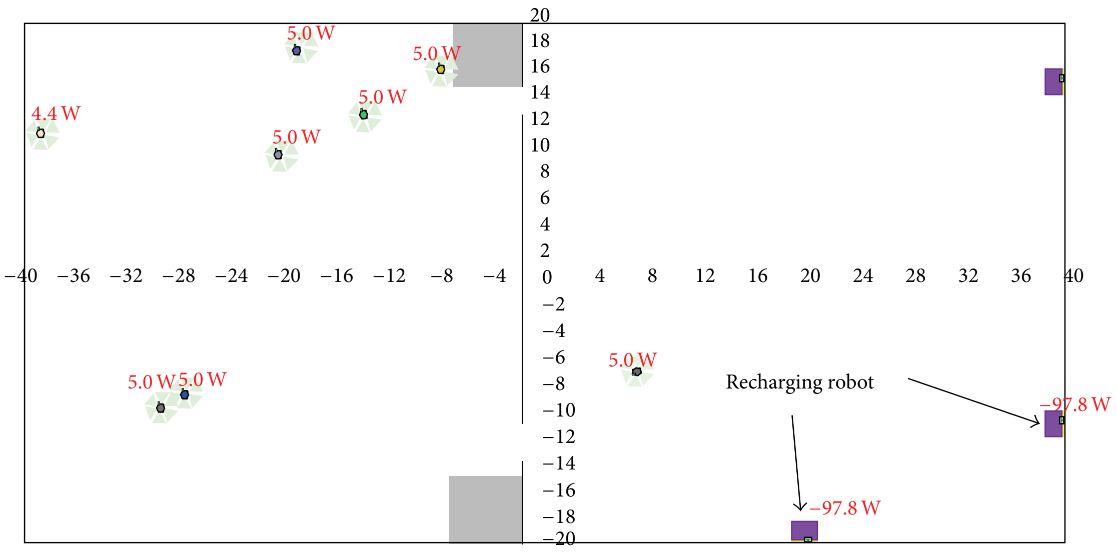
As in honeybees environment, the communication is done through the dance, which needs other nonknowledge honeybees face to face with the knowledge honeybees. Based on this, in mobile robots system, the movement of honeybees is converted to allocate the infrared (IR) surrounding the body of mobile robot. So the mobile robot does not need to move around during the communication, as location information is shared through infrared (IR) communication [39]. To adapt the IR communication in simulation platform, the UDP port is used as medium to transfer message between mobile robots as imitating an IR intercommunication in the real robot. Figure 6 illustrates 9 character strings of message format. Each message is determined by availability of message, robots' identifications (robot ID), and location of power station. Table 3 shows the detailed explanation of message format by each string. Example 1 is mobile Robot01 that detected power station by itself sending the message to mobile Robot02, while Example 2 is message that mobile Robot02 received information of location from Robot01 and lastly Example 3 shows initial message of Robot02 before it received the information from Robot01.

Figure 7 shows the flowchart of algorithm in behaviors of multimobile robots during the foraging power station for charging their battery, sending and receiving message process, randomly moving in home area for working task, and also docking and undocking function to the power station.

In this work, the mobile robot that has work and change from working to foraging behavior is known as finishing one working iteration. After that, when it finished the foraging, recharged the battery, and moved back to home



(a) Robot foraging for power station



(b) Mobile robot in the charging position

Figure 3: Some of mobile robots in stage of (a) foraging for power station (b) recharging at power station B and C.

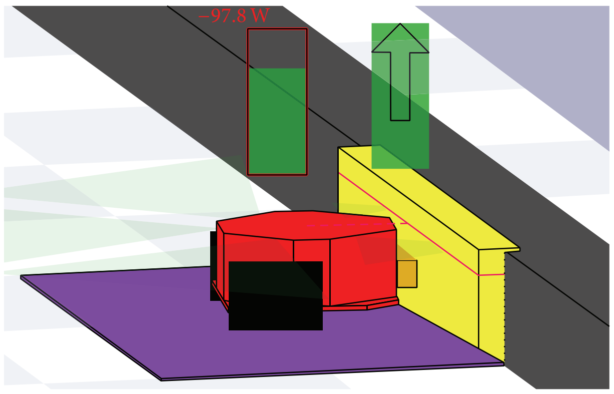


Figure 4: A closed view of mobile robots during charging at power station.

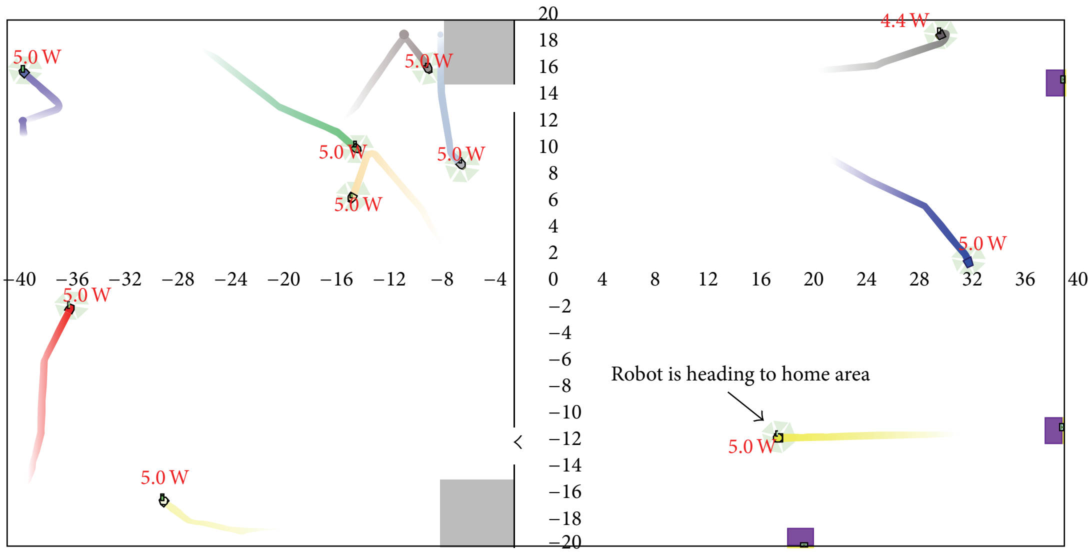


FIGURE 5: AMiR returning from charging station.

TABLE 3: Illustration of code used in the message format.

Number	Reference	Characters number	Example
1	Location information	**F**: knowledge got by itself **C**: nonknowledge **A**: knowledge got from other robots	**F**
2	Own Robot ID	Numbers	**1, 2, 3, . . .**
3	Separator mark	Robot ID separate mark	**$**
4	Opponent robot ID	Numbers	**1, 2, 3, . . .**
5	Separator mark	Robots ID separate mark	**$**
6	Coordinate-x	Floating number	**2.345**
7	Separator mark	Coordinate-x separate mark	**;**
8	Coordinate-y	Floating number	**2.345**
9	End mark of message	Mark of message end	**!**

Number of strings	1	2	3	4	5	6	7	8	9
Example 1	F	1	$	2	$	x_0	;	y_0	!
Example 2	A	2	$	1	$	x_0	;	y_0	!
Example 3	C	2	$	0	$	0	;	0	!

FIGURE 6: Message format in robots communication.

area, it is defined as finishing one foraging trip. Each of the working iterations or the foraging trips is measured by the performance metric.

3.2. Performance Metric. Three parameters are used to evaluate the foraging performance. The parameters are time consumption, power consumption, and traveling distance in foraging area. These three parameters are selected because

(i) time is determined by how long the mobile robot is foraging in the foraging area;

(ii) power is determined by how much power is utilized in the foraging area. The assumption is that the mobile robot that utilizes little power in the foraging area will utilize more power in the working area. This means that mobile robot can do more work based on the more power that has been decreased;

(iii) distance is determined by how long the distance can mobile robots take for their foraging. The short distance in the foraging behavior will result in the mobile robots consuming less time, less power consumption in the foraging area.

Foraging time, t, is measured as shown in (1) where t_{exit} is the time that mobile robot reaches H(o) and t_{charge} is time that mobile robots start to charge at the power station. The time unit is second (s). Consider

$$t_{\text{forage}} = t_{\text{charge}} - t_{\text{exit}}(0)\,. \tag{1}$$

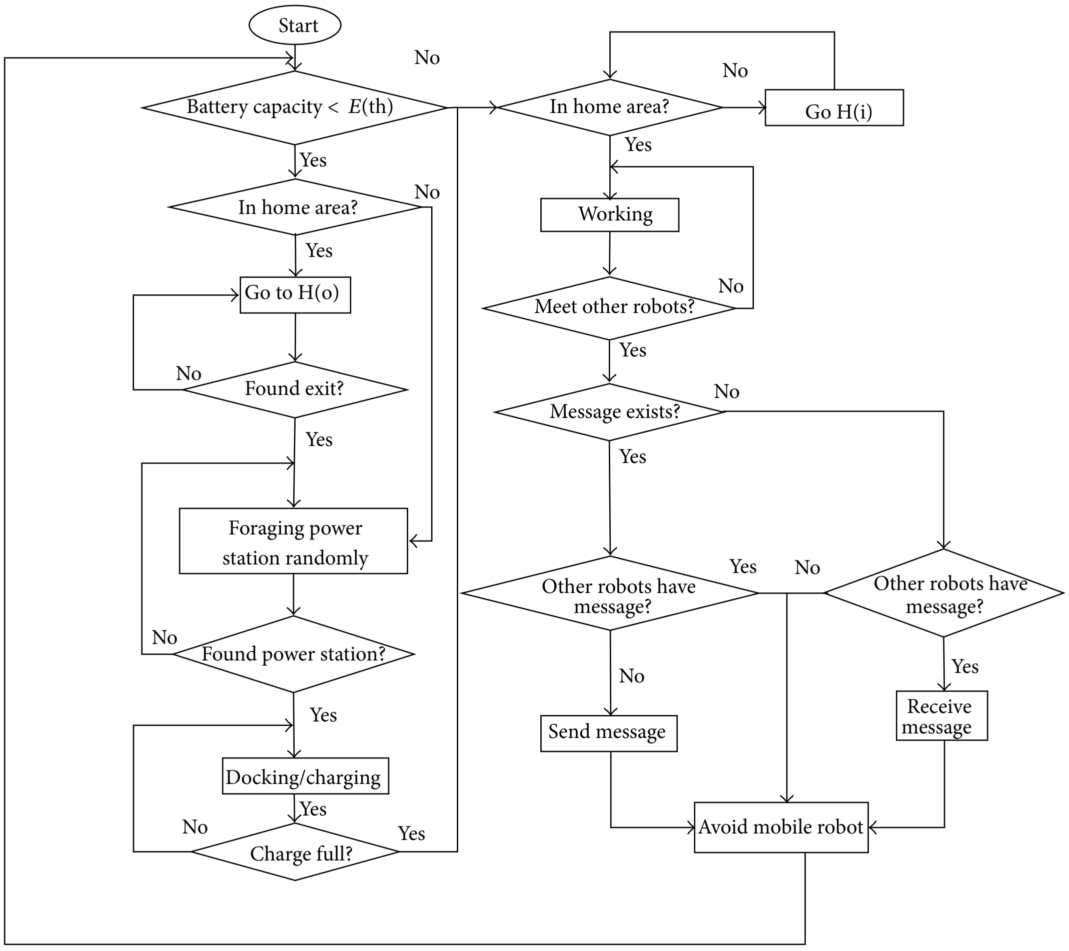


FIGURE 7: Algorithm of mobile robots behavior inspired by honeybees system.

Power consumption for foraging (P_{forage}) mobile robot is measured by (2), where P_{charge} is the remaining of power battery at the time that mobile robot reached the power station and P_{exit} is the remaining of power battery at H(o) position. Power is measured with unit Joule (J). Consider

$$P_{\text{forage}} = P_{\text{charge}} - P_{\text{exit}}. \tag{2}$$

Finally, the traveling distance by foraging robot n, D_n, is calculated using (3). Each of the iterations i of distance, d_i, is defined based on two points before the rotation angle is changed as equation (see Figure 8 for the illustration). Consider

$$D_n = \sum d_i,$$
$$d_i = \sqrt{(y_1 - y_0)^2 + (x_0 - x_1)^2}. \tag{3}$$

4. Result

The results of time consumption in the foraging behavior for power station are shown in Figure 9. Mobile robots with ID R01, R02, R08, and R09 consumed a lot of time to forage power station during the first foraging trip. After the first foraging trip, the mobile robots shared the locations information with other mobile robots such as R03 and R04. With this information, mobile robots will consumed less time to reach power station. The difference of the time consumption for mobile robot without information compared to mobile robot with information is quite large which is almost 6 times larger as shown in Figure 10(a).

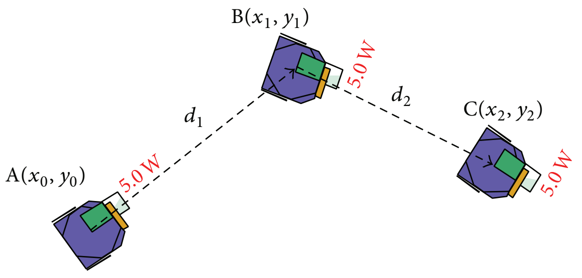


FIGURE 8: Traveling distance of mobile robot.

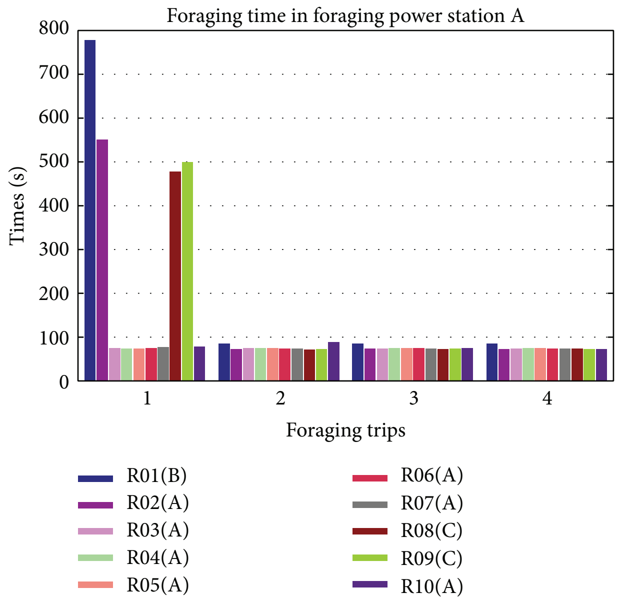


Figure 9: Time consumption in the foraging behavior of 10 mobile robots to power stations A, B, and C.

During the first foraging trip, R01, R02, R08, and R09 consumed a lot of time because the mobile robots need to forage power station randomly without information. They needed to forage until they found the station. The other robots with information did not consume a lot of time because they were heading directly to the power station without random forage. This difference is dynamic, because mobile robots forage in random way and without memorizing their path. This is proved by the result in the first trip of foraging mobile robots R01, R02, R08, and Robot09 which foraged to power stations A, B, and C (see Figure 9).

Figure 10(a) shows the comparison of the foraging time mobile robots with and without knowledge. R01, R09, and R10 are mobile robots that forage without information to the power stations A, C, and B, respectively. It is shown that the time consumption of mobile robot without knowledge is much longer than for the mobile robots that forage to power station with knowledge sharing. All the foraging was done in random way; R10 suddenly got high value in the third foraging trip compared to others. This is different with mobile robots that forage with the knowledge sharing environment, such as R03, R04, and R05, which forage with low value of time consumption beginning from the second foraging trip. When the time consumption is low in the foraging behavior, the time for mobile robots to do their task is increased. In that case, mobile robot should have much time to do task and work. These matters will make the task complete faster or earlier.

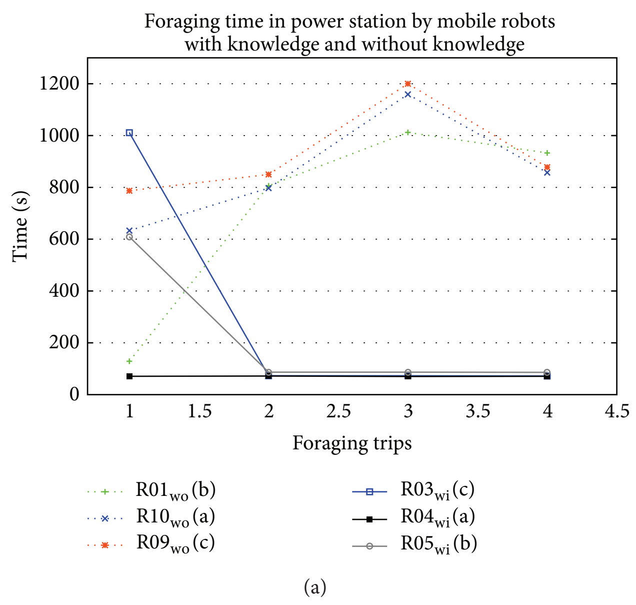
Figure 10(b) shows the average and standard deviation of foraging time by 10 mobile robots with sharing and without sharing knowledge. In environment of sharing knowledge, R01, R02, R08, and R09 show the high variance compared to other mobile robots. The mobile robots forage without knowledge in the first foraging trip and then memorized the information of the location power station. The average of time taken by mobile robots that forage for power station with information is less than 100 s, and their variance is very small compared to the mobile robots that forage without information at the first foraging trip. Meanwhile the mobile robots in environment without sharing knowledge show that the average of foraging is more than 600 s. The variance is also high compared to the mobile robots with sharing knowledge environment.

Figure 11 shows the power consumption of foraging power station by four foraging trips. The graph shows a lot of differences in foraging with knowledge and foraging without knowledge during the first foraging trip. As shown in the figure, R02 consumed high power to forage power station A during the first foraging trip because it does not have knowledge of the location. The mobile robot had foraged randomly until it found the power station and then memorized its location and in the next foraging trips the robot consumed less power. R02 not only memorized the location for itself but also shared the knowledge among other mobile robots such as R03, R04, and R05, so that other mobile robots do not need to forage and consumed a lot of power to do recharging.

Mobile robot with ID R01, R08 and R09 are consumed high power only at the first foraging trip, and then consumed low power for following trips with the knowledge. Among 10 mobile robots, in average R01, R02, R08, and R09 consumed high power in the foraging behavior. From the results of power consumption in foraging with knowledge sharing, three samples have been taken to compare with other three mobile robots that forage without knowledge sharing as shown in Figure 12(a). Mobile robots with ID R04 in environment with knowledge sharing show that they only consumed high power in the first trip, and other R03 and R05 consumed less power starting at the first foraging trip which is less than 1 kW, while for mobile robots that forage in environment without knowledge sharing, the power consumption is high which is more than 3 kW.

The average of power consumption in the foraging mobile robots with knowledge sharing and without knowledge sharing is shown in Figure 12(b). In without knowledge sharing case, the lowest value of power consumption is R03 and R09 with 2.9 kW. The maximum value is R04 with 4.6 kW. Robots with ID R01, R02, R08, and R09 consumed high power during foraging which is more than 650 W, while other mobile robots such as R03, R04, and R06 consumed less power which is less than 400 W. This happened because, during the first foraging trip, R01, R02, R08, and R09 need to forage in random way without the knowledge. But other mobile robots received the knowledge of the location power station before they changed to foraging behavior. In standard deviation the high variance of power consumption occurred on mobile robots that forage without knowledge in first foraging trip.

The result of traveling distance in foraging behavior for power station is depicted in Figure 13. Mobile robots with ID R01, R02, R08, and R09 traveled much longer than other mobile robots in the first trip as they are foraging without knowledge. But the other mobile robots such as R03 and R04



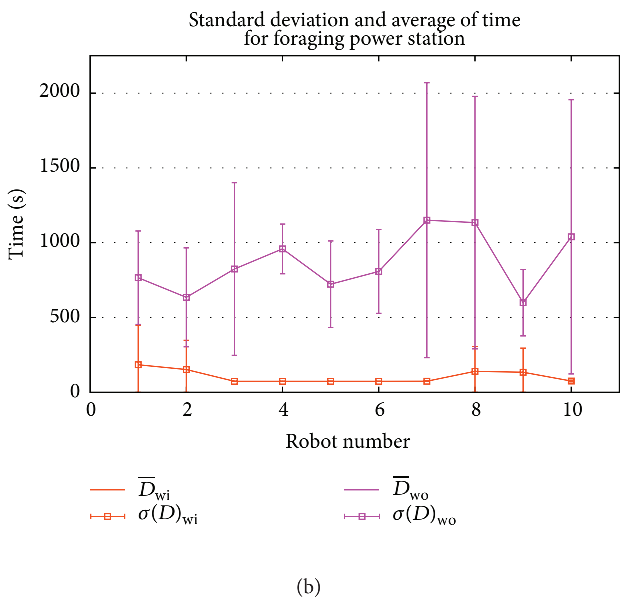


FIGURE 10: Analysis of foraging time consumption in environment with knowledge sharing and without knowledge sharing based on (a) 3 mobile robots in each environment and (b) average and standard deviation of times consumption for 10 mobile robots.

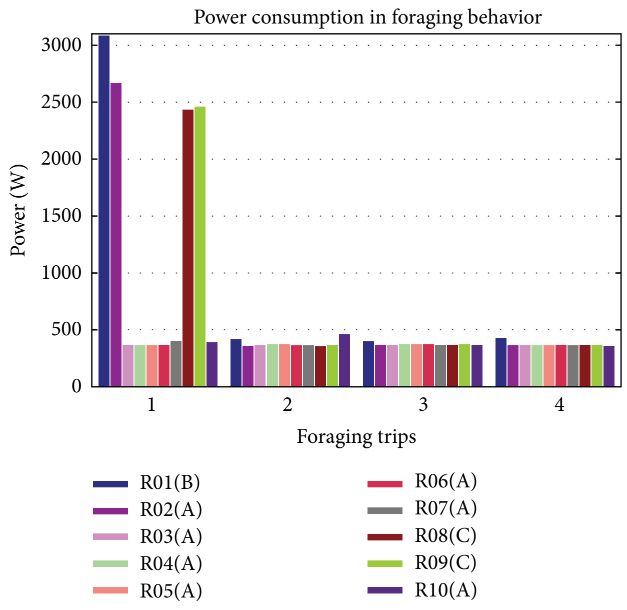


FIGURE 11: Foraging power consumption.

got the knowledge of the location on power station A from R02 and consumed shorter distance.

Figure 14(a) shows the analysis of traveling distance by choosing three mobile robots that forage with sharing knowledge and another three mobile robots without sharing knowledge. Mobile robots without knowledge sharing show long distance in traveling. Only R02 foraged in the first foraging trip with short distance but then foraged with long distance in the following foraging. Meanwhile, in environment with knowledge sharing, the short distance traveling is taken by mobile robots starting from second foraging trip. In the first foraging trip, the distance is far from the second trip, because the mobile robots did not have any information of the location power station. Whenever mobile robots got the information, they memorized the location and shared it to other mobile robots. Then for next trips, the mobile robots went to power station directly.

Average of foraging distance by mobile robots in the environment without knowledge sharing shows the high value compared to the mobile robot with knowledge sharing environment. The lowest foraging distance of mobile robot in environment without knowledge sharing is around 133 m compared to the mobile robots in knowledge sharing which is around 39 m. The difference in the foraging distance is around 4 times.

The average of foraging distance in knowledge sharing environment shows that the mobile robots that forage without knowledge for the first foraging trip are higher than the mobile robots that forage with knowledge (see Figure 14(a)). Mobile robots with ID R01, R02, R08, and R09 had to travel further compared to others that already got the knowledge and had foraged more than 30 m in average. Meanwhile mobile robots that already received the information from other robots had traveled less than 30 m.

Meanwhile, in the standard deviation, the mobile robots in environment without knowledge sharing show high variance compared to the mobile robots in environment with knowledge sharing. As a result of mobile robots with knowledge sharing, R01, R02, R08, and R09 have high variance

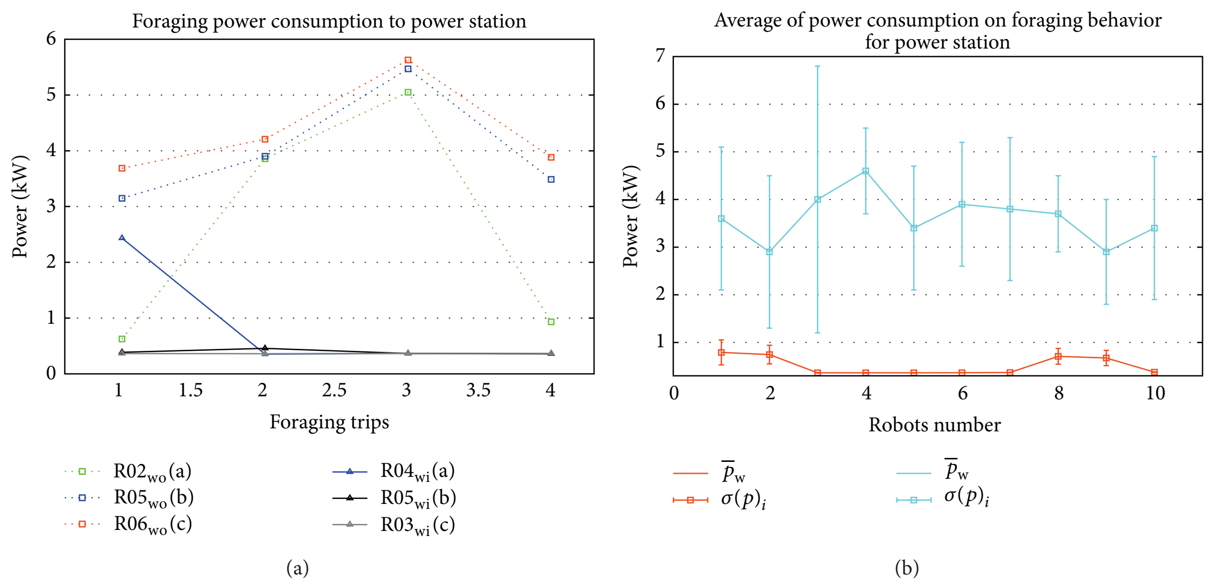


FIGURE 12: Analysis of foraging power consumption in environment with knowledge sharing and without knowledge sharing by (a) 3 selected mobile robots in each environment and (b) average and standard deviation of power consumption in foraging behavior.

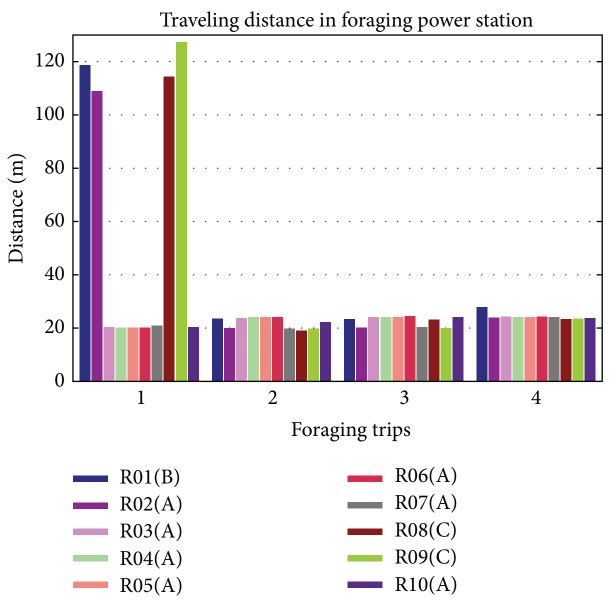


FIGURE 13: Traveling distance of 10 mobile robots on foraging behavior to power stations A, B, and C.

compared to others as they forage without knowledge during the first foraging trip, while others already had the knowledge of power station.

5. Discussion and Conclusion

Based on all results from the measuring parameters in the foraging behavior, decrease in the time consumption, power consumption, and traveling distance is shown. As for time consumption in the foraging area, the applied algorithm to mobile robot system had reduced almost 85% to 89% compared to the nonsharing environment. Meanwhile in the power consumption in the foraging area, the mobile robots with knowledge sharing have been reduced around 78% to 88% compared to mobile robots without knowledge sharing. And finally, the traveling distance in the foraging also reduced around 77% to 84%. The reducing of these parameters in the foraging mobile robots for recharging their battery will increase the time utilization, the power utilization, and the working distance of mobile robot in the working area. The longer the mobile robots in the working area with more power are being utilized and the longer the distance had been traveled to work, this would increase the work utilization in the mobile robot.

This work had identified the algorithm and environment inspired by honeybees system which is applied to multi-mobile robots by reducing the time consumption, power consumption, and traveling distance during the foraging behavior. Even though the threshold of the behavior of work and foraging is defined by a static threshold and the foraging behavior is random, the three parameters are still reduced with high value. This method has optimized the time consumption, power consumption, and traveling distance in mobile robots by reducing them in foraging area and increasing them in working area.

This system can be improved in the future by restructuring the methods of the system, for example, restructuring the threshold between working and foraging behavior. The threshold could be adaptive in order to maximize the energy in working area for mobile robots that already have the knowledge. Other methods are equipping the mobile

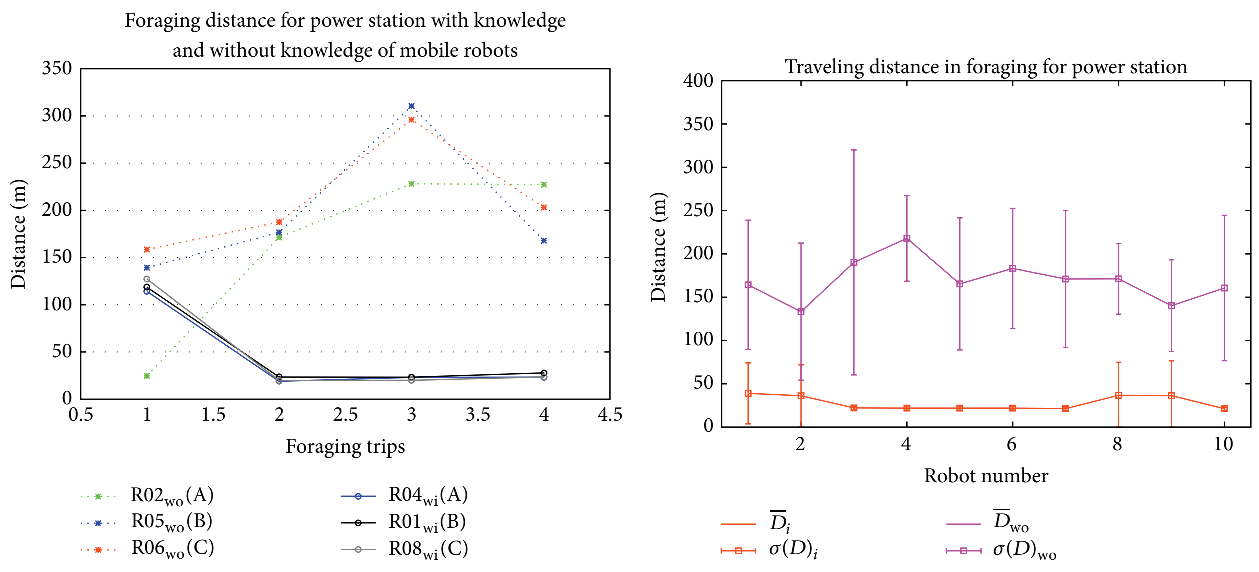


Figure 14: Traveling distance in foraging behavior with knowledge sharing and without knowledge sharing by (a) selected 3 mobile robots of each environment and (b) average and standard deviation of traveling distance in foraging for power station.

robots with the path identification, which means that mobile robots know the path that they already go through. Another matter that also needs to be concerned is the numbers of mobile robots in the environment. This could prevent the unwanted mobile robot that could increase the crowdedness environment.

Conflict of Interests

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

Acknowledgments

This study was supported in part by Research University Grant (RUGS) no. 05-01-12-1625RU. The authors acknowledge the comments of the reviewers.

References

[1] F. Mondada, G. C. Pettinaro, A. Guignard et al., "Swarm-bot: a new distributed robotic concept," *Autonomous Robots*, vol. 17, no. 2-3, pp. 193–221, 2004.

[2] T. Schmickl and K. Crailsheim, "Trophallaxis among swarm-robots: a biologically inspired strategy for swarm robotics," in *Proceedings of the 1st IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob '06)*, pp. 377–382, Department for Zoology Karl-Franzens-Universitat Graz, Pisa, Italy, February 2006.

[3] T. Schmickl and K. Crailsheim, "A navigation algorithm for swarm robotics inspired by slime mold aggregation," in *Swarm Robotics*, E. Sahin, W. Spears, and A. Winfield, Eds., vol. 4433 of *Lecture Notes in Computer Science*, pp. 1–13, Springer, Berlin, Germany, 2007.

[4] Daily Mail Reporter, "Nuke robots find conditions inside stricken Japanese power plant are 'sauna-like'," 2011, http://dailymail.co.uk/news/article-1378545/Fukushima-Daiichi-nuclear-power-plant-Robots-conditions-inside-sauna-like.html.

[5] M. J. Matarić, "Issues and approaches in the design of collective autonomous agents," *Robotics and Autonomous Systems*, vol. 16, no. 2-4, pp. 321–331, 1995.

[6] L. E. Parker, "ALLIANCE: an architecture for fault tolerant multirobot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[7] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith, "First results in the coordination of heterogeneous robots for large-scale assembly," in *Proceedings of the International Symposium on Experimental Robotics (ISER '00)*, 2000.

[8] N. Correll, *Coordination schemes for distributed boundary coverage with a swarm of miniature robots: synthesis, analysis and experimental validation [Ph.D. thesis]*, Ecole Polytechnique Federale de Lausanne (EPFL), 2007.

[9] M. Dorigo, D. Floreano, L. Maria Gambardella et al., "Swarmanoid: a novel concept for the study of heterogeneous robotic swarms," Tech. Rep. TR/IRIDIA/2011-014, Institut de Recherches Interdisciplinaires et de Developpements en Intelligence Artificielle (IRIDIA) Universite Libre de Bruxelle, Brussel, Belgium, 2011.

[10] N. Correll and A. Martinoli, "A challenging application in swarm robotics: the autonomous inspection of complex engineered structures," *Bulletin of the Swiss Society For Automatic Control*, no. 46, pp. 15–19, 2007.

[11] F. Arvin, K. Samsudin, and A. Rahman Ramli, "Development of a miniature robot for swarm robotic application," *International Journal of Computer and Electrical Engineering*, vol. 1, no. 4, pp. 1793–8163, 2009.

[12] A. Faisul Arif, A. R. Ramli, K. Samsudin, and S. J. Hashim, "Energy management in mobile robotics system based on biologically inspired honeybees behavior," in *Proceedings of*

the IEEE Conference on Computer Applications and Industrial Electronics (ICCAIE '11), pp. 32–35, December 2011.

[13] G. Beni, "From swarm intelligence to swarm robotics," in *Erol Sahin and William Spears*, S. Robotics, Ed., vol. 3342 of *Lecture Notes in Computer Science*, pp. 1–9., Springer, Berlin, Germany, 2005.

[14] F. Mondada, G. C. Pettinaro, I. Kwee et al., "SWARM-BOT: A swarm of autonomous mobile robots with self- assembling capabilities," in *Proceedings of the Workshop on Self-Organisation and Evolution of Social Behaviour*, C. K. Hemelrijk and E. Bonabeau, Eds., pp. 11–22, 2002.

[15] E. Yoshida, M. Yamamoto, T. Arai, J. Ota, and D. Kurabayashi, "Design method of local communication area in multiple mobile robot system," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2567–2572, May 1995.

[16] A. Birk and C. Condea, "Mobile robot communication without the drawbacks of wireless networking," in *RoboCup Robot Soccer World Cup IX*, I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, Eds., vol. 4020 of *Lecture Notes in Artificial Intelligence*, pp. 585–592, Springer, 2006.

[17] S. Oh, A. Zelinsky, and K. Taylor, "Autonomous battery recharging for indoor mobile robots," in *Proceedings of the Australian Conference on Robotics and Automation*, 2000.

[18] M. C. Silverman, D. Nies, B. Jung, and G. S. Sukhatme, "Staying alive: a docking station for autonomous robot recharging," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1050–1055, May 2002.

[19] T. Dung Ngo and H. Schioler, "A truly autonomous robotic system through self maintained energy," in *Proceedings of the International Symposium on Automation and Robotics in Construction*, 2006.

[20] W. L. Wenguo Liu, A. F. T. Winfield, J. S. Jin Sa, J. C. Jie Chen, and L. D. Lihua Dou, "Towards energy optimization: emergent task allocation in a swarm of foraging robots," *Adaptive Behavior*, vol. 15, no. 3, pp. 289–305, 2007.

[21] F. Mondada and M. Bonani, "E-puck—EPFL Education robot," 2011, http://www.e-puck.org.

[22] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: tools for multirobot and distributed sensor systems," in *Proceedings of the 11th International Conference on Advanced Robotics (ICAR '03)*, pp. 317–323, Coimbra, Portugal, June 2003.

[23] O. Michel, "Webotstm: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[24] R. T. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[25] R. Cassinis, F. Tampalini, P. Bartolini, and R. Fedrigotti, "Docking and charging system for autonomous mobile robots," Tech. Rep. R.T.2005-02-4, University of Brescia, Brescia, Italy, 2005.

[26] T. D. Ngo and H. Schiøler, "Sociable mobile robots through self-maintained energy," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2012–2017, October 2006.

[27] A. Couture-Beil and R. T. Vaughan, "Adaptive mobile charging stations for multi-robot systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '09)*, pp. 1363–1368, IEEE Press, Piscataway, NJ, USA, October 2009.

[28] 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.

[29] R. Ranjan Sahoo, P. Rakshit, M. T. Haidar, S. Swarnalipi, B. k. Balabantaray, and S. Mohapatra, "Navigational path planning of multi-robot using honey bee mating optimization algorithm (hbmo)," *International Journal of Computer Applications*, vol. 27, no. 11, pp. 1–8, 2011.

[30] Q. Ma and X. Lei, "Dynamic path planning of mobile robots based on abc algorithm," in *Artificial Intelligence and Computational Intelligence*, vol. 6320 of *Lecture Notes in Computer Science*, pp. 267–274, 2010.

[31] P. Bhattacharjee, P. Rakshit, I. Goswami, A. Konar, and A. K. Nagar, "Multi-robot path-planning using artificial bee colony optimization algorithm," in *Proceedings of the 3rd World Congress on Nature and Biologically Inspired Computing (NaBIC '11)*, pp. 219–224, October 2011.

[32] H. Yahya, *The Miracle of Honey-Bees*, GLOBAL PUBLISHING, 1st edition, 2007.

[33] S. E. Fahrbach and G. E. Robinson, "Behavioral development in the honey bee: toward the study of learning under natural conditions," *Learning & Memory*, vol. 2, no. 5, pp. 199–224, 1995.

[34] K. von Frisch, "Decoding the language of the bee," in *Nobel Lecture*, 1973.

[35] S. Kernbach, "Jasmine Open-source micro-robotic project," 2011, http://www.swarmrobot.org/index.html.

[36] S. Kernbach, R. Thenius, O. Kernbach, and T. Schmickl, "Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system," *SAGE Adaptive Behavior*, vol. 17, no. 3, pp. 237–259, 2009.

[37] G. Caprari and R. Siegwart, "Mobile micro-robots ready to use: alice," in *Proceedings of the IEEE IRS/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pp. 3845–3850, August 2005.

[38] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 947–951, 2001.

[39] F. Arvin, K. Samsudin, and A. R. Ramli, "A short-range infrared communication for swarm mobile robots," in *Proceedings of the International Conference on Signal Processing Systems (ICSPS '09)*, pp. 454–458, IEEE Computer Society, May 2009.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 596850, 11 pages
http://dx.doi.org/10.1155/2014/596850

Research Article

Hybrid Particle Swarm Optimization for Hybrid Flowshop Scheduling Problem with Maintenance Activities

Jun-qing Li,[1,2] Quan-ke Pan,[1] and Kun Mao[1]

[1] *State Key Laboratory of Synthetic Automation for Process Industries, Northeastern University, Shenyang 110819, China*
[2] *College of Computer Science, Liaocheng University, Liaocheng 252059, China*

Correspondence should be addressed to Jun-qing Li; p2p_jql@126.com

Received 21 February 2014; Accepted 30 March 2014; Published 29 April 2014

Academic Editors: P. Agarwal, V. Bhatnagar, and Y. Zhang

Copyright © 2014 Jun-qing 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.

A hybrid algorithm which combines particle swarm optimization (PSO) and iterated local search (ILS) is proposed for solving the hybrid flowshop scheduling (HFS) problem with preventive maintenance (PM) activities. In the proposed algorithm, different crossover operators and mutation operators are investigated. In addition, an efficient multiple insert mutation operator is developed for enhancing the searching ability of the algorithm. Furthermore, an ILS-based local search procedure is embedded in the algorithm to improve the exploitation ability of the proposed algorithm. The detailed experimental parameter for the canonical PSO is tuning. The proposed algorithm is tested on the variation of 77 Carlier and Néron's benchmark problems. Detailed comparisons with the present efficient algorithms, including hGA, ILS, PSO, and IG, verify the efficiency and effectiveness of the proposed algorithm.

1. Introduction

The hybrid flowshop scheduling (HFS) problem has been researched by more and more literatures during last decades. HFS is a typical version of the flowshop scheduling problem (FSP), which has been proved to be an NP-hard problem. Therefore, HFS is also an NP-hard problem and has been researched by more and more heuristics or metaheuristics [1–11]. In the most present literature about HFS, the common situation is assumed that all machines are available in the production horizon. However, for some critical factors, such as machine random breakdown and preventive maintenance (PM) activity, machines are not available during the whole production horizon. Allaoui and Artiba solved the HFS with maintenance constraints by using an integrating simulation and optimization [12]. Xie and Wang discussed the complexity and algorithms for two-stage flexible flowshop scheduling with availability constraints [13]. Allaoui and Artiba again considered the two-stage HFS with maintenance constraints [14]. Ruiz et al. considered scheduling and preventive maintenance in the flowshop sequencing problem [15]. Naderi et al. applied variable neighborhood structure (VNS) algorithm for solving flexible flow line problems with sequence dependent setup times and different preventive maintenance policies [16]. Berrichi et al. presented a biobjective optimization algorithm for joint production and maintenance scheduling in the parallel machine environments [17]. Luo et al. developed a genetic algorithm for solving two-stage HFS with blocking and machine availability [18]. Allaoui and Artiba investigated Johnson's algorithm for solving optimally or approximately flowshop scheduling problems with unavailability periods [19]. Jabbarizadeh et al. developed a hybrid algorithm for solving the hybrid flexible flowshops with sequence-dependent setup times and machine availability constraints [20]. Besbes et al. tackled hybrid flowshop problem with nonfixed availability constraints [21]. Ma et al. gave a survey of scheduling with deterministic machine availability constraints [22]. Luo et al. solved the HFS with batch-discrete processors and machine maintenance in time windows [23]. Safari and Sadjadi tackled the flowshop scheduling problem with condition-based maintenance constraint and machines breakdown through a hybrid method

[24]. Wang and Liu solved the two-stage hybrid flowshop scheduling with preventive maintenance using multiobjective tabu search method [25]. Rabiee et al. developed an intelligent hybrid metaheuristic for solving a case of no-wait two-stage flexible flowshop scheduling problem with unrelated parallel machines [26]. Allaoui and Artiba surveyed the maintenance constraints in HFS scheduling problems [27].

In this study, we developed a hybrid algorithm combining particle swarm optimization (PSO) and iterated local search (ILS) algorithms for solving the hybrid flowshop scheduling problems with PM activity. The rest of this paper is organized as follows: Section 2 briefly describes the problem. Next, the related algorithms are presented in Section 3. Section 4 reports the framework of the proposed algorithm. Section 5 illustrates the experimental results and compares them to the present performing algorithms from the literature to demonstrate the superiority of the proposed algorithm. Finally, the last section gives the concluding remarks and future research directions.

2. Problem Definition

In this study, we consider a hybrid flowshop scheduling problem in reality production system. The PM activity is considered in the considered HFS problems. Firstly, we give the following assumptions.

(1) Each machine can process only one operation at a time, while each operation can be processed by only one machine at a time.

(2) Preemption is not allowable; that is, each operation must be completed without interruption before its completion.

(3) At each stage, more than one machine from identical parallel machines can be selected for each operation.

(4) The processing time for each operation at each stage is determined.

Under the above assumption, the mathematical model for the problem is given as follows.

2.1. Variables

i: job index, $i = 1, 2, \dots, n$,

j: stage index, $j = 1, 2, \dots, s$,

k: machine index, $k = 1, 2, \dots, m$,

p_{ij}: the processing time of job i at stage j,

$s_{i,j}$: the starting time of job i at stage j,

$c_{i,j}$: the completion time of job i at stage j,

$\overline{s_{i,j}}$: the starting time of job i at stage j considering the PM activity,

$\overline{c_{i,j}}$: the completion time of job i at stage j considering the PM activity,

PM_s^k: the starting time point of the PM activity on M_k,

PM_e^k: the completion time point of the PM activity on M_k :

$$Z_{ijk} = \begin{cases} 1, & \text{if machine } k \text{ is selected to process job } i \\ & \text{at stage } j \\ 0, & \text{otherwise,} \end{cases}$$
$$Y_{ijk} = \begin{cases} 1, & \text{if } s_{ij} \in \left[PM_s^k, PM_e^k\right] \wedge Z_{ijk} = 1 \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

2.2. Problem Formula

$$f = \min\left\{\max_{1\le i\le n} c_{i,m}\right\} \quad (2)$$

s.t.

$$\overline{c_{i,j}} \ge \overline{s_{i,j}} + \overline{p_{i,j}} + Y_{ijk}\left(PM_e^k - PM_s^k\right), \quad (3)$$

$$\overline{s_{i+1,j}} \ge \overline{s_{i,j}} + \overline{p_{i,j}}, \quad (4)$$

$$\overline{s_{i+1,j+1}} \ge \overline{c_{i,j}}, \quad (5)$$

$$\sum_{1\le k\le m} Z_{ijk} = 1, \quad \forall i, j, \quad (6)$$

$$Z_{ijk} = \{0, 1\} \quad \forall i, j, k,$$
$$Y_{ij} = \{0, 1\} \quad \forall i, j, \quad (7)$$
$$0 \le w_1 \le 1.$$

In the mathematical model, the objective is given in formula (2). Constraint (3) guarantees that the PM time should be considered in processing any operation. In Constraint (4), the operation sequence is realized for the same job; that is, the following operation cannot be started until the completion of the predecessor operation of the same job. Constraint (5) shows that, on the same machine, the following operation must wait for the completion of the predecessor operation. Constraint (6) guarantees that each job can select only one available machine at each stage.

3. The Related Algorithm

In this study, we consider combining PSO and ILS to construct a hybrid algorithm for solving the HFS with PM activity. The following is to illustrate the literature review of the two related algorithms.

3.1. ILS Algorithm. Iterated local search (ILS), firstly proposed by Stützle [28], is a metaheuristic to increase the ability to jump out of the local optima for the canonical local search methods. It has attracted much attention of researchers for its simplicity, effectiveness, and efficiency, and it has been applied successfully to traveling salesman problem, flowshop scheduling problem, job shop scheduling problem, and vehicle scheduling problem, [28–31] during recent years. The main frame of the canonical ILS is as follows.

Step 1. Generate an initial solution x; let $x' = x$ and $x^* = x$.

Step 2. Generate a certain number of neighboring solutions around the given solution x', find the best neighboring solution x'', and update the best solution found so far.

Step 3. Let $x = \text{Accept}(x'', x)$.

Step 4. If the stop condition is not satisfied, generated $x' = \text{perturb}(x)$, go back to Step 2; otherwise, stop the algorithm.

3.2. Particle Swarm Optimization. In 1995, mimicking the flying behavior of a swarm of birds, a novel optimization algorithm named particle swarm optimization (PSO) was developed by Kennedy and Eberhart, which has been verified efficient for solving both continuous and discrete optimization problems [32]. During recent years, many researchers have applied PSO for solving lots of optimization problems [33–43].

The flowchart of the canonical PSO is given as follows.

Step 1. Set the system parameters, such as the initial population size, the possibility (p_l) for learning from local best, and the possibility (p_g) for learning from the best solution found so far.

Step 2. Generate the initial population of particles.

Step 3. Store each particle into a vector named local best, where each solution corresponds to the local best of the corresponding particle. Memorize the best solution found so far.

Step 4. For each particle, perform the following steps until the stop condition is satisfied.

Step 5. Randomly generate a number r_1 between 0 and 1, if r_1 is less than p_l, and then perform the learning process from the local best of the current particle.

Step 6. Randomly generate a number r_1 between 0 and 1, if r_1 is less than p_g, and then perform the learning process from the global best of the current particle.

Step 7. Record the local best for each particle and the global best found so far.

Step 8. Learn by itself.

Step 9. Go back to Step 4.

4. Framework of the Proposed Algorithm

4.1. Solution Representation. For solving the HFS scheduling problems with PM activity, we use the permutation representation mechanism. Give a HFS scheduling problem n jobs, s stages, and m machines; each solution is represented by a vector of integer values, where each integer value represents a job number. Therefore, the length of the solution equals the number of jobs. For example, for a HFS problem with ten jobs and three stages, Figure 1 gives one solution representation, where the scheduling sequence is $J_2, J_3, \ldots$, and J_7.

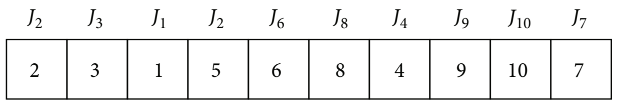

J_2 J_3 J_1 J_2 J_6 J_8 J_4 J_9 J_{10} J_7
2 3 1 5 6 8 4 9 10 7

Figure 1: Solution representation.

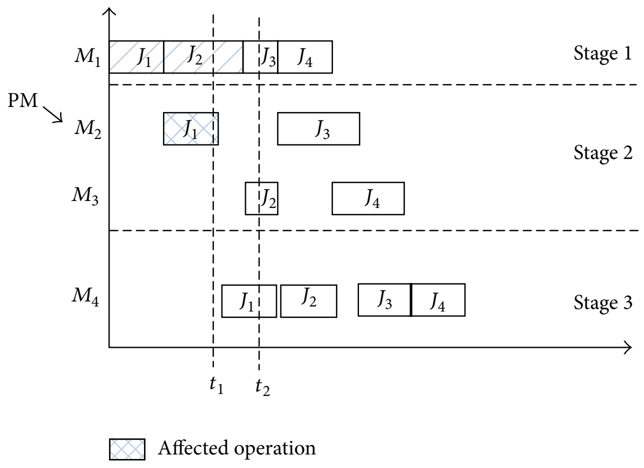


Figure 2: Situation 1 of PM activity.

The sequence in Figure 1 is only for the first stage; that is, at the first stage, each job is scheduled according to the above sequence, while for the following stages, the decoding mechanism is given as follows.

4.2. Decoding without Disruption. It can be seen from the solu-tion representation that the machine selection is not included in the solution representation. The decoding for the above solution representation is given as follows.

Step 1. For the first stage, each job is scheduled according to their sequence in the solution representation. In Figure 1 the first job to be scheduled is J_2 and the last one is J_7. Each job selects the first available machine.

Step 2. In the following stages, each job is to be scheduled just after its completion of the previous stage, and select the first available machine from the candidate machines.

4.3. Decoding with PM Activity. When considering the PM activity, that is, at time t, there is a PM activity occurring on a given machine M_k. Then two situations we should consider, that is, the first is that when an operation is just being processed on M_k when the disruption event occurs. The second situation is that the affected machine M_k is idle and no operation is affected by the PM activity.

(1) *Situation 1.* For the first situation, an operation is affected by the PM activity. Figure 2 gives the example chart for the situation. From Figure 2, we can see that, at time point t_1, the machine M_2 shows a PM activity. It will restart its work at

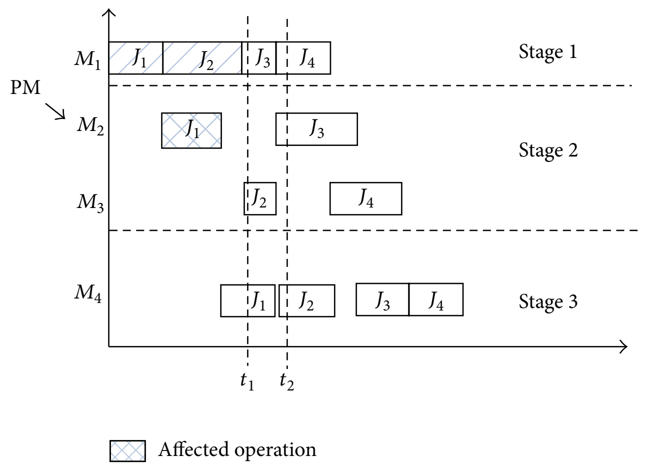


FIGURE 3: Situation 2 of PM activity.

time point t_2. However, before the PM activity of the machine, the operation J_1 has started its work and cannot complete its work at time point t_1. In this situation, we have to do the following works for different realistic production systems.

(i) When an operation is being processed and the processing machine needs to be maintenanced, we have to drop the affected operation and all its following operations. This is appliable for some certain realistic production system, such as steelmaking-casting system. Because of temperature restriction, an operation cannot wait for the restart work of the machine and has to be erased from the system because of its temperature loss. For example, for iron body, when its temperature decreases, its component structure will be destroyed.

(ii) In another situation, the affected operation will keep its previous work and wait for the restart of the affected machine. When the affected machine is available, the affected operation can restart its work and continue the following work.

(2) *Situation 2*. For the second situation, no working operation is affected by the PM activity. In this situation, we should consider whether there is any operation which is allocated to the affected machine during the PM activity. That is, if an operation is scheduled to be processed on the affected machine before its restart, then we should reconsider the assignment rule, which is given as follows.

(i) If an operation is scheduled to be processed on the affected machine, then the start time of the operation is located between the start and end time point of the PM activity. At that situation, we should assign a new machine for the affected operation if there is another available machine for the affected operation. For example, in Figure 3, the start time of the job J_3 is between the start and end time of the PM event on M_2. When the PM event occurs on the machine, we should assign another machine for J_3; here, we can select M_3 for processing J_3.

(ii) Another situation is that we cannot select another machine for the affected operation, because of the instability of the system. At that situation, we can only choose to keep the assignment machine for the affected operation and start its work after the availability of the affected machine.

4.4. Initialization Heuristic. In the initialization phase, we presented two heuristics, which are presented as follows.

(1) *The First Initial Heuristic.* The first initial heuristic is very simple and easy to implement, which is named INT-I with the following steps.

Step 1. Perform the following step for P_s times.

Step 2. Randomly generate a particle.

Step 3. Evaluate the new-generated particle and insert it into the current population.

(2) *The Second Initial Heuristic.* The second initial heuristic is named INT-II, which is given as follows.

Step 1. Generate a particle using the NEH approach [44] and insert it into the initial population.

Step 2. Perform the following step for $P_s - 1$ times.

Step 3. Randomly generate a particle and evaluate the new-generated particle.

Step 4. If the new-generated particle is not equal with any individual in the current population, then insert it into the initial population; otherwise, ignore it.

4.5. Discrete PSO Process. Each particle in the current population updates its status through the following three procedures: (1) learning through its history status, (2) learning through its local best, and (3) learning through the global best found so far.

Similar to [34], the discrete version of PSO is realized as follows.

(i) For the process of learning through its history status, we embed the mutation operator in the PSO algorithm. The mutation operators include swap, insert, multiple swap [34], and multiple insert. The multiple insert operator is developed firstly in this study. The detailed steps are as follows. Firstly, randomly produce a position r_1 range at $[2, l_{\text{en}} - 1]$, c, where l_{en} represents the length of the solution. Secondly, insert the element in the position $(r_1 - 1)$ to the position at $(r_1 + 1)$. Thirdly, evaluate the new-generated solution and replace the current solution if a better individual is found.

(ii) For the process of learning through its local best and learning through the global best, apply the crossover operator between the two selected solutions.

The detailed implementation of the crossover operators is discussed in the following section.

4.6. Crossover Operators. In [45], the authors verified many crossover operators for the regular flowshop (PMX or partially mapped crossover, OP or one point order crossover, TP or two-point order crossover, OX or order crossover, UOB or uniform order based, and several others). The results showed that the offspring generated after crossover tended to be worse than their progenitors on many occasions. In this study, we tested the following crossover operators in HFS with PM environments:

(i) PMX or partially mapped crossover;

(ii) OP or one point order crossover;

(iii) TP or two-point order crossover;

(iv) PTL crossover [34].

4.7. ILS-Based Local Search. To further improve the searching ability of the proposed algorithm, we apply the ILS-based local search for the best solution found so far in each iteration. That is, after the three learning processes discussed in the above section, the ILS-based local search will be applied for the best solution for enhanced searching. The detailed steps of the ILS-based local search are given as follows.

Step 1. For the best solution, perform the following steps until the stop condition is satisfied.

Step 2. Destruction phase: randomly generate aposition in the current solution. Delete the corresponding element from the current solution.

Step 3. Construction phase: for the deleted element, perform the following steps.

Step 3.1. For each candidate position in the current solution, insert the deleted element and evaluate the partial solution.

Step 3.2. Select the best position for the deleted element and insert it into the best position.

4.8. Framework of the Proposed Algorithm. In this study, we proposed a hybrid algorithm for solving the HFS problem with PM activity. In the decoding procedure, we select the following rules to decode each solution; in Situation 1, we choose to keep the work of the affected operation and continue its work after the affected machine is available. In Situation 2, we choose to assign another machine for the affected operation.

The flowchart of the proposed algorithm is given as follows.

Step 1. Set the system parameters.

Step 2. Produce the initial population of particles.

Step 3. Evaluate each particle and record the best solution found so far.

Step 4. If the stop condition is satisfied, stop the algorithm. Otherwise, perform the following steps.

Step 5. Perform learning phase.

Step 5.1. Perform the procedure of learning by itself.

Step 5.2. Perform the procedure of learning through its local best.

Step 5.3. Perform the procedure of learning through the global best.

Step 6. ILS-based local search phase: for the best solution found so far, perform the ILS-based local search procedure.

Step 7. Go back to Step 4.

5. Numerical Analysis

The proposed algorithm is coded in C++, on DELL i7 CPU with 16 GB memory. For each instance, we conduct 20 independently runs, and the best, worst, and average values are collected for comparisons.

5.1. Experimental Data. The proposed PSO-ILS algorithm was tested using the variation of the benchmark problems provided by Carlier and Néron [46]. There are 77 instances in Carlier and Néron's benchmark problems, which range from 10 jobs and 5 stages to 15 jobs and 10 stages. Each instance is represented by a three-number file name. The three numbers are number of jobs, number of stages, and problem structure index, which can be referred in [46]. For simplicity, the variations of the 77 benchmark problems are set with the same name. The variation implementation is implemented as follows.

(i) For each instance, run the proposed algorithm without considering any PM activity and get the baseline result.

(ii) In each baseline result, at each stage, randomly select a time point t at which a machine (hereafter called m_k) is working.

(iii) Select the working machine (m_k) and generate a random PM activity duration d_b.

(iv) Record the PM activity data, including the PM time window $[t, t + d_b]$, and the affected machine m_k.

5.2. Parameter Tuning for PSO. In the canonical PSO algorithm, the parameters are as follows:

(i) population size: P_s;

(ii) learning probability from the local best: c_1;

(iii) learning probability from the global best: c_2;

(iv) learning probability by itself: p_m;

(v) crossover operator type;

(vi) mutation operator type.

Table 1: Crossover type.

Crossover type	Description
CT-I	OP
CT-II	TP
CT-III	PMX
CT-IV	SJ2OX
CT-V	PTL

For each instance, we memorized the best solution found by all the compared algorithms and calculated the relative percentage deviation over the best solution for each compared algorithm, which is computed as follows:

$$\text{RPD}_i = \frac{\text{Comp}_i^k - \text{Best}_i}{\text{Best}_i} \times 100, \quad (8)$$

where Comp_i^k is the optimal solution found by the kth compared algorithm, while Best_i is the best solution found by all the compared algorithms. In the comparison results, we just calculated the average relative percentage deviation ($\overline{\text{RPD}}$) for each instance.

5.2.1. Crossover Type. To test the impact of different crossover operators, we implemented five kinds of crossover operators, that is, one-point crossover (OP), two-point crossover (TP), partially mapped crossover (PMX), similar job 2-point crossover (SJ2OX), and PTL crossover operator [34]. The description of the given crossover operators is given in Table 1. The comparisons results of different crossover types are given in Table 2. In Table 2, the instance name is given in the first column, while the following five columns report the $\overline{\text{RPD}}$ values for the five compared algorithms. From the results we can see that (1) the algorithm with PTL crossover operator gets better values for 75 out of 77 instances, except for the two instances, that is, Case 13 and Case 22; (2) for solving the given 77 instances with PM activity, in average, the algorithm with PTL crossover operator obtains a relative better result, which is obviously better than the other four compared algorithms. The following algorithms are SJ2OX, TP, PMX, and OP, respectively.

5.2.2. Crossover Probability. The crossover probability for learning from the local best (c_1) and the learning probability from the global best (c_2) are critical for the algorithm. In order to test different learning probabilities, we test five kinds of probabilities, which are given in Table 3. The comparison results for different learning probability are given in Table 4. It can be seen from Table 4 that CP-I is the best among the five compared algorithms. That is, the two crossover probabilities c_1 and c_2 are set to 0.2 and 0.2, respectively.

5.2.3. Mutation Type. To test the impact of different mutation operators, we implemented four kinds of mutation operators, that is, the swap, insert, multiple swap, and multiple insert operators, which are given in Table 5. Table 6 gives the

Table 2: Comparisons of different crossover types.

Case	$\overline{\text{RPD}}$				
	CT-I	CT-II	CT-III	CT-IV	CT-V
j10c5a2	0.00	0.00	0.00	0.00	0.00
j10c5a3	0.00	0.00	0.00	0.00	0.00
j10c5a4	0.00	0.00	0.00	0.00	0.00
j10c5a5	0.00	0.00	0.00	0.00	0.00
j10c5a6	0.00	0.00	0.00	0.00	0.00
j10c5b1	0.00	0.00	0.00	0.00	0.00
j10c5b2	0.00	0.00	0.00	0.00	0.00
j10c5b3	0.00	0.00	0.00	0.00	0.00
j10c5b4	0.00	0.00	0.00	0.00	0.00
j10c5b5	0.00	0.00	0.00	0.00	0.00
j10c5b6	0.00	0.00	0.00	0.00	0.00
j10c5c1	0.00	0.00	0.00	0.00	0.00
j10c5c2	0.00	1.35	1.35	1.35	1.35
j10c5c3	0.00	0.00	0.00	0.00	0.00
j10c5c4	1.52	0.00	0.00	0.00	0.00
j10c5c5	0.00	0.00	0.00	0.00	0.00
j10c5c6	0.00	0.00	0.00	0.00	0.00
j10c5d1	0.00	0.00	0.00	0.00	0.00
j10c5d2	0.00	0.00	0.00	0.00	0.00
j10c5d3	0.00	0.00	0.00	0.00	0.00
j10c5d4	0.00	0.00	0.00	0.00	0.00
j10c5d5	0.00	1.52	3.03	1.52	1.52
j10c5d6	0.00	0.00	0.00	0.00	0.00
j10c10a1	0.00	0.00	0.00	0.00	0.00
j10c10a2	0.00	0.00	0.00	0.00	0.00
j10c10a3	0.00	0.00	0.00	0.00	0.00
j10c10a4	0.00	0.00	0.00	0.00	0.00
j10c10a5	0.00	0.00	0.00	0.00	0.00
j10c10a6	0.00	0.00	0.00	0.00	0.00
j10c10b1	0.00	0.00	0.00	0.00	0.00
j10c10b2	0.00	0.00	0.00	0.00	0.00
j10c10b3	0.00	0.00	0.00	0.00	0.00
j10c10b4	0.00	0.00	0.00	0.00	0.00
j10c10b5	0.00	0.00	0.00	0.00	0.00
j10c10b6	0.00	0.00	0.00	0.00	0.00
j10c10c1	0.00	0.00	0.00	0.00	0.00
j10c10c2	0.00	0.00	0.00	0.00	0.00
j10c10c3	0.86	0.00	0.00	0.00	0.00
j10c10c4	0.00	0.00	0.00	0.00	0.00
j10c10c5	0.00	0.00	0.00	0.00	0.00
j10c10c6	0.00	0.00	0.00	0.00	0.00
j15c5a1	0.00	0.00	0.00	0.00	0.00
j15c5a2	0.00	0.00	0.00	0.00	0.00
j15c5a3	0.00	0.00	0.00	0.00	0.00
j15c5a4	0.00	0.00	0.00	0.00	0.00
j15c5a5	0.00	0.00	0.00	0.00	0.00
j15c5a6	0.00	0.00	0.00	0.00	0.00
j15c5b1	0.00	0.00	0.00	0.00	0.00
j15c5b2	0.00	0.00	0.00	0.00	0.00
j15c5b3	0.00	0.00	0.00	0.00	0.00

Table 2: Continued.

Case	$\overline{\text{RPD}}$				
	CT-I	CT-II	CT-III	CT-IV	CT-V
j15c5b4	0.00	0.00	0.00	0.00	0.00
j15c5b5	0.00	0.00	0.00	0.00	0.00
j15c5b6	0.00	0.00	0.00	0.00	0.00
j15c5c1	1.18	1.18	0.00	1.18	0.00
j15c5c2	0.00	0.00	0.00	0.00	0.00
j15c5c3	1.15	0.00	1.15	0.00	0.00
j15c5c4	1.12	1.12	1.12	1.12	0.00
j15c5c5	2.67	0.00	0.00	0.00	0.00
j15c5c6	0.00	0.00	0.00	0.00	0.00
j15c5d1	0.00	0.00	0.00	0.00	0.00
j15c5d2	1.18	0.00	0.00	0.00	0.00
j15c5d3	0.00	0.00	0.00	0.00	0.00
j15c5d4	2.38	1.19	1.19	1.19	0.00
j15c5d5	0.00	1.25	1.25	0.00	0.00
j15c5d6	1.23	1.23	0.00	1.23	0.00
j15c10a1	0.00	0.00	0.00	0.00	0.00
j15c10a2	0.00	0.00	0.00	0.00	0.00
j15c10a3	0.00	0.00	0.00	0.00	0.00
j15c10a4	0.00	0.00	0.00	0.00	0.00
j15c10a5	0.00	0.00	0.00	0.00	0.00
j15c10a6	0.00	0.00	0.00	0.00	0.00
j15c10b1	0.00	0.00	0.00	0.00	0.00
j15c10b2	0.00	0.00	0.00	0.00	0.00
j15c10b3	0.00	0.00	0.00	0.00	0.00
j15c10b4	0.00	0.00	0.00	0.00	0.00
j15c10b5	0.00	0.00	0.00	0.00	0.00
j15c10b6	0.00	0.00	0.00	0.00	0.00
Average	0.17	0.11	0.12	0.10	0.04

Table 3: Crossover probability.

Crossover probability	c_1	c_2
CP-I	0.2	0.2
CP-II	0.2	0.8
CP-III	0.5	0.5
CP-IV	0.8	0.2
CP-V	0.8	0.8

Table 4: Comparisons of different crossover probabilities.

Case	$\overline{\text{RPD}}$				
	CP-I	CP-II	CP-III	CP-IV	CP-V
Average	**0.08**	0.20	0.15	0.13	0.11

Table 5: Mutation probability.

Mutation type	Description
MT-I	Swap
MT-II	Insert
MT-III	Multiple swap
MT-IV	Multiple insert

Table 6: Comparisons of different mutation types.

Case	$\overline{\text{RPD}}$			
	MT-I	MT-II	MT-IIII	MT-IV
Average	0.08	0.08	0.02	**0.01**

Table 7: Mutation probability.

Mutation probability	p_m
MP-I	0.1
MP-II	0.2
MP-III	0.5
MP-IV	0.8
MP-V	0.9

Table 8: Comparisons of different mutation types.

Case	$\overline{\text{RPD}}$				
	MP-I	MP-II	MP-III	MP-IV	MP-V
Average	0.12	0.09	0.06	0.08	**0.02**

Table 9: Population size.

Population size	p_s
PS-I	10
PS-II	20
PS-III	30
PS-IV	50
PS-V	100

Table 10: Comparisons of different population sizes.

Case	$\overline{\text{RPD}}$				
	PS-I	PS-II	PS-III	PS-IV	PS-V
Average	0.06	0.08	0.03	0.06	**0.00**

Table 11: Different parameters for the canonical PSO.

	Parameter	Value	Description
1	Crossover type	CT-V	PTL
2	Crossover probability	CP-I	$c_1 = 0.2, c_2 = 0.2$
3	Mutation type	MT-IV	Multiple insert
4	Mutation probability	MP-V	0.9
5	Population size	PS-V	100

comparison results of different mutation types. It can be seen from Table 6 that the proposed multiple insert mutation operator performs the best among the compared algorithms.

5.2.4. Mutation Probability. To test the impact of different mutation probabilities, we implemented five kinds of mutation probabilities, that is, 0.1, 0.2, 0.5, 0.8, and 0.9, which

Table 12: Comparisons of the best $\overline{\text{RPD}}$ values.

Case	$\overline{\text{RPD}}$				
	PSO-ILS	ILS	IG	PSO	hGA
j10c5a2	0.00	0.00	0.00	0.00	0.00
j10c5a3	0.00	0.00	0.00	0.00	0.00
j10c5a4	0.00	0.00	0.00	0.00	0.00
j10c5a5	0.00	0.00	0.00	0.00	0.00
j10c5a6	0.00	0.00	0.00	0.00	0.00
j10c5b1	0.00	0.00	0.00	0.00	0.00
j10c5b2	0.00	0.00	0.00	0.00	0.00
j10c5b3	0.00	0.00	0.00	0.00	0.00
j10c5b4	0.00	0.00	1.64	0.00	0.00
j10c5b5	0.00	0.00	0.00	0.00	0.00
j10c5b6	0.00	0.00	0.00	0.00	0.00
j10c5c1	0.00	7.35	5.88	0.00	0.00
j10c5c2	0.00	2.70	2.70	0.00	0.00
j10c5c3	0.00	4.17	2.78	0.00	0.00
j10c5c4	0.00	4.55	4.55	0.00	0.00
j10c5c5	0.00	5.13	1.28	0.00	0.00
j10c5c6	0.00	4.35	1.45	0.00	0.00
j10c5d1	0.00	4.55	1.52	0.00	0.00
j10c5d2	0.00	1.35	0.00	0.00	0.00
j10c5d3	0.00	3.13	1.56	0.00	0.00
j10c5d4	0.00	2.86	2.86	0.00	0.00
j10c5d5	0.00	4.55	4.55	1.52	1.52
j10c5d6	0.00	4.84	3.23	0.00	0.00
j10c10a1	0.00	0.00	0.00	0.00	0.00
j10c10a2	0.00	2.53	3.16	0.00	0.00
j10c10a3	0.00	1.35	0.00	0.00	0.00
j10c10a4	0.00	0.00	0.00	0.00	0.00
j10c10a5	0.00	0.00	0.00	0.00	0.00
j10c10a6	0.00	2.05	4.11	0.00	0.00
j10c10b1	0.00	0.00	0.00	0.00	0.00
j10c10b2	0.00	0.64	0.64	0.00	0.00
j10c10b3	0.00	0.00	0.00	0.00	0.00
j10c10b4	0.00	0.00	0.00	0.00	0.00
j10c10b5	0.00	0.00	0.00	0.00	0.00
j10c10b6	0.00	0.00	0.00	0.00	0.00
j10c10c1	0.00	2.61	1.74	0.00	0.00
j10c10c2	0.00	2.52	1.68	0.00	0.00
j10c10c3	0.00	3.45	2.59	0.00	0.00
j10c10c4	0.00	2.50	1.67	0.00	0.00
j10c10c5	0.00	1.59	3.17	0.00	0.00
j10c10c6	0.00	1.89	3.77	0.00	0.00
j15c5a1	0.00	0.00	0.00	0.00	0.00
j15c5a2	0.00	0.00	0.00	0.00	0.00
j15c5a3	0.00	0.00	0.00	0.00	0.00
j15c5a4	0.00	0.00	0.00	0.00	0.00
j15c5a5	0.00	0.00	0.00	0.00	0.00
j15c5a6	0.00	0.00	0.00	0.00	0.00
j15c5b1	0.00	0.00	0.00	0.00	0.00
j15c5b2	0.00	0.00	0.00	0.00	0.00
j15c5b3	0.00	0.00	0.00	0.00	0.00
j15c5b4	0.00	0.00	0.00	0.00	0.00
j15c5b5	0.00	0.60	0.00	0.00	0.00
j15c5b6	0.00	0.00	0.00	0.00	0.00
j15c5c1	0.00	5.88	8.24	0.00	1.18
j15c5c2	0.00	4.40	4.40	0.00	0.00
j15c5c3	0.00	10.34	8.05	0.00	0.00
j15c5c4	0.00	3.37	5.62	0.00	0.00
j15c5c5	0.00	9.46	10.81	0.00	1.35
j15c5c6	0.00	7.69	6.59	0.00	0.00
j15c5d1	0.00	0.00	0.00	0.00	0.00
j15c5d2	0.00	9.52	9.52	1.19	0.00
j15c5d3	0.00	7.23	6.02	0.00	0.00
j15c5d4	0.00	7.14	5.95	1.19	1.19
j15c5d5	0.00	8.86	8.86	1.27	0.00
j15c5d6	0.00	4.94	4.94	1.23	1.23
j15c10a1	0.00	0.00	0.00	0.00	0.00
j15c10a2	0.00	2.00	2.00	0.00	0.00
j15c10a3	0.00	0.00	1.01	0.00	0.00
j15c10a4	0.00	0.00	1.78	0.00	0.00
j15c10a5	0.00	0.55	0.00	0.00	0.00
j15c10a6	0.00	1.00	0.00	0.00	0.00
j15c10b1	0.00	0.00	0.00	0.00	0.00
j15c10b2	0.00	0.00	0.00	0.00	0.00
j15c10b3	0.00	0.00	0.00	0.00	0.00
j15c10b4	0.00	0.00	0.00	0.00	0.00
j15c10b5	0.00	0.00	0.00	0.00	0.00
j15c10b6	0.00	0.00	0.00	0.00	0.00
Average	0.00	2.00	1.82	0.08	0.08

are given in Table 7. Table 8 gives the comparison results of different mutation probabilities. It can be seen from Table 8 that mutation probability with the value 0.9 performs the best among the compared algorithms.

5.2.5. Population Size. To test the impact of different population sizes, we implemented five kinds of population sizes, that is, 10, 20, 30, 50, and 100, which are given in Table 9. Table 10 gives the comparison results of different population sizes. It can be seen from Table 10 that population size with the value 100 performs the best among the compared algorithms.

5.2.6. The Final Parameters. After the comparison results for each kind of parameter, we can conclude the best parameters for the canonical PSO algorithm, which are given in Table 11.

5.3. Comparisons Analysis. To make a pair comparison with the present efficient algorithms, we coded the following algorithms to solve the HFS problem with PM activity. These compared algorithms include hGA by Ruiz and Maroto [47], IG by Ruiz and Stützle [48], ILS by Dong et al. [31], and PSO

Table 13: Comparisons of average $\overline{\text{RPD}}$ values.

Case	$\overline{\text{RPD}}$				
	PSO-ILS	ILS	IG	PSO	hGA
j10c5a2	0.00	0.00	0.00	0.00	0.00
j10c5a3	0.00	0.00	0.00	0.00	0.00
j10c5a4	0.00	0.00	0.00	0.00	0.00
j10c5a5	0.00	0.00	0.00	0.00	0.00
j10c5a6	0.00	0.73	1.45	0.00	0.00
j10c5b1	0.00	0.00	0.00	0.00	0.00
j10c5b2	0.00	0.00	0.00	0.00	0.00
j10c5b3	0.00	0.55	0.00	0.00	0.00
j10c5b4	0.00	0.49	3.44	0.00	0.00
j10c5b5	0.00	0.00	0.00	0.00	0.00
j10c5b6	0.00	0.00	0.00	0.00	0.00
j10c5c1	0.29	8.82	8.53	0.00	0.00
j10c5c2	0.81	3.50	3.77	0.00	0.00
j10c5c3	0.00	5.83	4.17	0.00	0.00
j10c5c4	0.00	6.36	6.06	0.00	0.00
j10c5c5	0.00	6.41	3.59	0.00	0.00
j10c5c6	0.00	5.80	4.93	0.00	0.00
j10c5d1	0.00	5.15	4.85	0.00	0.00
j10c5d2	0.00	2.97	2.70	0.00	0.00
j10c5d3	0.00	6.25	5.94	0.00	0.00
j10c5d4	0.00	3.71	4.57	0.00	0.00
j10c5d5	0.00	5.11	5.11	0.60	0.60
j10c5d6	0.00	6.45	5.81	0.00	0.00
j10c10a1	0.00	0.00	0.00	0.00	0.00
j10c10a2	0.00	3.67	3.67	0.00	0.00
j10c10a3	0.00	2.03	0.68	0.00	0.00
j10c10a4	0.00	0.00	1.48	0.00	0.00
j10c10a5	0.00	0.00	3.24	0.00	0.00
j10c10a6	0.00	3.70	4.79	0.00	0.00
j10c10b1	0.00	0.00	0.00	0.00	0.00
j10c10b2	0.00	0.89	2.80	0.00	0.00
j10c10b3	0.00	0.12	0.00	0.00	0.00
j10c10b4	0.00	0.00	0.00	0.00	0.00
j10c10b5	0.00	0.00	0.12	0.00	0.00
j10c10b6	0.00	0.36	0.36	0.00	0.00
j10c10c1	0.00	3.83	3.48	0.00	0.00
j10c10c2	0.00	3.53	2.69	0.00	0.00
j10c10c3	0.00	3.61	3.09	0.17	0.34
j10c10c4	0.00	3.17	2.83	0.00	0.00
j10c10c5	0.00	4.92	5.71	0.00	0.00
j10c10c6	0.00	2.45	4.53	0.00	0.00
j15c5a1	0.00	0.79	0.56	0.00	0.00
j15c5a2	0.00	0.00	0.00	0.00	0.00
j15c5a3	0.00	0.00	0.00	0.00	0.00
j15c5a4	0.00	0.00	0.26	0.00	0.00
j15c5a5	0.00	0.00	0.49	0.00	0.00
j15c5a6	0.00	0.00	0.00	0.00	0.00
j15c5b1	0.00	0.00	0.00	0.00	0.00
j15c5b2	0.00	0.00	0.00	0.00	0.00
j15c5b3	0.00	0.00	0.13	0.00	0.00
j15c5b4	0.00	0.54	0.68	0.00	0.00
j15c5b5	0.00	1.93	1.08	0.00	0.00
j15c5b6	0.00	0.00	0.11	0.00	0.00
j15c5c1	0.00	7.71	8.88	0.23	0.47
j15c5c2	0.00	6.81	6.59	0.22	0.00
j15c5c3	0.00	11.72	10.11	0.23	0.23
j15c5c4	0.00	5.58	5.80	0.22	0.00
j15c5c5	0.00	10.46	12.06	0.80	1.88
j15c5c6	0.00	9.45	9.01	0.00	0.00
j15c5d1	0.00	0.00	0.00	0.00	0.00
j15c5d2	0.00	9.48	9.48	0.95	0.47
j15c5d3	0.00	7.93	7.45	0.24	0.00
j15c5d4	0.00	7.58	7.58	0.71	0.71
j15c5d5	0.25	10.53	10.53	0.25	0.00
j15c5d6	0.00	6.14	6.39	0.74	0.74
j15c10a1	0.00	0.17	0.00	0.00	0.00
j15c10a2	0.00	3.30	3.20	0.00	0.00
j15c10a3	0.00	0.61	2.32	0.00	0.00
j15c10a4	0.00	0.00	1.96	0.00	0.00
j15c10a5	0.00	0.77	0.77	0.00	0.00
j15c10a6	0.00	2.70	0.10	0.00	0.00
j15c10b1	0.00	0.00	0.00	0.00	0.00
j15c10b2	0.00	0.00	0.21	0.00	0.00
j15c10b3	0.00	0.00	0.00	0.00	0.00
j15c10b4	0.00	0.00	0.00	0.00	0.00
j15c10b5	0.00	0.60	0.20	0.00	0.00
j15c10b6	0.00	0.00	0.00	0.00	0.00
Average	0.02	2.67	2.73	0.07	0.07

by Liao et al. [49]. The parameters for the compared algorithms are set to the same values in their literature, except that the stop condition is set to 20 seconds.

The comparison results for the best $\overline{\text{RPD}}$ values are given in Table 12. It can be seen from Table 12 that (1) for solving the HFS with PM activities, the proposed algorithm obtains all optimal results for 77 benchmark instances, which is obviously better than the other compared algorithms; (2) in average, the proposed algorithm is also better than the other compared algorithms; (3) the proposed PSO-ILS algorithm is better than the canonical PSO algorithm, which also verifies the efficiency of the ILS-based local search; (4) the proposed algorithm is better than the canonical IG algorithm, which shows the exploration ability of the proposed algorithm.

Table 13 reports the comparison results for the average $\overline{\text{RPD}}$ values. It can be seen from Table 13 that (1) the proposed algorithm obtains 74 optimal values out of 77 instances; (2) in average, the PSO-ILS algorithm obtains the best average $\overline{\text{RPD}}$ values, which is obviously better than the other algorithms. The following algorithms are PSO, hGA, ILS, and IG, respectively.

6. Conclusions

In this study, we proposed a hybrid algorithm for solving the HFS with PM activities. In the proposed algorithms, different crossover and mutation operators are applied for the learning procedure. The ILS-based local search procedure is embedded in the proposed algorithm to further improve the searching ability of the algorithm. Variation versions of 77 Carlier and Néron's benchmark problems are presented to adapt to the realistic industrial horizon. Experimental comparisons with four present algorithms show the efficiency and effectiveness of the proposed algorithm. The future work is to apply the proposed algorithm for solving rescheduling problems in hybrid and flexible environments [50–52].

Conflict of Interests

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

Acknowledgments

This research is partially supported by the National Science Foundation of China under Grant nos. 61104179 and 61174187, the Basic Scientific Research Foundation of Northeastern University under Grant no. N110208001, the Starting Foundation of Northeastern University under Grant no. 29321006, the Science Foundation of Liaoning Province in China (2013020016), and the Science Research and Development of Provincial Department of Public Education of Shandong under Grant no. J12LN39.

References

[1] A. Costa, F. A. Cappadonna, and S. Fichera, "A dual encoding-based meta-heuristic algorithm for solving a constrained hybrid flow shop scheduling problem," *Computers and Industrial Engineering*, vol. 64, no. 4, pp. 937–958, 2013.

[2] E. Figielska, "A heuristic for scheduling in a two-stage hybrid flowshop with renewable resources shared among the stages," *European Journal of Operational Research*, vol. 236, no. 2, pp. 433–444, 201.

[3] K. Mao, Q. K. Pan, X. Pang, and T. Chai, "A novel Lagrangian relaxation approach for a hybrid flowshop scheduling problem in the steelmaking-continuous casting process," *European Journal of Operational Research*, vol. 236, no. 1, pp. 51–60, 2014.

[4] H. Luo, A. Zhang, and G. Q. Huang, "Active scheduling for hybrid flowshop with family setup time and inconsistent family formation," *Journal of Intelligent Manufacturing*, 2013.

[5] L. C. Wang, Y. Y. Chen, T. L. Chen, C. Y. Cheng, and C. W. Chang, "A hybrid flowshop scheduling model considering dedicated machines and lot-splitting for the solar cell industry," *International Journal of Systems Science*, 2013.

[6] T. P. Chung and C. J. Liao, "An immunoglobulin-based artificial immune system for solving the hybrid flow shop problem," *Applied Soft Computing*, vol. 13, no. 8, pp. 3729–3736, 2013.

[7] M. Abbas Bozorgirad and R. Logendran, "Bi-criteria group scheduling in hybrid flowshops," *International Journal of Production Economics*, vol. 145, no. 2, pp. 599–612, 2013.

[8] F. D. Chou, "Particle swarm optimization with cocktail decoding method for hybrid flow shop scheduling problems with multiprocessor tasks," *International Journal of Production Economics*, vol. 141, no. 1, pp. 137–145, 2013.

[9] J. Yang, "A two-stage hybrid flow shop with dedicated machines at the first stage," *Computers and Operations Research*, vol. 40, no. 12, pp. 2836–2843, 2013.

[10] P. Ramezani, M. Rabiee, and F. Jolai, "No-wait flexible flowshop with uniform parallel machines and sequence-dependent setup time: a hybrid meta-heuristic approach," *Journal of Intelligent Manufacturing*.

[11] T. H. Tran and K. M. Ng, "A hybrid water flow algorithm for multi-objective flexible flow shop scheduling problems," *Engineering Optimization*, vol. 45, no. 4, pp. 483–502, 2013.

[12] H. Allaoui and A. Artiba, "Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints," *Computers and Industrial Engineering*, vol. 47, no. 4, pp. 431–450, 2004.

[13] J. Xie and X. Wang, "Complexity and algorithms for two-stage flexible flowshop scheduling with availability constraints," *Computers and Mathematics with Applications*, vol. 50, no. 10–12, pp. 1629–1638, 2005.

[14] H. Allaoui and A. Artiba, "Scheduling two-stage hybrid flow shop with availability constraints," *Computers and Operations Research*, vol. 33, no. 5, pp. 1399–1419, 2006.

[15] R. Ruiz, J. Carlos García-Díaz, and C. Maroto, "Considering scheduling and preventive maintenance in the flowshop sequencing problem," *Computers and Operations Research*, vol. 34, no. 11, pp. 3314–3330, 2007.

[16] B. Naderi, M. Zandieh, and S. M. T. Fatemi Ghomi, "A study on integrating sequence dependent setup time flexible flow lines and preventive maintenance scheduling," *Journal of Intelligent Manufacturing*, vol. 20, no. 6, pp. 683–694, 2009.

[17] A. Berrichi, L. Amodeo, F. Yalaoui, E. Châtelet, and M. Mezghiche, "Bi-objective optimization algorithms for joint production and maintenance scheduling: application to the parallel machine problem," *Journal of Intelligent Manufacturing*, vol. 20, no. 4, pp. 389–400, 2009.

[18] H. Luo, G. Q. Huang, Y. Zhang, Q. Dai, and X. Chen, "Two-stage hybrid batching flowshop scheduling with blocking and machine availability constraints using genetic algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 6, pp. 962–971, 2009.

[19] H. Allaoui and A. Artiba, "Johnson's algorithm: a key to solve optimally or approximately flow shop scheduling problems with unavailability periods," *International Journal of Production Economics*, vol. 121, no. 1, pp. 81–87, 2009.

[20] F. Jabbarizadeh, M. Zandieh, and D. Talebi, "Hybrid flexible flowshops with sequence-dependent setup times and machine availability constraints," *Computers and Industrial Engineering*, vol. 57, no. 3, pp. 949–957, 2009.

[21] W. Besbes, J. Teghem, and T. Loukil, "Scheduling hybrid flow shop problem with non-fixed availability constraints," *European Journal of Industrial Engineering*, vol. 4, no. 4, pp. 413–433, 2010.

[22] Y. Ma, C. Chu, and C. Zuo, "A survey of scheduling with deterministic machine availability constraints," *Computers and Industrial Engineering*, vol. 58, no. 2, pp. 199–211, 2010.

[23] H. Luo, G. Q. Huang, Y. Feng Zhang, and Q. Yun Dai, "Hybrid flowshop scheduling with batch-discrete processors and machine maintenance in time windows," *International Journal of Production Research*, vol. 49, no. 6, pp. 1575–1603, 2011.

[24] E. Safari and S. J. Sadjadi, "A hybrid method for flowshops scheduling with condition-based maintenance constraint and machines breakdown," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2020–2029, 2011.

[25] S. Wang and M. Liu, "Two-stage hybrid flow shop scheduling with preventive maintenance using multi-objective tabu search method," *International Journal of Production Research*, vol. 52, no. 5, 2014.

[26] M. Rabiee, R. S. Rad, M. Mazinani, and R. Shafaei, "An intelligent hybrid meta-heuristic for solving a case of no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machines," *The International Journal of Advanced Manufacturing Technology*, vol. 71, no. 5–8, pp. 1229–1245, 2014.

[27] H. Allaoui and A. Artiba, "Hybrid flow shop scheduling with availability constraints," in *Essays in Production, Project Planning and Scheduling*, pp. 277–299, Springer, 2014.

[28] T. Stützle, "Iterated local search for the quadratic assignment problem," *European Journal of Operational Research*, vol. 174, no. 3, pp. 1519–1539, 2006.

[29] H. Hashimoto, M. Yagiura, and T. Ibaraki, "An iterated local search algorithm for the time-dependent vehicle routing problem with time windows," *Discrete Optimization*, vol. 5, no. 2, pp. 434–456, 2008.

[30] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. van Oudheusden, "Iterated local search for the team orienteering problem with time windows," *Computers and Operations Research*, vol. 36, no. 12, pp. 3281–3290, 2009.

[31] X. Dong, H. Huang, and P. Chen, "An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion," *Computers and Operations Research*, vol. 36, no. 5, pp. 1664–1669, 2009.

[32] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Piscataway, NJ, USA, December 1995.

[33] B. Liu, L. Wang, and Y.-H. Jin, "An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers," *Computers and Operations Research*, vol. 35, no. 9, pp. 2791–2806, 2008.

[34] Q.-K. Pan, M. Fatih Tasgetiren, and Y.-C. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem," *Computers and Operations Research*, vol. 35, no. 9, pp. 2807–2839, 2008.

[35] J. Q. Li and Y. X. Pan, "A hybrid discrete particle swarm optimization algorithm for solving fuzzy job shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 66, no. 1–4, pp. 583–593, 2013.

[36] C. L. Chen, S. Y. Huang, Y. R. Tzeng, and C. L. Chen, "A revised discrete particle swarm optimization algorithm for permutation flow-shop scheduling problem," *Soft Computing*.

[37] Y. Marinakis and M. Marinaki, "Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem," *Soft Computing*, vol. 17, no. 7, pp. 1159–1173, 2013.

[38] P. Damodaran, A. G. Rao, and S. Mestry, "Particle swarm optimization for scheduling batch processing machines in a permutation flowshop," *International Journal of Advanced Manufacturing Technology*, vol. 64, no. 5–8, pp. 989–1000, 2013.

[39] G. Vijay chakaravarthy, S. Marimuthu, and A. Naveen Sait, "Performance evaluation of proposed differential evolution and particle swarm optimization algorithms for scheduling m-machine flow shops with lot streaming," *Journal of Intelligent Manufacturing*, vol. 24, no. 1, pp. 175–191, 2013.

[40] Y. Y. Chen, C. Y. Cheng, L. C. Wang, and T. L. Chen, "A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problems: a case study for solar cell industry," *International Journal of Production Economics*, vol. 141, no. 1, pp. 66–78, 2013.

[41] S. A. Torabi, N. Sahebjamnia, S. A. Mansouri, and M. A. Bajestani, "A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem," *Applied Soft Computing*, vol. 13, no. 12, pp. 4750–4762, 2013.

[42] J. Q. Li and Y. X. Pan, "A hybrid discrete particle swarm optimization algorithm for solving fuzzy job shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 66, no. 1–4, pp. 583–596, 20132013.

[43] J. T. Tsai, C. I. Yang, and J. H. Chou, "Hybrid sliding level Taguchi-based particle swarm optimization for flowshop scheduling problems," *Applied Soft Computing*, vol. 15, pp. 177–192, 2014.

[44] M. Nawaz, E. E. Enscore Jr., and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[45] 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.

[46] J. Carlier and E. Néron, "An exact method for solving the Multi-Processor Flow-Shop," *Operations Research*, vol. 34, no. 1, pp. 1–25, 2000.

[47] R. Ruiz and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," *European Journal of Operational Research*, vol. 169, no. 3, pp. 781–800, 2006.

[48] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[49] C.-J. Liao, C.-T. T. Tseng, and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Computers and Operations Research*, vol. 34, no. 10, pp. 3099–3111, 2007.

[50] J.-Q. Li, Q.-K. Pan, and Y.-C. Liang, "An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems," *Computers and Industrial Engineering*, vol. 59, no. 4, pp. 647–662, 2010.

[51] J.-Q. Li, Q.-K. Pan, and K.-Z. Gao, "Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems," *International Journal of Advanced Manufacturing Technology*, vol. 55, no. 9–12, pp. 1159–1169, 2011.

[52] J.-Q. Li, Q.-K. Pan, P. N. Suganthan, and T. J. Chua, "A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 52, no. 5–8, pp. 683–697, 2011.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 597278, 14 pages
http://dx.doi.org/10.1155/2014/597278

Research Article

A Fast Elitism Gaussian Estimation of Distribution Algorithm and Application for PID Optimization

Qingyang Xu, Chengjin Zhang, and Li Zhang

School of Mechanical, Electrical & Information Engineering, Shandong University, Weihai 264209, China

Correspondence should be addressed to Qingyang Xu; xuqy1981@163.com

Received 27 January 2014; Accepted 9 March 2014; Published 27 April 2014

Academic Editors: P. Agarwal and Y. Zhang

Copyright © 2014 Qingyang Xu 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.

Estimation of distribution algorithm (EDA) is an intelligent optimization algorithm based on the probability statistics theory. A fast elitism Gaussian estimation of distribution algorithm (FEGEDA) is proposed in this paper. The Gaussian probability model is used to model the solution distribution. The parameters of Gaussian come from the statistical information of the best individuals by fast learning rule. A fast learning rule is used to enhance the efficiency of the algorithm, and an elitism strategy is used to maintain the convergent performance. The performances of the algorithm are examined based upon several benchmarks. In the simulations, a one-dimensional benchmark is used to visualize the optimization process and probability model learning process during the evolution, and several two-dimensional and higher dimensional benchmarks are used to testify the performance of FEGEDA. The experimental results indicate the capability of FEGEDA, especially in the higher dimensional problems, and the FEGEDA exhibits a better performance than some other algorithms and EDAs. Finally, FEGEDA is used in PID controller optimization of PMSM and compared with the classical-PID and GA.

1. Introduction

Various optimization problems exist in engineering and academic research, which expect to find the best solution. If the problems are conventional or linear, the common mathematical methods will be effective. However, if the problems are too complicated to the common methods, some heuristic algorithms will be considered. Evolutionary algorithms (EAs) are very popular heuristic optimization techniques in the recent years. EAs are general terms of several optimization algorithms that are inspired by the Darwinian theory of natural evolution. It has the capability of solving the complicated optimization problems with nonlinear, high dimension and non-continuous characteristics. The algorithms search the optimal solution from many possible solutions, and the genetic operators, which simulate the principle of natural genetic evolution, are used to update the individuals. By several iterations, the optimal solution will be obtained, such as the genetic algorithms (GAs) [1], evolution strategies (ES), differential evolution (DE) [2, 3], and the artificial immune system (AIS) [4, 5] and also swarm evolutionary algorithm like particle swarm optimization (PSO) [2, 6, 7]. Although these algorithms have applied success to solve kinds of optimization problems [8], there are some inherent drawbacks. For example, if the building blocks spread all over the solutions, the standard EAs have very poor performance [9]. The recombination operators ether breaks the building blocks frequently or do not mix them effectively.

In recent years, estimation of distribution algorithms (EDAs) have attracted a lot of attention. It was proposed by Miuhlenbein and Paaß [10] and emerged as a generalization of EAs for overcoming some problems of EAs, like building blocks broken, poor performance in high dimensional problems, and the difficulty of modeling the distribution of solutions. Sometimes the gene blocks are built based on simple selection and crossover operators are not effective enough to get optimum solution as the blocks may be broken in EAs [9, 11]. Compared with building blocks in EAs, EDAs do not use the crossover or mutation operator to update individuals [12]. Instead, they extract the global statistical information from the superiority individual of previous iteration and build the distribution probability model of solution for sampling new individuals [13]. It is the main advantages of EDAs compared with EAs that the search process is guided by the probabilistic model with explanatory and transparent characteristics [14, 15]. The algorithms are based on the probabilistic models following two steps: (1) Statistics the information of selected individuals and establish the probability model and (2) generate new population by sampling the probability model. Therefore, the new offspring of EDAs is based on the probability distribution instead of performing recombination of individuals as EAs.

The type of probabilistic models used by EDAs and the methods employed to learn them may vary according to the characteristics of the optimization problem. Therefore, different EDAs have been proposed for discrete and continuous problems. In traditional EDAs, the individuals are encoded with binary strings inheritance from EAs. In the population-based incremental learning (PBIL) algorithm [16], the individuals are encoded as fixed length binary strings. The population of solutions is updated by the probability vector, which is initially set to probability 0.5 for each position of the binary strings. For univariate marginal distribution algorithm (UMDA) [10], the frequencies of values on each position are computed according to the selected individuals, which are then used to generate the new population. The compact genetic algorithm (cGA) [17] updates the population according to the probability vector like the PBIL. However, unlike the PBIL, it modifies the probability vector according to the contribution of individuals.

In case of real-valued problems, there are some approaches to extend EDAs to other domains, such as mapping other domains to the domain of fixed-length binary strings and then mapping the solution back to the problem's original domains, or extend or modify the class of probabilistic models to other domains. This first approach might lead to undesirable limitations and errors on real-coded problems. For the second method, the normal pdf is commonly used in continuous EDAs to represent univariate or multivariate distributions. Therefore, some EDAs based on the Gaussian distribution have been designed. In the stochastic hill-climbing with learning by vectors of normal distributions [18], the population of solutions is modeled by a vector of mean of Gaussian normal distribution μ_i for each variable. The standard deviation σ is stored globally and it is the same for all variables. After generating a number of new solutions, the mean μ_i are shifted towards the best solutions and the standard deviation σ is reduced to make future exploration more specifically. Various ways of modifying the σ parameter have been exploited in [19]. Regularized estimation of distribution algorithms (RegEDA) [20] makes use of regularized model estimation in EDAs for continuous optimization. The regularization techniques can lead to more robust model estimation in EDAs. Continuous Gaussian estimation of distribution algorithm (CGEDA) [14] which is a kind of multivariate EDAs is proposed for real-coded problems. Gaussian data distribution and dependent individuals are two assumptions that are considered in CGEDA. In the algorithm, the joint distribution of promising solutions is used in every dimension of the problem. In literature [21, 22], an estimation of distribution algorithm with Gaussian process based on a subspaces method was proposed, which can reduce the computation of complex problems. A real-coded EDA using multiple probabilistic models (RMM) was proposed [23], which includes multiple types of probabilistic models with different learning rates and diversities. There are also other EDAs, which adopt more involved probability models and mixtures of pdfs. However, the probability models cannot reflect the problem completely, because it is hard to obtain an accurate model. In particular, with the increases of number of variables and the number of mixture components, the optimization results become unreliable [24]. Therefore, we specifically focus on the use of the single normal distribution in this paper, as it is more intuitive to be analyzed. Moreover, the use of single and easy normal pdf will not prevent us from obtaining a better understanding of the exploitation of the solutions. We propose a fast elitism Gaussian EDA (FEGEDA) based on the standard process of EDA. A fast learning rule is used to parameters of pdf learning, and an elitism strategy is used for a better performance. Hence, the increased convergence exhibited in this study is expected.

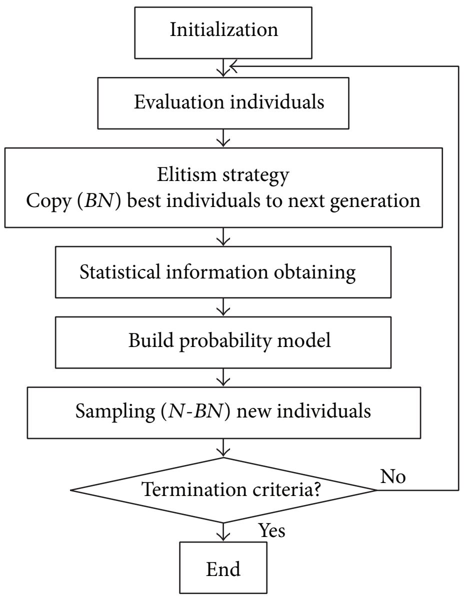

Initialization
Evaluation individuals
Elitism strategy
Copy (BN) best individuals to next generation
Statistical information obtaining
Build probability model
Sampling (N-BN) new individuals
Termination criteria?
No
Yes
End

Figure 1: Flowchart of FEGEDA.

2. The Fast Elitism Gaussian EDA

2.1. The Framework of the Algorithm. EDA is realized by probability estimation and sampling. The probability model is used to estimate the solution distribution, and the probability sampling is used to generate new individuals. In order to improve the performance of standard EDA, we adopt an elitism strategy in FEGEDA. Figure 1 is the flowchart of FEGEDA.

The steps of the FEGEDA are as follows.

Step 1 (initialization). Set the population size N, define the number BN of best individuals for probability model establishment and generate the initialized population Pop(0).

Step 2 (population evaluation). Evaluate the N individuals $x_1, x_2, \ldots x_N$ according to fitness function $f(x)$.

Step 3 (statistical information obtaining). Select BN best individuals according to the fitness and obtain the statistical information of mean μ and standard deviation σ.

Step 4 (probability model $P(x_1, x_2, \ldots, x_m)$ establishment). Use the fast learning rule and build the Gaussian normal distribution by the u of means and a covariance σ.

Step 5 (new population Pop(k) generation). Make use of sampling technique to generate a new population according to the probability model built in Step 4.

Step 6. Finally, the iteration is terminated according to the termination criteria. These criteria can be as simple as a fixed number of generations or imply a statistical analysis of the current population to evaluate the stopping condition criteria. If the stopping conditions do not meet, return to Step 2.

The probability model is built according to the distribution of the best solutions in the current population. Therefore, sampling solutions from this model should fall in promising areas with high probability. For some iterations, the solutions should be more likely to be close to the global optimum. The details of the main algorithm are explained in the following.

2.2. Initialization. In the algorithm, little parameters are needed to set except for the population size N and the best individuals size BN selected to build the probability model. Then, a random function is used to generate the initial population according to the variable domain $[L_i, H_i]$. Make use of random function generating variables $z_i \in [a_i, b_i]$ and then convert to the domain $[L_i, H_i]$ by

$$x_n^i = L_i + \frac{H_i - L_i}{b_i - a_i}(z_i - a_i), \tag{1}$$

where x_n^i is the ith optimization variable of nth individual, z_i is the ith random variable, a_i and b_i are the bounds of ith random variable, and L_i, and H_i are the bounds of ith optimization variable.

2.3. Population Evaluation. In the individuals' evaluation, it depends on the characteristics of the problem. Conventionally, we should define an objective function $f(x)$ in order to evaluate the fitness of individuals. Consider

$$\begin{aligned}
&\text{Min (\& Max)} && y = f(x) \\
&\text{S.t.} && g(x) = [g_1(x), g_2(x), \ldots, g_k(x)] \le 0 \\
& && h(x) = [h_1(x), h_2(x), \ldots, h_j(x)] = 0 \\
& && x = [x_1, x_2, \ldots, x_i, \ldots, x_m] \\
& && L_i \le x_i \le H_i \quad (i = 1, 2, \ldots, m),
\end{aligned} \tag{2}$$

where x is m dimensional optimization variable, $f(x)$ is the objective function, $g_k(x)$ is the kth inequality constraints, and $h_j(x)$ is the jth equality constraints. L_i and H_i are the bounds of variable.

2.4. The Establishment of Probability Model. The most important and crucial step of EDAs is the construction of probabilistic model for the solution distribution; to do this step of FEGEDA, Gaussian distribution of individuals is assumed to model and estimate the distribution of promising solutions in every dimension of the problem. Therefore, mean and standard deviation parameters of promising population are required which computed adaptively by maximum likelihood technique.

2.4.1. Statistical Information Acquisition. In order to construct a pdf model of the promising solutions, we should obtain the statistical information of promising solutions. Hence, statistical techniques have been extensively applied to the optimization problems. Fortunately, these parameters can be efficiently computed by the maximum-likelihood estimations [24].

In the pdf models that assume full independence, every variable is assumed independent of any variable. It must be noted that, in difficult optimization problems, different dependency relations can appear between variables and, hence, considering all of them independent may provide a model that does not represent the problem accurately. However, even if more involved probability models and mixtures of pdfs are defined and used in EDAs, the probability models cannot reflect the problem completely. For system modeling, the dependency relations between variables are very important. Conversely, for optimization problem, the problem decoupled as the combination of some independent variables is expected. Therefore, we specifically focus on the use of independent probability model to construct a fast elitism Gaussian EDA with better performance. That is, the probability distribution $P(x_1, x_2, \ldots, x_m)$ of the vector $(x_1, x_2, \ldots, x_m)$ of m variables is assumed to consist of a product of the distributions of individual variables:

$$P(x_1, x_2, \ldots x_m) = \prod_{i=1}^{m} P(x_i). \tag{3}$$

This is very suitable for calculation. Different from the discrete case, the number of parameters to be estimated does not grow exponentially with m. Therefore, it is relatively fast and efficient.

The mean and covariance parameters of the normal pdf can be estimated from the selected individuals. Consider

$$\begin{aligned}
\overline{\mu}_i(k) &= \frac{1}{N}\sum_{n=1}^{BN} x_i^n(k), \\
\sigma_i^2(k) &= \frac{1}{N}\sum_{n=1}^{BN} (x_i^n(k) - \overline{\mu}_i(k))(x_i^n(k) - \overline{\mu}_i(k))^T,
\end{aligned} \tag{4}$$

$\overline{\mu}_i(k)$ is the mean of ith variable in kth iteration, BN is the selected individuals size, and $\sigma_i^2(k)$ is the covariance of ith variable in kth iteration.

These parameters are always learned in the process of optimization. The iterative learning approaches are used in some literatures [23, 25–27] as follows:

$$\overline{\mu}_i(k) = \alpha\overline{\mu}_i(k) + \beta\overline{\mu}_i(k-1), \tag{5}$$

$$\sigma_i^2(k) = \alpha\sigma_i^2(k) + \beta\sigma_i^2(k-1), \tag{6}$$

where α and β are the weights of $\overline{\mu}_i(k)$ and $\overline{\mu}_i(k-1)$. The learning method depends on the class of models used; this step involves parametric or structural learning, also known as model fitting and model selection, respectively. This can improve the performance of EDAs, no matter how simple or complex the learning rule is. We adopt a fast learning method ($\alpha = 1$ and $\beta = 0$) in this paper, and an elitism strategy is adopted to maintain a smooth convergence process.

2.4.2. Probability Model. In this paper, the normal pdf $N(\mu_i, \sigma_i)$ for variables x_i is parameterized by the u of means and covariance σ, which is defined by

$$N(x_i, \mu_i, \sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-(x_i-\mu_i)^2/2\sigma_i^2}. \tag{7}$$

The probability distribution $P(x_1, x_2, \ldots, x_m)$ of the vector $(x_1, x_2, \ldots, x_m)$ of m variables is

$$P(x_1, x_2, \ldots x_m) = \prod_{i=1}^{m} \frac{1}{\sigma_i\sqrt{2\pi}} e^{-(x_i-\mu_i)^2/2\sigma_i^2}. \tag{8}$$

The parameters (μ_i, σ_i) have been estimated according to the selected best individuals. The estimation of marginal parameters (μ_i, σ_i) is updated in every iteration.

2.5. Probability Sampling. The probability sampling is used to generate new individuals using the learned probabilistic models. The sampling method depends on the type of probabilistic model and the characteristics of the problem. For normal pdf problem, a conversion is used in order to convert the normal pdf to a standard normal pdf.

Suppose

$$y = \frac{x-\mu}{\sigma}. \tag{9}$$

The normal pdf about x is converted to a standard normal pdf about y. Consider

$$N(x, \mu, \sigma) \longrightarrow N(y, 0, 1). \tag{10}$$

The variable x can be calculated by

$$x = \sigma y + \mu. \tag{11}$$

In the probability models, every variable $(x_1, x_2, \ldots, x_m)$ is assumed independent of any variable. The mean and variance of variable x_i are μ_i and σ_i; when $n \to \infty$,

$$y = \frac{\left(\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \mu_i\right)}{\sqrt{\sum_{i=1}^{n} \sigma_i^2} \to N(y, 0, 1)}. \tag{12}$$

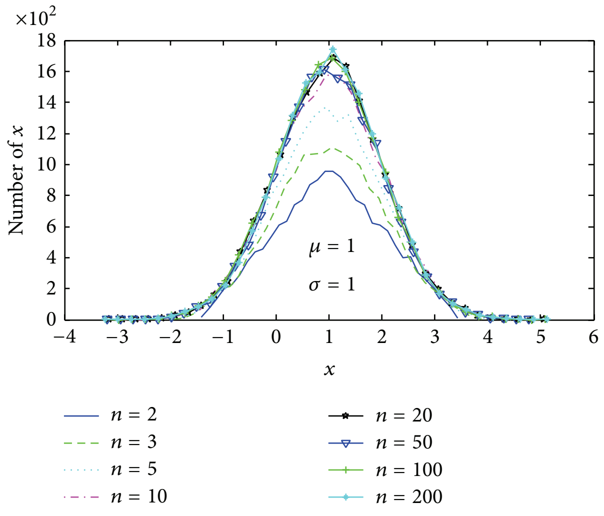


FIGURE 2: Cartogram of sampling data.

If x_i is the evenly distributed random number of $[0, 1]$,

$$\mu_i = E(x) = \frac{1}{2}, \qquad \sigma_i^2 = V(x) = \frac{1}{12}. \tag{13}$$

Therefore,

$$y = \frac{\left(\sum_{i=1}^{n} x_i - n/2\right)}{\sqrt{n/12}} \tag{14}$$

when $n \to \infty$ and $y \to N(0, 1)$. We can select an appropriate n to generate a normal pdf for probability sampling. Figure 2 shows the cartogram of sampling data in different n. From the figure, we can see the sampling data following the pdf better with the increasing of n.

2.6. Elitism Strategy. Elitism strategy is an effective strategy to ensure that the best individual is selected as the next generation in EAs, because the best individual may include the information of optimal solution. Therefore, elitism can improve the convergence performance of EAs in many cases [28], and elitism has long been considered an effective method for improving the efficiency of EAs [29]. This is achieved by simply copying the best individual directly to the new generation [30]. However, the number of best individuals selected as the next generation must be handled properly and carefully; otherwise it may lead to premature convergence or cannot improve the efficiency of algorithm. Figure 3 is the process of new population generation. The elitism individuals will be selected as the new generation directly, and the best individuals are used to establish a probability model to generate other individuals of the next generation. Consider

$$\text{Pop}(k) = \text{Elitism}(BN)_{k-1} \cup \text{Sample}(N - BN)_{k-1}, \tag{15}$$

where Elitism(BN) is the operator to copy the best solution of Pop($k-1$) to Pop(k) Sample() is the sampling function,

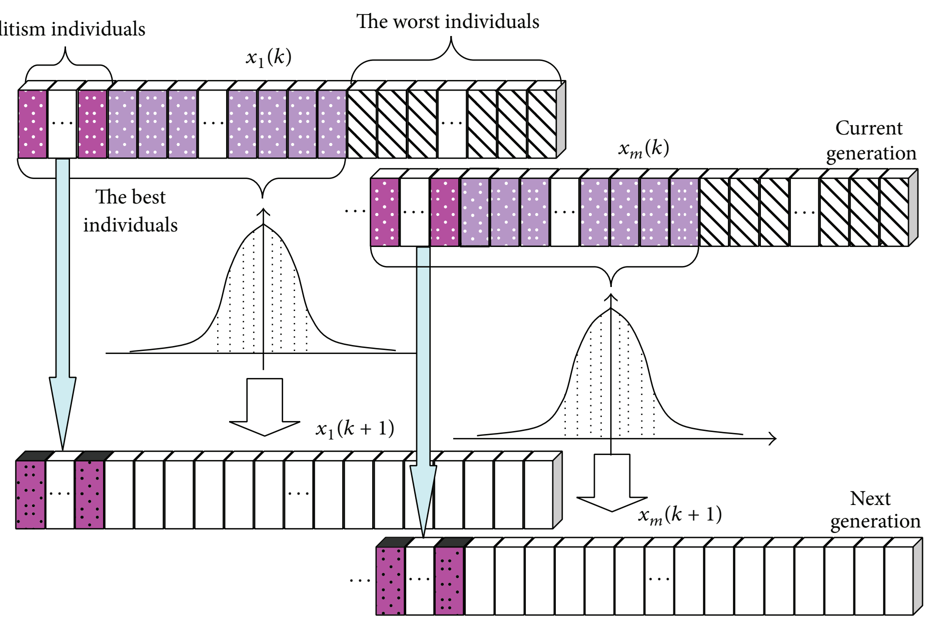


Figure 3: Population operation diagram.

N is the population size, and BN is the best selected individuals number.

3. Simulation

In the simulation, in order to exhibit the performance of FEGEDA, we carry out several different simulations, including one-dimensional benchmark, two-dimensional benchmarks, and higher dimensional benchmarks. Moreover, we compare the FEGEDA with other EDAs and other kinds of optimization algorithms.

3.1. One-Dimensional Benchmark. One-dimensional problem is easy for FEGEDA. In order to visualize the information of optimization process and models learning process during the evolution clearly, we carry out a one-dimensional benchmark optimization simulation:

$$f_0(x) = \sin(x) + \sin\left(\frac{10}{3}x\right) + \log(x) - 0.84x + 3, \quad (16)$$

where f_0 is a multimodal [31], $x \in [2.7, 7.5]$, with several local minimum value, and the global minimum value 1.6013 at $x = 5.19978$.

The best individuals number BN selected to build the probability model is a very important parameter for FEGEDA. The elitism strategy is a very important strategy to maintain a smooth optimization process in this paper. Therefore, in order to visualize the performance of corresponding part, we use different BN to testify the effect of outstanding individuals No. to the algorithm performance, and the elitism strategy is optional to testify the effect of the elitism strategy to the convergent performance of the algorithm. Many literatures [32–34] have proved that EDAs are convergent under certain conditions. From Figure 4, we can see that the optimization processes are unstable due to the

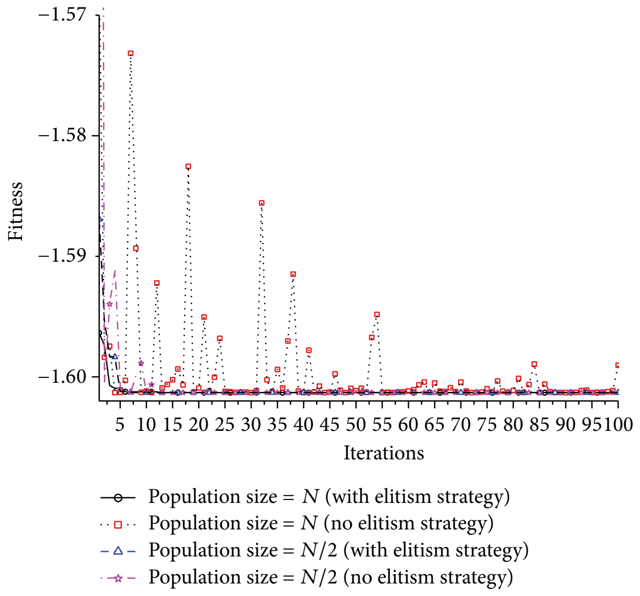
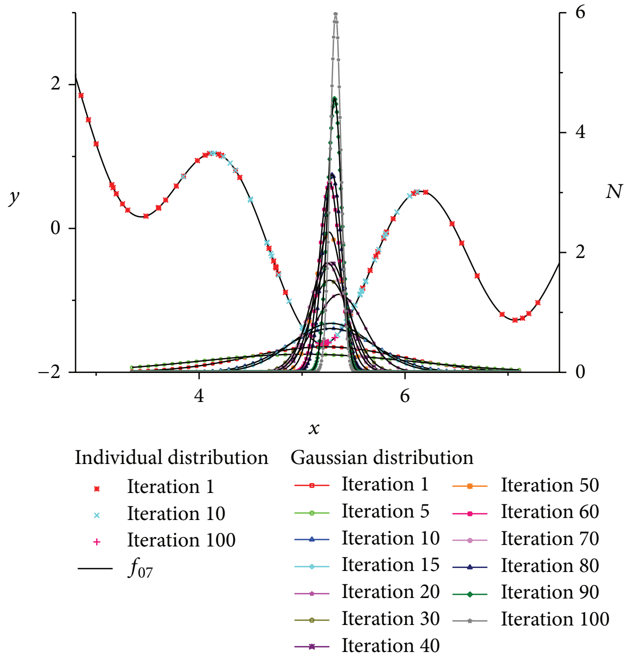


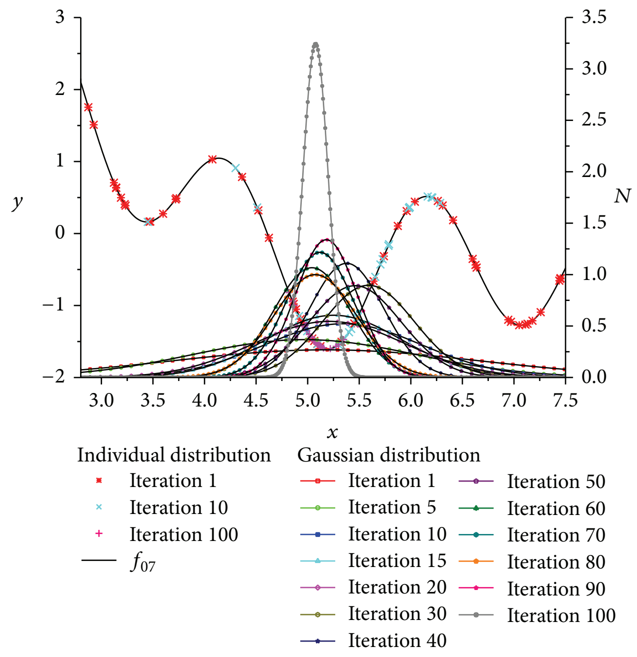
Figure 4: The optimum solutions of each iteration under different conditions.

use of fast learning rule when the algorithm is without elitism strategy no matter what BN is. The elitism strategy can make the convergent process smooth and improve the convergent performance too.

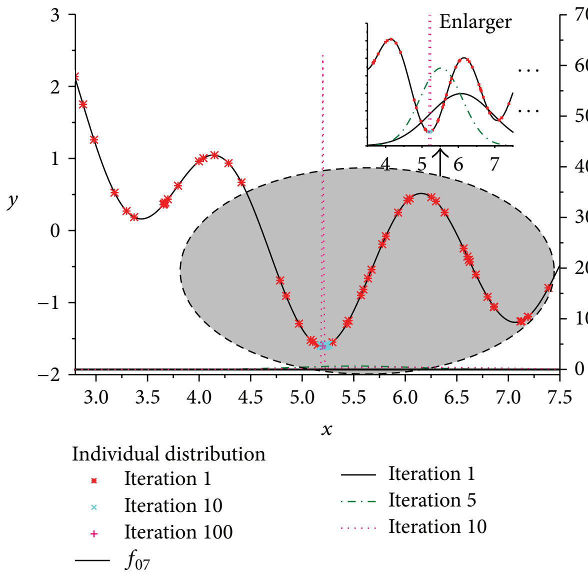
In Figure 5, the individuals' distribution and probability models of some iteration are exhibited. The individuals' distributions of iterations 1, 10, and 100 are shown in the Figure 5. The individuals spread over the optimized function at initial iteration, and then the individuals will focus on the area of optimum solution with the iterations going on.



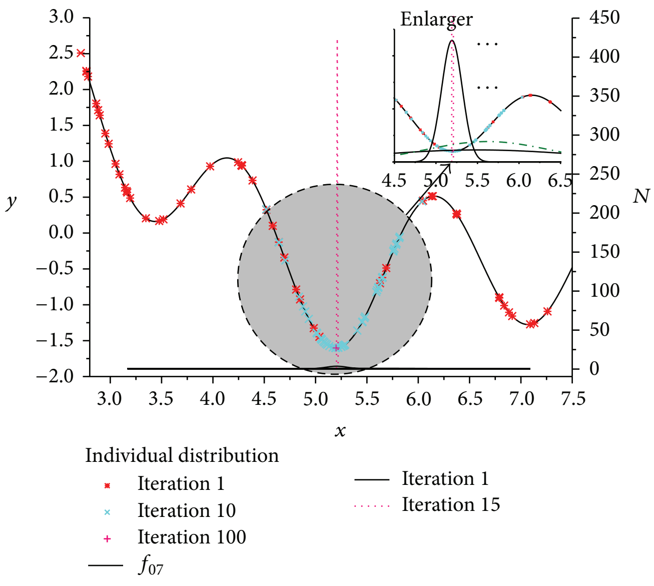
(a) $BN = N$, using elitism strategy



(b) $BN = N$, without elitism strategy



(c) $BN = N/2$, using elitism strategy



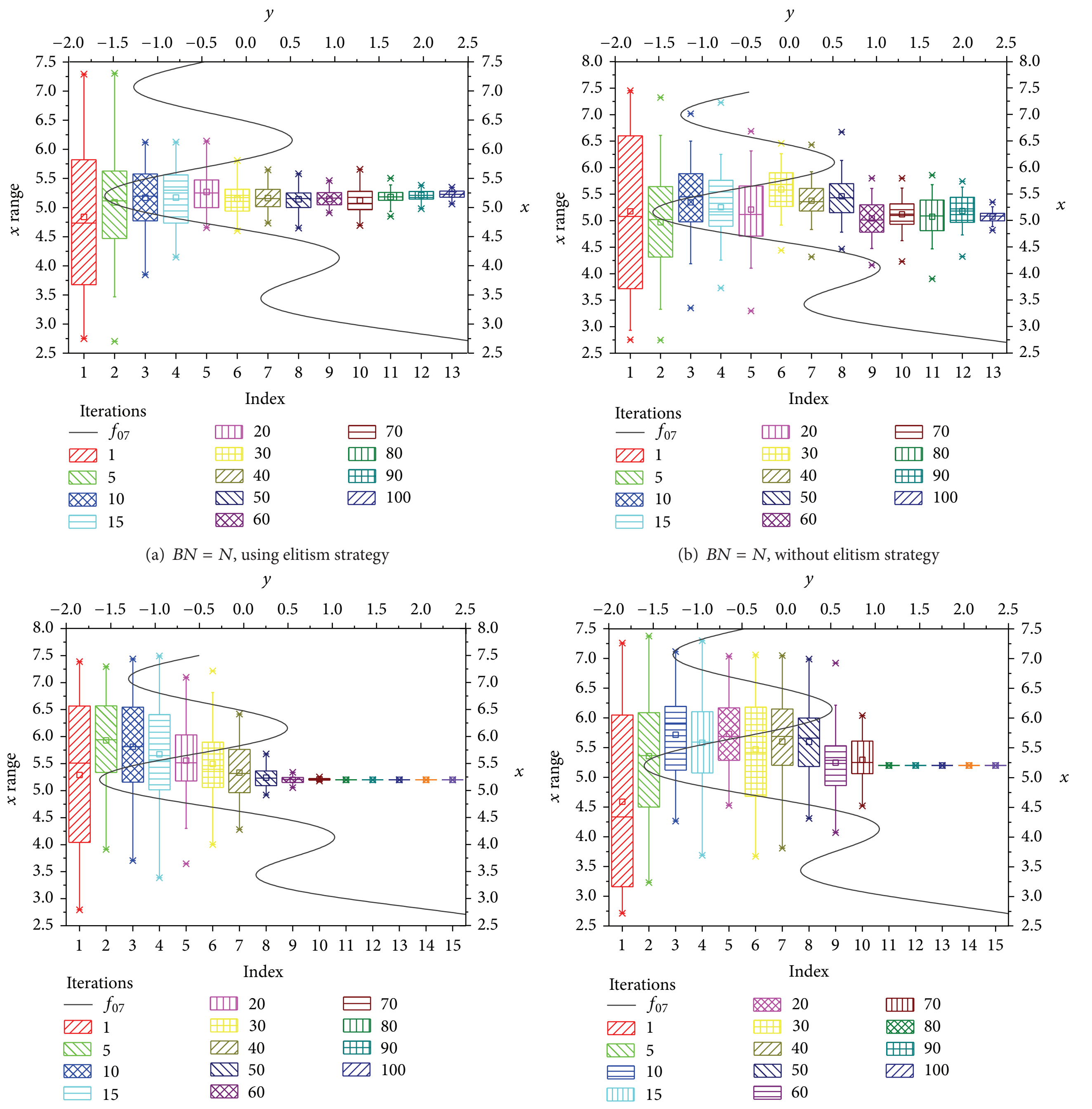
(d) $BN = N/2$, without elitism strategy

FIGURE 5: The individuals distribution and probability model of different iterations.

Therefore, the diagram of pdf is flat at the beginning. The parameter μ of pdf is smaller and smaller with the increase of iteration and focus on the global optimum solution. The probability models learning process is shown in Figure 5. The elitism strategy is a very important part of the algorithm. Form the exhibition of probability models learning process in Figure 5, we can see that the probability model learning process of solution is smooth when adopting elitism strategy; otherwise it is unstable.

The best selected individuals number is also an important parameter. The convergent speed is faster when the best selected individuals number BN is $N/2$. However, if it is too small, it will lead to premature.

Figure 6 is the statistics information of population of some iteration. Form Figure 6, we can see the population distribution when $BN = N$ using elitism strategy (Figure 6(a)) or without elitism strategy (Figure 6(b)), and $BN = N/2$ using elitism strategy (Figure 6(c)) or without elitism strategy (Figure 6(d)). According to Figure 6, the distribution of population is stable when using elitism strategy; otherwise it is fluctuant regardless of $BN = N$ or $BN = N/2$. A small BN can make the individuals focus on a certain area quickly.



(a) $BN = N$, using elitism strategy

(b) $BN = N$, without elitism strategy

(c) $BN = N/2$, using elitism strategy

(d) $BN = N/2$, without elitism strategy

FIGURE 6: The boxplot of population for different iterations.

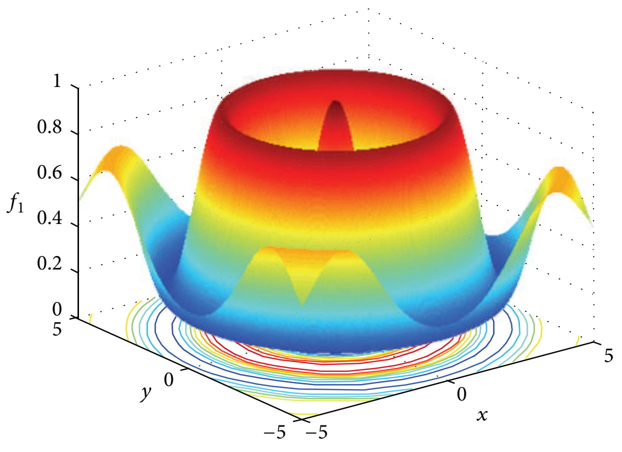
3.2. Two-Dimensional Problems. In order to testify the optimization capability of FEGEDA further, three two-dimensional complex functions are considered:

$$f_1(x, y) = 0.5 - \frac{\sin^2\sqrt{x^2 + y^2} - 0.5}{\left(1 + 0.001 * \left(x^2 + y^2\right)^2\right)^2},$$

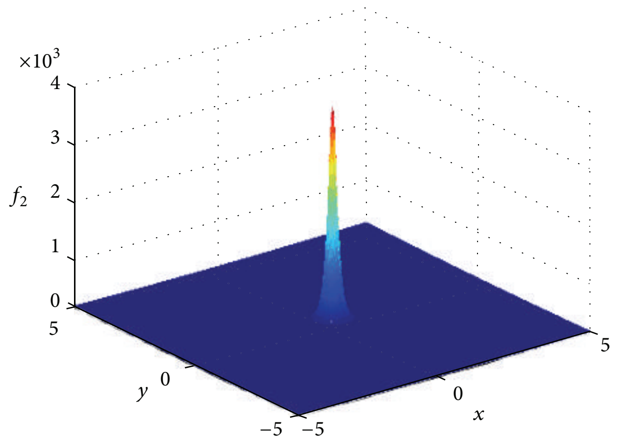
$$f_2(x, y) = \left(\frac{3}{0.05 + \left(x^2 + y^2\right)^2}\right)^2 + x^2 + y^2$$

$$f_3(x, y) = -\left(x^2 + y^2\right)^{0.25}\left(\sin^2 50 * \left(x^2 + y^2\right)^{0.1} + 0.1\right), \tag{17}$$

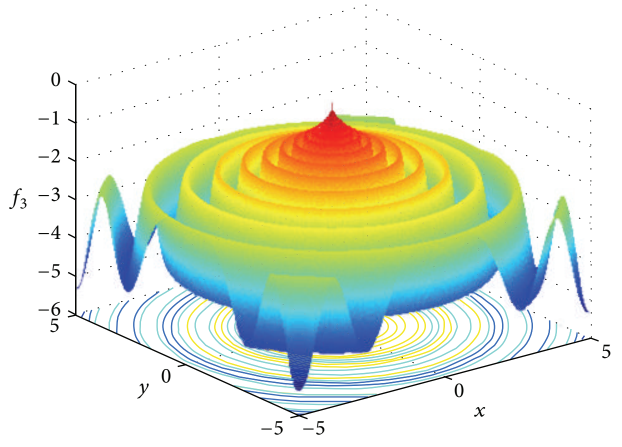
where $x, y \in [-5.12, 5.12]$. f_1 has infinite maximum value, and the global maximum value 1 is point $(0, 0)$. A circuit ridge surrounds the global maximum value. Hence, it is easy to fall into local maximum, which can be used to test the global searching capability of the algorithm. f_2 is a local peak function, and the maximum value is 3600 at point $(0, 0)$. This function can be used in determining the local



(a) Function f_1

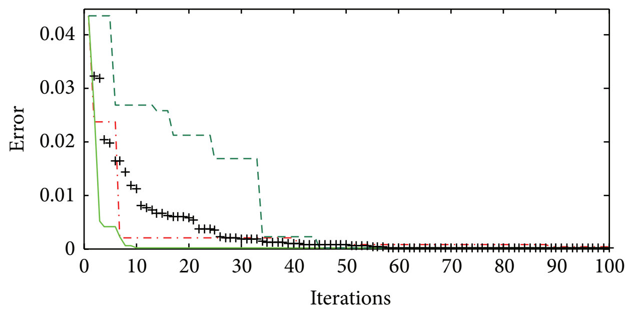


(b) Function f_2

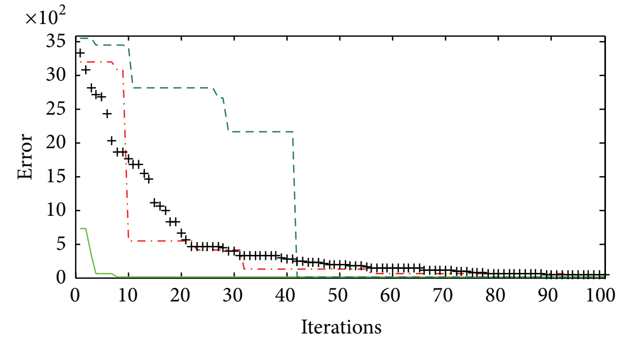


(c) Function f_3

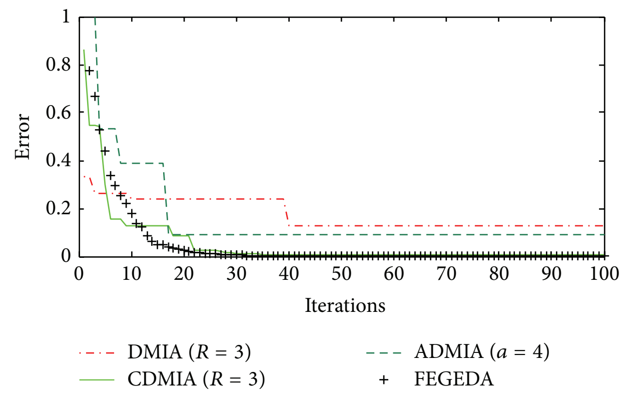
Figure 7: Function diagrams of f_1, f_2, and f_3.



(a) f_1



(b) f_2



(c) f_3

Figure 8: Comparisons of convergent results.

The population size N is set to 50, the maximum iteration is set to 100, and BN is set to $N/2$ in order to have comparison under the same conditions. From Figure 8, we can see that the FEGEDA can get the optimum solution faster. It has similar optimization capability of CDMIA, which has preferable performance for the three benchmarks.

searching capability of the algorithm. The f_3 function is a complicated function with a vibrating circuit ridge outside the global maximum value 0. This function can verify the global and local optimization capability of the algorithm simultaneously. Figure 7 shows the functions f_1, f_2, and f_3 correspondingly. We compare the optimization result with three other algorithms [35].

3.3. Higher Dimensional Problems. The advantage of FEGEDA is the capability of higher dimensional problems solution. Some typical benchmarks are considered, including Quadric, Rosenbrock, Ackley, Griewank, Rastrigrin, and Schaffer's f_7 function [21], which are shown in Table 1. In addition, they are configured with a dimension $n = 10$. In order to compare with other EDAs under the same

Table 1: High dimensional benchmarks.

Quadric $n = 10$	$f_4(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2, \quad -100 \le x_i \le 100$
Rosenbrock $n = 10$	$f_5(x) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_1\right)^2\right], \quad -50 \le x_i \le 50$
Ackley $n = 10$	$f_6(x) = -20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}\right) - \exp\left(\sqrt{\frac{\sum_{i=1}^{n}\cos(2\pi x_i)}{n}}\right) + 20 + e, \quad -30 \le x_i \le 30$
Griewank $n = 10$	$f_7(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i)^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad -300 \le x_i \le 300$
Rastrigrin $n = 10$	$f_8(x) = \sum_{i=1}^{n}\left[x_i^2 - A\cos(2\pi x_i) + A\right], \quad -5.12 \le x_i \le 5.12$
Schaffer's f_7 $n = 10$	$f_9(x) = \sum_{i=1}^{n}\left(x_i^2 + x_{i+1}^2\right)^{0.25} \times \left\{\sin\left[50 \times \left(x_i^2 + x_{i+1}^2\right)^{0.1}\right] + 1.0\right\}, \quad -100 \le x_i \le 100$

conditions, the population size N of FEGEDA is 300 and the maximum iteration is 100.

The algorithm is testified under different BN (from N to $N/20$). The convergent results under different BN are shown in Figure 9. Form Figure 9, we can see that the optimization process is slow when $BN = N$. With the decrease of BN, the convergent speed is faster. However, the increase of convergent speed is limited. If the BN is too small, the optimization will trap into local minimum easily.

We have a comparison of FEGEDA with other EDAs in [21]. Figure 10 is the comparison diagram. From Figure 10, we can see that FEGEDA is superior to standard EDA and other improved ones for the six benchmarks. For Ackley function, the performance of FEGEDA is the same as EDA. For Rosenbrock function, the initial fitness of FEGEDA is lower than other EDAs. Therefore, we put an enlarger diagram of corresponding area.

4. PID Controller Optimization

PID is the most used controller in the permanent magnet synchronous motors (PMSM) control. However, PID controller has poor performance in PMSM control due to the inappropriate parameters. During the past decades, great attention has been paid to the stochastic approach, which is potential in dealing with the problem [36, 37]. In this paper, we adopt FEGEDA to optimize the PID controller of PMSM.

4.1. Mathematic Model of PMSM. The mathematical model of PMSM in a d, q two-phase rotating coordinate system is shown below [38]. The voltage equation is

$$\begin{aligned} u_q &= R_s i_q + L_q \dot{i}_q + \omega_e L_d i_d + \omega_e \psi_f, \\ u_d &= R_s i_d + L_d \dot{i}_d - \omega_e L_q i_q, \end{aligned} \tag{18}$$

where u_d and u_q represent the stator winding shaft in a straight axis and the quadrature voltage, respectively; i_d and i_q are the direct-axis current and quadrature-axis current, respectively; R_s is the stator phase resistance; L_d is the straight axis inductance; L_q is the quadrature axis inductance; ψ_f is the permanent-magnet fundamental excitation magnetic field and stator winding of the magnetic chain; w_e is the electric angular speed of rotor.

The magnetic linkage equation can be expressed as follows:

$$\begin{aligned} \psi_d &= L_d i_d + \psi_f, \\ \psi_q &= L_q i_q, \end{aligned} \tag{19}$$

where ψ_d and ψ_q represent the syntheses of the magnetic fields in space-direct and quadrature-axis stator winding of the magnetic chain, respectively.

The electromagnetic torque of PMSM in the d, q coordinate is

$$T_e = p_n\left(\psi_f i_q - \left(L_d - L_q\right) i_p i_d\right), \tag{20}$$

where p_n is number of pole pairs.

According to the motion equation of motor,

$$\begin{aligned} Jp\dot{\Omega}_r &= T_e - T_l - B\Omega_r, \\ \Omega_r &= \frac{\omega_e}{p_n}, \end{aligned} \tag{21}$$

where Ω_r is mechanical angular speed of rotor, B is the viscous friction coefficient, J is the total moment inertia of rotor and load, and T_l is the load torque.

Thus, the state equation can be derived from the above equations as follows:

$$\begin{aligned} \dot{i}_q &= \frac{1}{L_q}\left(u_q - R_s i_q - L_d i_d w_e - \psi_f w_e\right), \\ \dot{i}_d &= \frac{u_d}{L_d}\left(u_d - R_s i_d - w_e L_q i_q\right), \\ \dot{w}_e &= \frac{1.5 p_n^2\left(\psi_f i_q + \left(L_d - L_q\right) i_d i_q\right) - p_n T_m - B w_e}{J}. \end{aligned} \tag{22}$$

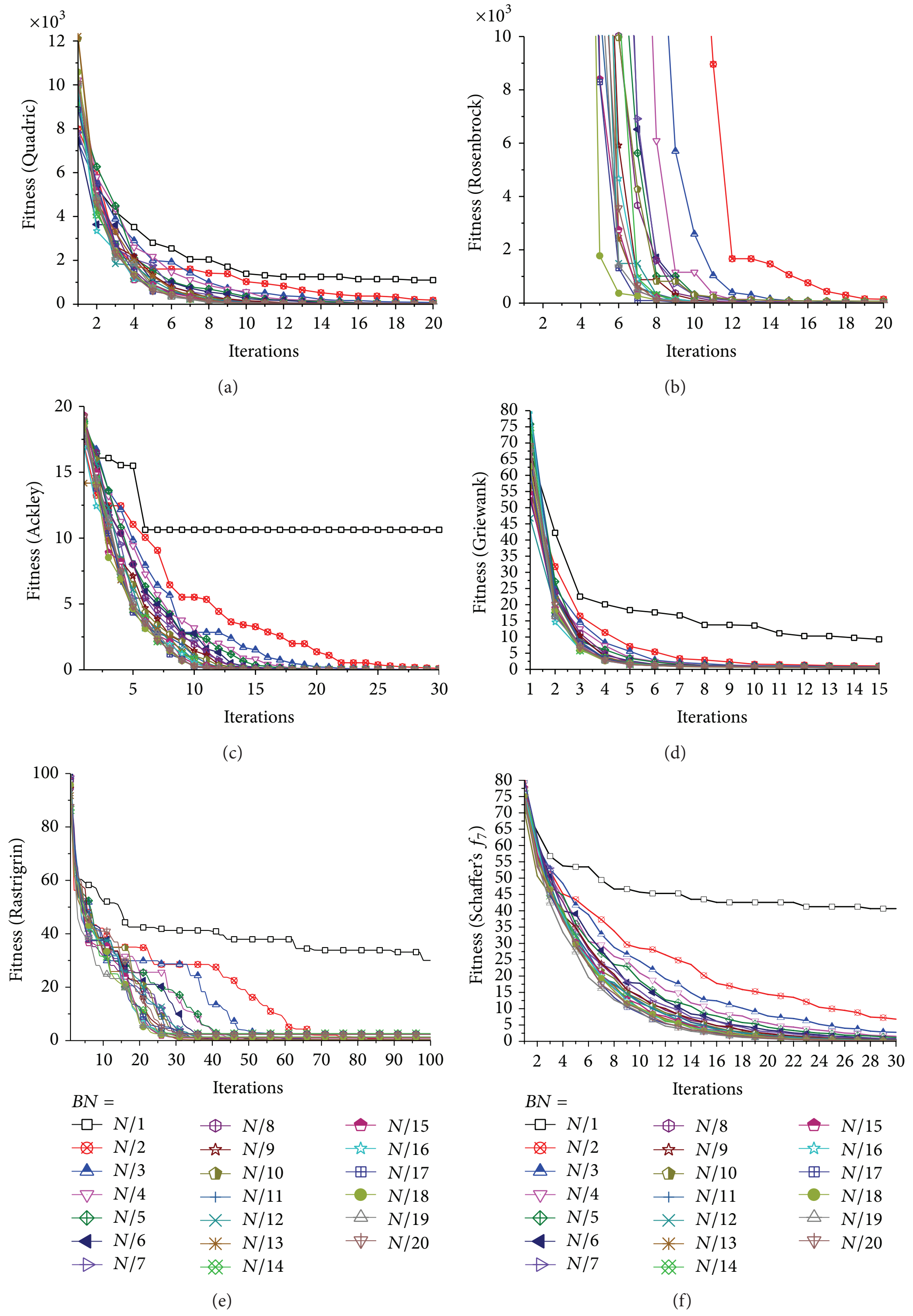


Figure 9: The convergent results under different *BN*.

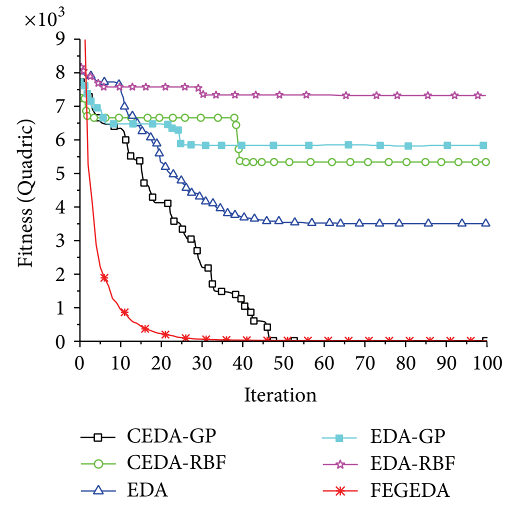
In the VC system of PMSM, $i_d = 0$. Therefore, the state space equation (22) is described as

$$\dot{i}_q = \frac{1}{L_q}\left(u_q - R_s i_q - \psi_f w_e\right),$$
$$\dot{w}_e = \frac{1.5 p_n^2 \psi_f i_q - p_n T_m - B w_e}{J}. \tag{23}$$

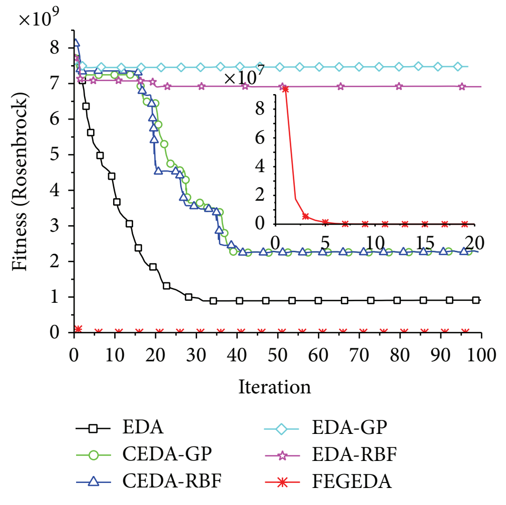
4.2. PID Controller. The continuous form of a PID controller, with input e and output u, is shown as follows:

$$u(t) = K_p e(t) + K_i \int e(t) + K_d \dot{e}(t), \tag{24}$$

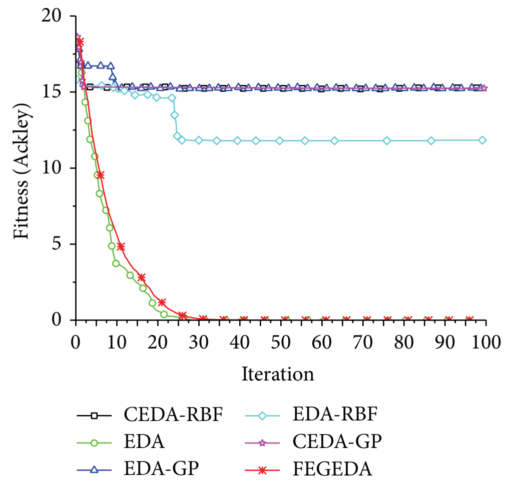
where K_p is the proportional gain, K_i is the integral gains, and K_d is the derivative gains.



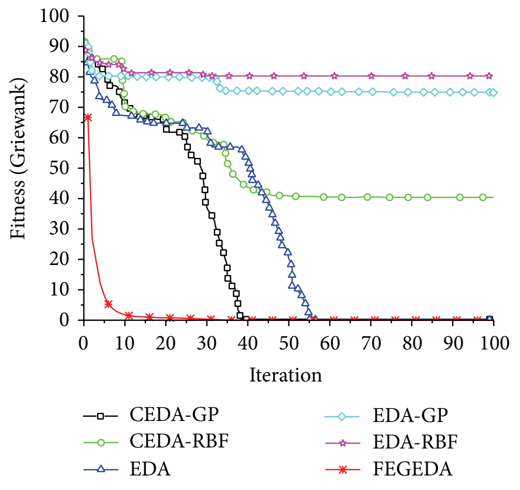
(a)



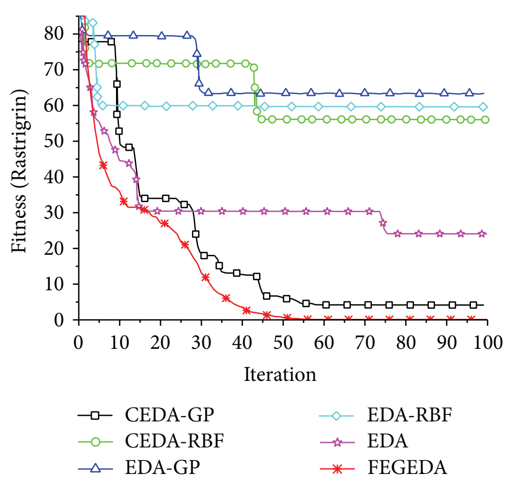
(b)



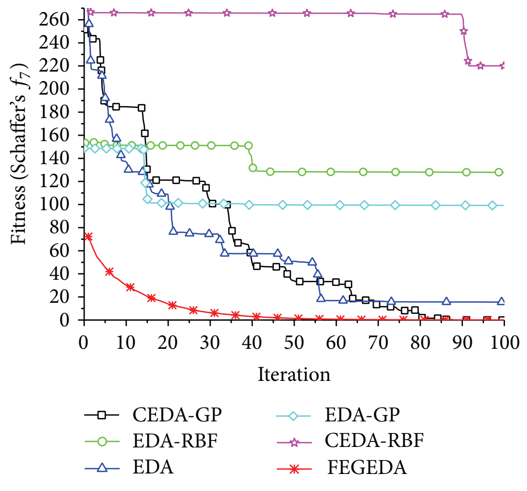
(c)



(d)



(e)



(f)

FIGURE 10: Comparisons of convergent results.

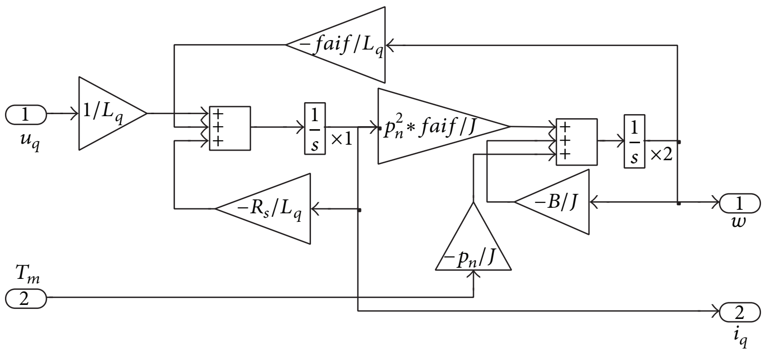


FIGURE 11: MATLAB/Simulink model of PMSM.

There are two types of discrete PID by discretization of continuous PID. The position-type discrete PID is described as

$$u(k) = K_p e(k) + K_i \sum_{j=0}^{k} T_s e(k) + \frac{K_d}{T_s}(e(k) - e(k-1)), \tag{25}$$

where $u(k)$ is the controller output, $e(k)$ is the error. In practical system control, the integral part is not flexible. Therefore, another velocity-type discrete PID is described as

$$\begin{aligned} &\Delta u(k) \\ &= K_p \Delta e(k) + K_i T_s e(k) + \frac{K_d}{T_s}(\Delta e(k) - \Delta e(k-1)), \quad (26) \\ &\qquad \Delta e(k) = e(k) - e(k-1), \end{aligned}$$

where T_s is the sample time. For velocity-type PID, we do not need to calculate the integral part, and the controller output is the increment of PID. Therefore, it is often used in practical system control.

Aggregation function is a conventional method, which can convert a multiobjective problem to a single-objective problem. Consider

$$\text{fitness} = \sum_{i=1}^{n} w_i f_i, \tag{27}$$

where fitness is the summation of fitness, w_i is the weight of ith objective, and f_i is the fitness value of ith objective.

In the optimization process, the objective is to evaluate the performance of PIDs. Thus, for PID, the fitness function is written as [39]

$$\begin{aligned} f_1 &= \int_0^\infty |e(t)|\,dt \\ f_2 &= \int_0^\infty u^2(t)\,dt \\ f_3 &= t_r, \end{aligned} \tag{28}$$

where $e(t)$ is the system error, $u(t)$ is the control output, and t_r is the rising time.

To avoid overshoot, a penalty value is adopted in the fitness function. That is, once overshoot occurs, the value of overshoot is added to the fitness function. Hence, the penalty function is written as

$$f_4 = \begin{cases} \int_0^\infty (y(t) - y(t-1))\,dt & \text{if } e(t) < 0 \\ 0 & \text{if } e(t) \geq 0, \end{cases} \tag{29}$$

where $y(t)$ is the control output.

Making use of the aggression function, the fitness function is constructed as follows:

$$f = w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4, \tag{30}$$

where w_1, w_2, w_3, and w_4 are the weight coefficients, and $w_4 \gg w_1$.

4.3. PID Controller Optimization Based on FEGEDA. According to state space equation (6), we can build the state space model of PMSM in MATLAB/Simulink as Figure 11. The parameters of PMSM are that R_s is 0.9664, L_q is 0.00621, P_n is 4, J is 0.00033, B is 0.0001619, and ψ_f is 0.09382 according to motor.

The component of PMSM is encapsulated into a module. A speed controller added to the speed closed loop. Figure 12 is the diagram of PMSM control system. The "simouterror," "simoutui," and "simout" units are used to record the simulation data for optimization.

In order to testify the algorithm, GA and traditional PID are selected to compare against FEGEDA. w_1, w_2, w_3, and w_4 of f_i are set according to the requirement of control system. w_1 is corresponding to the control variable of error, w_2 is a weight coefficient of controlled variable, w_3 is for the control variable of rising time, and w_4 is the penalty of overshoot. If we want a system without overshoot and have a small rising time, w_1, w_3, and w_4 will be set bigger, and w_2 is smaller. If the controlled variable is limited, w_2 will be set bigger. Therefore, these parameters can be set according to the practical requirement. In the test, w_1 is 1, w_2 is 0.1, w_3 is 2, and w_4 is 200. The parameters of GA are that the population size is 30, crossover probability is 0.9, and mutation probability is adaptive to individual fitness. The variable domain of K_p is $[0, 20]$ and K_i and K_d are $[0, 1]$. The iteration number is

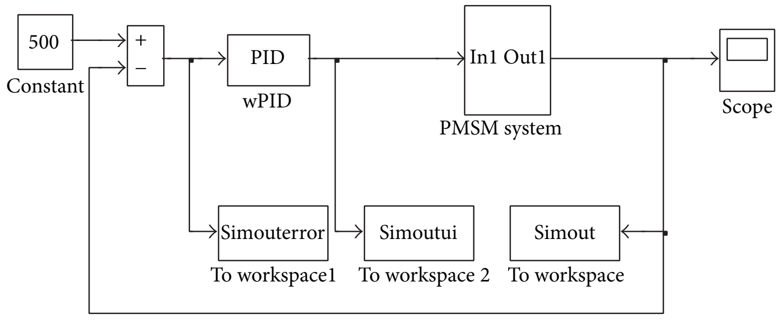


Figure 12: The diagram of PMSM control system.

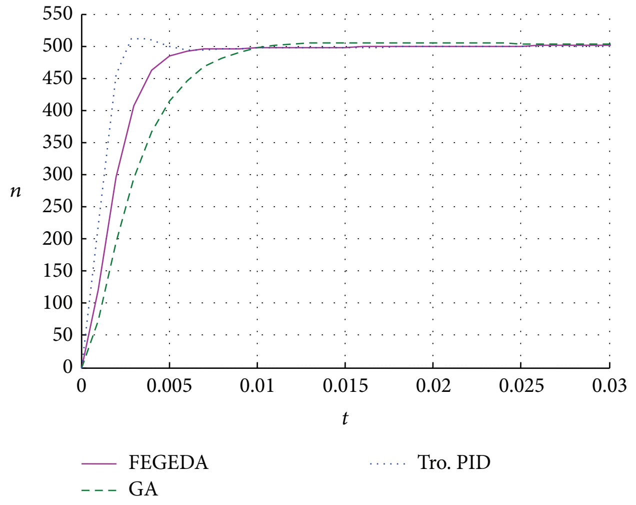


Figure 13: The system response of PMSM with different PIDs.

50. The optimal results are shown in Figure 13. Although the optimal result of traditional PID has shorter response time, the overshoot is bigger. The optimal result of GA has no overshoot, but the system response is slower. Therefore, the performance of FEGEDA is promising.

5. Conclusions

We studied the estimation of distribution algorithm in this paper and proposed a fast elitism Gaussian EDA for continuous optimization. Every dimension of individuals is represented by means and standard deviations of Gaussian distribution. These parameters are estimated using maximum likelihood technique by fast learning rule. Then the new population is generated by sampling and elitism strategy. The elitism strategy improves the convergent performance of the algorithm. In the one-dimensional test, we exhibit the optimization process and probability models learning process clearly. In the two-dimensional and higher dimensional problems, we compare the FEGEDA with danger immune algorithm and other EDAs, and the FEGEDA exhibits a good performance. Although the performance of FEGEDA is promising, further studies are still recommended, for instance, how to increase the diversity of population under fast convergence rate.

Conflict of Interests

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

Acknowledgment

This work is supported by the National Natural Science Foundation of China (under Grant 61174044).

References

[1] B. A. H. Al-Sarray and R. A. D. Al-Dabbagh, “Variants of hybrid genetic algorithms for optimizing likelihood ARMA model function and many of problems,” in *Evolutionary Algorithms*, pp. 219–246, InTech, Winchester, UK, 2011.

[2] S. Das, S. Maity, B. Qu, and P. N. Suganthan, “Real-parameter evolutionary multimodal optimization—a survey of the state-of-the-art,” *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 71–88, 2011.

[3] 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.

[4] S. Forrest, L. Allen, A. S. Perelson, and R. Cherukuri, “Self-nonself discrimination in a computer,” in *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pp. 202–212, May 1994.

[5] X. Cao, H. Qiao, and Y. Xu, “Negative selection based immune optimization,” *Advances in Engineering Software*, vol. 38, no. 10, pp. 649–656, 2007.

[6] G. Wu, W. Pedrycz, M. Ma, D. Qiu, H. Li, and J. Liu, “A particle swarm optimization variant with an inner variable learning strategy,” *The Scientific World Journal*, vol. 2014, Article ID 713490, 15 pages, 2014.

[7] R. Liu, C. Ma, W. Ma, and Y. Li, “A multipopulation PSO based memetic algorithm for permutation flow shop scheduling,” *The Scientific World Journal*, vol. 2013, Article ID 387194, 11 pages, 2013.

[8] N. Belgasmi, L. Ben Said, and K. Ghédira, “Evolutionary multiobjective optimization of the multi-location transshipment problem,” *Operational Research*, vol. 8, no. 2, pp. 167–183, 2008.

[9] M. Pelikan, D. E. Goldberg, and F. G. Lobo, "A survey of optimization by building and using probabilistic models," *Computational Optimization and Applications*, vol. 21, no. 1, pp. 5–20, 2002.

[10] H. Miuhlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, pp. 178–187, London, UK, 1996.

[11] 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.

[12] U. Aickelin, E. K. Burke, and J. Li, "An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering," *Journal of the Operational Research Society*, vol. 58, no. 12, pp. 1574–1585, 2007.

[13] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111–128, 2011.

[14] S. Shahraki and M. R. A. Tutunchy, "Continuous Gaussian estimation of distribution algorithm," in *Synergies of Soft Computing and Statistics for Intelligent Data Analysis*, pp. 211–218, Springer, Berlin, Germany, 2013.

[15] A. Zhou, B. Qu, H. Li, S. Zhao, P. N. Suganthan, and Q. Zhangd, "Multiobjective evolutionary algorithms: a survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[16] S. Baluja, "Population-based incremental learning. A method for integrating genetic search based function optimization and competitive learning," Technical Report CMU-CS-94-163, DTIC Document, Pittsburgh, Pa, USA, 1994.

[17] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 287–297, 1999.

[18] S. Rudlof and M. Köppen, "Stochastic hill climbing with learning by vectors of normal distributions," in *Proceedings of the 1st On-line Workshop on Soft Computing*, pp. 60–70, Nagoya, Japan, 1996.

[19] M. E. L. Sebag and A. Ducoulombier, "Extending population-based incremental learning to continuous search spaces," in *Proceedings of the 5th Parallel Problem Solving from Nature (PPSN '89)*, pp. 418–427, Amsterdam, The Netherlands, 1998.

[20] H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga, "Regularized continuous estimation of distribution algorithms," *Applied Soft Computing*, vol. 13, no. 5, pp. 2412–2432, 2013.

[21] N. Luo, F. Qian, L. Zhao, and W. Zhong, "Gaussian process assisted coevolutionary estimation of distribution algorithm for computationally expensive problems," *Journal of Central South University of Technology*, vol. 19, no. 2, pp. 443–452, 2012.

[22] H. Li, Y. Hong, and S. Kwong, "Subspace estimation of distribution algorithms: to perturb part of all variables in estimation of distribution algorithms," *Applied Soft Computing Journal*, vol. 11, no. 3, pp. 2974–2989, 2011.

[23] M. Nakao, T. Hiroyasu, M. Miki, H. Yokouchi, and M. Yoshimi, "Real-coded estimation of distribution algorithm by using probabilistic models with multiple learning rates," in *Proceedings of the 11th International Conference on Computational Science (ICCS '11)*, vol. 4, pp. 1244–1251, June 2011.

[24] P. A. N. Bosman and D. Thierens, "Numerical optimization with real-valued estimation-of-distribution algorithms," *Studies in Computational Intelligence*, vol. 33, pp. 91–120, 2007.

[25] L. Martí, J. Garca, A. Berlanga, C. A. Coello Coello, and J. M. Molina, "MB-GNG: addressing drawbacks in multi-objective optimization estimation of distribution algorithms," *Operations Research Letters*, vol. 39, no. 2, pp. 150–154, 2011.

[26] P. A. Bosman and J. Grahl, "Matching inductive search bias and problem structure in continuous Estimation-of-Distribution Algorithms," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1246–1264, 2008.

[27] M. Gallagher, M. Frean, and T. Downs, "Real-valued evolutionary optimization using a flexible probability density estimator," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 840–846, Orlando, Fla, USA, 1999.

[28] C. W. Ahn and R. S. Ramakrishna, "Elitism-based compact genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 367–385, 2003.

[29] R. C. Purshouse and P. J. Fleming, "Why use elitism and sharing in a multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 520–527, New York, NY, USA, 2002.

[30] I. J. Leno, S. S. Sankar, M. V. Raj, and S. G. Ponnambalam, "An elitist strategy genetic algorithm for integrated layout design," *The International Journal of Advanced Manufacturing Technology*, vol. 66, no. 9–12, pp. 1573–1589, 2013.

[31] L. Lasdon, A. Duarte, F. Glover, M. Laguna, and R. Martí, "Adaptive memory programming for constrained global optimization," *Computers and Operations Research*, vol. 37, no. 8, pp. 1500–1509, 2010.

[32] H. Muhlenbein, "Convergence theorems of estimation of distribution algorithms," in *Markov Networks in Evolutionary Computation*, pp. 91–108, Springer, Berlin, Germany, 2012.

[33] H. Miuhlenbein, J. U. R. Bendisch, and H. Voigt, "From recombination of genes to the estimation of distributions II. Continuous parameters," in *Proceedings of the 4th Parallel Problem Solving from Nature (PPSN '96)*, pp. 188–197, Springer, Berlin, Germany, 1996.

[34] Q. Zhang and H. Mühlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 127–136, 2004.

[35] Q. Xu, S. Wang, L. Zhang, and Y. Liang, "A novel chaos danger model immune algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 11, pp. 3046–3060, 2013.

[36] L. Huang, N. Wang, and J. Zhao, "Multiobjective optimization for controller design," *Acta Automatica Sinica*, vol. 34, no. 4, pp. 472–477, 2008.

[37] W. Wojsznis, A. Mehta, P. Wojsznis, D. Thiele, and T. Blevins, "Multi-objective optimization for model predictive control," *ISA Transactions*, vol. 46, no. 3, pp. 351–361, 2007.

[38] G. Chen, J. Kang, and J. Zhao, "Numeric analysis and simulation of space vector pulse width modulation," *Advances in Engineering Software*, vol. 65, pp. 60–65, 2013.

[39] J. Zhang, J. Zhuang, H. Du, and S. Wang, "Self-organizing genetic algorithm based tuning of PID controllers," *Information Sciences*, vol. 179, no. 7, pp. 1007–1018, 2009.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 364179, 21 pages
http://dx.doi.org/10.1155/2014/364179

Research Article

Improving Vector Evaluated Particle Swarm Optimisation Using Multiple Nondominated Leaders

Kian Sheng Lim,[1] Salinda Buyamin,[1] Anita Ahmad,[1] Mohd Ibrahim Shapiai,[1] Faradila Naim,[2] Marizan Mubin,[3] and Dong Hwa Kim[4]

[1] *Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia*
[2] *Faculty of Electrical & Electronic Engineering, Universiti Malaysia Pahang, 26600 Pekan, Malaysia*
[3] *Department of Electrical Engineering, Faculty of Engineering, Universiti Malaya, 50603 Kuala Lumpur, Malaysia*
[4] *Department of Instrumentation and Control Engineering, Hanbat National University, Daejeon 305-719, Republic of Korea*

Correspondence should be addressed to Faradila Naim; faradilan@ump.edu.my

Received 10 February 2014; Accepted 9 March 2014; Published 27 April 2014

Academic Editors: P. Agarwal, V. Bhatnagar, and Y. Zhang

Copyright © 2014 Kian Sheng Lim 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 vector evaluated particle swarm optimisation (VEPSO) algorithm was previously improved by incorporating nondominated solutions for solving multiobjective optimisation problems. However, the obtained solutions did not converge close to the Pareto front and also did not distribute evenly over the Pareto front. Therefore, in this study, the concept of multiple nondominated leaders is incorporated to further improve the VEPSO algorithm. Hence, multiple nondominated solutions that are best at a respective objective function are used to guide particles in finding optimal solutions. The improved VEPSO is measured by the number of nondominated solutions found, generational distance, spread, and hypervolume. The results from the conducted experiments show that the proposed VEPSO significantly improved the existing VEPSO algorithms.

1. Introduction

Multiobjective optimisation (MOO) problems involve the simultaneous minimisation/maximisation of multiple objective functions, which usually conflict with each other. Due to the conflict between objective functions, a single solution could not satisfy all objective functions. Hence, MOO problem usually results in a set of tradeoffs or nondominated solutions. The vector evaluated particle swarm optimisation (VEPSO) [1] algorithm has been widely used to solve MOO problems [2–7]. As an example, VEPSO algorithm has been implemented in solving DNA sequence problem by minimising four objective functions, namely, H_{measure}, similarity, continuity, and hairpin, and two constraints, namely, melting temperature and GC_{content} [7]. Compared to DNA sequence design using binary particle swarm optimization which produces single set of DNA sequences [8], VEPSO is able to generate several sets of good DNA sequences which fulfil the four objective functions and two constraints.

The VEPSO algorithm is adapted from the vector evaluated genetic algorithm (VEGA) [9], in which each swarm optimises one objective function by using the best solution from another swarm as a guidance. However, the VEPSO suffers from performance drawback. Therefore, it is improved by redefining the selection of the guidance from nondominated solution, known as VEPSOnds [10]. Although VEPSOnds has shown better performance than conventional VEPSO, the VEPSOnds suffers from weak performance in terms of lacking solution distributions and convergence to the true Pareto front.

Other than VEPSOnds, there are various MOO algorithms which used nondominated solution to guide particle in finding the optimum solutions for MOO problem. For example, in Multiobjective particle swarm optimisation (MOPSO) algorithm [11, 12], all nondominated solutions are separated into groups according to their location in the objective space. A guiding solution for each particle is then randomly selected from the group containing the fewest solutions. Besides, in nondominated sorting PSO (NSPSO) algorithm [13], which uses the main mechanism of the nondominated sorting fenetic algorithm-II [14], each particle is guided by a nondominated solution that is randomly selected using the niche count and the nearest neighbour density estimator. A nondominated solution is selected based on binary tournament selection for the purpose of guiding the other particles in the optimised MOPSO (OMOPSO) algorithm [15]. Additionally, Abido [16] introduces the use of two nondominated solutions, which are called the local set and the global set. The guide is selected based on the nearest distance in objective space between each particle and each member of the nondominated solution of both sets.

Noticeably, most particle swarm optimisation- (PSO-) based MOO algorithms, including conventional VEPSO and VEPSOnds, only use one solution as the particle guide. In particular, in VEPSOnds, particles from a swarm will be guided by the nondominated solution which has the best fitness at one objective function. Thus, the particles may guide the searching with limited information about the other objective functions during the optimisation process. Therefore, VEPSOnds can be further improved by using more than one nondominated solution as particle guide. In this context, this improved VEPSO algorithm will use the best solution from all swarms as guidance during the optimisation process.

The next section of this paper explains the particle swarm optimisation (PSO), the conventional VEPSO, VEPSOnds algorithm, and the proposed VEPSO algorithms. The following section presents the experimental work and the description of the benchmark test problems and performance measures and the discussion of the results. The final section concludes the proposed technique and discusses few possible future works.

2. Multiobjective Optimization

For explanation, consider a minimization problem

$$\begin{aligned} &\text{minimize fitness function,} \\ &\qquad \vec{F}(\vec{x}) = \{f_i(\vec{x}), i = 1, 2, \ldots, M\} \\ &\text{subject to} = \begin{cases} g_j(\vec{x}) \leq 0, & j = 1, 2, \ldots, p \\ h_k(\vec{x}) = 0, & k = 1, 2, \ldots, q, \end{cases} \end{aligned} \tag{1}$$

where $\vec{x} = \{x_1, x_2, \ldots, x_n\}$ is the decision variable vector which represents the possible solution, M is the number of objectives, and $f_i \in \mathfrak{R}^n \rightarrow \mathfrak{R}$ is the objective function. $\{g_j, h_k\} \in \mathfrak{R}^n \rightarrow \mathfrak{R}$ are the inequality and equality constraint function, respectively. The Pareto optimality concept is defined as follows.

Definition 1. Given $\{\overrightarrow{F^a}, \overrightarrow{F^b}\} \in \mathfrak{R}^m$ as two vectors, $\overrightarrow{F^a}$ *dominates* $\overrightarrow{F^b}$ (denote as $\overrightarrow{F^a} \prec \overrightarrow{F^b}$) if and only if $f_i^a \leq f_i^b$ for $i = 1, 2, \ldots, m$ and $f_i^a < f_i^b$ for at least once. Dominance relation of $\overrightarrow{F^a} \prec \overrightarrow{F^b}$ and $\overrightarrow{F^a} \prec \overrightarrow{F^c}$ can be illustrated as the labelled circles in Figure 1 for a two-objective problem.

Definition 2. A decision variable vector $\overrightarrow{x^a}$ is a *nondominated solution* when there is no other solution $\overrightarrow{x^b}$ such that $\vec{F}(\overrightarrow{x^a}) \prec \vec{F}(\overrightarrow{x^b})$. Nondominated solution is also known as Pareto optimal solution.

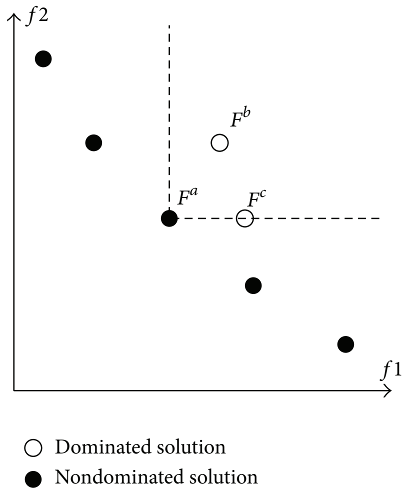


Figure 1: Dominance relation for two-objective problem.

Definition 3. The set of nondominated solutions of a MOO problem is known as *Pareto optimal set*, $\mathscr{P}$.

Definition 4. The set of objective vectors with respect to $\mathscr{P}$ is known as the *Pareto front*, $\mathscr{PF} = \{\vec{F}(\vec{x}) \in \mathfrak{R}^m \mid \vec{x} \in \mathscr{P}\}$. $\mathscr{PF}$ for a two-objective problem is illustrated as the black circles in Figure 1.

The motivation of MOO is to find as many nondominated solutions as possible according to the objective functions and constraints. However, it is possible to have different solutions which map to the same fitness value in objective space. Therefore, it will be more challenging to find more nondominated solutions.

3. Particle Swarm Optimisation

3.1. Original Particle Swarm Optimisation Algorithm. Particle swarm optimisation (PSO) is a population-based stochastic optimisation algorithm introduced by Kennedy and Eberhart [17]. This algorithm finds an optimal solution using a method inspired by the social behaviour of birds flocking and fish schooling. In the PSO algorithm, an individual is known as a particle, and it holds the possible solution to the optimisation problem, given its position. A particle explores the search space, looking for a better solution with respect to the objective functions defined by the optimisation problem. The search process requires the particle to compare its current position with the best positions that it and the whole swarm have found, so that all particles collaborate with each other.

The PSO algorithm is shown in Algorithm 1. Consider a minimisation problem in which a swarm of I particles are flying around in an N-dimensional search space, each with a position p_n^i $(i = 1, 2, \ldots, I; n = 1, 2, \ldots, N)$ representing the possible solution. At initialization stage, all particles are randomly positioned in the search space with random

```
begin
    Initialise position & velocity;
    Evaluate objective;
    Initialise pBest;
    Initialise gBest (2);
    while $i \leq i_{\max}$ do
        Update velocity (3);
        Update position (4);
        Evaluate objective;
        Update pBest;
        Update gBest (2);
        $i$ ++;
    end
end
```

Algorithm 1: The PSO algorithm.

velocity, $v_n^i(t)$. Subsequently, the objective fitness $\overrightarrow{F^i}(t)$ of each particle is evaluated based on the objective function for $p^i(t)$. After that, the particle's best position, $p\text{Best}^i(t)$, is set to its initial position. Additionally, the swarm's best position, $g\text{Best}(t)$, is the best $p\text{Best}^i(t)$ among all particles, as in (2), where S is the swarm of particles

$$g\text{Best} = \left\{p\text{Best}^i \in S \mid f\left(p\text{Best}^i\right) = \min f\left(\forall p\text{Best}^i \in S\right)\right\}. \tag{2}$$

In the search process, the algorithm will iterate until the maximum number of iterations is reached. Within an iteration, the velocity and position of each particle are updated using (3) and (4), respectively,

$$\begin{aligned} v_n^i(t+1) = \chi \left[\omega v_n^i(t) + c_1 r_1 \left(p\text{Best}_n^i - p_n^i(t)\right)\right. \\ \left. + c_2 r_2 \left(g\text{Best}_n - p_n^i(t)\right)\right], \end{aligned} \tag{3}$$

$$p_n^i(t+1) = p_n^i(t) + v_n^i(t+1), \tag{4}$$

where χ is the constriction factor and ω is the inertia weight. The r_1 and r_2 are both random numbers ranging from zero to one. The c_1 and c_2 are the cognitive and social constants, respectively, which control the attraction of the $p\text{Best}^i(t)$ and $g\text{Best}(t)$. Then, the $\vec{F^i}(t)$ for each particle is evaluated again. After updating the fitness, the new position of particle i is compared with $p\text{Best}^i(t)$, and the more optimal of the two is saved as $p\text{Best}^i(t)$. Next, the $g\text{Best}(t)$ is updated as well with the best among all $p\text{Best}^i(t)$, as in (2). When the search process ended, the $g\text{Best}(t)$ will then represent the best solution found for the problem by this algorithm.

3.2. Vector Evaluated Particle Swarm Optimisation Algorithm. The VEPSO algorithm, introduced by Parsopóulos and Vrahatis [1], uses the multiswarms concept from the VEGA algorithm [9]. Each swarm optimises one objective function using the $g\text{Best}(t)$ from another swarm. In the VEPSO algorithm, the $p\text{Best}^i(t)$ which has the best fitness with respect to the mth objective is the $g\text{Best}(t)$ for the mth swarm, as in (5)

$$\begin{aligned} g\text{Best}^m = \left\{p\text{Best}^i \in S^m \mid f_m\left(p\text{Best}^i\right)\right. \\ \left. = \min f_m\left(\forall p\text{Best}^i \in S^m\right)\right\}. \end{aligned} \tag{5}$$

The flow of the VEPSO algorithm is given as in Algorithm 2. For problem with M objective functions, VEPSO algorithm is similar to that of the PSO but some processes are repeated for all M-swarm and nondominated solutions are recorded in an archive. However, the velocity update is reformulated and it is given in (6). Note that the particles in the mth swarm will fly using $g\text{Best}^k(t)$ where k is defined by (7). The sharing of $g\text{Best}(t)$ between swarms is illustrated in Figure 2:

$$\begin{aligned} v_n^m i(t+1) = \chi \left[\omega v_n^{mi}(t) + c_1 r_1 \left(p\text{Best}_n^{mi} - p_n^{mi}(t)\right)\right. \\ \left. + c_2 r_2 \left(g\text{Best}_n^k - p_n^{mi}(t)\right)\right] \end{aligned} \tag{6}$$

$$k = \begin{cases} M, & m = 1 \\ m-1, & \text{otherwise.} \end{cases} \tag{7}$$

The nondominated solutions are recorded in an archive after the objective functions are evaluated. In the recording process, the fitness $\overrightarrow{F^i}(t)$ of each particle is compared to all others, before it is compared to the nondominated solutions in the archive, using the *Pareto optimality* criterion, so that the archive only contains nondominated solutions. At the end of the computation, all nondominated solutions are the possible solutions to the problem.

3.3. The Improved VEPSO Algorithm by Incorporating Nondominated Solutions. In the search process of conventional VEPSO, as in Figure 3(a), particles from a swarm are optimised using the $g\text{Best}^m(t)$ from another swarm that has the best fitness at the objective function optimised by the other swarm. However, based on the velocity update of conventional VEPSO in (5), the $g\text{Best}^m(t)$ is not updated

```
begin
    Initilise position & velocity for all $M$-swarm;
    Evaluate objective for all $M$-swarm;
    Initialise archive;
    Initialise $p$Best for all $M$-swarm;
    Initialise $g$Best (5) for all $M$-swarm;
    while $i \le i_{\max}$ do
        Update velocity (6) & (7) for all $M$-swarm;
        Update position (4) for all $M$-swarm;
        Evaluate objective for all $M$-swarm;
        Update archive;
        Update $p$Best for all $M$-swarm;
        Update $g$Best (5) for all $M$-swarm;
        $i$ ++;
    end
end
```

Algorithm 2: The VEPSO algorithm.

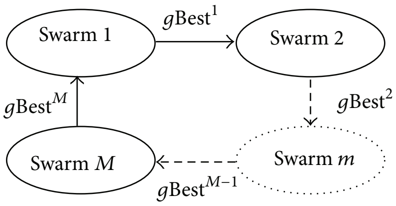


Figure 2: The best position found by the swarms, shared between all swarms.

unless there is a $p\text{Best}^{mi}(t)$ that has better fitness than that at the m-objective. Consequently, in a two-objective MOO problem, the $g\text{Best}^1(t)$ of the first swarm is not updated when particle in the first swarm has found a solution with equal fitness at the first objective and better fitness at the second objective. Thus, particles from the second swarm will be guided toward the $g\text{Best}^1(t)$.

Due to this limitation, Lim et al. [10] have introduced an improved VEPSO algorithm by incorporating nondominated solutions (VEPSOnds). In VEPSOnds, as specified by (8), the $g\text{Best}^m(t)$ is still the solution with best fitness at m-objective function but is selected from the set of nondominated solutions and not from all $p\text{Best}^{mi}(t)$ of the m-swarm

$$g\text{Best}^m = \{X \in \mathscr{P} \mid f_m(X) = \min f_m(\forall X \in \mathscr{P})\}, \quad (8)$$

where X is a nondominated solution and $\mathscr{P}$ is the set of nondominated solutions in the archive.

This improvement is illustrated in Figure 3(b) where the $g\text{Best}^m(t)$ is always the best solution with respect to m-objective function because the other objective functions are considered as well. Hence, particles from the second swarm can converge faster towards the $g\text{Best}^1(t)$, which is a nondominated solution. As a result, better quality of Pareto front is obtained. From an algorithm perspective, the VEPSOnds is similar to the conventional VEPSO except that (5) in Algorithm 2 is replaced with (8).

3.4. The Improved VEPSO Using Multiple nondominated Leader. Based on the results of VEPSOnds [10], this algorithm suffers weak performance in obtaining solutions that has a weak diversity performance where the solution distributions along the Pareto front are not well distributed. Besides, in comparison to other state-of-the-art MOO algorithm, the VEPSOnds also has a problem in convergence where the obtained solution is far distant from the Pareto front. This weak performance could possibly be caused by the fact that particles in each swarm are guided by one $g\text{Best}^m(t)$ only so the obtained solutions do not well diverse to the other objective functions.

Thus, the use of nondominated solutions to enhance the VEPSO algorithm can be further improved by the use of multileader concept in this work. According to (6), which is the velocity equation of the VEPSO, the particles of a swarm are guided by its $p\text{Best}(t)$ and another swarm's $g\text{Best}(t)$. For example, as shown in Figure 4(a), the particles from the second swarm optimise the second objective function using $g\text{Best}^1(t)$ only, which may not be the solution that has the best fitness with respect to the second objective function. Thus, this original mechanism of VEPSO may limit the convergence rate of the algorithm. Therefore, an improved VEPSO algorithm is proposed by including $g\text{Best}^2(t)$ as additional guidance to optimise both objective functions, as shown in Figure 4(b).

Hence, the general velocity equation of this improved VEPSO is formulated as in (9)

$$v_n^m i(t+1) = \chi \left[\omega v_n^{mi}(t) + c_1 r_1 \left(p\text{Best}_n^{mi} - p_n^{mi}(t) \right) + \sum_{q=1}^{M} c_2^q r_2^q \left(g\text{Best}_n^q - p_n^{mi}(t) \right) \right], \quad (9)$$

where for each q, c_2^q, and r_2^q are independent constant and random values, respectively. In addition, from (9), as compared to the improved VEPSO at previous section, the particles will search toward the nondominated solutions which located at different end of the Pareto front. Therefore,

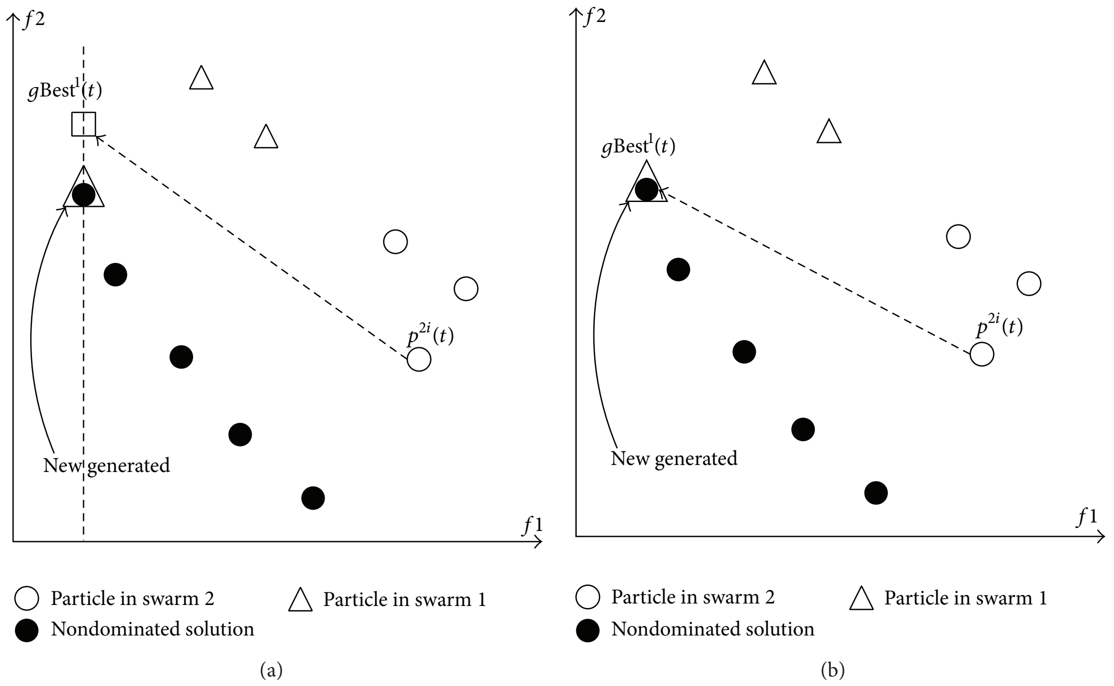


FIGURE 3: (a) Particles guided by the best solution from the other swarm (b) Particles guided by a nondominated solution with respect to another swarm.

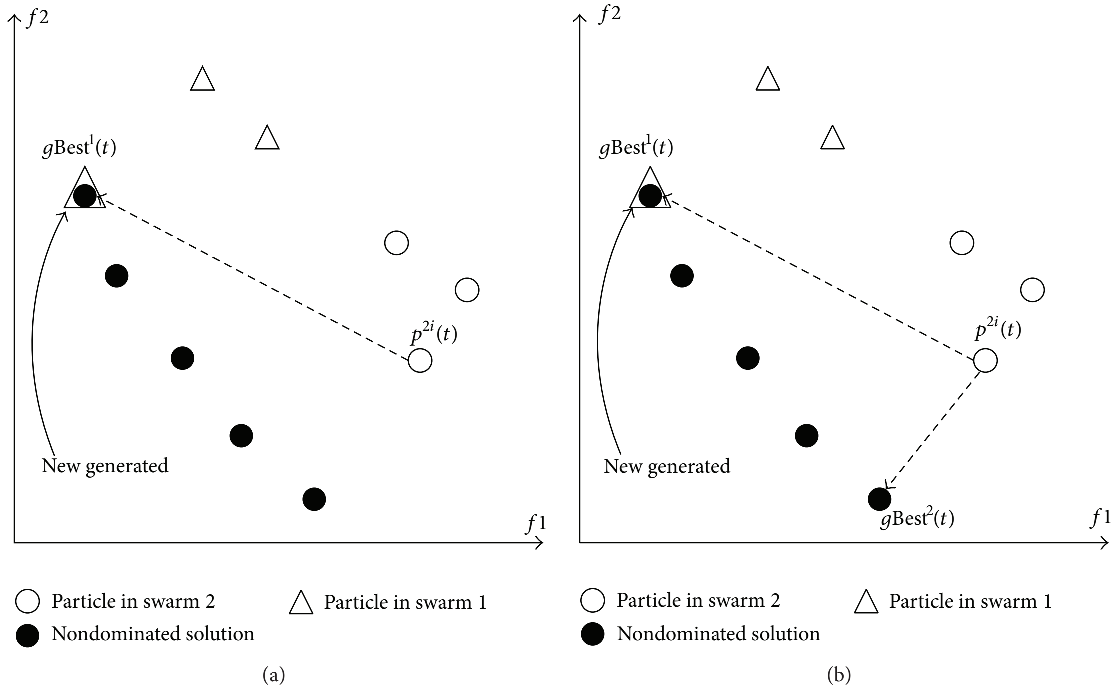


FIGURE 4: (a) A particle is guided based on $g\text{Best}^1(t)$. (b) A particle is guided based on $g\text{Best}^1(t)$ and $g\text{Best}^2(t)$.

the diversity performance of the algorithm is expected to be better as the search area is wider, rather than a single point.

Because the improved VEPSO algorithm uses multiple nondominated solutions as particle guides, or leaders, this algorithm is called VEPSO using multiple nondominated leaders (VEPSOml). Also, a polynomial mutation mechanism from NSGA-II [14] is used to modify particle positions at some probability. By mutating the position of some particles out of the locally optimal solution, this mechanism broadens the search for a globally optimal solution. In this study, the position of one out of every fifteen particles is mutated in the algorithm. Therefore, the complete VEPSO algorithm using multiple nondominated leaders is shown in Algorithm 3.

```
begin
    Initialise position & velocity for all $M$-swarm;
    Evaluate objective for all $M$-swarm;
    Initialise archive;
    Initialise pBest for all $M$-swarm;
    Initialise gBest (8) for all $M$-swarm;
    while $i \le i_{\max}$ do
        Update velocity (9) for all $M$-swarm;
        Update position (4) for all $M$-swarm;
        Mutate position for all $M$-swarm;
        Evaluate objective for all $M$-swarm;
        Update archive;
        Update pBest for all $M$-swarm;
        Update gBest (8) for all $M$-swarm;
        $i$ ++;
    end
end
```

Algorithm 3: The VEPSO algorithm using multinondominated leaders.

4. Experiment

4.1. Performance Measure. MOO algorithms face difficulty in converging to and distributing the nondominated solutions over the true Pareto front, $\mathcal{PF}_t$. Hence, the algorithm performance is measured by the quality of the obtained Pareto front, $\mathcal{PF}_o$. Several performance measures are used for comparison to highlight any improvement in the proposed algorithm.

The number of solutions (NS) measured will calculate the total number of nondominated solutions found by an algorithm. The best algorithm, by this measure, gives the most nondominated solutions. A more advanced measure uses the generalized distance (GD) [18], which is a popular measure of convergence [14]. This performance measure first evaluates the average distance between the true Pareto front and the one obtained by the algorithm. Equation (10) is used to compute the average distance, with a smaller value corresponding to a better performance. Then, the minimum distance of a nondominated solution from the true Pareto front is calculated using (11)

$$\mathrm{GD} = \frac{\left(\sum_{q=1}^{\|\mathcal{PF}_o\|} d_q^M\right)^{1/M}}{\|\mathcal{PF}_o\|} \tag{10}$$

$$d_q = \min_{1 \le g \le \|\mathcal{PF}_t\|} \sqrt{\sum_{m=1}^{M} \left(\mathcal{PF}_{o\,q}^{m} - \mathcal{PF}_{t\,g}^{m}\right)^2}. \tag{11}$$

In addition, SP [14] is a commonly used measure of the diversity performance, or the distribution of nondominated solutions [14] is used. Equations (12), (13), and (14) evaluate the diversity performance, as measured by SP. The d_f and d_l are the Euclidean distances between the boundary solution and the nondominated solutions returned by the algorithm and the true Pareto front, respectively. The Euclidean distance between two solutions can be calculated using (13). Thus, SP actually measures the average distance of one solution and of the next solution to all nondominated solutions returned by the algorithm as well as two boundary solutions in the true Pareto front. Hence, it is desirable that the Pareto front returned by the algorithm produces a small SP:

$$\text{Spread} = \frac{d_f + d_l + \sum_{q=1}^{\|\mathcal{PF}_o\|-1} \left|d_q - \overline{d}\right|}{d_f + d_l + \overline{d}\left(\|\mathcal{PF}_o\| - 1\right)}, \tag{12}$$

$$d_q = \sqrt{\left(\mathcal{PF}_{oq}^{1} - \mathcal{PF}_{oq+1}^{1}\right)^2 + \left(\mathcal{PF}_{oq}^{2} - \mathcal{PF}_{oq+1}^{2}\right)^2}, \tag{13}$$

$$\overline{d} = \frac{\sum_{q=1}^{\|\mathcal{PF}_o\|-1} d_q}{\|\mathcal{PF}_o\| - 1}. \tag{14}$$

Additionally, the hypervolume (HV) [19] measures the area (in a two-objective problem) or the volume between a reference point, R and the Pareto front with respect to the nondominated solutions obtained by the algorithm, as illustrated in Figure 5. Thus, it is desirable that the Pareto front returned by the algorithm produces a large HV.

4.2. Test Problems. Because different features in MOO problems are responsible for decreasing the likelihood of obtaining Pareto front with good convergence and diversity, the standard test functions with well-defined true Pareto fronts are important for testing optimisation algorithms. Five test functions from Zitzler et al. [20] (ZDT) are used here for this reason. The ZDT test problems have two objectives and are formulated with one feature in each problem. ZDT5 is not used because it is binary coded, whereas this work focuses on real-value problems. During testing, the GD, SP, and HV measure require the true Pareto front for the ZDT test problems, the standard database generated by jMetal (http://jmetal.sourceforge.net/problems.html) is used for this purpose. Additionally, all test problems used here are set up as recommended by [20].

4.3. Evaluation of VEPSO Algorithms. The performance comparison between conventional VEPSO, VEPSOnds, and

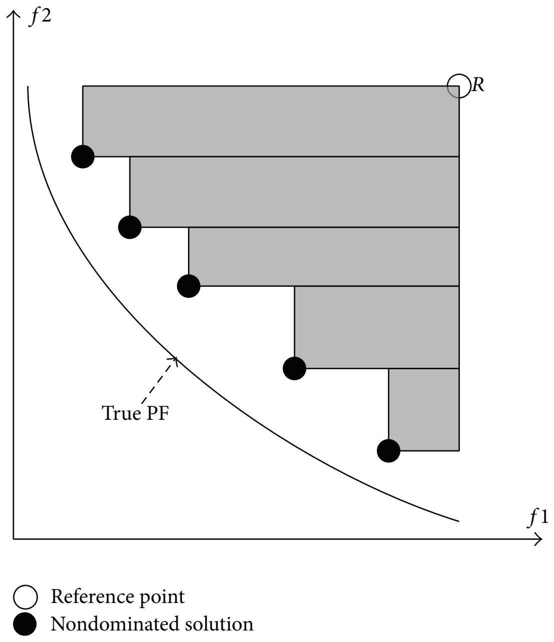


Figure 5: Hypervolume measure with area covered by nondominated solutions and reference point.

Table 1: Algorithm parameters.

Parameter	Value
Function evaluation	25000 (based on paper [14])
(i) Number of swarm	2
(ii) Particle for each swarm	50
(iii) Iterations for each run	250
c_1 and c_2	Random [1.5, 2.5]
ω	Linearly degrade from 1.0 to 0.4

VEPSOml is conducted without the use of polynomial mutation as to clarify that the polynomial mutation is not the sole reason for any performance improvement. Thus, this experiment compares the conventional VEPSO and the VEPSOnds without mutation against two different variations of VEPSOml: VEPSOml1 is the VEPSOml without mutation and VEPSOml2 is the VEPSOml with mutation, respectively.

All improved VEPSO algorithms are compared to the conventional VEPSO algorithm. Hence, similar parameters are used for all experimented algorithms which are listed in Table 1. In addition, the archive size is set to 100 solutions and is controlled by removing the nondominated solutions with the smallest crowding distance [14]. Each test problem is simulated for 100 runs on each algorithm to obtain statistical results for a fair comparison because the convergence and diversity performance varies in each run.

Table 2 lists the performance of each algorithm on the ZDT1 test problem. In the NS measure, the number of nondominated solutions significantly increases in all improved algorithms. Under the GD measure, VEPSOnds performs approximately 10 times better than conventional VEPSO. However, under the same measure, VEPSOml1 shows a more dramatic improvement, performing approximately 100 times better than VEPSO, as the concept of multiple nondominated leaders shows its benefit in finding more accurate solutions. Additionally, when the polynomial mutation is included, as in VEPSOml2, the GD performance improved much better at about 600% as compared to the conventional VEPSO. Under the SP measure, VEPSOnds also gives a significant improvement in performance. Meanwhile, the VEPSOml1 and VEPSOml2 show significant improvement in diversity performance as compared to the VEPSOnds. This shows the significance of using more than one nondominated solution which diversify the search toward the nondominated solutions at different end. The above mentioned improvements are supported by the higher HV measures when compared to the conventional VEPSO, which indicates that they return better Pareto fronts.

Figure 6 shows plots of the nondominated solutions with the best GD measure returned by each algorithm tested on ZDT1. From the first plot, it is clear that the nondominated solutions obtained by VEPSO are far away from the true Pareto front, which explains the poor performance of this algorithm for this test problem. In addition, the nondominated solutions are distributed unevenly, and so VEPSO has a larger SP value. Meanwhile for all the improved VEPSO algorithms, their nondominated solutions fall very close to the true Pareto front. However, VEPSOnds produces a distribution of nondominated solutions that contain empty spaces along the true Pareto front, which results in higher SP value as compared to the other improved VEPSO algorithms.

Table 3 lists the performance of the algorithms on the ZDT2 test problem. The average number of nondominated solutions found by VEPSOnds1 slightly improves over the number found by VEPSO, but VEPSOnds2, VEPSOml1, and VEPSOml2 greatly improve over VEPSO by this same

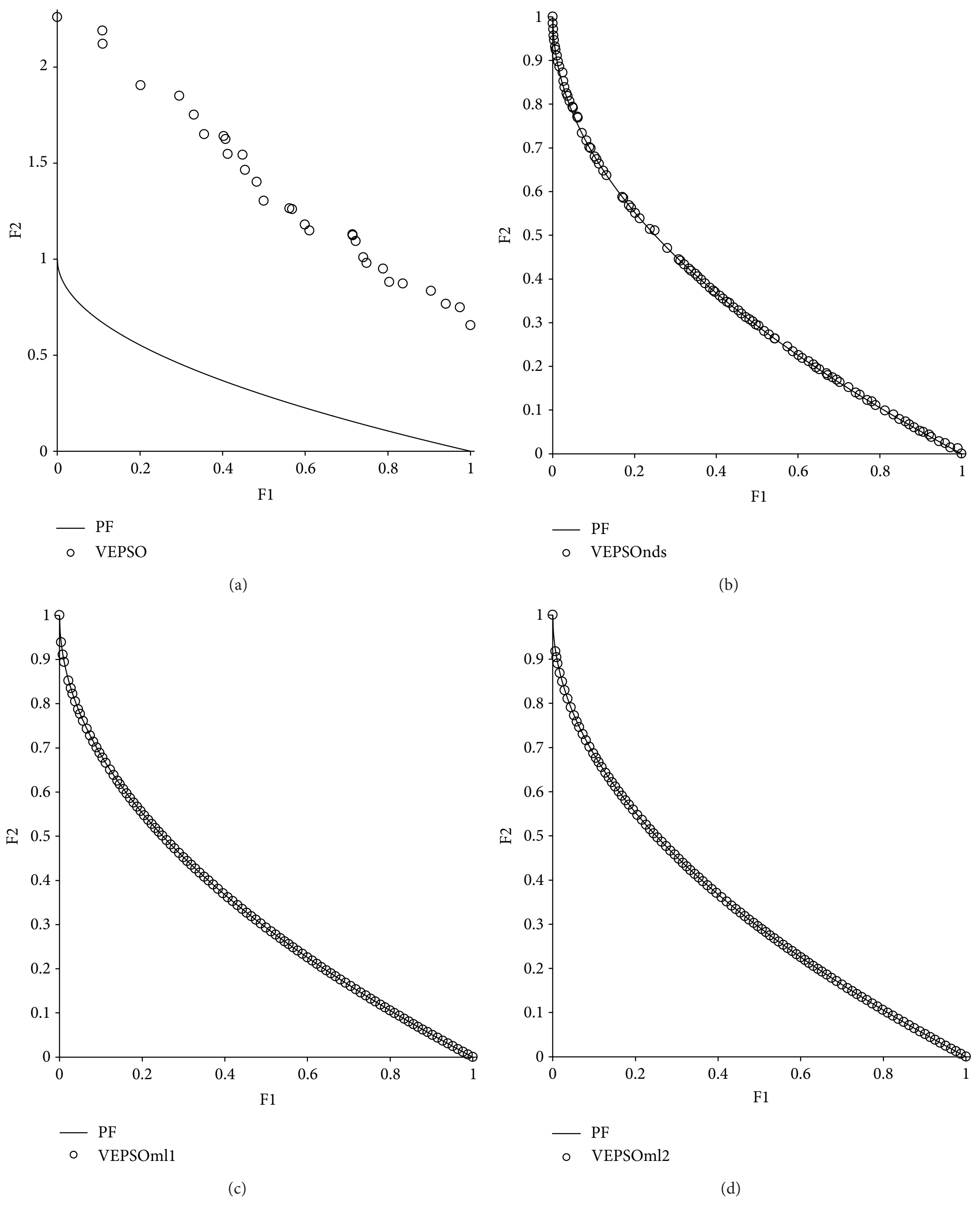


Figure 6: Plot of nondominated solutions returned by each algorithm for the ZDT1 test problem.

measure. Similarly, by the GD measure, VEPSOnds1 shows a small improvement, whereas VEPSOnds2 and VEPSOml1 show a larger improvement over the performance of VEPSO. In the same measure, VEPSOml2 shows a more significant improvement over the VEPSO and all other improved algorithms. Additionally, by the SP measure, VEPSOnds1 shows negligible improvement, whereas VEPSOnds2 shows a significant improvement over the performance of VEPSO. Besides, with the use of multileader, VEPSOml shows much better diversity performance than both the VEPSOnds. Finally, by the HV measure, VEPSO was unable to produce any hypervolume because its nondominated solutions are worse

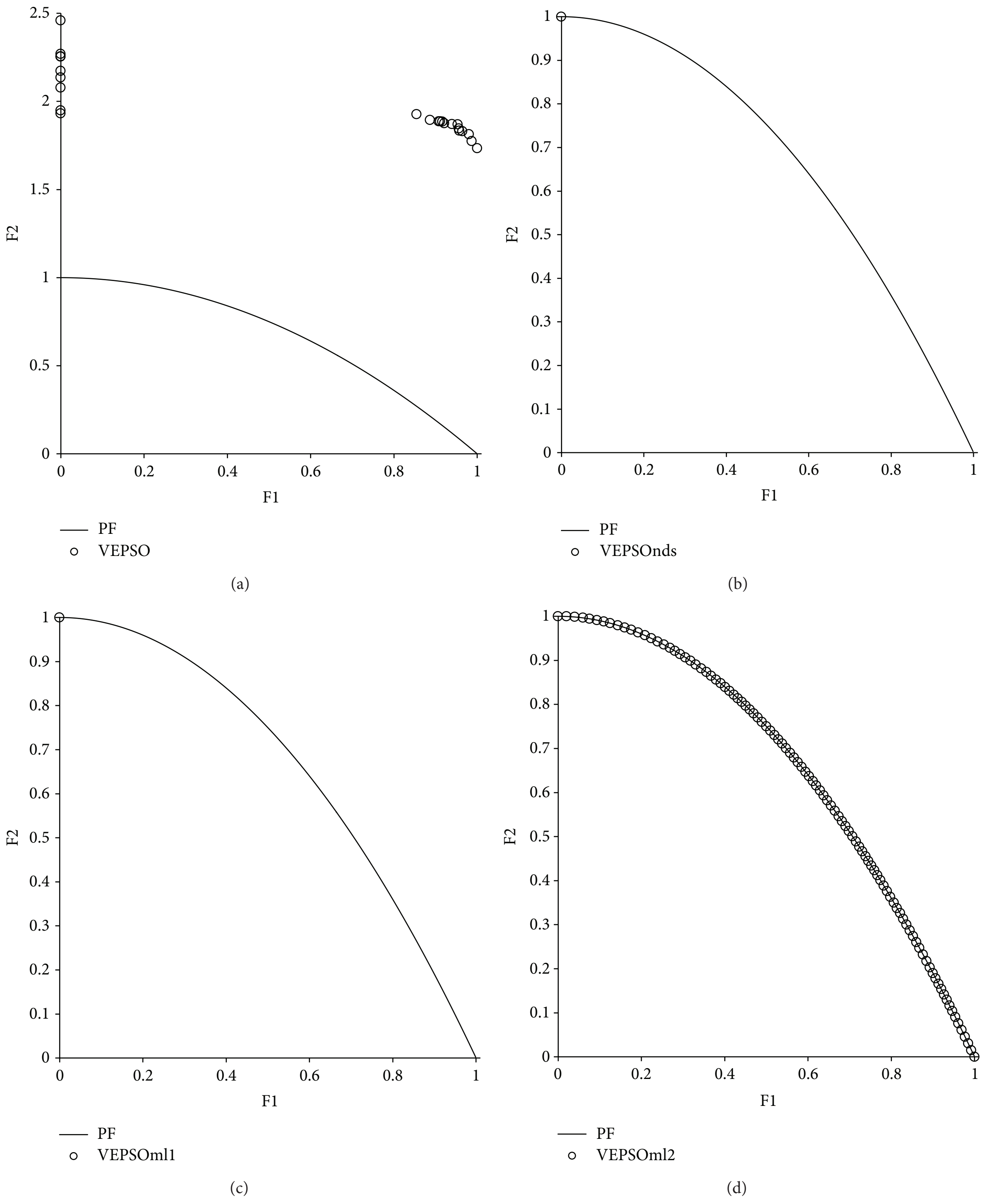


Figure 7: Plot of nondominated solutions returned by each algorithm for the ZDT2 test problem.

than the reference point, R. On the other hand, all improved algorithms are able to create a hypervolume, especially the VEPSOml2 which produce the largest hypervolume.

Figure 7 displays the nondominated solutions, plotted for each the best GD measure obtained for each algorithm using the ZDT2 test problem. The first plot shows that VEPSO returns nondominated solutions that are far from the true Pareto font and poorly distributed. Although VEPSOnds and VEPSOml1 return a low GD measure, the number of nondominated solutions is found to have low value, which

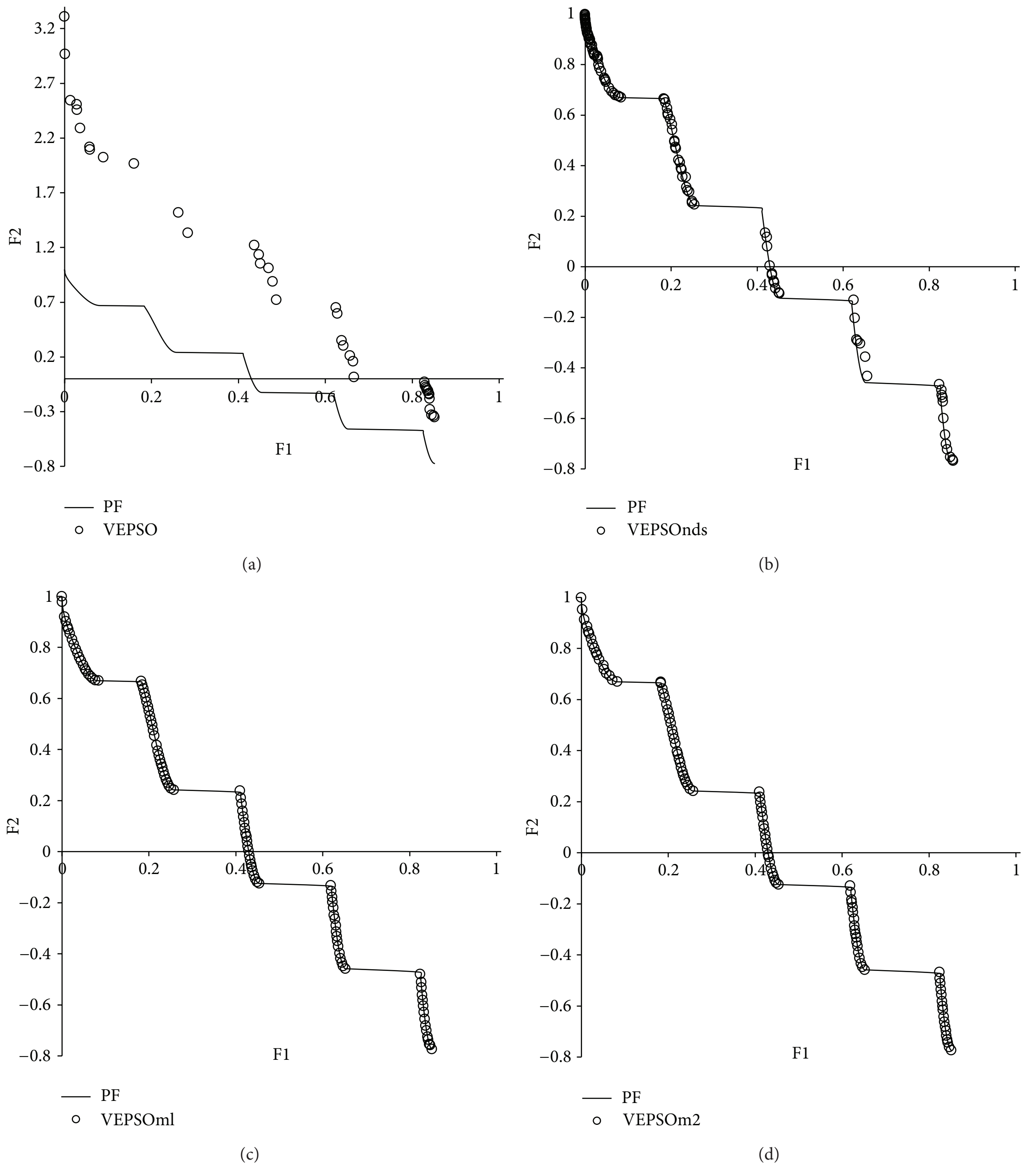


FIGURE 8: Plot of nondominated solutions returned by each algorithm for the ZDT3 test problem.

is clearly displayed in the second and third plots, respectively, of Figure 6. In fact, there is only one nondominated solution found by both algorithms which falls exactly on the true Pareto front and yields a GD value of zero. On the other hand, the fourth plot of Figure 6 shows that VEPSOml2 returns the nondominated solutions that converge nicely and are well distributed over the true Pareto front. Besides, the nondominated solutions found by VEPSOml2 distributed evenly which yield a good SP value.

Table 4 lists the performance of the algorithms on the ZDT3 test problem. All improved VEPSO algorithms are able to find more nondominated solutions than the conventional VEPSO algorithm. In addition, the performances of the improved VEPSO algorithms, with respect to convergence, improve on conventional VEPSO, while VEPSOml2 shows the greater improvement. However, by the SP measure, the VEPSOnds algorithm performs worse than the conventional VEPSO algorithm. However, although the SP value of the

Table 2: Algorithm performance tested on ZDT1 problem.

Measure		VEPSO	VEPSOnds	VEPSOml1	VEPSOml2
NS	Ave.	30.220000	100.000000	99.490000	98.820000
	SD	5.697031	0.000000	3.942760	6.979595
	Min.	16.000000	100.000000	63.000000	47.000000
	Max.	44.000000	100.000000	100.000000	100.000000
GD	Ave.	0.295865	0.022637	0.002730	0.000497
	SD	0.051645	0.014201	0.006219	0.002213
	Min.	0.139491	0.000283	0.000045	0.000047
	Max.	0.432478	0.073477	0.031891	0.015598
SP	Ave.	0.834481	0.729350	0.212479	0.182157
	SD	0.039111	0.160298	0.149696	0.113453
	Min.	0.705367	0.322322	0.106082	0.109998
	Max.	0.917087	1.219625	0.738619	0.779572
HV	Ave.	0.001886	0.428153	0.628841	0.657830
	SD	0.010058	0.113432	0.078273	0.023359
	Min.	—	0.185313	0.283932	0.456556
	Max.	0.087426	0.659603	0.662065	0.662022

Table 3: Algorithm performance tested on ZDT2 problem.

Measure		VEPSO	VEPSOnds	VEPSOml1	VEPSOml2
NS	Ave.	8.070000	38.120000	91.090000	99.620000
	SD	6.356822	25.747131	28.474726	3.800000
	Min.	1.000000	1.000000	1.000000	62.000000
	Max.	24.000000	100.000000	100.000000	100.000000
GD	Ave.	0.766956	0.039653	0.005109	0.000152
	SD	0.324444	0.063791	0.010867	0.001009
	Min.	0.240509	0.000000	0.000000	0.000043
	Max.	1.679803	0.310345	0.028380	0.010144
SP	Ave.	0.944524	0.947356	0.267797	0.098572
	SD	0.065266	0.111963	0.315008	0.065826
	Min.	0.797757	0.695715	0.059578	0.064648
	Max.	1.080351	1.278655	1.000004	0.721104
HV	Ave.	—	0.137784	0.250495	0.328291
	SD	—	0.117596	0.127625	0.004182
	Min.	—	—	0.000000	0.286901
	Max.	—	0.311075	0.328807	0.328816

conventional VEPSO algorithm is better, the superior convergence of the VEPSOnds algorithm maintains its performance advantage. In contrast, both improved VEPSO algorithm using multiple nondominated leaders show better SP measure than the conventional VEPSO, which strengthen the hypothesis that using multiple nondominated leaders will improve diversity performance. In addition, the HV value of the conventional VEPSO algorithm is smaller than of all improved algorithms which suggest that the improved algorithms have better performance.

Figure 8 displays the nondominated solutions, plotted for the best GD measure obtained for each algorithm using the ZDT3 test problem. The nondominated solutions returned by the conventional VEPSO algorithm were distributed equally but not well converged with respect to the true Pareto front. On the other hand, the nondominated solutions from all improved VEPSO algorithms are well converged with respect to the true Pareto front. However, the nondominated solutions returned by the VEPSOnds algorithm are denser at the upper left of the Pareto front, which causes the increase in its SP value. In contrast, the nondominated solutions obtained by both VEPSOml algorithms are equally distributed over the Pareto front and yield better SP value.

Table 5 lists the performance of the algorithms on the ZDT4 test problem. The average number of nondominated solutions obtained by VEPSO is relatively low, while all improved VEPSO algorithms found most of the solutions. In this test, the conventional VEPSO algorithm produced a very large GD value due to the multimodality feature in the test problem, and so the improved VEPSO algorithms

Table 4: Algorithm performance tested on ZDT3 problem.

Measure		VEPSO	VEPSOnds	VEPSOml1	VEPSOml2
NS	Ave.	35.150000	99.600000	95.710000	96.500000
	SD	6.853997	3.405284	11.528003	11.372037
	Min.	21.000000	66.000000	46.000000	49.000000
	Max.	53.000000	100.000000	100.000000	100.000000
GD	Ave.	0.173060	0.009607	0.002586	0.001456
	SD	0.031253	0.008293	0.003904	0.002533
	Min.	0.079595	0.000433	0.000153	0.000159
	Max.	0.276801	0.039481	0.017547	0.007328
SP	Ave.	0.871146	1.109448	0.761061	0.752151
	SD	0.043319	0.086041	0.056129	0.050459
	Min.	0.701884	0.902861	0.701924	0.703181
	Max.	1.001428	1.322024	0.934796	0.981492
HV	Ave.	0.004722	0.373133	0.476679	0.493073
	SD	0.021699	0.083015	0.060626	0.045211
	Min.	—	0.112859	0.289513	0.391275
	Max.	0.167359	0.506222	0.515919	0.515941

Table 5: Algorithm performance tested on ZDT4 problem.

Measure		VEPSO	VEPSOnds	VEPSOml1	VEPSOml2
NS	Ave.	6.610000	95.250000	82.730000	51.470000
	SD	3.920665	16.518967	30.304800	35.623864
	Min.	1.000000	15.000000	6.000000	4.000000
	Max.	21.000000	100.000000	100.000000	100.000000
GD	Ave.	5.062543	0.383646	0.231380	0.449095
	SD	3.167428	0.478535	0.841726	1.060986
	Min.	0.000000	0.000155	0.000062	0.000146
	Max.	13.350278	2.049212	7.013747	6.835452
SP	Ave.	0.858655	1.035510	0.572461	0.735715
	SD	0.147255	0.347336	0.286004	0.201246
	Min.	0.483073	0.077112	0.135264	0.269484
	Max.	1.236461	1.419225	1.139773	1.088971
HV	Ave.	0.228824	0.399914	0.357553	0.307568
	SD	0.188151	0.159971	0.281263	0.272435
	Min.	—	—	—	—
	Max.	0.573978	0.661941	0.661917	0.660309

clearly performed better in this respect. However, the diversity performance of nondominated solutions returned by conventional VEPSO is small compared to the VEPSOnds algorithm. Once again, the use of multiple nondominated leaders in VEPSO algorithms could diversify the search and result in better diversity performance. Additionally, all algorithms produce a hypervolume from the reference point, and all improved algorithms return larger HV values than the conventional algorithm.

Figure 9 displays the nondominated solutions, plotted for the best GD measure obtained for each algorithm using the ZDT4 test problem. The first plot shows that VEPSO converges to the Pareto front but only manages to obtain a single nondominated solution. The VEPSOnds algorithm not only converges to the Pareto front but also returns a diverse set of nondominated solutions. On the other hands, both VEPSOml also returned the nondominated solutions with good convergence but they are not well distributed as compared to the VEPSOnds, in this case. Thus, the VEPSOnds shows better HV value as compared to the VEPSOml.

Table 6 lists the performance of the algorithms on the ZDT6 test problem. All algorithms find a similar number of nondominated solutions. In the GD measure, all algorithms are capable of returning the nondominated solutions that converge well to the Pareto front. On the other hand, both VEPSOml1 and VEPSOml2 algorithms outperform the conventional VEPSO and VEPSOnds algorithm in the GD measure. In addition, the SP and HV values for each

Table 6: Algorithm performance tested on ZDT6 problem.

Measure		VEPSO	VEPSOnds	VEPSOml1	VEPSOml2
NS	Ave.	76.590000	78.040000	86.920000	88.590000
	SD	32.884891	26.684055	23.586368	23.600674
	Min.	11.000000	22.000000	25.000000	16.000000
	Max.	100.000000	100.000000	100.000000	100.000000
GD	Ave.	0.338537	0.260666	0.217503	0.217929
	SD	0.370336	0.158592	0.214344	0.263966
	Min.	0.001746	0.044137	0.031135	0.000482
	Max.	1.552521	0.709692	1.184075	1.316312
SP	Ave.	1.201796	1.276529	1.301493	1.273612
	SD	0.146782	0.083293	0.085611	0.186562
	Min.	0.492064	0.987981	0.931549	0.082405
	Max.	1.435395	1.437289	1.430321	1.439400
HV	Ave.	0.304584	0.303381	0.303676	0.315964
	SD	0.134813	0.102216	0.123985	0.121842
	Min.	0.000000	0.038143	0.000779	0.000001
	Max.	0.400964	0.400780	0.401403	0.401483

algorithm are similar. However, the VEPSOml2 algorithm shows superiority in getting the minimum SP value and average HV value.

As can be predicted from the similar quantitative performance of the algorithms on ZDT6, the plot of nondominated solutions returned by each algorithm is very similar, especially in convergence performance, as shown in Figure 10. The plots do show that VEPSO has slightly less diversity compared to VEPSOnds and VEPSOml2 because of some small gaps in coverage along the middle of the Pareto front. On the other hand, the VEPSOml1 shows weak distribution of nondominated solutions over the Pareto front. In contrast, the nondominated solutions found by VEPSOnds and VEPSOml2 completely cover the true Pareto front and are spaced out equally.

As seen from the results of all the test problems, the VEPSO algorithms using multiple nondominated leaders shows more improvement in terms of convergence and diversity of the nondominated solutions found than the VEPSOnds. The additional leader, specifically the nondominated solution with respect to the objective function optimised by a swarm, not only guides the particles to optimise the objective function with respect to the swarm. It also increased the search area because all leaders used to guide the particles are located at the different end of the Pareto front.

4.4. Analysis of the Number of Particles. This experiment analysed the performance of the VEPSOml2 algorithm with various numbers of particles. Similar parameters from the previous experiment were used except for the total number of particles as it is equally divided into two swarms; the total number of particles was varied to be 10, 30, 50, 100, 300, 500, and 1000. Figure 11 shows plots of the performance measures for each benchmark problem against the total number of particles.

The VEPSOml2 algorithm performance improved as the number of particles increased. The performance of the VEPSOml2 algorithm was sufficient when there were 100 particles computed for 250 iterations, which corresponds to 25000 function evaluations. However, the performance of the VEPSOml2 algorithm exhibited better results when the total number of particles was increased. Unfortunately, when the number of particles is increased, the algorithm requires more computational effort to solve the problem.

4.5. Analysis of the Number of Iterations. This experiment investigated the performance of VEPSOml2 for various numbers of iterations. The number of iterations was fixed to be 10, 30, 50, 100, 300, 500, 1000, 3000, 5000, and 10 000. Meanwhile, the other parameters were kept the same as in the previous experiment except that the number of particles, which were divided equally among swarms, was fixed to 100 divided equally between all swarms. Figure 12 plots the performance measures for each benchmark problem against the number of iterations.

As expected, the performance of VEPSOml2 is improved when the number of iterations was increased. When 100 particles were used, the VEPSOml2 algorithm started to yield acceptable results when there were 500 iterations, which is equivalent to 50000 function evaluations. However, if computational cost is not critical, the VEPSOml2 algorithm could use 3000 iterations because the performance saturated after this value.

4.6. Benchmarking with the State-of-the-Art Multiobjective Optimisation Algorithms. For benchmarking, the VEPSOml2 algorithm was compared to four other state-of-the-art MOO algorithms: nondominated sorting genetic algorithm-II (NSGA-II) [14], strength Pareto evolutionary algorithm 2 (SPEA2) [21], archive-based hybrid scatter search

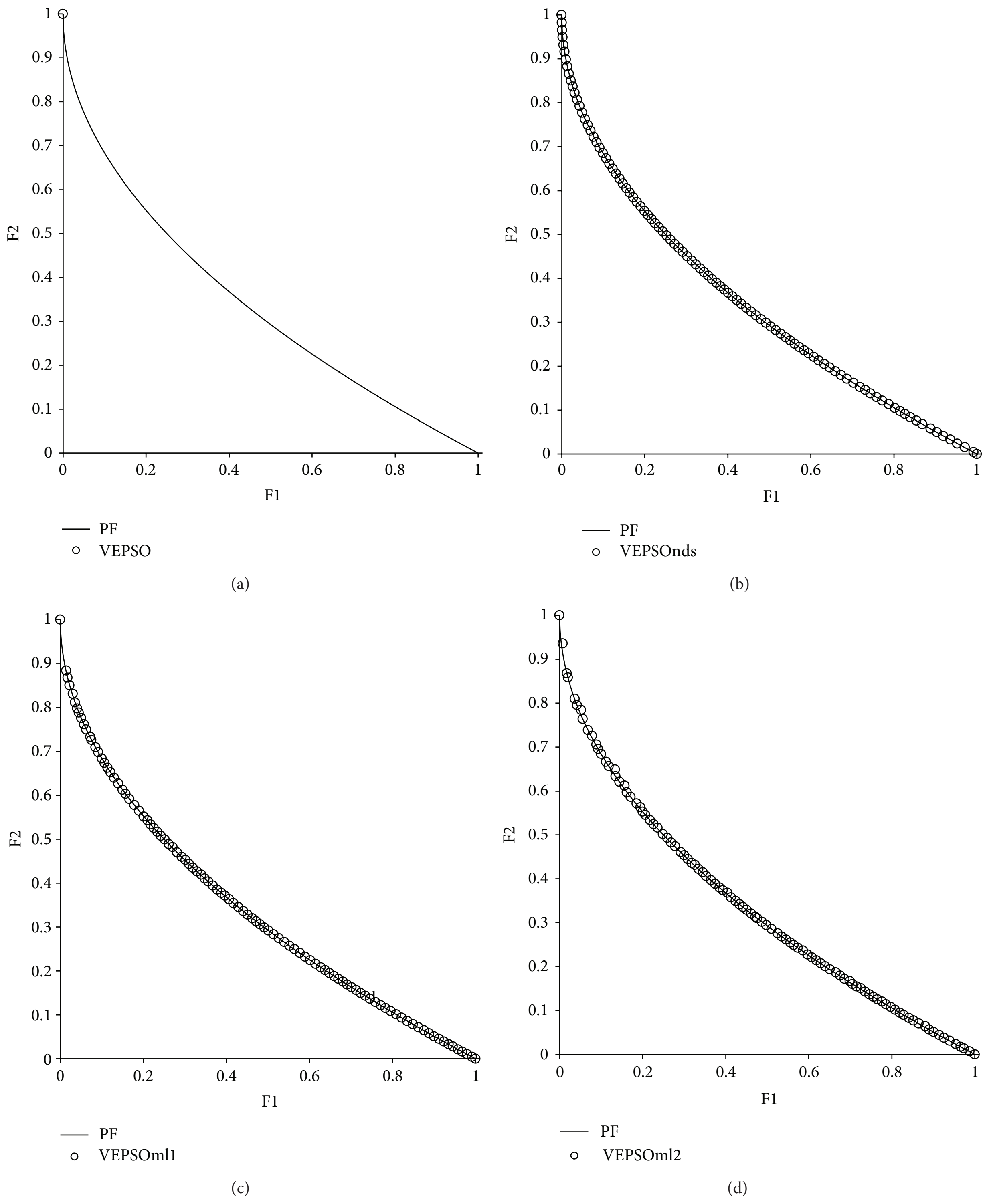
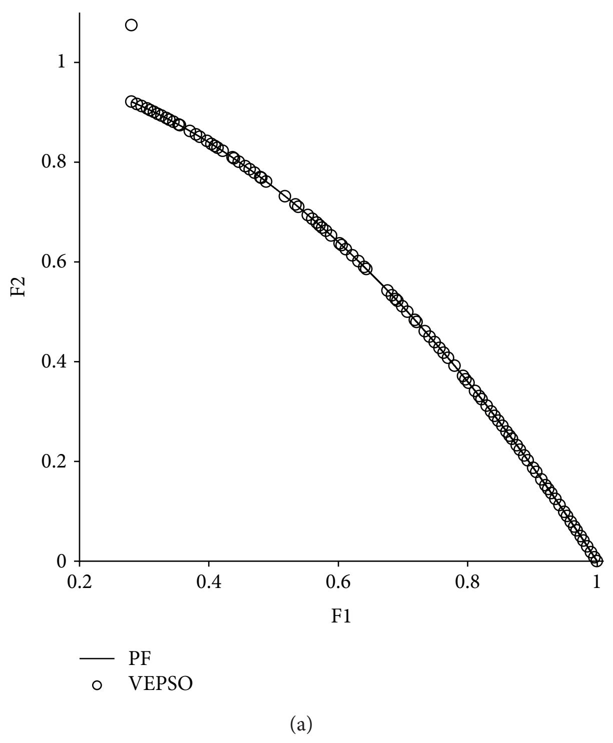
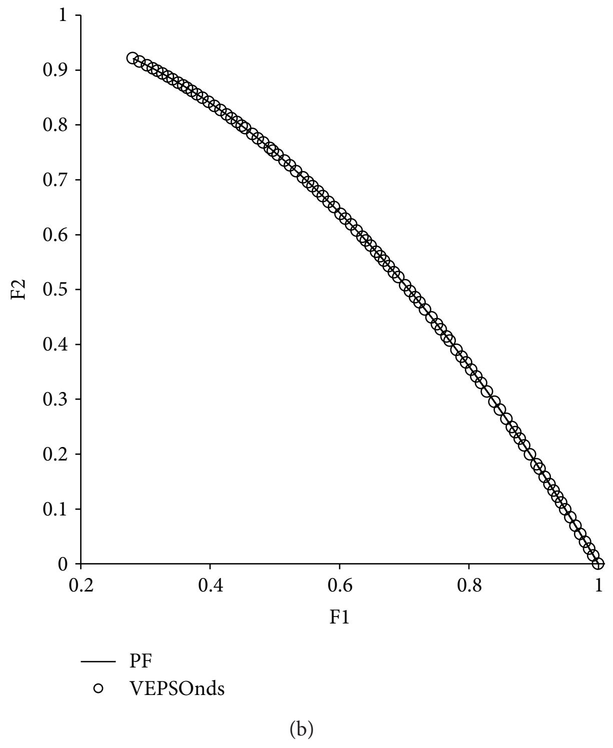
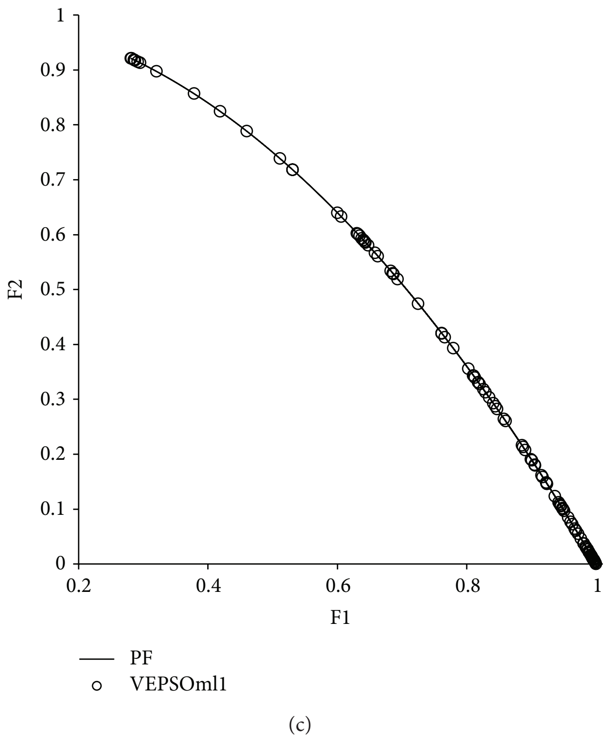


Figure 9: Plot of nondominated solutions returned by each algorithm for the ZDT4 test problem.

(AbYSS) [22], and the speed-constrained multiobjective PSO (SMPSO) algorithm [23]. All algorithms only computed 25000 function evaluations, and the archive size was set to 100 for fair comparison. The population size for NSGA-II was set to 100 for optimisation. The Simulated Binary Crossover (SBX) operator was used with crossover probability $p_c = 0.9$. The polynomial mutation [24] operator was also used with mutation probability $p_m = 1/N$. Meanwhile, the distribution indices for both operators were set to $\mu_n = \mu_m = 20$. The parameters in SPEA2 were set the same as in NSGA-II. The







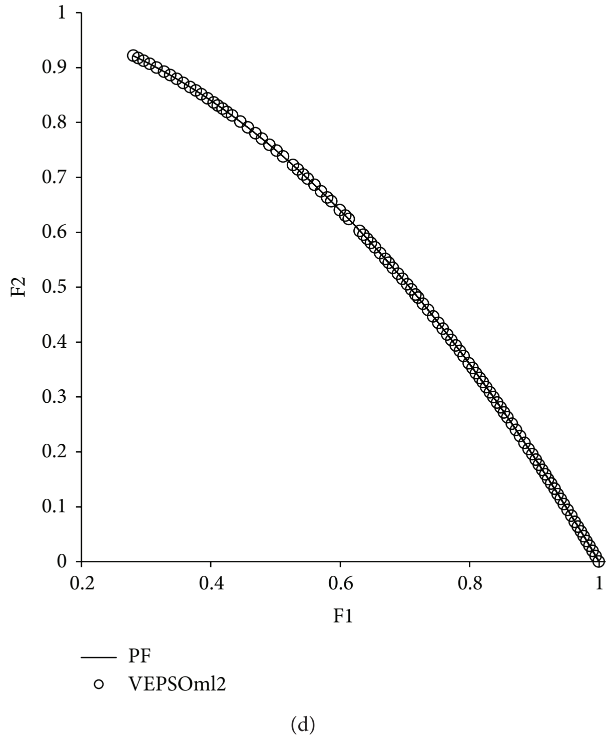


Figure 10: Plot of nondominated solutions returned by each algorithm for the ZDT6 test problem.

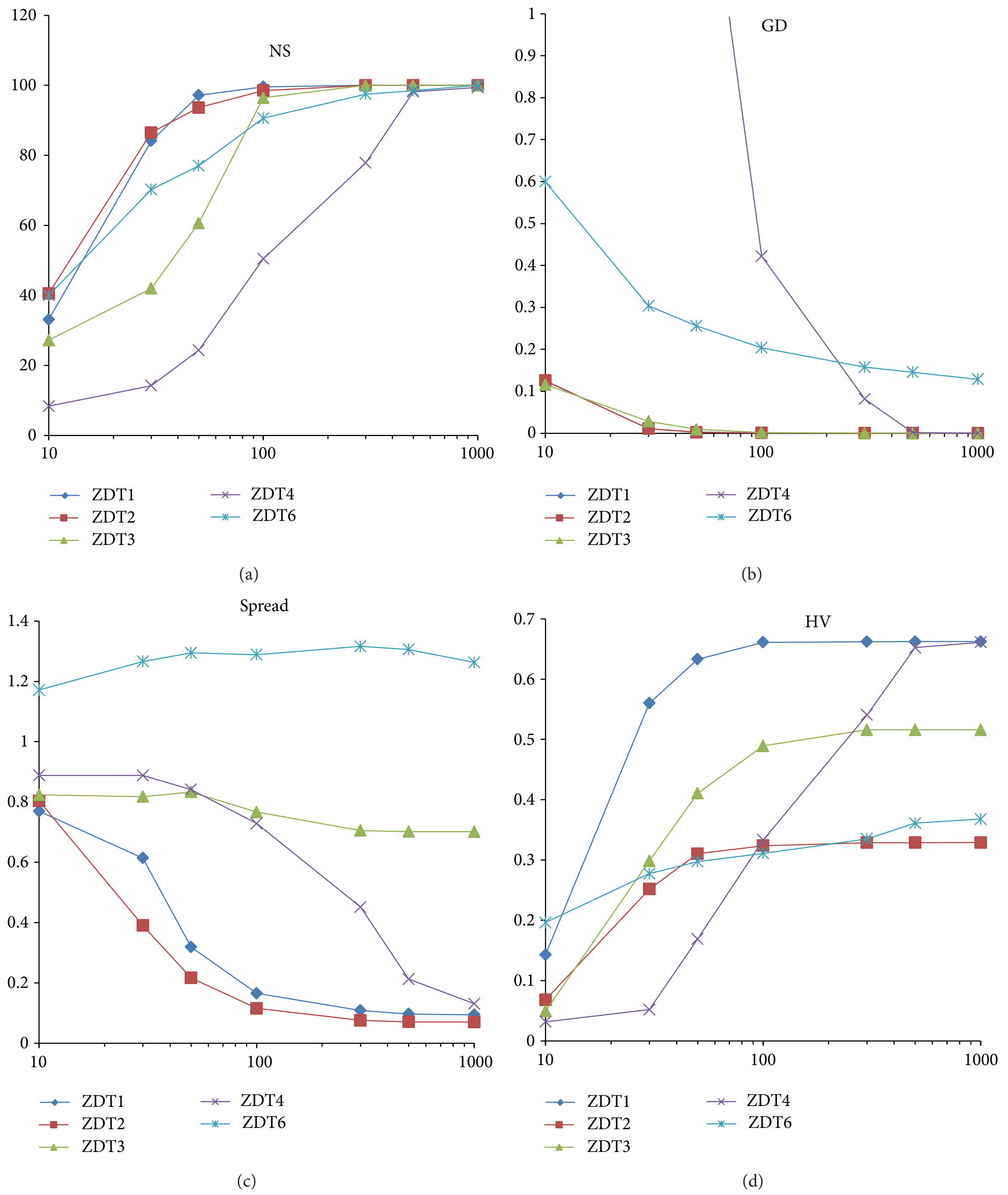


FIGURE 11: Plots of the performance measures versus numbers of particles. (a) Number of solutions. (b) Generational distance. (c) Spread. (d) Hypervolume.

population size for AbYSS was set to 20 and the pairwise combination parameters $RefSet_1$ and $RefSet_2$ were both set to 10. In addition, the polynomial mutation parameters in AbYSS were also set similarly as in NSGA-II and SPEA2. Finally, SMPSO was set to have a population size of 100 particles and a total number of iterations of 250. Moreover, the $r_1 = r_2 = \text{random}[0.1, 0.5]$, and the terms $c_1 = c_2 = \text{random}[1.5, 2.0]$. The polynomial mutation [25] operator was also used in SMPSO with $p_m = 1/N$ and $\mu_m = 20$.

The performance measures for the ZDT1 problem for all algorithms are listed in Table 7. The average number of solutions obtained by the VEPSOml2 was very similar to the other algorithms. Although VEPSOml2 algorithm had a GD measure approximately twice as large as those of the other

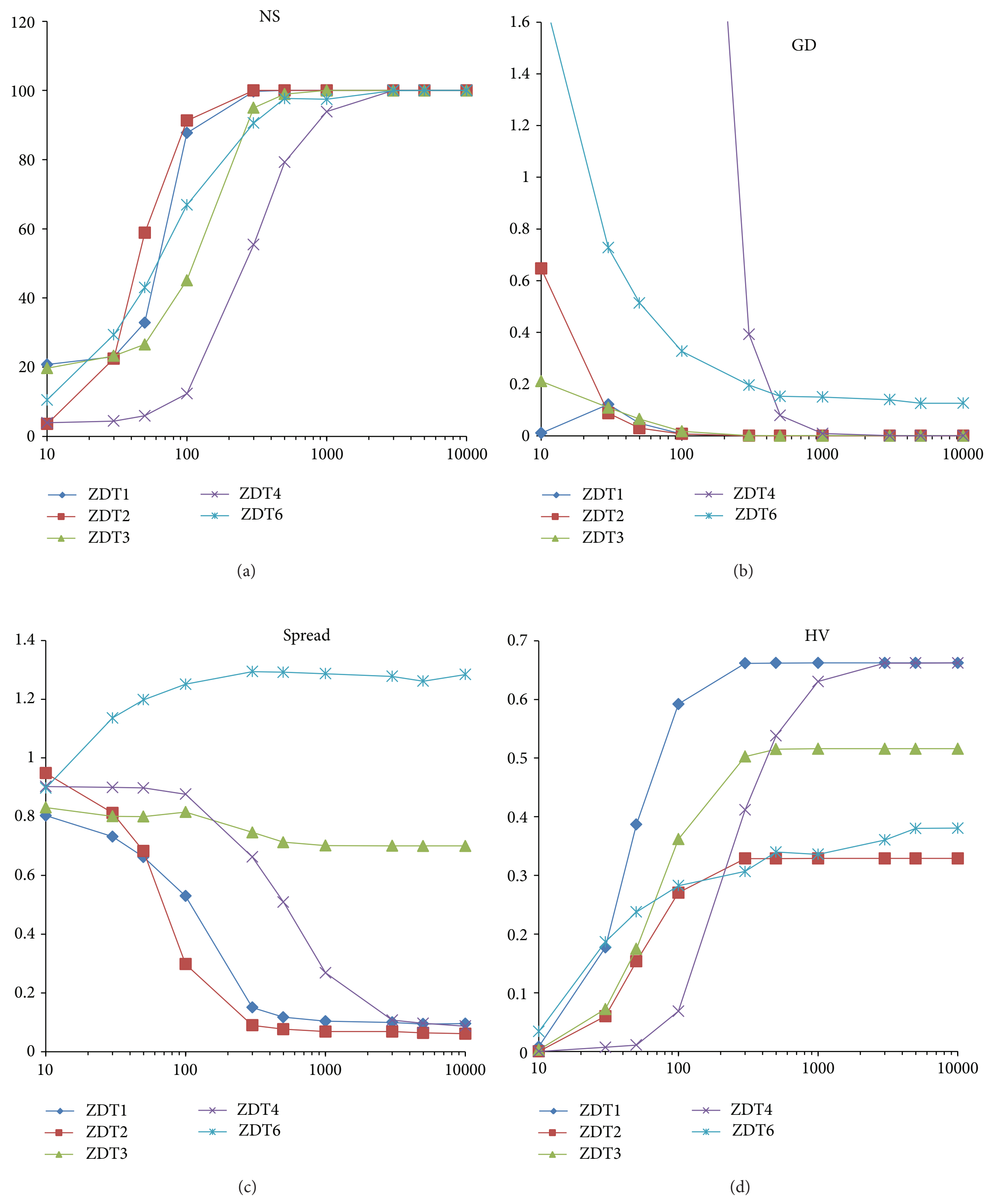


Figure 12: Plots of the performance metrics for various numbers of iterations. (a) Number of solution. (b) Generational distance. (c) Spread. (d) Hypervolume.

algorithms, its minimum GD was still the smallest among them. However, the SP was, on average, better than NSGA-II. Interestingly, the HV measure of VEPSOml2 was as good as those of the other algorithms.

Table 8 presents the performance measure of the algorithms for the ZDT2 problem. The VEPSOml2 was sufficiently competitive at obtaining a reasonable number of solutions. In the GD measure, on average, the VEPSOml2

Table 7: Performance comparison based on ZDT1 test problem.

Measure		AbYSS	NSGA-II	SPEA2	SMPSO	VEPSOml2
NS	Ave.	100.000000	100.000000	100.000000	100.000000	98.820000
	SD	0.000000	0.000000	0.000000	0.000000	6.979595
	Min.	100.000000	100.000000	100.000000	100.000000	47.000000
	Max.	100.000000	100.000000	100.000000	100.000000	100.000000
GD	Ave.	0.000185	0.000223	0.000220	0.000117	0.000497
	SD	0.000035	0.000038	0.000028	0.000031	0.002213
	Min.	0.000125	0.000146	0.000154	0.000053	0.000047
	Max.	0.000343	0.000374	0.000400	0.000172	0.015598
SP	Ave.	0.105387	0.379129	0.148572	0.076608	0.182157
	SD	0.012509	0.028973	0.012461	0.009200	0.113453
	Min.	0.080690	0.282485	0.116765	0.056009	0.109998
	Max.	0.136747	0.441002	0.174986	0.099653	0.779572
HV	Ave.	0.661366	0.659333	0.659999	0.661801	0.657830
	SD	0.000269	0.000301	0.000301	0.000100	0.023359
	Min.	0.660267	0.658486	0.659347	0.661372	0.456556
	Max.	0.661724	0.659909	0.660629	0.661991	0.662022

Table 8: Performance comparison based on ZDT2 test problem.

Measure		AbYSS	NSGA-II	SPEA2	SMPSO	VEPSOml2
NS	Ave.	100.000000	100.000000	100.000000	100.000000	99.620000
	SD	0.000000	0.000000	0.000000	0.000000	3.800000
	Min.	100.000000	100.000000	100.000000	100.000000	62.000000
	Max.	100.000000	100.000000	100.000000	100.000000	100.000000
GD	Ave.	0.000131	0.000176	0.000182	0.000051	0.000152
	SD	0.000067	0.000066	0.000039	0.000003	0.000152
	Min.	0.000056	0.000093	0.000090	0.000044	0.000043
	Max.	0.000433	0.000707	0.000304	0.000060	0.010144
SP	Ave.	0.130425	0.378029	0.158187	0.071698	0.098572
	SD	0.090712	0.028949	0.027529	0.013981	0.065826
	Min.	0.080831	0.311225	0.118114	0.035786	0.064648
	Max.	0.833933	0.430516	0.365650	0.106749	0.721104
HV	Ave.	0.325483	0.326117	0.326252	0.328576	0.328291
	SD	0.023209	0.000297	0.000908	0.000077	0.004182
	Min.	0.096409	0.325278	0.318785	0.328349	0.286901
	Max.	0.328505	0.326696	0.327559	0.328736	0.328816

algorithm was as good as the other algorithms, but SMPSO had greater performance. Surprisingly, the VEPSOml2 algorithm was able to obtain a better minimum GD measure than the SMPSO algorithm. Additionally, the SP measure of the VEPSOml2 algorithm was better than those of the other algorithms except SMPSO. All algorithms had similar HV values, but VEPSOml2 yielded the best HV performance.

The performance measures for the ZDT3 problem for all algorithms are listed in Table 9. Both SMPSO and VEPSOml2 were unable to obtain the maximum number of solutions consistently for all 100 runs but still yielded solutions within a reasonable range. Noticeably, the average GD measure for VEPSOml2 was the largest among all algorithms. However, the diversity for VEPSOml2 was similar to that of the others. Moreover, although the HV value of VEPSOml2 was the smallest, it still yielded a very large HV.

Table 10 presents the performance measures for the algorithms for the ZDT4 problem. VEPSOml2 faced great challenges from the multiple local optima featured in this problem, where it cause the algorithm to obtain a very small number of solutions. Additionally, the convergence and diversity of VEPSOml2 were bad, as indicated by the very large GD and SP values. As expected, the HV performance was also very poor because the multiple local optima feature is one of the natural weaknesses of PSO-based algorithms [26, 27].

Table 9: Performance comparison based on ZDT3 test problem.

Measure		AbYSS	NSGA-II	SPEA2	SMPSO	VEPSOml2
NS	Ave.	100.000000	100.000000	100.000000	99.900000	96.500000
	SD	0.000000	0.000000	0.000000	0.904534	11.372037
	Min.	100.000000	100.000000	100.000000	91.000000	49.000000
	Max.	100.000000	100.000000	100.000000	100.00000	100.000000
GD	Ave.	0.000193	0.000211	0.000230	0.000203	0.001456
	SD	0.000019	0.000013	0.000019	0.000061	0.002533
	Min.	0.000144	0.000180	0.000184	0.000155	0.000159
	Max.	0.000264	0.000268	0.000327	0.000717	0.007328
SP	Ave.	0.707651	0.747853	0.711165	0.717493	0.752151
	SD	0.013739	0.015736	0.008840	0.032822	0.050459
	Min.	0.696859	0.715199	0.698590	0.697943	0.703181
	Max.	0.796404	0.793183	0.775317	0.950901	0.981492
HV	Ave.	0.512386	0.514813	0.513996	0.514996	0.493073
	SD	0.011314	0.000159	0.000675	0.001737	0.045211
	Min.	0.463776	0.514449	0.510764	0.500484	0.391275
	Max.	0.515960	0.515185	0.514668	0.515818	0.515941

Table 10: Performance comparison based on ZDT4 test problem.

Measure		AbYSS	NSGA-II	SPEA2	SMPSO	VEPSOml2
NS	Ave.	99.680000	100.000000	100.000000	100.000000	51.470000
	SD	3.100603	0.000000	0.000000	0.000000	35.623864
	Min.	69.000000	100.000000	100.000000	100.000000	4.000000
	Max.	100.000000	100.000000	100.000000	100.000000	100.000000
GD	Ave.	0.001231	0.000486	0.000923	0.0001347	0.449095
	SD	0.002632	0.000235	0.001428	0.000027	1.060986
	Min.	0.000148	0.000163	0.000176	0.000070	0.000146
	Max.	0.014472	0.001374	0.012292	0.000187	6.835452
SP	Ave.	0.159842	0.392885	0.298269	0.092281	0.735715
	SD	0.120180	0.037083	0.125809	0.011777	0.201246
	Min.	0.078244	0.324860	0.137934	0.067379	0.269484
	Max.	1.073669	0.473358	0.884091	0.124253	1.088971
HV	Ave.	0.646058	0.654655	0.645336	0.661401	0.307568
	SD	0.034449	0.003406	0.018773	0.000162	0.272435
	Min.	0.472299	0.642177	0.505799	0.660934	—
	Max.	0.661594	0.659710	0.658784	0.661726	0.660309

Finally, the performance measures for the ZDT6 problem for all algorithms are listed in Table 11. VEPSOml2 algorithm was inconsistent in obtaining the maximum number of solutions. Moreover, the convergence and diversity measures for VEPSOml2 were significantly larger than those for the other algorithms. However, the VEPSOml2 algorithm was able to obtain the minimum GD value. Additionally, the HV performance for VEPSOml2 was relatively weak, on average, but its maximum HV value was the largest of all the algorithms.

An overall performance comparison for state-of-the-art algorithms against VEPSOml2 was investigated in this experiment. In some cases, the VEPSOml2 algorithm yielded better results than some of the other algorithms.

5. Conclusions

Most PSO-based MOO algorithms, including conventional VEPSO and VEPSOnds, only use one solution as the particle guide. Thus VEPSOml is proposed in this study where the particles are guided by multiple nondominated solutions while retaining the unique information shared between swarms that are inherent in conventional VEPSO.

Table 11: Performance comparison based on ZDT6 test problem.

Measure		AbYSS	NSGA-II	SPEA2	SMPSO	VEPSOml2
NS	Ave.	100.000000	100.000000	100.000000	100.000000	88.590000
	SD	0.000000	0.000000	0.000000	0.000000	23.600674
	Min.	100.000000	100.000000	100.000000	100.000000	16.000000
	Max.	100.000000	100.000000	100.000000	100.000000	100.000000
GD	Ave.	0.000549	0.001034	0.001761	0.012853	0.217929
	SD	0.000015	0.000102	0.000192	0.024813	0.263966
	Min.	0.000510	0.000804	0.001267	0.000502	0.000482
	Max.	0.000596	0.001360	0.002207	0.092434	1.316312
SP	Ave.	0.097740	0.357160	0.226433	0.390481	1.273612
	SD	0.013129	0.031711	0.020658	0.497140	0.186562
	Min.	0.070455	0.282201	0.179482	0.042666	0.082405
	Max.	0.130389	0.441311	0.292897	1.377582	1.439400
HV	Ave.	0.400346	0.388304	0.378377	0.401280	0.315964
	SD	0.000172	0.001604	0.002714	0.000076	0.121842
	Min.	0.399821	0.383637	0.371907	0.401081	0.000001
	Max.	0.400842	0.392123	0.385626	0.401402	0.401483

Five ZDT test problems were used to investigate the performance of the improved VEPSO algorithm based on the measures of the number of nondominated solutions found, the *generational distance*, the *spread,* and the *hypervolume*. The proposed VEPSOml algorithm obtained a higher-quality Pareto front as compared to conventional VEPSO and VEPSOnds. The VEPSOml2 algorithm that included polynomial mutation has exhibited further improvement for most of the performance measures.

Using more nondominated solutions as particle guides yielded faster convergence performance improvements, especially for the ZDT1, ZDT2, and ZDT3 test problems. The use of more than one leader reduced the risk of trapping at local Pareto front. In future, the success of using two leaders motivates the investigation of the use of more than two leaders during the optimisation process.

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 Research Acculturation Grant Scheme (RDU 121403) and MyPhD Scholarship from Ministry of Higher Education of Malaysia (MOHE), Program Rakan Penyelidikan (CG031-2013) from Universiti Malaya, and Research University Grant (GUP 04J99) from Universiti Teknologi Malaysia.

References

[1] K. E. Parsopóulos and M. N. Vrahatis, "Particle swarm optimization method in multiobjective problems," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 603–607, ACM, Madrid, Spain, March 2002.

[2] D. Gies and Y. Rahmat-Samii, "Vector evaluated particle swarm optimization (VEPSO): optimization of a radiometer array antenna," in *Proceedings of the IEEE Antennas and Propagation Society Symposium*, vol. 3, pp. 2297–2300, Monterey, Calif, USA, June 2004.

[3] S. M. V. Rao and G. Jagadeesh, "Vector evaluated particle swarm optimization (VEPSO) of supersonic ejector for hydrogen fuel cells," *Journal of Fuel Cell Science and Technology*, vol. 7, no. 4, Article ID 041014, 7 pages, 2010.

[4] S. N. Omkar, D. Mudigere, G. N. Naik, and S. Gopalakrishnan, "Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures," *Computers & Structures*, vol. 86, no. 1-2, pp. 1–14, 2008.

[5] J. G. Vlachogiannis and K. Y. Lee, "Multi-objective based on parallel vector evaluated particle swarm optimization for optimal steady-state performance of power systems," *Expert Systems with Applications*, vol. 36, no. 8, pp. 10802–10808, 2009.

[6] J. Grobler, *Particle swarm optimization and differential evolution for multi objective multiple machine scheduling [M.S. thesis]*, University of Pretoria, 2009.

[7] Z. Ibrahim, N. K. Khalid, J. A. A. Mukred et al., "A DNA sequence design for DNA computation based on binary vector evaluated particle swarm optimization," *International Journal of Unconventional Computing*, vol. 8, no. 2, pp. 119–137, 2012.

[8] Z. Ibrahim, N. K. Khalid, S. Buyamin et al., "DNA sequence design for DNA computation based on binary particle swarm optimization," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 5, pp. 3441–3450, 2012.

[9] J. D. Schaffer, *Some experiments in machine learning using vector evaluated genetic algorithms (artificial intelligence, optimization, adaptation, pattern recognition) [Ph.D. thesis]*, Vanderbilt University, 1984.

[10] K. S. Lim, Z. Ibrahim, S. Buyamin et al., "Improving vector evaluated particle swarm optimisation by incorporating nondominated solutions," *The Scientific World Journal*, vol. 2013, Article ID 510763, 19 pages, 2013.

[11] C. A. C. Coello and M. S. Lechuga, "Mopso: a proposal for multiple objective particle swarm optimization," in *Proceedings*

of the Congress on Evolutionary Computation (CEC '02), vol. 2, pp. 1051–1056, Honolulu, Hawaii, USA, 2002.

[12] C. A. C. 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.

[13] X. Li, "A non-dominated sorting particle swarm optimizer for multiobjective optimization," in *Genetic and Evolutionary Computation*, E. Cantú-Paz, J. Foster, K. Deb et al., Eds., vol. 2723 of *Lecture Notes in Computer Science*, pp. 37–48, Springer, Berlin, Germany, 2003.

[14] 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.

[15] M. Reyes-Sierra and C. A. C. Coello, "Improving PSO-based multiobjective optimization using crowding, mutation and ϵ-dominance," in *Evolutionary Multi-Criterion Optimization*, C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., vol. 3410 of *Lecture Notes in Computer Science*, pp. 505–519, Springer, Berlin, Germany, 2005.

[16] M. A. Abido, "Multiobjective particle swarm optimization with nondominated local and global sets," *Natural Computing*, vol. 9, no. 3, pp. 747–766, 2010.

[17] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Wash, USA, December 1995.

[18] D. A. V. Veldhuizen, *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations [Ph.D. thesis]*, Air Force Institute of Technology, Air University, 1999.

[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, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[21] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN '01)*, K. C. Giannakoglou, Ed., pp. 95–100, International Center for Numerical Methods in Engineering (CIMNE), 2002.

[22] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham, "AbYSS: adapting scatter search to multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 4, pp. 439–457, 2008.

[23] J. Durillo, J. García-Nieto, A. Nebro, C. A. C. Coello, F. Luna, and E. Alba, "Multi-objective particle swarm optimizers: an experimental comparison," in *Evolutionary Multi-Criterion Optimization*, M. Ehrgott, C. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds., vol. 5467 of *Lecture Notes in Computer Science*, pp. 495–509, Springer, Berlin, Germany, 2009.

[24] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, vol. 16 of *Systems and Optimization Series*, John Wiley & Sons, Chichester, UK, 2001.

[25] A. J. Nebro, J. J. Durillo, G. Nieto, C. A. C. Coello, F. Luna, and E. Alba, "SMPSO: a new PSO-based metaheuristic for multi-objective optimization," in *Proceedings of the IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM '09)*, pp. 66–73, Nashville, Tenn, USA, April 2009.

[26] E. Özcan and M. Yılmaz, "Particle swarms for multimodal optimization," in *Adaptive and Natural Computing Algorithms*, B. Beliczynski, A. Dzielinski, M. Iwanowski, and B. Ribeiro, Eds., vol. 4431 of *Lecture Notes in Computer Science*, pp. 366–375, Springer, Berlin, Germany, 2007.

[27] I. Schoeman and A. Engelbrecht, "A parallel vector-based particle swarm optimizer," in *Adaptive and Natural Computing Algorithms*, B. Ribeiro, R. F. Albrecht, A. Dobnikar, D. W. Pearson, and N. C. Steele, Eds., pp. 268–271, Springer, Vienna, Austria, 2005.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 620617, 7 pages
http://dx.doi.org/10.1155/2014/620617

Research Article

A Hybrid Multiuser Detector Based on MMSE and AFSA for TDRS System Forward Link

Zhendong Yin, Xu Jiang, Zhilu Wu, and Xiaohui Liu

School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China

Correspondence should be addressed to Zhendong Yin; yinzhendong@hit.edu.cn

Received 21 February 2014; Accepted 8 April 2014; Published 22 April 2014

Academic Editors: V. Bhatnagar and Y. Zhang

Copyright © 2014 Zhendong Yin 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 mainly focuses on multiuser detection in tracking and data relay satellite (TDRS) system forward link. Minimum mean square error (MMSE) is a low complexity multiuser detection method, but MMSE detector cannot achieve satisfactory bit error ratio and near-far resistance, whereas artificial fish swarm algorithm (AFSA) is expert in optimization and it can realize the global convergence efficiently. Therefore, a hybrid multiuser detector based on MMSE and AFSA (MMSE-AFSA) is proposed in this paper. The result of MMSE and its modified formations are used as the initial values of artificial fishes to accelerate the speed of global convergence and reduce the iteration times for AFSA. The simulation results show that the bit error ratio and near-far resistance performances of the proposed detector are much better, compared with MF, DEC, and MMSE, and are quite close to OMD. Furthermore, the proposed MMSE-AFSA detector also has a large system capacity.

1. Introduction

In 1963, due to the limited coverage of low-altitude orbiting spacecraft by a practical number of ground stations, F. O. Vonbun conceived the idea of tracking and data relay satellites (TDRS). Decades of space technology development now offer a practical extension from present ground-to-ground and air-to-ground communication via satellites to new applications [1].

Tracking and data relay satellite system (TDRSS) can provide services of data relaying, continuous tracking, and telemetry tracking and command (TT&C) for communications between spacecraft such as low earth orbit (LEO), middle earth orbit (MEO), and ground stations, which constitute important part of global space-based integrated information networks [2]. TDRSS provides S-band services through the S-band multiple access (SMA) phased array [3]. Actually the multiple access interference (MAI) is a serious limiting condition for improving the performance and the user capacity of this MA system, particularly when the number of users in this system is large.

Multiuser detection is a useful method to eliminate the bad effect of MAI. The best performance is acquired by OMD provided by Verdu in 1986, which is based on the maximum likelihood function [4]. However, this method tends to be quite complex. Consequently, multiuser detectors based on compressive sensing [5], Tikhonov regularization [6], ant colony optimization [7], adaptive LMS, and GA [8] have been devoted to the development of lower-complexity techniques that can achieve some of the benefits of the optimal procedures. However, the tradeoff problem between computational complexity and BER performance still exists.

Swarm intelligence (SI) is an innovative artificial intelligence technique for optimization [9]. The underlying perception in most of the biological case studies of SI has been that the individual animal is cognitively relatively simple and restricted in what it can achieve, whereas the group collectively is capable of astonishing feats [10]. As ones of the latest methods in the field of signal processing [11] (especially for combinatorial optimization problems [12]), several detectors based on swarm intelligence, such as ant colony optimization [13], particle swarm optimization [14, 15], and improved particle swarm optimization [16], have been considered. The artificial fish swarm algorithm (AFSA) reflects many excellent properties in applications such as insensitivity to initial values, strong robustness and much flexibility in practice, optimization precision, rapidness to search the global optimum, tolerance of parameter setting, and searching adaptation [17]. It is applied in various optimization applications such as solving III-conditioned linear systems of equations [18] and reactive power optimization for power system [19]. And several improved AFSAs have been proposed [20, 21]. In [20], an artificial fish swarm algorithm based on chaos search is proposed, which can not only overcome the disadvantage of easily getting into the local optimum in the later evolution period but also keep the rapidity of the previous period. In [21], two artificial fish swarm algorithms based on fuzzy system are proposed; the overall results show that proposed algorithms can surprisingly be effective.

In this paper, a multiuser detector in TDRS system forward link is employed. In order to accelerate the speed of global convergence and reduce the number of iterations for AFSA, the result of MMSE and its modified formations are used as the initial values of artificial fishes. Experimental results demonstrate that the BER and near-far resistance performances of the proposed MMSE-AFSA detector are better, compared with matched filter (MF), decorrelating detector (DEC), and MMSE, and are quite close to OMD.

This paper is organized as follows. Section 2 introduces the system model of multiuser detector in TDRS systems and several existing detectors. In Section 3, basic principle of AFSA and the proposed MMSE-AFSA detector are illustrated. Then in Section 4, experiments that compare with the performances of MF, DEC, MMSE, MMSE-AFSA, and OMD are analyzed. The paper is concluded in Section 5.

2. System Model and Several Existing Methods

2.1. System Model. Consider a TDRS system with S-band code division multiple access (CDMA). Assume there are K simultaneously active users. Over additive white Gaussian noise (AWGN) channel, the equivalent low-pass received waveform can be expressed as

$$y(t) = \sum_{k=1}^{K} \sqrt{E_k} s_k(t) b_k + n(t) \qquad 0 \le t \le T, \tag{1}$$

where E_k, $s_k(t)$, and $b_k \in \{-1, 1\}$ represent energy per bit, unit-energy signature waveform, and bit value of the kth user, respectively, $n(t)$ is the noise, and T is the bit interval.

The output of the matched filter of user k sampled at T is achieved by the following equation:

$$y_k = \int_0^T y(t) s_k(t)\,\mathrm{d}t = \sqrt{E_k} b_k + \sum_{\substack{i=1 \\ i \ne k}}^{K} \sqrt{E_i} \rho_{ik} b_i + n_k, \tag{2}$$

where the noise at the output of the kth matched filter is $n_k = \int_0^T s_k(t) n(t) \mathrm{d}t$, and the cross correlation of the signature waveforms of users i and k is $\rho_{ik} = \int_0^T s_i(t) s_k(t) \mathrm{d}t$.

The matched filter outputs can be expressed in vector form as follows:

$$\mathbf{y} = [y_1 y_2 \dots y_K]^T = \mathbf{RAb} + \mathbf{n}, \tag{3}$$

where $\mathbf{R}$ is the normalized cross correlation matrix of the signature waveforms, $\mathbf{R}_{ij} = \rho_{ij}$, $\mathbf{A} = \operatorname{diag}[\sqrt{E_1}\sqrt{E_2}\cdots\sqrt{E_K}]_{K\times K}$, and $\mathbf{n}$ is the zero-mean AWGN noise vector.

The symbol decisions of matched filter are given by

$$\hat{\mathbf{b}} = \operatorname{sgn}(\mathbf{y}). \tag{4}$$

2.2. Decorrelating Detector. Numerous suboptimal approaches to multiuser detection have been proposed to trade off performance and complexity. A widely studied linear solution is decorrelating detector. In this category, the decorrelator completely eliminates the MAI by orthogonalizing the users. The transformation $\mathbf{R}^{-1}$ is applied to the output of matched filters; the symbol decisions are given by

$$\hat{\mathbf{b}}_{\mathrm{DEC}} = \operatorname{sgn}\left(\mathbf{R}^{-1} y\right) = \operatorname{sgn}\left(\mathbf{Ab} + \mathbf{R}^{-1}\mathbf{n}\right). \tag{5}$$

It can be immediately inferred that each component of the decision vector $\mathbf{y}_{\mathrm{dec}}$ is interference-free. On the other hand, the background noise can be enhanced by the transformation $\mathbf{R}^{-1}$.

2.3. Minimum Mean Square Error Detector. Another important linear detector is minimum mean square error detector. The aim of MMSE detector is to choose the $K \times K$ matrix $\mathbf{M}$ that minimizes

$$\Omega(\mathbf{M}) = \min E\left\{\|\mathbf{b} - \mathbf{My}\|^2\right\}. \tag{6}$$

It can be easily seen that $\mathbf{M} = \mathbf{A}^{-1}[\mathbf{R} + \sigma^2\mathbf{A}^{-2}]^{-1}$ is the solution to (6). The symbol decisions are

$$\hat{\mathbf{b}}_{\mathrm{MMSE}} = \operatorname{sgn}(\mathbf{My}). \tag{7}$$

It balances the desire to completely eliminate the MAI with the desire to avoid the background noise enhancement.

2.4. Optimal Multiuser Detector. On the basis of matched filter, optimal detector takes advantage of the maximum likelihood sequence detection algorithm to improve the performance of multiuser detector. The likelihood function of y given b is given by

$$p(\mathbf{y} \mid \mathbf{b}) = \exp\left(\frac{-(1/2)(\mathbf{y} - \mathbf{RAb})^T\left(\sigma^2\mathbf{R}\right)^{-1}(\mathbf{y} - \mathbf{RAb})}{(2\pi)^{K/2}\sigma|\mathbf{R}|^{1/2}}\right), \tag{8}$$

where $|\mathbf{R}|$ denotes the determinant of $\mathbf{R}$. The maximum likelihood symbol decisions are determined as

$$\hat{\mathbf{b}}_{\mathrm{OMD}} = \arg\max_{\mathbf{b}}\left\{2\mathbf{b}^T\mathbf{Ay} - \mathbf{b}^T\mathbf{ARAb}\right\}. \tag{9}$$

The above maximization problem is a combinatorial optimization problem which is known to be NP-hard: its computational complexity increases exponentially with the number of users in TDRS system. This $O(2^K)$ implementation complexity required by OMD makes it impractical for real system. OMD represents, however, a basis for comparison for other suboptimal detectors.

3. MMSE-AFSA Detector

3.1. Basic Principles of AFSA. Artificial fish swarm algorithm is a new bionic optimization algorithm based on the study of fish swarm's intelligence and behaviors in nature. There are mainly three types of fish behaviors: preying behavior, swarming behavior, and following behavior. The general AFSA is introduced below.

3.1.1. Several Definitions for AFSA. In the AFSA, suppose there are n artificial fishes. The state of each artificial fish can be expressed as a K-dimensional vector $\mathbf{X} = (x_1, x_2, \ldots, x_K)^T$. The objective function $\mathbf{Y} = f(\mathbf{X})$ denotes the food concentration level of this state. The distance between states $\mathbf{X}_i$ and $\mathbf{X}_j$ is defined as

$$\begin{aligned} d_{ij} &= \left\|\mathbf{X}_i - \mathbf{X}_j\right\| \\ &= \sqrt{\left(x_{i1} - x_{j1}\right)^2 + \left(x_{i2} - x_{j2}\right)^2 + \cdots + \left(x_{iK} - x_{jK}\right)^2}. \end{aligned} \quad (10)$$

Besides, *Visual* denotes the local visual (or searching) distance of artificial fishes; δ is the factor of crowdedness that affects the number of artificial fishes in the local space; step is the movement size of artificial fishes; *try_number* is the random searching times in preying behavior.

3.1.2. Behaviors of AFSA

Preying Behavior. Suppose that the current state of an artificial fish is $\mathbf{X}_i$. $\mathbf{X}_j$ is a random state chosen in its visual field. In the maximum problem, if $f(\mathbf{X}_j) > f(\mathbf{X}_i)$, this artificial fish will move from state $\mathbf{X}_i$ to $\mathbf{X}_j$ as

$$\mathbf{X}_{i\text{next}} = \mathbf{X}_i + \text{rand}(0,1) \times \text{step} \times \frac{\mathbf{X}_j - \mathbf{X}_i}{\left\|\mathbf{X}_j - \mathbf{X}_i\right\|}. \quad (11)$$

Otherwise, choose a new state $\mathbf{X}_j$ randomly again and judge whether it satisfies the movement condition $(f(\mathbf{X}_j) > f(\mathbf{X}_i))$. If there is no such $\mathbf{X}_j$ that can satisfy this condition after trying *try_number* times, this artificial fish will move one step randomly at last

$$\mathbf{X}_{i\text{next}} = \mathbf{X}_i + \text{rand}(0,1) \times \text{step}. \quad (12)$$

Swarming Behavior. The current state of an artificial fish is $\mathbf{X}_i$, and n_f is the number of companions within its visual range. Thus, the central state of these artificial fishes is given by

$$\mathbf{X}_c = \sum_{j=1}^{n_f} \frac{\mathbf{X}_j}{n_f}. \quad (13)$$

If $f(\mathbf{X}_c)/n_f > \delta f(\mathbf{X}_i)$, which means the food concentration of $\mathbf{X}_c$ is sufficient and this area is not too crowded, then this artificial will move to the central state $\mathbf{X}_c$ as

$$\mathbf{X}_{i\text{next}} = \mathbf{X}_i + \text{rand}(0,1) \times \text{step} \times \frac{\mathbf{X}_c - \mathbf{X}_i}{\left\|\mathbf{X}_c - \mathbf{X}_i\right\|}. \quad (14)$$

Otherwise, preying behavior will be executed.

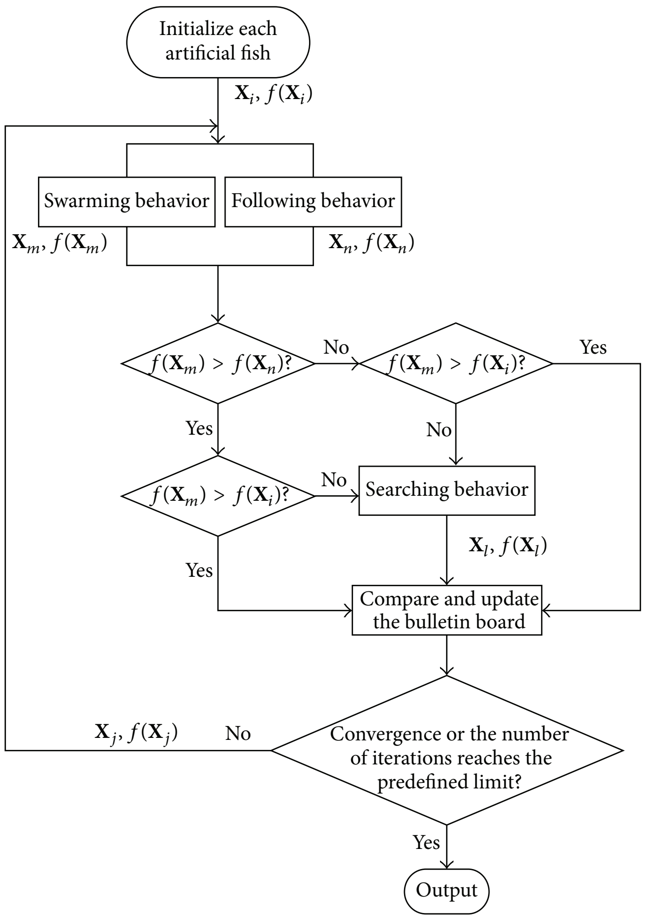

Initialize each artificial fish
$\mathbf{X}_i, f(\mathbf{X}_i)$
Swarming behavior
Following behavior
$\mathbf{X}_m, f(\mathbf{X}_m)$
$\mathbf{X}_n, f(\mathbf{X}_n)$
$f(\mathbf{X}_m) > f(\mathbf{X}_n)$?
No
$f(\mathbf{X}_m) > f(\mathbf{X}_i)$?
Yes
Yes
No
$f(\mathbf{X}_m) > f(\mathbf{X}_i)$?
No
Searching behavior
$\mathbf{X}_l, f(\mathbf{X}_l)$
Yes
Compare and update the bulletin board
$\mathbf{X}_j, f(\mathbf{X}_j)$
No
Convergence or the number of iterations reaches the predefined limit?
Yes
Output

FIGURE 1: The behavior of each artificial fish in AFSA.

Following Behavior. Within the visual range of $\mathbf{X}_i$, $\mathbf{X}_{\max}$ denotes the state whose food concentration $f(\mathbf{X}_{\max})$ is maximum. If $f(\mathbf{X}_{\max})/n_f > \delta f(\mathbf{X}_i)$ and $f(\mathbf{X}_{\max}) > f(\mathbf{X}_i)$, this artificial fish will move to state $\mathbf{X}_{\max}$ as follows:

$$\mathbf{X}_{i\text{next}} = \mathbf{X}_i + \text{rand}(0,1) \times \text{step} \times \frac{\mathbf{X}_{\max} - \mathbf{X}_i}{\left\|\mathbf{X}_{\max} - \mathbf{X}_i\right\|}. \quad (15)$$

Otherwise, preying behavior will be executed.

3.1.3. Bulletin Board. A bulletin board is established to record the optimal state and the optimal value of these artificial fishes. Each artificial fish will compare its current state to the state on the bulletin board. If its food concentration is better, update the bulletin board with the better state.

3.1.4. Behavior Selection. Evaluate the current environment of artificial fishes according to the problem to be solved, and then select a behavior. In the maximum problem, simulate swarming behavior and following behavior of each artificial fish and compare the food concentration of two behaviors, and the better behavior will be implemented. If none of them can improve the former state of the certain artificial fish, preying behavior will be executed. The behavior of each artificial fish in AFSA is shown in Figure 1.

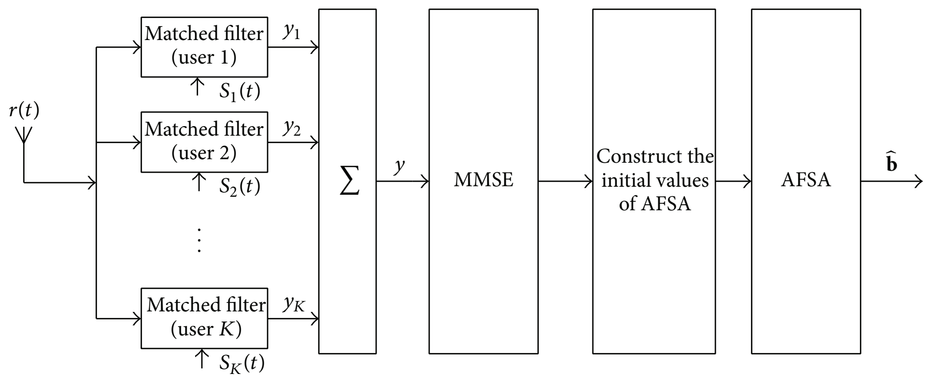


FIGURE 2: Diagram of MMSE-AFSA detector.

3.2. The Discretization of AFSA. The process of OMD is similar to that of a function's optimization. Whereas AFSA is expert in optimization and it can realize the global convergence efficiently, the optimization function for OMD is shown in (9), which is a discrete optimization function. Therefore, the model of AFSA should be discretized. AFSA applied to multiuser detection problem with some additional explications in the discrete Euclidean solution space $\mathbf{E}^K$ are expressed as follows.

(1) In the Euclidean solution space $\mathbf{E}^K$, the state of each fish is encoded by +1 or −1. If there are K active users in a TDRS system, the state is a K-dimensional vector, like $\mathbf{X} = (x_1, x_2, \ldots, x_K)^T$, where $x_i \in \{+1, -1\}$, $i = 1, 2, \ldots, K$.

(2) The initial value of each artificial fish is selected randomly in the discrete solution space with 2^K likely solutions.

(3) In this case, the operator XOR is used to calculate distance between states of two artificial fishes. For instance, the state of an artificial fish $\mathbf{X}_i = (+1, +1, -1, +1, -1)$, the state of another artificial fish $\mathbf{X}_j = (+1, -1, +1, -1, +1)$, then the distance between the two artificial fishes $d_{ij} = \mathbf{X}_i \text{ XOR } \mathbf{X}_j = 4$.

(4) The central state of a certain artificial fish is given by

$$\mathbf{X}_c = \operatorname{sgn}\left(\mathbf{X}_1 + \mathbf{X}_2 + \cdots + \mathbf{X}_n\right), \tag{16}$$

where n is the number of artificial fishes.

(5) The fitness function for AFSA is the criterion of OMD given by

$$f(\mathbf{X}) = 2\mathbf{X}^T\mathbf{A}\mathbf{y} - \mathbf{X}^T\mathbf{ARAX}, \tag{17}$$

where $\mathbf{X}$ is the state of a certain artificial fish.

(6) Equations (11), (14), and (15) are, respectively, modified as follows:

$$\begin{aligned} \mathbf{X}_{i\text{next}} &= \mathbf{X}_j, \\ \mathbf{X}_{i\text{next}} &= \mathbf{X}_c, \\ \mathbf{X}_{i\text{next}} &= \mathbf{X}_{\max}. \end{aligned} \tag{18}$$

3.3. The Procedure of the Proposed MMSE-AFSA Detector. Since AFSA is a random searching swarm intelligence algorithm, the initial values have a great effect on its convergence speed. This suggests that, in order to decrease the number of iterations, the initial states of these artificial fishes should be selected with the a priori knowledge rather than selected randomly. Therefore, a novel MMSE-AFSA detector is proposed here. The result of MMSE and its modified formations are used as the initial values of artificial fishes. The initialization of artificial fishes is described below.

Step 1. Execute MMSE detector to get a suboptimal solution. Assign the result $\mathbf{b}_1 = (b_{11}, b_{12}, \ldots, b_{1K})^T$ as the initial state of the first artificial fish, where $b_{1i} \in \{+1, -1\}$ and $i = 1, 2, \ldots, K$.

Step 2. Then, randomly change an element b_{1i} of $\mathbf{b}_1$; that is, let $b_{2i} = -b_{1i}$. And assign the new state $\mathbf{b}_2$ which is modified from $\mathbf{b}_1$ to another artificial fish.

Step 3. Repeat Step 2 and initialize the rest of the artificial fishes in the same way.

After initialization, run AFSA to get the optimal solution of multiuser detection. As described above, the overall structure of MMSE-AFSA detector is shown in Figure 2.

3.4. Convergence Analysis of the Proposed Algorithm. After each iteration in this algorithm, preying behavior obviously provides a better solution than the previous solution; swarming behavior improves the state of each artificial fish in their own visual range; artificial fishes move towards the optimal state within their visual range after following behavior. The behavior selection described in Section 3.1.4 chooses the best behavior after each alteration. All these processes are beneficial to the convergence of the proposed algorithm.

Besides, appropriate parameters have great influence on the convergence of the algorithm. A smaller *try_number* helps artificial fishes to avoid local optimum and move towards global optimal solution. Artificial fishes are easier to find global optimal solution with a bigger *Visual*, whereas smaller *try_number* and bigger *Visual* usually mean higher computational complexity.

Table 1: Simulation parameters.

Parameter	Value
Communication link	Forward link
Multiple access	CDMA
Modulation	QPSK
Spreading codes	Gold sequences
Length of spreading codes	1023
Communication channel	AWGN
Number of tested information bits	1,000,000
Number of artificial fishes	3
Visual	4
try_number	3
Number of iterations	5

The number of artificial fishes also affects the performance of the algorithm. With more artificial fishes, the algorithm is easier to converge and achieve global optimum. However, the price is higher computational complexity.

So appropriate parameters and proper number of artificial fishes are beneficial to the convergence of the proposed algorithm.

4. Simulations and Discussions

In this Section, Monte Carlo simulations are utilized to verify the proposed MMSE-AFSA detector. And the performances of MF, DEC, MMSE, OMD, and MMSE-AFSA are compared over AWGN channel. Most of the parameters used for these simulations are summarized in Table 1.

4.1. The BER Performance versus E_b/N_0. The BER performance versus E_b/N_0 with perfect power control over AWGN channel is shown in Figure 3. There are 10 users in the TDRS system and E_b/N_0 ranges from 0 to 10.

It can be easily seen from Figure 3 that the BER performance versus E_b/N_0 of MMSE-AFSA is superior compared with MF, DEC, and MMSE. In addition, it even coincides with OMD. MMSE is a suboptimal method of multiuser detection, and AFSA can efficiently find the optimal solution with the result of MMSE and its modified formations as initial states of artificial fishes. Rather than random initial values, MMSE and its modified formations are approximations of the optimal solution. That is the reason why the BER performance of MMSE-AFSA is quite close to OMD and why only 5 iterations are needed in MMSE-AFSA.

4.2. The BER Performance versus Number of Users K. The BER performance curves of these detectors with different number of active users are explored here. In this experiment, E_b/N_0 is set to 5 for all the detectors.

Figure 4 shows the simulation results. As an overall trend, BER of all the detectors increases when there are more active users in the system. OMD shows the best BER performance versus the number of active users among all these detectors. The performance of MMSE-AFSA is also better than MF, DEC, and MMSE. In this experiment, as the number of users increases, the solution space expands, while the parameters such as *Visual* and *try_number* of AFSA remain unchanged. Thus, there exists a gap between MMSE-AFSA and OMD.

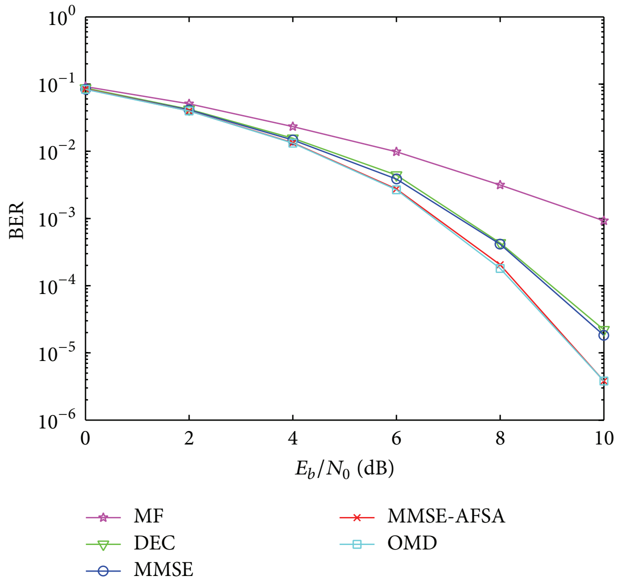


Figure 3: The BER performance versus E_b/N_0.

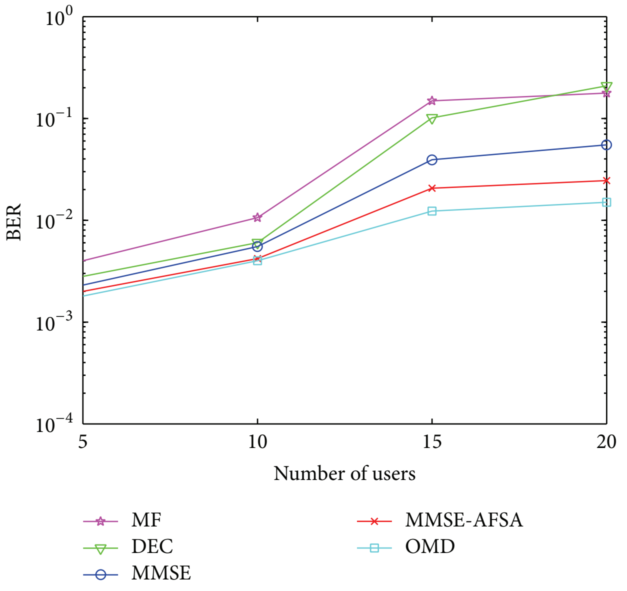


Figure 4: The BER performance versus user number K.

4.3. The Near-Far Resistance of MMSE-AFSA. In this experiment, the BER performance of these detectors with imperfect power control is employed. The user number is set to 10 and E_b/N_0 of the first user is 5. While E_b/N_0 of the remaining users changes from 1 to 10 simultaneously. Simulation results, compared with MF, DEC, MMSE, and OMD, are shown in Figure 5.

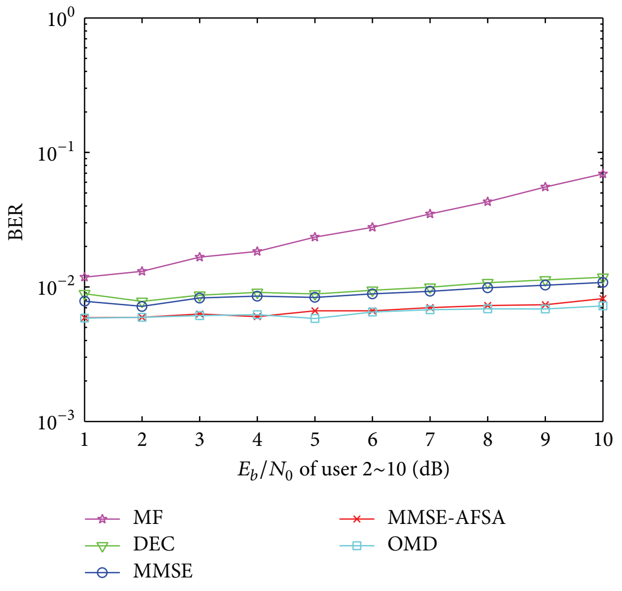


Figure 5: The near-far resistance of different detectors.

Table 2: Relative computational complexity of different detectors.

MF	DEC	MMSE	MMSE-AFSA	OMD
1	1.42	1.67	2.13	76.9

As is revealed in Figure 5, OMD shows the best near-far resistance, while MF shows the worst. The near-far resistance performance of MMSE-AFSA is better than MF, DEC, and MMSE. MMSE-AFSA takes advantages from the suboptimal result of MMSE and AFSA is expert in optimization and it can realize the global convergence efficiently.

4.4. Different Initial States of Artificial Fishes. As an iterative optimization scheme, the convergence rate reflects the computational complexity. Different initial values have great influence on iteration times. In this experiment, an AFSA detector whose initial values are generated randomly and the MMSE-AFSA detector whose initial values are the result of MMSE and its modified formations are discussed. The BER performance of different iteration times is shown in Figure 6, respectively. The number of users is 10.

From Figure 6, we can see that, even after only 5 times of iterative in MMSE-AFSA detector, the BER performance is quite close to OMD. However, the BER performance of AFSA detector with random initial states is worse than MMSE-AFSA despite the number of iterations being 30. It is because that there are only 3 artificial fishes, and they cannot reach global optimum easily with randomly selected initial values.

4.5. Computational Complexity Analysis. In order to measure the computational complexity of these detectors, relative execution time is used in this experiment. Let the execution time of MF be equal to 1; the relative execution time of DEC, MMSE, MMSE-AFSA, and OMD is shown in Table 2, respectively (suppose there are 10 active users and $E_b/N_0 = 5$; other simulation parameters are the same as shown in Table 1).

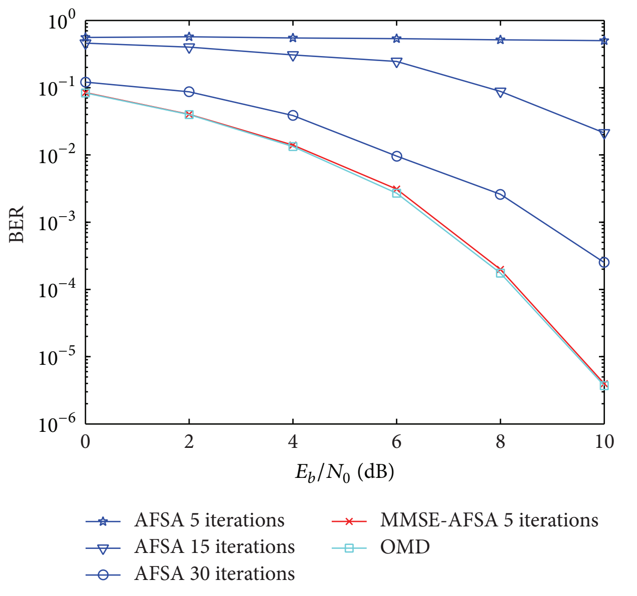


Figure 6: Convergence rate of AFSA detector and MMSE-AFSA detector.

It can be seen from Table 2 that the computational complexity of MMSE-AFSA increases slightly compared with MMSE and is much lower than that of OMD, because, in a K-user TDRS system forward link, the number of iterations of OMD is 2^K for OMD is known to be NP-hard. DEC and MMSE are linear detectors so that they have a low computational complexity. The computational complexity of MMSE-AFSA contains two parts. The first part is the computational complexity of MMSE; another part is the complexity of AFSA. From Section 4.4, we can see that only 3 artificial fishes and 5 iterations are needed for MMSE-AFSA to coincide performance of OMD.

5. Conclusion

In this paper, a hybrid multiuser detector based on MMSE and AFSA in TDRS system forward link is explored. In order to apply AFSA in multiuser detection, the discretization of AFSA is employed. Then the result of MMSE and its modified formations are used as the initial values of discrete artificial fishes. Simulation results demonstrate that the BER performance, user capacity, near-far resistance, and computational complexity of MMSE-AFSA are superior, compared with MF, DEC, and MMSE, and are quite close to OMD. Besides, the convergence rate of the novel MMSE-AFSA detector is much quicker than AFSA detector.

Conflict of Interests

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

Acknowledgments

The research in this paper is supported by the National Natural Science Foundation of China (Grant no. 61102084), the Foundation of China Academy of Space Technology (CAST), and the China Postdoctoral Science Foundation (Grant no. 2011M500665).

References

[1] R. A. Stampfl and A. E. Jones, "Tracking and data relay satellites," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-6, no. 3, pp. 276–289, 1970.

[2] S.-W. Cheng, Z. Hui, W. Chao, C. Jing, and J.-S. Zhou, "Operation planning modeling of tracking and data relay satellite: a sketch," in *Proceedings of the 2nd International Conference on Intelligent Computing Technology and Automation (ICICTA '09)*, pp. 144–147, October 2009.

[3] J. Teles, M. V. Samii, and C. E. Doll, "Overview of TDRSS," in *Proceedings of the PSD Meeting of the COSPAR Technical Panel on Satellite Dynamics, at the 30th COSPAR Scientific Assembly*, vol. 16 of *no. 12*, pp. 67–76, 1995.

[4] S. Verdu, "Minimum probability of error for asynchronous gaussian multiple-access channels," *IEEE Transactions on Information Theory*, vol. IT-32, no. 1, pp. 85–96, 1986.

[5] B. Shim and B. Song, "Multiuser detection via compressive sensing," *IEEE Communication Letters*, vol. 16, no. 7, pp. 972–974, 2012.

[6] L. Hu, X. Zhou, and L. Zhang, "Blind multiuser detection based on tikhonov regularization," *IEEE Communications Letters*, vol. 15, no. 5, pp. 482–484, 2011.

[7] N. Zhao, Z. Wu, Y. Zhao, and T. Quan, "Population declining ant colony optimization multiuser detection in asynchronous CDMA communications," *Wireless Personal Communications*, vol. 62, no. 4, pp. 783–792, 2012.

[8] A. Zahedi and H. Bakhshi, "Multiuser detection based on adaptive LMS and modified genetic algorithm in DS-CDMA systems," *Wireless Personal Communications*, vol. 73, no. 3, pp. 931–947, 2013.

[9] J. Ding, J. Shao, Y. Huang, L. Sheng, W. Fu, and Y. Li, "Swarm intelligence based algorithms for data clustering," in *Proceedings of the International Conference on Computer Science and Network Technology (ICCSNT '11)*, pp. 577–581, Haerbin, China, December 2011.

[10] J. Krause, G. D. Ruxton, and S. Krause, "Swarm intelligence in animals and humans," *Trends in Ecology and Evolution*, vol. 25, no. 1, pp. 28–34, 2010.

[11] D. Merkle and M. Middendorf, "Swarm intelligence and signal processing: DSP exploratory," *IEEE Signal Processing Magazine*, vol. 25, no. 6, pp. 152–158, 2008.

[12] M. Dorigo, M. Birattari et al., "Swarm intelligence," in *Proceedings of the 7th International Conference (ANTS '10)*, Springer, Brussels, Belgium, September 2010.

[13] N. Zhao, Z. Wu, Y. Zhao, and T. Quan, "A population declining mutated ant colony optimization multiuser detector for MC-CDMA," *IEEE Communications Letters*, vol. 14, no. 6, pp. 497–499, 2010.

[14] M. A. S. Chaudhry, M. Zubair, and I. M. Qureshi, "Particle swarm optimization based MUD for overloaded MC-CDMA system," in *Proceedings of the IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS '10)*, pp. 1–5, Beijing, China, June 2010.

[15] V. K. Pamula, S. R. Vempati et al., "Multiuser detection for DS-CDMA systems over Nakagami-m fading channels using particle swarm optimization," in *Proceedings of the IEEE 9th International Colloquium on Signal Processing and its Applications (CSPA '13)*, pp. 275–279, 2013.

[16] N. Liu, F. Zheng, and K. Xia, "CDMA multiuser detection based on improved particle swarm optimization algorithm," *Applied Mechanics and Materials*, vol. 50-51, pp. 3–7, 2011.

[17] X. L. Li, *A new intelligent optimization—artificial fish swarm algorithm [Ph.D. thesis]*, Zhejiang University, 2003.

[18] Y. Q. Zhou, H. J. Huang, and J. L. Zhang, "Hybrid artificial fish swarm algorithm for solving III-conditioned linear systems of equations," *International Computing and Information Science*, vol. 134, pp. 656–661, 2011.

[19] S. K. Liu, N. Dong, Z. Zheng et al., "Application of modified artificial fish swarm algorithm in power system reactive power optimization," *Mechatronics and Industrial Informatics, PTS 1–4*, vol. 321–324, pp. 1361–1364, 2013.

[20] Y. Xu and H. A. Chen, "A novel global artificial fish swarm algorithm with improved chaotic search," *Materials Processing Technology , PTS 1–4*, vol. 538–541, pp. 2594–2597, 2012.

[21] D. Yazdani, A. N. Toosi, and M. R. Merbodi, "Fuzzy adaptive artificial fish swarm algorithm," in *Proceedings of the Advances in Artificial Intelligence (AI '10)*, vol. 6464, pp. 334–343, 2010.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 214615, 10 pages
http://dx.doi.org/10.1155/2014/214615

Research Article

Hybrid Metaheuristics for Solving a Fuzzy Single Batch-Processing Machine Scheduling Problem

S. Molla-Alizadeh-Zavardehi,[1] R. Tavakkoli-Moghaddam,[2] and F. Hosseinzadeh Lotfi[3]

[1] *Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*
[2] *School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran*
[3] *Department of Mathematics, Science and Research Branch, Islamic Azad University, Tehran, Iran*

Correspondence should be addressed to S. Molla-Alizadeh-Zavardehi; saber.alizadeh@gmail.com

Received 14 February 2014; Accepted 30 March 2014; Published 22 April 2014

Academic Editors: P. Agarwal, V. Bhatnagar, and Y. Zhang

Copyright © 2014 S. Molla-Alizadeh-Zavardehi 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 deals with a problem of minimizing total weighted tardiness of jobs in a real-world single batch-processing machine (SBPM) scheduling in the presence of fuzzy due date. In this paper, first a fuzzy mixed integer linear programming model is developed. Then, due to the complexity of the problem, which is NP-hard, we design two hybrid metaheuristics called GA-VNS and VNS-SA applying the advantages of genetic algorithm (GA), variable neighborhood search (VNS), and simulated annealing (SA) frameworks. Besides, we propose three fuzzy earliest due date heuristics to solve the given problem. Through computational experiments with several random test problems, a robust calibration is applied on the parameters. Finally, computational results on different-scale test problems are presented to compare the proposed algorithms.

1. Introduction

A batch-processing machine (BPM) is a special variant of a scheduling problem, in which several jobs can be simultaneously processed in such a way that all the jobs in a batch start and complete their processing at the same time. The main advantage is to reduce setups and/or facilitation of material handling. The problem of BPM scheduling is often encountered in real industries. The industrial application of these machines can be found in semiconductor burn-in operations, environmental stress-screening (ESS) chambers, chemical, food, and mineral processing, pharmaceutical and construction materials industries, and so forth.

The BPM scheduling problem is important because the scheduling of batching operations has a significant economic impact. It is mainly motivated by an industrial application, namely, the burn-in operation found in the final testing phase in semiconductor manufacturing [1, 2]. In the semiconductor manufacturing, the jobs have different processing times and sizes that are both required by the customers. The jobs are grouped in batches where a batch means a subset of jobs. The BPM can process a batch of jobs as long as the sum of all the job sizes in the batch does not violate the capacity of the machine. The processing time of a batch is equal to the longest processing time of all the jobs in that batch.

Ikura and Gimple [3] were the first researchers who studied the BPM problem and Lee et al. [4] first presented a detailed description for burn-in operation. As reported in the studies, the exact algorithms have a slow convergence rate and they can solve only small instances to optimality.

As this study addresses SBPM with fuzzy due dates using metaheuristics, the review on SBPM scheduling under a fuzzy environment and the application of metaheuristics to these problems is carried out. For an extensive review on BPM scheduling problems, we refer to Potts and Kovalyov [5] and Mathirajan and Sivakumar [6].

In BPM scheduling problems, Wang and Uzsoy [7] firstly proposed a metaheuristic algorithm. Considering dynamic job arrivals, they combined a dynamic programming algorithm with a random key genetic algorithm (GA) to minimize the maximum lateness. Melouk et al. [8] used a simulated annealing (SA) to minimize the makespan. Koh et al. [9] proposed a random key representation-based GA for the problems of minimizing the makespan and total weighted completion time. Sevaux and Dauzère-Pérès [10], Husseinzadeh Kashan et al. [11], and Damodaran et al. [12] used a GA and redesigned the coding and decoding methods.

Mönch et al. [13] presented a GA combined with dominance properties to minimize the earliness tardiness of the jobs. Chou et al. [14] and Wang et al. [15] presented a hybrid GA and a hybrid forward/backward approach to minimize the makespan. Kashan and Karimi [16] developed two versions of an ant colony optimization (ACO) framework under the situation considered in Koh et al. [9]. Chou and Wang [17], Mathirajan et al. [18], and Wang [19] proposed a hybrid GA, SA, and iterated heuristic for the objective of the total weighted tardiness, respectively. Husseinzadeh Kashan et al. [20] considered bicriteria scheduling for the simultaneous minimization of the makespan and maximum tardiness.

In the classic scheduling problems, it is usually assumed that the aspects of the problem in hand are certain. Most existing models neglect the presence of uncertainty within a scheduling environment. In many real-world scheduling problems, however, uncertainty and vagueness in due date often do exist that make the models more complex. This uncertainty may come about because of production problems (e.g., defect in raw material and machine malfunctioning) or problems with delivery itself (e.g., transportation delay and traffic jam). Although classic BPM scheduling models are extensively studied in the literature, there are only three studies on fuzzy-based BPM models.

Ishii et al. [21] introduced the concept of fuzzy due dates to scheduling problems; fuzzy due dates scheduling problems have been investigated by many researchers. Harikrishnan and Ishii [22] presented a polynomial time algorithm for bicriteria scheduling of serial-batching problem with fuzzy due dates to minimize the total weighted resource consumption and maximize the minimal satisfaction degree. Yimer and Demirli [23] considered a fuzzy goal programming problem for batch scheduling of jobs in a two-stage flow shop to minimize the total weighted flow time of jobs. Cheng et al. [24] proposed ACO to minimize the fuzzy makespan on an SBPM with triangular fuzzy processing times.

Till now, none has considered the objective of minimizing the fuzzy total weighted fuzzy tardiness penalties. So, a new approach to solve a fuzzy SBPM (FSBPM) is proposed and a related fuzzy number is considered for due dates and modeled by fuzzy sets, in which the corresponding membership functions represent satisfaction degree with respect to jobs' completion times. Hence, for the first time, we present a new programming approach. Since the problem is NP-hard for solving the addressed problem, two hybrid metaheuristics (GA-VNS and VNS-SA) are developed to obtain better results.

The remainder of this paper is as follows. Section 2 describes the problem in detail and presents the fuzzy mathematical model. Section 3 explains the proposed algorithms. Section 4 describes the experimental design and compares the computational results. Finally, conclusions are provided and some areas of further research are then suggested in Section 5.

2. Fuzzy Mathematical Model and Problem Descriptions

2.1. Deterministic Model. The objective of this problem is to minimize the total weighted tardiness penalties. Suppose that there are n jobs to be processed and each job $j \in J$ has a processing time p_j and a corresponding size s_j. The total size of all the jobs in a batch does not exceed machine capacity S. The processing time of a batch b is given by the longest job in the batch (i.e., $P^b = \max\{p_j \mid j \in \text{batch } b\}$). The formulation is as follows.

Notations
Sets

J: Jobs, $j \in J$

B: Batches, $b \in B$.

Parameters

p_j: Processing time of job j

s_j: Size of job j

cap: Machine capacity

β_j: Tardiness penalty (/unit/h) of job j

d_j: Due date of job j.

Decision Variables

X_{jb}: A binary variable indicates the assignment of job j to batch b

P^b: Processing time of batch b

c_j: Completion time of job j

C^b: Completion time of batch b

T_j: Tardiness of job j.

According to the mentioned sets, parameters, and decision variables, the mathematical formulation of the total weighted tardiness penalties can be written below:

$$\text{Min} \quad Z = \sum_{j\in J} \beta_j T_j s_j \tag{1}$$

$$\text{s.t:} \quad \sum_{b\in B} X_{jb} = 1 \quad \forall j \in J \tag{2}$$

$$\sum_{j\in J} s_j X_{jb} \leq \text{cap} \quad \forall b \in B \tag{3}$$

$$P^b \geq p_j X_{jb} \quad \forall j \in J,\ \forall b \in B \tag{4}$$

$$C^b = \sum_{i=1}^{b} P^i \quad \forall b \in B \tag{5}$$

$$c_j \geq C^b - M\left(1 - X_{jb}\right) \quad \forall j \in J,\ \forall b \in B; \quad M \text{ is a very large positive number} \tag{6}$$

$$T_j \geq c_j - d_j \quad \forall j \in J \tag{7}$$

$$X_{jb} \in \{0,1\} \quad \forall j \in J,\ \forall b \in B. \tag{8}$$

The objective function is to minimize the total weighted tardiness penalties of jobs. Constraint set (2) ensures that each job can be processed in only one batch. Constraint set (3) ensures that the machine capacity is not exceeded when jobs are assigned to a batch. Constraint set (4) states that the processing time of a batch is the longest processing time among all the jobs in that batch. Constraint set (5) determines the completion time of each batch. Constraint set (6) defines the completion time of each job as the completion time of the batch that it is processed in. Constraint set (7) defines the tardiness of a job as the difference between the due date of a job and its completion time or 0 if it is negative. Constraint set (8) specifies the type of decision variable X_{jb}.

Due to minimization of just only tardiness or total weighted tardiness penalties in the objective function, the model chooses the minimum P^b in the constraint sets (4) to reach the longest processing time among all the jobs in that batch. The smaller the completion time of jobs, the more desirable the objective function. Similarly, the model finds the minimum c_j and T_j in the constraint sets (6) and (7).

2.2. Fuzzy Model. We briefly introduce some basic concepts and results about fuzzy measure theory.

Definition 1. If X is a collection of objects denoted generically by x, then a fuzzy set in X is a set of the ordered pairs:

$$\tilde{d} = \left\{x, \tilde{d}(x) \mid x \in X\right\}, \tag{9}$$

where $\tilde{d}(x)$ is called the membership function that is associated with each $x \in X$ a number in $[0,1]$ indicating to what degree x is a number.

Definition 2. $\tilde{d}_j = \{d_{j,l}, d_{j,u}\}$ denotes a fuzzy number as shown in Figure 1.

As mentioned in the literature, the concept of fuzzy due dates has been used in scheduling problems. Here, this concept is being firstly utilized in the BPM scheduling problem. In a fuzzy due date, the membership function assigned to each job represents the customer satisfaction degree for the delivery or completion time of that job. The membership function of a fuzzy due date of a job is represented below:

$$\mu_j\left(C_j\right) = \begin{cases} 1 & \text{if } c_j \leq d_{j,l} \\ \dfrac{d_{j,u} - c_j}{d_{j,u} - d_{j,l}} & \text{if } d_{j,l} < c_j < d_{j,u} \\ 0 & \text{if } c_j \geq d_{j,u}. \end{cases} \tag{10}$$

From Figure 1, we can see that the full satisfaction (i.e., $\mu_j(C_j) = 1$) is attained if $c_j \leq d_{j,l}$, and the satisfaction grade is positive if $d_{j,l} < c_j < d_{j,u}$ in the membership function (8). If $d_{j,l} = d_{j,u}$, the fuzzy due date is transformed to interval due date or due window.

According to the mentioned fuzzy due date, the studied problem can be formulated as a maximization problem of the total degree of satisfaction over given jobs or equivalently a minimization problem of the total degree of dissatisfaction. For the fuzzy mathematical formulation, the objective function (11) and constraint sets (12) and (13) are replaced instead of objective function (1) and constraint set (7) to calculate the total degree of satisfaction:

$$\text{Max } Z = \sum_{j \in J} w_j \mu_j s_j \tag{11}$$

$$\mu_j = 1 \quad \text{if } c_j \leq d_{j,l} \tag{12}$$

$$\mu_j = \frac{d_{j,u} - c_j}{d_{j,u} - d_{j,l}} \quad \text{if } d_{j,l} < c_j < d_{j,u}. \tag{13}$$

We can also use the following objective function (20) to calculate the total degree of satisfaction instead of expressions (11)–(13):

$$\begin{aligned} \text{Max } Z = \sum_{j \in J} w_j s_j \Bigg(&\left(\frac{\max\left(0, d_{j,l} - c_j\right)}{d_{j,l} - c_j}\right) \\ &+ \left(\frac{\max\left(0, c_j - d_{j,l}\right)}{c_j - d_{j,l}}\right) \\ &\times \left(\frac{\max\left(0, d_{j,u} - c_j\right)}{d_{j,u} - c_j}\right) \\ &\times \left(\frac{d_{j,u} - c_j}{d_{j,u} - d_{j,l}}\right) \Bigg). \end{aligned} \tag{14}$$

It is clear that $\max(0, d_{j,l} - c_j)/(d_{j,l} - c_j) = 1$ results in $c_j \leq d_{j,l}$, while $\max(0, c_j - d_{j,l})/(c_j - d_{j,l})$ and $\max(0, d_{j,u} - c_j)/(d_{j,u} - c_j) = 1$ result in $d_{j,l} < c_j < d_{j,u}$. For more explanation of $(\max(0, d_{j,l} - c_j)/(d_{j,l} - c_j)) + (\max(0, c_j - d_{j,l})/(c_j - d_{j,l}))(\max(0, d_{j,u} - c_j)/(d_{j,u} - c_j))((d_{j,u} - c_j)/(d_{j,u} - d_{j,l}))$, consider a simple example of $\tilde{d}_j = \{d_{j,l}, d_{j,u}\} = \{4, 7\}$ with different completion times:

$$\text{(1) } c_j = 3 \Longrightarrow \left(\frac{\max(0,4-3)}{4-3}\right) + \left(\frac{\max(0,3-4)}{3-4}\right) \times \left(\frac{\max(0,7-3)}{7-3}\right)\left(\frac{7-3}{7-4}\right) = 1 + 0 \times 1 \times \frac{4}{3} = 1.$$

$$\text{(2) } c_j = 5 \Longrightarrow \left(\frac{\max(0,4-5)}{4-5}\right) + \left(\frac{\max(0,5-4)}{5-4}\right) \times \left(\frac{\max(0,7-5)}{7-5}\right)\left(\frac{7-5}{7-4}\right) = 0 + 1 \times 1 \times \frac{2}{3} = \frac{2}{3}.$$

$$\text{(3) } c_j = 8 \Longrightarrow \left(\frac{\max(0,4-8)}{4-8}\right) + \left(\frac{\max(0,8-4)}{8-4}\right) \times \left(\frac{\max(0,7-8)}{7-8}\right)\left(\frac{7-8}{7-4}\right) = 0 + 1 \times 0 \times \frac{-1}{3} = 0. \quad (15)$$

As mentioned above, similar to the expressions (11)–(14), expressions (1) and (16)–(18) can be used for the equivalent fuzzy mathematical formulation of the total degree of dissatisfaction as follows:

$$T_j = \frac{c_j - d_{j,l}}{d_{j,u} - d_{j,l}} \quad \text{if } d_{j,l} < c_j < d_{j,u} \quad (16)$$

$$T_j = 1 \quad \text{if } c_j \geq d_{j,u} \quad (17)$$

$$\text{Min } Z = \sum_{j\in J} \beta_j s_j \left(\left(\frac{\max\left(0, c_j - d_{j,l}\right)}{c_j - d_{j,l}} \right) \times \left(\frac{\max\left(0, d_{j,u} - c_j\right)}{d_{j,u} - c_j} \right) \times \left(\frac{c_j - d_{j,l}}{d_{j,u} - d_{j,l}} \right) + \left(\frac{\max\left(0, c_j - d_{j,u}\right)}{c_j - d_{j,u}} \right) \right). \quad (18)$$

Linearization. Obviously, the proposed fuzzy model is a nonlinear mathematical model because of the conditional expressions in the constraint sets (12), (13), (16), and (17). Also, multiplication of variables and max function in the objective functions (14) and (18) are used. An attempt is made in this part to linearize the fuzzy model via introducing binary variable. Hence, the following constraints should be used instead of nonlinear constraint sets (12) and (13):

$$d_{j,u} - c_j \geq M\left(y_j - 1\right) \quad \forall j \in J \quad (19)$$

$$\mu_j \leq \frac{d_{j,u} - c_j}{d_{j,u} - d_{j,l}} + M\left(1 - y_j\right) \quad \forall j \in J \quad (20)$$

$$\mu_j \leq y_j \quad \forall j \in J \quad (21)$$

$$y_j \in \{0, 1\} \quad \forall j \in J. \quad (22)$$

Similarly, the constraint (22), following objective function and constraints, should be used instead of objective function (1) and nonlinear constraint sets (16) and (17):

$$\begin{aligned} &\text{Min } Z = \sum_{j\in J} \beta_j \left(T_{j,1} + T_{j,2}\right) s_j \\ &c_j - d_{j,u} \leq M T_{j,1} \quad \forall j \in J \\ &d_{j,u} - c_j \leq M y_j \quad \forall j \in J \\ &T_{j,2} \geq \frac{c_j - d_{j,l}}{d_{j,u} - d_{j,l}} - M\left(1 - y_j\right) \quad \forall j \in J. \end{aligned} \quad (23)$$

3. Solution Approach

The evolutionary computation community has shown for many years significant interest in optimization problems, in particular in the global optimization of real valued problems, for which exact and analytical methods are not productive. These techniques have shown great promise in several real-world applications [25, 26]. Hence, these methods are often utilized in order to solve the problem in a shorter run time.

3.1. Proposed Earliest Due Date Heuristics. In this subsection, we propose three constructive greedy heuristics based on EDD as a well-known heuristic method related to the due date. The details of these proposed heuristics are as follows.

(i) Calculate the index of jobs to be scheduled.
(ii) Sort jobs in increasing order of their index.
(iii) Apply the first-first (FF) heuristic to group jobs into batches.

Accordingly, the details of these three variants of EDDs are as follows:

EDD Algorithm. In this variant, the indexes are equal to the EDD of the respective jobs. The centroid-based distance method is used for ranking fuzzy numbers as follows:

$$\begin{aligned} &\text{Crisp due date } (d) \\ &= \frac{\int_0^{d_l} x\,dx + \int_{d_l}^{d_u} x\left(\left(d_u - x\right)/\left(d_u - d_l\right)\right) dx}{\int_0^{d_l} dx + \int_{d_l}^{d_u} \left(\left(d_u - x\right)/\left(d_u - d_l\right)\right) dx} \\ &= \frac{1}{3}\left(d_l + d_u - \frac{d_l d_u}{d_l + d_u}\right). \end{aligned} \quad (24)$$

The jobs are sorted in increasing order of their crisp due dates. So, the job that has the earliest due date will be allotted first.

EDDL Algorithm. Sort jobs in increasing order of their d_l.

EDDU Algorithm. Sort jobs in increasing order of their d_u.

3.2. Encoding Scheme and Initialization. As mentioned earlier in the literature, the random key (RK) method is used for solving BPM scheduling problems. To generate a sequence by this method, random real numbers between zero and one are generated for each job. By ascending sorting of the value corresponding to each job, the sequence of job is obtained and then the FF heuristic is applied to group the jobs into the batches. After having a permutation and forming the batches, we can use it to compute the objective function value of this solution.

3.3. Hybrid Metaheuristics. Over the last years, considerable research has been conducted in hybrid metaheuristics in the field of optimization. The trade-off between intensification and diversification mechanisms is the main aspect of these algorithms. Generally, metaheuristics can be categorized into two main classes: local search methods and population based methods. Population based methods deal with a set of solutions in every iteration of the algorithm, while local search heuristics only deal with a single solution.

Although local search heuristics only deal with a single solution, it has shown its potential in both exploring and exploiting the promising regions in the search space with high quality solutions such as VNS. On the other hand, the basic scheme of VNS and its extensions requires few and sometimes no parameters. However, it is still prone to inferior solutions due to the limited exploration and exploitation ability.

There are two major approaches to hybridize the VNS with other metaheuristics to improve its performance: hybridizing with a local base metaheuristic and hybridizing with a population based metaheuristic. The first idea is to embed SA into VNS, so that it is replaced with local search, whereas SA in hybrid VNS addresses how to get out of large valleys. Besides, SA acts as the local search method, because it is good at searching the neighborhood of a solution. The three neighborhoods employed are swap, insertion, and inversion.

As one of the most well-known population based methods, genetic algorithm (GA) shows robust performance with various problems. Usually, GA has been proven to be very good at shuffling the solution space or global exploration ability but fail to intensify the search towards promising regions. Nevertheless, GA usually takes more computing efforts to locate the optimal in the region of convergence [27], owing to the lack of local search ability. Therefore, hybridization with local search methods may overcome this weakness and lead to powerful search schemes. So, the second idea is to embed VNS as a local search into GA and may be a likely choice to consider the hybridization of them. In GA, VNS is applied as a local search to a subset of offspring generated by one-point crossover and swap mutation operator to search for better solutions.

Table 1: Test problems characteristics.

Parameters	Levels	Count
Number of jobs (N)	10, 20, 30, 50, 75, 100, 125, 150, 175 and 200	10
Processing time of jobs (P)	Uniform distributions $[1, 10]$, $[1, 20]$	2
Size of jobs (S)	Uniform distributions $[1, 10]$, $[2, 4]$, $[4, 8]$	3
Tardiness cost (T)	$[5, 8]$	1
$d_{j,u}$	Uniform distributions $(0.7 \times \mathrm{BP}, \mathrm{BP})$	1
$d_{j,l}$	Uniform distributions $(0.6 \times d_{j,u}, 0.8 \times d_{j,u})$	1
Cap	10	1
Total number of problem instances		60

4. Computational Experiments

4.1. Instances. To compare the proposed algorithms, some test problems are needed. In this regard, we generate the required data that can affect the performance of the algorithms including the number of jobs (n), range of processing time of jobs (p_j), size of jobs (s_j), tardiness costs (β_i), and due date of jobs (d_j). The crisp due dates in Tavakkoli-Moghaddam et al. [28] test problems are generated from a uniform distribution. We use such procedure with some modifications to adapt the procedure for our problem as follows:

$$\overline{P} = \frac{\sum_{j=1}^{n} p_j}{n},$$
$$\overline{B} = \frac{\sum_{j=1}^{n} s_j}{0.8 \times \mathrm{cap}}, \qquad (25)$$
$$BP = \overline{B} \times \overline{P}.$$

After generating the BP, the $d_{j,l}$, and $d_{j,u}$ are generated as explained in Table 1.

4.2. Parameter Setting. Because of the dependency of metaheuristic algorithms on the correct selection of parameters and operators, we study the behavior of different parameters of proposed algorithms. The parameters of proposed algorithms are as follows: initial temperature (T_0), number of neighborhood search ($n_{\max}$), reduction ratio of temperature (α) population size (*popsize*), crossover percentage (p_c), and mutation probability (p_m). Levels of these factors are illustrated in Table 2.

In order to be fair, the stopping criterion for all algorithms is equal to $6 \times n$ milliseconds. This criterion is sensitive to the problem size. Using this stopping criterion, searching time increases according to the rise in number of jobs. To yield more reliable information and due to having stochastic nature

Table 2: Factors and their levels.

SA, VNS and VNS-SA				GA and GA-VNS		
Parameters	SA levels	VNS levels	VNS-SA levels	Parameters	GA levels	GA-VNS levels
	A(1)—300	—	A(1)—200		A(1)—45	A(1)—30
T_0	A(2)—350	—	A(2)—250	*popsize*	A(2)—50	A(2)—35
	A(3)—400	—	A(3)—300		A(3)—60	A(3)—40
	B(1)—600	A(1)—400	B(1)—400		B(1)—80%	B(1)—80%
$n_{\max}$	B(2)—650	A(2)—450	B(2)—450	p_c	B(2)—85%	B(2)—85%
	B(3)—700	A(3)—500	B(3)—500		B(3)—90%	B(3)—90%
	C(1)—0.91	—	C(1)—0.89		C(1)—0.1	C(1)—0.1
α	C(2)—0.92	—	C(2)—0.9	p_m	C(2)—0.15	C(2)—0.15
	C(3)—0.93	—	C(3)—0.91		C(3)—0.2	C(3)—0.2
						D(1)—300
				$n_{\max}$		D(2)—350
						D(3)—400

of algorithms, we tackle each test problem ten times. Because the scale of objective functions in each instance is different, they cannot be used directly. To solve this problem, the relative percentage deviation (RPD) is used for each instance. The RPD is obtained by the following formula:

$$\text{RPD} = \frac{\text{Alg}_{\text{sol}} - \text{Min}_{\text{sol}}}{\text{Min}_{\text{sol}}} \times 100, \quad (26)$$

where Alg_{sol} and Min_{sol} are the obtained objective value and minimum objective value found from both proposed algorithms for each instance, respectively. So, we use the RPD measure in the proposed algorithms.

After obtaining the results of the test problems, the results are transformed into RPD measures. The RPD measures are averaged and their value is depicted in Figures 2–6. In SA, better robustness happens when parameters T_0, $n_{\max}$ and α are 350, 650, and 0.92, respectively, as depicted in Figure 2. In Figure 3, the RPD measure for the single parameter of VNS ($n_{\max}$) is depicted and the second level or 450 is the best. Also, for hybrid VNS, as illustrated in Figure 4, T_0, $n_{\max}$, and α are defined as 250, 450, and 0.9. In conformity with Figure 5, best magnitude for *popsize*, p_c, and p_m in GA are 50, 85%, and 15. Besides, in accordance with Figure 6, in the proposed GA-VNS, best quantity for *popsize*, p_c, p_m, and $n_{\max}$ are 35, 85%, 15, and 350, respectively.

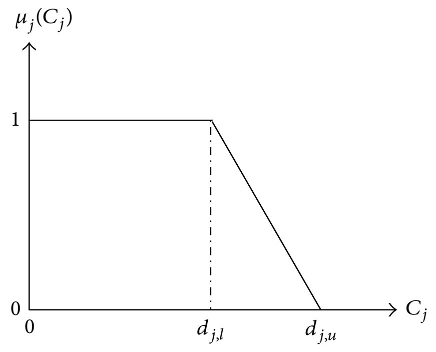


Figure 1: Membership function.

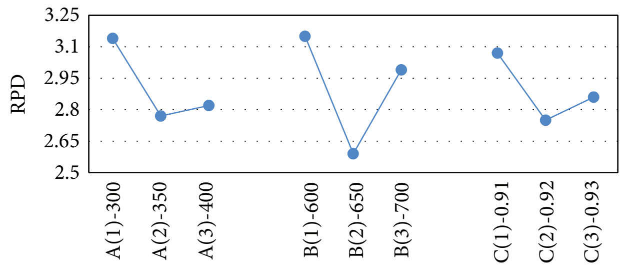


Figure 2: Mean RPD plot for each level of the factors in SA.

4.3. Experimental Results. In this section, we present and compare the results of EDDL, EDDU, SA, VNS, GA, VNS-SA, and GA-VNS with the EDD dispatching rule as a well-known heuristic algorithm related to the due date. As mentioned above, we have 60 problem instances, in which each one includes 10 performed replications to achieve the more reliable results. Table 3 demonstrated the results obtained from EDD, EDDL, and EDDU, in which the first and fourth columns represent the data sets characteristics and the remaining columns show the results on instances.

According to Table 3, among heuristics, EDDL has the worst results, and it can be concluded that EDDU is better than EDD. In order to analyze the interaction between quality of the algorithms and different problem sizes more concisely, the RPD results are calculated for test problems and averaged for each problem size. The average RPDs obtained by each algorithm are shown in Figures 7 and 8. In these figures, each point represents the average results obtained from six test problems considered in each size of problems with ten replications in each algorithm.

It is noticeable that with increasing the problem size, gradually the RPDs of the proposed EDD, EDDL, and EDDU decrease. In spite of decreasing the RPDs, they are not capable to be completive even in the four last sizes.

As it can be seen, the GA-VNS keeps its robust performance in all ranges of problem sizes. On the contrary to SA

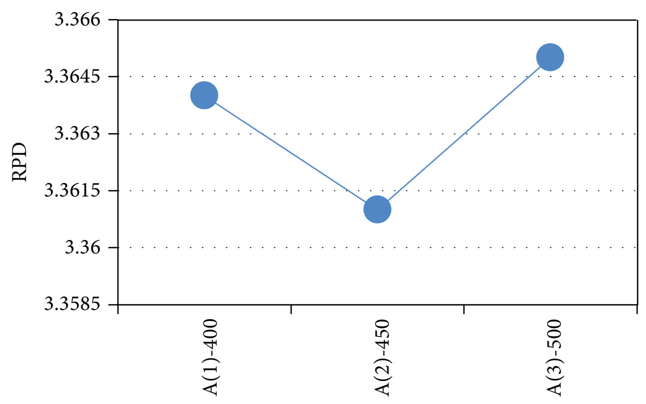


FIGURE 3: Mean RPD plot for each level of the factors in the VNS.

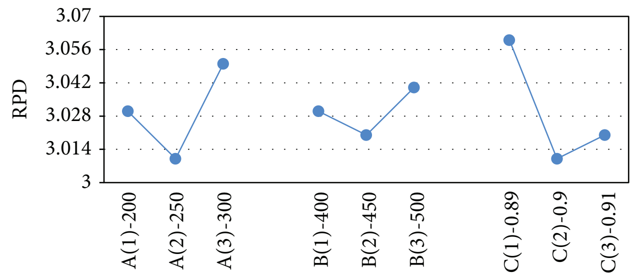


FIGURE 4: Mean RPD ratio plot for each level of the factors in VNS-SA.

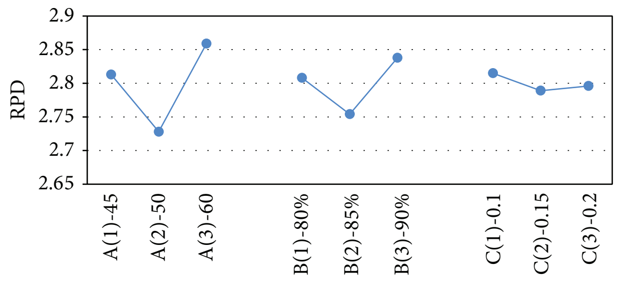


FIGURE 5: Mean RPD ratio plot for each level of the factors in GA.

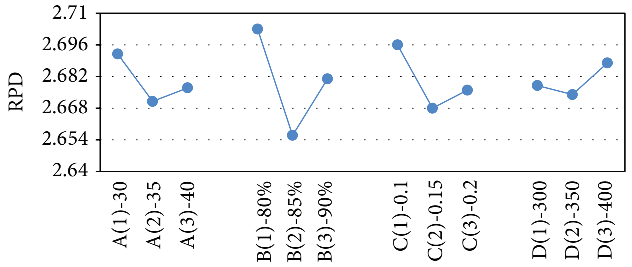


FIGURE 6: Mean RPD ratio plot for each level of the factors in GA-VNS.

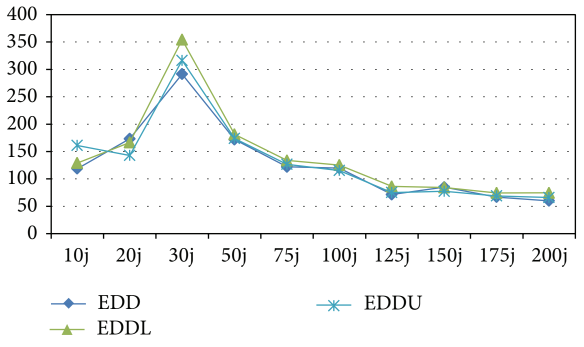


FIGURE 7: Means plot for the interaction among heuristic algorithms.

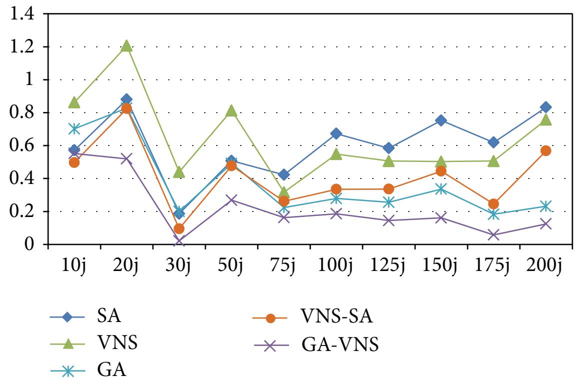


FIGURE 8: Means plot for the interaction among metaheuristic algorithms.

and VNS, both VNS-SA and GA have good results. VNS-SA has better performance in 10 j and 30 j; however, in the last six problem sizes, GA outperforms it. In the first problem size, SA yields the best result, but with increasing the problem size, gradually its RPDs increase. In the first four problem sizes, VNS does not show a good performance, but with increasing the problem size, its RPDs decrease, while in latter sizes it outperforms SA.

From Figures 7 and 8, there is no significant difference between proposed EDD and EDDU or SA and VNS. So, we perform an analysis of variance (ANOVA) to accurately analyze the results among them. The means plot and LSD intervals (at the 95% confidence level) for the presented algorithms are shown in Figures 9 and 10. According to the results, the average RPD obtained by the proposed EDD, EDDL, and EDDU are 128.04, 141, and 132.33 respectively. So, EDD is better than EDDU and EDDL.

Also, the average RPD obtained by the proposed GA-VNS is 0.22, while for GA, VNS-SA, SA, and VNS are 0.37, 0.41, 0.6, and 0.65, respectively. As is evident, GA-VNS has outperformed other algorithm. As it can be seen, between GA and VNS-SA, and also between SA and VNS, there is not a significant difference. However, they failed to statistically overcome each other. However, based on the results, we conclude that the proposed GA-VNS can be used to effectively solve the problem.

Table 3: Results of EDD, EDDL, and EDDU on test problems.

Problem	EDD	EDDL	EDDU
10jp1s1	113.9	**101.15**	113.9
10jp1s2	**140.49**	189.19	**140.49**
10jp1s3	123.16	**91.03**	132.06
10jp2s1	**119.33**	130.42	185.23
10jp2s2	**113.42**	114.74	**113.42**
10jp2s3	**155.56**	166.44	185.35
20jp1s1	203.7	208.63	**189.57**
20jp1s2	196.16	190.16	**175.72**
20jp1s3	256.24	204.22	**203.58**
20jp2s1	**120.96**	134.36	122.45
20jp2s2	214.69	193.68	**162.22**
20jp2s3	153.54	172.27	**153.37**
30jp1s1	195.61	**193.39**	198.37
30jp1s2	252.3	267.01	**252.26**
30jp1s3	**488.24**	772.72	569.51
30jp2s1	**191.91**	234.04	197.27
30jp2s2	**351.17**	362.43	379.27
30jp2s3	**396.51**	400.85	410.41
50jp1s1	**539.52**	686.09	476.37
50jp1s2	424.3	443.42	**398.52**
50jp1s3	708.06	682.57	**701.39**
50jp2s1	**559.46**	591.87	665.65
50jp2s2	**410.91**	423.84	412.4
50jp2s3	527.77	**390.2**	565.44
75jp1s1	761.27	707.47	**701.15**
75jp1s2	594.23	635.03	**515.88**
75jp1s3	**682.1**	774.99	716.68
75jp2s1	**896.54**	1085.56	1117.63
75jp2s2	517.24	**507.16**	507.64
75jp2s3	872.84	**852.19**	917.32
100jp1s1	**573.12**	631.37	655.84
100jp1s2	874.41	932.05	**864.1**
100jp1s3	**1015.52**	1032.15	1052.11
100jp2s1	**991.97**	1047	946.9
100jp2s2	810.87	**716.69**	636.58
100jp2s3	1365.03	1402.12	**1342.32**
125jp1s1	1263.82	1602.84	**1236.04**
125jp1s2	978.41	**910.7**	989.08
125jp1s3	1090.05	1125.03	**1088.53**
125jp2s1	**968.01**	1303.24	1148.23
125jp2s2	962.34	**942.8**	947.46
125jp2s3	1261.83	**1250.44**	1254.45
150jp1s1	1523.47	1473.88	**1199.04**
150jp1s2	1185.04	**1079**	1417.27
150jp1s3	1654.88	1730.38	**1579.02**
150jp2s1	992.93	**992.75**	897.83
150jp2s2	1167.75	1279.56	**1143.29**
150jp2s3	1727.18	1684.63	**1676.66**
175jp1s1	1150.47	1261.93	**1025.12**
175jp1s2	**1303.52**	1414.09	1335.53

Table 3: Continued.

Problem	EDD	EDDL	EDDU
175jp1s3	1888.85	**1805.08**	2013.34
175jp2s1	**1623.61**	1761.51	1876.28
175jp2s2	1408.78	1437.47	**1406.21**
175jp2s3	1651.22	1699.95	**1520.91**
200jp1s1	1198.46	**1142.2**	1160.17
200jp1s2	**1556.72**	1828.91	1724.02
200jp1s3	2266.57	2261.79	**1935.25**
200jp2s1	**1486.81**	1935.52	1912.32
200jp2s2	**1708.57**	1774.06	1731.21
200jp2s3	1892.47	2006.77	**1885.73**

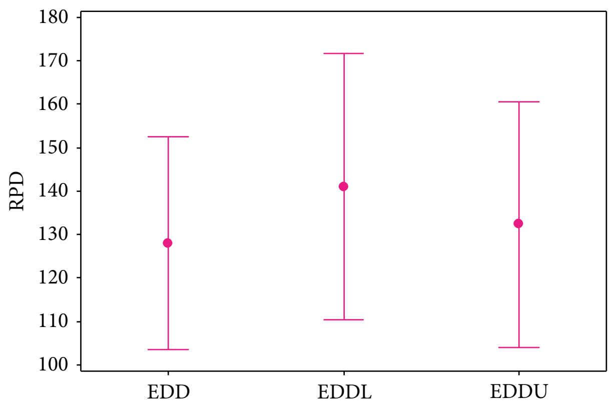


Figure 9: Means plot and LSD intervals for proposed heuristics.

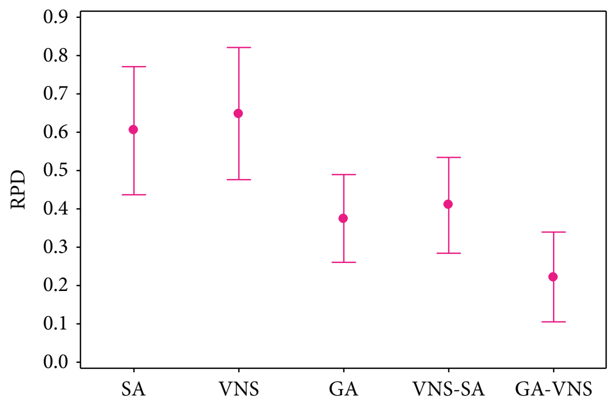


Figure 10: Means plot and LSD intervals for proposed metaheuristics.

5. Conclusions and Future Research Directions

In this paper, we discussed the single batch-processing machine (SBPM) scheduling problem in the presence of fuzzy due date to minimize the total weighted tardiness. We developed a mixed integer linear programming model with the objective functions of the total satisfaction or dissatisfaction degree. To solve this model, three heuristics (EDD, EDDL, and EDDU), three metaheuristics (GA, VNS, and SA), and two hybrid metaheuristics (GA-VNS and VNS-SA) are developed. Also, a plan was developed and utilized to generate test problems in a fuzzy environment. To enhance the performance of the proposed method, the experimental design method was used by setting their parameters. The computational results showed that the hybrid GA-VNS were robust and superior to other proposed algorithms. As a future work, total weighted earliness tardiness can be considered as the objective function and the same proposed algorithms can be developed for it. Another direction is to work on other algorithms, such as Cuckoo Optimization Algorithm [29], Honey Bees Optimization [30], Differential Evolution [31], Cuckoo Search [32], and Firefly Algorithm [25, 26].

Conflict of Interests

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

References

[1] R. Uzsoy, C. Lee, and L. Martin-Vega, "A review of production planning and scheduling models in the semiconductor industry. Part I. System characteristics, performance evaluation and production planning," *IIE Transactions*, vol. 24, pp. 47–60, 1992.

[2] R. Uzsoy, C. Y. Lee, and L. A. Martin-Vega, "Review of production planning and scheduling models in the semiconductor industry part II: shop-floor control," *IIE Transactions*, vol. 26, no. 5, pp. 44–55, 1994.

[3] Y. Ikura and M. Gimple, "Efficient scheduling algorithms for a single batch processing machine," *Operations Research Letters*, vol. 5, no. 2, pp. 61–65, 1986.

[4] C. Y. Lee, R. Uzsoy, and L. A. Martin-Vega, "Efficient algorithms for scheduling semiconductor burn-in operations," *Operations Research*, vol. 40, no. 4, pp. 764–775, 1992.

[5] C. N. Potts and M. Y. Kovalyov, "Scheduling with batching: a review," *European Journal of Operational Research*, vol. 120, no. 2, pp. 228–249, 2000.

[6] M. Mathirajan and A. I. Sivakumar, "A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor," *International Journal of Advanced Manufacturing Technology*, vol. 29, no. 9-10, pp. 990–1001, 2006.

[7] C. S. Wang and R. Uzsoy, "A genetic algorithm to minimize maximum lateness on a batch processing machine," *Computers and Operations Research*, vol. 29, no. 12, pp. 1621–1640, 2002.

[8] S. Melouk, P. Damodaran, and P.-Y. Chang, "Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing," *International Journal of Production Economics*, vol. 87, no. 2, pp. 141–147, 2004.

[9] S. G. Koh, P. H. Koo, D. C. Kim, and W. S. Hur, "Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families," *International Journal of Production Economics*, vol. 98, no. 1, pp. 81–96, 2005.

[10] M. Sevaux and S. Dauzère-Pérès, "Genetic algorithms to minimize the weighted number of late jobs on a single machine," *European Journal of Operational Research*, vol. 151, no. 2, pp. 296–306, 2003.

[11] A. Husseinzadeh Kashan, B. Karimi, and F. Jolai, "Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with nonidentical job sizes," *International Journal of Production Research*, vol. 44, pp. 2337–2360, 2006.

[12] P. Damodaran, P. Kumar Manjeshwar, and K. Srihari, "Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms," *International Journal of Production Economics*, vol. 103, no. 2, pp. 882–891, 2006.

[13] L. Mönch, R. Unbehaun, and Y. I. Choung, "Minimizing earliness-tardiness on a single burn-in oven with a common due date and maximum allowable tardiness constraint," *OR Spectrum*, vol. 28, no. 2, pp. 177–198, 2006.

[14] F. D. Chou, P. C. Chang, and H. M. Wang, "A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 31, no. 3-4, pp. 350–359, 2006.

[15] H. M. Wang, P. C. Chang, and F. D. Chou, "A hybrid forward/backward approach for single batch scheduling problems with non-identical job sizes," *Journal of the Chinese Institute of Industrial Engineers*, vol. 24, no. 3, pp. 191–199, 2007.

[16] A. H. Kashan and B. Karimi, "Scheduling a single batch-processing machine with arbitrary job sizes and incompatible job families: an ant colony framework," *Journal of the Operational Research Society*, vol. 59, no. 9, pp. 1269–1280, 2008.

[17] F. D. Chou and H. M. Wang, "Scheduling for a single semiconductor batch-processing machine to minimize total weighted tardiness," *Journal of the Chinese Institute of Industrial Engineers*, vol. 25, no. 2, pp. 136–147, 2008.

[18] M. Mathirajan, V. Bhargav, and V. Ramachandran, "Minimizing total weighted tardiness on a batch-processing machine with non-agreeable release times and due dates," *International Journal of Advanced Manufacturing Technology*, vol. 48, no. 9-12, pp. 1133–1148, 2010.

[19] H. M. Wang, "Solving single batch-processing machine problems using an iterated heuristic," *International Journal of Production Research*, vol. 49, no. 14, pp. 4245–4261, 2011.

[20] A. Husseinzadeh Kashan, B. Karimi, and F. Jolai, "An effective hybrid multi-objective genetic algorithm for bi-criteria scheduling on a single batch processing machine with non-identical job sizes," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 6, pp. 911–922, 2010.

[21] H. Ishii, M. Tada, and T. Masuda, "Two scheduling problems with fuzzy due-dates," *Fuzzy Sets and Systems*, vol. 46, no. 3, pp. 339–347, 1992.

[22] K. K. Harikrishnan and H. Ishii, "Single machine batch scheduling problem with resource dependent setup and processing time in the presence of fuzzy due date," *Fuzzy Optimization and Decision Making*, vol. 4, no. 2, pp. 141–147, 2005.

[23] A. D. Yimer and K. Demirli, "Fuzzy scheduling of job orders in a two-stage flowshop with batch-processing machines," *International Journal of Approximate Reasoning*, vol. 50, no. 1, pp. 117–137, 2009.

[24] B. Cheng, K. Li, and B. Chen, "Scheduling a single batch-processing machine with non-identical job sizes in fuzzy environment using an improved ant colony optimization," *Journal of Manufacturing Systems*, vol. 29, no. 1, pp. 29–34, 2010.

[25] Y. Zhang, S. Wang, G. Ji, and Z. Dong, "Genetic pattern search and its application to brain image classification," *Mathematical Problems in Engineering*, vol. 2013, Article ID 580876, 8 pages, 2013.

[26] Y. Zhang, L. Wu, and S. Wang, "Solving two-dimensional HP model by firefly algorithm and simplified energy function," *The Scientific World Journal*, vol. 2013, Article ID 398141, 9 pages, 2013.

[27] A. S. Wu, H. Yu, S. Jin, K.-C. Lin, and G. Schiavone, "An incremental genetic algorithm approach to multiprocessor scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 9, pp. 824–834, 2004.

[28] R. Tavakkoli-Moghaddam, A. Rahimi-Vahed, and A. H. Mirzaei, "A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: weighted mean completion time and weighted mean tardiness," *Information Sciences*, vol. 177, no. 22, pp. 5072–5090, 2007.

[29] S. Balochian and E. Ebrahimi, "Parameter optimization via cuckoo optimization algorithm of fuzzy controller for liquid level control," *Journal of Engineering*, vol. 2013, Article ID 982354, 7 pages, 2013.

[30] Y. Celik and E. Ulker, "An improved marriage in honey bees optimization algorithm for single objective unconstrained optimization," *The Scientific World Journal*, vol. 2013, Article ID 370172, 11 pages, 2013.

[31] 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.

[32] Y. Zhou and H. Zheng, "A novel complex valued cuckoo search algorithm," *The Scientific World Journal*, vol. 2013, Article ID 597803, 6 pages, 2013.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 194706, 14 pages
http://dx.doi.org/10.1155/2014/194706

Research Article

Human Behavior-Based Particle Swarm Optimization

Hao Liu,[1,2] Gang Xu,[3] Gui-yan Ding,[2] and Yu-bo Sun[2]

[1] *School of Mathematics and Statistics, Beijing Institute of Technology, Beijing 100081, China*
[2] *School of Science, University of Science and Technology Liaoning, Anshan 114051, China*
[3] *Department of Mathematics, Nanchang University, Nanchang 330031, China*

Correspondence should be addressed to Hao Liu; liuhao123@ustl.edu.cn

Received 3 December 2013; Accepted 17 March 2014; Published 17 April 2014

Academic Editors: P. Agarwal, V. Bhatnagar, and Y. Zhang

Copyright © 2014 Hao Liu 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.

Particle swarm optimization (PSO) has attracted many researchers interested in dealing with various optimization problems, owing to its easy implementation, few tuned parameters, and acceptable performance. However, the algorithm is easy to trap in the local optima because of rapid losing of the population diversity. Therefore, improving the performance of PSO and decreasing the dependence on parameters are two important research hot points. In this paper, we present a human behavior-based PSO, which is called HPSO. There are two remarkable differences between PSO and HPSO. First, the global worst particle was introduced into the velocity equation of PSO, which is endowed with random weight which obeys the standard normal distribution; this strategy is conducive to trade off exploration and exploitation ability of PSO. Second, we eliminate the two acceleration coefficients c_1 and c_2 in the standard PSO (SPSO) to reduce the parameters sensitivity of solved problems. Experimental results on 28 benchmark functions, which consist of unimodal, multimodal, rotated, and shifted high-dimensional functions, demonstrate the high performance of the proposed algorithm in terms of convergence accuracy and speed with lower computation cost.

1. Introduction

Particle swarm optimization (PSO) [1] is a population-based intelligent algorithm, and it has been widely employed to solve various kinds of numerical and combinational optimization problems because of its simplicity, fast convergence, and high performance.

Researchers have proposed various modified versions of PSO to improve its performance; however, there still are premature or lower convergence rate problems. In the PSO research, how to increase population diversity to enhance the precision of solutions and how to speed up convergence rate with least computation cost are two vital issues. Generally speaking, there are four strategies to fulfill these targets as follows.

(1) Tuning control parameters. As for inertial weight, linearly decreasing inertial weight [2], fuzzy adaptive inertial weight [3], rand inertial weight [4], and adaptive inertial weight based on velocity information [5], they can enhance the performance of PSO. Concerning acceleration coefficients, the time-varying acceleration coefficients [6] are widely used. Clerc and Kennedy analyzed the convergence behavior by introducing constriction factor [7], which is proved to be equivalent to the inertial weight [8].

(2) Hybrid PSO, which hybridizes other heuristic operators to increase population diversity. The genetic operators have been hybridized with PSO, such as selection operator [9], crossover operator [10], and mutation operator [11]. Similarly, differential evolution algorithm [12], ant colony optimization [13], and local search strategy [14] have been introduced into PSO.

(3) Changing the topological structure. The global and local versions of PSO are the main type of swarm topologies. The global version converges fast with the disadvantage of trapping in local optima, while the local version can obtain a better solution with slower convergence [15]. The Von Neumann topology is helpful for solving multimodal problems and may perform better than other topologies including the global version [16].

(4) Eliminating the velocity formula. Kennedy proposed the bare-bones PSO (BPSO) [17] and variants of BPSO [18, 19]. Sun et al. proposed quantum-behaved PSO (QPSO) and relative convergence analysis [20, 21].

```
**Initialize Parameters:**
    $N \Leftarrow$ population size;
    $D \Leftarrow$ the dimensionality of search space;
    $T \Leftarrow$ the number of maximum iteration;
    $w \Leftarrow$ the inertial weight;
    $[x^d_{\min}, x^d_{\max}] \Leftarrow$ the allowable position boundaries, $d = 1, 2, \dots, D$;
    $[v^d_{\min}, v^d_{\max}] \Leftarrow$ the allowable velocity boundaries, $d = 1, 2, \dots, D$;
**Initialize Population:** $V_i = (v^1_i, v^2_i, \dots, v^D_i)$, $X_i = (x^1_i, x^2_i, \dots, x^D_i)$,   $i = 1, 2, \dots, N$;
    $v^d_i \Leftarrow v^d_{\min} + \mathrm{rand}^d_i \cdot (v^d_{\max} - v^d_{\min})$;
    $x^d_i \Leftarrow x^d_{\min} + \mathrm{rand}^d_i \cdot (x^d_{\max} - x^d_{\min})$;
**Initialize** $P$best**,** Gbest **and** Gworst**:**
    Evaluate fitness of all particles in $X = \{X_1, X_2, \dots, X_N\}$;
    $P$best $\Leftarrow X$;
    Gbest $\Leftarrow \arg\min \{f(P\mathrm{best}_1), f(P\mathrm{best}_2), \dots, f(P\mathrm{best}_N)\}$;
    Gworst $\Leftarrow \arg\max \{f(P\mathrm{best}_1), f(P\mathrm{best}_2), \dots, f(P\mathrm{best}_N)\}$;
***For*** $t = 1, 2, \dots, T$
    ***For*** each particle $i = 1, 2, \dots, N$
        Update velocity according to (5) and check the boundaries;
        Update position according to (3) and check the boundaries;
    ***Endfor***
    ***Evaluate fitness of all particles in*** $\{X\}$;
    ***Update*** Pbest, Gbest ***and*** Gworst;
***Endfor***.
***Return the best solution***.
```

Algorithm 1: HPSO.

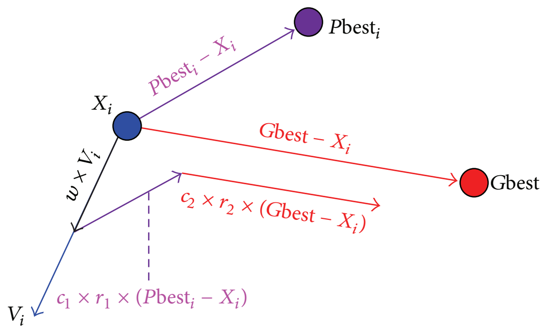

Pbest$_i$
$Pbest_i - X_i$
X_i
$Gbest - X_i$
Gbest
$w \times V_i$
$c_2 \times r_2 \times (Gbest - X_i)$
V_i
$c_1 \times r_1 \times (Pbest_i - X_i)$

Figure 1: Cognition and social terms in PSO.

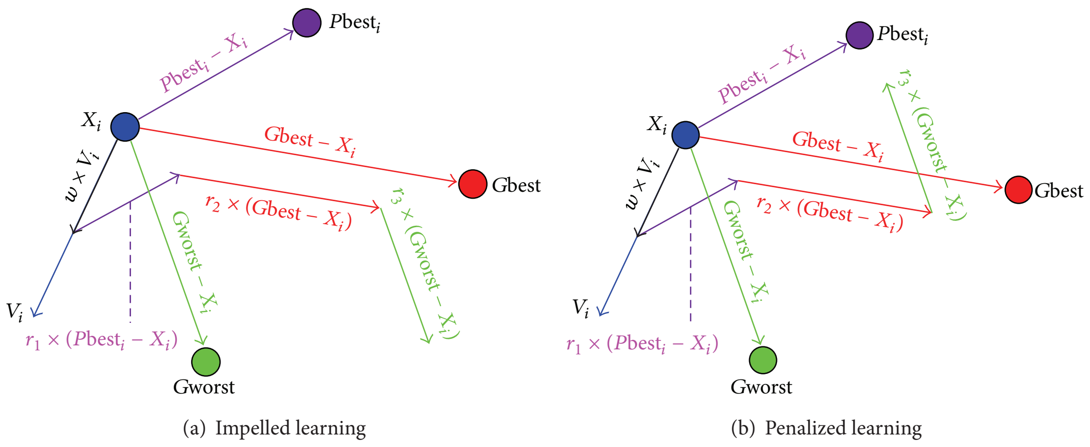


Figure 2: Impelled/penalized term in HPSO.

Table 1: Functions' names, dimensions, ranges, and global optimum values of benchmark functions used in the experiments.

Number	Function name	Dimension (D)	$[\text{Range}]^D$	F_{opt}
F_1	Sphere model	30/50/100	$[-100, 100]^D$	0
F_2	Schwefel's problem 2.22	30/50/100	$[-10, 10]^D$	0
F_3	Schwefel's problem 1.2	30/50/100	$[-100, 100]^D$	0
F_4	Schwefel's problem 2.21	30/50/100	$[-100, 100]^D$	0
F_5	Step function	30/50/100	$[-100, 100]^D$	0
F_6	Quartic function, that is, noise	30/50/100	$[-1.28, 1.28]^D$	0
F_7	Rosenbrock's function	30/50/100	$[-10, 10]^D$	0
F_8	Schwefel's function	30/50/100	$[-500, 500]^D$	0
F_9	Generalized Rastrigin's function	30/50/100	$[-5.12, 5.12]^D$	0
F_{10}	Noncontinuous Rastrigin's function	30/50/100	$[-5.12, 5.12]^D$	0
F_{11}	Ackley's function	30/50/100	$[-32, 32]^D$	0
F_{12}	Generalized Griewank's function	30/50/100	$[-600, 600]^D$	0
F_{13}	Weierstrass's function	30/50/100	$[-0.5, 0.5]^D$	0
F_{14}	Generalized penalized function	30/50/100	$[-50, 50]^D$	0
F_{15}	Cosine mixture problem	30/50/100	$[-1, 1]^D$	$-0.1 \times D$
F_{16}	Rotated elliptic function	30/50/100	$[-1.28, 1.28]^D$	0
F_{17}	Rotated Schwefel's function	30/50/100	$[-500, 500]^D$	0
F_{18}	Rotated Ackley's function	30/50/100	$[-32, 32]^D$	0
F_{19}	Rotated Griewank's function	30/50/100	$[-600, 600]^D$	0
F_{20}	Rotated Weierstrass's function	30/50/100	$[-0.5, 0.5]^D$	0
F_{21}	Rotated Rastrigin's function	30/50/100	$[-5.12, 5.12]^D$	0
F_{22}	Rotated Salomon's function	30/50/100	$[-100, 100]^D$	0
F_{23}	Rotated Rosenbrock's function	30/50/100	$[-100, 100]^D$	0
F_{24}	Shifted Rosenbrock's function	30/50/100	$[-100, 100]^D$	390
F_{25}	Shifted Rastrigin's function	30/50/100	$[-5, 5]^D$	−330
F_{26}	Shifted Schwefel's problem 2.21	30/50/100	$[-100, 100]^D$	−450
F_{27}	Shifted rotated Ackley's function	30/50/100	$[-32, 32]^D$	−140
F_{28}	Shifted rotated Weierstrass's function	30/50/100	$[-0.5, 0.5]^D$	90

In recent years, some modified PSO have extremely enhanced the performance of PSO. For example, Zhan et al. proposed adaptive PSO (APSO) [22] and Wang et al. proposed so-called diversity enhanced particle swarm optimization with neighborhood search (DNSPSO) [23]. The former introduces an evolutionary state estimation (ESE) technique to adaptively adjust the inertia weight and acceleration coefficients. The later ones, a diversity enhancing mechanism and neighborhood-based search strategies, were employed to carry out a tradeoff between exploration and exploitation.

Though all kinds of variants of PSO have enhanced performance of PSO, there are still some problems such as hardly implement, new parameters to just, or high computation cost. So it is necessary to investigate how to trade off the exploration and exploitation ability of PSO and reduce the parameters sensitivity of the solved problems and improve the convergence accuracy and speed with the least computation cost and easy implementation. In order to carry out the targets, in this paper, the global worst position (solution) was introduced into the velocity equation of the standard PSO (SPSO), which is called impelled/penalized learning according to the corresponding weight coefficient. Meanwhile, we eliminate the two acceleration coefficients c_1 and c_2 from the SPSO to reduce the parameters sensitivity of the solved problems. The so-called HPSO has been employed to some nonlinear benchmark functions, which compose unimodal, multimodal, rotated, and shifted high-dimensional functions, to confirm its high performance by comparing with other well-known modified PSO.

The remainder of the paper is structured as follows. In Section 2, the standard particle swarm optimization (SPSO) is introduced. The proposed HPSO is given in Section 3. Experimental studies and discussion are provided in Section 4. Some conclusions are given in Section 5.

2. Standard PSO (SPSO)

The PSO is inspired by the behavior of bird flying or fish schooling; it is firstly introduced by Kennedy and Eberhart in 1995 [1] as a new heuristic algorithm. In the standard PSO (SPSO) [2], a swarm consists of a set of particles, and each particle represents a potential solution of an optimization problem. Considering the ith particle of the swarm with N particles in a D-dimensional space, its position and velocity at iteration t are denoted by $X_i(t) = (x_i^1(t), x_i^2(t), \ldots, x_i^D(t))$ and $V_i(t) = (v_i^1(t), v_i^2(t), \ldots, v_i^D(t))$. Then, the new velocity

Table 2: Experimental results obtained by SPSO and HPSO on function from F_1 to F_{10}.

Fun	Dim		Best	Worst	Meadian	Mean	SD	Significant
F_1	30	SPSO	1.1992*e* − 04	1.0000*e* + 04	9.9690*e* − 04	666.6686	2.5371*e* + 03	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	9.4288*e* − 04	1.0000*e* + 04	0.0078	3.6667*e* + 03	3.6667*e* + 03	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	100	SPSO	1.0013*e* + 04	7.0017*e* + 04	4.0087*e* + 04	4.0698*e* + 04	2.0974*e* + 04	
		HPSO	**0**	**10000**	**0**	**333.3333**	**1.8257*e* + 03**	+
F_2	30	SPSO	6.8555*e* − 04	30.0018	10.0017	11.3364	10.0777	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	0.0329	70.0010	40.0006	37.3438	15.2918	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	100	SPSO	51.0214	181.4054	110.5934	114.3039	29.0723	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
F_3	30	SPSO	6.4613*e* + 03	3.7311*e* + 04	2.2333*e* + 04	2.1337*e* + 04	6.7035*e* + 03	
		HPSO	**0**	**5.1779e + 03**	**0**	**172.5975**	**945.3557**	+
	50	SPSO	4.0023*e* + 04	1.0191*e* + 05	6.5660*e* + 04	7.0328*e* + 04	1.7603*e* + 04	
		HPSO	**0**	**6.9787*e* + 03**	**0**	**232.6222**	**1.2741*e* + 03**	+
	100	SPSO	1.7694*e* + 05	3.0086*e* + 05	2.4789*e* + 05	2.4752*e* + 05	3.6623*e* + 04	
		HPSO	**0**	**2.6987e + 04**	**0**	**3.8008e + 03**	**6.9150e + 03**	+
F_4	30	SPSO	8.6091	21.2711	12.9945	13.3502	3.5341	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	24.2031	39.5127	31.0562	31.1715	4.2886	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	100	SPSO	54.1172	75.3686	64.7834	64.2358	4.2202	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
F_5	30	SPSO	**0**	10001	**0**	1.0005*e* + 03	3.0512*e* + 03	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	**0**	20004	4.5000	5.0028*e* + 03	6.8230*e* + 03	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	100	SPSO	127	90040	40068	4.3086*e* + 04	2.2747*e* + 04	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
F_6	30	SPSO	0.0344	18.8556	0.0959	3.5587	5.1400	
		HPSO	**1.4522e − 04**	**0.0030**	**0.0012**	**0.0012**	**8.5738e − 04**	+
	50	SPSO	0.0780	72.6594	13.6489	19.6604	19.3860	
		HPSO	**7.4623e − 05**	**0.0017**	**5.3645e − 04**	**6.3534e − 04**	**4.7283e − 04**	+
	100	SPSO	86.7855	381.9209	200.8146	211.9720	88.3159	
		HPSO	**3.5210e − 05**	**0.0019**	**2.9387e − 04**	**4.0826e − 04**	**3.5395e − 04**	+
F_7	30	SPSO	**14.3237**	1.0083*e* + 04	140.5176	2.4686*e* + 03	4.2581*e* + 03	
		HPSO	28.6353	**28.9456**	**28.8793**	**28.8461**	**0.0932**	+
	50	SPSO	97.0317	9.4285*e* + 05	376.2306	3.4093*e* + 04	1.7169*e* + 05	
		HPSO	**48.4886**	**48.8766**	**48.7600**	**48.7513**	**0.0875**	+
	100	SPSO	706.1328	2.8333*e* + 06	9.4375*e* + 05	8.8851*e* + 05	8.9157*e* + 05	
		HPSO	**98.4280**	**98.8373**	**98.7133**	**98.7129**	**0.0818**	+
F_8	30	SPSO	**2.0226e + 03**	**4.8935e + 03**	**3.5787e + 03**	**3.6128e + 03**	**733.1063**	
		HPSO	3.5886*e* + 03	8.0516*e* + 03	6.6047*e* + 03	6.3505*e* + 03	1.0893*e* + 03	−
	50	SPSO	**5.8499e + 03**	**9.7913e + 03**	**7.8862e + 03**	**7.7139e + 03**	**1.0101e + 03**	
		HPSO	6.5496*e* + 03	1.4460*e* + 04	1.1191*e* + 04	1.0866*e* + 04	2.1757*e* + 03	−
	100	SPSO	1.8110*e* + 04	**2.4259e + 04**	**2.0949e + 04**	**2.1084e + 04**	**1.7384e + 03**	
		HPSO	**1.2615e + 04**	3.1402*e* + 04	2.4302*e* + 04	2.4077*e* + 04	4.9510*e* + 03	−

Table 2: Continued.

Fun	Dim		Best	Worst	Meadian	Mean	SD	Significant
F_9	30	SPSO	28.7299	160.3815	87.6754	92.5142	32.6994	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	175.2643	351.6480	260.4359	258.0518	48.4078	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	100	SPSO	555.8950	993.3887	750.1694	749.1658	749.1658	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
F_{10}	30	SPSO	61.4129	221.0445	132.7694	134.5414	33.8073	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	157.1020	440.0897	324.2632	310.3595	64.3675	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	100	SPSO	623.5658	1.0433e + 03	804.6981	813.3435	88.5932	
		HPSO	**0**	**25**	**0**	**0.8333**	**4.5644**	+

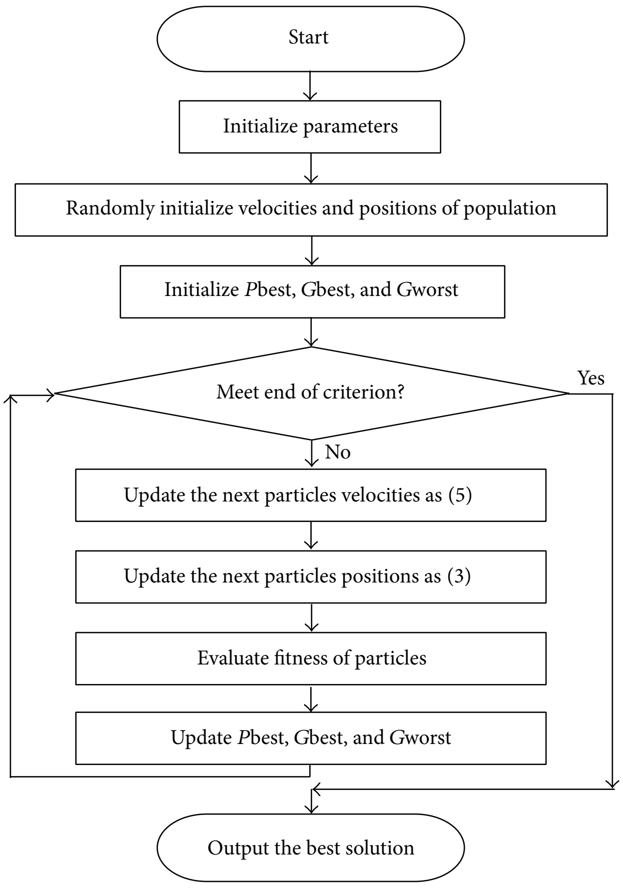

Start
Initialize parameters
Randomly initialize velocities and positions of population
Initialize Pbest, Gbest, and Gworst
Meet end of criterion?
Yes
No
Update the next particles velocities as (5)
Update the next particles positions as (3)
Evaluate fitness of particles
Update Pbest, Gbest, and Gworst
Output the best solution

Figure 3: HPSO flowchart.

and position on the d-dimension of this particle at iteration $t+1$ will be calculated by using the following:

$$v_i^d(t+1) = w \cdot v_i^d(t) + c_1 \cdot r_1^d(t) \cdot \left(P\text{best}_i^d(t) - x_i^d(t)\right) + c_2 \cdot r_2^d(t) \cdot \left(G\text{best}^d(t) - x_i^d(t)\right), \quad (1)$$

where $i = 1, 2, \ldots, N$, and N is the population size; $d = 1, 2, \ldots, D$, and D is the dimension of search space; r_1^d and r_2^d are two uniformly distributed random numbers in the interval $[0, 1]$; acceleration coefficients c_1 and c_2 are nonnegative constants which control the influence of the cognitive and social components during the search process. $P\text{best}_i(t) = (P\text{best}_i^1(t), \ldots, P\text{best}_i^D(t))$, called the personal best solution, represents the best solution found by the ith particle itself until iteration t; $G\text{best}(t) = (G\text{best}^1(t), \ldots, G\text{best}^D(t))$, called the global best solution, represents the global best solution found by all particles until iteration t. w is the inertial weight

TABLE 3: Experimental results obtained by SPSO and HPSO on functions from F_{11} to F_{20}.

Fun	Dim		Best	Worst	Median	Mean	SD	Significant
F_{11}	30	SPSO	0.0043	19.9630	0.0595	2.3935	5.4041	
		HPSO	**8.8818e − 16**	**8.8818e − 16**	**8.8818e − 16**	**8.8818e − 16**	**0**	+
	50	SPSO	0.0598	19.9646	12.6912	10.5673	6.3042	
		HPSO	**8.8818e − 16**	**8.8818e − 16**	**8.8818e − 16**	**8.8818e − 16**	**0**	+
	100	SPSO	15.4237	20.2143	19.5200	19.4135	0.8672	
		HPSO	**8.8818e − 16**	**8.8818e − 16**	**8.8818e − 16**	**8.8818e − 16**	**0**	+
F_{12}	30	SPSO	7.0274e − 04	90.8935	0.0178	12.0794	31.2763	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	0.0014	270.8170	0.0415	45.1971	70.1274	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	100	SPSO	1.1140	721.0594	361.0858	376.1758	158.6584	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
F_{13}	30	SPSO	0.1403	4.3952	0.3210	1.0567	1.4863	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	0.8657	15.2389	7.5828	8.2388	3.6607	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	100	SPSO	27.6235	64.4826	49.3984	47.7138	10.0126	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
F_{14}	30	SPSO	6.4114e − 05	2.2031	0.4202	0.5373	0.5730	
		HPSO	0.0710	0.2803	0.1301	0.1444	0.0513	+
	50	SPSO	0.1882	6.9784	2.2774	2.3889	1.5688	
		HPSO	0.1016	0.3137	0.1652	0.1702	0.0438	+
	100	SPSO	32.5063	5.1200e + 08	457.9143	7.6801e + 07	1.5257e + 08	
		HPSO	0.1866	0.5097	0.2703	0.2736	0.0653	+
F_{15}	30	SPSO	−3.0000	−2.8522	−3.0000	−2.9507	0.0709	
		HPSO	**−3**	**−3**	**−3**	**−3**	**0**	+
	50	SPSO	−5.0000	−2.3044	−4.4827	−4.2127	0.6865	
		HPSO	**−5**	**−5**	**−5**	**−5**	**0**	+
	100	SPSO	−7.9165	4.7637	−5.2127	−4.6977	2.8465	
		HPSO	**−10**	**−10**	**−10**	**−10**	**0**	+
F_{16}	30	SPSO	2.3604e + 03	3.8233e + 04	3.8233e + 04	1.2375e + 04	9.2463e + 03	
		HPSO	0	5.8369e + 03	0	390.6710	1.2756e + 03	+
	50	SPSO	7.1213e + 03	1.0427e + 05	3.3195e + 04	3.4891e + 04	2.2914e + 04	
		HPSO	0	4.0529e + 03	0	224.6749	873.6249	+
	100	SPSO	6.2317e + 04	2.7386e + 05	1.4222e + 05	1.4697e + 05	5.7699e + 04	
		HPSO	0	1.9403e + 04	0	1.0583e + 03	3.8088e + 03	+
F_{17}	30	SPSO	6.7986e + 03	9.7587e + 03	8.3387e + 03	8.2508e + 03	739.7223	
		HPSO	8.3590e + 03	9.8803e + 03	9.0866e + 03	9.0790e + 03	442.4330	−
	50	SPSO	1.3020e + 04	1.7080e + 04	1.4999e + 04	1.5149e + 04	1.0581e + 03	
		HPSO	1.5003e + 04	1.7349e + 04	1.6514e + 04	1.6310e + 04	669.3538	−
	100	SPSO	2.7400e + 04	2.7400e + 04	3.1087e + 04	3.1149e + 04	2.1654e + 03	
		HPSO	3.0329e + 04	3.5493e + 04	3.4226e + 04	3.3586e + 04	1.5320e + 03	−
F_{18}	30	SPSO	20.7888	21.0951	21.0053	20.9848	0.0712	
		HPSO	8.8818e − 16	21.1210	20.9931	11.2354	10.6894	+
	50	SPSO	21.0515	21.2478	21.1455	21.1436	0.0536	
		HPSO	8.8818e − 16	21.2404	21.1366	12.0016	10.6745	+
	100	SPSO	21.2367	21.3931	21.3368	21.3358	0.0364	
		HPSO	8.8818e − 16	21.3949	21.3545	15.6658	9.6084	+

Table 3: Continued.

Fun	Dim		Best	Worst	Median	Mean	SD	Significant
F_{19}	30	SPSO	1.0517	495.3131	273.6408	243.6176	154.3551	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	265.0558	$1.4393e+03$	798.8065	786.0782	289.8401	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	100	SPSO	$1.9937e+03$	$4.0158e+03$	$2.9388e+03$	$2.9263e+03$	543.9053	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
F_{20}	30	SPSO	22.5705	34.8494	28.6842	28.8734	3.5028	
		HPSO	0	39.9834	0	3.1393	9.7817	+
	50	SPSO	45.9462	70.7399	55.5532	55.6014	5.7839	
		HPSO	0	66.4051	0	2.2135	12.1239	+
	100	SPSO	106.4483	139.8394	120.6118	121.4481	7.8030	
		HPSO	0	129.4941	0	8.3487	31.7918	+

to balance the global and local search abilities of particles in the search space, which is given by

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{T} \times t, \tag{2}$$

where $w_{\max}$ is the initial weight, $w_{\min}$ is the final weight, t is the current iteration number, and T is the maximum iteration number. Then, update particle's position using the following:

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{3}$$

and check $x_{\min}^d \leq x_i^d(t+1) \leq x_{\max}^d$, where $x_{\min}^d$ and $x_{\max}^d$ represent lower and upper bounds of the dth variable, respectively.

3. Human Behavior-Based PSO (HPSO)

In this section, a modified version of SPSO based on human behavior, which is called HPSO, is proposed to improve the performance of SPSO. In SPSO, all particles only learn from the best particles Pbest and Gbest. Obviously, it is an ideal social condition. However, considering the human behavior, there exist some people who have bad habits or behaviors around us, at the same time, as we all known that these bad habits or behaviors will bring some effects on people around them. If we take warning from these bad habits or behaviors, it is beneficial to us. Conversely, if we learn from these bad habits or behaviors, it is harmful to us. Therefore, we must give an objective and rational view on these bad habits or behavior.

In HPSO, we introduce the global worst particle, who is of the worst fitness in the entire population at each iteration. It is denoted as Gworst and defined as follows:

$$G\text{worst}(t) = \operatorname{argmax}\left\{f\left(P\text{best}_1\right), f\left(P\text{best}_2\right), \ldots, f\left(P\text{best}_N\right)\right\}, \tag{4}$$

where $f(\cdot)$ represents the fitness value of the corresponding particle.

To simulate human behavior and make full use of the Gworst, we introduce a learning coefficient r_3, which is a random number obeying the standard normal distribution; that is, $r_3 \in N(0,1)$. If $r_3 > 0$, we consider it as an impelled learning coefficient, which is helpful to enhance the "flying" velocity of the particle; therefore, it can enhance the exploration ability of particle. Conversely, if $r_3 < 0$, we consider it as a penalized learning coefficient, which can decrease the "flying" velocity of the particle; therefore, it is beneficial to enhance the exploitation. If $r_3 = 0$, it represents that these bad habits or behaviors have not effect on the particle. Meanwhile, in order to reduce the parameters sensitivity of the solved problems, we take place of the two acceleration coefficients c_1 and c_2 with two random learning coefficients r_1 and r_2, respectively. Therefore, the velocity equation has been changed as follows:

$$\begin{aligned} v_i^d(t+1) = {} & w \cdot v_i^d(t) + r_1(t) \cdot \left(P\text{best}_i^d(t) - x_i^d(t)\right) \\ & + r_2(t) \cdot \left(G\text{best}^d(t) - x_i^d(t)\right) \\ & + r_3(t) \cdot \left(G\text{worst}^d(t) - x_i^d(t)\right), \end{aligned} \tag{5}$$

where r_1 and r_2 are two random numbers in range of $[0,1]$ and $r_1 + r_2 = 1$. The random numbers r_1, r_2, and r_3 are the same for all $d = 1, 2, \ldots, D$ but different for each particle, and they are generated anew in each iteration. If $v_i^d(t+1)$ overflows the boundary, we set boundary value to it. Consider

$$v_i^d(t+1) = \begin{cases} v_{\min}^d, & \text{if } v_i^d(t+1) < v_{\min}^d, \\ v_{\max}^d, & \text{if } v_i^d(t+1) > v_{\max}^d, \\ v_i^d(t+1), & \text{otherwise,} \end{cases} \tag{6}$$

where $v_{\min}^d$ and $v_{\max}^d$ are the minimum and maximum velocity of the d-dimensional search space, respectively. Similarly, if $x_i^d(t+1)$ flies out of the search space, we limit it to the corresponding bound value.

In SPSO, the cognition and social learning terms move particle i towards good solutions based on $P\text{best}_i$ and Gbest in the search space as shown in Figure 1. This strategy makes a particle fly fast to good solutions, so it is easy to trap in

Table 4: Experimental results obtained by SPSO and HPSO on functions from F_{21} to F_{28}.

Fun	Dim		Best	Worst	Median	Mean	SD	Significant
F_{21}	30	SPSO	67.1541	307.3070	213.8939	203.8842	61.8125	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	158.2955	715.0245	518.1705	500.5593	135.5998	
		HPSO	0	269.3463	0	8.9782	49.1757	+
	100	SPSO	1.0850e + 03	1.9021e + 03	1.5793e + 03	1.5669e + 03	190.5584	
		HPSO	0	582.0882	0	35.5882	136.0270	+
F_{22}	30	SPSO	0.7999	14.9999	1.2522	2.9025	4.3553	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	50	SPSO	2.0999	26.0999	13.9628	12.8291	6.9033	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
	100	SPSO	16.5013	41.9999	35.4551	33.9791	6.3075	
		HPSO	**0**	**0**	**0**	**0**	**0**	+
F_{23}	30	SPSO	81.0577	4.0119e + 09	2.0685e + 08	6.8745e + 08	1.0469e + 09	
		HPSO	28.8214	28.9856	28.9323	28.9252	0.0421	+
	50	SPSO	3.7253e + 03	2.1495e + 10	3.6515e + 09	3.6515e + 09	5.3957e + 09	
		HPSO	48.7069	48.8900	48.8205	48.8139	0.0479	+
	100	SPSO	6.7997e + 09	9.2655e + 10	3.3160e + 10	3.8223e + 10	2.0050e + 10	
		HPSO	98.6590	98.8846	98.8109	98.7983	0.0545	+
F_{24}	30	SPSO	6.2312e + 08	2.3418e + 10	4.9110e + 09	5.8767e + 09	5.6099e + 09	
		HPSO	5.9432e + 05	6.2859e + 09	7.6373e + 06	3.7982e + 08	1.2316e + 09	+
	50	SPSO	4.3540e + 09	3.3195e + 10	1.3961e + 10	1.6077e + 10	8.3270e + 09	
		HPSO	3.9454e + 06	8.9387e + 09	3.1766e + 07	7.0962e + 08	1.9565e + 09	+
	100	SPSO	4.9031e + 10	1.5465e + 11	9.1986e + 10	9.7151e + 10	2.8460e + 10	
		HPSO	2.0551e + 08	5.4553e + 09	6.7593e + 08	1.1373e + 09	1.2367e + 09	+
F_{25}	30	SPSO	−229.5551	−78.6646	−176.9746	−174.7148	35.8633	
		HPSO	−204.3636	−100.1465	−148.1389	−149.7299	27.1636	−
	50	SPSO	−77.4305	156.8323	22.8512	24.6168	62.2086	
		HPSO	−102.9219	132.8077	−16.6107	−4.1921	58.2317	+
	100	SPSO	475.3838	860.0386	612.6947	632.8693	100.6069	
		HPSO	394.3532	805.2473	581.1779	590.3932	80.6175	+
F_{26}	30	SPSO	−425.5452	−331.1195	−385.1191	−387.6682	22.2647	
		HPSO	−439.6877	−399.0205	−423.4928	−422.5533	11.3496	+
	50	SPSO	−399.6029	−326.6739	−379.4869	−370.8387	18.7600	
		HPSO	−415.6822	−391.7124	−401.4635	−400.8395	6.5162	+
	100	SPSO	−358.3688	−300.6930	−322.8060	−324.4641	15.5861	
		HPSO	−380.3478	−360.8031	−369.0319	−370.4683	5.1369	+
F_{27}	30	SPSO	−119.2212	−118.8710	−119.0179	−119.0258	0.0866	
		HPSO	−119.1100	−118.8700	−118.9469	−118.9589	0.0545	−
	50	SPSO	−119.0222	−118.7656	−118.8316	−118.8535	0.0603	
		HPSO	−118.9117	−118.7327	−118.7780	−118.7911	0.0421	−
	100	SPSO	−118.7259	−118.6013	−118.6485	−118.6537	0.0310	
		HPSO	−118.6872	−118.5986	−118.6231	−118.6289	0.0204	−
F_{28}	30	SPSO	113.2663	126.0977	118.5782	119.4693	3.6330	
		HPSO	114.4722	132.2305	124.3094	124.5205	4.3399	−
	50	SPSO	137.8303	153.5400	145.1433	145.1503	4.2018	
		HPSO	141.9493	162.4008	153.9547	153.1087	5.4273	−
	100	SPSO	194.1222	232.4306	215.9257	215.9174	8.6772	
		HPSO	212.5258	245.0126	229.4886	230.4426	7.4650	−

Table 5: Some well-known PSOs algorithms in the literature.

Algorithm	Year	Topology	Parameter settings
GPSO	1998	Global star	w: $0.9 - 0.4$, $c_1 = c_2 = 2.0$
LPSO	2002	Local ring	w: $0.9 - 0.4$, $c_1 = c_2 = 2.0$
FIPS	2004	Local Uring	$\chi = 0.729$, $\sum c_i = 4.1$
HPSO-TVAC	2004	Global star	w: $0.9 - 0.4$, c_1 : $2.5 - 0.5$, and c_2: $0.5 - 2.5$
UPSO	2004	Global star	w: $0.9 - 0.4$, $c_1 = c_2 = 2.0$, and $U = 0.5$
DMS-PSO	2005	Dynamic multiswarm	w: $0.9 - 0.2$, $c_1 = c_2 = 2.0$, $m = 3$, and $R = 5$
VPSO	2006	Local Von Neumann	w: $0.9 - 0.4$, $c_1 = c_2 = 2.0$
CLPSO	2006	Comprehensive learning	w: $0.9 - 0.4$, $c = 1.49445$, and $m = 7$
QIPSO	2007	Global star	w: $0.9 - 0.4$, $c_1 = c_2 = 2.0$
APSO	2009	Global star	w: 0.9, $c_1 = c_2 = 2.0$; δ : random in $[0.05, 0.1]$, σ: $1 - 0.1$
AFPSO	2011	Global star	w: $0.9 - 0.4$, c_1, and c_2 are based on fuzzy rule
AFPSO-QI	2011	Global star	w: $0.9 - 0.4$, c_1, and c_2 are based on fuzzy rule

local optima. From Figure 2, we can clearly observe that both impelled learning term and penalized term provide a particle with the chance to change flying direction. Therefore, the impelled/penalized term plays a key role in increasing the population diversity, which is beneficial in helping particles to escape from the local optima and enhance the convergence speed. In HPSO, the impelled/penalized learning term performs a proper tradeoff between the exploration and exploitation.

To sum up, Figure 3 illustrates the flowchart of HPSO. Meanwhile, the pseudocodes of implementing the HPSO are listed as shown in Algorithm 1.

4. Experimental Studies and Discussion

To evaluate the performance of HPSO, 28 minimization benchmark functions are selected [22, 24, 25] as detailed in Section 4.1. HPSO is compared with SPSO in different search spaces and the results are given in Section 4.2. In addition, HPSO is compared with some well-known variants of PSO in Section 4.3.

4.1. Benchmark Functions. In the experimental study, we choose 28 minimization benchmark functions, which consist of unimodal, multimodal, rotated, shifted, and shifted rotated functions. Table 1 lists the main information; please refer to papers [22, 24, 25] to obtain further detailed information about these functions. Among these functions, F_1–F_6 are unimodal functions. F_7 is the Rosenbrock function, which is unimodal for $D = 2$ and $D = 3$ but may have multiple minima in high dimension cases. F_8–F_{15} are unrotated multimodal functions and the number of their local minima increases exponentially with the problem dimension. F_{16}–F_{23} are rotated functions. F_{24}–F_{26} are shifted functions and F_{27} and F_{28} are shifted rotated multimodal functions and $O = (o^1, o^2, \dots, o^D)$ is a randomly generated shift vector located in the search space. To obtain a rotated function, an orthogonal matrix M [26] is considered and the rotated variable $y = M \times x$ is computed. Then, the vector y is used to evaluate the objective function value.

4.2. Comparison of HPSO with SPSO. The performance on the convergence accuracy of HPSO is compared with that of SPSO. The test functions listed in Table 1 are evaluated. For a fair comparison, we set the same parameters value. Population size is set to 30 ($N = 30$), upper bounds of velocity $V_{\max} = 0.5 \times (X_{\max} - X_{\min})$, and the corresponding lower bounds $V_{\min} = -V_{\max}$, where $X_{\min}$ and $X_{\max}$ are the lower and upper bounds of variables, respectively. Inertia weight w is linearly decreased from 0.9 to 0.4 in SPSO and HPSO. Acceleration coefficients c_1 and c_2 in SPSO are set to 2. The two algorithms are independently run 30 times on the benchmark functions. The results in terms of the best, worst, median, mean, and standard deviation (SD) of the solutions obtained in the 30 independent runs by each algorithm in different search spaces are as shown in Tables 2, 3, and 4. At the same time, the maximum iteration T is 1000 for $D = 30$, 2000 for $D = 50$, and 3000 for $D = 100$, respectively.

From Tables 2–4, we can clearly observe that the convergence accuracy of HPSO is better than SPSO on the most benchmark functions. An interesting result is that HPSO can find the global optimal solutions on functions F_2, F_4, F_5, F_9, F_{12}, F_{13}, F_{15}, F_{19}, and f_{22} in all search spaces; that is to say, HPSO can obtain the 100% success rate on the listed functions. Considering F_1 and F_{10}, though HPSO can find the global optimal solutions in all different search ranges, it only obtained the mean values 333.3333 and 0.8333, respectively, in 100-dimensional space. At the same time, HPSO offers the higher convergence accuracy on functions F_3, F_6, F_7, F_{11}, F_{14}, F_{16}, F_{20}, F_{21}, F_{23}, and F_{26}. However, we must observe that SPSO has higher performance on function F_8. As for F_{25}, SPSO has better performance in 30-dimensional search space, but HPSO has better performance in 50- and 100-dimensional search spaces. As for shifted rotated functions F_{27} and F_{28}, both SPSO and HPSO have worst convergence accuracy. As seen, the dimension of the selected functions has great effect on SPSO. For example, considering function F_1, SPSO has mean value 666.6686, $3.6667e + 03$, and $4.0698e + 04$ in 30-dimensional, 50-dimensional, and 100-dimensional search spaces, respectively, while HPSO has mean values 0, 0, and 333.333 in the corresponding search space. Therefore, we

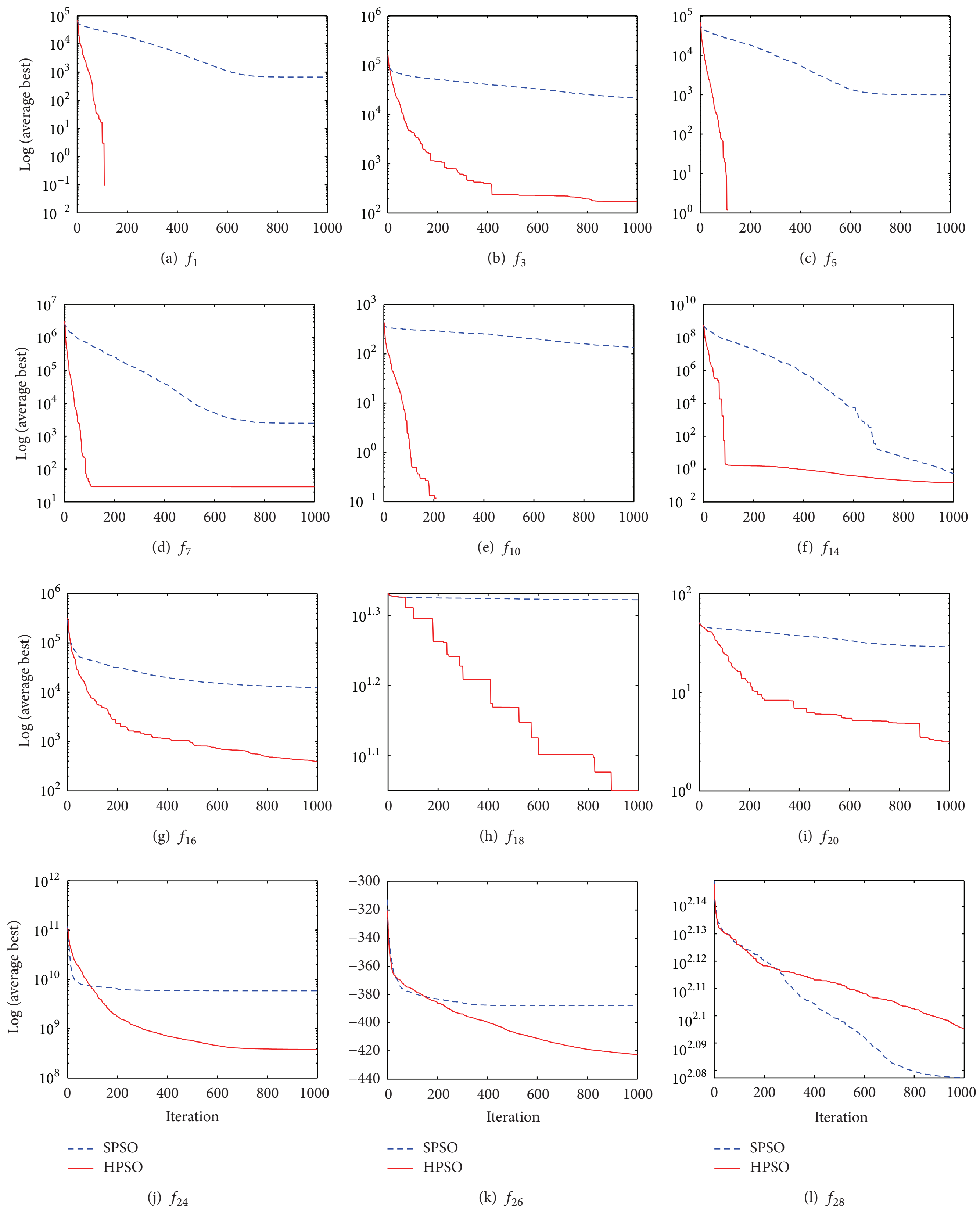


FIGURE 4: Convergence comparison of HPSO and SPSO on the selected test functions with $D = 30$, $N = 30$, and $T = 1000$.

Table 6: Comparison results of eight PSO algorithms [22] with HPSO on 10 functions ($N = 20$, $D = 30$, and FEs $= 2 \times 10^5$).

Function	GPSO	LPSO	VPSO	FIPS	HPSO-TVAC	DMS-PSO	CLPSO	APSO	HPSO
F_1									
Mean	$1.98e-53$	$4.77e-29$	$5.11e-38$	$3.21e-30$	$3.38e-41$	$3.85e-54$	$1.89e-19$	$1.45e-150$	**0**
SD	$7.08e-53$	$1.13e-28$	$1.91e-37$	$3.60e-30$	$8.50e-41$	$1.75e-53$	$1.49e-19$	$5.73e-150$	**0**
Rank	4	8	6	7	5	3	9	2	**1**
F_2									
Mean	$2.51e-34$	$2.03e-20$	$6.29e-27$	$1.32e-17$	$6.9e-23$	$2.61e-29$	$1.01e-13$	$5.15e-84$	**0**
SD	$5.84e-34$	$2.89e-20$	$8.68e-27$	$7.86e-18$	$6.89e-23$	$6.6e-29$	$6.51e-14$	$1.44e-83$	**0**
Rank	3	7	5	8	6	4	9	2	**1**
F_3									
Mean	$6.45e-2$	18.60	1.44	0.77	$2.89e-7$	47.5	395	**1.0e − 10**	167
SD	$9.45e-2$	30.71	1.55	0.86	$2.97e-7$	56.4	142	**2.13e − 10**	913
Rank	3	6	5	4	2	7	9	**1**	8
F_5									
Mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
F_6									
Mean	$7.77e-3$	$1.49e-2$	$1.08e-2$	$2.55e-3$	$5.54e-2$	$1.1e-2$	$3.92e-3$	$4.66e-3$	**1.03e − 04**
SD	$2.42e-3$	$5.66e-3$	$3.24e-3$	$6.25e-4$	$2.08e-2$	$3.94e-3$	$1.14e-3$	$1.7e-3$	**8.99e − 05**
Rank	5	8	6	2	9	7	3	4	**1**
F_9									
Mean	30.7	34.90	34.09	29.98	2.39	28.1	$2.57e-11$	$5.8e-15$	**0**
SD	8.68	7.25	8.07	10.92	3.71	6.42	$6.64e-11$	$1.01e-14$	**0**
Rank	7	9	8	6	4	5	3	2	**1**
F_{10}									
Mean	15.5	30.40	21.33	35.91	1.83	32.8	0.167	$4.14e-16$	**0**
SD	7.4	9.23	9.46	9.49	2.65	6.49	0.379	$1.45e-15$	**0**
Rank	5	7	6	9	4	8	3	2	**1**
F_{11}									
Mean	$1.15e-14$	$1.85e-14$	$1.4e-14$	$7.69e-15$	$2.06e-10$	$8.52e-15$	$2.01e-12$	$1.11e-14$	**8.88e − 16**
SD	$2.27e-15$	$4.80e-15$	$3.48e-15$	$9.33e-16$	$9.45e-10$	$1.79e-15$	$9.22e-13$	$3.55e-15$	**0**
Rank	5	7	6	2	9	3	8	4	**1**
F_{12}									
Mean	$2.37e-2$	$1.10e-2$	$1.31e-2$	$9.04e-4$	$1.07e-2$	$1.31e-2$	$6.45e-13$	$1.67e-2$	**0**
SD	$2.57e-2$	$1.60e-2$	$1.35e-2$	$2.78e-3$	$1.14e-2$	$1.73e-2$	$2.07e-12$	$2.41e-2$	**0**
Rank	9	5	6	3	4	7	2	8	**1**
F_{14}									
Mean	$1.04e-2$	$2.18e-30$	$3.46e-3$	$1.22e-31$	$7.07e-30$	**2.05e − 32**	$1.59e-21$	$3.76e-31$	$1.70e-2$
SD	$3.16e-2$	$5.14e-30$	$1.89e-2$	$4.85e-32$	$4.05e-30$	**8.12e − 33**	$1.93e-21$	$1.2e-30$	$1.42e-2$
Rank	8	4	7	2	5	**1**	6	3	9
Average rank	5	6.2	5.6	4.4	4.9	4.6	5.3	2.9	**2.5**
Final rank	6	9	8	3	5	4	7	2	**1**

also conclude that HPSO has better stability than SPSO from the data in different search spaces.

In the 9th columns of Tables 2–4, we report the statistical significance level of the difference of the means of the two algorithms. Note that here "+" indicates that the t value is significant at a 0.05 level of significance by two-tailed test, and "−" stands for the difference of means that is not statistically significant.

Figure 4 graphically presents the comparison in terms of convergence characteristics of the evolutionary processes in solving the selected benchmark functions in 30-dimensional search space with $N = 30$ and $T = 1000$.

4.3. Comparison of HPSO with Other PSO Algorithms. In this section, a comparison of HPSO with some well-known PSO algorithms which are listed in Table 5 is performed to evaluate the efficiency of the proposed algorithm.

At first, we choose 10 unimodal and multimodal test functions for this evaluation. According to [22], the algorithms

Table 7: Comparison results of seven PSO algorithms [25] with HPSO on six functions ($N = 30$, $D = 30$, and $T = 10{,}000$).

Function	SPSO	QIPSO	UPSO	FIPS	CLPSO	AFSO	AFSO-Q1	HPSO
F_9								
Mean	52.30	25.61	59.40	106.1	74.39	17.93	15.69	**0**
SD	27.35	15.98	58.05	30.54	9.77	5.63	4.47	**0**
Rank	5	4	6	8	7	3	2	**1**
F_{13}								
Mean	0.534	36.38	8.70	6.40	$1.39e-03$	$4.52e-03$	$1.50e-03$	**0**
SD	1.74	4.66	3.08	3.04	$3.28e-04$	$9.20e-03$	$3.48e-03$	**0**
Rank	5	8	7	6	2	4	3	**1**
F_{21}								
Mean	320.2	317.5	309.5	434.1	263.3	266.3	253.3	**0**
SD	14.70	23.24	25.88	34.99	11.96	12.00	12.63	**0**
Rank	7	6	5	8	3	4	2	**1**
F_{22}								
Mean	17.03	15.20	14.29	26.60	11.94	10.38	8.46	**0**
SD	2.55	1.32	2.15	1.42	1.37	1.38	0.948	**0**
Rank	7	6	5	8	4	3	2	**1**
F_{27}								
Mean	−119.10	−119.10	−119.10	−119.90	−119.00	−119.70	−119.80	−119.05
SD	$7.09e-02$	$5.68e-02$	$3.24e-02$	$3.78e-02$	$4.28e-02$	$3.85e-02$	$5.45e-02$	$5.50e-02$
Rank	4	4	4	1	6	3	2	5
F_{28}								
Mean	115.90	121.90	**113.20**	113.60	118.30	123.20	123.10	117.32
SD	2.90	4.90	6.14	3.63	2.40	**2.25**	3.01	3.65
Rank	3	6	**1**	**2**	**5**	8	7	4
Average rank	5.17	5.67	4.67	5.50	4.50	4.17	3.00	**2.17**
Final rank	6	8	5	7	4	3	2	**1**

GPSO [2], LPSO [16], VPSO [27], FIPS [28], HPSO-TVAC [6], DMS-PSO [29], CLPSO [24], and APSO [22] are considered as detailed in Table 5. The experimental results of the algorithms are directly from [22] as shown in Table 6. In this trial, the population size $N = 20$, the dimension $D = 30$, and the maximum fitness evaluations (FEs) were set to 2×10^5 also. The parameter configurations of the selected algorithms have been set according to their corresponding references. The inertia weight w is linearly decreased from 0.9 to 0.4 in HPSO. HPSO is independently run 30 times and the mean and SD are shown in Table 6. As seen, HPSO has the first rank among the algorithms and obtains the global minimum on functions F_1, F_2, F_5, F_9, F_{10}, and F_{12} and gives the good near-global optima on functions F_6 and F_{11}. Meanwhile, HPSO has the worst performance on functions F_3 and F_{14}. As for F_3, APSO has the best convergence accuracy, and HPSO only wins CLPSO. Considering F_{14}, DMS-PSO has the best performance.

Then, in the next step, we choose six functions from [25] and seven algorithms of GPSO, QIPSO [30], UPSO [31], FIPS, AFSO [25], and AFSO-Q1 [25] as detailed in Table 5. For a fair comparison, the population size $N = 30$, the dimension $D = 30$, and the maximum iteration $T = 10,000$ also in HPSO, and the inertia weight w is linearly decreased from 0.9 to 0.4. HPSO is independently run 30 times and the mean and SD are shown in Table 7. As seen, HPSO shows better performance and has the first rank. HPSO finds the global optimal solution on functions F_9, F_{13}, F_{21}, and F_{22}. FIPS and UPSO have better convergence accuracy on functions F_{27} and F_{28}, respectively.

Therefore, it is worth saying that the proposed algorithm has considerably better performance than the other well-known PSO algorithms in unimodal and multimodal high-dimensional functions.

5. Conclusion

In this paper, a modified version of PSO called HPSO has been introduced to enhance the performance of SPSO. To simulate the human behavior, the global worst particle was introduced into the velocity equation of SPSO, and the learning coefficient which obeys the standard normal distribution can balance the exploration and exploitation abilities by changing the flying direction of particles. When the coefficient is positive, it is called impelled leaning coefficient, which is helpful to enhance the exploration ability. When the coefficient is negative, it is called penalized learning coefficient, which is beneficial for improving the exploitation ability. At the same time, the acceleration coefficients c_1 and c_2 have been replaced with two random numbers, whose sum is equal to 1 in [0, 1]; this strategy decreases the dependence on parameters of the solved problems. The proposed algorithm has been evaluated on 28 benchmark functions including unimodal, unrotated multimodal, rotated, shifted, and shifted rotated functions, and the experimental results confirm the high performance of HPSO on the main functions. However, as seen, HPSO has the worst performance on shifted rotated functions, so it is worth researching how to enhance the performance of HPSO on shifted rotated functions in the future. Meanwhile, applying HPSO to solve real-world problems is also a research field.

Conflict of Interests

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

Acknowledgments

The project is supported by the National Natural Science Foundation of China (Grant no. 61175127) and the Science and Technology Project of Department of Education of Jiangxi Province China (Grant no. GJJ12093).

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[2] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, May 1998.

[3] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 101–106, May 2001.

[4] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the Congress on Evolutionary Computation*, pp. 94–100, May 2001.

[5] G. Xu, "An adaptive parameter tuning of particle swarm optimization algorithm," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4560–4569, 2013.

[6] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[7] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[8] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '00)*, pp. 84–88, July 2000.

[9] P. J. Angeline, "Using selection to improve particle swarm optimization," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 84–89, May 1998.

[10] Y.-P. Chen, W.-C. Peng, and M.-C. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 6, pp. 1460–1470, 2007.

[11] P. S. Andrews, "An investigation into mutation operators for particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 1044–1051, July 2006.

[12] W.-J. Zhang and X.-F. Xie, "DEPSO: Hybrid particle swarm with differential evolution operator," in *Proceedings of the IEEE International Conference on System Security and Assurance*, pp. 3816–3821, October 2003.

[13] M. S. Kıran, M. Gündüz, and K. Baykan, "A novel hybrid algorithm based on particle swarm and ant colony optimization for finding the global minimum," *Applied Mathematics and Computation*, vol. 219, no. 4, pp. 1515–1521, 2012.

[14] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 522–528, September 2005.

[15] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1931–1938, 1999.

[16] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1671–1676, 2002.

[17] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 80–87, 2003.

[18] R. A. Krohling and E. Mendel, "Bare bones particle swarm optimization with Gaussian or cauchy jumps," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 3285–3291, May 2009.

[19] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *European Journal of Operational Research*, vol. 196, no. 1, pp. 128–139, 2009.

[20] J. Sun, W. Fang, V. Palade, X. Wu, and W. Xu, "Quantum-behaved particle swarm optimization with Gaussian distributed local attractor point," *Applied Mathematics and Computation*, vol. 218, no. 7, pp. 3763–3775, 2011.

[21] J. Sun, X. Wu, V. Palade, W. Fang, C.-H. Lai, and W. Xu, "Convergence analysis and improvements of quantum-behaved particle swarm optimization," *Information Sciences*, vol. 193, pp. 81–103, 2012.

[22] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[23] H. Wang, H. Sun, C. Li, S. Rahnamayan, and J.-S. Pan, "Diversity enhanced particle swarm optimization with neighborhood search," *Information Sciences*, vol. 223, pp. 119–135, 2013.

[24] 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.

[25] Y.-T. Juang, S.-L. Tung, and H.-C. Chiu, "Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions," *Information Sciences*, vol. 181, no. 20, pp. 4539–4549, 2011.

[26] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms," *BioSystems*, vol. 39, no. 3, pp. 263–278, 1996.

[27] J. Kennedy and R. Mendes, "Neighborhood topologies in fully informed and bestof-neighborhood particle swarms," *IEEE*

Transactions on Systems, Man, and Cybernetics, Part C, vol. 36, no. 4, pp. 515–519, 2006.

[28] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[29] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)*, pp. 127–132, June 2005.

[30] M. Pant, T. Radha, and V. P. Singh, "A new particle swarm optimization with quadratic interpolation," in *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '07)*, pp. 55–60, December 2007.

[31] K. E. Parsopoulos and M. N. Vrahatis, "UPSO: a unified particle swarm scheme," in *Proceedings of the International Conference of Computational Methods in Sciences and Engineering*, vol. 1 of *Lecture Series on Computer and Computational Sciences*, pp. 868–873, 2004.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 189164, 8 pages
http://dx.doi.org/10.1155/2014/189164

Research Article

Application of the Artificial Bee Colony Algorithm for Solving the Set Covering Problem

Broderick Crawford,[1,2] Ricardo Soto,[1,3] Rodrigo Cuesta,[1] and Fernando Paredes[4]

[1] *Pontificia Universidad Católica de Valparaíso, 2362807 Valparaíso, Chile*
[2] *Universidad Finis Terrae, 7500000 Santiago, Chile*
[3] *Universidad Autónoma de Chile, 7500000 Santiago, Chile*
[4] *Escuela de Ingeniería Industrial, Universidad Diego Portales, 8370179 Santiago, Chile*

Correspondence should be addressed to Broderick Crawford; broderick.crawford@ucv.cl

Received 20 February 2014; Accepted 30 March 2014; Published 16 April 2014

Academic Editors: S. Balochian and Y. Zhang

Copyright © 2014 Broderick Crawford 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 set covering problem is a formal model for many practical optimization problems. In the set covering problem the goal is to choose a subset of the columns of minimal cost that covers every row. Here, we present a novel application of the artificial bee colony algorithm to solve the non-unicost set covering problem. The artificial bee colony algorithm is a recent swarm metaheuristic technique based on the intelligent foraging behavior of honey bees. Experimental results show that our artificial bee colony algorithm is competitive in terms of solution quality with other recent metaheuristic approaches for the set covering problem.

1. Introduction

The set covering problem (SCP) is a classic problem in combinatorial analysis, computer science, and theory of computational complexity. It is a problem that has led to the development of fundamental technologies for the field of the approximation algorithms. Also it is one of the problems from the list of 21 Karp's NP-complete problems; its NP-completeness was demonstrated in 1972 [1].

SCP has many applications, including those involving routing, scheduling, stock cutting, electoral redistricting, and other important real life situations [2]. Although the best known application of the SCP is airline crew scheduling [3], where a given set of trips has to be covered by a minimum-cost set of pairings, a pairing being a sequence of trips that can be performed by a single crew.

Different solving methods have been proposed in the literature for the set covering problem. Exact algorithms are mostly based on branch-and-bound and branch-and-cut techniques [4–6], linear programing, and heuristic methods [7]. However, these algorithms are rather time consuming and can only solve instances of very limited size. For this reason, many research efforts have been focused on the development of heuristics to find a good result or near-optimal solutions within a reasonable period of time.

Classical greedy algorithms are very simple, fast, and easy to code in practice, but they rarely produce high quality solutions for their myopic and deterministic nature [8]. In [9] a greedy algorithm was improved incorporating randomness and memory into it and obtained promising results. Compared with classical greedy algorithms, heuristics based on Lagrangian relaxation with subgradient optimization are much more effective. The most efficient ones are those proposed by [10, 11].

Metaheuristics were also applied to the SCP as top-level general search strategies. An incomplete list of this kind of metaheuristics for the SCP includes genetic algorithms [12, 13], simulated annealing [14], tabu search [15], cultural algorithms [16, 17], and ant colony optimization [18]. For a deeper comprehension of effective algorithms for the SCP in the literature, we refer the interested reader to the survey by [7].

In this paper we propose a novel application of artificial bee colony (ABC) to solve SCP. This paper is organized as

Table 1: Incidence matrix example.

$$A = \begin{array}{c|ccccc} & S_1 & S_2 & S_3 & S_4 & S_5 \\ \hline x_1 & 0 & 0 & 1 & 0 & 1 \\ x_2 & 0 & 1 & 0 & 0 & 0 \\ x_3 & 1 & 0 & 1 & 0 & 0 \\ x_4 & 0 & 1 & 0 & 0 & 1 \\ x_5 & 0 & 0 & 0 & 0 & 1 \\ x_6 & 0 & 0 & 0 & 1 & 0 \\ x_7 & 1 & 0 & 1 & 1 & 0 \\ x_8 & 0 & 1 & 1 & 0 & 0 \\ x_9 & 1 & 0 & 0 & 0 & 1 \\ x_{10} & 0 & 1 & 0 & 0 & 0 \end{array}$$

follows. In Section 2, we explain the problem. In Section 3, we describe the ABC framework. Our ABC proposal is described in Section 4. In Section 5, we present the experimental results obtained. Finally, in Section 6, we conclude the paper and give some perspectives for further research.

2. Problem Description

The set covering problem is a fundamental problem in the class of covering problems. Given a finite set X and a family $F = S_1, S_2, \ldots, S_n$ of subsets of X (i.e., $S_j \subseteq X$, $j = 1, \ldots, n$), the SCP aims to find a minimum cardinality $J \subseteq \{1, \ldots, n\}$ such that $\bigcup_{j \in J} S_j = X$. The elements of X are called *points*. Given a $J \subseteq \{1, \ldots, n\}$, a point is set to be covered if belongs to $\bigcup_{j \in J} S_j$. In the minimum-cost set covering problem each set S_j, $1 \le j \le n$, has a cost c_j and the problem is to find a $J \subseteq \{1, \ldots, n\}$, where each point is covered and $\sum_{j \in J} c_j$ is minimum. This minimum-cost optimization version of SCP is NP-hard.

Let us define the incidence matrix A of a set covering problem as follows. There are $|X|$ rows in A, one for each point of $x_i \in X$, and n columns in A, one for each set S_j. The entry a_{ij} at A (the entry at the intersection of the ith row and the jth column) is one if point x_i is in set S_j; otherwise a_{ij} is zero. Table 1 shows an example of an incidence matrix.

For the upcoming reference cases, a general mathematical model of the SCP can be formulated as follows:

$$\text{Minimize} \quad Z = \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \ge 1 \quad \forall i = \{1, 2, 3, \ldots, m\} \tag{2}$$

$$x_j \in \{0, 1\} \quad \forall j = \{1, 2, 3, \ldots, n\}. \tag{3}$$

Equation (1) is the objective function of the SCP, where c_j refers to the weight or cost of j-column and x_j is the decision variable. Equation (2) is a constraint to ensure that each row is covered by at least one column; $m \times n$ matrix $A = (a_{ij})$ is a constraint coefficient matrix whose elements can be "1" or "0" to indicate the covering possibilities. Finally, (3) is the integrality constraint where the value x_j can be "1" if column j is activated (selected) or "0" otherwise.

3. Artificial Bee Colony Algorithm

ABC is one of the most recent algorithms in the domain of the collective intelligence. It was created by Dervis Karaboga in 2005, who was motivated by the intelligent behavior observed in the domestic bees to take the process of foraging [19].

ABC is an algorithm of combinatorial optimization based on populations, in which the solutions of the problem of optimization, the sources of food, are modified by the artificial bees that function as operators of variation. The aim of these bees is to discover the food sources with major nectar.

In the ABC algorithm, an artificial bee moves in a multidimensional search space choosing sources of nectar depending on its past experience and its companions of beehive or fitting its position. Some bees (exploratory) fly and choose food sources randomly without using experience. When they find a source of major nectar, they memorize their positions and forget the previous ones. Thus, ABC combines methods of local search and global search, trying to balance the process of the exploration and exploitation of the search space.

Although, the performance of different optimization algorithm is dependent on applications, some recent works demonstrate that the artificial bee colony is more rapid than either genetic algorithm or particle swarm optimization solving certain problems [20–24]. Additionally, ABC has demonstrated an ability to attack problems with a lot of variables (high-dimensional problems) [25].

3.1. Elements and Behavior. The model defines three principal components which are enunciated as follows.

Food Source. The value of a food source depends on many different factors, as its proximity to the beehive, wealth or the concentration of the energy, and the facility of extraction of this energy.

Employed Bees or Workers. They are associated with a current food source, or in exploitation, they take with them information about this source, especially the distance, location, and profitability, to share this with a certain probability with other companions.

Unemployed or Exploratory Bees. They are in constant search of a food source. There are two types:

(i) scouts: they are the ones in charge of searching in the environment that surrounds the beehive for new sources of food.

(ii) onlookers (curious or in wait): they look for a food source across the information shared by the employees or by other explorers in the nest.

3.2. Biological Behavior. The exchange of information between the bees is the most important incident in the formation of a collective knowledge, since the meaning of this

Table 2: Summary of ABC main elements.

Generation of food sources	A solution to the optimization problem is a food source. It moves in a random way and with base in the low and superior limits of every variable of the problem
Working bees	Their number is proportional to the number of food sources, where for every source there is a working bee, and its function is to evaluate and to modify the current solutions to improve them (it looks for new sources near the current one). If the new position is not better than the current, it will keep the original position
Bees in wait	The number of bees in wait must be proportional to the number of sources. These bees will choose a food source, based on the information of the working bees by means of the waggle dance, where the food source with better value of objective function is selected
Scout bees	These bees generate new sources of food in a random way to replace existing sources that have not been improved
Limit	It defines the maximum number of cycles that a food source can keep without improving before being replaced. The limit increases from the source that is not modified by the bees; already they are used or in wait, up to obtaining its maximum allowed value. After this, the scout bees take charge of initializing the limit to 0 for every new generated position. The limit is initialized to 0 whenever a source is modified (improved) by a used bee or in wait
Column adding	When solving SCP, it defines the number of columns to be added to the current food source
Column elimination	When solving SCP, it defines the number of columns to be eliminated from the current food source

```
(1)  Begin
(2)    InitPopulation()
(3)    While remain iterations do
(4)      Select sites for the local search
(5)      Recruit bees for the selected sites and to evaluate fitness
(6)      Select the bee with the best fitness
(7)      Assign the remaining bees to looking for randomly
(8)      Evaluate the fitness of remaining bees
(9)      UpdateOptimum()
(10)   End While
(11)   Return BestSolution
(12) End
```

Algorithm 1: ABC pseudocode.

interaction the bees will decide the behavior that must take the beehive. The principal ways of bee behaviour are

(i) the incorporation to a source of nectar: the communication between the bees related to the quality of food sources is realized in the zone of dance (dance of the bees), where with the information obtained about all the sources of food that are available, they decide which of all the sources is the most profitable to join.

(ii) the abandon of a source: by means of the dance it is determined if a source is no longer profitable and consequently it must be abandoned.

3.3. Artificial Behavior. In Table 2 the elements of the ABC are described in a general way.

The pseudocode of artificial bee colony is as in Algorithm 1.

The procedure for determining a food source in the neighborhood of a particular food source which depends on the nature of the problem. Karaboga [26] developed the first ABC algorithm for continuous optimization. The method for determining a food source in the neighborhood of a particular food source is based on changing the value of one randomly chosen solution variable while keeping other variables unchanged. This is done by adding to the current value of the chosen variable the product of a uniform variable in $[-1, 1]$ and the difference in values of this variable—current food source—and some other randomly chosen food source. This approach cannot be used for discrete optimization problems for which it generates at best a random effect.

Singh [27] subsequently proposed a method, which is appropriate for subset selection problems. In his model, to generate a neighboring solution, an object is randomly dropped from the solution and in its place another object, which is not already present in the solution, is added. The object to be added is selected from another randomly chosen solution. If there are more than one candidate object for addition, then ties are broken arbitrarily.

This approach is based on the idea that if an object is present in one good solution, then it is highly likely that this object is present in many good solutions. This method

333	10	5	300	88	⋯	657	99

Figure 1: Representation of a solution.

provides another advantage, consisting in that if the method fails to find an object different from the others objects in the original solution, this means that the two solutions are equal, such situation is called "collision" and it is resolved by making the employed bee associated with the original solution, a scout bee eliminating duplication.

4. Description of the Proposed Approach to Solve SCP

Step 1 (initialization). This step includes initializing the parameters of ABC as size of the colony, number of workers and curious (onlookers or "in wait") bees, limit of attempts, and maximum number of cycles.

Step 2 (generation of initial population). To generate the initial population by every row—SCP constraint—a column—SCP variable—is selected at random from the set of columns with covering possibilities. A solution is represented by means of an entire vector as shown in Figure 1 keeping the columns considered in the solution. Then, we use an integer encoding as the encoding rule.

Step 3 (evaluation of the fitness of the population). The fitness function is equal to the objective function of the SCP (1).

Step 4 (modification of position and selection of sites for worker bees). A hard-working bee modifies its position by means of the creation of a new solution based on a different food source selected randomly. It sees if at least it has a different column, in case of having not even a different column, the hard-working bee is transformed to an explorer in order to eliminate duplicated solutions. In opposite case, it proceeds to add a certain random number of columns between 0 and the maximum number of columns to be added.

After this, it proceeds to eliminate a certain random number of columns between 0 and the maximum number of columns to be eliminated. In case that new solution does not meet constraints, it is repaired. The fitness of the solution is evaluated; if the fitness (cost) is minor than the solution obtained in the beginning, the solution is replaced. In opposite case, it increases the number of attempts for improving this solution (limit).

Step 5 (recruiting curious bees for the selected sites). A curious bee evaluates the information of the nectar through the workers and it chooses a source of food with the fitness proportionate selection method or roulette-wheel selection.

Step 6 (modification of position for the curious bees). They work alike to hard-working bees in Step 4.

Step 7 (leaving a source exploited by the bees). If the solution representing a source of food does not improve for a predetermined number of attempts (limit), then the source of food is left and is replaced by a new source of food generated as in Step 1.

Step 8. This step involves memorizing the best solution and increasing the counter of the cycle.

Step 9. The process stops if the criteria of satisfaction expire; in opposite case return to Step 3.

5. Experimental Results

The ABC algorithm has been implemented in C in a 2.5 GHz Dual Core with 4 GB RAM computer, running windows 7.

Parameter values have a profound influence on the performance of ABC. The parameters were empirically adjusted, we determined their values in an experimental way, and for each parameter, a set of candidate values were considered. We modified the value of one parameter while keeping the others fixed. According to the best results, as parameter values in our experiments, we use ABC runs 1000 iterations with a population of 200 bees, where 100 corresponds to hard-working and 100 to curious. Limit = 50, maximum number of columns to add = 0.5% of columns in the SCP instance, and maximum number of columns to eliminate = 1.2% of the SCP instance.

These parameters showed good results, but they cannot be the ideal ones for all the instances. ABC has been tested on 65 standard non-unicost SCP instances available from OR Library at http://people.brunel.ac.uk/~mastjjb/jeb/info.html. Table 3 summarizes the characteristics of each of these sets of instances, each set contains 5 or 10 problems and the column labeled Density shows the percentage of nonzero entries in the matrix of each instance. ABC was executed 30 times on each instance, each trial with a different random seed.

5.1. Comparison with Other Works. In comparison with very recent works solving SCP—with cultural algorithms [16] and ant colony + constraint programming techniques [28]—our proposal performs better than the SCP instances reported in those works. In order to bring out the efficiency of our proposal, the solutions of the complete set of instances have been compared with other metaheuristics. We compared our algorithm solving the complete set of 65 standard non-unicost SCP instances from OR Library with the newest ACO-based algorithm for SCP in the literature: ant-cover + local search (ANT + LS) [29], genetic algorithm (GA) proposed by Beasley and Chu (1996) [13], and simulated annealing (SA) proposed by Brusco et al. (1999) [14].

Tables 4 and 5 show the detailed results obtained by four algorithms. Column 2 reports the optimal or the best known solution value of each instance. The third and fourth columns show the best value and the average obtained by our ABC algorithm in the 30 runs (trials). The next columns show the average values obtained by GA, SA, and ANT + LS, respectively. The last column shows the relative percentage deviation (RPD) value over the instances tested with ABC. The quality of solutions can be evaluated using the RPD; its value quantifies the deviation of the objective value Z from

TABLE 3: Details of the 65 test instances.

Instance set	Number of instances	m	n	Cost range	Density (%)	Optimal solution
4	10	200	1000	[1, 100]	2	Known
5	10	200	2000	[1, 100]	2	Known
6	5	200	1000	[1, 100]	5	Known
A	5	300	3000	[1, 100]	2	Known
B	5	300	3000	[1, 100]	5	Known
C	5	400	4000	[1, 100]	2	Known
D	5	400	4000	[1, 100]	5	Known
NRE	5	500	5000	[1, 100]	10	Unknown
NRF	5	500	5000	[1, 100]	20	Unknown
NRG	5	1000	10000	[1, 100]	2	Unknown
NRH	5	1000	10000	[1, 100]	5	Unknown

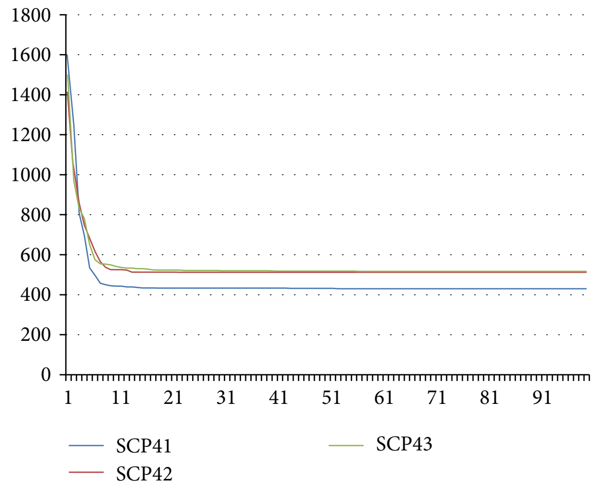


FIGURE 2: Convergence analysis to a better solution (benchmarks: SCP41, SCP42, and SCP43).

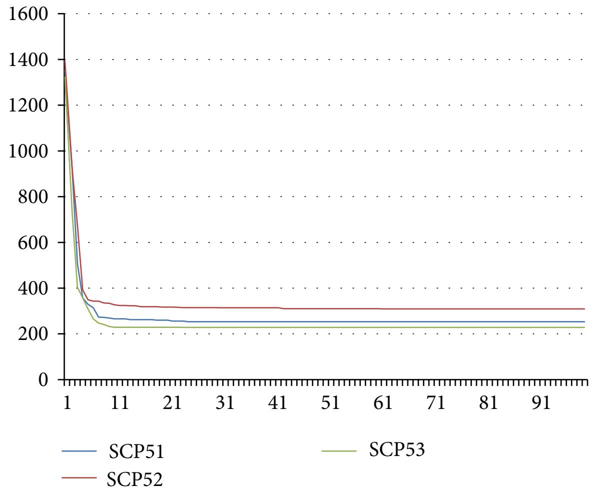


FIGURE 3: Convergence analysis to a better solution (benchmarks: SCP51, SCP52, and SCP53).

Z_{opt} which in our case is the best known cost value for each instance. This measure is computed as follows:

$$\text{RPD} = \frac{(Z - Z_{\text{opt}})}{Z_{\text{opt}}} \times 100. \quad (4)$$

Examining Tables 4 and 5, we observe the following.

(i) ABC is able to find the optimal solution consistently, that is, in every trial, for 43 of 65 problems.

(ii) ABC is able to find the best known value in all instances of Table 5.

(iii) ABC is able to find the best known value in all trials of Table 5.

(iv) ABC has higher success rate compared to genetic algorithm, simulated annealing, and ants in sets NRE, NRF, NRG, and NRH. The RPD of BEE is 0.00%, the RPD of GA is 1.04%, the RPD of SA is 0.72%, and the RPD of ANT + LS is 0.86%.

(v) ABC can obtain optimal solutions in some instances where the other metaheuristics failed.

5.2. Convergence to the Best Solution. Our approach shows an excellent tradeoff between the quality of the solutions obtained and the computational effort required. In all cases, ABC converged very quickly (mainly from the 10th iteration) and its computation time in the runs was less than 2 seconds (except for NRG and NRH instances where the computation time was less than 30 secs).

Figures 2 and 3 illustrate how ABC converges through the iterations to a better solution. We consider only 3 problems per chart in favor of clarity and readability: scp41, scp42, and scp43 for the first chart and scp51, scp52, and spc53 for the second one. x-axis represents the iteration number while y-axis represents the reached fitness value.

6. Conclusion

In this paper we have presented an ABC algorithm for the SCP. We have performed experiments throught several ORLIB instances; our approach has been shown to be very effective, providing an unattended solving method, for quickly producing solutions of a good quality. Experiments

Table 4: Experimental results—instances with optimal.

Instance	Optimum	Best value found	ABC Avg.	GA Avg.	SA Avg.	ANT-LS Avg.	RPD (%)
4.1	429	430	430.5	429.7	—	429	0.35
4.2	512	512	512	512	—	512	0
4.3	516	516	516	516	—	516	0
4.4	494	494	494	494.8	—	494	0
4.5	512	512	512	512	—	512	0
4.6	560	561	561.7	560	—	560	0.30
4.7	430	430	430	430.2	—	430	0
4.8	492	493	494	492.1	—	492	0.41
4.9	641	643	645.5	643.1	—	641	0.70
4.10	514	514	514	514	—	514	0
5.1	253	254	255	253	—	253	0.79
5.2	302	309	310.2	303.5	—	302	2.72
5.3	226	228	228.5	228	—	226	1.11
5.4	242	242	242	242.1	—	242	0
5.5	211	211	211	211	—	211	0
5.6	213	213	213	213	—	213	0
5.7	293	296	296	293	—	293	1.02
5.8	288	288	288	288.8	—	288	0
5.9	279	280	280	279	—	279	0.36
5.10	265	266	267	265	—	265	0.75
6.1	138	140	140.5	138	—	138	1.81
6.2	146	146	146	146.2	—	146	0
6.3	145	145	145	145	—	145	0
6.4	131	131	131	131	—	131	0
6.5	161	161	161	161.3	—	161	0
A.1	253	254	254	253.2	—	253	0.40
A.2	252	254	254	253	—	252	0.79
A.3	232	234	234	232.5	—	232.8	0.86
A.4	234	234	234	234	—	234	1.10
A.5	236	237	238.6	236	—	236	0
B.1	69	69	69	69	—	69	0
B.2	76	76	76	76	—	76	0
B.3	80	80	80	80	—	80	0
B.4	79	79	79	79	—	79	0
B.5	72	72	72	72	—	72	0
C.1	227	230	231	227.2	—	227	1.76
C.2	219	219	219	220	—	219	0
C.3	243	244	244.5	246.4	—	243	0.62
C.4	219	220	224	219.1	—	219	2.28
C.5	215	215	215	215.1	—	215	0
D.1	60	60	60	60	—	60	0
D.2	66	67	67	66	—	66	1.52
D.3	72	73	73	72.2	—	72	1.39
D.4	62	63	63	62	—	62	1.61
D.5	61	62	62	61	—	61	1.64

Table 5: Experimental results—instances with best known solution.

Instance	Optimum	Best value found	ABC Avg.	GA Avg.	SA Avg.	ANT-LS Avg.	RPD (%)
NRE.1	29	29	29	29	29	29	0
NRE.2	30	30	30	30.6	30	30	0
NRE.3	27	27	27	27.7	27	27	0
NRE.4	28	28	28	28	28	28	0
NRE.5	28	28	28	28	28	28	0
NRF.1	14	14	14	14	14	14	0
NRF.2	15	15	15	15	15	15	0
NRF.3	14	14	14	14	14	14	0
NRF.4	14	14	14	14	14	14	0
NRF.5	13	13	13	13.7	13.7	13.5	0
NRG.1	176	176	176	177.7	176.6	176	0
NRG.2	154	154	154	156.3	155.3	155.1	0
NRG.3	166	166	166	167.9	167.6	167.3	0
NRG.4	168	168	168	170.3	170.7	168.9	0
NRG.5	168	168	168	169.4	168.4	168.1	0
NRH.1	63	63	63	64	64	64	0
NRH.2	63	63	63	64	63.7	67.9	0
NRH.3	59	59	59	59.1	59.4	59.4	0
NRH.4	58	58	58	58.9	58.9	58.7	0
NRH.5	55	55	55	55.1	55	55	0

showed interesting results in terms of robustness, where using the same parameters for different instances gave good results.

The promising results of the experiments open up opportunities for further research. We visualize different directions for future work as follows.

(i) The fact that the presented algorithm is easy to implement clearly implies that ABC could also be effectively applied to other combinatorial optimization problems.

(ii) An interesting proposal by Teodor Crainic et al. at [30] involves parallelizing strategies for metaheuristics. The author sets a basis on the idea that the central goal of parallel computing is to speed up computation by dividing the work load among several threads of simultaneous execution; then a type of metaheuristic parallelism could come from the decomposition of the decision variables into disjoint subsets. The particular heuristic is applied to each subset and the variables outside the subset are considered fixed.

(iii) An interesting extension of this work would be related to hybridization with other metaheuristics or applying a hyperheuristic approach [31].

(iv) The use of autonomous search (AS) represents a new research field, and it provides practitioners with systems that are able to autonomously self-tune their performance while effectively solving problems. Its major strength and originality consist in the fact that problem solvers can now perform self-improvement operations based on analysis of the performances of the solving process [32–34].

(v) Furthermore, we are considering to use different preprocessing steps from the OR literature, which allow to reduce the problem size [35].

Conflict of Interests

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

Acknowledgments

Broderick Crawford is supported by Grant CONICYT/FONDECYT/1140897. Ricardo Soto is supported by Grant CONICYT/FONDECYT/INICIACION/11130459. Fernando Paredes is supported by Grant CONICYT/FONDECYT/1130455.

References

[1] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds., The IBM Research Symposia Series, pp. 85–103, Plenum Press, New York, NY, USA, 1972.

[2] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67–71, 1989.

[3] M. Mesquita and A. Paias, "Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem," *Computers and Operations Research*, vol. 35, no. 5, pp. 1562–1575, 2008.

[4] E. Balas and M. C. Carrera, "A dynamic subgradient-based branch-and-bound procedure for set covering," *Operations Research*, vol. 44, no. 6, pp. 875–890, 1996.

[5] M. L. Fisher and P. Kedia, "Optimal solution of set covering/partitioning problems using dual heuristics," *Management Science*, vol. 36, no. 6, pp. 674–688, 1990.

[6] J. E. Beasley and K. Jørnsten, "Enhancing an algorithm for set covering problems," *European Journal of Operational Research*, vol. 58, no. 2, pp. 293–300, 1992.

[7] A. Caprara, P. Toth, and M. Fischetti, "Algorithms for the set covering problem," *Annals of Operations Research*, vol. 98, no. 1-4, pp. 353–371, 2000.

[8] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.

[9] G. Lan and G. W. DePuy, "On the effectiveness of incorporating randomness and memory into a multi-start metaheuristic with application to the Set Covering Problem," *Computers & Industrial Engineering*, vol. 51, no. 3, pp. 362–374, 2006.

[10] S. Ceria, P. Nobili, and A. Sassano, "A Lagrangian-based heuristic for large-scale set covering problems," *Mathematical Programming, Series B*, vol. 81, no. 2, pp. 215–228, 1998.

[11] A. Caprara, M. Fischetti, and P. Toth, "Heuristic method for the set covering problem," *Operations Research*, vol. 47, no. 5, pp. 730–743, 1999.

[12] U. Aickelin, "An indirect genetic algorithm for set covering problems," *Journal of the Operational Research Society*, vol. 53, no. 10, pp. 1118–1126, 2002.

[13] J. E. Beasley and P. C. Chu, "A genetic algorithm for the set covering problem," *European Journal of Operational Research*, vol. 94, no. 2, pp. 392–404, 1996.

[14] M. J. Brusco, L. W. Jacobs, and G. M. Thompson, "A morphing procedure to supplement a simulated annealing heuristic for cost- and coverage-correlated set-covering problems," *Annals of Operations Research*, vol. 86, pp. 611–627, 1999.

[15] M. Caserta, "Tabu search-based metaheuristic algorithm for large-scale set covering problems," in *Metaheuristics: Progress in Complex Systems Optimization*, K. F. Doerner, M. Gendreau, P. Greistorfer, W. Gutjahr, R. F. Hartl, and M. Reimann, Eds., vol. 39 of *Operations Research/Computer Science Interfaces Series*, pp. 43–63, Springer, 2007.

[16] B. Crawford, R. Soto, and E. Monfroy, "Cultural algorithms for the set covering problem," in *Advances in Swarm Intelligence*, Y. Tan, Y. Shi, and H. Mo, Eds., vol. 7929 of *Lecture Notes in Computer Science*, pp. 27–34, Springer, 2013.

[17] B. Crawford, C. Lagos, C. Castro, and F. Paredes, "A evolutionary approach to solve set covering," in *Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS '07)*, pp. 356–360, June 2007.

[18] B. Crawford, C. Castro, and E. Monfroy, "A new ACO transition rule for set partitioning and covering problems," in *Proceedings of the International Conference on Soft Computing and Pattern Recognition (SoCPaR '09)*, pp. 426–429, December 2009.

[19] 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.

[20] Y. Zhang and L. Wu, "Artificial bee colony for two dimensional protein folding," *Advances in Electrical Engineering Systems*, no. 1, pp. 19–23, 2012.

[21] Y. Zhang and L. Wu, "Optimal multi-level thresholding based on maximum Tsallis entropy via an artificial bee colony approach," *Entropy*, vol. 13, no. 4, pp. 841–859, 2011.

[22] Y. Zhang, L. Wu, S. Wang, and Y. Huo, "Chaotic artificial bee colony used for cluster analysis," in *Intelligent Computing and Information Science*, R. Chen, Ed., vol. 134 of *Communications in Computer and Information Science*, pp. 205–211, Springer, Heidelberg, Germany, 2011.

[23] Y. Zhang, L. Wu, and S. Wang, "Magnetic resonance brain image classification by an improved artificial bee colony algorithm," *Progress in Electromagnetics Research*, vol. 116, pp. 65–79, 2011.

[24] Y. Zhang, L. Wu, and S. Wang, "UCAV path planning based on FSCABC," *Information*, vol. 14, no. 3, pp. 687–692, 2011.

[25] B. Akay and D. Karaboga, "Parameter tuning for the artificial bee colony algorithm," *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, Springer, Berlin, Germany, vol. 5796, pp. 608–619, 2009.

[26] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR06, Computer Engineering Department, Erciyes University, Kayseri, Turkey, 2005.

[27] 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.

[28] B. Crawford, R. Soto, E. Monfroy, C. Castro, W. Palma, and F. Paredes, "A hybrid soft computing approach for subset problems," *Mathematical Problems in Engineering*, vol. 2013, Article ID 716069, 12 pages, 2013.

[29] Z.-G. Ren, Z.-R. Feng, L.-J. Ke, and Z.-J. Zhang, "New ideas for applying ant colony optimization to the set covering problem," *Computers & Industrial Engineering*, vol. 58, no. 4, pp. 774–784, 2010.

[30] F. Glover and G. A. Kochenberger, *Handbook of Metaheuristics*, chapter 17, Springer, New York, NY, USA, 2003.

[31] C. Valenzuela, B. Crawford, R. Soto, E. Monfroy, and F. Paredes, "A 2-level metaheuristic for the set covering problem," *International Journal of Computers, Communications & Control*, vol. 7, no. 2, pp. 377–387, 2012.

[32] 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.

[33] 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.

[34] B. Crawford, C. Castro, E. Monfroy, R. Soto, W. Palma, and F. Paredes, "A hyperheuristic approach for guiding enumeration in constraint solving," in *A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation II (EVOLVE '12)*, vol. 175 of *Advances in Intelligent Systems and Computing*, pp. 171–188, Springer, 2012.

[35] M. V. Krieken, H. Fleuren, and M. Peeters, "Problem reduction in set partitioning problems," Discussion Paper 2003-80, Center for Economic Research, Tilburg University, 2003.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 262368, 12 pages
http://dx.doi.org/10.1155/2014/262368

Research Article

Improved GSO Optimized ESN Soft-Sensor Model of Flotation Process Based on Multisource Heterogeneous Information Fusion

Jie-sheng Wang,[1,2] Shuang Han,[1] and Na-na Shen[1]

[1] *School of Electronic and Information Engineering, University of Science & Technology Liaoning, Anshan 114044, China*
[2] *National Financial Security and System Equipment Engineering Research Center, University of Science & Technology Liaoning, Anshan 114044, China*

Correspondence should be addressed to Jie-sheng Wang; wang_jiesheng@126.com

Received 19 February 2014; Accepted 18 March 2014; Published 10 April 2014

Academic Editors: P. Agarwal and V. Bhatnagar

Copyright © 2014 Jie-sheng 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.

For predicting the key technology indicators (concentrate grade and tailings recovery rate) of flotation process, an echo state network (ESN) based fusion soft-sensor model optimized by the improved glowworm swarm optimization (GSO) algorithm is proposed. Firstly, the color feature (saturation and brightness) and texture features (angular second moment, sum entropy, inertia moment, etc.) based on grey-level co-occurrence matrix (GLCM) are adopted to describe the visual characteristics of the flotation froth image. Then the kernel principal component analysis (KPCA) method is used to reduce the dimensionality of the high-dimensional input vector composed by the flotation froth image characteristics and process datum and extracts the nonlinear principal components in order to reduce the ESN dimension and network complex. The ESN soft-sensor model of flotation process is optimized by the GSO algorithm with congestion factor. Simulation results show that the model has better generalization and prediction accuracy to meet the online soft-sensor requirements of the real-time control in the flotation process.

1. Introduction

Based on the differences of the surface property of solid materials, flotation process is to separate useful minerals with gangue [1], in which the economic and technical indexes (concentrate grade and flotation recovery rate) are the key controlled indicators in the production process. Their control in the flotation process is mainly according to the flotation operators' experiences by observing the states (such as the color, the size, the flow rate, texture features, etc.) of the flotation froth on the flotation cell surface to adjust the flotation tank level and change the pharmacy addition. This method of artificial observation on flotation froth has limitations of the space, time, and subjectivity, and it cannot be organically combined with computer control system to achieve high-level control [2, 3]. Inferential estimation (soft-sensor) technology can effectively solve the problem that the flotation process is difficult to online estimate the economic and technical indicators.

Domestic and foreign scholars apply digital image processing techniques to the froth feature extraction and the soft-sensor modeling of the key technical indicators in the flotation process and make a lot of achievements [4–11]. Hargrave and Hall study the diagnosis and analysis methods of the metal grade and quality and flow rate in flotation process by using the color and surface tissue. Then the statistical methods and mathematical models are utilized to find the relationship between parameters. The research results show that the color parameters of flotation froth can be used to forecast the concentrate grade in the beneficiation production process [5]. Bartolacci et al. use multivariate image analysis (MIA) and partial least squares (PLS) methods to establish the experience prediction model of flotation grade. On the other hand, the GLCM and wavelet transform analysis (WTA) methods are utilized to get the flotation froth characteristics [6]. Morar et al. utilize the machine vision method to predict the performance of the flotation process, such as concentrate grade and tailings recovery rate [7].

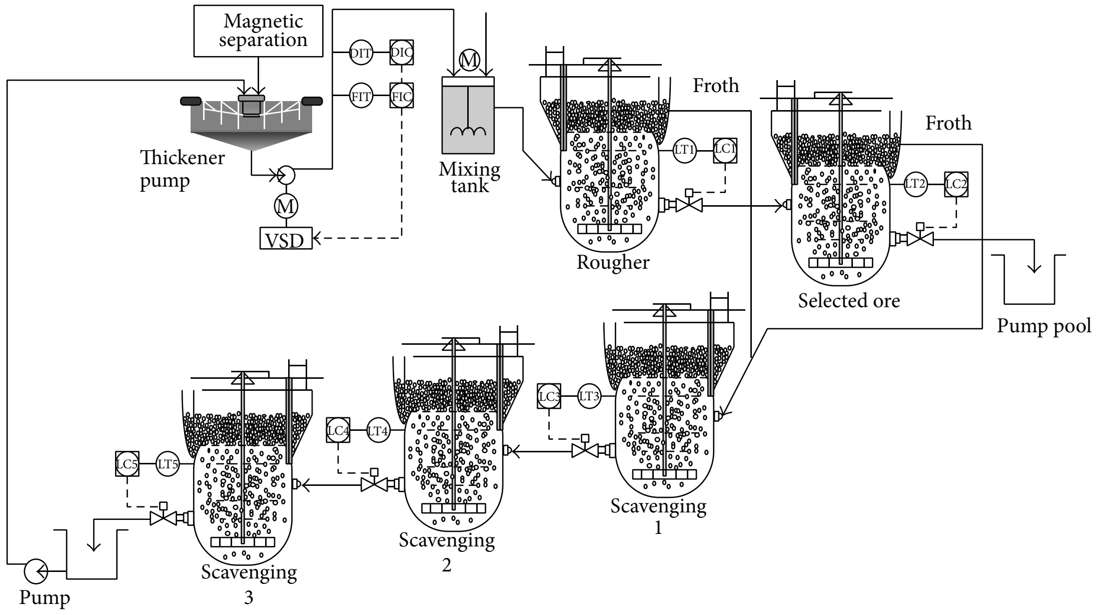


Figure 1: Technique flowchart of flotation process.

At home, Yang et al. aim at the question that bubble image quality is not ideal and the bubbles' size, shape, and gray scale are uneven in mineral flotation process and put forward a bubble image segmentation method based on the clustering presplit and the accuracy distance reconstruction. The size of the bubbles and other physical characteristics provide the basis for flotation control [8]. In that the multiple models additive can improve the overall prediction accuracy and the idea of robustness, Wang et al. present a multi-T-S fuzzy neural network soft-sensor model of flotation process based on the FCM clustering algorithm [9]. Yang et al. use the flotation froth video image features as auxiliary variables, establish a soft-sensor model of the flotation pulp pH value based on the sparse polynuclear least squares support vector machine (SVM), which combines the weighted local kernel function and global kernel function, and use Schmidt orthogonalization theory to reduce the multinuclear matrix [10]. Li et al. set up a soft-sensor mode by combining the principal component analysis (PCA) and extreme learning machine (ELM) methods [11].

The above established soft-sensor models of the flotation process only make use of the part of multisource heterogeneous information (real-time process datum, image feature information, and laboratory datum), not realizing information integration, coordination, and optimization of flotation process. The paper proposes ESN fusion soft-sensor method based on process datum and flotation froth image visual characteristic parameters (color features and texture features). Simulation results demonstrate the effectiveness of the proposed method.

The paper is organized as follows. In Section 2, the technique flowchart of flotation process is introduced. The ESN fusion soft sensing model of flotation process based on improved glowworm swarm optimization algorithm is presented in Section 3. In Section 4, experiment and simulation results are introduced in details. Finally, the conclusion illustrates the last part.

2. Technique Flowchart of Flotation Process

Flotation process is used to separate useful minerals and gangue based on the differences of the surface property of solid materials. Figure 1 is a typical iron ore flotation process consisting of the roughing, concentration, and scavenging [11]. The system input is the fine concentrate pulp which is early output of beneficiation process in the forepart. The pulp density is about 38% and concentrate grade is about 64%. Inlet pulp is fed into the high-stirred tank through the pulp pipeline by feed pump. At the same time, the flotation reagent according to a certain concentration ratio is also fed into high-stirred tank through dosing pump. On the other hand, the pulp temperature must reach a suitable flotation temperature by heating. If the dosage is appropriate, the flotation cells can output a grade of 68.5%–69.5% concentrate.

The control objective of flotation process is to ensure the concentrate grade and the tailings recovery rate are within a certain target range. In common, based on the off-line artificial laboratory to get grade values, the operators adjust the flotation cell level and the amount of flotation reagent addition. Due to the artificial laboratory for two hours at a time, when the process variables and boundary conditions change in the flotation process, they cannot timely adjust the flotation operation variables, which results in such phenomena that the flotation concentrate grade and the tailings recovery rate are too high or too low [11]. By analyzing the flotation technique, the process variables and boundary conditions mainly include feed grade x_1, feed flow rate x_2, feed concentration x_3, feed granularity x_4, and medicament flow rate x_5.

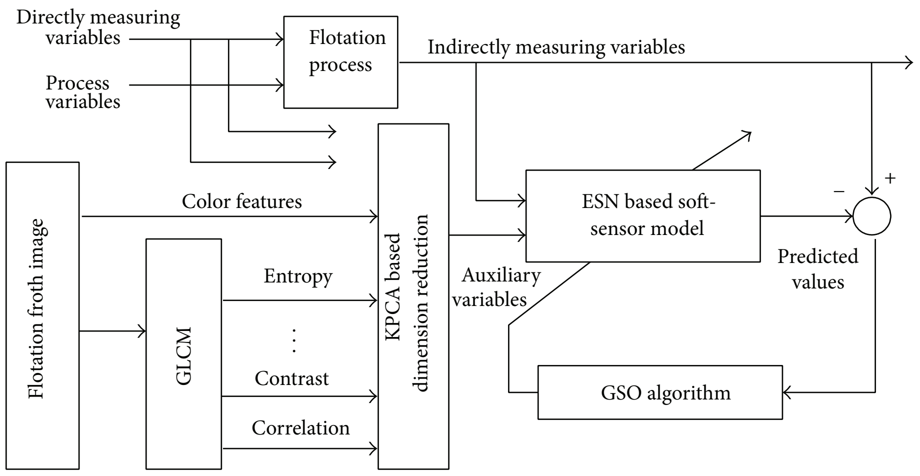


Figure 2: Soft-sensor model structure of flotation process.

3. Soft-Sensor Modeling of Flotation Process

3.1. Structure of Soft-Sensor Model. The structure of the proposed ESN soft-sensor model optimized by the glowworm swarm optimization algorithm is shown in Figure 2 [12].

The auxiliary variables of the proposed soft-sensor model are process variables, color features, and texture features (14 texture parameters based on the calculated gray-level co-occurrence matrix by flotation froth images, such as angular second moment, contrast, correlation, sum of squares, inverse difference moment, etc.). Then KPCA method is used to realize the dimension reduction of the high-dimensional input vector composed by the normalized auxiliary variables datum in order to reduce the ESN complex. Finally, the ESN network structure parameters are optimized by the improved GSO algorithm to realize the accurate prediction of the concentrate grade and tailings recovery rate in the flotation process.

Considering a multiinput multioutput (MIMO) system, the training sample set can be expressed as $D = \{Y, X_i \mid i = 1, 2 \ldots, m\}$. Y is the output variable. X_i represents the ith input vector and can be expressed as $X_i = [x_{1i}, x_{2i}, \ldots, x_{ni}]'$ (n is the number of samples in the training set and m is the number of input variables). Soft-sensing modeling requires a datum set from the normal conditions as the modeling data. Assume that the system has m process variable and n data vectors composing the test sample datum matrix $X \in R^{n\times m}$. In order to avoid the different dimension of the process variables affecting the results and obtain the easy mathematical treatment, it is necessary to normalize the datum. Set μ is the mean vector of X and σ is the standard deviation vector of X. So the normalized process variable is expressed as follows:

$$\widehat{X} = \frac{X - \mu}{\sigma}. \tag{1}$$

The input vector $\widehat{X}$ of the training samples is fed into the ESN to obtain the predicted output $\widehat{Y}$. Then the root mean square error (RMSE) is selected as the fitness of the soft-sensor model:

$$\text{RMSE} = \sqrt{\frac{\sum_{k=1}^{n} \left(\widehat{Y}_k - Y_k^*\right)^2}{n}}, \tag{2}$$

where Y^* is the actual output of training samples.

3.2. Extraction of Flotation Froth Color Features. Flotation froth images are obtained by the CCD camera above the flotation tank, and the computer image acquisition card converts continuous analog signal into discrete digital signals, which is conveyed into the computer for the extraction of visual characteristics of flotation froth.

Typical flotation froth image is shown in Figure 3 [13]. The froth images can be divided into three categories according to the flotation process and expert experience. (1) Bubble size is bigger, that is to say the big bubbles are mixed in the froth, the texture is shallow, texture is coarse, image complexity is small, color is pale, and SiO_2 of froth is less. In this case, the refined iron ore grade is low. (2) Bubble size is appropriate, more uniform, and stable, color is gray, the texture is fine, and the image is more complex. At this time, the flotation process is stable and the refined iron ore slurry grade meets the requirements. (3) Froth color becomes darker, even partial black, froth is finer, and even some bubbles are difficult to distinguish, and texture is very complicated. At this case, the SiO_2 content of froth is higher, although the iron concentrate grade is higher and the pharmaceutical is added excessively, which does not meet the economic requirements.

Flotation operators are mainly based on the color and gray closeness of the flotation froth, the luminance information of the froth surface, and the measured process variables to realize the real-time optimal control of the flotation process. Therefore, the froth color (or light intensity) reflects the information of the minerals concentration in the surface froth. The collected images are *RGB* true color images, which adopt the red, green, and blue three components. But they are often closely related. In addition, the color information of

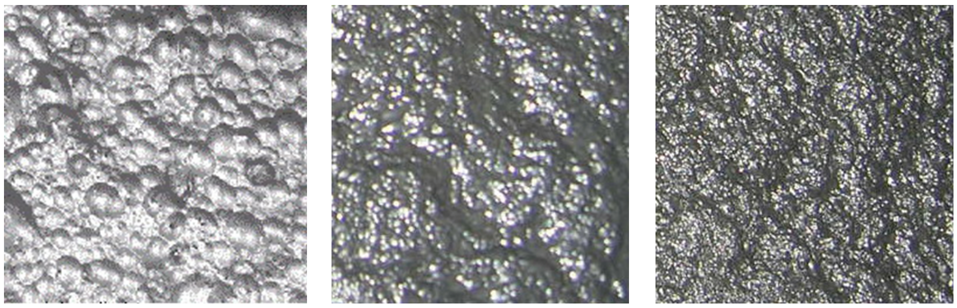


Figure 3: Typical iron ore flotation froth images.

hue, saturation, and intensity (*HIS* model) is relatively close to the people color visual perception. In *HIS* model, H is hue representing the different colors such as red, green, and blue; S is saturation representing the color, such as magenta, red; I is the brightness indicating the brightness level of the color. In industrial applications, the range of S is $[0, 1]$ corresponding from unsaturated to fully saturated state (without any white). The range of I is $[0, 1]$ corresponding from dark to light color. The conversion equations from *RGB* to *HIS* are expressed as follows:

$$I = \frac{R+G+B}{3}, \qquad S = 1 - \frac{3}{R+G+B}\left[\min(R, G, B)\right]. \tag{3}$$

Thus, the saturation (S) and brightness (I) of the flotation froth images have got to be applied in representing the relationship between the concentrate grade, the tailings recovery rate, and the color characteristics of flotation froth images.

3.3. Extraction of Flotation Froth Texture Features Based on GLCM. The texture statistical characteristics of the flotation froth image can reflect the working conditions of the flotation process. Image texture is formed by different gray values distributed in the space position and repeated alternate changes; thus two pixels will exist in a certain gradation relationship, which is known as the correlation characteristics of the gray space. Gray-level co-occurrence matrix (GLCM) is an important method used to analyze the image texture features, which is based on the second combination condition probability density function of estimated image [14]. Figure 4 is a GLCM schematic diagram, wherein i and j denote the gray scale value of the corresponding pixel.

The GLCM means a kind of statistical form of the joint distribution of two pixels, that is to say, the simultaneous occurrence probability $P(i, j, \delta, \theta)$ of two pixels. They are the pixel with gray scale i from the image $f(x, y)$ and the pixel $(x+\Delta x, y+\Delta y)$ with gray scale j at declination θ and distance δ. The mathematical formula is expressed as follows:

$$P(i, j, \delta, \theta) = \{[(x, y), (x+\Delta x, y+\Delta y)] \mid f(x, y) = i, f(x+\Delta x, y+\Delta y) = j;\ x = 0, 1, \ldots, N_x - 1;\ y = 0, 1, \ldots, N_y - 1\}, \tag{4}$$

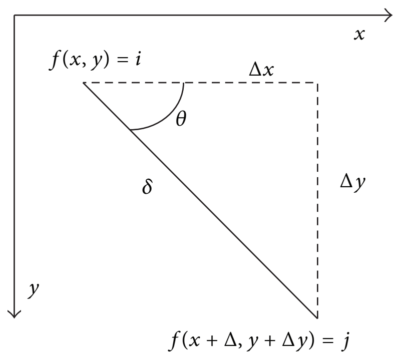


Figure 4: Grey-level co-occurrence matrix.

where $i, j = 0, 1, \ldots, L-1$, x and y are the coordinates of the image pixel, and L is the image gray level.

According to the above definitions, the element in ith row and jth column of the constituted GLCM represents the appearance frequency of all pixels with the i and j gray level in the θ direction and δ length. GLCM has rich characteristics parameters describing the image textures with different angles. Haralick et al. [15] once proposed 14 GLCM based texture parameters, whose calculation formulas are shown in Table 1.

3.4. KPCA Based Dimension Reduction of Soft-Sensor Model. The visual characteristic parameters (2 color features and 14 texture features) of the flotation froth images and 5 process variables are served as the input variables of the ESN fusion soft-sensor model to predict the concentrate grade and flotation recovery rate. A batch of flotation froth images and the measured values of the process variables in corresponding period are collected to establish the soft-sensor model. The input-output samples are shown in Table 2.

The flotation froth image characteristics and process variables and boundary conditions are selected as the auxiliary variables of the proposed soft-sensor model to realize the integration of multisource heterogeneous information in the flotation process. But there are the problems of the jumbled information and repeated expression. If the input vector dimension of the ESN model is too long, the network topology will be complex and training will become very complex. Therefore, the kernel principal component analysis (KPCA) method [16] is adopted to reduce the model dimension of the ESN soft-sensor model.

Table 1: Grey-level co-occurrence matrix.

Texture features	Calculation equations
ASM: angular second moment	$f_1 = \sum_{i=1}^{L}\sum_{j=1}^{L}\{P(i,j)\}^2$
Contrast	$f_2 = \sum_{n=0}^{L-1} n^2 \left\{ \sum_{\substack{i=1 \\ \lvert i-j\rvert=n}}^{L}\sum_{j=1}^{L} P(i,j) \right\}$
Correlation	$f_3 = \dfrac{\sum_{i=1}^{L}\sum_{j=1}^{L}(ij)P(i,j)-\mu_x\mu_y}{\sigma_x\sigma_y}$
SS: sum of squares	$f_4 = \sum_{i=1}^{L}\sum_{j=1}^{L}(i-\mu)^2 P(i,j)$
IDM: inverse difference moment	$f_5 = \sum_{i=1}^{L}\sum_{j=1}^{L}\dfrac{1}{1+(i-j)^2}P(i,j)$
SA: sum average	$f_6 = \sum_{i=2}^{2L} iP_{x+y}(i)$
SV: sum variance	$f_7 = \sum_{i=2}^{2L}(i-f_8)^2 P_{x+y}(i)$
SE: sum entropy	$f_8 = -\sum_{i=2}^{2L} P_{x+y}(i)\log\{P_{x+y}(i)\}$
Entropy	$f_9 = -\sum_{i=1}^{L}\sum_{j=1}^{L}P(i,j)\log\{P(i,j)\}$
DV: difference variance	$f_{10} = \text{variance of } P_{x-y}$
DE: difference entropy	$f_{11} = 0\sum_{i=0}^{L-1}P_{x-y}(i)\log\{P_{x-y}(i)\}$
IOC: information measures of correlation	$f_{12} = \dfrac{HXY-HXY1}{\max\{HX,HY\}}$ $f_{13} = (1-\exp[-2.0(HXY2-HXY)])^{1/2}$ $HXY = -\sum_i\sum_j P(i,j)\log(P(i,j))$ Where HX and HY are the entropy of P_x and P_y. $HXY1 = -\sum_i\sum_j P(i,j)\log(P_x(i)P_y(j))$ $HXY2 = -\sum_i\sum_j P_x(i)P_y(j)\log\{P_x(i)P_y(j)\}$
MCC: Maximal Correlation Coefficient	$f_{14} = (\text{second largest eigenvalue of } Q)^{1/2}$ $Q(i,j) = \sum_k \dfrac{P(i,k)P(j,k)}{P_x(i)P_y(k)}$

KPCA is a nonlinear promotion of introducing the concept of the kernel function into the principal component analysis (PCA) method, which has better ability to handle nonlinear problems than PCA. Its basic principle is described as follows [17].

Given sample set x_i ($i = 1, 2, \ldots, M$) and $x_i \in R^N$, the nonlinear mapping relation is given as follows:

$$\phi : R^N \longrightarrow F, \qquad x \longmapsto \phi(x). \tag{5}$$

So the sample x_i is mapped as $\phi(x_i)$. Then the covariance matrix of new sample space is calculated according to the following equation:

$$R = \frac{1}{M}\sum_{i=1}^{M}\phi(x_i)(x_i)^T. \tag{6}$$

The eigenvalue decomposition is carried out according to the following equation:

$$\lambda Q = RQ, \tag{7}$$

Table 2: Predictive data set of the soft-sensor model.

Serial number	Color features		Texture features				Process variables and boundary conditions					Predicted variables	
	Saturation	Brightness	f_1	f_2	$\cdots$	f_{14}	Feed grade (%)	Feed flow rate (m^3/h)	Feed concentration (%)	Feed granularity (%)	Pharmacy flow rate (L/min)	Concentrate grade (%)	Tailings recovery rate (%)
1	0.077	0.633	0.107	1.154	$\cdots$	0.268	62.76	329	35	90	15.5	70.51	97.7
2	0.061	0.602	0.131	1.168	$\cdots$	0.472	63.67	297	35	90	11.5	69.74	97.2
3	0.059	0.591	0.147	1.359	$\cdots$	0.502	65.07	285	37	92	11.3	69.69	97.0
4	0.038	0.464	0.103	2.786	$\cdots$	0.618	65.48	214	36	95	7.5	68.98	93.5
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
300	0.034	0.415	0.076	2.125	$\cdots$	0.584	65.9	310	36	96	5.5	67.29	90.2

where λ ($\lambda > 0$) is the eigenvalue of R and Q is the corresponding eigenvector. By multiplying $\phi(x_i)$ on both sides of (7), we obtain the following:

$$\lambda(\phi(x_i)\cdot Q) = (\phi(x_i)\cdot RQ), \quad i = 1,2,\ldots,M. \tag{8}$$

And coefficient α_i ($i = 1,2,\ldots,M$) exists to make the following equation:

$$Q = \sum_{i=1}^{M}\alpha_i\phi(x_i). \tag{9}$$

By combining the above two equations, matrix $K(M\times M)$ is defined as follows:

$$\begin{gathered}\lambda\sum_{i=1}^{M}\alpha_i(\phi(x_k),\phi(x_i)) \\ = \frac{1}{M}\sum_{i=1}^{M}\alpha_i\left(\phi(x_k),\sum_{j=1}^{M}\phi(x_j)\right)(\phi(x_j),\phi(x_i)), \\ K_{i,j} = (\phi(x_i)\phi(x_j)) = K(x_i,x_j).\end{gathered} \tag{10}$$

Set α is the corresponding eigenvector of the kernel matrix K. Then, consider the following:

$$K\alpha = M\lambda\alpha, \tag{11}$$

where $\alpha = (\alpha_1,\alpha_2,\ldots,\alpha_M)^T$.

Assume that the solution of (11) is $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_P \geq \cdots \geq \lambda_M$. λ_P is the last nonzero eigenvalue, whose corresponding eigenvector is $\alpha_1^k,\ldots,\alpha_p^k,\ldots\alpha_M^k$. Then the eigenvector of F is normalized according to the following equation:

$$\left(Q^k\cdot Q^k\right) = I, \quad k = 1,2,\ldots,p. \tag{12}$$

Put $Q = \sum_{i=1}^{M}\alpha\phi(x_i)$ and $K_{ij} = (\phi(x_i)\phi(x_j))$ into (12) to obtain the following:

$$\begin{aligned}I &= \sum_{i,j=1}^{M}\alpha_i^k\alpha_j^k(\phi(x_i)\phi(x_j)) = \sum_{i.j=1}^{M}\alpha_i^k\alpha_j^kK_{ij} \\ &= \alpha^kK\alpha^k = \lambda_k\left(\alpha^k\cdot\alpha^k\right), \quad k = 1,2,\ldots,p.\end{aligned} \tag{13}$$

The principal component of a new sample x_i is obtained by projecting mapping sample $\phi(x)$ of F into Q^k, which is described in the following equation:

$$\left(Q^k\cdot\phi(x)\right) = \sum_{j=1}^{M}\alpha_j^k(\phi(x_i)\phi(x)) = \sum_{j=1}^{M}\alpha_j^kK(x_j,x). \tag{14}$$

For the sake of simplicity, $\widehat{K} = K - I_MK - KI_M + I_MKI_M$ is used to substitute kernel matrix of all mapping samples, among which $(I_M)_{ij} = 1/M$. The paper adopts the Gaussian function as the KPCA kernel function, which is described as follows:

$$K(x_j,x) = \exp\left\{-\frac{\left|x_j - x\right|^2}{\sigma^2}\right\}. \tag{15}$$

Based on the above mentioned principles, the procedure of KPCA algorithm is described as follows.

Calculate kernel matrix $\widehat{K}$;

calculate eigenvalues and eigenvectors of kernel matrix $\widehat{K}$;

sort eigenvalues with the descend order. Assume that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$. Calculate the contribution ratio by (16) to decide the number of the extracted character information (p). Consider the following:

$$\varphi(p) = \frac{\sum_{i=1}^{p}\lambda_i}{\sum_{i=1}^{M}\lambda_i}. \tag{16}$$

(1) The eigenvectors in accordance with the previous p ($1 \leq p \leq M$) biggest eigenvalues are normalized according to (13).

(2) Calculate a new principal component according to (14).

The historical datum of input variables in the soft-sensor model is carried out by kernel principal component analysis, whose results are described in the Table 3. It can be seen that

TABLE 3: Contribution rate of principle component.

Number of principal components	Percentage of variance (%)	Cumulative percentage of variance (%)
1	45.72	45.72
2	21.82	67.52
3	13.42	80.94
4	7.14	88.08
5	2.15	90.23
⋮	⋮	⋮
21	0.07	100.00

the contribution ratio of the previous 5 principal components has already exceeded 90%. Thus, the principal components obtained by the KPCA on the original variables datum are the input of the ESN, which not only reserved the character information of original variables but also simplified the structure of ESN.

3.5. Echo State Network. Echo state network (ESN) is a new type of recurrent neural network proposed by Jaeger [18–20]. Its internal dynamic reserve (Dynamic Reservoir, DR) pool has a large number of sparse connected neural units, which contain the operational status of the system and have the short-term memory function (the ESN echo effect). The echo effect makes the network realize the approximation effect on the learning system. A typical ESN structure is shown in Figure 5. Its basic equations can be represented as follows:

$$x(k+1) = f\left(W^{\text{in}}u(k+1) + Wx(k) + W^{\text{fb}}y(k)\right),$$
$$y(k+1) = f^{\text{out}}\left(W^{\text{out}}\left(u(k+1), x(k+1), y(k)\right)\right), \tag{17}$$

where f is the DR internal activation function, usually using the Sigmoid type function to make the ESN have good nonlinear characteristic; $x(k)$ is the DR state variable on k time; $u(k)$ is the system input vector on k time; $y(k)$ is the network output; $W^{\text{in}}(N \times K)$ is the input weight matrix; $W(N \times N)$ is the connection matrix among the DR internal neurons, which usually keeps the sparse connection of 1%~5% and the spectral radius less than 1 in order to make the DR have dynamic memory ability; $W^{\text{fb}}(N \times L)$ is the feedback matrix between the output neurons and DR; f^{out} is the activation function of the input and output units, usually using a linear function; $W^{\text{out}}\ (L\times(K+N+L))$ is the output weights matrix. W^{in}, W, and W^{fb} are constructed before the network learning, but W^{out} is calculated after learning period.

3.6. ESN Soft-Sensor Model Optimized by Improved GSO Algorithm

3.6.1. Glowworm Swarm Optimization Algorithm. The glowworm swarm optimization (GSO) algorithm is a new swarm intelligent optimization algorithm proposed by Krishnanand et al. in 2005, which intimates the firefly's phenomena, such as self-luminous, communication, courtship and foraging, and

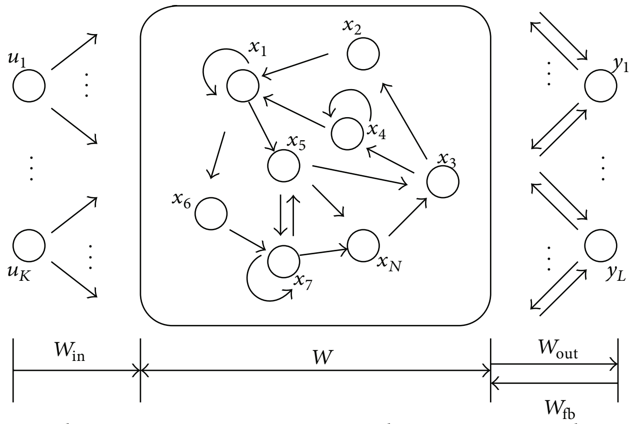


FIGURE 5: Diagram of an echo state network.

so on. GSO has been successfully used on many fields, such as multimodal function optimization and multisource tracking and location [21, 22]. Suppose the number of fireflies is n, which is randomly distributed in the search space of objective function. $x_i(t)$ represents the location of the ith firefly, $J(x_i(t))$ is the fitness function, and $l_i(t)$ is the fluorescein concentration of the ith firefly at the moment t. The movements of fireflies are updated according to the following equation:

$$l_i(t) = (1-\rho)\,l_i(t-1) + \gamma J(x_i(t)), \tag{18}$$

where $\rho \in (0,1)$ is the volatilization coefficient of fluorescein and γ is the enhancement factor of the volatilization coefficient. Suppose r_s is the perception scope of fireflies and $r_d^i(t)$ is the dynamic decision range (namely, decision radius) belonging to the ith firefly at the moment t, whose upper bound of the perception scope is r_s $(0 < r_d^i(t) < r_s)$. So the updating formula of decision domain range is represented as

$$r_d^i(t+1) = \min\left\{r_s, \max\left\{0, r_d^i(t) + \beta\left(n_t - |N_i(t)|\right)\right\}\right\}, \tag{19}$$

where β is the changeable rate of field, n_t is the neighborhood threshold controlling the neighbor number of fireflies, and $N_t(t)$ is neighbors set of the ith firefly at the moment t. Then the formula determining the number of fireflies within the decision domain is

$$N_t(t) = \left\{j : \left\|x_j(t) - x_i(t)\right\| < r_d^i(t); l_i(t) < l_j(t)\right\}, \tag{20}$$

where $\|\vec{x}\|$ is the norm of $\overrightarrow{x}$.

During the movement of fireflies, the fluorescein concentration of each firefly in its neighbor set determines the moving direction. Suppose that $p_{ij(t)}$ is the moving probability of the ith firefly moving to the jth firefly in the neighbor set at the moment t, which is calculated by the following equation:

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k\in N_i(t)} l_k(t) - l_i(t)}. \tag{21}$$

Based on the moving probability $p_{ij(t)}$, the roulette method is adopted to decide the moving direction of the ith

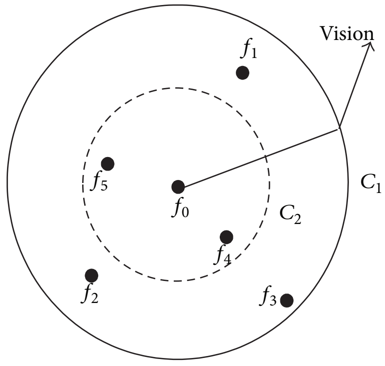


FIGURE 6: Attraction behavior description among glowworms.

firefly. l_0 is the initial fluorescein value. Suppose the moving step s. Thus the following formula determines the location of the ith firefly at the moment $t+1$:

$$x_i(t+1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\left\| x_j(t) - x_i(t) \right\|}\right). \tag{22}$$

3.6.2. Crowded Degree Factor. The crowded degree factor δ is introduced to avoid the local optimum phenomena caused by the overcrowding fireflies, which can make fireflies located near the optimum point reject each other. Its principle is shown in Figure 6. For maximum optimization problem, consider the following:

$$\delta = \frac{1}{an_{\max}}, \quad a \in (0,1], \tag{23}$$

where a is the close level to the optimum value and $n_{\max}$ is the maximum number of fireflies belonging to its neighbor field. Suppose Y_i is the states of fireflies themselves, Y_c is the preceptor state value, and n_f is the number of partners in fireflies' neighborhood. If $Y_c/(Y_i n_f) < \delta$, Y_c is the overcongestion state. When it comes to the minimum optimization problem, $\delta = an_{\max}$, $a \in (0,1]$. When $Y_c n_f/(Y_i) > \delta$, Y_c is at the state of over congestion.

Unifying the crowded degree factor and the number of fireflies in their neighborhood, the behavior of the fireflies attracting each other determines its influence on the optimization results. In Figure, the firefly f_0 is the best one among the fireflies f_1, f_2, f_3, f_4, and f_5, whose attracting degree is Y_j. C_1 and C_2 are rounds having the same center f_0 and different radiuses. The closer to the round center the much greater attraction to the fireflies.

If $\delta n_f \leq 1$, all fireflies are attracted to f_0. If $\delta n_f > 1$ and C_2 (attracting degree is $Y_j/\delta n_f$) is the attracting degrees circle, the fireflies between C_1 and C_2 are attracted to f_0. At this moment, the larger δn_f is, the less the fireflies are attracted. If $n_f/\delta \leq 1$, all fireflies in the vision are attracted to f_0. If $n_f/\delta > 1$, the fireflies, whose degree is greater than $Y_j n_f/\delta$, are attracted to move to f_0. The larger n_f/δ is, the less the fireflies are attracted.

3.6.3. Algorithm Procedure. The main parameters of ESN soft-sensor model are the input weight matrix W^{in}, the DR pool weight matrix W, the output feedback weight matrix W^{fb}, and the output weight matrix W^{out}. There are two kinds of cases to optimize the ESN: one is to optimize W^{in}, W, and W^{fb}; the other is to optimize W^{out}. The paper adopts the locations of the fireflies in the improved GSO algorithm to correspond with the output connection weights matrix W^{out} of ESN during the ESN training stage. Through the optimized search, the output weight matrix W^{out} of ESN is trained in less samples and time. Its algorithm procedure is shown as follows.

Step 1 (initialize the algorithm parameters). Initialize the parameters W^{in}, W, and W^{fb} of ESN, the parameters n, ρ, γ, β, s, n_t, l_0, and a of GSO, and the maximum iteration time $\max t$.

Step 2 (initialize population). Aiming at the output weight matrix W^{out} of ESN, randomly generate n fireflies to consist of the initial population $P = \{x_1(t), \ldots, x_i(t), \ldots, x_n(t)\}$ $(i = 1, \ldots, n)$. Set the iteration count value $t = 0$.

Step 3 (calculate fitness). Each firefly $x_i(t)$ is set as the output weight matrix of ESN, and then the training samples are fed into the ESN soft-sensor model. The predict output is calculated by (17), and the fitness value $J(x_i(t))$ is calculated by (2). In the end, (18) is used to convert the objective function values $J(x_i(t))$ of firefly $x_i(t)$ into the fluorescein value $l_i(t)$.

Step 4 (update of the individual firefly position). Each firefly within $r_d^i(t)$ makes up its neighborhood set $N_t(t)$ $(0 < r_d^i(t) < r_s)$ according to (20) by selecting those fireflies whose fluorescein values are higher to itself, and $N_t(t)$ is regulated based on the crowded degree factor. The probability $p_{ij}(t)$ that the ith firefly moves to the jth firefly in its neighborhood at the moment t is calculated by (21). The roulette wheel method is used to select individuals to move. Then the location is updated according to (22). In the end, the dynamic decision domain radius is updated according to (19).

Step 5 (judge the termination conditions of the proposed algorithm). If it meets the termination conditions (e.g., it reaches the maximum iteration number $\max t$), the best firefly is recorded. Otherwise, $t = t+1$ and go to the Step 3.

4. Simulation Results

With a typical flotation process as the research object, an ESN fusion soft-sensor model is established for predicting the concentrate grade and the flotation recovery rate. Firstly, the 300 input-output datum sets are determined as shown in Table 2 for training and testing the ESN soft-sensor model. Then the five nonlinear component variables obtained through KPCA dimension reduction processing process are selected as the input variables of the soft-sensor model. The former 240 samples are the training datum and the rest of the samples are used to verify the soft-sensor model's performances. The paper selects the normalized root mean square error (NRMSE), the mean square error (MSE), and

Table 4: Predictive error comparison of soft-sensor model.

Predicted variables	Predictive method	NRMSE	MSE	MAPE
Concentrate grade (%)	ESN	0.0122	0.6828	1.0310
	GA-ESN	0.0093	0.2379	0.6020
	PSO-ESN	0.0089	0.1918	0.5435
	GSO-ESN	0.0102	0.3562	0.7205
	IGSO-ESN	0.0069	0.0681	0.3306
Flotation recovery rate (%)	ESN	0.0096	1.0195	0.8807
	GA-ESN	0.0075	0.3593	0.5468
	PSO-ESN	0.0078	0.4158	0.5929
	GSO-ESN	0.0088	0.6626	0.7482
	IGSO-ESN	0.0060	0.1467	0.3530

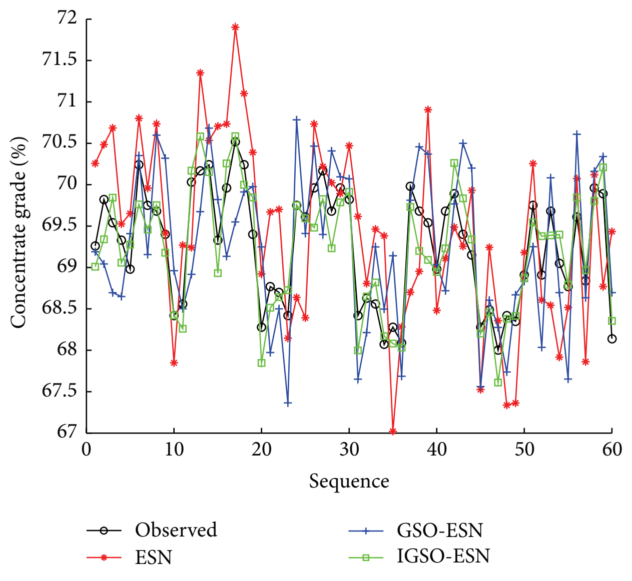
the mean absolute percentage error (MAPE) as the judgment on prediction effects [17], which are defined as follows:

$$\text{NRMSE} = \sqrt{\frac{1}{T\|y_d\|^2}\sum_{t=1}^{T}(y(t) - y_d(t))^2},$$
$$\text{MSE} = \frac{1}{T}\sum_{t=1}^{T}(y(t) - y_d(t))^2, \quad (24)$$
$$\text{MAPE} = \frac{100}{T}\sum_{t=1}^{T}\frac{|y(t) - y_d(t)|}{y_d(t)},$$

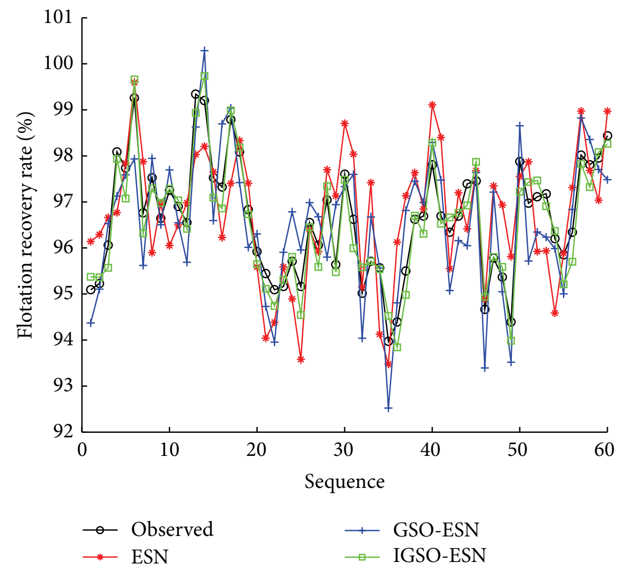
where T is the number of the predictive samples, $y(t)$ is the predicted number, and $y_d(t)$ is the actual sample values.

The input dimension of ESN is 5 and the output dimension is 2. Moreover, the size of the DR pool is 100, the sparse connection rate of weight matrix of DR pool is 5%, the activation function of DR pool is tanh(), and the output unit uses the linear activation function. The initial values of parameters of ESN are selected as follows: $W^{\text{in}} = 0.3$, $W = 0.2$, and $W^{\text{fb}} = 0.03$. The initial values of parameters of GSO are selected as follows: $n = 100$, $\rho = 0.4$, $\gamma = 0.6$, $\beta = 0.08$, $s = 0.03$, $n_t = 5$, $l_0 = 5$, and $a = 0.2$. The maximum iteration time max $t = 500$.

To illustrate the effectiveness of the proposed soft-sensor model, the improved glowworm swarm optimization (IGSO) based ESN soft-sensor model is compared with the original ESN method and the glowworm swarm optimization (IGSO) based ESN soft-sensor model. The predictive outputs and actual outputs under three methods are shown in Figure 7. The predictive error curves are shown in Figure 8. The prediction accuracies of three methods are shown in Table 4. Seen from Figures 7 and 8 and Table 4, the proposed IGSO-ESN soft-sensor model has higher predictive precision and generalization ability for the key technique index (concentrate grade and flotation recovery rate) of the flotation process than ESN soft-sensor model and GSO-ESN soft-sensor model. The proposed GSO algorithm with the crowded degree factor



(a) Concentrate grade

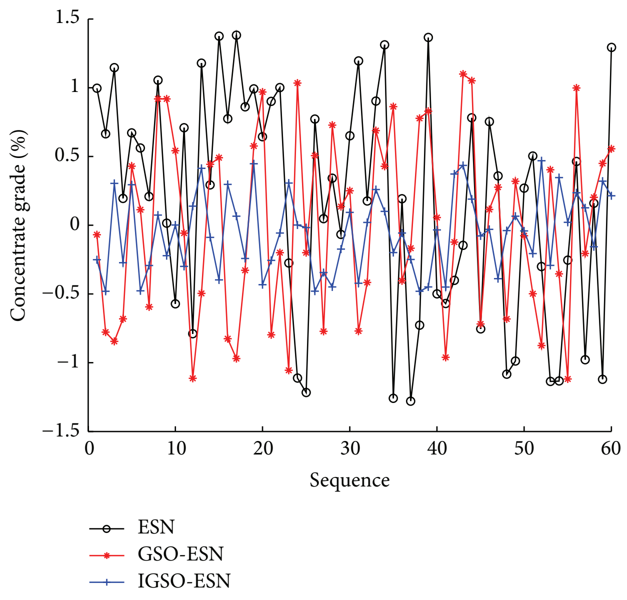


(b) Flotation recovery rate

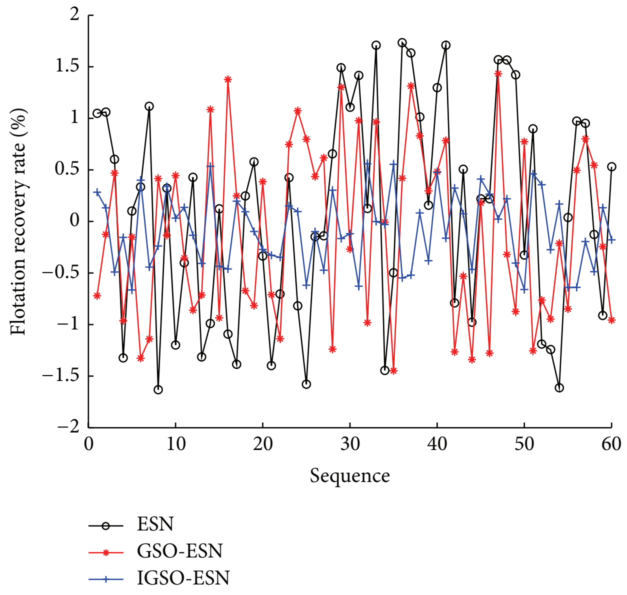
Figure 7: Predictive results of soft-sensor models (ESN, GSO-ESN, and IGSO-ESN).

can adjust the structure parameters of the soft-sensor model effectively.

In order to highlight the superiority of the proposed method, the comparisons have been made among IGSO-ESN soft-sensor models with two swarm intelligence based ESN soft-sensor models (GA-ESN and PSO-ESN). The predictive outputs and errors curves under three methods are shown in Figures 9 and 10. The predictive simulation has been carried out 10 times. Then the statistics analysis results of the model performances with 10 runs are listed in Table 4 based on the definition of predictive performance index.

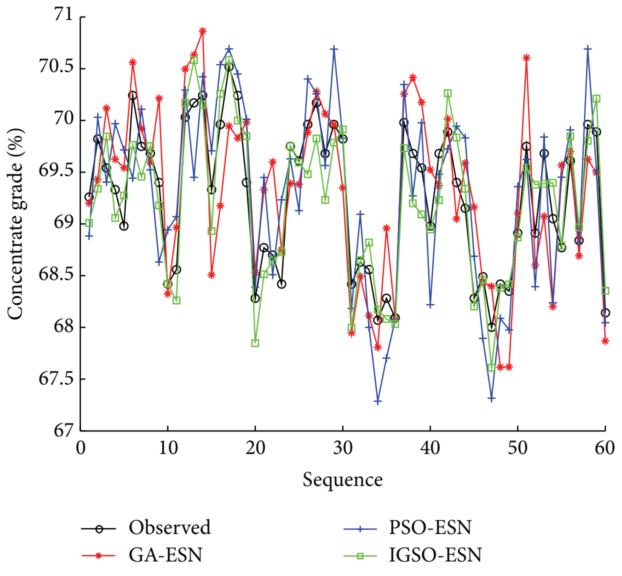


(a) Concentrate grade

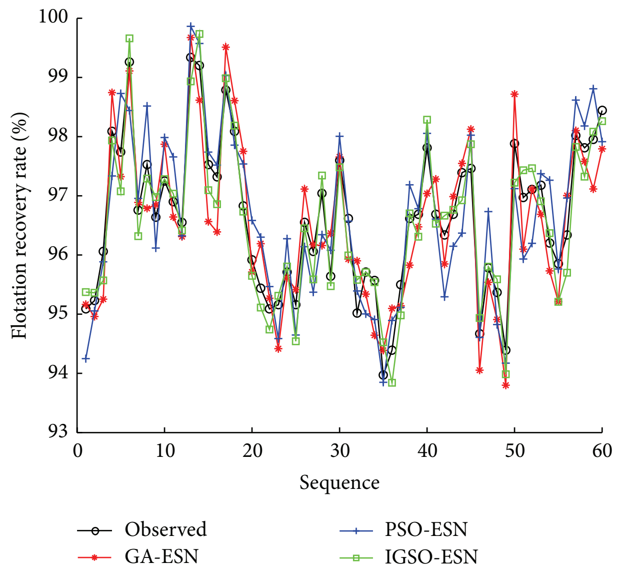


(b) Flotation recovery rate

FIGURE 8: Predictive errors of soft-sensor models (ESN, GSO-ESN, and IGSO-ESN).



(a) Concentrate grade



(b) Flotation recovery rate

FIGURE 9: Predictive results of soft-sensor models (GA-ESN, PSO-ESN, and IGSO-ESN).

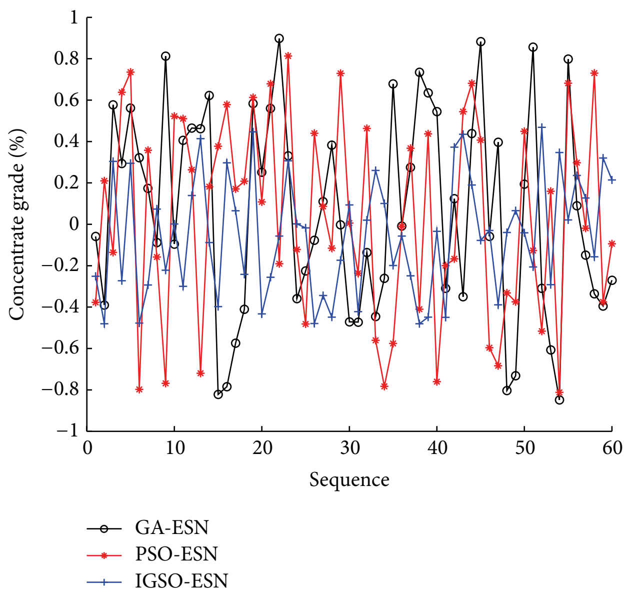
Seen from the simulation results, the proposed IGSO-ESN predictive model has higher accuracy than the GA-ESN and PSO-ESN soft-sensor model. The successful adoption of the predictive model in the flotation process for obtaining the real-time concentrate grade and flotation recovery rate has important significance in the field of improving the production capacity and reducing production costs.

5. Conclusions

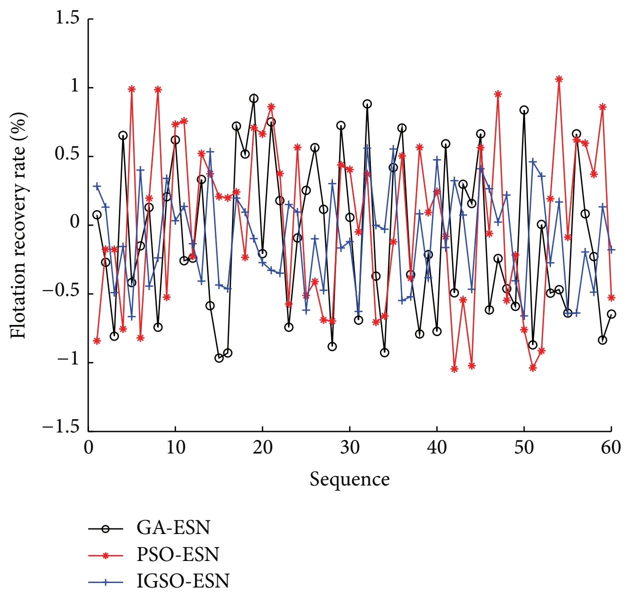
(1) For predicting the key technical index of the flotation process (concentrate grade and tailings recovery rate), an ESN fusion soft-sensor model based on the improved GSO algorithm is proposed.

(2) The fusion, coordination, and optimization of multisource heterogeneous information of flotation process are realized based on the process datum and the visual characteristic parameters of the flotation froth images (color features and texture features).

(3) KPCA method is used to reduce dimension of the high dimension input variables of the soft-sensor model to extract the nonlinear principal element.



(a) Concentrate grade



(b) Flotation recovery rate

Figure 10: Predictive errors of soft-sensor models (GA-ESN, PSO-ESN, and IGSO-ESN).

Then the GSO algorithm with the crowded degree factor is used to optimize the ESN soft-sensor model.

(4) The simulation results show the effectiveness of the proposed soft-sensor model for meeting the real-time monitoring requirements of the flotation process.

Conflict of Interests

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

Acknowledgments

This work is partially supported by the Program for China Postdoctoral Science Foundation (20110491510), the Program for Liaoning Excellent Talents in University (LJQ2011027), the Program for Anshan Science and Technology Project (2011MS11), and the Program for Research Special Foundation of University of Science and Technology of Liaoning (2011ZX10).

References

[1] D. Hodouin, S.-L. Jämsä-Jounela, M. T. Carvalho, and L. Bergh, "State of the art and challenges in mineral processing control," *Control Engineering Practice*, vol. 9, no. 9, pp. 995–1005, 2001.

[2] J. Kaartinen, J. Hätönen, H. Hyötyniemi, and J. Miettunen, "Machine-vision-based control of zinc flotation—a case study," *Control Engineering Practice*, vol. 14, no. 12, pp. 1455–1466, 2006.

[3] B. J. Shean and J. J. Cilliers, "A review of froth flotation control," *International Journal of Mineral Processing*, vol. 100, no. 3-4, pp. 57–71, 2011.

[4] C. Aldrich, C. Marais, B. J. Shean, and J. J. Cilliers, "Online monitoring and control of froth flotation systems with machine vision: a review," *International Journal of Mineral Processing*, vol. 96, no. 1-4, pp. 1–13, 2010.

[5] J. M. Hargrave and S. T. Hall, "Diagnosis of concentrate grade and mass flowrate in tin flotation from colour and surface texture analysis," *Minerals Engineering*, vol. 10, no. 6, pp. 613–621, 1997.

[6] G. Bartolacci, P. Pelletier Jr., J. Tessier Jr., C. Duchesne, P.-A. Bossé, and J. Fournier, "Application of numerical image analysis to process diagnosis and physical parameter measurement in mineral processes—part I: flotation control based on froth textural characteristics," *Minerals Engineering*, vol. 19, no. 6–8, pp. 734–747, 2006.

[7] S. H. Morar, M. C. Harris, and D. J. Bradshaw, "The use of machine vision to predict flotation performance," *Minerals Engineering*, vol. 36–38, pp. 31–36, 2012.

[8] C.-H. Yang, J.-Y. Yang, X.-M. Mou, K.-J. Zhou, and W.-H. Gui, "Segmentation method based on clustering pre-segmentation and high-low scale distance reconstruction for colour froth image," *Journal of Electronics and Information Technology*, vol. 30, no. 6, pp. 1286–1290, 2008.

[9] J. Wang, Y. Zhang, and S. Sun, "Multiple T-S fuzzy neural networks soft sensing modeling of flotation process based on fuzzy C-means clustering algorithm," in *Advances in Neural Network Research and Applications*, vol. 67 of *Lecture Notes in Electrical Engineering*, pp. 137–144, Springer, 2010.

[10] C.-H. Yang, H.-F. Ren, C.-H. Xu, and W.-H. Gui, "Soft sensor of key index for flotation process based on sparse multiple kernels least squares support vector machines," *Chinese Journal of Nonferrous Metals*, vol. 21, no. 12, pp. 3149–3154, 2011.

[11] H. Li, T. Chai, and H. Yue, "Soft sensor of technical indices based on KPCA-ELM and application for flotation process," *CIESC Journal*, vol. 63, no. 9, pp. 2892–2898, 2012.

[12] J. S. Wang, S. Han, and N. N. Shen, "Echo state network fusion soft sensing model of flotation process based on glowworm swarm optimization algorithm," *ICIC Express Letters Part B: Applications*, vol. 4, no. 3, pp. 519–524, 2013.

[13] J.-S. Wang, X.-W. Gao, and Y. Zhang, "Research on recognizing flotation states based on image texture features and multi-layer SVMs," *Control and Decision*, vol. 25, no. 10, pp. 1523–1535, 2010.

[14] J. F. Vargas, M. A. Ferrer, C. M. Travieso, and J. B. Alonso, "Off-line signature verification based on grey level information using texture features," *Pattern Recognition*, vol. 44, no. 2, pp. 375–385, 2011.

[15] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, no. 6, pp. 610–621, 1973.

[16] J.-M. Lee, C. Yoo, S. W. Choi, P. A. Vanrolleghem, and I.-B. Lee, "Nonlinear process monitoring using kernel principal component analysis," *Chemical Engineering Science*, vol. 59, no. 1, pp. 223–234, 2004.

[17] J. S. Wang and Q. P. Guo, "Kernel principal component analysis: radial basis function neural networks based soft-sensor modeling of polymerizing process optimized by cultural differential evolution algorithm," *Instrumentation Science & Technology*, vol. 41, no. 1, pp. 18–36, 2013.

[18] H. J. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Advances in Neural Information Processing Systems*, S. Thrun and K. Obermayer, Eds., MIT Press, Cambridge, Mass, USA, 2002.

[19] Y. Liu, J. Zhao, W. Wang, Y.-P. Wu, and W.-C. Chen, "Improved echo state network based on data-driven and its application to prediction of blast furnace gas output," *Acta Automatica Sinica*, vol. 35, no. 6, pp. 731–738, 2009.

[20] X. Lin, Z. Yang, and Y. Song, "Intelligent stock trading system based on improved technical analysis and Echo State Network," *Expert Systems with Applications*, vol. 38, no. 9, pp. 11347–11354, 2011.

[21] K. N. Krishnanand and D. Ghose, "Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions," *Swarm Intelligence*, vol. 3, no. 2, pp. 87–124, 2009.

[22] B. Wu, C. Qian, W. Ni, and S. Fan, "The improvement of glowworm swarm optimization for continuous optimization problems," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6335–6342, 2012.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 906546, 7 pages
http://dx.doi.org/10.1155/2014/906546

Research Article

An Improved Kernel Based Extreme Learning Machine for Robot Execution Failures

Bin Li,[1,2] Xuewen Rong,[1] and Yibin Li[1]

[1] *School of Control Science and Engineering, Shandong University, Jinan, Shandong 250061, China*
[2] *School of Science, Qilu University of Technology, Jinan, Shandong 250353, China*

Correspondence should be addressed to Xuewen Rong; rongxw@sdu.edu.cn

Received 8 February 2014; Accepted 18 March 2014; Published 8 April 2014

Academic Editors: S. Balochian and V. Bhatnagar

Copyright © 2014 Bin 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.

Robot execution failures prediction (classification) in the robot tasks is a difficult learning problem due to partially corrupted or incomplete measurements of data and unsuitable prediction techniques for this prediction problem with little learning samples. Therefore, how to predict the robot execution failures problem with little (incomplete) or erroneous data deserves more attention in the robot field. For improving the prediction accuracy of robot execution failures, this paper proposes a novel KELM learning algorithm using the particle swarm optimization approach to optimize the parameters of kernel functions of neural networks, which is called the AKELM learning algorithm. The simulation results with the robot execution failures datasets show that, by optimizing the kernel parameters, the proposed algorithm has good generalization performance and outperforms KELM and the other approaches in terms of classification accuracy. Other benchmark problems simulation results also show the efficiency and effectiveness of the proposed algorithm.

1. Introduction

The reliability of robot is very important for improving the interactive ability between robot and the changing conditions. In the complex environments in which execution failures can have disastrous consequences for robots and the objects in the surroundings, the prediction ability of robot execution failures is equally important in the robotic field.

However, the prediction of robot execution failures is a difficult learning problem. The first reason is the partially corrupted or incomplete measurements of data. And the second reason is that some prediction techniques are not suitable for predicting the robot execution failures with little samples.

For improving the prediction accuracy of the robot execution failures, in 2009, Twala formulated the robot execution failures problem as a classification task that works with the probabilistic approach-decision tree for handling incomplete data [1]. In 2011, the performance of base-level and meta-level classifiers is compared by Koohi et al. and the Bagged Naïve Bayes is found to perform consistently well across different settings [2]. However, the learning techniques were not incorporated in the aforementioned studies in order to improve the prediction accuracy of robot execution failures. In 2013, Diryag et al. presented a novel method for prediction of robot execution failures based on BP neural networks [3]. The results show that the method can successfully be applied for the robot execution failures with prediction accuracy of 95.4545%. However, it is clear that the learning speed of BP neural networks is generally very slow and may easily converge to local minima. Therefore, some algorithms of machine learning field with better learning performance should be used for the robot execution failures.

The applications of neural networks are very diverse, and, in robotics, many artificial intelligence approaches are applied. Among the approaches of neural networks, extreme learning machine (ELM) proposed by Huang et al. in 2006 has fast learning speed and good generalization performance and has been used in many fields except for the robot execution failures.

The ELM is a learning algorithm for single hidden layer feed-forward neural networks (SLFNs), which determines the initial parameters of input weights and hidden biases randomly with simple kernel function. However, the stability and the generalization performance are influenced by the above learning technique [4]. And some improvements to the ELM learning algorithm have been presented [5].

Among the influence factors of the learning performance of the ELM algorithm, the hidden neurons of the ELM learning algorithm are very important for improving generalization performance and stability of the SLFNs. In [6], we proposed an extreme learning machine with tunable activation function learning algorithm for solving the data dependent on hidden neurons. However, how to choose the suitable combination of activation functions of hidden neurons is still unresolved. In addition, when the feature mapping function of hidden neurons is unknown, kernel function can be used for improving the stability of algorithm [7], which is called the kernel based extreme learning machine (KELM). However, the kernel parameter should be chosen properly for improving the generalization performance of the KELM learning algorithm.

In order to improve the classification accuracy (generalization performance) of robot execution failures, we propose a novel kernel based extreme learning machine in this paper. The kernel parameters of kernel functions of the proposed algorithm are optimized based on the particle swarm optimization approach, which can improve the generalization performance with stable learning process of the proposed algorithm. The simulation results in terms of robot execution failures and some other benchmark problems show the efficiency and effectiveness of the proposed algorithm and are suitable for the robot execution failures problem of little (incomplete) or erroneous data.

The remainder of this paper is organized as follows. The kernel based extreme learning machine (KELM) is introduced in Section 2. Section 3 describes the particle swarm optimization for KELM learning algorithm. Then, the performance analysis of the proposed algorithm and simulation results of robot execution failures are analyzed in Section 4. Section 5 gives the performance analysis of the algorithms by two regression and two classification problems. The last section is the conclusions of this paper.

2. Kernel Based Extreme Learning Machine

Recently, the ELM learning algorithm with fast learning speed and good generalization performance has been attracting much attention from an increasing number of researchers [4, 7]. In ELM, the initial parameters of hidden layer need not be tuned and almost all nonlinear piecewise continuous functions can be used as the hidden neurons. Therefore, for N arbitrary distinct samples $\{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, i = 1, \dots, N\}$, the output function in ELM with L hidden neurons is

$$f_L(x) = \sum_{i=1}^{L} \beta_i h_i(x) = h(x)\beta, \tag{1}$$

where $\beta = [\beta_1, \beta_2, \dots, \beta_L]$ is the vector of the output weights between the hidden layer of L neurons and the output neuron and $h(x) = [h_1(x), h_2(x), \dots, h_L(x)]$ is the output vector of the hidden layer with respect to the input x, which maps the data from input space to the ELM feature space [7].

For decreasing the training error and improving the generalization performance of neural networks, the training error and the output weights should be minimized at the same time, that is,

$$\text{Minimize: } \|H\beta - T\|, \|\beta\|. \tag{2}$$

The least squares solution of (2) based on KKT conditions can be written as

$$\beta = H^T\left(\frac{1}{C} + HH^T\right)^{-1} T, \tag{3}$$

where H is the hidden layer output matrix, C is the regulation coefficient, and T is the expected output matrix of samples.

Then, the output function of the ELM learning algorithm is

$$f(x) = h(x) H^T\left(\frac{1}{C} + HH^T\right)^{-1} T. \tag{4}$$

If the feature mapping $h(x)$ is unknown and the kernel matrix of ELM based on Mercer's conditions can be defined as follows:

$$M = HH^T : m_{ij} = h(x_i) h(x_j) = k(x_i, x_j), \tag{5}$$

thus, the output function $f(x)$ of the kernel based extreme learning machine (KELM) can be written compactly as

$$f(x) = [k(x, x_1), \dots, k(x, x_N)] \left(\frac{1}{C} + M\right)^{-1} T, \tag{6}$$

where $M = HH^T$ and $k(x, y)$ is the kernel function of hidden neurons of single hidden layer feed-forward neural networks.

There are many kernel functions satisfying the Mercer condition available from the existing literature, such as linear kernel, polynomial kernel, Gaussian kernel, and exponential kernel. In this paper, we use three typical kernel functions for simulation and performance analysis and the chosen kernel functions are as follows.

(1) Gaussian kernel:

$$k(x, y) = \exp(-a\|x - y\|); \tag{7}$$

(2) hyperbolic tangent (sigmoid) kernel:

$$k(x, y) = \tanh\left(bx^T y + c\right); \tag{8}$$

(3) wavelet kernel:

$$k(x, y) = \cos\left(d\frac{\|x - y\|}{e}\right)\exp\left(-\frac{\|x - y\|^2}{f}\right), \tag{9}$$

where Gaussian kernel function is a typical local kernel function and tangent kernel function is a typical global nuclear function, respectively [8]. Furthermore, the complex wavelet kernel function is also used for testing the performance of algorithms.

In the above three kernel functions, the adjustable parameters a, b, c, e, and f play a major role in the performance of neural networks and should be tuned carefully based on the solved problem.

Compared with the ELM learning algorithm, the hidden layer feature mapping need not be known and the number of hidden neurons need not be chosen in the KELM. Moreover, the KELM learning algorithm achieves similar or better generalization performance and is more stable compared to traditional ELM and it is faster than support vector machine (SVM) [7, 9].

3. Particle Swarm Optimization for KELM

In KELM learning algorithm, the regulation coefficient C and kernel parameters should be chosen appropriately for improving the generalization performance of neural networks. In [7], the parameters are tried in a wide range and are time consuming. And in [10], a hybrid kernel function is proposed for improving the generalization performance of KELM. However, how to choose the optimal value of the parameters of kernel function has not been resolved.

In this paper, an optimization approach is introduced to the KELM for choosing the optimal parameters of kernel function. There are many optimization approaches in machine learning field and, compared with other methods, the particle swarm optimization (PSO) is a biologically inspired computational stochastic optimization technique developed by Eberhart and Kennedy [11]. The PSO approach is becoming popular because of its little memory requiring, simplicity of implementation, and ability to converge to a reasonably optimal solution quickly [12].

Similar to other population based optimization approaches, the PSO algorithm works by initialing the population of individuals randomly in the search space. Each particle of PSO can fly around to find the best solution in the search space; meanwhile, the particles all look at the best solution (best particle) in their path.

Suppose that the dimension of search space of PSO is D and the population size is $\widehat{N}$. Then, x_i^d and v_i^k are denoted by the current position and the current velocity of ith particle at iteration t, respectively. Then, the new velocity and position of the particles for the next iteration time are calculated as follows:

$$\begin{aligned} v_i^k(t+1) = {} & w \cdot v_i^k(t) + c_1 \cdot \text{rand}()\left(p_i^k(t) - x_i^k(t)\right) \\ & + c_2 \cdot \text{rand}()\left(g_i^k(t) - x_i^k(t)\right), \end{aligned} \tag{10}$$

$$\begin{gathered} x_i^k(t+1) = x_i^k(t) + v_i^k(t+1), \\ 1 \le i \le \widehat{N}, \quad 1 \le k \le D, \end{gathered} \tag{11}$$

where p_i^k denotes the best position of the ith particle during the search process until now, g_i^k represents the global best position, which constitutes the best position found by the entire swarm until now, w is the inertia weight, $c_1$and c_2 are the acceleration constants, and rand() is a random number between 0 and 1.

In PSO algorithm, the inertia weight w maintains the expansion ability of exploring new areas in the search space. Therefore, in order to ensure higher exploring ability in the early iteration and fast convergence speed in the last part iteration, the parameter w can reduce gradually at the generation increases and is calculated as [13]

$$w(t) = w_{\max} - \text{iter} \times \frac{(w_{\max} - w_{\min})}{\text{max iter}}, \tag{12}$$

where $w_{\max}$ and $w_{\min}$ are the initial inertial weight and the final inertial weight, respectively. The parameter max iter is the maximum searching iteration number and iter is the iteration number at the present, respectively.

In addition, in order to enhance the global search in the early part iteration, to encourage the particles to converge to the global optimal solution, and to improve the convergence speed in the final iteration period [12, 14], the acceleration parameters $c_1$and c_2 are described as

$$c_1 = (c_{1\,\min} - c_{1\,\max})\frac{\text{iter}}{\text{max iter}} + c_{1\,\max}, \tag{13}$$

$$c_2 = (c_{2\,\max} - c_{2\,\min})\frac{\text{iter}}{\text{max iter}} + c_{2\,\min}, \tag{14}$$

where $c_{1\,\max}$ and $c_{1\,\min}$ are the initial acceleration constant and the final acceleration constant of c_1 and $c_{2\,\min}$ and $c_{2\,\max}$ are the initial acceleration constant and the final acceleration constant of c_2, respectively. Therefore, by changing the acceleration coefficients with time, the cognitive component is reduced and the social component is increased in (10), respectively.

Based on the optimization technology of PSO with self-adaptive parameters w and c, the parameters of kernel functions of KELM are optimized for improving the generalization performance. Since the number of parameters of kernel functions is different, the dimension of the particle of the proposed algorithm in this paper also changes with the different kernel functions. Therefore, the particle (individual) θ of search space in the proposed algorithm can be defined as

$$\begin{gathered} \theta \in [a] \ \text{ for Gaussian kernel}, \\ \theta \in [b, c] \ \text{ for tangent kernel}, \\ \theta \in [d, e, f] \ \text{ for wavelet kernel, respectively}. \end{gathered} \tag{15}$$

Thus, the kernel parameter optimization strategy of KELM based on the PSO (which is called the AKELM (adaptive kernel based extreme learning machine) learning algorithm) is summarized as follows.

Given the type of the kernel function, the training set $\{(x_i, t_i) \mid x_i \in R^n,\ t_i \in R^m,\ i = 1, \ldots, N\}$, and the initial value of regulation coefficient C, consider the following steps.

Step 1. Initiate the population (particle) based on the kernel function and the velocity and position of each particle.

Step 2. Evaluate the fitness function of each particle (the root means standard error for regression problems and the classification accuracy for classification problems).

Table 1: Feature information and class distribution of the robot execution failures.

Datasets	Instances	Classes
LP1	88	4 (1 = 24%; 2 = 19%; 3 = 18%; 4 = 39%)
LP2	47	5 (1 = 43%; 2 = 13%; 3 = 15%; 4 = 11%; 5 = 19%)
LP3	47	4 (1 = 43%; 2 = 19%; 3 = 32%; 4 = 6%)
LP4	117	3 (1 = 21%; 2 = 62%; 3 = 18%)
LP5	164	5 (1 = 27%; 2 = 16%; 3 = 13%; 4 = 29%; 5 = 16%)

Step 3. According to formulas (10)–(14), the velocity and position of the particle are modified.

Step 4. Step 2 and Step 3 are iterated repetitively until the maximal iteration time is satisfied.

Step 5. The optimal parameters of kernel function can be determined. Then, based on the optimized parameters, the hidden layer kernel matrix is computed.

Step 6. Determine the final output weights β in terms of the following equation: $\beta = H^T((1/C) + HH^T)^{-1}T$.

4. Robot Execution Failures Based on AKELM

In this paper, the AKELM learning algorithm and the KELM algorithm for robot execution failures prediction and the other benchmark problems in machine learning field are conducted in the MATLAB 7.0 with 3.2 GHz CPU and 2G RAM. The number of populations of the PSO for optimizing the kernel parameters is 200 and the maximum iteration number is 100. The initial inertial weights $w_{\max}$ and $w_{\min}$ are 0.9 and 0.4, respectively. And the initial acceleration constant values $c_{\max}$ and $c_{\min}$ are 2.5 and 0.5, respectively, which means that c_1 changes from 2.5 to 0.5 and c_2 changes from 0.5 to 2.5 over the full range of the search. The regulation coefficient C is 100 and the kernel parameters of the KELM learning algorithm are set to 1.

4.1. Data Processing. The original robot execution failures data has 90 features, which includes the evolution of forces Fx (15 samples; the following is the same), Fy, and Fz and the evolution of torques Tx, Ty, and Tz measurements on a robot after failure of detection [15].

The robot execution failures problem includes 5 datasets, each of them defining a different learning problem:

(i) LP1: failures in approach to grasp position,

(ii) LP2: failures in transfer of a part,

(iii) LP3: position of part after a transfer failure,

(iv) LP4: failures in approach to ungrasp position,

(v) LP5: failures in motion with part.

The feature information and class distribution of the robot execution failures datasets is denoted in Table 1.

As shown from Table 1, the dataset of robot execution failure has small size with 90 features and many classes with 4 for LP1, 5 for LP2, 4 for LP3, 3 for LP4, and 5 for LP5, respectively, which increases the classification difficulty of algorithms.

In [16], a set of five feature transformation strategies was defined for improving the classification accuracy. In the learning of the AKELM and KELM algorithms in neural networks, in order to ensure that different units of data have the same influence on the algorithm, the original data need to be normalized. In this paper, the data is normalized to the interval $[-1, +1]$ and can be described by the following equation:

$$x = 2\frac{x - x_{\min}}{x_{\max} - x_{\min}} - 1, \tag{16}$$

where $x_{\max}$ and $x_{\min}$ represent the maximum and minimum values in the original datasets, x on the left of the above equation is the original data, and x on the right of the above equation is the normalized output data.

For improving the generalization of the robot execution failures data, the positions of samples in each dataset are changed randomly. Then, 90% of samples of the dataset are used for training the neural networks, and the other 10% are testing samples.

4.2. Simulation and Performance Analysis. In this study, the performance of the proposed AKELM learning algorithm is compared with the KELM using the robot execution failures data. In the KELM learning algorithm, the learning ability and the generalization performance are influenced mainly by the kernel parameters of different kernel functions. In this paper, the Gaussian kernel function, tangent kernel function, and wavelet kernel function are used to construct different classifier for predicting the robot execution failures.

Firstly, in order to reduce the search space and accelerate the convergence speed of the PSO algorithm, this paper gives the relationship between the classification accuracy and the number of some parameters of kernel function on robot execution failures using the LP1 dataset. As shown in Figure 1, the classification accuracy in the interval $(0, 4]$ has good performance with the difference of the parameters 1 (the values are a, b, and d for Gaussian kernel, tangent kernel, and wavelet kernel, resp.), the parameters 2 (the values are c and e for tangent kernel and wavelet kernel, resp.), and the parameters 3 (the value is f for wavelet kernel). Therefore, the search space of the PSO algorithm is set in the interval between 0 and 4.

Since the simulation results are the same for different running times of the AKELM algorithm and the KELM algorithm, Table 2 shows the comparison of classification

Table 2: Classification accuracy of robot execution failures based on KELM and AKELM algorithms.

Kernel data	Gaussian		Tangent		Wavelet	
	KELM	AKELM	KELM	AKELM	KELM	AKELM
LP1	**100%**	**100%**	62.50%	**87.50%**	**100%**	**100%**
LP2	57.14%	57.14%	57.14%	**85.71%**	57.14%	**85.71%**
LP3	57.14%	**71.43%**	57.14%	**85.71%**	57.14%	**100%**
LP4	75%	**100%**	75%	**83.33%**	83.33%	**100%**
LP5	57.14%	**64.29%**	0%	**78.57%**	50%	**71.43%**

Table 3: Specification of benchmarks of regression and classification problems.

Datasets	Names	Attributes	Classes	Training data	Testing data
Regression	Box and Jenkins gas furnace data	10	1	200	90
	Auto-Mpg	7	1	320	78
Classification	Wine	13	3	150	28
	Diabetes	8	2	576	192

Table 4: Comparison of performance by AKELM and KELM learning algorithms for the regression problems.

Algorithms with different kernel functions	Box and Jenkins gas furnace data			Auto-Mpg		
	Training error	Testing error	Training time (seconds)	Training error	Testing error	Training time (seconds)
KELM (parameters = 1, Gaussian)	0.0120	0.0188	0.0394	0.0529	0.0599	0.1213
KELM (parameters = 1, tangent)	0.0627	0.0655	0.0116	0.6680	0.7756	0.0346
KELM (parameters = 1, wavelet)	0.0121	0.0206	0.0177	0.0509	**0.0597**	0.0415
KELM (parameters = 10, Gaussian)	0.0183	0.0213	0.0149	0.0685	0.0732	0.0286
KELM (parameters = 10, tangent)	0.2245	0.1986	0.0044	0.2071	0.2085	0.0261
KELM (parameters = 10, wavelet)	0.0306	0.0382	0.0101	0.0662	0.0712	0.0360
AKELM (Gaussian)	0.0133	**0.0183**	26.1250	0.0503	**0.0597**	74.7656
AKELM (tangent)	0.0223	**0.0242**	25.2500	0.0735	**0.0735**	73.8906
AKELM (wavelet)	0.0133	**0.0183**	28.3906	0.0502	**0.0597**	84.9688

results of robot execution failures datasets with three different kernel functions in one running time. As can be seen from the table, the proposed AKELM learning algorithm shows better classification accuracy than the KELM with different kernel functions in most cases and the best classification accuracies are given in boldface. Especially in the LP1 dataset, the proposed algorithm has 100% classification accuracy with Gaussian and wavelet kernel functions, and the generalization performance is better than the best classification approach, Bagged Naïve Bayes in [2], until now to the authors' best knowledge.

5. Performance Analysis of AKELM Using Other Benchmark Problems

In this section, the performance of AKELM learning algorithm is compared with the KELM in terms of two regression benchmarks and two classification benchmarks. Specification of the benchmark problems is shown in Table 3. The performance of classification benchmark problems is measured by the classification accuracy and the root mean squares error is used to measure the error of the regression benchmark problems.

Tables 4 and 5 show the performance comparison of AKELM and KELM with Gaussian kernel, tangent kernel, and wavelet kernel neurons; apparently, better test results are given in boldface. The parameters = 1 and parameters = 10 represent the total kernel parameters of different kernel functions set to 1 and 10, respectively.

It can be seen that the proposed AKELM algorithm can always achieve similar or better generalization performance than KELM with different kernel functions and kernel parameters. Moreover, seen from Tables 4 and 5, the KELM learning algorithm with different kernel functions has obviously different generalization performance. However, the proposed AKELM learning algorithm has similar generalization performance to different kernel functions, which means that the proposed algorithm has stable performance with kernel parameters optimized by means of the PSO algorithm, although searching the optimal parameters needs some time as the training time shown in Tables 4 and 5.

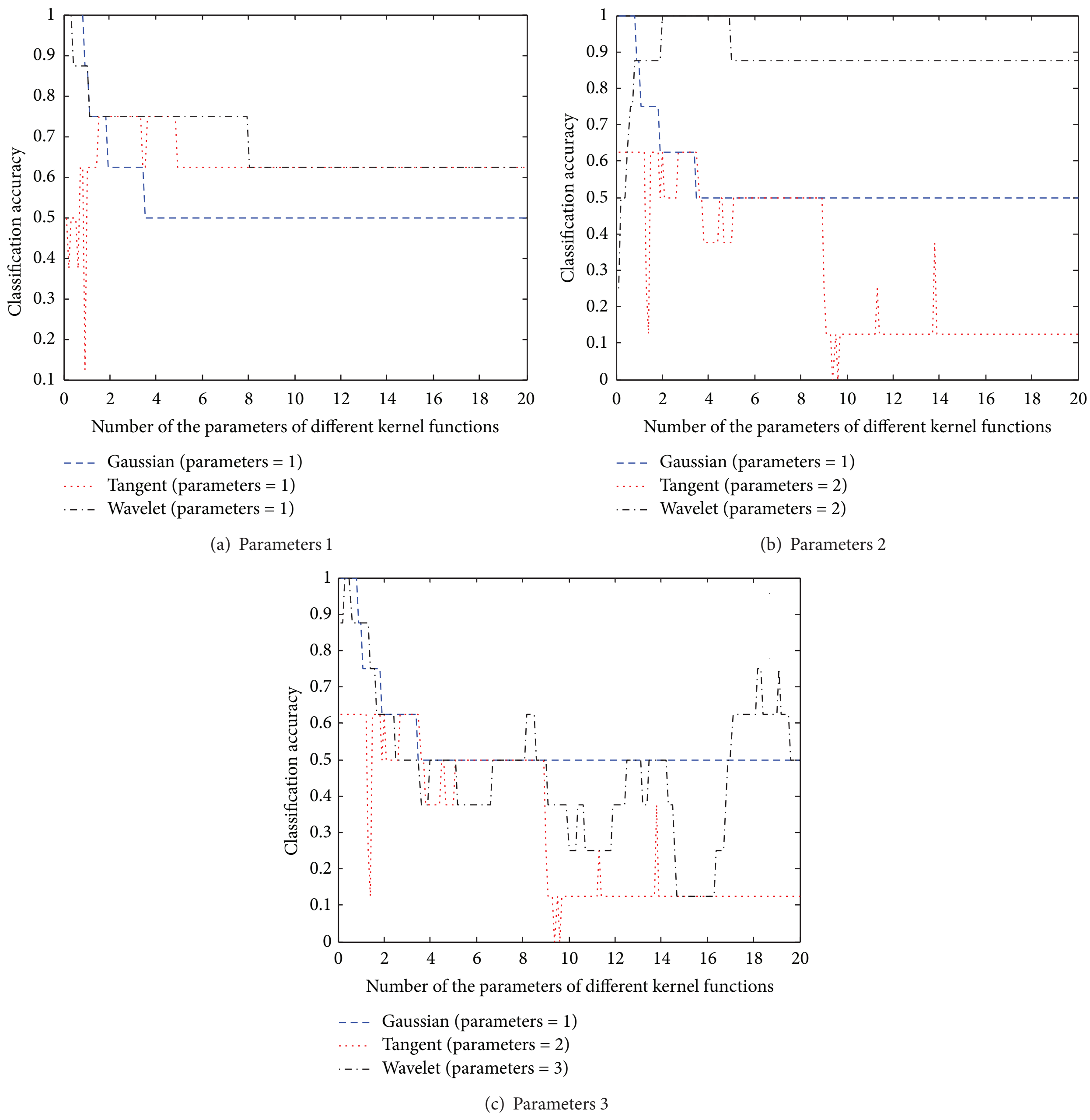


FIGURE 1: Relationship between the classification accuracy and the number of some parameters of kernel function on LP1 dataset.

6. Conclusions

In this study, a novel learning algorithm AKELM has been developed based on the KELM learning algorithm and the PSO approach with self-adaptive parameters. In the proposed AKELM learning algorithm, the parameters of kernel functions of neural networks are adjusted for searching the optimal values by the PSO algorithm.

As shown from the simulation results, the generalization performance of the proposed algorithm in terms of the robot execution failures datasets was found to be significantly improved compared to the KELM learning algorithm. And the other benchmark of regression and classification problems also shows that the proposed algorithm can achieve better generalization performance and has more stable ability than KELM algorithm.

Conflict of Interests

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

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (61233014 and 61305130), the Shandong Provincial Natural Science Foundation (ZR2013FQ003

Table 5: Comparison of performance by AKELM and KELM learning algorithms for the classification problems.

Algorithms with different kernel functions	Wine			Diabetes		
	Training accuracy	Testing accuracy	Training time (seconds)	Training accuracy	Testing accuracy	Training time (seconds)
KELM (parameters = 1, Gaussian)	100%	**100%**	0.0277	84.38%	77.08%	0.1394
KELM (parameters = 1, tangent)	51.33%	50%	0.0067	73.78%	73.44%	0.1326
KELM (parameters = 1, wavelet)	100%	**100%**	0.0070	86.81%	76.56%	0.1347
KELM (parameters = 10, Gaussian)	100%	**100%**	0.0083	78.99%	79.17%	0.0919
KELM (parameters = 10, tangent)	39.33%	42.86%	0.0023	65.80%	65.63%	0.0904
KELM (parameters = 10, wavelet)	100%	96.43%	0.0061	80.03%	77.08%	0.1361
AKELM (Gaussian)	100%	**100%**	17.8594	90.45%	**80.21%**	260.7031
AKELM (tangent)	97.33%	**100%**	13.9375	73.26%	**79.17%**	313.8750
AKELM (wavelet)	100%	**100%**	16	89.06%	**79.69%**	335.5469

and ZR2013EEM027), and China Postdoctoral Science Foundation (2013M541912).

References

[1] B. Twala, "Robot execution failure prediction using incomplete data," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pp. 1518–1523, Guilin, China, 2009.

[2] T. Koohi, E. Mirzaie, and G. Tadaion, "Failure prediction using robot execution data," in *Proceedings of the 5th Symposium on Advances in Science and Technology*, pp. 1–7, Mashhad, Iran, 2011.

[3] A. Diryag, M. Mitic, and Z. Miljkovic, "Neural networks for prediction of robot failures," *Journal of Mechanical Engineering Science*, 2013.

[4] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[5] R. Rajesh and J. Siva Prakash, "Extreme learning machines—a review and state-of-the-art," *International Journal of Wisdom Based Computing*, vol. 1, pp. 35–49, 2011.

[6] B. Li, Y. Li, and X. Rong, "The extreme learning machine learning algorithm with tunable activation function," *Neural Computing and Applications*, vol. 22, pp. 531–539, 2013.

[7] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 42, no. 2, pp. 513–529, 2012.

[8] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine Learning*, vol. 42, no. 1-2, pp. 143–175, 2001.

[9] W. Huang, N. Li, Z. Lin et al., "Liver tumor detection and segmentation using kernel-based extreme learning machine," in *Proceedings of the 35th Annual International Conference of the IEEE EMBS*, pp. 3662–3665, Osaka, Japan, 2013.

[10] S. F. Ding, Y. A. Zhang, X. Z. Xu, and L. N. Bao, "A novel extreme learning machine based on hybrid kernel function," *Journal of Computers*, vol. 8, no. 8, pp. 2110–2117, 2013.

[11] R. C. 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.

[12] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[13] F. Han, H. Yao, and Q. Ling, "An improved extreme learning machine based on particle swarm optimization," *Neurocomputing*, vol. 116, pp. 87–93, 2013.

[14] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[15] P. M. Murphy and D. W. Aha, "UCI Repository of Machine Learning Databases [EB/OL]," http://archive.ics.uci.edu/ml/datasets.html.

[16] L. Seabra Lopes and L. M. Camarinha-Matos, "Feature transformation strategies for a robot learning problem," *Feature Extraction, Construction and Selection*, vol. 453, pp. 375–391, 1998.