Journal of Applied Mathematics

Harmony Search and Nature-Inspired Algorithms for Engineering Optimization

Guest Editors: Zong Woo Geem, Xin-She Yang, and Chung-Li Tseng

Harmony Search and Nature-Inspired Algorithms for Engineering Optimization

Journal of Applied Mathematics

Harmony Search and Nature-Inspired Algorithms for Engineering Optimization

Guest Editors: Zong Woo Geem, Xin-She Yang, and Chung-Li Tseng

Editorial Board

Contents

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 438158, 2 pages
http://dx.doi.org/10.1155/2013/438158

Editorial

Harmony Search and Nature-Inspired Algorithms for Engineering Optimization

Zong Woo Geem,[1] Xin-She Yang,[2] and Chung-Li Tseng[3]

[1] *Department of Energy and Information Technology, Gachon University, Seongnam 461-701, Republic of Korea*
[2] *Department of Design Engineering and Mathematics, Middlesex University, London NW4 4BT, UK*
[3] *The Australian School of Business, The University of New South Wales, Sydney, NSW 2052, Australia*

Correspondence should be addressed to Zong Woo Geem; geem@gachon.ac.kr

Received 7 November 2013; Accepted 7 November 2013

Since the emergence of swarm intelligence in the 1990s, especially the appearance of ant colony optimization and particle swarm optimization, nature-inspired algorithms started to mushroom [1–7]. In the last two decades, new optimization algorithms have become popular, and the algorithms such as harmony search [1, 2] and firefly algorithms [5] have shown to be superior and efficient. Significant progress has been made in a wide range of nature-inspired algorithms and their applications. Metaheuristics and swarm intelligence are becoming more popular for design optimization [3–5].

Most of these algorithms belong to evolutionary computation in general, and they have been developed and inspired from natural phenomenon. However, not all algorithms were inspired by nature. For example, harmony search, developed by Geem et al. in 2001, was inspired by the improvisation characteristics of a musician, and therefore, harmony search is a music-inspired algorithm [1, 2]. However, a vast majority of algorithms have been developed by mimicking the characteristics of biological systems in nature. For example, the firefly algorithm was developed by Yang in 2009 [5] and was inspired by the flashing patterns of tropical fireflies, while the cuckoo search was developed by Yang and Deb in 2010, inspired by the brooding parasitism of some cuckoo species [6]. The diversity of these algorithms and their applications has permeated into almost every area of engineering and industry [3–5].

The aim of this special issue is to review the latest developments in nature-inspired algorithms and their applications. The call for papers was well received, leading to a high number of submissions. After a rigorous peer-review process, sixteen high-quality papers have been selected. These papers represent a snapshot of the relevant research progress in these areas. Ten of the papers published in this special issue are dedicated to harmony search (HS) and the others to other types of nature-inspired algorithms. M. A. Al-Betar et al. present a new variant of HS, called cellular harmony search, to solve a set of 25 test functions with promising results. S. S. Im et al. also propose using cellular harmony search for unconstrained optimization. D. Oliva et al. use harmony search to carry out multilevel thresholding segmentation in image processing. Z. W. Geem demonstrates a parameter-setting-free version of harmony search and uses it to solve the economic dispatch of power plant operations. J. Fourie et al. carry out a comparative analysis of modern harmony search, providing new insights into the working mechanism of the method. R. Mallipeddi combines harmony search with differential evolution to enhance differential evolution by using parameter ensemble adaptation. In another paper, H.-Y. Yun et al. combine harmony search and ant colony optimization for traveling salesman problem. Furthermore, S. Lee et al. determine pavement rehabilitation activities by a permutation algorithm within the framework of harmony search. In the other two applications using harmony search, S. Lee et al. solve electroencephalography signal grouping, and D.-S. An et al. estimate parameters for traffic noise.

For those papers on nature-inspired algorithms, A. Gálvez and A. Iglesias use a firefly algorithm-based approach to do polynomial Bézier surface parameterization for

complex surfaces. O. Baskan uses the cuckoo search to determine optimal link capacity expansions in road networks, while H. Zheng and Y. Zhou develop a variant of cuckoo search, called a cooperative coevolutionary cuckoo search algorithm. They have solved 28 test problems and achieved better results. In addition, Ma et al. use the cuckoo search to estimate key parameters in photovoltaic models and obtain high accuracy in terms of a low root-mean-squared-error value. For practical applications, J. Xie et al. use the bat algorithm to solve multiple runways aircraft landing problem, such as up to 500 aircrafts landing safely in a short time. S. Fong et al. use swarm search to select optimal feature set in high-dimensional data.

The papers published in this special issue give a glimpse of the current development and the active research in harmony search, firefly algorithm, cuckoo search, and bat algorithms as well as other algorithms. This special issue also reports new real-world applications, indicating that the applicability of these algorithms continues to expand. Obviously, this special issue can only represent a fraction of current research activities among the more diverse studies and applications.

Despite the success of nature-inspired algorithms, in our view, there are at least three challenges that require special attention of the researchers: large-scale problems, combinatorial optimization, and proper constraint handling.

Currently, most studies in the literature concern problems with a small or moderate number of design variables. It is rare to see a study dealing with hundreds of design variables. Recently, large-scale problems have started to be tackled by researchers (e.g., in [3, 4]). However, more research activities and efforts are needed as real-world applications can become extremely complex with potentially millions of design variables.

The vast majority of the applications of evolutionary algorithms have focused on continuous optimization problems. There is another important class of challenging problems called combinatorial optimization, such as the well-known traveling salesman problem. Since combinatorial optimization problems can be NP-hard, nature-inspired algorithms are good candidates for tackling this type of problems. While there have been researchers attempting to address this issue, such as water distribution networks (one of which has 1E454 candidate solutions) and the traveling salesman problem [3, 4], for truly hard problems and large-scale ones, more systematic efforts and studies are demanded to provide better results in a more efficient way.

Another important issue is constraint handling, from the implementation point of view. Even if an efficient algorithm is correctly implemented, its performance can be hampered if the constraints of the problem are not properly handled. While some may view constraint handling as a separate topic, it is part of the integrated optimization process. If the solution of a real application generated by algorithms does not satisfy the constraints, the solution is infeasible and cannot be implemented. This highlights the importance of constraint-handling techniques in combination with optimization algorithms.

In this special issue, the selected papers only address a small subset of the vast number of optimization problems. We hope this special issue will encourage and inspire more needed research activities in the near future.

Acknowledgment

We would like to thank those who participated in this special issue, including the contributing authors and reviewers.

Zong Woo Geem
Xin-She Yang
Chung-Li Tseng

References

[1] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[2] Z. W. Geem, *Music-Inspired Harmony Search Algorithm*, Springer, Heidelberg, Germany, 2009.

[3] Z. W. Geem, "Optimal cost design of water distribution networks using harmony search," *Engineering Optimization*, vol. 38, no. 3, pp. 259–280, 2006.

[4] A. Ouaarab, B. Ahiod, and X. S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Computing and Applications*, 2013.

[5] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, vol. 5792 of *Lecture Notes in Computer Science*, pp. 169–178, Springer, Berlin, Germany, 2009.

[6] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[7] X. S. Yang, M. Karamanoglu, and X. S. He, "Flower pollination algorithm: a novel approach for multiobjective optimization," *Engineering Optimization*, 2013.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 635608, 9 pages
http://dx.doi.org/10.1155/2013/635608

Research Article

Smallest-Small-World Cellular Harmony Search for Optimization of Unconstrained Benchmark Problems

Sung Soo Im,[1] Do Guen Yoo,[2] and Joong Hoon Kim[3]

[1] *Daelim Industrial Co. Ltd., Seoul 110-140, Republic of Korea*
[2] *Research Institute of Engineering and Technology, Korea University, Seoul 136-713, Republic of Korea*
[3] *School of Civil, Environmental and Architectural Engineering, Korea University, Seoul 136-713, Republic of Korea*

Correspondence should be addressed to Joong Hoon Kim; jaykim@korea.ac.kr

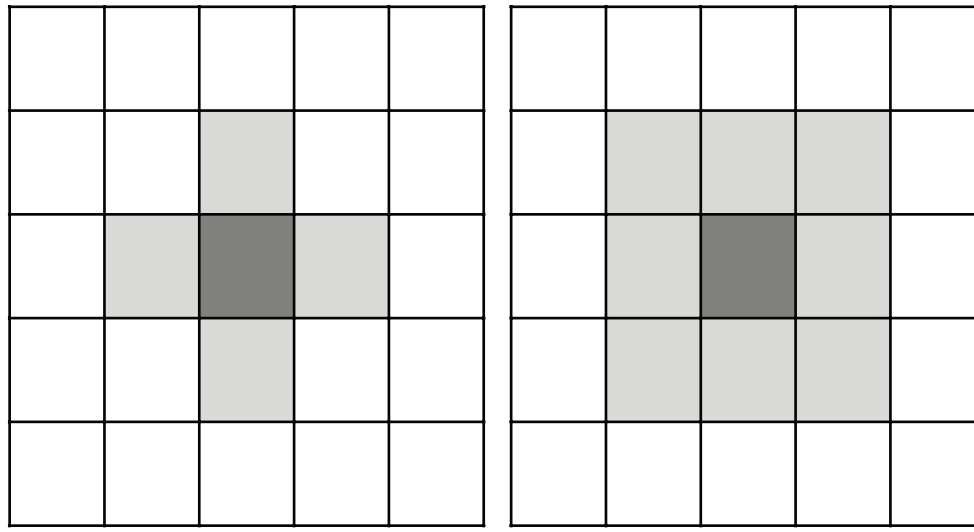
Received 28 June 2013; Revised 28 September 2013; Accepted 8 October 2013

Academic Editor: Zong Woo Geem

We presented a new hybrid method that combines cellular harmony search algorithms with the Smallest-Small-World theory. A harmony search (HS) algorithm is based on musical performance processes that occur when a musician searches for a better state of harmony. Harmony search has successfully been applied to a wide variety of practical optimization problems. Most of the previous researches have sought to improve the performance of the HS algorithm by changing the pitch adjusting rate and harmony memory considering rate. However, there has been a lack of studies to improve the performance of the algorithm by the formation of population structures. Therefore, we proposed an improved HS algorithm that uses the cellular automata formation and the topological structure of Smallest-Small-World network. The improved HS algorithm has a high clustering coefficient and a short characteristic path length, having good exploration and exploitation efficiencies. Nine benchmark functions were applied to evaluate the performance of the proposed algorithm. Unlike the existing improved HS algorithm, the proposed algorithm is expected to have improved algorithmic efficiency from the formation of the population structure.

1. Introduction

Studies on network maps provide in-depth understanding of the basic features and requirements of various systems. Many network connection topologies, assumed to be either completely regular or completely random, have been studied [1, 2]. Networks, which can be formally described by the tools of graph theory, are critical for describing many scientific, social, and technological phenomena (Newman [1]). Typical examples include the Internet, World Wide Web, social acquaintances, electric power networks, and neural networks. In recent years, new theoretical and applied results have motivated substantial researches in network science. The pioneering studies of Watts and Strogatz [2] have been performed, and they have been followed by many others in the subsequent years. They constructed a simple computer model of a regular network or lattice, in which each node of the network was connected by a line or edge to each of its four nearest neighbors. This network structure or topology is highly clustered or cliquish by design; however, movement from one node to another node on the opposite side of the lattice requires traversing a large number of short-range connections. In other words, although the path length (or number of mediating edges) between neighboring nodes is short, the path length between distant nodes is long. Therefore, the minimum path length averaged over all possible pairs of nodes in the network which is quite long. Watts and Strogatz [2] investigated the change in network topology (measured in terms of local clustering and minimum path length) that resulted from randomly rewiring some of the lattice edges to create long-range connections between distant nodes. If many lattice edges were randomly rewired, the network would naturally acquire the topological characteristics of a random graph (short path length and low clustering). It is worth mentioning that they found that even a few long-range connections greatly reduced the minimum path length of the network without affecting its local clustering. Thus, they defined algorithmically for the first time a class of networks having topological properties similar to social networks and demonstrated both the high



(a) Neumann (b) Moore

Figure 1: Neighboring cells.

clustering of a lattice and the short path length of a random graph. These networks were called Small-World networks. On the other hand, Nishikawa et al. [3] suggested a Smallest-Small-World network (SSWN) and confirmed their theory with examples. They showed that the average path length of a Small-World network with fixed shortcuts became the minimum when there was a "center" node, from which all shortcuts are connected to uniformly distributed nodes in the network [4]. Research on efficient network structures can be applied to fields of optimization, Genetic Algorithms (GAs), Artificial Neural Networks (ANNs), and Particle Swarm Optimization (PSO). In GAs, cellular Genetic Algorithms (cGAs) have been widely studied using cellular automata (CA). Kang [5] particularly proposed Smallest-Small-World cellular Genetic Algorithms (SSWcGAs) to apply the SSWN to the cGAs and evaluated the performance of the algorithm.

The harmony search (HS) algorithm [6, 7] is based on musical performance processes that occur when a musician searches for a better state of harmony. The HS has successfully been applied to many mathematical functions and a wide variety of practical optimization problems like pipe-network design, structural optimization, vehicle routing, combined heat and power economic dispatching, multiple-dam-system scheduling, and so forth. In addition, hybrid HS algorithms that combine with other optimization techniques such as the PSO and ANNs have been proposed.

So far, researches have been carried out to improve the performance of the HS algorithm by changing the parameters such as pitch adjusting rate (PAR) and harmony memory considering rate (HMCR). However, there has been a lack of studies on the performance improvement of the HS through the formation of population structures to transform it into a cGA. In this study, therefore, we proposed the improved HS algorithm, which is used the CA and has the topological structure of the SSWN. The improved HS algorithm has a high clustering coefficient and a short characteristic path length possessing good exploration and exploitation efficiencies.

2. Related Theory

2.1. Cellular Automata. A cellular automaton is a discrete model studied in computation theory, mathematics, physics, complexity science, theoretical biology, and microstructure modeling [8]. Von Neumann [9] has used the concept of cellular automaton space regularly arranged in a grid cell. Individual cells updated simultaneously in a discrete time step. Each cell is a finite state machine. Each cell entered the state of its own and neighboring and, then, printout the state in the next time step. A collection of these cellular automatons is called CA space. Other terms for the CA are "cellular space," "tessellation automata," "homogeneous structures," "cellular structures," "tessellation structures," "iterative arrays," and so on [8].

The cell's dimension in the CA can be defined as one-, two-, and three-dimensional cells. Typically, a 2D cell is the most widely used because it can deal with spatial phenomena. Talking about the shape of the cell, a square cell is usually used because of its highest computer-processing efficiency, compared to the other types of cells. Neighbors in the CA can be defined depending on the form of the adjacent surrounding cells. In the form of a 2D square cell CA, it can be defined as a Von Neumann neighborhood with neighbors in four directions or as a Moore neighborhood having neighbors in eight directions as shown in Figure 1.

The concept of the CA, as described in this subsection, has been applied to a variety of optimization techniques such as GAs [10, 11], ANNs [12], and PSO [13, 14]. For instance, cGA is a subclass of GA in which each individual is placed in a given topological distribution. It is a lattice graph, as one individual can only interact with its nearest

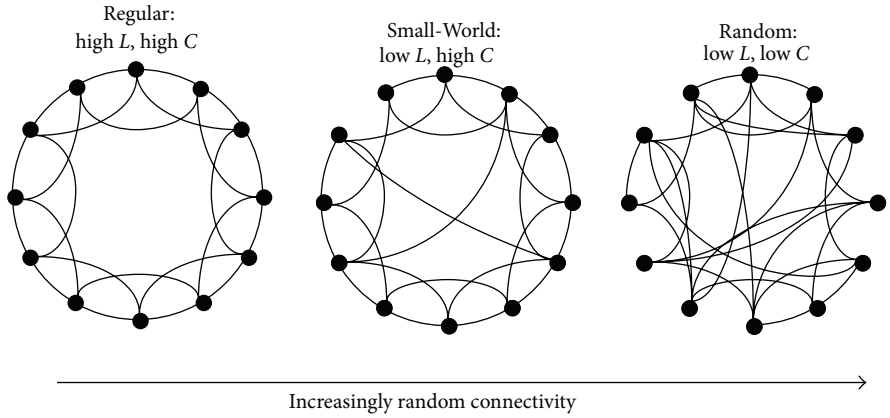


Figure 2: Classification of networks according to connectivity.

Table 1: Efficiency of average path length (L) and clustering coefficient (C).

Property	Description
Average path length (L)	Exploitation (local search) Low $L \Rightarrow$ good exploitation efficiency
Clustering coefficient (C)	Exploration (Global Search) High $C \Rightarrow$ good exploration efficiency

neighbors. Therefore, these relations can be applied on a set of each individual and the surrounding neighbors, promoting neighborhood exploitation and exploration of the search space.

2.2. Smallest-Small-World Theory. The Small-World network models have received much attention since their introduction by Watts and Strogatz [2] as models of real networks having characteristics which lie between random and regular as shown in Figure 2. They are characterized by two factors: the average path length (L) which measures efficiency of communication or passage time between nodes and the clustering coefficient (C) which represents the degree of local order. The average path length (L) is defined as the average number of links in the shortest path between a pair of nodes in the network. Accordingly, clustering coefficient (C) is defined as the probability that two nodes connected to a common node are also connected to each other. In general, cGA's population space is a regular network that has relatively high L and C from the point of view of the network theory. High L makes the interactions of remote nodes difficult. Otherwise, a Small-World network represents low L between any two arbitrary nodes and contains high C. It can be characterized by lots of local connectivity between nodes as well as the occasional longer links, defined as a shortcut. Only a few of these longer links are required in order to obtain a Small-World network state.

Nishikawa et al. [3] suggested a SSWN and verified their theory with examples. They showed that the L of a Small-World network with fixed shortcuts is a minimum when there is a "center" node, from which all shortcuts are connected to uniformly distributed nodes in the network [4]. An example of such a configuration is illustrated in Figure 3.

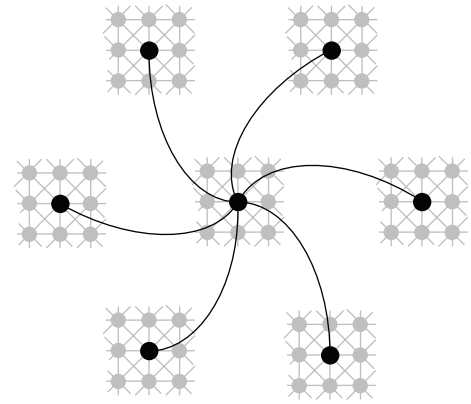


Figure 3: Examples of shortcut configuration with a center node.

They defined the SSWN theory as follows.

(i) A SSWN is composed of two parts: the underlying network (e.g., a regular lattice) and the subnetwork of shortcuts containing only the shortcuts and their nodes.

(ii) The nodes in the subnetwork of shortcuts must be uniformly distributed over the network.

(iii) Finally, among all possible configurations of connected subnetworks of shortcuts with uniformly distributed nodes, the ones with a single center involve the largest number of nodes.

These arguments indicate that, given a fixed number of shortcuts, the networks connected with a sub-network of shortcuts having uniformly distributed nodes have smaller L than a typical random configuration, and, among these, the ones with a single center minimize L. In other words, the Smallest-Small-World networks are characterized by these structures.

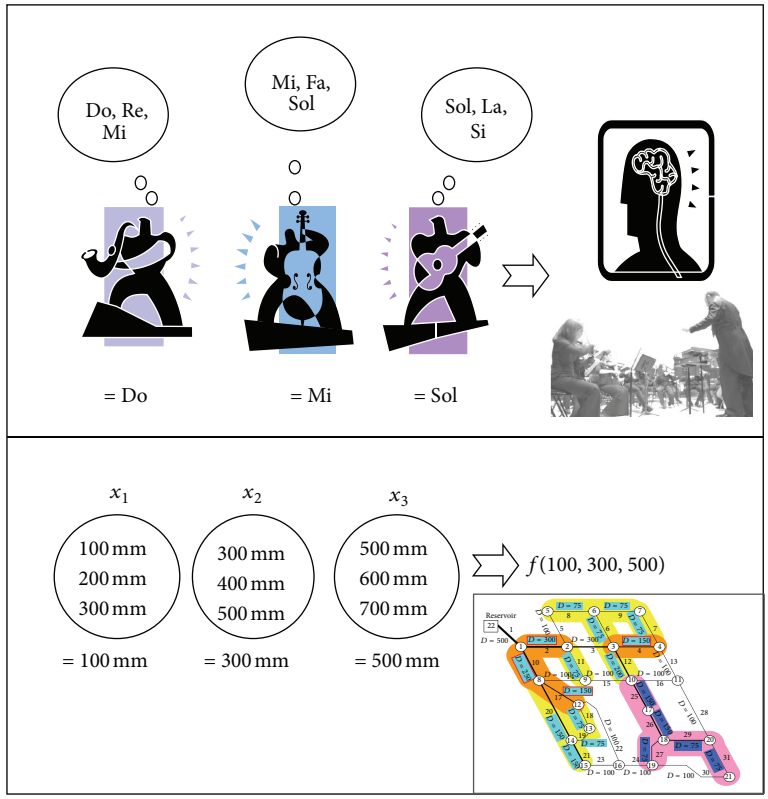


Figure 4: Relation between music improvisation and engineering optimization [15, 16].

Table 2: Definitions and specifications of two design variables problems.

Functions	Definition
Rosenbrock's valley	Minimize $f(x_1, x_2) = 100(x_1^2 - x_2^2)^2 + (1 - x_1)^2$ $-2.048 \le x_1 \le 2.048, -2.048 \le x_2 \le 2.048$ $x_1 = 1, x_2 = 1, f(1, 1) = 0$
Branin's function	Minimize $f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos(x_1) + 10$ $a = 1, b = 5.1/4\pi^2, c = 5/\pi, d = 6, e = 10, f = 1/8\pi$ $-5 \le x_1 \le 10, 0 \le x_2 \le 15$ $(x_1, x_2) = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$ $f(x_1, x_2) = 0.397887$
Easom's function	Minimize $f(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2\right)$ $-100 \le x_1 \le 100, -100 \le x_2 \le 100$ $(x_1, x_2) = (\pi, \pi), f(x_1, x_2) = -1$
Goldstein price's function	Minimize $f(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $\times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$ $-2 \le x_1 \le 2, -2 \le x_2 \le 2$ $(x_1, x_2) = (0, -1), f(x_1, x_2) = 3$
Six-hump camel back function	Minimize $f(x_1, x_2) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$ $-3 \le x_1 \le 3, -2 \le x_2 \le 2$ $(x_1, x_2) = (-0.0898, 0.7126), (0.0898, -0.7126)\ f(x_1, x_2) = -1.0316$

2.3. Harmony Search Algorithm. The HS algorithm is based on musical performance processes that occur when a musician searches for a better state of harmony such as during jazz improvisation. Jazz improvisation seeks musically pleasing harmony (a perfect state) as determined by an aesthetic standard, just as an optimization process seeks a global solution (a perfect state) as determined by an objective function. The pitch of each musical instrument determines the aesthetic quality, just as a set of values assigned to each decision determines the value of the objective function. Figure 4 shows the details of the analogy between music improvisation and engineering optimization [15, 16].

In musical improvisation, each player sounds any pitch within the possible range, together making one harmony vector. If all the pitches make a good harmony, that experience is stored in each player's memory, and the possibility of making a good harmony is increased next the time. Similarly, in engineering optimization, each decision variable initially chooses any value within the possible range, together making one solution vector. If all the values of the decision variables result in a good solution, that experience is stored in each variable's memory, and the possibility of making a good solution is also increased at the next iteration. In brief, the steps of HS algorithm are given as follows [16, 17].

Step 1 (initialize the problem and algorithm parameters). The optimization problem is defined as minimize $f(x)$ subject to $x_{iL} \le x_i \le x_{iU}$ $(i = 1, 2, \ldots, N)$ and other existing constraints. x_{iL} and x_{iU} are the lower and upper bounds of decision variables, respectively. The HS algorithm parameters are also specified in this step. They are the harmony memory size (HMS), or the number of solution vectors in the harmony memory, HMCR, bandwidth (b_w), PAR, and the number of improvisations (K), or stopping criterion.

Step 2 (initialize the harmony memory). The initial harmony memory (HM) is generated from a uniform distribution in the ranges $[x_{iL}, x_{iU}]$ $(i = 1, 2, \ldots, N)$, as given in (1):

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_N^1 \\ x_1^2 & x_2^1 & \cdots & x_N^2 \\ \vdots & \vdots & & \vdots \\ x_1^{\mathrm{HMS}-1} & x_2^{\mathrm{HMS}-1} & \cdots & x_N^{\mathrm{HMS}-1} \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \cdots & x_N^{\mathrm{HMS}} \end{bmatrix}. \quad (1)$$

Step 3 (improvise a new harmony). Generating a new harmony is called improvisation. The new harmony vector $x' = (x_1', x_2', \ldots, x_N')$ is determined by three rules: memory consideration, pitch adjustment, and random selection. The pseudocode for generating a new harmony search is given as shown in Algorithm 1 [17].

x_i' $(i = 1, 2, \ldots, N)$ is the ith component of x', and x_i^j $(j = 1, 2, \ldots, \mathrm{HMS})$ is the ith component of the jth candidate solution vector in the HM. *rand* is a uniformly generated random number in the region of $[0, 1]$, and b_w is an arbitrary distance bandwidth.

Step 4 (update harmony memory). If the fitness of the improvised harmony vector $x' = (x_1', x_2', \ldots, x_N')$ is better than that of the worst harmony, replace the worst harmony in the HM with x'.

Step 5 (check the stopping criterion). If the stopping criterion (maximum number of iterations, K) is satisfied, computation is terminated. Otherwise, go to Step 3. The most important step of the HS algorithm is Step 3, which includes memory consideration, pitch adjustment, and random selection. The PAR and b_w have a profound effect on the performance of the HS.

Table 3: Obtained statistical results of 2D benchmark problem.

	SSWCHS	CHS	SHS
1. Rosenbrock			
Mean	$1.74E-06$	$3.99E-08$	$2.37E-04$
Best	$3.60E-11$	$1.60E-11$	$4.07E-09$
Worst	$2.30E-05$	$4.29E-07$	$1.63E-02$
SD	$3.77E-06$	$7.16E-08$	$1.65E-03$
Feasible solution	100	100	89
Mean iteration	698.68	1,102.25	20,142.62
Iteration SD	574.89	1,030.11	1,091.92
2. Branin's function			
Mean	$3.98E-01$	$3.98E-01$	$3.98E-01$
Best	$3.58E-07$	$3.58E-07$	$3.58E-07$
Worst	$3.58E-07$	$3.58E-07$	$4.33E-07$
SD	$2.04E-11$	$1.29E-12$	$8.20E-09$
Feasible solution	100	100	89
Mean iteration	211.94	444.13	2,593.45
Iteration SD	124.18	225.88	225.88
3. Easom's function			
Mean	$-6.70E-01$	$-9.41E-01$	$-4.90E-01$
Best	$3.60E-13$	$3.60E-13$	$8.21E-13$
Worst	$1.00E+00$	$7.44E-01$	$1.00E+00$
SD	$4.70E-01$	$1.45E-01$	$5.00E-01$
Feasible solution	67	67	49
Mean iteration	1,501.27	2,461.70	10,817.10
Iteration SD	564.37	969.64	1,133.84
4. Goldstein price's function			
Mean	$3.00E+00$	$3.00E+00$	$1.92E+01$
Best	$0.00E+00$	$0.00E+00$	$0.00E+00$
Worst	$1.21E-05$	$1.12E-05$	$2.70E+01$
SD	$1.62E-06$	$1.93E-06$	$1.32E+01$
Feasible solution	100	100	40
Mean iteration	257.89	423.43	1,009.08
Iteration SD	150.64	177.71	280.99
5. Six-hump camel back function			
Mean	$-1.03E+00$	$-1.03E+00$	$-1.03E+00$
Best	$2.85E-05$	$2.85E-05$	$2.84E-05$
Worst	$2.85E-05$	$2.85E-05$	$2.85E-05$
SD	$2.51E-11$	$2.79E-10$	$1.36E-09$
Feasible solution	100	100	100
Mean iteration	87.34	152.80	852.06
Iteration SD	33.13	67.30	67.30

"SD" stands for standard deviation.

```
for each i ∈ [1, N] do
  if rand() ≤HMCR then
    x_i' = x_i^j (j = 1, 2, ..., HMS) % memory consideration
    if rand() ≤ PAR then
      x_i' = x_i' ± rand() × b_w % pitch adjustment
      if x_i' > x_iU
        x_i' = x_iU
      elseif x_i' < x_iL
        x_i' = x_iL
      end
    end
  else
    x_i' = x_iL + rand() × (x_iU − x_iL) % random selection
  end
end
```

Algorithm 1

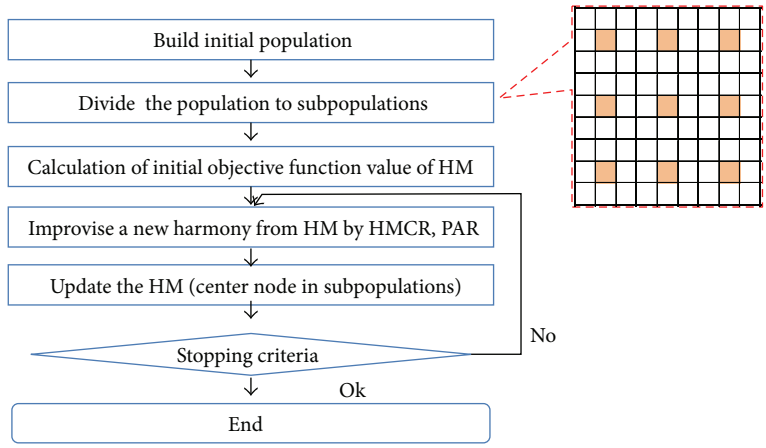


Figure 5: Flowchart of the CHS.

3. Smallest-Small-World Cellular Harmony Search

In the present paper, we proposed the improved HS algorithm that uses the CA concept and has the topological structure of the SSWN. A population is just a group of certain number of arbitrary objects in the same generation as for the generation in the original HS. Meanwhile, these objects are not related to each other. In the configuration of these populations as a kind of random network, the L and C are also very low. In contrast, if the population only consists of a grid of cellular networks, the C is relatively high; however, the L is also high. Because high L makes the interactions of remote nodes difficult, we need to reduce the L. Table 1 shows the efficiency and comparison of clustering coefficient and path length. The Small-World network models have the advantages of both random and regular network. They have low L for fast interaction between nodes, and they have high C ensuring sufficient redundancy for high fault tolerance [3]. In this study, therefore, the population of the original HS consists of a form of cellular networks. And then we can reduce characteristic path length (L) and increase the clustering coefficient (C) using the shortcuts concepts of Small-World network.

3.1. Cellular Harmony Search (CHS). The operation process of the cellular harmony search (CHS) is shown in Figure 5. The CHS's operation process is the same as the existing original HS. However, the CHS's initial population has a cellular form. In this process, new generations in the sub-network from the PAR and HMCR processes are compared with the existing population, and then replace the lowest ranked object. After each small grid within the objects of the highest priority, a node is located in the center of the sub-network. Finally, the centers of the sub-network nodes are compared, and the object of the entire population of the highest priority and the central node are replaced.

3.2. Smallest Small World Cellular Harmony Search (SSWCHS). Smallest Small World cellular harmony search (SWCHS) performs its operation between the center nodes

Table 4: Definitions and specification of 30D benchmark problems.

Functions	Definition
Step function	Minimize $f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2 - 30 \le x_i \le 30$ $\min(f) = f(0, \ldots, 0) = 0$
The Rastrigin function	Minimize $f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] - 5.12 \le x_i \le 5.12$ $\min(f) = f(0, \ldots, 0) = 0$
The Griewank function	Minimize $f(x) = (1/4000)\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1 - 600 \le x_i \le 600$ $\min(f) = f(0, \ldots, 0) = 0$
The Ackley function	Minimize $f(x) = -a\exp\left(-b\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}\right) - \exp\left((1/n)\sum_{i=1}^{n} \cos(cx_i)\right) + a + \exp(1)$ $a = 20, b = 0.2, c = 2\pi - 32.768 \le x_i \le 32.768$ $\min(f) = f(0, \ldots, 0) = 0$

Table 5: Results of 30ND Problems.

	SSWCHS	CHS	SHS
1. Step function			
Mean	$0.00E+00$	$1.09E+00$	$5.60E+00$
Best	$0.00E+00$	$0.00E+00$	$0.00E+00$
Worst	$0.00E+00$	$3.00E+00$	$1.20E+01$
SD	$0.00E+00$	$9.81E-01$	$2.53E+00$
Feasible solution	100	36	1
Mean iteration	30,673.72	148,717.39	91,353.00
Iteration SD	41,902.01	49,240.37	—
2. Rastrigin function			
Mean	$4.69E-02$	$2.99E+00$	$6.09E+00$
Best	$2.78E-03$	$1.12E-02$	$1.01E+00$
Worst	$1.00E+00$	$5.01E+00$	$8.98E+00$
SD	$1.95E-01$	$1.31E+00$	$1.99E+00$
Feasible solution	92	0	0
Mean iteration	113,790.16	—	—
Iteration SD	47,257.60	—	—
3. Griewank function			
Mean	$1.03E-01$	$1.43E-01$	$6.94E-01$
Best	$1.46E-06$	$2.71E-02$	$5.47E-06$
Worst	$4.86E-01$	$4.85E-01$	$1.25E+00$
SD	$9.80E-02$	$8.06E-02$	$4.47E-01$
Feasible solution	10	0	3
Mean iteration	73,642.00	—	208,726.00
Iteration SD	43,912.52	—	26,918.84
4. Ackley function			
Mean	$4.62E-02$	$1.31E+00$	$1.58E+00$
Best	$3.51E-03$	$6.10E-03$	$5.97E-03$
Worst	$1.50E+00$	$1.65E+00$	$3.46E+00$
SD	$2.11E-01$	$2.78E-01$	$7.17E-01$
Feasible solution	96	3	8
Mean iteration	79,262.84	141,376.33	160,133.00
Iteration SD	64,770.38	35,365.89	58,642.35

by adding shortcuts in the CHS. The CHS operation between the center nodes is added in the calculation process as shown in Figure 6. In this process, the SSWCHS is performed to obtain the final optimal solution through operations between the small grids of optimum object.

4. Applications and Results

In this study, the proposed SSWCHS was applied for solving unconstrained benchmark functions widely examined in the literature. The SSWCHS, CHS, and original simple harmony search (SHS) problems were performed in this study to have comparisons among optimizers. The optimization task was carried out using 100 independent runs based on the results depending on the type of problem. Statistical values, including best, worst, and mean values, and mean iteration number were obtained to evaluate the performance of the reported algorithms. Benchmark functions were utilized to evaluate the performance of considered optimization techniques. Among benchmark functions, five benchmark functions have 2 design variables and the rest have 30 design variables [18]. The benchmark functions have difficulty in terms of number of local optimum points and also the search space of these functions is almost wide and can challenge the efficiency of methods. The SSWCHS, CHS, and SHS algorithms were applied and run separately. The SSWCHS and CHS that occur at random initial solution were applied equally, and the initial solution in the SHS was applied differently because of the size of the HM. The HMS in the SSWCHS and CHS was composed by 225 cellular structures, and the SHS was composed by 5 HMS. Therefore, the number of shortcuts can be predetermined as shown in Figure 7. Two and 30 design variable problems were initialized using 0.3 PAR and 0.95 HMCR, respectively. The 2 and 30 design variable functions were performed with 50,000 iterations and 250,000 iterations, respectively. Feasible solution has a margin of error of 10^{-4} and 10^{-2} in the 2 and 30 design variable functions. Mean value and standard deviation of iteration can indicate the degree of convergence. From the number of feasible solution, the accuracy percentage of each algorithm can be compared.

4.1. Problems Having Two Design Variables. In this study, the following problems were applied as shown in Table 2.

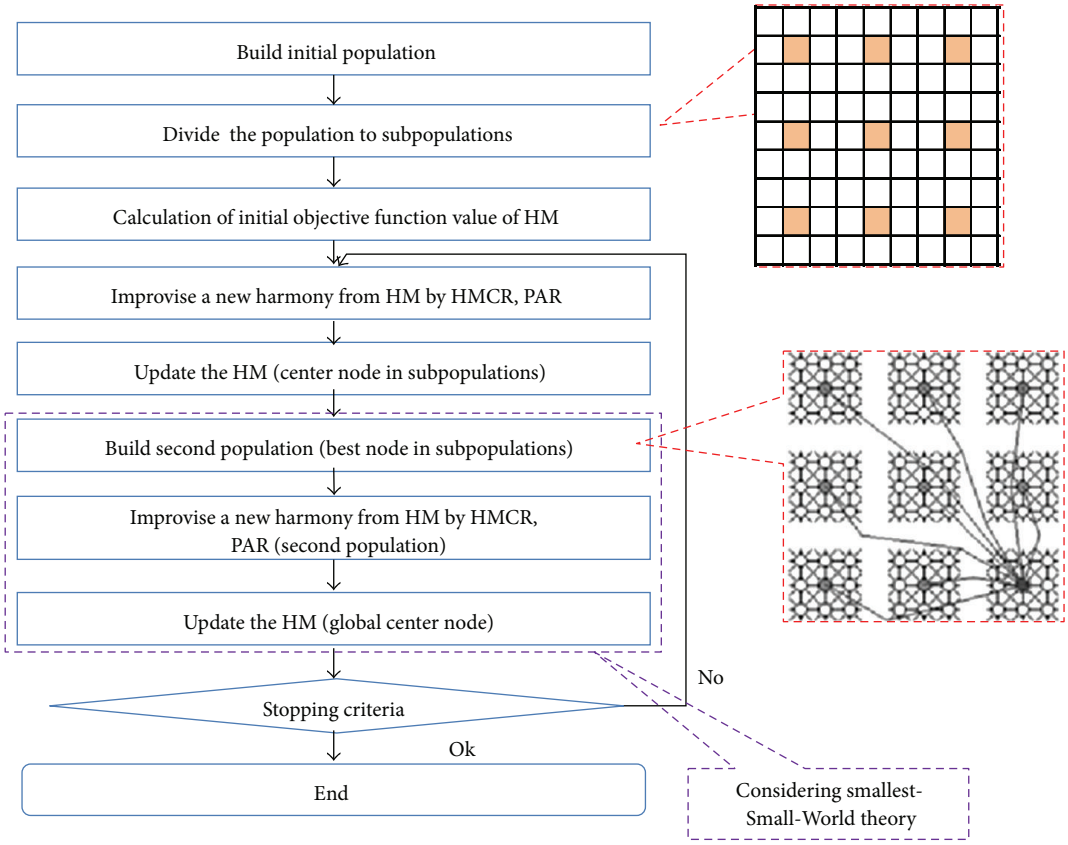


Figure 6: Flowchart of the SSWCHS.

Table 3 shows the obtained results of the 2D benchmark functions. The SSWCHS and CHS relatively showed good results, compared to the SHS. The result of each benchmark function indicated that Easom's function could not find perfect optimal solution by the SSWCHS and CHS. The parameters of Easom's function have a wide search range. For this reason, Easom's function showed less search capability for the optimal value than the other functions. In such case, increasing the number of iterations is considered to reach the optimal solution so that the ratio can be improved. One can see that the value of the entire functions which is mean iteration and iteration standard deviation value of the SSWCHS are smaller than those of the CHS. These results can conclude that the SSWCHS converged faster than the CHS, and it can estimate a reliable optimal solution.

4.2. Problems Having 30 Design Variables. In this study, the following benchmark problems were applied as shown in Table 4.

Table 5 shows the attained results of 30D benchmark functions. The SSWCHS relatively showed good results compared with the CHS and SHS. For all benchmark problems given in Table 4, except Griewank's function, the SSWCHS showed its superiority over other reported methods. In case of Griewangk's function, we can see that with the ratio of 10% the optimal solution is reached. Because it had wide search ranges of parameters from −600 to 600. By contrast to the 2D problems, 30D functions need more time to reach the optimal solution. However, the SSWCHS showed much higher performance than the CHS and SHS. In particular, the arithmetic mean and the standard deviation in the SSWCHS are lower than those in other algorithms, indicating stability and faster convergence. For the Ackley function and the Griewangk function, however, the CHS and SHS offered lower standard deviation of iteration than the SWSCHS. However, in this case, the number of feasible solutions was very small and mean iteration was larger than that obtained by the SSWCHS. In case of the percentage of optimal solutions of the SHS, the values of the Griewangk function and Ackley function were larger than those of the CHS.

5. Conclusions

In this study, an improved HS algorithm which combines the CA and the topological structure of Smallest-Small-World network is proposed. Most of previous studies, there have been a lack of studies on the performance improvement of the harmony search algorithm by use of population or memory structures. A new hybrid harmony search algorithm having high clustering coefficient and short characteristic path length was required. The hybrid HS algorithm developed in this paper has good exploration and exploitation efficiencies. Nine benchmark functions were applied to assess the performance of the proposed algorithm. The applied benchmark

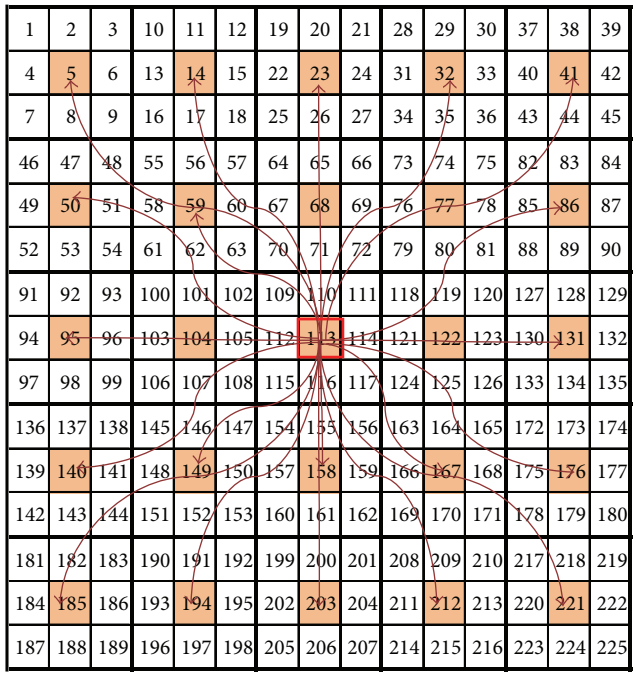


Figure 7: Smallest Small World harmony search network.

functions consist of five 2D functions and four 30D functions. The evaluation indexes of the SSWCHS were better than those of CHS and SHS in terms of solution quality. The SSWCHS algorithm showed generally faster convergence and more stability than the CHS or SHS. It shows very competitive solutions with less number of iterations than other considered algorithms. It is recommended that the optimization techniques, as new algorithms became available, be used in a wide range of engineering optimization problems. However, the SSWCHS has so many of the HS structures that it can affect computation time. Therefore, it remains a complementary part. As a further research, parameter variations are expected to develop using the proposed SSWCHS.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean government (MSIP) (no. 2013R1A2A1A01013886).

References

[1] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[2] D. J. Watts and S. H. Strogatz, "Collective dynamics of "small-world" network," *Nature*, vol. 398, pp. 440–442, 1998.

[3] T. Nishikawa, A. E. Motter, Y. C. Lai, and F. C. Hoppensteadt, "Smallest small-world network," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 66, no. 4, Article ID 046139, 2002.

[4] V. Latora and M. Marchiori, "Notions of local and global efficiency of a network," *Physical Review Letters*, vol. 87, Article ID 198701, 2001.

[5] T. W. Kang, "Smallest-small-world cellular genetic algorithm," *Journal of Computing Science and Engineering*, vol. 34, no. 11, pp. 971–983, 2007.

[6] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[7] J. H. Kim, Z. W. Geem, and E. S. Kim, "Parameter estimation of the nonlinear muskingum model using harmony search," *Journal of the American Water Resources Association*, vol. 37, no. 5, pp. 1131–1138, 2001.

[8] S. Wolfram, "Statistical mechanics of cellular automata," *Reviews of Modern Physics*, vol. 55, no. 3, pp. 601–644, 1983.

[9] J. Von Neumann, *Theory of Self-Reproducing Automata*, University of Illinois Press, Urbana, Ill, USA, 1966.

[10] B. Dorronsoro and E. Alba, "A simple cellular genetic algorithm for continuous optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 2838–2844, Vancouver, Canada, July 2006.

[11] G. Folino, C. Pizzuti, and G. Spezzano, "Combining cellular genetic algorithms and local search for solving satisfiability problems," in *Proceedings of the IEEE 10th International Conference on Tools with Artificial Intelligence*, pp. 192–198, November 1998.

[12] G. Timar and D. Balya, "Regular small-world cellular neural networks: key properties and experiments," in *Proceedings of the IEEE International Symposium on Cirquits and System*, pp. 69–72, British Columbia, Canada, May 2004.

[13] A. B. Hashemi and M. R. Meybodi, "A multi-role cellular PSO for dynamic environments," in *Proceedings of the 14th International CSI Computer Conference (CSICC '09)*, pp. 412–417, October 2009.

[14] V. Noroozi, A. B. Hashemi, and M. R. Meybodi, "CellularDE: a cellular based differential evolution for dynamic optimization problems," in *Proceedings of the ICANNGA*, vol. 6593 of *Lecture Notes in Computer Science*, pp. 340–349, 2011.

[15] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36–38, pp. 3902–3933, 2005.

[16] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[17] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation*, vol. 198, no. 2, pp. 643–656, 2008.

[18] M. Molga and C. Smutnicki, "Test functions for optimization needs," 2005, http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 590614, 18 pages
http://dx.doi.org/10.1155/2013/590614

Research Article

Selecting Optimal Feature Set in High-Dimensional Data by Swarm Search

Simon Fong,[1] Yan Zhuang,[1] Rui Tang,[1] Xin-She Yang,[2] and Suash Deb[3]

[1] *Department of Computer and Information Science, University of Macau, Macau*
[2] *Faculty of Science and Technology, Middlesex University, UK*
[3] *Department of Computer Science and Engineering, Cambridge Institute of Technology, Ranchi, India*

Correspondence should be addressed to Simon Fong; ccfong@umac.mo

Received 22 July 2013; Revised 1 October 2013; Accepted 8 October 2013

Academic Editor: Zong Woo Geem

Selecting the right set of features from data of high dimensionality for inducing an accurate classification model is a tough computational challenge. It is almost a NP-hard problem as the combinations of features escalate exponentially as the number of features increases. Unfortunately in data mining, as well as other engineering applications and bioinformatics, some data are described by a long array of features. Many feature subset selection algorithms have been proposed in the past, but not all of them are effective. Since it takes seemingly forever to use brute force in exhaustively trying every possible combination of features, stochastic optimization may be a solution. In this paper, we propose a new feature selection scheme called Swarm Search to find an optimal feature set by using metaheuristics. The advantage of Swarm Search is its flexibility in integrating any classifier into its fitness function and plugging in any metaheuristic algorithm to facilitate heuristic search. Simulation experiments are carried out by testing the Swarm Search over some high-dimensional datasets, with different classification algorithms and various metaheuristic algorithms. The comparative experiment results show that Swarm Search is able to attain relatively low error rates in classification without shrinking the size of the feature subset to its minimum.

1. Introduction

With the advances of information technology, it is not uncommon nowadays that data that are stored in database are structured by a wide range of attributes, with the descriptive attributes in columns and instances in rows. Often among the attributes, a column is taken as a target class. Therefore a classifier can be built by training it to recognize the relation between the target class and the rest of the attributes, with all the samples from the instances. This task is generally known as supervised learning or model training in classification. The model is induced by the values of attributes which are known as features because each one of these contributes to the generalization of the model with respective to its dimension of information.

For example, in the wine dataset [1] which is a popular sample used for benchmarking classification algorithms in machine learning community, thirteen attributes (features) of chemical analysis are being used to determine the origin of wines. Each one of these features is describing the origin of the wines which is the target class, pertaining to their respective constituents found in each of the three types of wines. The features essentially are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The features include variables of alcohol content, malic acid level, magnesium, flavonoids, phenols, proline, and color intensity. The multiple features essentially give rise to thirteen different dimensions of influences. In mutual consideration of these influences, a classifier is able to tell the type of wine when it is subject to an unknown sample (which again is described by these thirteen attributes). The multidimensions of these features are visualized in Figure 1. It can be seen that classification is a matter of computing the relations of these variables in a hyperspace to the respective classes; and of course the computation gets complicated as the amount of feature grows.

In machine learning, feature selection or input selection is often deployed to choose the most influential features

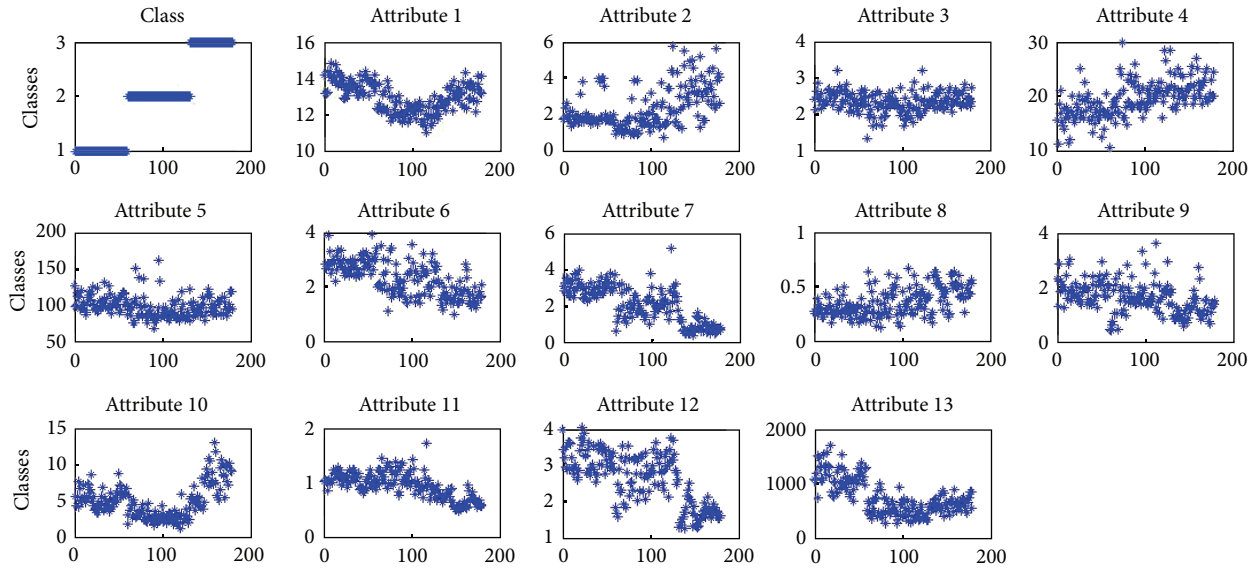


Figure 1: Visualization of multidimensions of features that describe the types of wine.

(inputs) from the original feature set for better classification performance. Feature selection is to find an optimal feature subset from a problem domain while improving classification accuracy in representing the original features. It can be perceived as an optimization process that searches for a subset of features which ideally is sufficient and appropriate to retain their representative power describes the target concept from the original set of features in a given data set. Through the process of feature selection, dimensions in the data are reduced, and the following advantages can be potentially attained: accuracy of the classifier is enhanced; resource consumption is lowered, both in computation and memory storage; it is easier for human to cognitively understand the causal relationship between features and classes with only few significant features.

In general, there is no best method known so far that can be unanimously agreed on among data mining researchers in implementing the optimum feature selection. Due to the nonlinear relations of the features and the concept targets, sometimes removing an attribute which has little correlation with the target variable (but unknowingly it is an important factor in other composite dependencies with the other attributes) may lead to worse performance. Tracing along the nonlinearity of the relation for evaluating the worthiness of the attributes is an exhaustive search in the high-dimensional space. Heuristics are often used to overcome the complexity of exhaustive search.

In the literature, many papers have reported on the use of heuristics to tackle the feature selection problems. However it is noticed by the authors that many proposed methods are limited to one or more of the following constraints in their designs. See Table 1 for details.

(1) The size of the resultant feature set is assumed to be fixed. Users are required to explicitly specify the maximum dimension as the upper bound of the feature set. Though this would significantly cut down the search time, the search is confined to only the combinations of feature subsets in the same dimensions. Assume that there is a maximum of k features in the original dataset. There are 2^k possible combinations of arranging the features in a subset of any variable size. With a bound s (for all $s \geq 1$) imposed on the size of the feature subset, it becomes a combinational problem of picking exactly s features into the subset from a total of k features. The number of combinations reduces from 2^k to $k!/(k-s)!s!$. The major drawback is that users may not know in advance what would be the ideal size of s.

(2) The feature becomes minimal. By the principle of removing redundancy, the feature set may shrink to its most minimal size. Features are usually ranked by evaluating their worthiness with respective to the predictive power of the classifier, and then eliminating those from the bottom of the ranked list. Again, there is no universal rule on how many features ought to be removed. Users may have to arbitrarily define a threshold value above which the ranked features form a feature set.

(3) The feature selection methods are custom designed for some particular classifier and optimizer. An optimizer is referred to some heuristics that explore the search space looking for the right feature set. As reported in most of the papers in the literature, a specific pair of classifier and optimizer is formulated out of many possible choices of algorithms.

The limitations above gave impetuses to the objective of research paper. We opt to design a flexible metaheuristic called Swarm Search that can efficiently find an optimal group of features from the search space; at the same time, there is no need for the user to fix the feature set size as Swarm Search would provide an optimal length of the feature set as well.

Table 1: A brief survey on feature selection models.

Reference	Classifier	Metaheuristic	No. of features	Fixed subset size	Domain
Talbi et al. [2]	SVM	PSO, GA	N/A	N/A	Gene microarray
Vieira et al. [3]	SVM	BPSO, GA	12, 28	No	SEPSIS data
Wang et al. [4]	RS	SS	14, 15	No	Credit scoring
Abd-Alsabour and Moneim [5]	SVM	ACO	17–70	No	General
Casado et al. [6]	DA	TS	54–121	Yes (5–8)	General
Jona and Nagaveni [7]	SVM	ACO, Cuckoo	78	Yes (5)	Mammogram
Unler et al. [8]	SVM	PSO	10–267	Min	General
Korycinski et al. [9]	BHC	TS	242	Yes (3–8)	Hyperspectral
Yusta [10]	N/A	GRASP, TS, MA	18–57	Yes (3–7)	General
Unler and Murat [11]	LR	PSO, SS, TS	8–93	Yes (3–8)	General
García-Torres et al. [12]	NB	TS	9–70	No	General
El Ferchichi and Laabidi [13]	SVM	TS, GA	24	No	Urban transport
Al-Ani [14]	ANN	ACO	40, 50	No	Speech, image

Legends: Particle swarm optimization (PSO), genetic algorithm (GA), support vector machines (SVM), binary particle swarm optimization (BPSO), scatter search (SS), rough set (RS), ant colony optimization (ACO), tubu search (TS), discriminant analysis (DA), binary hierarchical classifier (BHC), memetic algorithm (ma), greedy randomized adaptive search (GRASP), logistic regression (LR) classifier: naive Bayes (NB), Classifier, and Artificial neural network (ANN).

The mechanism of Swarm Search is universal in a sense that different classifiers and optimizers can just plug and play. As a comparative study, several popular classification algorithms coupled with several different most recent metaheuristic methods are integrated into Swarm Search in turn; their performances are evaluated over a collection of datasets of various multitudes of features. In essence the contribution of the paper is a Swarm Search feature selection method which can find optimal features as well as feature set size. The remaining paper is structured as follow. Some related feature selection methods are reviewed in Section 2. The Swarm Search method is described in detail in Section 3. Comparative experiments and their results are presented and discussed in Section 4. Section 5 at the end concludes the paper.

2. Related Work

In the past decades, many methods of feature selection have been proposed and studied extensively [15]. In general, there are three types of feature selection designs at the operation level and two types of feature searches at the data exploration level.

The three different operational designs are filter-based, wrapper-based, and embedded-based. Often features are evaluated individually in filter-based feature selection model. Features that are found redundant by means of some statistical computation are eliminated. Classification model that adopts filter-based approach evaluates and selects the features during the preprocessing step but ignores the learning algorithm. Hence the aspect of predictive strength of the classifier resulted from different combinations of features is not taken in account during the selection. These are simple selection procedures based on statistical criteria that can be found in a number of commercial data mining software programs. It was pointed out [16] that such methods are not very efficient; especially, when the data carry features, the maximum accuracy is rarely attained. Some typical measure criteria include information gain [17, 18] mutual information like redundancy relevance and class conditional interaction information, for measuring net relevance of features. Another classical filter method is correlation-based filter method (CFS) from [19] where correlation is used as selection criterion. CFS uses a scoring mechanism to rate and include those features that are highly correlated to the class attribute but have low correlation to each other. However, all the filter-based methods are considering relations among individual attributes and/or between pairs of attributes and target classes. The main drawback of this filter-type method is that it ignores the effect of the subset of features in the induction algorithm, as claimed in [20].

The other type of model is called wrapper-based model, which is popularly on par with the filter-type but generally yields better performance. Wrapper-based method works in conjunction with the induction algorithm using a wrapper approach [20]. The wrapper-based model selects features by considering every possible subset of them; the subsets of features are ranked according to their predictive power, while treating the inbuilt classifier as a black box. An improved version of wrapped-based model is proposed in [21]. It considers information of relevance and redundancy of the features in addition to their influences on the predictive power. The last type is embedded model, which in contrast to wrapper approaches, selects features while taking into account the classifier design [22]. This type is relatively rare for it involves specialized software design in order to enable its dual. Its popularity is hampered due to the complexity of the coding and implementation.

It is apparent that exhaustive search under all the wrapper-based and perhaps embedded approaches can be performed, if the number of feature is not too large. However, the problem is known to be NP-hard (Nondeterministic Polynomial Time), and the search quickly becomes computationally intractable. This is especially true for wrapper-based

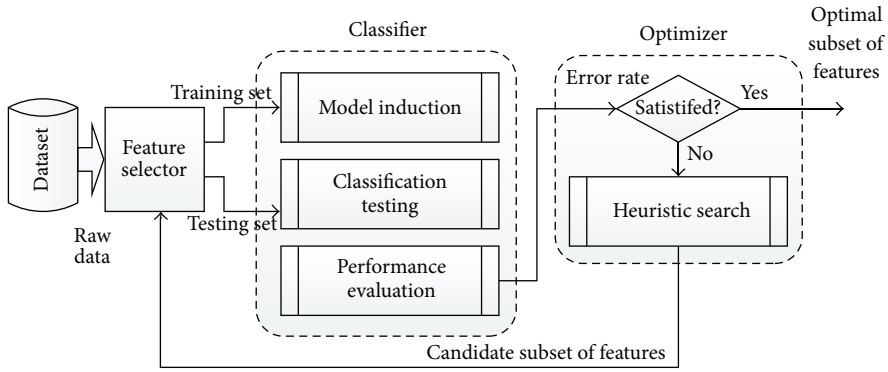


Figure 2: Process model of the proposed swarm search.

models that do rely on a classifier to determine whether a feature subset is good or not. The classifier is being induced continually as the model tries out a new feature subset in a vast search space. The repetition of classifier induction surely incurs high computational costs when the search of feature subsets is brute-force.

In order to alleviate the problem of intensive computation, metaheuristic search instead of brute-force search is augmented with certain classifiers in the wrapper-based models. Therefore, an optimal (reasonably good) solution is obtained in lieu of deterministic solution which may take forever to discover. In the literature, a number of contributions belong to this category of hybrid models. Not meant to be exhaustive, Table 1 lists some works in which metaheuristic algorithms are used together with classifiers, in a wrapper approach. In common they are proposed to be targeted at solving the high-dimensionality problems in feature selection by searching for optimal feature subset stochastically. However, some of the works assume that a user-input threshold value is used for defining the size of the feature subset, which was already mentioned as one of the disadvantages in Section 1.

Comparing to the earlier research in the 80s and 90s where mostly feature selection belonged to the filter-type focusing mainly on the pairwise relations between the features and the targets recent advances trend towards using wrapper methods to guide the feature selection based on the direct predictive influences by an augmented classifier. Different metaheuristics are used to generate an optimal solution. However, most of the works concentrate on a single type of classifier as well as one or a few specific metaheuristics only. As it can be seen from Table 1, the amount of original features that were being verified by their models in experiment, limit from 8 to 267, except may be the work [2] on microarray. These relatively small feature numbers may not be sufficiently for stress-testing the performance of the proposed algorithms. Lastly but importantly, the model designs in quite a number of works [6, 7, 9–11] are constrained by the need of inputting a pre-defined value for the length of the resultant feature subset; [8], on the other hand, programmed the selection to produce the least amount of features in a feature subset. Often the globally optimal solution, that yields the highest classification accuracy or other performance criterion by the fitness function, resides in feature subsets beyond this specific predefined subset length.

Given these shortcomings as observed from the literature review, it inspires to design an improved feature selection model that is flexible so that different classifiers and metaheuristic optimizer can be easily and compatibly interchanged; the optimal feature subset would be sought on the fly without the need of setting an arbitrary subset length and putting the model under tougher tests of more features (hence higher dimensionality).

3. Proposed Model

3.1. System Work Flow. The design of our model is wrapper-based that consists of two main components—classifier and optimizer as shown in Figure 2. Each of these components can be installed with the corresponding choice of algorithm at the user's will. Generically, a classifier can embrace software codes of SVM, ANN, decision tree, and so forth. optimizer can be implemented by any metaheuristic algorithm which searches for the optimal feature subset in the high-dimensional search space.

Over the original dataset which is formatted by all the original features vertically and the full set of instances horizontally, the feature selector module retrieves and cuts the data as per requested by the classifier module and optimizer module. The optimizer module signals the feature selector with the requested features, as resulted from its latest search. The feature selector then harvests the required data of only those feature columns and sends them to the classifier. The classifier uses a 10-fold cross validation method for dividing the corresponding dataset chosen by the feature selector, for training and testing a classifier model (in memory). Classification performance results such as accuracy % and/or other measures could be used as the fitness to the fitness function. In this case, the quality of the classifier model is regarded as the fitness function. As usual a convergence threshold can be set, so that if the fitness value falls below the threshold which is the stopping criterion, the iteration of the stochastic search would stop.

Overall, we can see that the process model of our proposed Swarm Search is comprised of two phases. In

the first phase, which is called feature subset exploration and selection, the optimizer searches for the next best feature subset according to the logics of the metaheuristic algorithm installed at the optimizer module which selects the best subset. At the first run, the feature subset would be chosen randomly. In the second phase, which is called learning and testing, at the classifier module, a classifier model is induced from the training data, that is partitioned by feature selector with the best feature subset, and is tested on the test data.

The two phases are repeated continually until some stopping criterion is reached, either the maximum number of runs is reached or the accuracy of the classifier is assessed to be satisfactory. In each repetition, the relative usefulness of the chosen subset of features is evaluated and supposedly will improve in comparison to that of the previous round. The classifier here is taken as a black box evaluator, informing how useful each chosen subset of features is by its accuracy.

This wrapper approach by default could be brute-force by starting with the first feature subset and tests on all other possible subsets until the very last subset. Obviously this involves massive computation time, or even intractable. For only 100 features, there are $2^{100} \approx 1.2677 \times 10^{30}$ possible subsets for the classifier model to be repeatedly built and rebuilt upon. This high computation costs may further intensify if the training data has many instances.

In this regard, we propose to use coarse search strategies like those described in Section 3.1 for the Swarm Search model. Instead of testing on every possible feature subset, the Swarm Search which is enabled by multiple search agents who work in parallel would be able to find the most currently optimal feature subset at any time. It was however pointed out by Yang [23] that there is no free lunch in achieving the optimal feature subset. The feature subset may be the best so far the search has been outreached in the search space, but the absolute global best solution will not be known until all have been tried which may never occur because of the computational infeasibility.

3.2. Swarm Search. A flexible metaheuristic called Swarm Search (SS) is formulated for feature selection by combining any swarm-based search method and classification algorithm. Given the original feature set of size k, $A = \{a_1, a_2, \ldots, a_k\}$, SS supports constructing a classifier with classification accuracy rate e by providing an optimal feature subset $S \in A$ such that $e(A) \geq e(S)$, where S is one of all the possible subsets that exist in the hyperspace of A.

The core of SS is the metaheuristic algorithm that scrutinizes the search space for an optimal feature subset. Three metaheuristics are used in the experiments in this paper. They are particle swarm optimization (PSO), bat algorithm (BAT), and wolf search algorithm (WSA).

3.2.1. Metaheuristics Algorithms. PSO is inspired by the swarm behaviour such as fish, birds, and bees schooling in nature. The original version was developed by Kennedy and Eberhart in 1995 [24]. The term called Swarm Intelligence was coined since PSO became popular. The dual movement of a swarming particle is defined by its social sense and its own cognitive sense. Each particle is attracted towards the position of the currently global best g^* and its own locally best position l_i^* in history. While at the same time with the effect of the attraction, it has a tendency of moving randomly.

Let l_i and v_i be the location vector and velocity for particle i, respectively. The new velocity and location and location updating formulas are determined by

$$v_i^{t+1} = v_i^t + \alpha\varepsilon_1\left[g^* - l_i^t\right] + \beta\varepsilon_2\left[l_i^* - l_i^t\right], \quad l_i^{t+1} = l_i^t + v_i^{t+1}, \tag{1}$$

where ε_1 and ε_2 are two random vectors, and each entry taking the values between 0 and 1. The parameters α and β are the learning parameters or acceleration constants, which can typically be taken as [25], $\alpha \approx \beta \approx 2$.

BAT is a relatively new metaheuristic, developed by Yang in 2010 [25]. It was inspired by the echolocation behaviour of microbats. Microbats use a type of sonar, called echolocation, to detect prey, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more often use constant-frequency signals for echolocation. Their signal bandwidth varies depending on the species and often increases by using more harmonics. Three generalized rules govern the operation of the bat algorithm, and they are as follow.

(1) All bats use echolocation to sense distance, and they also know the difference between food/prey and background barriers by instinct.

(2) Bats fly randomly with velocity v_i at position x_i with a fixed frequency $f_{\min}$, varying wavelength λ, and loudness A_0 to search for prey. They can automatically adjust the wavelength of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target.

(3) Although the loudness can vary in many ways, we assume that the loudness varies from a large and positive A_0 to a minimum constant value $A_{\min}$.

The new solution x_i^t and velocities v_i^t at time step t are given by

$$f_i = f_{\min} + \left(f_{\max} - f_{\min}\right)\beta, \quad v_i^t = v_i^{t-1} + \left(x_i^t - x_*\right) f_i, \quad x_i^t = x_i^{t-1} + v_i^t, \tag{2}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution. Here x_* is the current global best location which is located after comparing all the solutions among all the n bats. As the product $\lambda_i f_i$ is the velocity increment, we can use either f_i or λ_i to adjust the velocity change while fixing

the other factor. Initially, each bat is randomly assigned a frequency which is drawn uniformly from $[f_{\min}, f_{\max}]$. For local search, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A^t, \tag{3}$$

where $\varepsilon \in [-1, 1]$ is a random number, while $A^t = \langle A_i^t \rangle$ is the average loudness of all the bats at this time step. The update of the velocities and positions of bats have some similarity to the procedure in the PSO as f_i essentially controls the pace and range of the movement of the swarming particles. The loudness and rate of pulse emission have to be updated accordingly as the iterations proceed. As the loudness decreases once, a bit has found its prey, while the rate of pulse emission increases. Consider

$$\begin{gathered} A_i^{t+1} = \alpha A_i^t, \qquad r_i^{t+1} = r_i^0 \left[1 - \exp\left(-\gamma t\right)\right], \\ A_i^t \longrightarrow 0, \quad r_i^t \longrightarrow r_i^0, \quad \text{as } t \longrightarrow \infty. \end{gathered} \tag{4}$$

WSA [26] is one of the latest metaheuristic algorithms by the authors; its design is based on wolves' hunting behavior. Three simplified rules that govern the logics of WSA are presented as follow.

(1) Each wolf has a fixed visual area with a radius defined by v for X as a set of continuous possible solutions. In 2D, the coverage would simply be the area of a circle by the radius v. In hyperplane, where multiple attributes dominate, the distance would be estimated by the Minkowski distance, such that

$$v \le d\left(x_i, x_c\right) = \left(\sum_{k=1}^{n} \left|x_{i,k} - x_{c,k}\right|^{\lambda}\right)^{1/\lambda}, \quad x_c \in X, \tag{5}$$

where x_i is the current position, x_c are all the potential neighboring positions near x_i and the absolute distance between the two positions must be equal to or less than v, and λ is the order of the hyperspace. For discrete solutions, an enumerated list of the neighboring positions would be approximated. Each wolf can only sense companions who appear within its visual circle, and the step distance by which the wolf moves at a time is usually smaller than its visual distance.

(2) The result or the fitness of the objective function represents the quality of the wolf's current position. The wolf always tries to move to better terrain, but rather than choosing the best terrain it opts to move to better terrain that already houses a companion. If there is more than one better position occupied by its peers, the wolf will chose the best terrain inhabited by another wolf from the given options. Otherwise, the wolf will continue to move randomly in Brownian motion.

The distance between the current wolf's location and its companion's location is considered. The greater this distance is, the less attractive the new location becomes, despite the fact that it might be better. This decrease in the wolf's willingness to move obeys the inverse square law. Therefore, we get a basic formula of betterment:

$$(r) = \frac{I_o}{r^2}, \tag{6}$$

where I_o is the origin of food (the ultimate incentive) and r is the distance between the food or the new terrain and the wolf. It is added with the absorption coefficient, such that using the Gaussian equation, the incentive formula is

$$\beta(r) = \beta_o e^{-r^2}, \tag{7}$$

where β_o equals I_o.

(3) At some point, it is possible that the wolf will sense an enemy. The wolf will then escape to a random position far from the threat and beyond its visual range. The movement is implemented using the following formula:

$$x(i) = x(i) + \beta_o e^{-r^2}\left(x\left(j\right) - x\left(i\right)\right) + \text{escape}(), \tag{8}$$

where escape() is a function that calculates a random position to jump to with a constraint of minimum length, v, x is the wolf which represents a candidate solution, and $x(j)$ is the peer with a better position as represented by the value of the fitness function. The second term of the above equation represents the change in value or gain achieved by progressing to the new position. r is the distance between the wolf and its peer with the better location. Step size must be less than the visual distance.

Given these rules, we summarize the WSA in pseudocode as in Algorithm 1.

3.2.2. Searching for Optimal Feature Subsets. Each search agent in the metaheuristic (PSO/BAT/WSA) is coded as a solution vector which contains a list of feature indices that form a feature subset. The length of a solution vector is variable and it represents the length of the feature subset. At initialization, the vector length is randomized for each search agent. So they have different lengths to start with, in a way similar to placing the search agents in random dimensions of the search space. Each search agent has a variable k, which is the current length of feature subset it represents, where $1 \le k \le K$. K is a constant that is the maximum cardinality of the search space that is also the maximum number of the original features. During the Swarm Search, the search agent steps in the same dimension in each local step movement. The agent explores feature subsets in same dimension. It will only change its dimension at exceptional events. The exceptional events are similar in context to the escape function in WSA, mutation in GA, where a search agent suddenly deviates largely from its local search area (dimension). The exceptional events are usually programmed to happen randomly with

```
Objective function f(x), x = (x_1, x_2, ..., x_d)^T
Initialize the population of wolves, x_i (i = 1, 2, ..., W)
Define and initialize parameters:
r = radius of the visual range
s = step size by which a wolf moves at a time
α = velocity factor of wolf
p_a = a user-defined threshold [0..1], determines how frequently an enemy appears
WHILE (t < generations && stopping criteria not met)
  FOR i = 1 : W  // for each wolf
    Prey_new_food_initiatively();
    Generate_new_location();
    // check whether the next location suggested by the random number generator is new. If
not, repeat generating random location.
    IF (dist(x_i, x_j) < r && x_j is better as f(x_i) < f(x_j))
      x_i moves towards x_j // x_j is a better than x_i
    ELSE IF
      x_i = Prey_new_food_passively();
    END IF
    Generate_new_location();
    IF (rand ()> pa)
      x_i = x_i + rand () + v; // escape to a new pos.
    END IF
  END FOR
END WHILE
```

Algorithm 1: Pseudocode for the WSA algorithm.

a user-defined probability. At the occurrence of such events, the k variable changes randomly to other value. In other words, the search agent "jumps" out of its local proximity to another dimension of the search space. For example, an agent has the current dimension $k = 4$; in the next iteration k may change to 5, 9785 or any other dimension when an exceptional event occurs. Or else, it would remain the same dimension and explore other combinatorial subsets in the same dimension. The vector of a search agent with $k = 4$ may take a sample form $\langle 1, 2, 5, 8\rangle$ that means that the vector consists of the 1st, 2nd, 5th, and 8th features as the representative indices. A sample snapshot of a search agent exploring the search space and its possible moves is shown in Figure 3.

In our design, search agents normally explore in the same dimension in each movement. This is shown as the grey arrows for the case of WSA, hunting for food (looking for a feature subset that produces higher accuracy) randomly in the same dimension. In this way, it only can achieve a local optimal in the same dimension. Because the agents have the tendency to swarm towards their neighbour that have a better fitness function, an active agent (a_1, a_3) may merge to its peer agent (a_1, a_3, a_4) provided that the peer agent is having a better fitness.

At times, a dimension change function gets activated by the randomizer; the active agent, for example, may "jump" out of its dimensions to another dimension. The random function rand() is a standard Matlab function which produces uniformly distributed pseudorandom numbers. Another random function is to compute about how far the dimension displacement can take, with the direction of (+/−) randomly generated too. The length of jump, however, cannot exceed beyond the boundary. For instance, the active search agent is now at (a_1, a_3), it cannot fall beyond dimension 1. Likewise it can never go beyond the maximum dimension K.

The operation of the optimizer called Swarm Search takes the following steps.

(1) Initialize search agents. For each search agent, randomly assign a different feature combination as a subset.

(2) Calculate the fitness of the objective function for each agent, rank the agents by their fitness values, and record the one that has the highest fitness as the most promising solution.

(3) Perform local search and optimize the current best solution. This step may vary slightly for different metaheuristic algorithms. However, generaly, it tries to update the promising solution unless some satisfactory terminal condition is met. In PSO, the swarming particles have velocities. So we need not only to update their positions, but also to update their velocities. Recording the best local solution and global solution in each generation is required. In BAT, for each generation, rank the current best position according to the pulse rate and loudness; then update the velocity and position for each agent. In WSA, each search agent is updated by three intermediate procedures: first randomly move to update the position; second, check if they are ready to swarm towards to the closest and best companions; third, activate the escape function at random-relocate agents that are qualified to jump dimensions.

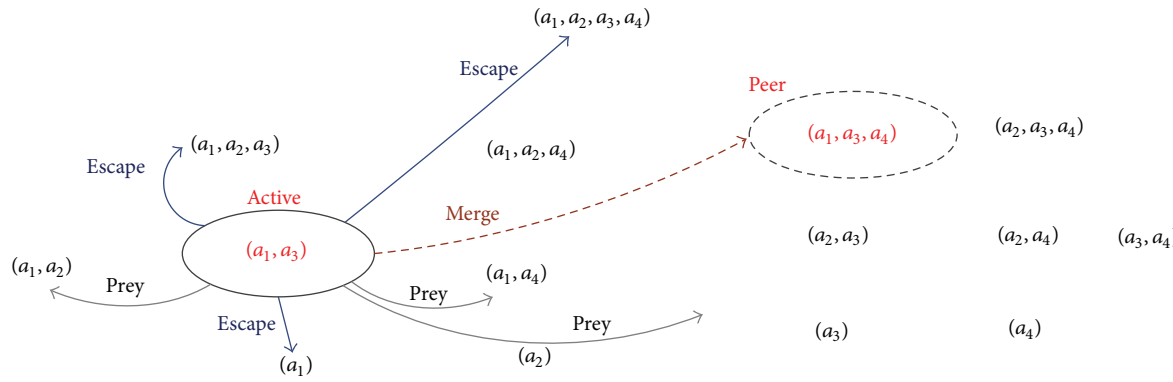


Figure 3: An illustration of possible moves of a search agent in the search space.

Table 2: List of feature selection methods used in experiments.

Method abbreviation	Classifier	Metaheuristic	Approach
FS-PSO-PN	Pattern Network	Particle swarm optimization	Swarm
FS-PSO-DT	Decision Tree	Particle swarm optimization	Swarm
FS-PSO-NB	Naïve Bayes	Particle swarm optimization	Swarm
FS-BAT-PN	Pattern Network	Bat algorithm	Swarm
FS-BAT-DT	Decision Tree	Bat algorithm	Swarm
FS-BAT-NB	Naïve Bayes	Bat algorithm	Swarm
FS-WSA-PN	Pattern Network	Wolf search algorithm	Semiswarm
FS-WSA-DT	Decision Tree	Wolf search algorithm	Semiswarm
FS-WSA-NB	Naïve Bayes	Wolf search algorithm	Semiswarm
FS-Cfs-PN	Pattern Network	Correlated-based filter	Filter
FS-Cfs-DT	Decision Tree	Correlated-based filter	Filter
FS-Cfs-NB	Naïve Bayes	Correlated-based filter	Filter

(4) Change dimension randomly.

(5) Check if the terminating criteria are met; if not go back to step (2).

(6) Terminate and output the result.

4. Experiment

The Swarm Search feature selection is designed for achieving the following merits (1) low error rate for the classifier as a feature selection tool to harvest the optimal feature subset out from a prohibitively large number of subsets; (2) reasonable time for the convergence of the metaheuristic search when compared to brute-force exhaustive search (whose time-taken we can safely assume to be extremely long); (3) an optimal size of feature subset that leads to optimal accuracy for the particular classifier; and (4) its universality in working across different classifiers and metaheuristic optimizers. Concerning the proposed four merits, experiments by computer simulation are then carried out accordingly. We target to evaluate the performance with respect to the first three virtues: (i) average error rate which is simply the number of incorrectly classified instances divided by the total number of testing instances; (ii) the total time consumed in finding the optimal feature subset including the training and testing of the respective classifier, in unit of seconds; and (iii) the length of the feature subset as the percentage of the maximum dimension of the original features.

In order to validate the universality of the proposed Swarm Search model, three popular classification algorithms have been chosen to be the kernels of the Classifier, and three contemporary metaheuristic methods are utilized as the core optimizer to perform the Swarm Search exploration. Furthermore, we use an industrial standard feature selection method called *correlation-based filter selection* (Cfs) for as a baseline benchmark for comparing with the proposed methods. Total there are twelve different feature selection methods that are programmed and tested in our experiments; see Table 2.

The classification algorithms as mentioned in Table 2 have been defined and studied in [27]. For the metaheuristic methods, readers who want to have the full details are referred to [24] for PSO, [25] for BAT, and [26] for WSA. Both PSO and BAT have substantial swarming behaviour for their movements that are governed with velocity; a single leader who has the greatest attractiveness is among the agents the remaining agents follow the leader's major direction, while at the same time they roam locally. In contrast, the agents in WSA are autonomous; they do merge nevertheless only when they come in contact with their visual range. So

Table 3: Datasets that are used in feature selection experiments.

Dataset	Attributes	Instances	Abstract/URL
Lung Cancer	56	32	The data described 3 types of pathological lung cancers http://archive.ics.uci.edu/ml/datasets/Lung+Cancer
Heart Disease	75	303	The data refers to the presence of heart disease in the patient http://archive.ics.uci.edu/ml/datasets/Heart+Disease
Libra Movement	91	360	The data refers to hand movement type in LIBRAS, Brazilian signal language http://archive.ics.uci.edu/ml/datasets/Libras+Movement
Hill Valley	101	606	Either a Hill (a "bump" in the terrain) or a Valley (a "dip" in the terrain) http://archive.ics.uci.edu/ml/datasets/Hill-Valley
CNAE-9	875	1080	Nine categories of free text business descriptions of Brazilian companies http://archive.ics.uci.edu/ml/datasets/CNAE-9

their approach is type of semiswarm. Cfs [19] evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low intercorrelation are preferred. The operation of Cfs is filter-based and sequential, hence no swarm.

4.1. Experimental Data and Setup. The twelve feature selection methods with combinations of different classifiers and optimizers are put under test across five well-known dataset from UC Irvine Machine Learning Repository (short for UCI). UCI is popular data archive for benchmarking the performance of machine learning algorithms. The five datasets that are chosen contain various numbers of features, ranging from 56 to 875. They are Lung Cancer, Heart Disease, Libra Movement, Hill Valley, and CNAE-9. Details of these datasets are outlined in Table 3. In addition to a variety of attribute numbers for stress testing the feature selection algorithms, the five datasets are chosen with purposes. Lung cancer is solely a categorical classification problem, where all predictive attributes are nominal, taking on integer values 0–3. The data described 3 types of pathological lung cancers. However this dataset is having relatively few instances available for model training in respective to attributes, with the ratio of attributes-to-instances 56 : 32. It represents a tough disproportional problem in training an accurate classification model. The dataset Heart Disease represents a mixed multivariate data-type problem. Libra Movement dataset represents a challenging hand movement recognition problem, where the data attributes are coordinates, solely numeric, and the data are in sequential order as they were extracted from video frames. Similarly, Hill Valley is a purely numeric and sequential time-series. Sampling at regular intervals over the x-axis, a variable value fluctuates on y-axis over time. An accurate prediction model for recognizing whether the sequential points are forming a hill or valley is known to be hard to build in the context of pattern recognition with a single sequential variable. CNAE-9 is extremely challenging, because of the large number of attributes that resulted from the word frequency counts. The words and their combinations that appear in each document may have subtle relations to the nature of the document. All these five chosen datasets present different levels and perspectives of challenges in constructing an accurate classification model and hence in testing the efficacy of each feature selection algorithm.

The main performance indicator in the experiments is classification error rate which is also used as stop criterion. The Swarm Search is programmed to explore and rank the solution continually until the difference of the average error rate of the current iteration and that of the recorded best are less than 1.0×10^{-8}. The other criterion is the maximum iteration which is set arbitrarily at 100,000,000 cycles. The maximum run cycle is to prevent the Swarm Search from running exceedingly long. The dataset, CNAE-9 with the greatest amount of features 875, potentially has 875 dimensions in the search space, and this amounts to 2^{875} ($\approx 2.519 \times e^{+263}$) combinatorial feature subsets. This enormously large number obviously prevents any normal computer from an exhaustive search. Even using Swarm Search, it will take a very long time indeed, while a deterministic solution is not guaranteed.

For the optimizer, default parameter settings are adopted for the metaheuristic methods; the default values are shown in Table 4.

We have to be very careful in choosing the parameters as they are sensitive to performance. For different metaheuristic methods, the parameters vary. In BAT, the bat position is influenced by the bat's echo loudness and echo pulse rate, so adjusting the loudness and emitted pulse depends on the proximity of their target. Here, we use 0.5 as their values, which shall result in a balance of intensification and exploration. The PSO has two learning factors, c_1, c_2. Generally, $c_1 = c_2$ and the value is smaller than 2 as this has been widely used in the literature. In our experiment, we use 1.5 which is moderately less than 2. For WSA, each search agent has a visual circle and the visual distance is the visual radius. In WSA, there is an escape function that simulates how a wolf runs far away from the current position when danger is encountered. The jump over is assumed farther than its visual radius. In our experiment, we set 5 as the escape distance that is five times larger than

Table 4: Parameter settings for metaheuristic methods.

FS using BA		FS using PSO		FS using WSA	
Parameter	value	Parameter	value	Parameter	value
A (loudness)	0.5	Populations	5	Visual distance	1
R (pulse rate)	0.5	c_1	1.5	Escape distance	5
Populations	5	c_2	1.5	Escape probability	0.25
$Q_{\min}$	0			Population	5
$Q_{\max}$	0.2				

Table 5: Comparative results in classification error for all the classification algorithms with Cfs feature selection and swarm search feature selection and without any feature selection.

Algorithm	Lung Cancer (56)	Heart Disease (75)	Libra Movement (91)	Hill Valley (101)	CNAE (875)
PN	0.5938	0.4653	0.1944	0.4447	0.8741
DT	0.5	0.4719	0.3028	0.505	0.112
NB	0.375	0.4422	0.3722	0.5149	0.0685
Cfs-PN	0.3125	0.462	0.1972	0.5017	0.187
Cfs-DT	0.3438	0.4818	0.2583	0.4951	0.1889
Cfs-NB	0.2813	0.4224	0.3528	0.4884	0.1824
PSO-PN	0.125	0	0.075	0.3465	0.0083
PSO-DT	0.225	0	0.2806	0.2112	0.0407
PSO-NB	0.1513	0.105	0	0.0495	0.8889
BAT-PN	0.1188	0	0.0833	0.1997	0.037
BAT-DT	0.21	0	0.3444	0.2442	0.0491
BAT-NB	0.1625	0.27	0.1028	0.5132	0.8889
WSA-PN	0.1338	0	0.0694	0.1997	0.037
WSA-DT	0.0407	0	0.2028	0.2178	0.037
WSA-NB	0.1575	0	0	0.1151	0.8889

the visual range. In addition, the escape probability is the probability of encountering danger. For a fair comparison, each metaheuristic method has 5 searching agents which are supposed to run in parallel. They are however programmed to run in sequential on a normal computer (instead of a parallel computer); the five agents move stochastically in round robin under the optimization loop. The programs are coded in Matlab R2010b. The computing environment is a MacBook Pro (with CPU: 2.3 GHZ, RAM: 4 GB).

4.2. Experimental Results. The experiments are mainly grouped into three kinds, testing for the performance of feature selection in classification error, in time consumption, and in amount of features obtained in the feature subset. For classification errors, Table 5 tabulates the results for classifiers using algorithms of Pattern Network (PN), which is also known as Artificial Neural Network, Decision Tree (DT), and naïve Bayes (NB), as well as the results that produced with their feature selection counterparts. Around the five different datasets, the results pertaining to classification errors, are plotted in radar charts visually in Figures 4, 5, and 6, respectively. Likewise, the results for consumption time are in Table 6 and Figures 7, 8, and 9. Table 7 and Figures 10, 11, and 12 are for feature amount in feature subgroup. Please note that for clarify in radar charts, the feature amounts are normalized to percentage with respect to the original feature

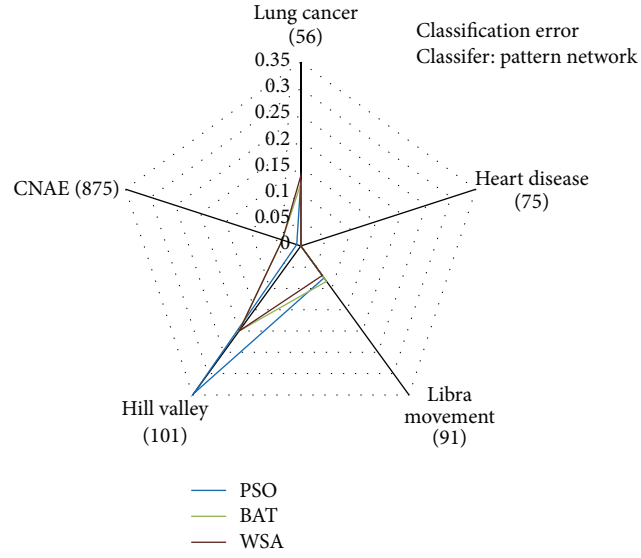


Figure 4: Results in classification error for Pattern Network are shown visually in radar chart.

numbers when charted. While the tabulated results allow comparison between swarm search and no swarm search at easy glance, the radar charts show individually the efficacy of

Table 6: Comparative results in time consumption (in seconds) for all the classification algorithms with Cfs feature selection and swarm search feature selection and without any feature selection.

Algorithm	Lung Cancer (56)	Heart Disease (75)	Libra Movement (91)	Hill Valley (101)	CNAE (875)
PN	3.07	2.35	18.08	58.73	5066.72
DT	0.02	0.08	0.32	0.16	3.39
NB	≈0	0.01	0.05	0.08	0.26
Cfs-PN	0.18	≈0	3.79	0.3	10.51
Cfs-DT	≈0	0.01	0.04	≈0	0.08
Cfs-NB	≈0	1.26	0.02	≈0	≈0
PSO-PN	853.23123	382.3217	1714.34495	921.28525	70538.3472
PSO-DT	177.3407	11.9377	232.6246	491.7232	2886.9232
PSO-NB	22.9799	3.92775	82.7123	34.344	2358.2769
BAT-PN	1532.321	347.0041	686.557	524.367	26365.1231
BAT-DT	179.1062	16.5652	240.737	448.7332	2341.6668
BAT-NB	25.8876	2.436514	28.6922	31.001802	1441.7241
WSA-PN	1521.8085	1305.6519	3005.4028	1159.367	26365.1231
WSA-DT	1521.8085	49.2341	106.6606	176.6559	82688.1231
WSA-NB	42.2628	23.3551	404.6113	24.7167	236365.123

Table 7: Comparative results in feature amount in feature subgroup for all the classification algorithms with Cfs feature selection and swarm search feature selection and without any feature selection.

Algorithm	Lung Cancer (56)	Heart Disease (75)	Libra Movement (91)	Hill Valley (101)	CNAE (875)
PN/DT/NB	56	75	91	101	875
Cfs-PN/DT/NB	11	9	26	1	28
PSO-PN	41	21	75	96	853
PSO-DT	39	44	66	79	854
PSO-NB	37	18	19	9	76
BAT-PN	41	17	39	88	849
BAT-DT	29	50	79	84	781
BAT-NB	38	5	6	43	243
WSA-PN	41	21	45	30	849
WSA-DT	41	23	40	44	851
WSA-NB	35	19	20	82	323

each swarm search variant with respect to the five datasets in visual comparisons.

4.3. Result Analysis. The classification error rates vary a lot across different datasets and with different classifiers. For instance, perfect accuracy is achieved for Heart Disease dataset with all three different optimizers and three different classifiers, except PSO-NB and BAT-NB where their accuracies are 89.5% and 73%, respectively. Moderate error rates are observed for Lung Cancer dataset, but relatively high error rate is incurred on the remaining datasets like Libra Movement, Hill Valley, and CNAE. In one extreme case, for CNAE dataset, Naïve Bayes classifier suffers at almost 89% error rate in all situations of optimizers, but over the same CNAE dataset, classifiers of Pattern Network and Decision Tree can achieve very low error rate less than 5%. This extreme performance is shown in Figure 6. As CNAE represents the worst case of highest dimensionality, Pattern Network classifier managed to conquer the problem of high dimensionality quite well especially with PSO optimizer (less than 1% error); See Figure 4. It works quite well with BAT and WSA too (at 3.7% error) considering that the dataset is text mining in nature and there are many variables for determining the category of documents that the text would belong to. Naïve Bayes apparently is rated down upon this CNAE dataset. However, it works well for the rest except Hill Valley which is basically time-series dataset with 101 data points forming a shape of hill or valley. The hills and valleys in the time series form nonlinear recursion which is post-nonlinear combination of predecessors. The classifier needs to recognize the shape by processing the values of the data points. The mathematical challenge is due to the problem of post-nonlinear multidimensional data projection. Naïve Bayes did not do well with Bat but otherwise with PSO (<5% error) and with WSA (<12% error), as shown in Figure 6. An interesting phenomenon is observed here, that BAT seems work quite well with Pattern Network and Decision Tree with relatively low or moderate errors. But in general BAT does not perform well with Naïve Bayes, especially over datasets that are very unstructured in nature like text mining and

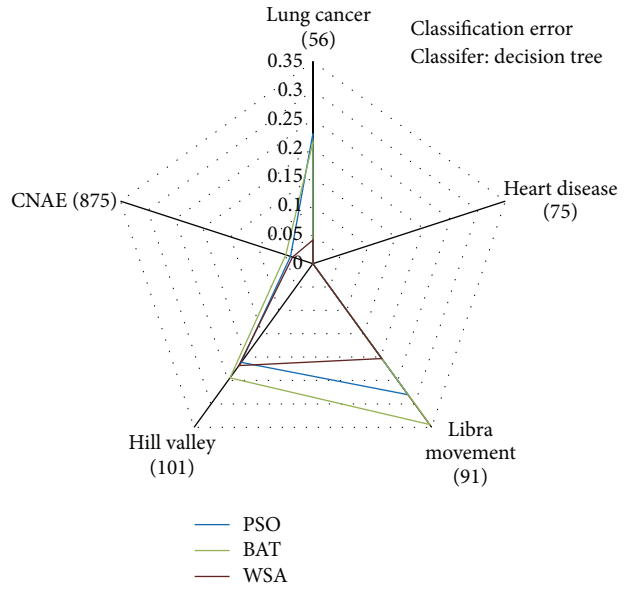


Figure 5: Results in classification error for Decision Tree are shown visually in radar chart.

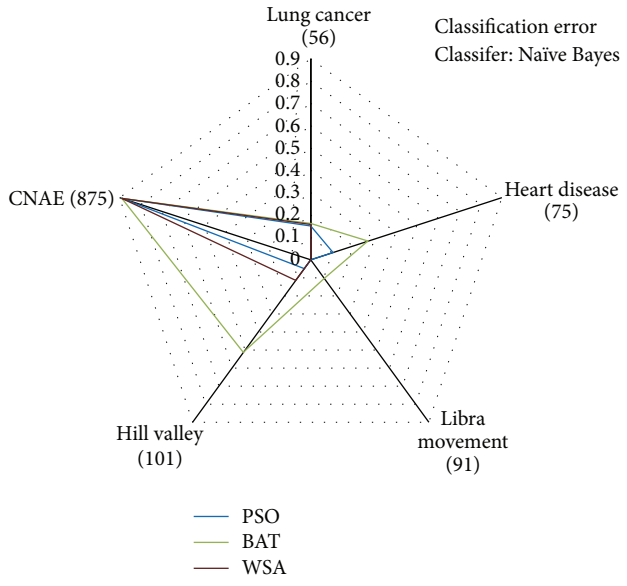


Figure 6: Results in classification error for Naïve Bayes are shown visually in radar chart.

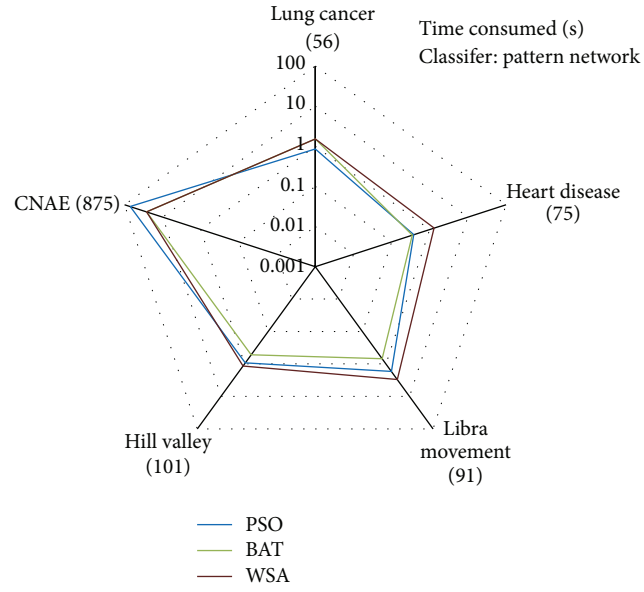


Figure 7: Results in consumption time for Pattern Network are shown visually in radar chart.

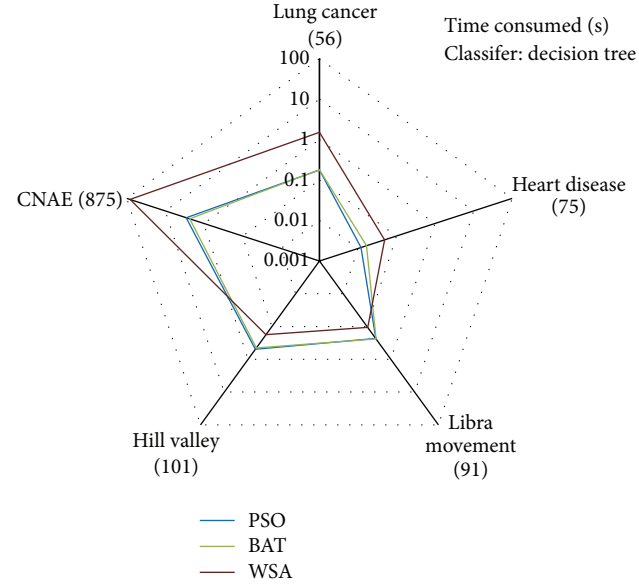


Figure 8: Results in consumption time for Decision Tree are shown visually in radar chart.

time-series data points. This may be due to the algorithmic characteristic of BAT where it may become too intensified in local search and trapped in some local optima; hence low quality feature subsets were found as solutions. PSO seems to have the same problem too, but its strong swarming ability might have led the search out local optima relatively easily, for example, Hill Valley with Naïve Bayes, and the same with DT where PSO shows its superiority by swarming out of the local optimal time-series data points. In general, WSA seems to work best with Naïve Bayes, except for CNAE. Nevertheless for relatively well-structured datasets like Lung Cancer and Heart Disease, in general, all the three classifiers and the tree optimizers generate good results. For time-series data like Hill Valley and Naïve Bayes with PSO and WSA are good choices. For text mining like CNAE that has huge dimensionality, Pattern Network is superior regardless of what optimizer is to be used; this may be attributed to the high capacity of nonlinearity of a neural network where the relations in the dimensional data variables can be modeled well in layers of complex neurons and their weights. On the other hand, the Swarm Search methods do not work well with Naïve Bayes as classification error is more than 80%

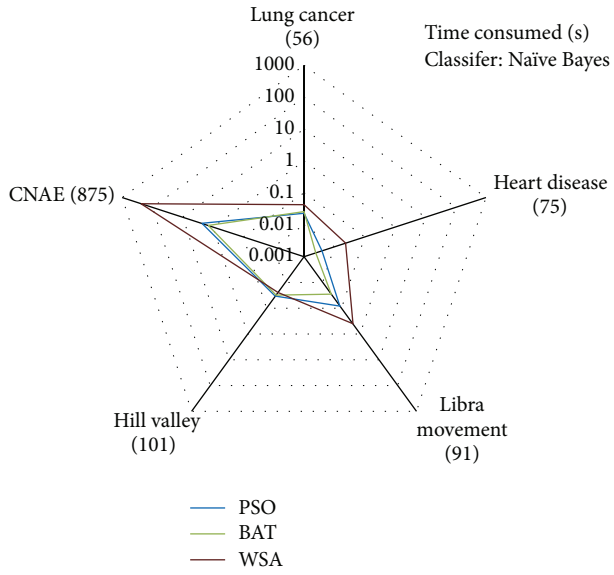


Figure 9: Results in consumption time for Naïve Bayes shown visually in radar chart.

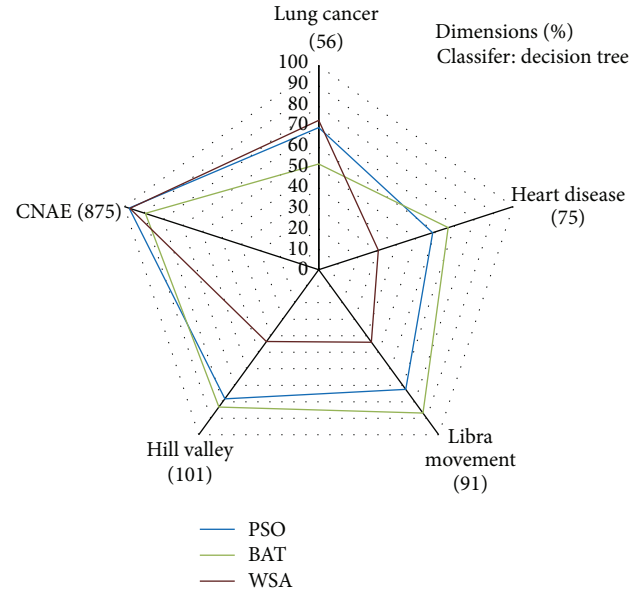


Figure 11: Results in % dimension for Decision Tree shown visually in radar chart.

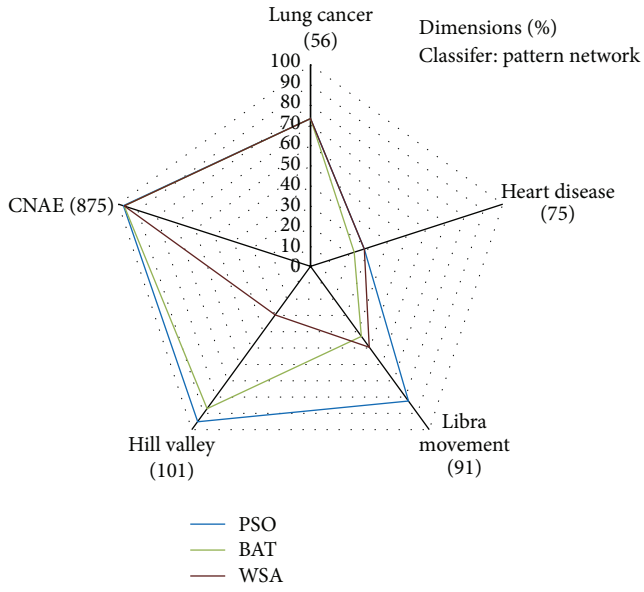


Figure 10: Results in % dimension for Pattern Network are shown visually in radar chart.

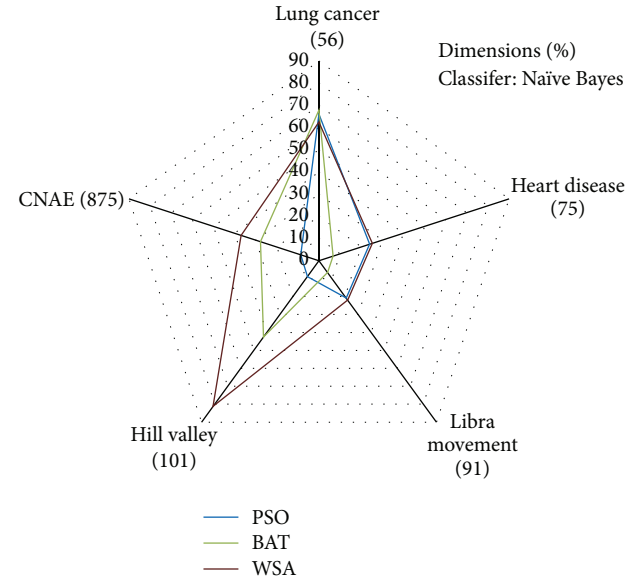


Figure 12: Results in % dimension for Naïve Bayes shown visually in radar chart.

for CNAE. That is basically because Naïve Bayes classifier has an assumption where all the attributes are independent. However for the data of which the attributes represent, it is not actually the case. Attributes for CNAE indeed have contextual relations though they may be very subtle and very nonlinear. Hence, the actual relations between attributes are never random; for example, certain frequently appearing words in a document follow some semantic grammars. Words and sentences do exist together in some sequential orders. Swarm Search however operates in stochastic manner where the order of the features in the subset is randomized in each move, so is the change of feature subset size at random intervals.

As shown in Figure 5, Decision Tree may not be a good candidate in Swarm Search as it incurs high error rates in three datasets such as Lung Cancer, Libra, and Hill Valley, though it can overcome the high-dimensionality problem imposed by CNAE. However, in Table 5, PSO-DT has lower error rate than PSO-PN, and DT works well for CNAE for all Swarm Search methods. As shown in Figures 7, 8, and 9, the time consumption for the combined process of feature selection and classifier induction generally increases as the

amount of attributes and instances increases. Generally, the time taken is highest for Pattern Network and shortest for Naïve Bayes, which is quite common as it is well know that neural network takes a long time to train. While PSO and BAT took about the same and reasonable time in processing every dataset, WSA shows its disadvantage prominently in CNAE and Libra. It implies that in search space that represents data of very high dimensionality, WSA takes exceptional long time to converge. This is clearly due to its semiswarm characteristics; the search agents are actively out exploration in wide dimensions, and they hardly would converge until a global optimum is found which may be rare. The unusually long time is observed at processing Lung Cancer and Heart disease datasets as well; these datasets are considered relatively easy and lead to fast convergence for the other optimizer but not for WSA, as shown in Figure 8. But for time-series data like Hill Valley, WSA can perform on-par or even outperform the other optimizers despite the fact that Hill Valley has high 101 dimensions. The data points in the time series have relatively low variation and high continuation, and the search agents by WSA are able to track along the feature subsets because the agents rely much on their visual range to swarm and search. BAT, in general, is fast and consumes the least amount of time in all the datasets. PSO nevertheless took slightly longer time than WSA in processing CNAE and Hill Valley when the classifiers Pattern Network and Decision Tree are used respectively.

Figures 10, 11 and 12 essentially show about how much in percentage of the size of the original feature set that the selected optimal feature subset contain. In other words, this is the length of the selected feature subset in percentage of the maximum dimension of the original feature set. For example, if there are originally 100 features in the data and the optimal feature subset is found to be $\langle a_x, a_y, a_z \rangle$, the *% dimension* is 3% for the optimal feature subset. As it was pointed out in Section 2, sometimes it may not be desirable to obtain the very minimum number of features but rather the sufficient amount of features in the right dimension that contribute to the greatest accuracy. This is based on a fair assumption that the best solution may not always reside in the low dimensions. Reaching out high to high dimension is one of the important abilities for Swarm Search.

As it can be shown in the results, classifiers Pattern Network and Decision Tree generally can retain most of the feature lengths (>50%); but the methods with Naïve Bayes tend to concentrate on solutions in the lower dimensions for finding the resultant feature subset. But there is an exception in Figure 12, WSA paired with Naïve Bayes retains over 80% of the maximum dimension in the Hill Valley dataset. It shows that this particular combo is able to reach out to a very high dimension for obtaining the optimal feature subset. It does not mean however that the absolute optimum solution must be there, because verifying so would be computationally intractable. PSO + NB on the other hand in Figure 12 acquires its solutions in relatively low dimensions in datasets of Heart Disease, Libra, and Hill Valley. Similar but not so obviously the same phenomenon is observed with Decision Tree in Figure 11, and in contrast PSO can acquire solutions at high dimension with Pattern Network in Figure 10. This demonstrates an interesting phenomenon that PSO being a strongly swarming method, when equipped with very nonlinear classifiers like Pattern Network and Decision Tree, its search agents can swarm their ways far and wide to high dimensions for scouting for a solution. On the other hand, when paired with probabilistic model like Naïve Bayes, PSO does not swarm far and wide like how it did with nonlinear functions. The same phenomenon is observed in BAT as well; BAT swarms relatively well in Decision Tree and Pattern Network, except for the Libra dataset in Figure 10. In fact, BAT did not swarm out far with Pattern Network in Figure 10, but it did so in superior with Decision Tree in Figure 11. With Decision Tree, BAT also beats the other optimizers in datasets like Heart Disease, Libra, and Hill Valley, in outreaching to space of high dimensions. See Figure 11. Perhaps the decision rules in Decision Tree model guide the balance of intensification and exploration well during the navigation of the agents in the search space for the BAT operation. In conclusion, WSA relatively does not reach out high because it does not swarm well in Pattern Network and Decision Tree, but it did extremely well in time-series data, Hill Valley when paired with Naïve Bayes, which is known as coarse strategy.

The results over different datasets are now aggregated for comparison of feature selection method groups. In detail, the results are tabulated in Table 8. The corresponding bar charts that show each performance aspect (error, time and % dimension) are shown in Figures 13, 14, and 15 for easy comparison. The error performance and % dimension of correlation-based feature selection method (Cfs) are included as well, as benchmarking references in comparison.

Obviously, it can be seen that, in Figure 13, the errors by Cfs are very high compared to those of all the other swarm-based methods. As a general trend, Pattern Network yields the lowest error rate given its complex and nonlinear weight distribution in its interconnected neuron networks. It seems to be able to handle classification problems very well. Second is Decision Tree and third is Naïve Bayes. Overall, as in Figure 13, WSA yields relatively lowest error among the other optimizers. But these benefits of low errors are out-weighted by the very high time consumption as shown in Figure 12 for WSA. Regarding the classifier algorithm, Naïve Bayes when paired with WSA has a very high time consumption; it shows that the searching agents in WSA have difficulty in converging when Naïve Bayes is being used, while, at the same time, a high cost in error rate too for WSA-NB exists. This may be due to the semiswarm behavior in WSA. But paring Naïve Bayes with strong swarming optimizers like PSO and BAT, they converge more quickly. As shown in Figures 13 and 14, however, pairing Pattern Network with PSO incurs a much long computation time than pairing with BAT or WSA. That is when a highly nonlinear classifier meets the most swarming optimizer. Nevertheless, this pairing does not offer proportionally good error rate either. It is worth mentioning that Cfs executes extremely fast (<1 second), and relatively classifiers such as Decision Tree and Naïve Bayes when coupled with PSO and BAT work very fast too. The exception is WSA which is quite slow and it was already mentioned.

Table 8: Comparison of individual performances of different algorithm combinations.

	FS-PSO			FS-BAT			FS-WSA			FS-Cfs		
	FS-PSO-PN	FS-PSO-DT	FS-PSO-NB	FS-BAT-PN	FS-BAT-DT	FS-BAT-NB	FS-WSA-PN	FS-WSA-DT	FS-WSA-NB	FS-Cfs-PN	FS-Cfs-DT	FS-Cfs-NB
Average error (per classifier)	0.11096	0.1515	0.23894	0.08776	0.16954	0.38748	0.08798	0.09966	0.2323	0.46562	0.58604	0.62314
Average error (per algo. group)		0.167133333			0.214926667			0.13998			0.558266667	
St. dev. error (per algo. group)		0.065406582			0.154928957			0.080164471			0.082350811	
Time consumption (per classifier)	14881.90607	760.10988	500.44817	5891.07444	645.36168	305.9484432	6671.47066	16908.49644	47372.0138	≈0	≈0	≈0
Av. time con. (per algo. group)		5380.821372			2280.794854			23650.6603			≈0	
St. dev. time con. (per algo. group)		8229.204933			3131.196153			21171.35171			≈0	
% Dimension (per classifier)	75.23341747	75.33096362	25.70943097	64.57907591	75.53820549	30.29258042	55.47927538	57.73169913	45.58275197	16.94849599	16.9489599	16.94849599
Av. % Dimension (per algo. group)		58.75793735			56.80328727			52.93124216			16.94849599	
St. dev. % dim. (per algo. group)		28.62088765			23.60378812			6.462861923			≈0	

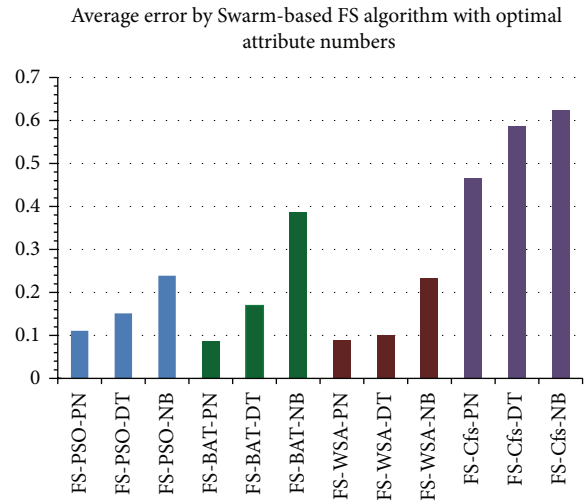


Figure 13: Comparison of different Swarm-based FS algorithms with optimal features in average error.

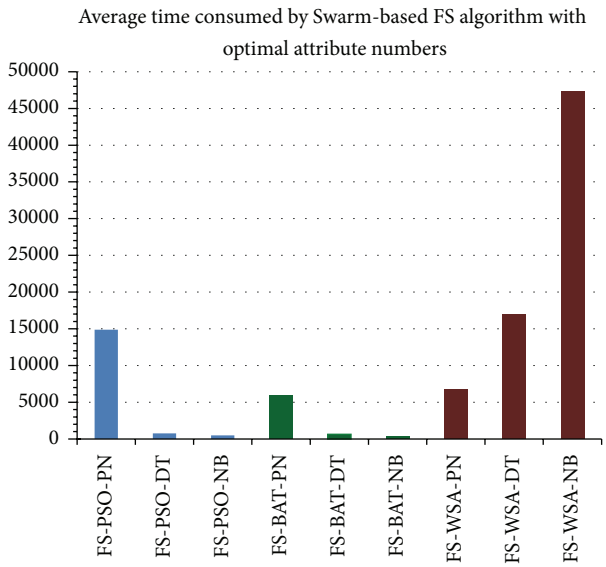


Figure 14: Comparison of different Swarm-based FS algorithms with optimal features in consumed time.

With respect to dimension exploitation, as shown in Figure 15, Cfs rates out as it only reaches to about 18% of the maximum dimension. The strongly swarming optimizers like PSO and BAT are able to climb up to 75% of the maximum dimension for acquiring optimal solutions. PSO with Pattern Network or Decision Trees is equally well, but PSO works poorly with Naïve Bayes in dimension exploitation. Similarly, BAT does not work well with Naïve Bayes too in outreaching to high data dimensions.

From the above results, it is verified that Swarm Search is in superior to the standard feature selection method, in terms of error rate and how high the data dimensions that the search agents can exploit. The advantage of Swarm Search however is at the cost of time consumption in the performing the heuristic search in high dimensional space.

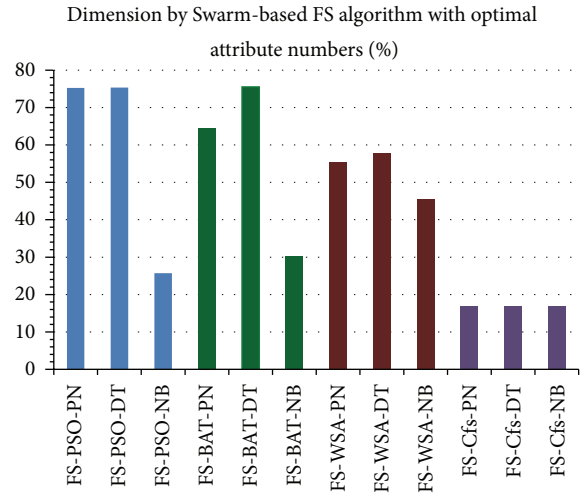


Figure 15: Comparison of different Swarm-based FS algorithms with optimal features in % dimension.

Lastly,the performance results in terms of error, time, and % dimensions are further aggregated into groups characterized by the metaheuristic methods, such as FS-PSO, FS-BAT, FS-WSA, and FS-Cfs (which is just for reference). The ultimate aggregated results are tabulated in Table 9. The corresponding performance with respect to error rate versus time consumption is shown in Figure 16. Such that the methods that are closer to the zero coordinate the better (low error and short time).

Here we observe a very interesting phenomenon; FS-CFS correlation-based group is the quickest, but it has the highest error rate. FS-WSA group which represents the semiswarm heuristics conversely yields the lowest error rate but costs the longest time. The two remaining groups of strong swarm-type sit near the optimal position along the cost-and-benefit curve. When considering all the results aggregated over different data sets and different popular classifiers, FS-BAT group has a relatively higher error rate than FS-PSO, but they run at relatively shorter convergence time than FS-PSO.

5. Conclusion

Dimensionality reduction in input dataset is crucial in machine learning, especially when dealing with a high-dimensional feature space. The original feature space is mapped onto a new space of reduced dimensions. Identification of relevant features is extremely important for classification tasks (improving accuracy and reducing computational costs). Given the high dimensionality in the data, selecting a right subset of useful features from all the original features is a challenging problem. There is no golden rule of thumbs how this should be done albeit a lot of research efforts have been going on, advocating different feature selection methods.

However, to the best of the authors' knowledge, no universal method is being claimed, though recently a high surge in hybrid modes of integrating metaheuristics into

Table 9: Comparison of overall performances of different Swarm-based feature selection algorithms.

Performance	FS-PSO	FS-BAT	FS-WSA	FS-Cfs
Average error	0.16713333	0.21492667	0.13998	0.558267
Average time	5380.82137	2280.79485	23650.6603	≈0
Average % dim	58.7579374	56.8032873	52.9312422	16.9485

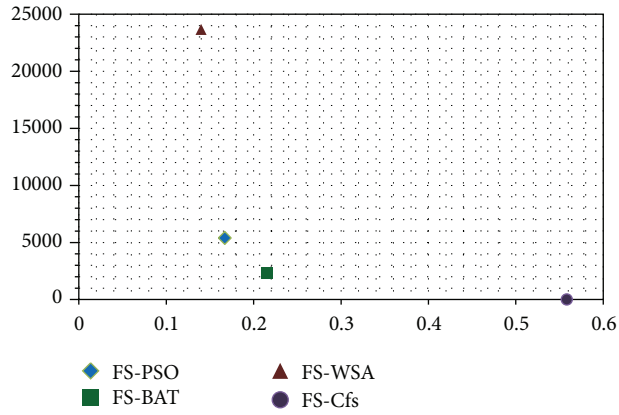


Figure 16: Overall comparison of Swarm-search models in error rates versus time consumption.

wrapper-based feature selection method is seen in the literature. The designs of the proposed feature selection models in most of those works focus on either a single classification algorithms or a single metaheuristic method. Also in some works, the feature selection is limited to a specific feature set length that has to be inputted manually by the user. Moreover, the experiments are subject to datasets of merely dozens of features in some of them.

In view of these, we proposed in this paper a general feature selection model, called Swarm Search which can potentially work with any classification algorithms as the fitness function and any swarm-based metaheuristics for searching in the search space stochastically for an optimal feature subset. The optimal feature subset needs not to be defined of its length in advance; we leave it to the search agents to scrutinize high dimensional data space to find it.

Under the framework of Swarm Search, nine metaheuristic feature selection methods are devised by combing three popular classifiers and three contemporary bioinspired optimizers. The proposed Swarm Search methods were put under test across five different types of datasets with a high dimensionality up to 875. Their performances were evaluated via the computer simulation in Matlab; the results suggest that Swarm Search models have superiority over correlation-based feature selections. In general, Swarm Search methods can yield relatively lower error rate and be able to outreach to high dimensional space for searching for an optimal feature subset. The performance of feature selection by Swarm Search is superb in nominal, pure numeric, and mixed-typed datasets. It pairs well with different classifiers with an exception of Naïve Bayes where the attributes in the data are not independent. The performance results are compared and contrasted in this paper, which can serve as a future reference guideline for scientists who want to choose these metaheuristic-based feature selection methods.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper. The authors of this paper do not have a direct financial relationship with the commercial identities mentioned in this paper that might lead to a conflict of interests.

Acknowledgments

The authors are thankful for the financial support from the research grant of Grant no. MYRG152(Y3-L2)-FST11-ZY, offered by the University of Macau, RDAO. Special thanks go to Mr. Rui Tang, who was a research assistant and MSc software engineering graduate of University of Macau, for programming the Matlab codes and conducting the data mining experiments.

References

[1] S. Aeberhard, D. Coomans, and O. de Vel, "Comparison of classifiers in high dimensional settings," Tech. Rep. 92-02, Department of Computer Science and Department of Mathematics and Statistics, James Cook University, North Queensland, Australia, 1992.

[2] E.-G. Talbi, L. Jourdan, J. García-Nieto, and E. Alba, "Comparison of population based metaheuristics for feature selection: application to microarray data classification," in *Proceedings of the 6th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA '08)*, pp. 45–52, Doha, Qatar, April 2008.

[3] S. M. Vieira, L. F. Mendonca, G. J. Farinha, and J. M. C. Sousa, "Metaheuristics for feature selection: application to sepsis outcome prediction," in *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 1–8, Brisbane, Australia, June 2012.

[4] J. Wang, A.-R. Hedar, S. Wang, and J. Ma, "Rough set and scatter search metaheuristic based feature selection for credit scoring," *Expert Systems with Applications*, vol. 39, no. 6, pp. 6123–6128, 2012.

[5] N. Abd-Alsabour and A. Moneim, "Diversification with an ant colony system for the feature selection problem," in *Proceedings of the 2nd International Conference on Management and Artificial Intelligence (IPEDR' 12)*, vol. 35, pp. 35–39, IACSIT Press, 2012.

[6] S. Casado, J. Pacheco, and L. Núñez, "A new variable selection method for classification," *XV Jornadas de ASEPUMA y III Encuentro Internacional*, pp. 1–11, 2007.

[7] J. B. Jona and N. Nagaveni, "Ant-cuckoo colony optimization for feature selection in digital mammogram," *Pakistan Journal of Biological Sciences*, pp. 1–6, 2013.

[8] A. Unler, A. Murat, and R. B. Chinnam, "Mr2PSO: a maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification," *Information Sciences*, vol. 181, no. 20, pp. 4625–4641, 2011.

[9] D. Korycinski, M. M. Crawford, and J. W. Barnes, "Adaptive feature selection for hyperspectral data analysis," in *9th Image and Signal Processing for Remote Sensing*, vol. 5238 of *Proceedings of SPIE*, pp. 213–225, Barcelona, Spain, 2004.

[10] S. C. Yusta, "Different metaheuristic strategies to solve the feature selection problem," *Pattern Recognition Letters*, vol. 30, no. 5, pp. 525–534, 2009.

[11] A. Unler and A. Murat, "A discrete particle swarm optimization method for feature selection in binary classification problems," *European Journal of Operational Research*, vol. 206, no. 3, pp. 528–539, 2010.

[12] M. García-Torres, C. F. García López, B. Melián-Batista, A. J. Moreno-Pérez, and J. M. Moreno-Vega, "Solving feature subset selection problem by a hybrid metaheuristic," *Hybrid Metaheuristics*, pp. 59–68, 2004.

[13] S. El Ferchichi and K. Laabidi, "Genetic algorithm and tabu search for feature selection," *Studies in Informatics and Control*, vol. 18, no. 2, pp. 181–187, 2009.

[14] A. Al-Ani, "Feature subset selection using ant colony optimization," *International Journal of Information and Mathematical Sciences*, vol. 2, article 1, pp. 53–58, 2006.

[15] L. C. Molina, L. Belanche, and À. Nebot, "Feature selection algorithms: a survey and experimental evaluation," in *Proceedings of the IEEE International Conference on Data Mining, (ICDM '02)*, pp. 306–313, Maebashi, Japan, December 2002.

[16] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice Hall, Englewood Cliffs, NJ, USA, 3rd edition, 1992.

[17] M. Deriche and A. Al-Ani, "Feature selection using a mutual information based measure," in *Proceedings of the 16th International Conference on Pattern Recognition*, vol. 4, pp. 82–85, Quebec, Canada, August 2002.

[18] G. Kumar and K. Kumar, "A novel evaluation function for feature selection based upon information theory," in *Proceedings of the 24th Canadian Conference on Electrical and Computer Engineering (CCECE '11)*, pp. 395–399, Niagara Falls, NY, USA, May 2011.

[19] M. A. Hall and L. Smith, "A. Practical feature subset selection for machine learning," in *Proceedings of the Australian Computer Science Conference*, pp. 181–191, Springer, New York, NY, USA, 1998.

[20] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[21] R. Ruiz, J. C. Riquelme, and J. S. Aguilar-Ruiz, "Heuristic search over a ranking for feature selection," in *Proceedings of the 8th International Workshop on Artificial Neural Networks, (IWANN '05)*, pp. 742–749, Barcelona, Spain, June 2005.

[22] T. N. Lal, O. Chapelle, J. Western, and A. Elisseeff, "Embedded methods," *Studies in Fuzziness and Soft Computing*, vol. 207, pp. 137–165, 2006.

[23] X. S. Yang, "Swarm-based metaheuristic algorithms and no-free-lunch theorems," in *Theory and New Applications of Swarm Intelligence*, R. Parpinelli and S. Heitor Lopes, Eds., InTech, 2012.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks. Part 1*, vol. 4, pp. 1942–1948, December 1995.

[25] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," *Nature Inspired Cooperative Strategies for Optimization (NICSO '10)*, vol. 284, pp. 65–74, 2010.

[26] R. Tang, S. Fong, X. S. Yang, and S. Deb, "Wolf search algorithm with ephemeral memory," in *Proceedings of the IEEE 7th International Conference on Digital Information Management (ICDIM '12)*, pp. 165–172, August 2012.

[27] S. Fong, K. Lan, and R. Wong, "Classifying human voices by using hybrid SFX time-series pre-processing and ensemble feature selection," *Biomed Research International*, vol. 2013, Article ID 720834, 27 pages, 2013.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 123738, 8 pages
http://dx.doi.org/10.1155/2013/123738

Research Article

Advanced Harmony Search with Ant Colony Optimization for Solving the Traveling Salesman Problem

Ho-Yoeng Yun,[1] Suk-Jae Jeong,[2] and Kyung-Sup Kim[1]

[1] *Department of Industrial Information Engineering, Yonsei University, Seoul 120-749, Republic of Korea*
[2] *Department of Business School, Kwangwoon University, Seoul 139-701, Republic of Korea*

Correspondence should be addressed to Kyung-Sup Kim; kyungkim@yonsei.ac.kr

Received 27 June 2013; Revised 23 September 2013; Accepted 30 September 2013

Academic Editor: Chung-Li Tseng

We propose a novel heuristic algorithm based on the methods of advanced Harmony Search and Ant Colony Optimization (AHS-ACO) to effectively solve the Traveling Salesman Problem (TSP). The TSP, in general, is well known as an NP-complete problem, whose computational complexity increases exponentially by increasing the number of cities. In our algorithm, Ant Colony Optimization (ACO) is used to search the local optimum in the solution space, followed by the use of the Harmony Search to escape the local optimum determined by the ACO and to move towards a global optimum. Experiments were performed to validate the efficiency of our algorithm through a comparison with other algorithms and the optimum solutions presented in the TSPLIB. The results indicate that our algorithm is capable of generating the optimum solution for most instances in the TSPLIB; moreover, our algorithm found better solutions in two cases (kroB100 and pr144) when compared with the optimum solution presented in the TSPLIB.

1. Introduction

The Traveling Salesman Problem (TSP) is a typical example of an NP-complete problem of computational complexity theory and can be understood as a "Maximum Benefit with Minimum Cost" that searches for the shortest closed tour that visits each city once and only once. As is well known, the TSP belongs to a family of NP-complete problems. Generally, when solving this type of problem with integer programming (IP), determining the optimum solution is impossible because the computational time to search the solution space increases exponentially with increasing problem sizes. As a result, the general approach involves determining a near-optimal solution within a reasonable time by applying metaheuristics. In the last decade, TSP has been well studied by many metaheuristic approaches, such as Genetic Algorithm (GA), Simulated Annealing (SA), Tabu Search (TS), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Harmony Search (HS), Cuckoo Search (CS), and Firefly Algorithm (FA). Among these approaches, the general procedures of GA, SA, and TS have already been introduced in many articles [1–3]. ACO is a metaheuristic approach that is inspired by the behavior of ants searching for their food source [4–7]. PSO is originally attributed to Kennedy and Eberhart [8] and was first intended for simulating social behavior as a stylized representation of the movement of organisms in a bird flock or fish school. HS, proposed by Geem et al. [9, 10], is a metaheuristic that was inspired by the improvisation process of musicians. CS is modeled after the obligate brood parasitism of some Cuckoo species by laying their eggs in the nests of other host birds (of other species) [11]. FA is a metaheuristic algorithm that mimics the flashing behavior of fireflies [12]. Freisleben and Merz [13] realized GA by using a new mutation operator of Lin-Kernighan-Opt for finding the high-quality solution in a reasonable amount of time of the asymmetric TSP. Wang and Tian [14] introduced the improved simulated annealing (ISA), which is the integration of the basic simulated annealing (BSA) with the four vertices and three lines inequality to search the optimal Hamiltonian circuit (OHC) or near-OHC. Fiechter [15] proposed the TS for obtaining near-optimal solution of large-size TSPs. The remarkable idea of his research is that while TS seeks a high global quality of the solution, the local search inside TS performed several independent searches

without much loss of quality. Stüzle and Hoos [16] proposed max-min ant system and demonstrated that their proposed algorithm can be significantly improved in performance over the general ant system in most cases discussed of the TSP examples. Wang et al. [17] designed an advanced PSO with the concept of a swap operator and a swap sequence that exhibited good results in a small-size TSP having 14 nodes. Geem et al. [9] applied HS for solving the 20-cities TSP. They combined two operators (neighboring city-going and city-inverting operators) inside the HS to arrive at the global optimum quickly. One of two operators was able to find the closest city that will be visited next after the current city. The other is used to produce a new path by exchanging the sequence of the two nodes selected randomly in one feasible path. Ouyang et al. [18] presented the advanced CS with the "search-new-nest" and "study" operator, derived from idea of "inver-over" operator for solving the spherical TSP. Their experiments demonstrated that CS provided better solutions over GA in HA30 from TSPLIB. Kumbharana and Pandey [19] implemented FA and demonstrated that it provides better solutions than ACO, GA, and SA in most cases of the TSP examples. Many articles mentioned previously are examples of the application of only single metaheuristics or a metaheuristic with a local search.

Recently, however, to complement the weakness of single metaheuristics, a few research studies involving the hybridization of two or more heuristics have been introduced. Pang et al. proposed the combinations of PSO with Fuzzy theory for solving the TSP. In their study, Fuzzy matrices were used to represent the position and velocity of the particles in the PSO, and the symbols and operators in the original PSO formulas were redefined for transformation into the form of the Fuzzy matrices [21]. Thamilselvan and Balasubramanie [22] presented a genetic Tabu Search algorithm, a combined heuristics with the dynamic switching of the GA and the TS. The experimental results indicated that the combination has better solution over the respective individual use of the GA and the TS. Yan et al. [23] introduced a mixed heuristic algorithm to solve the TSP. In their algorithm, SA and ACO were mixed to obtain improved performance. By comparison with the TSPLIB, they determined that the mixed form is much better than (a) the original ACO and the max-min ant system in the convergence rate and (b) the SA in the probability of converging to optimal solution. Chen and Chien [24] presented the parallelized genetic ACO for solving the TSP. They demonstrated improved solutions in three cases of the TSPLIB over Chu et al. [25] with original ACO. Chen and Chien [26] proposed the combination of four metaheuristics (GA, SA, ACO, and PSO) for obtaining a better solution in the TSP. Their experiments tested the combination of four metaheuristics by using 25 datasets of the TSPLIB and demonstrated that it provided better solutions through a comparison with four articles previously published. According to a review of many articles that focused on the combinations of two or more heuristics published since 2006, the combination of HS and other heuristics for solving TSP has been little studied. Therefore, in this paper, we propose the hybridized HS and ACO to solve the TSP. In Section 4, our algorithm will be introduced in detail. The rest of this paper is organized as follows; in Section 2, we introduce the simple overview of both ACO and HS. In Section 3, we describe the advanced HS for solving the TSP, and we explain the overall procedures of the algorithm proposed in Section 4. In Section 5, experiments performed with 20 data sets of TSPLIB are described, and the results of our algorithm and others are compared in the cases of 11 instances involved in TSPLIB. Finally, the conclusion is provided in Section 6.

2. Overview of Ant Colony Optimization and Harmony Search

2.1. Ant Colony Optimization. Ant Colony Optimization (ACO), originally proposed by Dorigo, [4] is a stochastic-based metaheuristic technique that uses artificial ants to find solutions to combinatorial optimization problems. The concept of ACO is to find shorter paths from their nests to food sources. Ants deposit a chemical substance called a pheromone to enable communication among other ants. As an ant travels, it deposits a constant amount of pheromone that the other ants can follow. Each ant moves in a somewhat random fashion, but when an ant encounters a pheromone trail, it must decide whether to follow it. If the ant follows the trail, the ants own pheromone reinforces the existing trail, and the increase in pheromone increases the probability that the next ant will select the path. Therefore, the more ants that travel on a path, the more attractive the path becomes for the subsequent ants. In addition, an ant using a shorter route to a food source will return to the nest sooner. Over time, as more ants are able to complete the shorter route, pheromone accumulates more rapidly on shorter paths and longer paths are less reinforced. The evaporation of pheromone also makes less desirable routes more difficult to detect and further decreases their use. However, the continued random selection of paths by individual ants helps the colony discover alternate routes and ensures successful navigation around obstacles that interrupt a route. ACO, thus, is an algorithm that reflects the stochastic travels of ants by the probability, the evaporation, and the update of pheromone over time. ACO is composed of the state transition rule, the local updating rule, and the global updating rule. Based on the state transition rule as expressed in (1), ants move between nodes. Consider

$$s = \begin{cases} \arg\max\limits_{u \in J_k(r)} [\tau(r,u)]^{\alpha} [\eta(r,u)]^{\beta}, & \text{if } q \le q_0, \\ S, & \text{otherwise,} \end{cases} \tag{1}$$

$\tau(r,u)$ in state transition rule, s is the reciprocal of distance between nodes r and u. $J_k(r)$ means the set of nodes to which ant k in node r can visit in the next time. α, β are parameters that determine the relative importance of pheromone and distance of nodes, respectively.

Whenever ants visit their nodes through the state transition rule, pheromone is updated by the local updating rule. It can be expressed by

$$\tau(r,s) \longleftarrow (1-\rho)\,\tau(r,s) + \rho \Delta\tau(r,s). \tag{2}$$

The pheromone evaporation coefficient ρ is a decimal number in range of 0 to 1. $\Delta\tau(r,s) = \tau_0 = (n \times L_{nn})^{-1}$ is

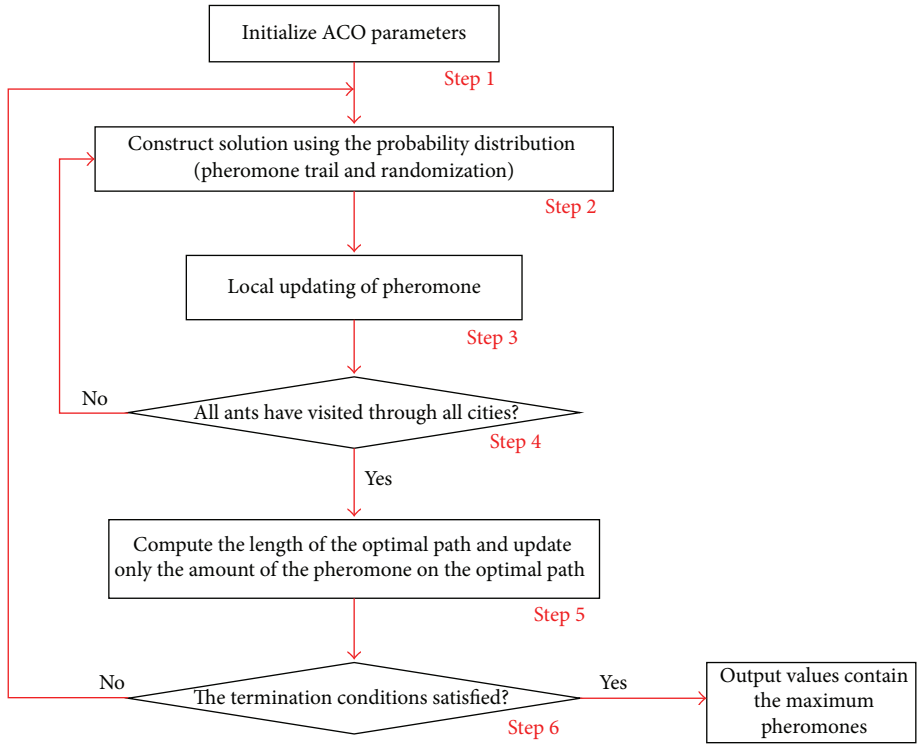


Figure 1: Flow chart of the Ant Colony Optimization.

the amount of initial pheromone. Here, n means the number of cities and L_{nn} is the cost produced by the nearest neighbor heuristic. After all ants have visited through all cities, global updating rule is performed with

$$\tau(r,s) \longleftarrow (1-\rho)\,\tau(r,s) + \rho\Delta\tau(r,s),$$
$$\Delta\tau(r,s) = \begin{cases} L_{gb}^{-1}, & \text{if } (r,s) \in \text{global best tour}, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where ρ is constant and L_{gb} is the global best tour. The general structure of ACO algorithms can be described as follows, and Figure 1 shows the flow chart of the ACO algorithm.

Step 1. Initialize the pheromone table and the ACO parameters.

Step 2. Randomly allocate ants to every node. Every ant must move to next city, depending on the probability distribution.

Step 3. The local pheromone update is performed.

Step 4. If all ants have not visited through all cities, go to Step 2.

Step 5. Compute the optimal path and global update of pheromone.

Step 6. If stopping criteria are not satisfied, go to Step 2.

2.2. Harmony Search. Harmony Search (HS) is a metaheuristic algorithm that mimics the improvisation process of music players and has been very successful in wide variety of optimization problems [9, 10]. In the HS algorithm, the fantastic harmony, the aesthetic standard, pitches of instruments, and each practice in performance process of HS indicate the global optimum, the objective function, the value of variables, and each iteration in optimization process, respectively. HS is composed of optimization operators, such as the harmony memory (HM), the harmony memory size (HMS), the harmony memory considering rate (HMCR), and the pitch adjusting rate (PAR).

HS is conducted by the following steps, and the overall flow chart of the HS algorithm is shown in Figure 2.

Step 1. Initialize the HM and the algorithm parameters.

Step 2. Improvise a new harmony from the HM. A new harmony vector is generated from the HM, based on memory consideration, pitch adjustments, and randomization. $pHMCR$ and $pPAR$ were generated randomly between 0 and 1, respectively, and each operator is selected according to the following conditions.

(i) Condition 1: $pHMCR \le HMCR$ and $pPAR > PAR$; select the memory consideration.

(ii) Condition 2: $pHMCR \le HMCR$ and $pPAR \le PAR$; select the pitch adjustments.

(iii) Condition 3: $pHMCR > HMCR$; select the randomization.

Step 3. If a new harmony is better than the worst harmony in the HM, update the HM.

Step 4. If stopping criteria are not satisfied, go to Step 2.

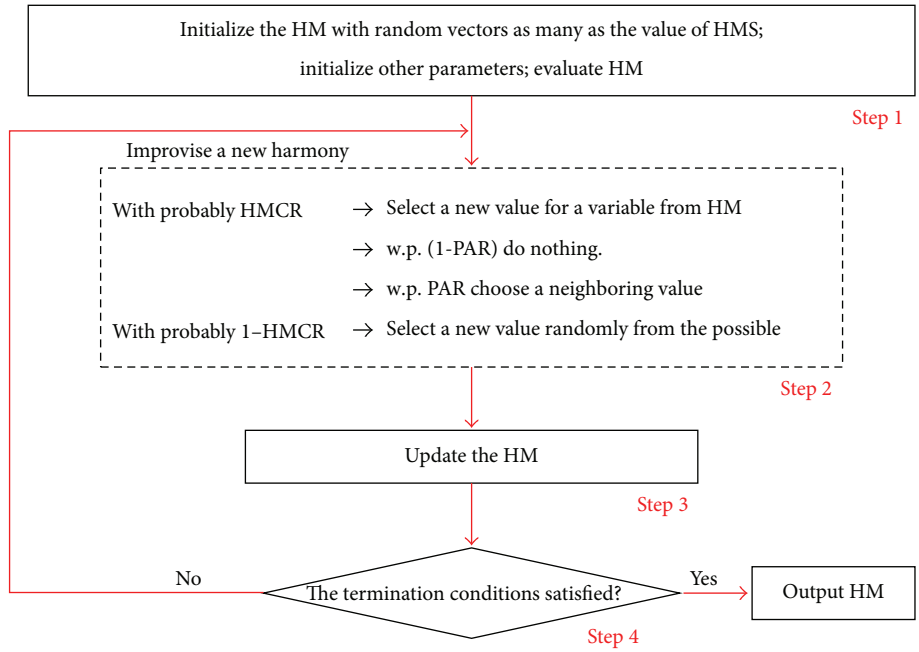


Figure 2: The flow chart for the Harmony Search [20].

3. Advanced Harmony Search for Traveling Salesman Problems

The HS algorithm exhibits good performance in solving a diverse set of problems; however, it has some drawbacks in terms of the sequential problems, such as the TSP and the vehicle routing problem. In case of sequential problems, a close positioning between the nodes implies a strong correlation. HS, however, uses a uniform probability regardless of the correlation between nodes when choosing the new value in a new harmony from the historic values stored in the same index of the existing HM. The memory consideration operator does not even function under the following case: when generating the value of a new harmony under that ith value of index is city 1, if all the values of $(i + 1)$th in the existing HM are city 1, if all the values of $(i+1)$th in the existing HM are city 1. To remedy these shortcomings, we propose the advanced HS (AHS), which includes the fitness, elite strategy, and mutation operators of the GA.

3.1. Revised Memory Consideration and Pitch Adjustments. The memory consideration operator of the original HS runs randomly from the historic values in the HM. In the advanced HS algorithm, however, the memory consideration operator is implemented by using a roulette wheel, so that the fittest index of HM has a greater chance of survival than the weaker ones. Fitness and distance are inversely related. Meanwhile, although the memory consideration operator runs a certain number of times, if the ith value and the candidate $(i + 1)$th value that are selected by the memory consideration operator are the same, the $(i + 1)$th value of a new harmony is determined by a randomization operator. In the case of satisfying condition 1 of Section 2.2, the pitch adjustment generates the $(i + 1)$th index value of a new harmony that is the closest value to the ith value from the possible range of values.

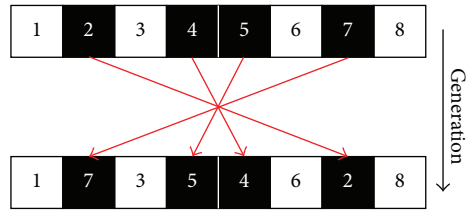


Figure 3: Operator of inversion mutation.

3.2. Elite Preserving Rule and Mutation. The HS algorithm updates in a manner that a new harmony is included in the HM and the existing worst one is excluded from the HM when a new harmony is better than the worst harmony in the existing HM. This mechanism forces to the convergence of all the elements in the HM to the same value that could be the local optimum, when it is repeated infinitely. To escape such case, we consider the inversion operator, one of all mutations of the GA. It is performed for HM that satisfies the following equations: (1 − Elite Rate) × HMS, where (1 − Elite Rate) means the rate of noting the performance of the elite strategy. Inversion mutation operator meanwhile selects a few nodes among all nodes randomly, and the nodes selected are rearranged in inverse order. As shown in example of Figure 3, the previous node (1, 2, 3, 4, 5, 6, 7, 8) is converted to new node (1, 7, 3, 5, 4, 6, 2, 8) through the inversion mutation.

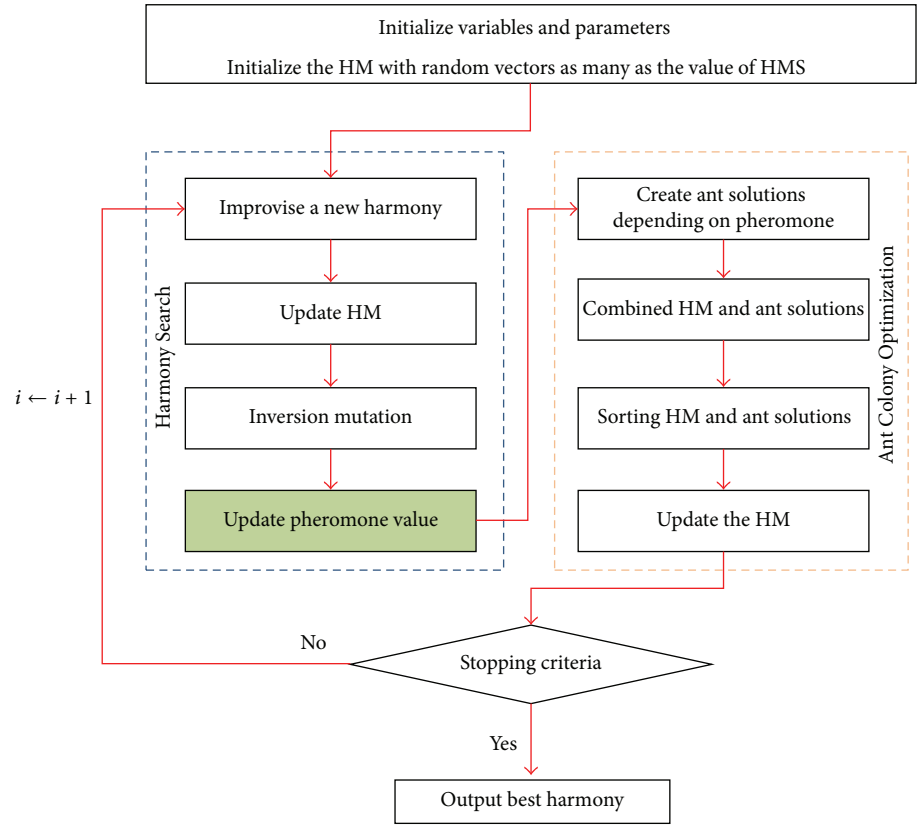


Figure 4: The flow chart for the proposed algorithm.

4. The Proposed Algorithm for the TSP

The overall procedures of our algorithm that combines the AHS and ACO algorithms are shown in Figure 4. First, we generate an initial solution randomly. A pheromone trail is updated. By using memory consideration, pitch adjustment, and randomization under each condition mentioned in Section 2.2, we create a new harmony and check whether an update occurs. When the mutation operator is implemented at a certain probability, the inversion mutation is performed to the rest, except the HM as regarded as Elite, and then the pheromone is updated. After that, ant solutions with the size of HMS are generated by using the ACO algorithm, based on the pheromone trail determined by the HS algorithm. The combined ant solution and HM are stored in a temporary memory that has twice the size of HMS, and they are sorted in ascending order by the total distance, defined as the objective function. The top 50% with higher value in the temporary memory are determined as the new HM. These procedures are repeated until the stopping criteria are satisfied. Pseudocode 1 describes the pseudocode of the proposed algorithm.

5. Experimental Results

Table 1 lists the parameter setting of the proposed algorithm. To show the performance of the proposed algorithm, we performed experiments using a computer with an Intel Core-i5 processor and 2 GB RAM and used C# as the programming language to implement the algorithm. We tested the algorithm using 20 datasets from the TSPLIB (e.g., berlin52, st70, eil76, kroA100, kroB100, kroC100, kroD100, kroE100, eil101, lin105, ch130, pr144, ch150, pr152, d198, tsp225, pr226, pr264, a280, and pr299). For the exact comparison with other algorithms and known best solutions obtained from TSPLIB, the distance between any two cities is calculated as the Euclidian distance and rounded off to 1 decimal place. Each experiment was performed using 1000 iterations and 10 runs, and the best, worst, mean, and standard deviation were recorded for each run. As seen in Table 2, among the 20 datasets tested, we found the optimum solution in 19

Table 1: Parameters settings of the proposed algorithm.

Parameters	Values
HMS	100~200
HMCR	0.9
PAR	0.4
ρ	0.05
α	0.8
β	1.0
p_m	0.01
Iteration	1000

```
Procedure: The proposed algorithm for the TSP
Begin
    Objective function f(x), x = (x_1, x_2, ..., x_d)
    Generate initial harmonics (real number arrays)
    Define harmony memory considering rate (p_HMCR), pitch adjusting rate (p_pa), mutation rate (p_m)
    Initialize the pheromone tables
    Generate initial harmony randomly and apply pheromone update
    while (not_termination)
        for i = 1: number of nodes
            Generate random number variable (rand)
            if (rand < p_HMCR)
                Generate random number variable (rand)
                if (rand < p_pa), generate the nearest city to the previous harmonic
                else choose an existing harmonic the highest fitness probability
                end if
            else generate new harmonics via randomization
            end if
        end for
        Accept the new harmonics (solutions) if better
        Generate random number variable (rand)
        if (rand < p_m) operate inversion mutation end if
        Apply the pheromone update
        Create as many cities as the HMS based pheromone using Ant Colony Optimization
        Update harmony memory and apply pheromone update
    end while
    Find the current best solutions
End
```

Pseudocode 1: The pseudo-code for the proposed algorithm (AHS-ACO).

Table 2: Results of our algorithm (AHS-ACO) for 20 TSP instances from the TSPLIB.

TSPLIB	Known best solutions	Solution				Running Time	Relative error (%)
		Best	Worst	Mean	STDEV	Second (s)	Best
berlin52	7542	7542	7542	7542	0.000	15.12	0.000
st70	675	675	677	675.375	0.812	19.45	0.000
eil76	538	538	542	540.494	1.473	22.27	0.000
kroA100	21282	21282	21378	21307.554	34.983	40.97	0.000
kroB100*	22141	**22139***	22271	22193.114	53.678	42.87	**−0.009**
kroC100	20749	20749	20868	20770	34.937	39.62	0.000
kroD100	21294	21294	21467	21338.03	63.797	41.24	0.000
kroE100	22068	22068	22117	22093.099	14.819	42.38	0.000
eil101	629	629	643	634.355	4.479	49.47	0.000
lin105	14379	14379	14541	14434.947	59.097	56.27	0.000
ch130	6110	6110	6200	6173.038	24.544	69.24	0.000
pr144*	58537	**58534***	58902	58659.283	144.807	80.27	**−0.003**
ch150	6528	6528	6586	6554.589	17.303	97.34	0.000
pr152	73682	73682	74754	73846.437	325.185	111.59	0.000
d198	15780	15780	15963	15876.892	70.497	180.27	0.000
tsp225	3919	**3859**	4013.724	3977.047	26.516	208.48	0.000
pr226	80369	80369	80882	80558.519	174.915	213.78	0.000
pr264	49135	49135	49379	49205.87	75.675	279.69	0.000
a280	2579	2579	2726	2641.61	44.91	303.36	0.000
pr299	48191	48195	49989	49121.289	577.831	367.72	0.016

Table 3: Comparison result of our algorithm with the results of Randall and Montgomery (2003) [27].

TSPLIB	Randall and Montgomery (2003) [27]			Proposed algorithm		
	Best	Mean	Worst	Best	Mean	Worst
berlin52	7547	7790	8148	**7542**	**7542**	**7542**
st70	678	687	712	**675**	**675.376**	**677**
eil76	546	555	559	**538**	**540.494**	**542**
kroA100	21373	21512	21915	**21282**	**21307.554**	**21378**
ch130	6180	6269	6407	**6110**	**6173.038**	**6200**
d198	16044	16209	16465	**15780**	**15876.892**	**15963**

Table 4: Comparison result of our algorithm with the results of Chen and Chien (2011) [26].

TSPLIB	Chen and Chien (2011) [26]			Proposed algorithm		
	Best	Mean	SD	Best	Mean	SD
berlin52	7542	7542	0.00	**7542**	**7542**	**0.00**
eil76	538	540.20	2.94	**538**	540.494	**1.473**
kroA100	21282	21370	123.36	**21282**	**21307.554**	**34.983**
kroB100	22141	22282.87	183.99	**22139**	**22193.114**	**53.678**
kroC100	20749	20878.97	158.64	**20749**	**20770**	**34.937**
kroD100	21309	21620.47	226.60	**21294**	**21338.03**	**63.797**
kroE100	22068	22183.47	103.32	**22068**	**22093.099**	**14.819**
eil101	630	635.23	3.59	**629**	**634.355**	4.479
lin105	14379	14406.37	37.28	**14379**	14434.907	59.097
ch130	6141	6205.63	43.70	**6110**	6173.038	**24.544**
ch150	6528	6563.70	22.45	**6528**	**6554.589**	**17.303**

datasets, except for pr299, indicated from the TSPLIB. In the case of kroB100 and pr144, in particular, our algorithm outperformed the known best solutions from the TSPLIB (see the asterisks of Table 2 for details).

To validate the superiority of our algorithm, we compared it with Randall and Montgomery [27] and Chen and Chien [24, 26]. Randall and Montgomery [27] proposed accumulated experience ant colony (AEAC) for using the previous experiences of the colony to guide in the choice of elements, and Chen an Chien [24, 26] solved TSP with combination of four metaheuristics having GA, SA, ACO, and PSO. Tables 3 and 4 show the comparative results with two previous researches, respectively.

6. Conclusion

In this paper, we proposed the AHS-ACO algorithm, which is a combination of the advanced Harmony Search and the Ant Colony Optimization algorithms, to solve the TSP. We modified the generic HS algorithm to produce a new HS algorithm that includes the fitness, elite strategy, and mutation operators in the GA, and we combined the ACO algorithm inside the HS algorithm to overcome the shortcomings of the HS algorithm for solving sequential problems. We performed experiments using the AHS-ACO algorithm on 20 datasets of the TSPLIB. As shown in the experimental results, we found the optimal solution obtained from the TSPLIB in almost all cases of the TSPLIB; moreover, our algorithm provided a better solution over the TSPLIB solution in the cases of kroB100 and pr144. The results of this paper indicate that the HS algorithm can be a good method, in combination with other heuristics, to solve sequential problems such as TSP, as well as many other problems.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0023236).

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, Mass, USA, 1975.

[2] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[3] F. Glover and M. Laguna, *Tabu Search*, Springer, 1997.

[4] M. Dorigo, *Learning and Nature Algorithm [Ph.D. thesis]*, Dipartimento di Electtonica, Politecnico di Milano, Milano, Italy, 1992.

[5] L. M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pp. 622–627, May 1996.

[6] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem,"

IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53–66, 1997.

[7] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.

[8] 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.

[9] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[10] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36–38, pp. 3902–3933, 2005.

[11] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, IEEE Publications, December 2009.

[12] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Lunvie Press, Frome, UK, 2008.

[13] B. Freisleben and P. Merz, "Genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pp. 616–621, May 1996.

[14] Y. Wang and D. Tian, "An improve simulated annealing algorithm for traveling salesman problem," in *Proceedings of the International Conference on Information Technology and Software Engineering*, vol. 211 of *Lecture Notes In Electrical Engineering*, pp. 525–532, 2013.

[15] C.-N. Fiechter, "A parallel tabu search algorithm for large traveling salesman problems," *Discrete Applied Mathematics*, vol. 51, no. 3, pp. 243–267, 1994.

[16] T. Stüzle and H. Hoos, "MAX-MIN Ant system and local search for the traveling salesman problem," *Reference Future Generations Computer Systems*, vol. 16, no. 8, pp. 889–914, 1997.

[17] K.-P. Wang, L. Huang, C.-G. Zhou, and W. Pang, "Particle swarm optimization for traveling salesman problem," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 1583–1585, Xi'an, China, November 2003.

[18] X. Ouyang, Y. Zhou, Q. Luo, and H. Chen, "A novel discrete cuckoo search algorithm for spherical traveling salesman problem," *Applied Mathematics & Information Sciences*, vol. 7, no. 2, pp. 777–784, 2013.

[19] S. N. Kumbharana and G. M. Pandey, "A comparative study of ACO, GA and SA for solving traveling salesman problem," *International Journal of Societal Applications of Computer Science*, vol. 2, no. 2, pp. 224–228, 2013.

[20] A. Kaveh and S. Talatahari, "Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures," *Computers and Structures*, vol. 87, no. 5-6, pp. 267–283, 2009.

[21] W. Pang, K.-P. Wang, C.-G. Zhou, and L.-J. Dong, "Fuzzy discrete particle swarm optimization for solving traveling salesman problem," in *Proceedings of the 4th International Conference on Computer and Information Technology (CIT '04)*, pp. 796–800, September 2004.

[22] R. Thamilselvan and P. Balasubramanie, "A genetic algorithm with a Tabu search(GTA) for traveling salesman problem," *International Journal of Recent Trends in Engineering*, vol. 1, no. 1, 2009.

[23] Y. Yan, X. Zhao, J. Xu, and Z. Xiao, "A mixed heuristic algorithm for traveling salesman problem," in *Proceedings of the 3rd International Conference on Multimedia Information Networking and Security (MINES '11)*, pp. 229–232, November 2011.

[24] S.-M. Chen and C.-Y. Chien, "Parallelized genetic ant colony systems for solving the traveling salesman problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3873–3883, 2011.

[25] S.-C. Chu, J. F. Roddick, and J.-S. Pan, "Ant colony system with communication strategies," *Information Sciences*, vol. 167, no. 1–4, pp. 63–76, 2004.

[26] S.-M. Chen and C.-Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14439–14450, 2011.

[27] M. Randall and J. Montgomery, "The accumulated experience Ant colony for the traveling salesman problem," *International Journal of Computational Intelligence and Applications*, vol. 03, no. 2, pp. 189–198, 2003.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 754539, 9 pages
http://dx.doi.org/10.1155/2013/754539

Research Article

Electroencephalography Signal Grouping and Feature Classification Using Harmony Search for BCI

Tae-Ju Lee, Seung-Min Park, and Kwee-Bo Sim

School of Electrical and Electronics Engineering, College of Engineering, Chung-Ang University, Heukseok-dong, Dongjak-gu, Seoul 156-756, Republic of Korea

Correspondence should be addressed to Kwee-Bo Sim; kbsim@cau.ac.kr

Received 28 June 2013; Revised 27 September 2013; Accepted 27 September 2013

Academic Editor: Zong Woo Geem

This paper presents a heuristic method for electroencephalography (EEG) grouping and feature classification using harmony search (HS) for improving the accuracy of the brain-computer interface (BCI) system. EEG, a noninvasive BCI method, uses many electrodes on the scalp, and a large number of electrodes make the resulting analysis difficult. In addition, traditional EEG analysis cannot handle multiple stimuli. On the other hand, the classification method using the EEG signal has a low accuracy. To solve these problems, we use a heuristic approach to reduce the complexities in multichannel problems and classification. In this study, we build a group of stimuli using the HS algorithm. Then, the features from common spatial patterns are classified by the HS classifier. To confirm the proposed method, we perform experiments using 64-channel EEG equipment. The subjects are subjected to three kinds of stimuli: audio, visual, and motion. Each stimulus is applied alone or in combination with the others. The acquired signals are processed by the proposed method. The classification results in an accuracy of approximately 63%. We conclude that the heuristic approach using the HS algorithm on the BCI is beneficial for EEG signal analysis.

1. Introduction

The brain-computer interface (BCI) is a system that connects a human brain and a machine. Without using an external control device, humans would be able to control devices such as brain-controlled mobile robots, wheelchairs, and robot manipulators [1–3] with their thoughts. Therefore, we greatly anticipate that the BCI system will become a technology not only for the average person but also for the disabled including people suffering from a neurodegenerative disorder such as amyotrophic lateral sclerosis and brainstem stroke [4]. To implement the BCI, various signal acquisition methods have been proposed. Among the various BCI systems, the electroencephalography (EEG)-type BCI is promising technology. The BCI system based on EEG does not require surgical operation and is considered as the ultimate goal of the systems for practical use because of its relative convenience [5]. Furthermore, the EEG signal has a higher temporal resolution than functional magnetic resonance imaging (fMRI) or functional near-infrared spectroscopy (fNIRS) signals. This characteristic provides the detailed states of brain activity in a very short time. However, several restrictions exist in the use of the EEG signals in the BCI system. Although the EEG signal has a high temporal resolution, it has very poor spatial resolution. If we want to observe the details of the brain state in a small area, we have to use many electrodes on the scalp, which will result in channel increment; that is, channel increment is an ineluctable requirement and induces a "curse of dimensionality."

Similar to other systems, the BCI system consists of three stages: sensing, signal processing, and applications. The brain-wave signal from a sensor is used for the application according to the class of the signal. The curse of dimension causes a problem in signal processing, which adds more complexity, increases the computational time, and complicates the classification. To avoid a high-dimensional computational problem resulting from the use of many channels, existing research suggests two primary methods: the multimodal integration method [6, 7] and the channel-selection method. The first method, which combines several signals, requires the solution to several limitations. fMRI or fNIRS requires

a larger space for the equipment, and building the entire system is expensive [8]. Further, the multimodal experimental method can be potentially contaminated [9]. On the other hand, the channel-selection method is a preferable and convenient method although some data are dropped. This method restricts the channels using the brain area or mathematical models [10–12]. However, this method focuses only on one stimulus at a time. Therefore, it cannot handle simultaneous stimuli in the human brain. The BCI system that employs the EEG signal for actual use overcomes the disadvantage of the curse of dimension.

In the BCI system, the performance is dependent on feature extraction and classification. From the raw-data signal, we can extract the EEG features. In this process, the common spatial pattern (CSP) algorithm is commonly used as a feature extraction algorithm [13, 14]. In the classification, even though many approaches have been attempted and the support vector machine (SVM) is preferred [15–17], a dominant algorithm is absent. The linear discriminant analysis and the original SVM algorithm are linear classifications, and their performance in nonlinear EEG signal analysis is poor. The SVM algorithm with a kernel function is not only redundant when the dimension becomes higher but also difficult in unsupervised cases. The artificial neural network is powerful and can classify nonlinear data; however, it suffers from a weakness because the length of time needed for learning cannot be approximated. Thus, a new approach to BCI feature classification is needed.

The present paper presents a solution to the two problems of conventional channel selection and feature classification using the harmony search (HS) algorithm, a music-inspired metaheuristic algorithm. The metaheuristic algorithm finds the solution using rules and randomness; thus, it does not require too many computational resources [18]. In nondeterministic polynomial-time hard (NP-hard) problems, we can solve the problem using the heuristic algorithm faster than that using the exact computational method. The HS algorithm, one of the nature-inspired heuristic algorithms, possesses advantages that are not available in the other heuristic algorithms. HS can maintain diversity; thus, the search solution effectively and easily escapes from the local optimum [19]. In addition, the HS algorithm is easy to use, has a simple structure, and requires only two parameters. By using the HS algorithm, we can determine the group of EEG signal channels to eliminate the restriction of the curse of dimension in signal processing and develop a classification for BCI system applications. In Section 2, we present the theoretical background of the HS and feature extraction algorithms. In Section 3, we show the proposed methods. Sections 4 and 5 present the experiment and analysis of the channel grouping and classification. We conclude this paper in Section 6.

2. Theoretical Background

2.1. Harmony Search Algorithm. As the name suggests, the HS algorithm originates from musical harmony [20]. If we assume that the variables are the instruments, the pitches of the instruments are the values of the variables. Similar to the orchestra player who determines the harmony to create a good balance, the HS algorithm searches the best state of a given problem defined by an objective function. The HS algorithm has a harmony memory (HM) that saves the previous results, and its structure is described in a matrix form:

$$\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_N^2 \\ \vdots & \vdots & & \vdots \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \cdots & x_N^{\mathrm{HMS}} \end{bmatrix}, \tag{1}$$

where x_n^i is the selected solution of the nth variable, HMS is the HM size, and N is the number of variables. The HM obtains the latest best result; thus, the HM members change when the result of the new value is better than the worst result in the HM. The new value is selected in three ways: selected by HM, randomly selected from the range of possible solutions, or selected by a local area search called the pitch adjustment. The selection by the HM is performed using the HM consideration rate (HMCR). The pitch adjustment, a mutation of the HM member, also has an occurrence rate, which is called the pitch adjustment rate (PAR). The random search and the pitch adjustment prevent premature convergence and maintain the diversity of the solution. These processes are represented as follows:

$$x_n^{\mathrm{New}} \longleftarrow \begin{cases} x_n \in \{X_n, X \text{ is possible solution}\} & \text{w.p. } R_{\mathrm{Random}} \\ x_n \in \{x_n^1,\ x_n^2, \ldots, x_n^{\mathrm{HMS}}\} & \text{w.p. } R_{\mathrm{Memory}} \\ x_n^i \pm m & \text{w.p. } R_{\mathrm{pitch}}, \end{cases} \tag{2}$$

where m is the pitch adjustment size, and R_{Random}, R_{Memory}, and R_{pitch} are the probability of the random search, HMCR, and PAR, respectively.

2.2. Parameter-Setting-Free Harmony Search. In the basic HS algorithm principle, we use several parameters, namely, HMS, HMCR, and PAR. Generally, the HMS value ranges from 10 to 100. Although this value depends on the iteration number, the result is not sensitive to the HMS. In contrast, the HMCR and PAR control the result of the HS algorithm. To select a proper parameter value, some investigations have been conducted [21, 22]. The parameter-setting-free (PSF) HS proposed by Geem and Sim [23] is one of the parameter control methods for automatic selection of the parameters. This method uses one more memory to represent the method that generates the HM values. The memory is the same as that of the HM. Then, we obtain the new parameter values on the basis of the following equations:

$$\mathrm{HMCR}_n = \frac{\text{Number of } x_n \text{ made by ``random selection''}}{\mathrm{HMS}}$$

$$\mathrm{PAR}_n = \frac{\text{Number of } x_n \text{ made by ``pitch adjustment''}}{\mathrm{HMS}}. \tag{3}$$

In each iteration time, the PSF HS calculates the HMCR and PAR. Then, we obtain the proper adaptive parameters. Using this PSF technique, the value to be initialized is only the HMS.

2.3. Common Spatial Patterns. The CSP algorithm is one of the useful methods for feature extraction of the EEG signals. For two different classes c_1 and c_2, we obtain multiple spatial filters that maximize the variance of one class and minimize that of the other class. The spatial filter matrix W is obtained by solving the following equation [24]:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}\in R^N} \left\{ \frac{\mathbf{w}^{\mathrm{T}} R_{\mathbf{s}|c_1} \mathbf{w}}{\mathbf{w}^{\mathrm{T}} R_{\mathbf{s}|c_2} \mathbf{w}} \right\}, \tag{4}$$

where $R_{\mathbf{s}|c_1}$ and $R_{\mathbf{s}|c_2}$ are the covariance matrices of signal $\mathbf{s}$ given in each class. The W matrix consists of L eigenvectors of $\mathbf{w}^*$. Thus, the final goal of the CSP is to obtain the linear transform $\widehat{\mathbf{s}}$, which is represented by the following equation:

$$\widehat{\mathbf{s}} = W^{\mathbf{T}} \mathbf{s}. \tag{5}$$

3. Methodology

3.1. Mean Square Error and Euclidean Matrices. To derive the objective function, the mean square error (MSE) and the Euclidean distances were used. The MSE matrix was used for grouping the EEG channels. A low MSE means that the two signals are almost the same. Moreover, if the MSE is zero, the two signals are the same. Therefore, the group with a minimum MSE has similar characteristics and can be regarded to have the same reaction for the same stimulus. For the grouping of the signals, we first normalized the EEG data acquired from 64 channels. We divided the EEG signals with the maximum values in each channel. Then, the elements in the jth row and kth column of the MSE matrix were represented as follows (here, the diagonal members of the MSE matrix were zero):

$$\mathrm{MSE}_{j,k} = \frac{1}{\mathrm{T}} \sum_t \left\{ s_j(t) - s_k(t) \right\}^2 \quad \text{where } j,k = 1,2,\ldots,64. \tag{6}$$

In (6), $s_j(t)$ and $s_k(t)$ are the normalized signal vectors of the jth and kth channels at time t, respectively.

For the feature extraction, we used the Euclidean distance matrix. Similar to grouping, we assumed that the same class members were located close to one another. From this assumption, we determined the group with the shortest overall distance. After feature extraction was completed, the features were expressed in a vector space. Then, the (j,k) elements of the Euclidean distance matrix were represented as follows (the diagonal of the MSE matrix is zero, and s^* represents the mean feature vector in the L-D space):

$$\mathrm{Dist}_{j,k} = \left\| s_j^* - s_k^* \right\|_2 \quad \text{where } j,k = 1,2,\ldots,L. \tag{7}$$

3.2. Minimum-Seeking Harmony Search. To represent the member of a group or class, we used a binary switch vector (BSV), which indicates which channel or vector was selected. If the jth member of the BSV is one, the jth channel or vector is included in the group or class. Therefore, we determine the BSV values that generate the minimum result of the objective function using the HS algorithm. The objective function used in this study is the following:

$$\arg\min_{D,\widetilde{s^*},h} \frac{1+D}{\widetilde{s^*}^h}, \tag{8}$$

$$D = \frac{1}{2} \sum_j \sum_k d_{jk}. \tag{9}$$

In (8), $\widetilde{s^*}$ is the channel number or feature selected by the BSV, and h is a constant determined by $\widetilde{s^*}$. The numeral "1" in the numerator is used to prevent the value of (8) from becoming zero. D is the summation of the entire MSE or the distances selected by the BSV. In (9), d_{jk} is the selected member of the MSE matrix or the Euclidean distance matrix selected by the BSV.

BSV contains the information in which the data, especially the channels or features, are selected. To apply the BSV to the HS algorithm, we can use the binary HS algorithm [25]. However, the binary HS algorithm has difficulty in applying pitch adjustment and requires more memory than the basic HS algorithm. This characteristic affects the speed performance and accuracy of the binary HS algorithm. Thus, in the HS, we converted each r binary number in the BSV to a 2^r-radix. Then, the solution ranged from 0 to $2^r - 1$, and the pitch adjustment size m was r. In this study, we used $r = 4$. When the value was above the range of the pitch adjustment, we used the circular method.

The value of h was determined by the following process: when the value of $\widetilde{s^*}$ is from 0 to $\widetilde{S^*}$, the objective function is represented as follows:

$$\text{Objective function} = \begin{cases} \dfrac{1}{0^h} = \infty, & \text{if } \widetilde{s^*} = 0 \\ \dfrac{1+0}{1^h} = 1, & \text{if } \widetilde{s^*} = 1 \\ \dfrac{1+D}{\widetilde{s^*}^h}, & \text{if } 1 < \widetilde{s^*} < \widetilde{S^*}. \end{cases} \tag{10}$$

We assumed that the optimal value of $\widetilde{s^*}$ was $\widehat{s}$, and the value of the objective function with $\widehat{s}$ was represented as $f_{\widehat{s}}$. Then, if $\widetilde{s^*} = \widehat{s} - 1$, the objective function satisfies the following equation:

$$\begin{aligned} \frac{1+D_{\widehat{s}-1}}{(\widehat{s}-1)^h} &= \frac{1+D_{\widehat{s}} - \sum_j d_{jk}}{\widehat{s}^h} \left(\frac{\widehat{s}}{\widehat{s}-1} \right)^h \\ &= \left(f_{\widehat{s}} - \frac{\sum_j d_{jk}}{\widehat{s}^h} \right) \left(1 + \frac{1}{\widehat{s}-1} \right)^h. \end{aligned} \tag{11}$$

From the definition of $\widehat{s}$, the following relationship should be satisfied:

$$\left| f_{\widehat{s}-1} - f_{\widehat{s}} \right| > 0. \tag{12}$$

If $\widetilde{s^*} = \hat{s} + 1$, the objective function satisfies the following equation:

$$\frac{1 + D_{\hat{s}+1}}{(\hat{s}+1)^h} = \frac{1 + D_{\hat{s}} + \sum_j d_{jk}}{\hat{s}^h}\left(\frac{\hat{s}}{\hat{s}+1}\right)^h = \left(f_{\hat{s}} + \frac{\sum_j d_{jk}}{\hat{s}^h}\right)\left(1 - \frac{1}{\hat{s}+1}\right)^h. \tag{13}$$

From the definition of $\hat{s}$, the following relationship should also be satisfied:

$$\left|f_{\hat{s}+1} - f_{\hat{s}}\right| > 0. \tag{14}$$

From (11) and (12), we obtain the following relationship:

$$f_{\hat{s}} > \frac{(1 + (1/\hat{s} - 1))^h \left(\sum_j d_{jk}/\hat{s}\right)}{(1 + (1/\hat{s} - 1))^h - 1}. \tag{15}$$

From (10), $f_{\hat{s}} \leq 1$. Then, (15) is transformed into the following relationship:

$$\hat{s}^h - (\hat{s} - 1)^h > \sum_j d_{jk}. \tag{16}$$

The same process can be applied to (13) and (14). From (13), (14), and (16), we obtain the following:

$$\sum_{j=1}^{\widetilde{S^*}} d_{jk} + (\hat{s} - 1)^h < \hat{s}^h < \sum_{j=1}^{\widetilde{S^*}+1} d_{jk} + (\hat{s} + 1)^h. \tag{17}$$

Equation (17) should be satisfied for all k values. Therefore, $\sum_{j=1}^{\widetilde{S^*}} d_{jk}$ and $\sum_{j=1}^{\widetilde{S^*}+1} d_{jk}$ can be, respectively, converted to the maximum summation of d_{jk}, which has a value of one in the BSV (d_{maxone}), and the minimum summation of d_{jk}, which has a value of zero in the BSV (d_{minzero}). Equation (17) is rewritten as follows:

$$d_{\text{maxone}} + (\hat{s} - 1)^h < \hat{s}^h < d_{\text{minzero}} + (\hat{s} + 1)^h. \tag{18}$$

In this equation, d_{maxone} and d_{minzero} are constant, and $\hat{s}$ is determined by the HS process. Therefore, the value of h should satisfy (18).

4. Experiment

In the experiments, we used a commercial product, the Neuroscan EEG signal acquisition device (Neuroscan system, Compumedics, USA), to acquire the EEG signals. This device consists of the Synamps2 and Stim2 hardware, and the software for acquisition and preprocessing is Curry 7. We used an EEG cap with 64 electrodes for the EEG signals and five additional electrodes for noise reduction. The subjects were three healthy men in their mid-twenties. The subjects were selected without using any selection criteria to ensure generality of the results. All subjects had undergone one or more brain-wave experiments not related to this study. The subjects were fully aware of the entire experimental process, risk, and objective of the experiment. The entire experimental process was conducted under the experimental ethics and experimental safety regulations of the Chung-Ang University. The subjects wore EEG caps on their heads and sat in a comfortable position in front of the monitor. Electrodes were placed in accordance with the international 10–20 system [26]. The sampling rate of the EEG signals was 250 Hz, and we used a band-pass filter to obtain signals from 8 to 70 Hz. Commonly, a low-frequency signal below 8 Hz contains biological noise such as blinking eyes and heartbeat. The following Figure 1 shows the location of the electrodes. The actual electrodes were placed on the human scalp so that the real position of the electrodes represented an arc. In this study, however, we installed the electrodes in a line for ease in understanding the position of the electrodes.

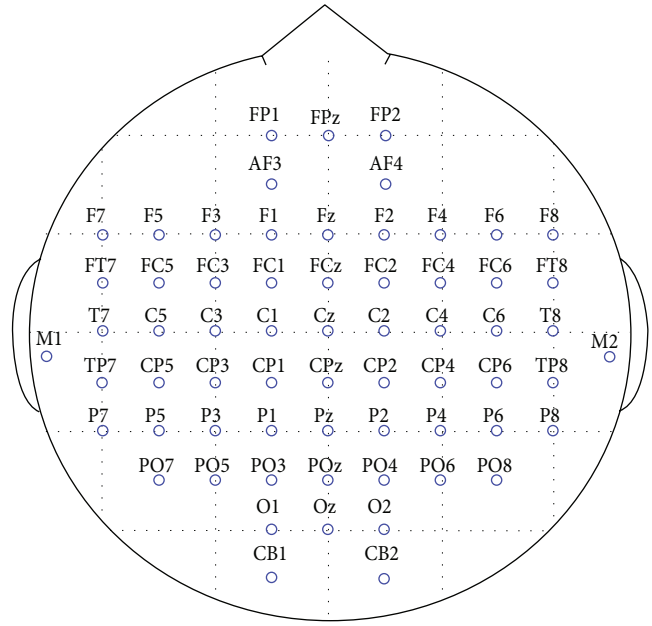


Figure 1: Electrode placement in accordance with 64CH international 10–20 system.

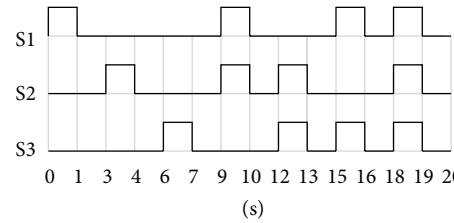


Figure 2: Timing chart of the stimuli in one session.

The subjects received three types of stimuli without feedback. To reduce noise, the subject movements were restricted. The experiment was repeated 10 times per subject. One experiment involved six sessions, and one session consisted of seven epochs. The order of the epochs was random. The same epoch was repeated 60 times, and each stimulus was repeatedly administered 240 times. The nonstationary

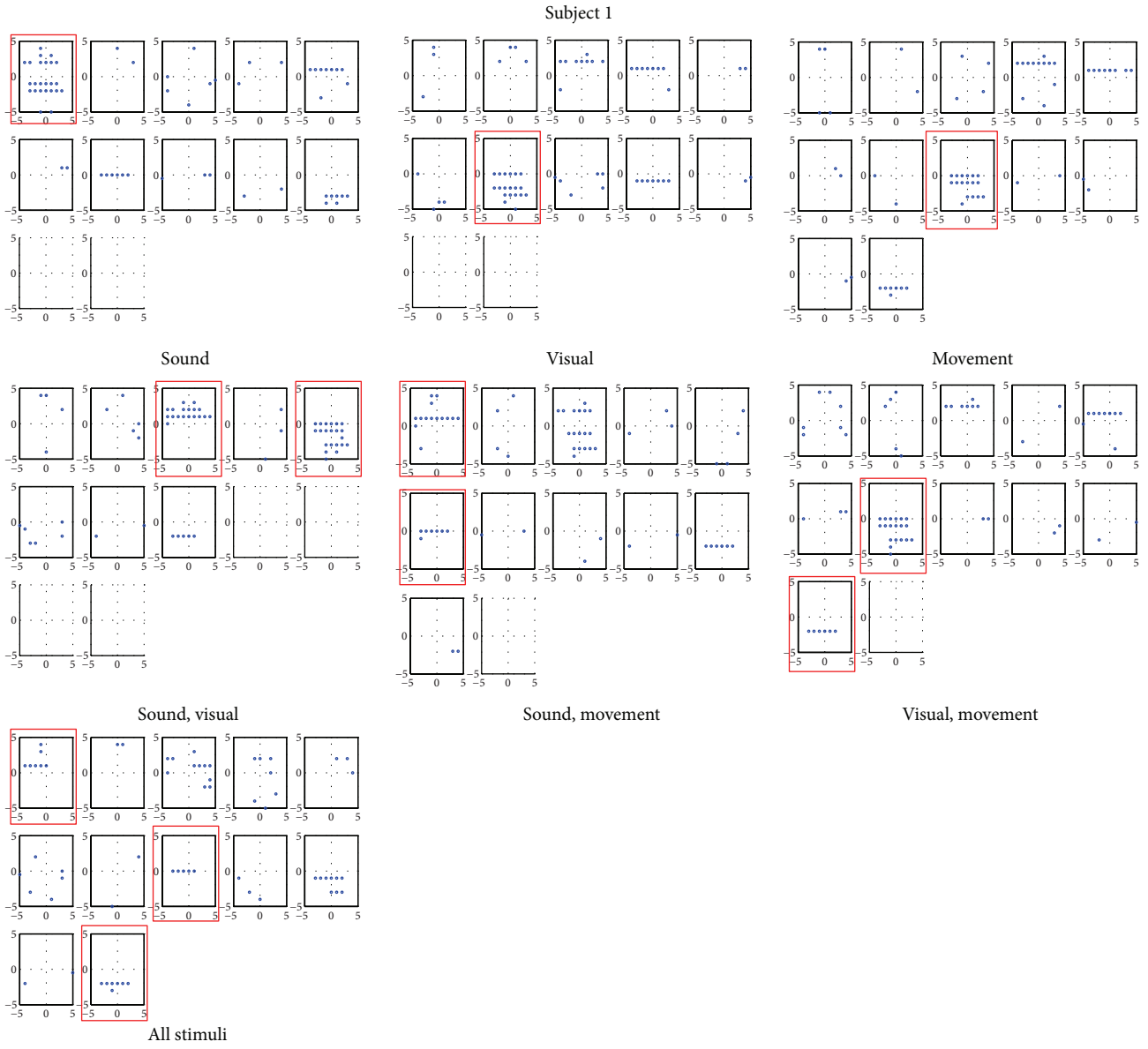


Figure 3: Results of grouping for Subject 1.

characteristic of the EEG signals allowed us to obtain results from a number of trials despite the small number of subjects.

We used three types of stimuli in this study: audio, visual, and finger movement. Each stimulus was given in 1 s. A monitor, an earphone, and a button pad were used to introduce the stimuli, which were given in rotation. The timing of the stimuli in one session is presented in the following timing chart, Figure 2.

S1, S2, and S3 represent the three stimuli, and their order was randomly changed. The resting time between stimuli was 2 s, and the entire session lasted for 20 s. First, one stimulus was administered three times. Subsequently, two stimuli were concurrently administered. At the final, three stimuli were given at once. At 300 ms before the motion stimulus was given, a white cross mark appeared on the black screen. The audio stimulus was a 1000-Hz stereo pure tone made by Stim2. The monitor screen was black when the audio stimulus was administered, with the earphone controlled by a specific amplifier for the least delay stimulus. The visual stimulus was administered in 1 s when the monitor screen was green, and the other stimuli were controlled so that the unrelated channels were unaffected. The motion stimulus was quite different from the other stimuli because the subject made a physical movement by pressing the button pad using their fingers. The motion image was difficult to notice in the experiments and affected the concentration of the subject. Therefore, we used real movements in this stimulus even though noise from the movements was present. To reduce the noise, the subject used only one finger and kept the other body parts still. The motion stimulus indicates a white cross

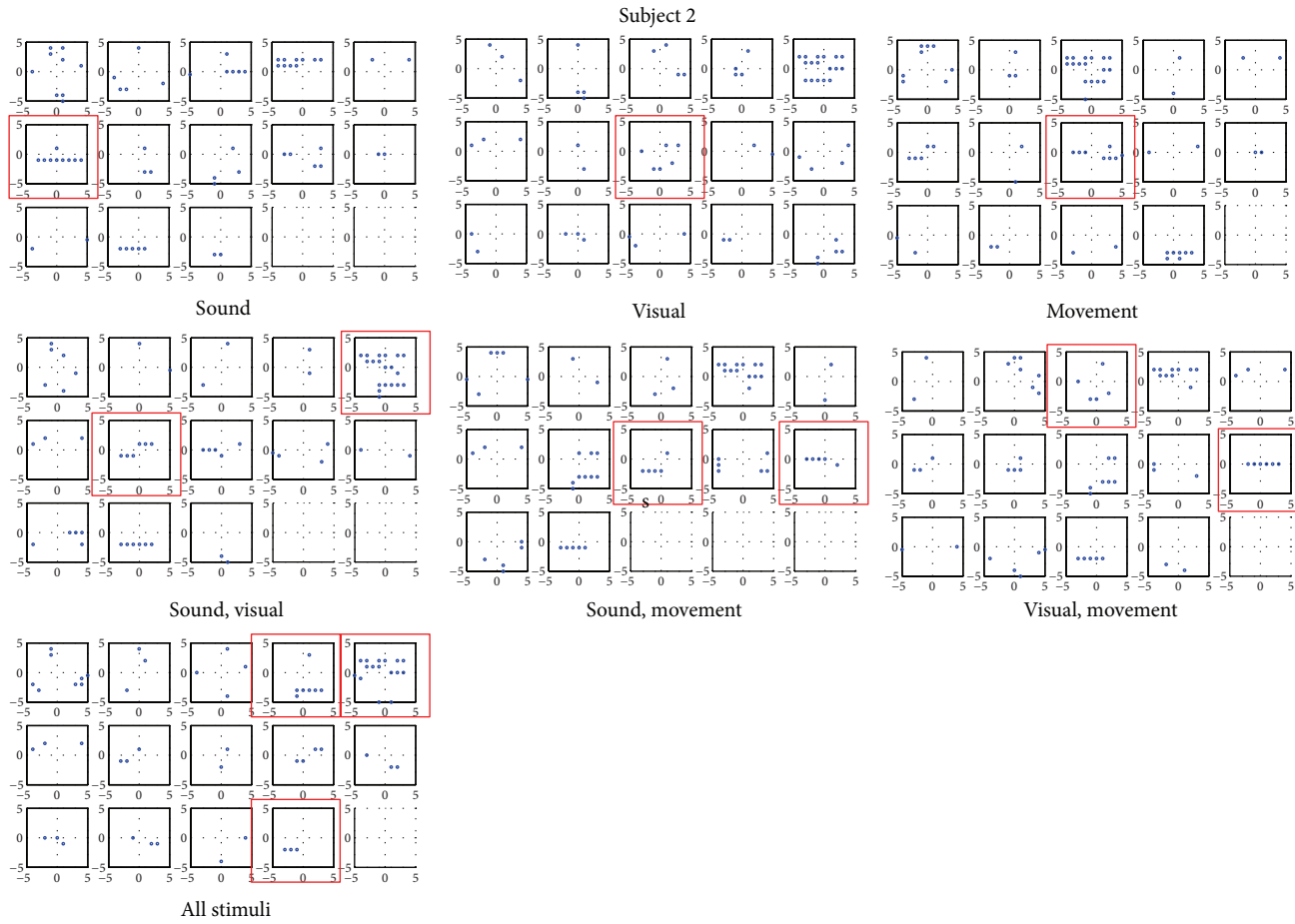


Figure 4: Results of grouping for Subject 2.

symbol on the screen to prepare the subject, and a white sentence "press button" appeared on the screen to signal to the subject to push the button pad.

4.1. Channel Grouping. For the channel grouping, we used the PSF HS algorithm and the MSE method for the objective function. In the previous research, we used the binary HS and the correlated coefficient. However, the previous method exhibited poor performance in terms of computational time. In this study, we used the proposed method to improve the speed and accuracy. The number of iterations of the HS and HMS used in this study was 5000 and 20, respectively. The number of trials for one epoch of the HS algorithm was 10. We identified the groups by determining the number of times each channel was grouped with the other channels. All 30 experiments for the three subjects were summarized in three results in Section 5.1.

4.2. Feature Classification. In the classification, we used the HS algorithm with PSF where the HMS was 20 and the number of iterations was 50000. The results of channel grouping which dimension of the signal data decreased were used for classification. The CSP algorithm was used for feature extraction. To match the number of channels, we selected a minimum number of channels among the stimuli. In the CSP procedure, we used the data from the given two stimuli and determined the eigenvectors of the filtered signal. The data were classified without using any training method.

5. Results and Analysis

5.1. Channel Grouping. The results of the channel grouping are shown in the following Figures 3, 4, and 5 and Table 1. We searched the group by comparing all cases. The results for subject 1 exhibited were distinct. The audio stimulus activated the front of the brain, the visual stimulus activated the backside of the brain, and the motion stimulus activated the middle of the brain. Generally, the functions of the brain are known to be specialized by the location. The area starting with "C" includes the vicinity from the Cz channel to the C6 channel, which is also known as the motor cortex. The visual stimulus activated the visual cortex and the back of the brain. A similar tendency could be observed in the other subjects. All subjects showed the relationship between the motion stimulus and the "C" area. Although slight differences existed, the results agreed with those of the existing research, which also explained the visual and audio stimuli. Therefore,

Figure 5: Results of grouping for Subject 3.

Table 1: Channel group results.

Subject	Stimuli	Channel name
Subject 1	Audio	FP1, AF3, FT7, FC5, FC3, FC1, FCz
	Visual	P5, P3, P1, Pz, P2, P4, PO4
	Motions	C5, C3, C1, Cz, C2
Subject 2	Audio	P5, P3, P1, Pz
	Visual	PO7, PO5, PO3, POz, O1
	Motions	C5, C3, C1, Cz, C2, CP2
Subject 3	Audio	F7, F5, F3, F1, Fz, F2, F4, AF4
	Visual	PO3, POz, PO4, PO6, PO8, O1, O2, CB1
	Motions	C5, C3, C1, Cz

we conclude that the results of the grouping are related to the given stimuli.

Furthermore, the similar group made by a single stimulus also could be found in the groups subjected to two or three stimuli including a previous single stimulus. Therefore, we can analyze the EEG and the brain by separated stimuli. Additionally, the number of channels used in the analysis was reduced by the proposed method. These characteristics can make the EEG analysis easier.

However, in some cases, determining the relationship between the stimulus and the group was difficult. This problem implied two limitations of this work. First is the difficulty in strictly controlling the experimental conditions including the state of the subject. A human brain constantly and unconsciously receives stimuli. If an unwanted stimulus is administered, the accuracy would worsen. The second is the problem resulting from averaging the signals. EEG signals have a high temporal resolution, and the data sampling rate used in this study was 250 Hz. However, the grouping method

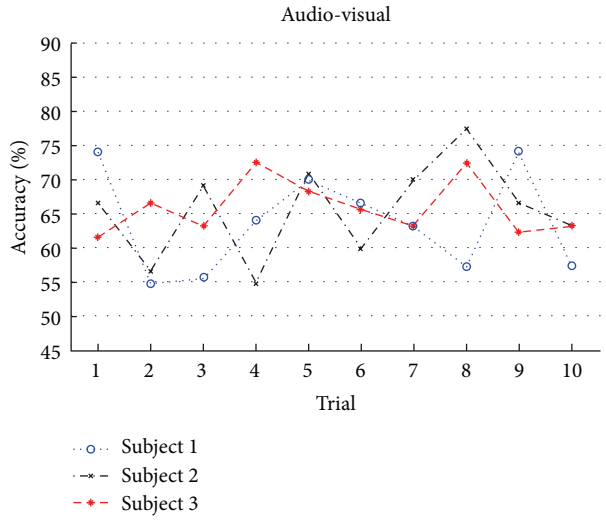


Figure 6: Classification accuracy of the audio and visual stimuli.

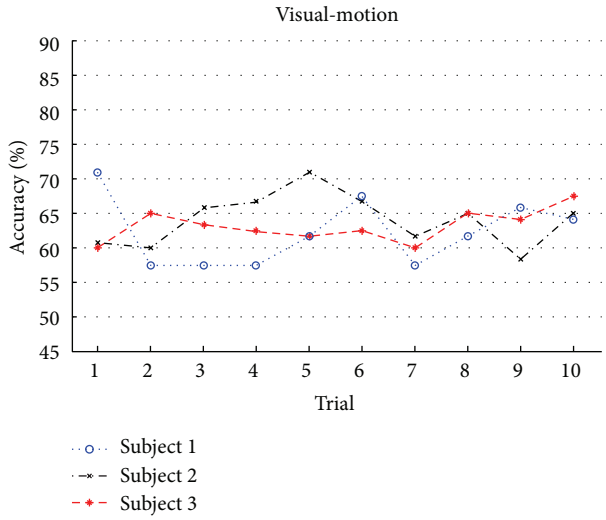


Figure 7: Classification accuracy of the visual and motion stimuli.

focused on the summation of the MSEs in 1 s. Therefore, the advantage of EEG was not well presented.

5.2. Feature Classification. We used the HS algorithm for classification. The classification in the feature space was attempted 10 times for 120 features. Figures 6, 7, and 8 show the results of the classification for two stimuli.

For Subject 1, the best classification result was 74.17%, and the worst classification result was 55%. The average accuracy rate was 62.83%. For Subject 2, the best classification result was 77.50%, and the worst classification result was 55%. The average accuracy rate was 64.61%. For Subject 3, the best classification result was 72.50%, and the worst classification result was 50.83%. The average accuracy rate was 62.53%. The HS algorithm is a heuristic algorithm; thus, the results had an accuracy-rate range. However, without any learning method or prior information, the proposed method

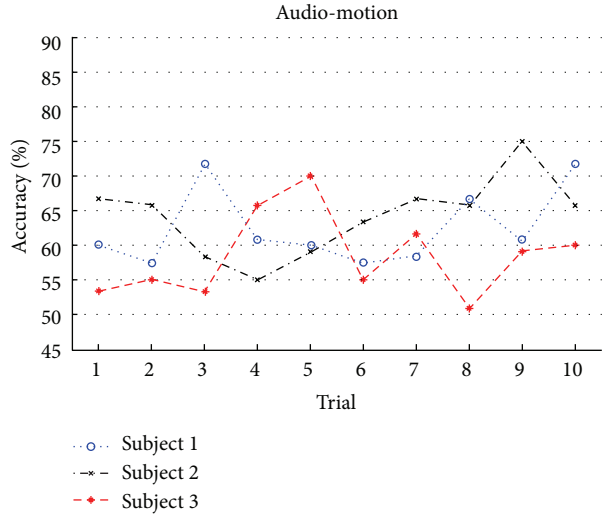


Figure 8: Classification accuracy of the audio and motion stimuli.

exceeded the decision boundary of the two classes by a rate of approximately 63%. Furthermore, the classified data were nonlinear high-dimensional data.

6. Conclusion

In this study, we have built a group to easily analyze EEG signals and proposed an EEG signal classification method based on HS. Signal processing and classification are important, as they determine the accuracy of the entire system. Therefore, we focused on solving the problems of a BCI system in signal processing and classification. First, we employed EEG signal grouping using the HS algorithm to reduce the channel complexity. From the EEG signal grouping, we did not only reduce the channel used in the analysis but also separately analyzed the signal of each stimulus. Therefore, the computational time and the complexity of the data were reduced. The proposed method could be the smartest solution for EEG signal analysis for problems with large data and dimension. Next, we classified the EEG signals using HS. We proposed a novel classification method with a metaheuristic algorithm for nonlinear data such as the EEG signals. The proposed method also focused on unsupervised classification; thus, we could classify the data without training or prior information. We believe that the proposed two methods could be helpful in enhancing the BCI implementation and the EEG analysis field.

Acknowledgment

This work was supported by the Mid-Career Researcher Program through an NRF Grant funded by the MEST (no. 2012-0008726).

References

[1] L. Bi, X. Fan, and Y. Liu, "EEG-based brain-controlled mobile robots: a survey," *IEEE Transactions on Human-Machine Systems*, vol. 43, no. 2, pp. 161–176, 2013.

[2] I. Iturrate, J. M. Antelis, A. Kübler, and J. Minguez, "A noninvasive brain-actuated wheelchair based on a P300 neurophysiological protocol and automated navigation," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 614–627, 2009.

[3] A. Frisoli, C. Loconsole, D. Leonardis et al., "A new gaze-BCI-driven control of an upper limb exoskeleton for rehabilitation in real-world tasks," *IEEE Transactions on Systems, Man, and Cybernetics C*, vol. 42, no. 6, pp. 1169–1179, 2012.

[4] J. R. Wolpaw, "Brain-computer interface research comes of age: traditional assumptions meet emerging realities," *Journal of Motor Behavior*, vol. 42, no. 6, pp. 351–353, 2010.

[5] E. E. Fetz, "Real-time control of a robotic arm by neuronal ensembles," *Nature Neuroscience*, vol. 2, no. 7, pp. 583–584, 1999.

[6] P. LeVan, J. Maclaren, M. Herbst, R. Sostheim, M. Zaitsev, and J. Hennig, "Ballistocardiographic artifact removal from simultaneous EEG-fMRI using an optical motion-tracking system," *NeuroImage*, vol. 75, pp. 1–11, 2013.

[7] A. Ishikawa, H. Udagawa, Y. Masuda, S. Kohno, T. Amita, and Y. Inoue, "Development of double density whole brain fNIRS with EEG system for brain machine interface," in *Proceedings of the 33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS '11)*, pp. 6118–6122, September 2011.

[8] O. A. P. Sosa, Y. Quijano, M. Doniz, and J. E. Chong-Quero, "BCI: a historical analysis and technology comparison," in *Proceedings of the Pan American Health Care Exchanges Conference (PAHCE '11)*, pp. 205–209, Rio de Janeiro, Brazil, April 2011.

[9] Y. O. Halchenko, S. J. Hanson, and B. A. Pearlmutter, "Multimodal integration: FMRI, MRI, EEG, MEG," in *Advanced Image Processing in Magnetic Resonance Imaging,*, CRC Press, Boca Raton, Fla, USA, 2005.

[10] T. N. Lal, M. Schröder, T. Hinterberger et al., "Support vector channel selection in BCI," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1003–1010, 2004.

[11] J. Meng, G. Huang, D. Zhang, and X. Zhu, "Optimizing spatial spectral patterns jointly with channel configuration for brain-computer interface," *Neurocomputing*, vol. 104, pp. 115–126, 2013.

[12] C. Brunner, M. Naeem, and G. Pfurtscheller, "Dimensionality reduction and channel selection of motor imagery electroencephalographic data," *Computational Intelligence and Neuroscience*, vol. 2009, Article ID 537504, 8 pages, 2009.

[13] H. Lu, H.-L. Eng, C. Guan, K. N. Plataniotis, and A. N. Venetsanopoulos, "Regularized common spatial pattern with aggregation for EEG Classification in small-sample setting," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 12, pp. 2936–2946, 2010.

[14] B. Wang, C. M. Wong, F. Wan, P. U. Mak, P.-I. Mak, and M. I. Vai, "Trial pruning based on genetic algorithm for single-trial EEG classification," *Computers and Electrical Engineering*, vol. 38, no. 1, pp. 35–44, 2012.

[15] B. Blankertz, S. Lemm, M. Treder, S. Haufe, and K.-R. Müller, "Single-trial analysis and classification of ERP components: a tutorial," *NeuroImage*, vol. 56, no. 2, pp. 814–825, 2011.

[16] S. Siuly and Y. Li, "Improving the separability of motor imagery EEG signals using a cross correlation-based least square support vector machine for brain-computer interface," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, no. 4, pp. 526–538, 2012.

[17] A. Vuckovic and F. Sepulveda, "Delta band contribution in cue based single trial classification of real and imaginary wrist movements," *Medical and Biological Engineering and Computing*, vol. 46, no. 6, pp. 529–539, 2008.

[18] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36–38, pp. 3902–3933, 2005.

[19] X.-S. Yang, "Harmony search as a metaheuristic algorithm," *Music-Inspired Harmony Search Algorithm*, vol. 191, pp. 1–14, 2009.

[20] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[21] Q.-K. Pan, P. N. Suganthan, M. F. Tasgetiren, and J. J. Liang, "A self-adaptive global best harmony search algorithm for continuous optimization problems," *Applied Mathematics and Computation*, vol. 216, no. 3, pp. 830–848, 2010.

[22] L. D. S. Coelho and V. C. Mariani, "An improved harmony search algorithm for power economic load dispatch," *Energy Conversion and Management*, vol. 50, no. 10, pp. 2522–2526, 2009.

[23] Z. W. Geem and K.-B. Sim, "Parameter-setting-free harmony search algorithm," *Applied Mathematics and Computation*, vol. 217, no. 8, pp. 3881–3889, 2010.

[24] M. G. Wentrup and M. Buss, "Multiclass common spatial patterns and information theoretic feature extraction," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 8, pp. 1991–2000, 2008.

[25] L. Wang, Y. Mao, Q. Niu, and M. Fei, "A multi-objective binary harmony search algorithm," *Advances in Swarm Intelligence*, vol. 6729, no. 2, pp. 74–81, 2011.

[26] E. Niedermeyer and F. L. da Silva, "EEG recording and operation of the apparatus," in *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*, Lippincott Williams and Wilkins, Baltimore, Md, USA, 2004.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 953641, 6 pages
http://dx.doi.org/10.1155/2013/953641

Research Article

Parameter Estimation for Traffic Noise Models Using a Harmony Search Algorithm

Deok-Soon An,[1] Young-Chan Suh,[2] Sungho Mun,[3] and Byung-Sik Ohm[1]

[1] *Highway Research Division, SOC Research Institute, Korea Institute of Construction Technology, Goyang-si, Gyeonggi-do 411-712, Republic of Korea*
[2] *Department of Transportation & Logistics Engineering, Hanyang University, Ansan-si, Gyeonggi-do 426-791, Republic of Korea*
[3] *Department of Civil Engineering, Seoul National University of Science & Technology, Seoul 139-743, Republic of Korea*

Correspondence should be addressed to Sungho Mun; smun@seoultech.ac.kr

Received 28 June 2013; Revised 22 September 2013; Accepted 22 September 2013

Academic Editor: Zong Woo Geem

A technique has been developed for predicting road traffic noise for environmental assessment, taking into account traffic volume as well as road surface conditions. The ASJ model (ASJ Prediction Model for Road Traffic Noise, 1999), which is based on the sound power level of the noise emitted by the interaction between the road surface and tires, employs regression models for two road surface types: dense-graded asphalt (DGA) and permeable asphalt (PA). However, these models are not applicable to other types of road surfaces. Accordingly, this paper introduces a parameter estimation procedure for ASJ-based noise prediction models, utilizing a harmony search (HS) algorithm. Traffic noise measurement data for four different vehicle types were used in the algorithm to determine the regression parameters for several road surface types. The parameters of the traffic noise prediction models were evaluated using another measurement set, and good agreement was observed between the predicted and measured sound power levels.

1. Introduction

It is important to evaluate the primary impact of traffic noise for several different road surfaces in order to predict the interactive noise between road surfaces and vehicle tires. A number of noise prediction models [1–6] have been developed for environmental estimation of traffic noise levels in terms of vehicle and pavement types. For example, the vehicle types in the ASJ model [1, 2] are large vehicles, medium vehicles, light trucks, and cars, and the pavement types are dense-graded asphalt (DGA) and permeable asphalt (PA). Thus, the ASJ traffic noise prediction model is restricted to four vehicle types, varying vehicle speeds, and two pavement types.

In order to expand the applicability of the ASJ model, the present paper introduces a parameter estimation procedure based on a harmony search (HS) algorithm as follows: (a) the traffic noise for a targeted road surface is measured, (b) the parameters of the noise prediction model are estimated via the HS algorithm, and (c) the resulting coefficients are evaluated using another set of measurements (consisting of vehicle speeds and vehicle types) for a different traffic volume on the targeted road surface. To validate the proposed traffic noise prediction technique, traffic noise measurement sets from three different surface types were used in this study: stone mastic asphalt surfaces (SMA), 30 mm transversely tined Portland cement concrete surfaces (30 mm trans.), and 18 mm longitudinally tined Portland cement concrete surfaces (18 mm long.), as shown in Figure 1. The measurement site is a 7.7 km long. 2-lane highway along the side of the south bound of Jungbu inland highway in South Korea. This measurement section includes both asphalt and Portland cement concrete pavements.

This paper is organized as follows. Section 2 describes the ASJ model and vehicle characterization. Section 3 explains the application of the HS algorithm to estimate the parameters of ASJ-based noise prediction models, and Section 4 presents the conclusions of this research.

(a) (b) (c)

Figure 1: Pavement surface types: (a) SMA, (b) 30 mm trans., and (c) 18 mm long.

2. ASJ Model

The Acoustic Society of Japan (ASJ) published the ASJ model [1, 2] for calculating road traffic noise. The procedure involves calculating the noise level generated by traffic as well as the attenuation during noise propagation. The octave-band power spectrum for nominal midband frequencies (63 Hz to 8 kHz) can be generated according to the ISO 9613-2 standard [7]. The ASJ model classifies vehicles into four types listed in Table 1.

The A-weighted overall sound power levels (L_{WA}) of the noise emitted by interactions between vehicles and pavement are listed in Table 2 for a dense-graded asphalt (DGA) surface. In terms of nominal midband frequencies, the individual A-weighted sound power level for each octave band ($L_{\mathrm{WA},i}$) is calculated as follows:

$$L_{\mathrm{WA},i} = L_{\mathrm{WA}} + \Delta L_i + \Delta L_{A,i} + \Delta L_{\mathrm{adj}}, \tag{1}$$

where L_{WA} is the overall sound power level (dB) and ΔL_i is the relative level (dB) at the ith nominal mid-band frequency given by

$$\Delta L_i = -10\log_{10}\left\{1 + \left(\frac{f_i}{2000}\right)^2\right\} - 2.5\log_{10}\left(\frac{f_i}{1000}\right). \tag{2}$$

$\Delta L_{A,i}$ is a factor of the standard A-weighting frequency given by the following equation:

$$\begin{aligned}\Delta L_{A,i} = {} & 2.0 + 20\log_{10} \\ & \times \left[\left(12200^2 \cdot f_i^4\right)\right. \\ & \times \left(\left(f_i^2 + 20.6^2\right)\left(f_i^2 + 12200^2\right)\right. \\ & \left.\left. \times \sqrt{\left(f_i^2 + 107.7^2\right)\left(f_i^2 + 737.9^2\right)}\right)^{-1}\right],\end{aligned} \tag{3}$$

where f_i is the nominal mid-band frequency. Finally, ΔL_{adj} is the correction factor defined by

$$\Delta L_{\mathrm{adj}} = -10\log_{10}\left\{\sum_{i=1}^{8} 10^{0.1(\Delta L_i + \Delta L_{A,i})}\right\}. \tag{4}$$

2.1. Equivalent Sound Power Level for a Road. The A-weighted sound power level emitted by a specific type of vehicle moving along a road over a specified time period, L_{WAT}, can be calculated via the following equation:

$$L_{\mathrm{WAT}} = L_{\mathrm{WA}} + 10\log_{10}\left(\frac{3.6\Delta l}{V} \cdot \frac{N}{3600}\right), \tag{5}$$

where L_{WA} is the basic sound power level (dB) emitted by the vehicle (listed in Table 2), Δl is the length of the road segment (in meters), V is the mean speed for the vehicle type (in km/h), and N is the hourly traffic flow for the vehicle type (in vehicles/h). The equivalent sound power level, $L_{\mathrm{eq(WA)}}$, emitted by all vehicles moving along the road can then be calculated as follows:

$$L_{\mathrm{eq(WA)}} = 10\log_{10}\left\{\sum_i 10^{L_{\mathrm{WAT},i}/10}\right\}, \tag{6}$$

where $L_{\mathrm{WAT},i}$ is the A-weighted sound power level for each vehicle type. Here, $L_{\mathrm{WAT},1}$, $L_{\mathrm{WAT},2}$, $L_{\mathrm{WAT},3}$, and $L_{\mathrm{WAT},4}$ are the A-weighted sound power levels for a large vehicle, a medium vehicle, a light truck, and a car, respectively, as listed in Table 2. Furthermore, the attenuation (e.g., geometrical divergence, atmospheric absorption, ground effect, and screening structures) was calculated based on ISO 9613-2 [7].

3. Application of the Harmony Search Algorithm

3.1. Harmony Search Algorithm. This section describes the procedure for estimating the parameters of noise prediction models, using a harmony search (HS) algorithm that employs a heuristic algorithm based on an analogy with natural

Table 1: Vehicle types used in this study.

Type		Specification
Heavy vehicle	Large vehicle	(i) Trucks equipped with three or more axles, which are greater than 8 tons or above 5-ton cargo trailers (ii) Buses providing 30 or more passengers (iii) Large special purpose vehicles
	Medium vehicle	(i) Trucks equipped with two axles (ii) Buses providing less than 30 passengers
Light vehicle	Light truck	Light trucks equipped with 50 to 2000 cc engine
	Car	Car designed for the transportation of less than and equal to 10 passenger.

Table 2: A-weighted sound power levels (L_{WA}) in dB for a dense-graded asphalt (DGA) surface.

Vehicle types	Steady traffic flow (40 km/h to 140 km/h)
Large vehicle	$54.4 + 30\log_{10}V^*$
Medium vehicle	$51.5 + 30\log_{10}V$
Light truck	$47.6 + 30\log_{10}V$
Car	$46.4 + 30\log_{10}V$

*V is a velocity.

phenomena [8–12]. The detailed procedure for applying a harmony search consists of four steps as follows.

(1) The algorithm parameters are specified. These include the harmony memory size (HMS), initialized as the number of solution vectors in the harmony memory (HM), the harmony memory consideration rate (HMCR, between 0 and 1), the pitch adjustment rate (PAR, between 0 and 1), and the maximum number of improvisations (or stopping criterion), which terminates the HS program. The optimization problem is specified as follows:

$$\text{Minimize } f(X) \text{ subject to } x_i \in X_i = 1, 2, \ldots, N, \quad (7)$$

where $f(X)$ is the objective function, X is the set of decision variables x_i, N is the number of decision variables, and X_i is the possible range of values for the ith decision variable; that is, $X_{Li} \le X_i \le X_{Ui}$, where X_{Li} and X_{Ui} are the respective lower and upper bounds for the ith decision variable. To estimate the parameters of a noise prediction model, the following minimization function can be used:

$$\begin{aligned} &\text{Minimize } \left| L_{P\_\text{eq(WA)}} - L_{M\_\text{eq(WA)}} \right| \\ &\text{subject to } x_i \in X_i = 1, 2, \ldots, N, \end{aligned} \quad (8)$$

where $L_{P\_\text{eq(WA)}}$ is the predicted equivalent sound power level of (5), which can be calculated as follows:

$$\begin{aligned} &L_{P\_\text{eq(WA)}} \\ &\quad = 10\log_{10}\left[10^{0.1\{L_{\text{WAT},1}+L_{\text{WAT},2}+L_{\text{WAT},3}+L_{\text{WAT},4}\}}\right], \end{aligned} \quad (9)$$

where $L_{\text{WAT},1}$, $L_{\text{WAT},2}$, $L_{\text{WAT},3}$, and $L_{\text{WAT},4}$ are the A-weighted sound power levels for a large vehicle, a medium vehicle, a light truck, and a car, respectively. $L_{M\_\text{eq(WA)}}$ is the measured equivalent sound power level obtained from previous research [13, 14]. In this optimization problem, the A-weighted sound power levels can be defined as given in Table 3. Thus, the coefficients of x_1, x_2, x_3, and x_4 must be determined via the HS algorithm. The slope is fixed in the ASJ models for both surface types (DGA and PA); therefore, the slope given in Table 3 is fixed at 30.

(2) The HM matrix is initially filled with randomly generated solution vectors up to the HMS, together with the corresponding function values of the random vectors $f(X)$:

$$\text{HM} = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & x_3^1 & x_4^1 & f\left(X^1\right) \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & f\left(X^2\right) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & x_3^{\text{HMS}} & x_4^{\text{HMS}} & f\left(X^{\text{HMS}}\right) \end{array}\right]. \quad (10)$$

(3) A column vector of the newly generated harmony, $X' = (x_1'x_2'x_3'x_4')$, is improvised utilizing the following three mechanisms: (a) random selection, (b) memory consideration, and (c) pitch adjustment. In the random selection, the value of each decision variable, x_i', in the column vector is randomly chosen within the range of values with a probability of (1-HMCR). HMCR (which is between 0 and 1) is the rate at which a single value is chosen from the historical values stored in the HM. The value of each decision variable selected by memory consideration is examined in terms of pitch adjustment. This operation uses the PAR parameter (which is the rate of the necessary pitch adjustment according to the neighboring pitches) with a probability of HMCR ×

Table 3: A-weighted sound power levels (L_{WA}) in dB for the different surface types.

Vehicle types	Steady traffic flow (40 km/h to 140 km/h)
Large vehicle	$x_1 + 30\log_{10}V^*$
Medium vehicle	$x_2 + 30\log_{10}V$
Light truck	$x_3 + 30\log_{10}V$
Car	$x_4 + 30\log_{10}V$

*V is a velocity.

Table 4: Vehicle velocities and hourly traffic flows.

Vehicle types	Average velocities	Hourly traffic flows
Large vehicle	90 km/h	48 vehicles/h
Medium vehicle	95 km/h	40 vehicles/h
Light truck	97 km/h	96 vehicles/h
Car	101 km/h	208 vehicles/h

PAR. Pitch adjustment is applied to each variable as follows:

$$x_i' \longleftarrow \begin{cases} x_i' + rand \times bw, \\ \quad \text{with a probability of HMCR} \times \text{PAR} \times 0.5 \\ x_i' - rand \times bw, \\ \quad \text{with a probability of HMCR} \times \text{PAR} \times 0.5 \\ x_i', \text{ with a probability of HMCR} \times (1 - \text{PAR}). \end{cases} \tag{11}$$

If the newly generated column vector is better than the worst harmony in the HM, based on evaluation of the objective function, the newly generated column vector is included in the HM, and the existing worst harmony is excluded from the HM.

(4) If the stopping criterion (or maximum number of improvisations) is satisfied, the computation is terminated. Otherwise, Steps 3 and 4 are repeated.

3.2. Application to Parameter Estimation for Noise Prediction Models. In order to estimate the parameters of noise prediction models based on the objective function of (8), noise measurements for three different road surfaces were obtained from previous research, which was conducted on a test track [13, 14]. The vehicle velocities and hourly traffic flows are listed in Table 4.

To apply the HS algorithm to parameter estimation for the noise prediction models, the four coefficients were determined for each road surface type (stone mastic asphalt (SMA) surface, 30 mm transversely tined Portland cement concrete surface (30 mm trans.), and 18-mm longitudinally tined Portland cement concrete surface (18 mm long.)), as shown in Figure 2 and Table 5, based on the training data from Table 4. In this way, the coefficients of the noise prediction models, which are dependent on the road surface type, can be updated via the HS algorithm. As a result, noise prediction models can be provided for various surface types by using the HS algorithm to update the ASJ model equations.

Another set of testing data (given in Table 6) was used to evaluate whether or not the updated noise prediction models

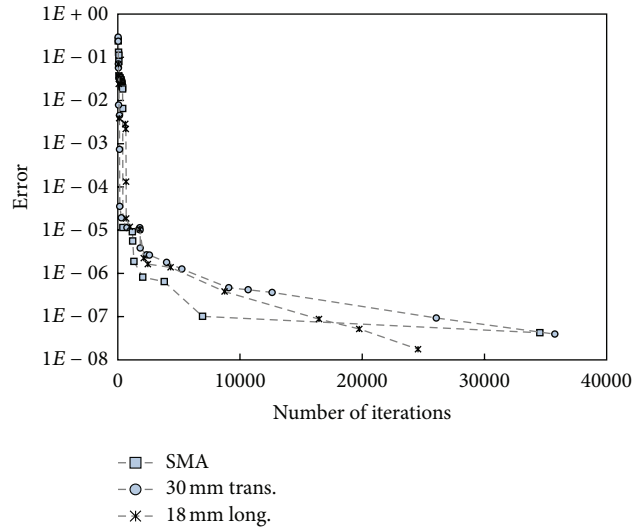


Figure 2: Minimizing the error function of (8) through the HS algorithm.

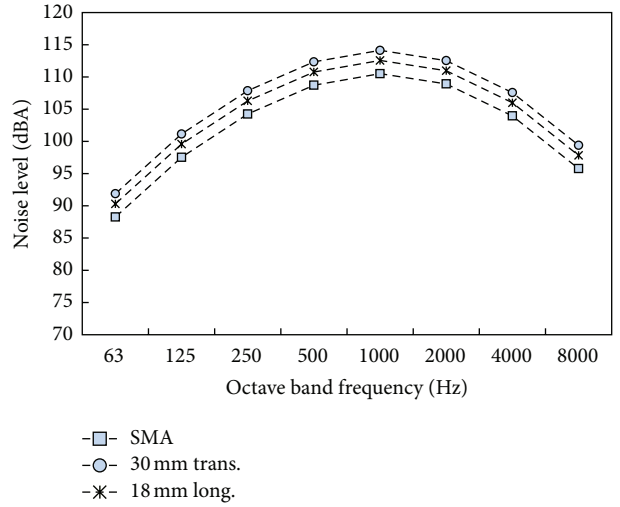


Figure 3: Octave band frequency versus noise level.

(with the four coefficients estimated by the HS algorithm) provided results consistent with measured noise levels. The predictions and measurements are compared in Table 7; good agreement was noted for all three surface types. Using the original ASJ model, the prediction value is resulted in a same noise level regardless of surface types and in bad agreement when compared with the measured noise levels.

Finally, the A-weighted sound power levels for the individual octave bands were estimated for the SMA, 30 mm trans., and 18 mm long. surface types, utilizing the parameters estimated via the HS algorithm and (1). For example, Figure 3 shows the A-weighted sound power levels for the octave bands for the three different surface types in the case of the large vehicle speeding at 80 km/h.

Table 5: Determination of the model coefficients via the HS algorithm.

Surface types	Large vehicle (x_1 in Table 3)	Medium vehicle (x_2 in Table 3)	Light truck (x_3 in Table 3)	Car (x_4 in Table 3)
SMA	58.050	53.693	48.456	45.787
30 mm trans.	61.669	58.473	51.246	46.020
18 mm long.	60.097	54.428	50.095	45.360

Table 6: Vehicle velocities and hourly traffic flows used as evaluation data.

Vehicle types	Average velocities	Hourly traffic flows
Large vehicle	85 km/h	35 vehicles/h
Medium vehicle	90 km/h	58 vehicles/h
Light truck	95 km/h	87 vehicles/h
Car	103 km/h	198 vehicles/h

Table 7: Comparison of predicted and measured traffic noise levels.

Surface types	Predicted levels	Measured levels	ASJ model prediction
SMA	96.1 dB	96.2 dB	94.4 dB
30 mm trans.	99.5 dB	99.7 dB	94.4 dB
18 mm long.	97.3 dB	97.6 dB	94.4 dB

4. Conclusions

In this study, it was shown that the optimization problem related to updating the noise prediction models for several surface types could be solved using an HS algorithm process. The process involves (a) obtaining measurements for different road surfaces, (b) estimating the coefficients of the noise prediction models using this measurement set as training data, and (c) evaluating the estimated coefficients using another measurement set as testing data. When this procedure was utilized, an evaluation of the parameters of the traffic noise prediction model yielded good agreement between predicted and measured sound power levels.

Conflict of Interests

The authors of the paper do not have a direct financial relationship with the commercial identities mentioned in this paper.

Acknowledgment

This research was supported by a grant from a Strategic Research Project "Development of Traffic Noise Modeling and Noise Reduction Technology as Road Traffic Noise by Location" funded by the Korea Institute of Construction Technology.

References

[1] Acoustic Society of Japan, "ASJ prediction model for road traffic noise," *Bull. Acoustic Society of Japan*, vol. 55, no. 1, pp. 281–321, 1999.

[2] K. Yamamoto, "Road traffic noise prediction model "ASJ RTN-Model 2008": report of the research committee on road traffic noise," *Acoustical Science and Technology*, vol. 31, no. 1, pp. 2–55, 2010.

[3] U. Sandberg, "Road traffic noise—the influence of the road surface and its characterization," *Applied Acoustics*, vol. 21, no. 2, pp. 97–118, 1987.

[4] C. Steele, "Critical review of some traffic noise prediction models," *Applied Acoustics*, vol. 62, no. 3, pp. 271–287, 2001.

[5] D. S. Cho, J. H. Kim, T. M. Choi, B. H. Kim, and D. Manvell, "Highway traffic noise prediction using method fully compliant with ISO 9613: comparison with measurements," *Applied Acoustics*, vol. 65, no. 9, pp. 883–892, 2004.

[6] T. Bennert, D. Hanson, A. Maher, and N. Vitillo, "Influence of pavement surface type on tire/pavement generated noise," *Journal of Testing and Evaluation*, vol. 33, no. 2, pp. 94–100, 2005.

[7] International Organization for Standardization, ISO 9613-2:1996. Acoustics—attenuation of sound during propagation outdoors—part 2: general method of calculation, 1996.

[8] S. Mun and Z. W. Geem, "Determination of viscoelastic and damage properties of hot mix asphalt concrete using a harmony search algorithm," *Mechanics of Materials*, vol. 41, no. 3, pp. 339–353, 2009.

[9] S. Mun and Z. W. Geem, "Determination of individual sound power levels of noise sources using a harmony search algorithm," *International Journal of Industrial Ergonomics*, vol. 39, no. 2, pp. 366–370, 2009.

[10] Z. W. Geem, "Particle-swarm harmony search for water network design," *Engineering Optimization*, vol. 41, no. 4, pp. 297–311, 2009.

[11] J. Fourie, R. Green, and Z. W. Geem, "Generalised adaptive harmony search: a comparative analysis of modern harmony search," *Journal of Applied Mathematics*, vol. 2013, Article ID 380985, 13 pages, 2013.

[12] S. Lee, S. Mun, and H. Moon, "Determination of pavement rehabilitation activities through a permutation algorithm," *Journal of Applied Mathematics*, vol. 2013, Article ID 252808, 5 pages, 2013.

[13] S. Mun, D.-S. Cho, and T.-M. Choi, "Influence of pavement surface noise: the Korea Highway Corporation test road,"

Canadian Journal of Civil Engineering, vol. 34, no. 7, pp. 809–816, 2007.

[14] D.-S. Cho and S. Mun, "Determination of the sound power levels emitted by various vehicles using a novel testing method," *Applied Acoustics*, vol. 69, no. 3, pp. 185–195, 2008.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 139464, 20 pages
http://dx.doi.org/10.1155/2013/139464

Research Article

Cellular Harmony Search for Optimization Problems

Mohammed Azmi Al-Betar,[1,2] Ahamad Tajudin Khader,[1] Mohammed A. Awadallah,[1] Mahmmoud Hafsaldin Alawan,[1] and Belal Zaqaibeh[3]

[1] *School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia*
[2] *Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, P.O. Box 50, Al-Huson, Irbid, Jordan*
[3] *Department of Computer Science, Jadara University, P.O. Box 733, Irbid, Jordan*

Correspondence should be addressed to Mohammed Azmi Al-Betar; mohbetar@cs.usm.my

Received 8 July 2013; Revised 15 August 2013; Accepted 16 August 2013

Academic Editor: Zong Woo Geem

Structured population in evolutionary algorithms (EAs) is an important research track where an individual only interacts with its neighboring individuals in the breeding step. The main rationale behind this is to provide a high level of diversity to overcome the genetic drift. Cellular automata concepts have been embedded to the process of EA in order to provide a decentralized method in order to preserve the population structure. Harmony search (HS) is a recent EA that considers the whole individuals in the breeding step. In this paper, the cellular automata concepts are embedded into the HS algorithm to come up with a new version called cellular harmony search (cHS). In cHS, the population is arranged as a two-dimensional toroidal grid, where each individual in the grid is a cell and only interacts with its neighbors. The memory consideration and population update are modified according to cellular EA theory. The experimental results using benchmark functions show that embedding the cellular automata concepts with HS processes directly affects the performance. Finally, a parameter sensitivity analysis of the cHS variation is analyzed and a comparative evaluation shows the success of cHS.

1. Introduction

The optimization techniques have the utility of navigating the search space using effective operators driven by control parameters. The tricky point of the success of any optimization method is its ability to strike a suitable balance between exploration (diversification) and exploitation (intensification) of the problem search space [1]. Exploration is the optimization method capability of navigating a promising region of the search space, if necessary, while exploitation refers to the capability of fine-tuning the already-navigated regions to converge into the local optima [1].

Harmony search (HS) algorithm is a recent evolutionary algorithm (EA) proposed by Geem et al. [2] to imitate the musical improvisation process. Due to its advantages over other optimization methods, it stipulates fewer mathematical requirements in the initial search [3]. It has a novel stochastic derivative which reduces the number of iterations required to converge towards local minima [4], in addition to being simple, adaptable, general, and scalable [5]. Therefore, HS algorithm has been intensively tailored for several optimization problems such as timetabling [6, 7], nurse restoring [8], space allocation [9], and many others [10–13]. However, due to the complex nature of some optimization problems and the avoidance of a premature convergence situation, HS theories are modified [14]. Furthermore, the control parameter adaptation for HS is also studied [5, 15–17].

In a procedural context, HS, which is an iterative improvement algorithm, initiates with a population of random individuals stored in harmony memory (HM). At each iteration, a new individual is generated based on three operators: (i) memory consideration, which *selects* the variables of a new individual from whole HM individuals; (ii) pitch adjustment, which is responsible for local improvement, and (iii) random consideration, used to provide random elements for the new individual. The new individual is then evaluated

Table 1: CEC'2005 functions.

Abb.	Function name	Search range	x^*	$f(x^*)$
f_1	Shifted sphere function	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	−450
f_2	Shifted Schwefel's problem 1.2	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, \ldots, o_D)$	−450
f_3	Shifted rotated high conditioned elliptic function	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	−450
f_4	Shifted Schwefel's problem 1.2 with noise in fitness	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	−450
f_5	Schwefel's problem 2.6 with global optimum on bounds	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	−310
f_6	Shifted Rosenbrock's function	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	390
f_7	Shifted rotated Griewank's function without bounds	$[0, 600]$	$\mathbf{o} = (o_1, o_2, o_D)$	−180
f_8	Shifted rotated Ackley's function with global optimum on bounds	$[-32, 32]$	$\mathbf{o} = (o_1, o_2, o_D)$	−140
f_9	Shifted rastrigin's function	$[-5, 5]$	$\mathbf{o} = (o_1, o_2, o_D)$	−330
f_{10}	Shifted rotated Rastrigin's function	$[-5, 5]$	$\mathbf{o} = (o_1, o_2, o_D)$	−330
f_{11}	Shifted rotated Weierstrass function	$[-0.5, 0.5]$	$\mathbf{o} = (o_1, o_2, o_D)$	90
f_{12}	Schwefel's problem 2.13	$[-\pi, \pi]$	α	−460
f_{13}	Expanded extended Griewank's plus Rosenbrock's function (F8F2)	$[-5, 5]$	$\mathbf{o} = (o_1, o_2, o_D)$	−130
f_{14}	Shifted rotated expanded Scaffer's F6	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	−300
f_{15}	Hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	120
f_{16}	Rotated hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	120
f_{17}	Rotated hybrid composition function with noise in fitness	$[-5, 5]$	$\mathbf{o}_1$	120
f_{18}	Rotated hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	10
f_{19}	Rotated hybrid composition function with a narrow basin for the global optimum	$[-5, 5]$	$\mathbf{o}_1$	10
f_{20}	Rotated hybrid composition function with the global optimum on the bounds	$[-5, 5]$	$\mathbf{o}_1$	10
f_{21}	Rotated hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	360
f_{22}	Rotated hybrid composition function with high condition number matrix	$[-5, 5]$	$\mathbf{o}_1$	360
f_{23}	Noncontinuous rotated hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	360
f_{24}	Rotated hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	260
f_{25}	Rotated hybrid composition function without bounds	$[2, 5]$	$\mathbf{o}_1$	260

and replaces the worst individual in HM, if it is better. This process is cycled until a desired condition is met (see Table 3).

As other EAs, HS algorithm interacts with whole individuals in the HM during each breeding step. The update process selects the worst individual from a single HM and replaces it with a new one, if better. The decentralized methods used in other structured EAs have shown that the performance of EA is improved. Examples include cellular genetic algorithm (cGA) [18], distributed EA (dEA) [19], cellular PSO [20], and others [21]. The main idea of these structured methods is to partition the population into several sets with common features [22].

Cellular genetic algorithm (cGA), in particular, is a decentralized method where the population is represented as a toroidal grid of two-dimensions, as shown in Figure 3 [23, 24]. The individuals are located in this toroidal in a predefined topology and solely interact with their nearest neighbors in the breeding step [22]. Note that all the neighborhoods have the same size and identical shape. This concept embedded in cGA provides useful advantages for the optimization domain [23] and parallel implementations [25, 26] because it assists in providing a high-level of diversity and yields a small diffusion of solutions through the search. The cGA provides a proper exploitation power inside each neighborhood of an individual by the operators of GA [22]. Theoretically, it has been shown that nonrandom mating keeps genetic diversity at higher level, thus preventing the algorithms from converging prematurely to the local optima [27].

The main objective of this paper is to embed the cellular automata concepts in the HS algorithm optimization framework, where the HM is arranged as a two-dimensional toroidal grid. The population diffusion will be expectably preserved, maintaining a high-level of diversity during the search, thus avoiding genetic drift. The improvisation process of HS algorithm is adjusted to interact with the neighborhoods of specific individuals. The updating process of HM is done within the neighborhoods of that individual. The results show that the new decentralized version of HS (i.e., cHS) algorithm improves the performance of HS using standard benchmark functions.

The rest of the paper is organized as follows. The basics of HS algorithm are described in Section 2. The proposed cellular harmony search (cHS) is discussed in Section 3. Results of the experiments are presented in Section 4. Finally, the conclusion and promising future research directions are provided in Section 5.

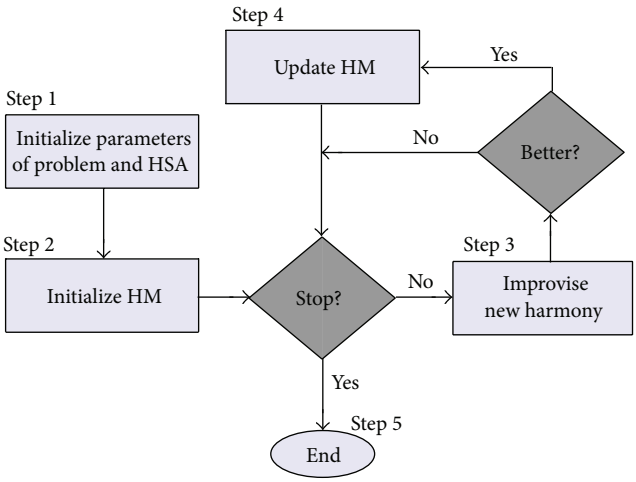


Figure 1: The flowchart of the HS algorithm.

2. The Harmony Search Algorithm

The harmony search (HS) algorithm is a recent evolutionary approach proposed by Geem et al. [2]. It is initiated with a set of individuals stored in an augmented matrix called harmony memory (HM). It is a centralized algorithm where at each breeding step, it generates a new individual by interacting with the whole individuals in HM. HS follows three rules in the breeding step to generate a new individual: memory consideration, random consideration, and pitch adjustment. If the new individual is better than the worst individual in the whole HM, the replacement process is triggered. This process is repeated as many times as the HS is stagnated. The steps of HS algorithm are flowcharted in Figure 1, where each step is described below in more detail.

Step 1 (initialize parameters). The optimization problem is initially represented as $\min\{f(\mathbf{x}) \mid \mathbf{x} \in \mathbf{X}\}$, where $f(\mathbf{x})$ is the objective function and $\mathbf{x} = \{x_i \mid i = 1, \ldots, N\}$ is the set of decision variables. $\mathbf{X} = \{\mathbf{X}_i \mid i = 1, \ldots, N\}$ is the possible value range for each decision variable, where $\mathbf{X}_i \in [LB_i, UB_i]$, and LB_i and UB_i are the lower and upper bounds for the decision variable x_i, respectively, and N is the number of decision variables. The parameters of the HS algorithm are also predefined in the following step.

(a) The harmony memory consideration rate (HMCR), used in the breeding step to determine whether the value of a decision variable is to be selected from the individuals stored in the harmony memory (HM).

(b) The harmony memory size (HMS) which determines the number of individuals in HM.

(c) The pitch adjustment rate (PAR), which is used to decide the adjustments of some decision variables selected from memory.

(d) The distance bandwidth (BW) which determines the distance of the adjustment that occurs to the individual in the pitch adjustment operator.

(e) The number of improvisations (NI) which is similar to the number of generations.

Step 2 (initialize the harmony memory (HM)). The harmony memory (HM) is a matrix of size $N \times \text{HMS}$ which includes sets of individuals determined by HMS (see (1)). In this step, these individuals are randomly generated as follows: $x_i^j = LB_i + (UB_i - LB_i) \times U(0,1)$, for all $i = 1, 2, \ldots, N$ and for all $j = 1, 2, \ldots, \text{HMS}$, and $U(0,1)$ generate a random number between 0 and 1. Consider

$$\text{HM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_N^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & \cdots & x_N^{\text{HMS}} \end{bmatrix}. \tag{1}$$

Step 3 (improvise a new individual). The HS algorithm generates a new individual, $\mathbf{x}' = (x_1', x_2', \ldots, x_N')$, using three operators: (1) memory consideration, (2) random consideration, and (3) pitch adjustment.

Memory Consideration. In memory consideration, the value of the first decision variable in the new individual x_1' is randomly *selected* from the historical values, $\{x_1^1, x_1^2, \ldots, x_1^{\text{HMS}}\}$, stored in whole HM individuals. Values of the other decision variables, $(x_2', x_3', \ldots, x_N')$, are sequentially assigned in a similar way with a probability of HMCR, where $\text{HMCR} \in (0,1)$. It is worth mentioning that this process interacts with the whole individuals in HM which might lead to a premature convergence situation due to the genetic drift.

Random Consideration. Decision variables that are not assigned with values according to memory consideration are randomly assigned according to their possible range by random consideration with a probability of (1-HMCR) as follows:

$$x_i' \longleftarrow \begin{cases} x_i' \in \left\{x_i^1, x_i^2, \ldots, x_i^{\text{HMS}}\right\} & \text{w.p. HMCR,} \\ x_i' \in \mathbf{X}_i & \text{w.p. } 1 - \text{HMCR.} \end{cases} \tag{2}$$

Pitch Adjustment. Each decision variable x_i' of a new individual, $\mathbf{x}' = (x_1', x_2', x_3', \ldots, x_N')$, that has been assigned a value by memory consideration is pitch adjusted with the probability of PAR, where $\text{PAR} \in (0,1)$ as follows:

$$\text{Pitch adjusting decision for } x_i' \longleftarrow \begin{cases} \text{Yes} & \text{w.p. PAR,} \\ \text{No} & \text{w.p. } 1 - \text{PAR.} \end{cases} \tag{3}$$

In pitch adjustment, if the decision for x_i' is Yes, the value of x_i' is modified to its neighboring value as follows: $x_i' = x_i' \pm U(0,1) \times \text{BW}$.

Step 4 (update HM). If the new individual, $\mathbf{x}' = (x_1', x_2', \ldots, x_N')$, has better objective function value than that of the worst individual $\mathbf{x}^{\text{worst}}$ stored in HM (i.e., $\mathbf{x}^{\text{worst}} = \mathbf{x}^{\text{HMS}}$ in case HM is sorted), the worst individual will be replaced

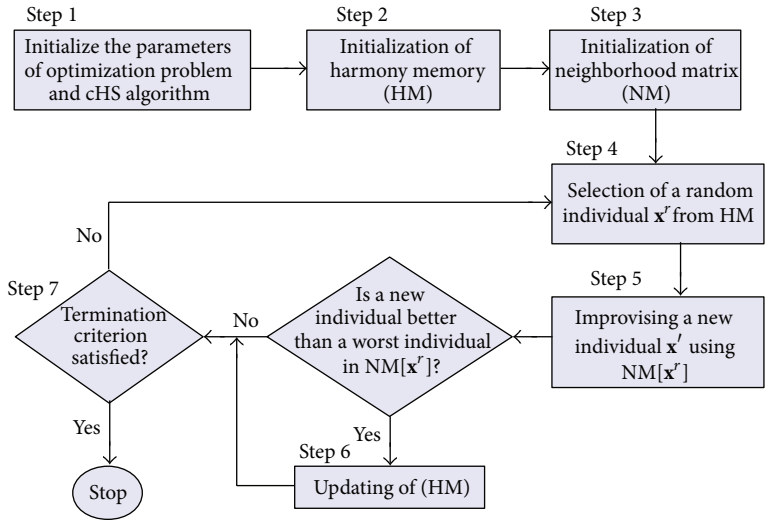


Figure 2: The flowchart of cHS.

by the new one. Note that the worst individual is selected from the whole HM where the decentralized structure of the population is not observed.

Step 5 (check the termination rule). The HS algorithm will repeat Steps 3 and 4 of HS until a termination rule is met, which is normally decided by the value of NI parameter.

3. Cellular Harmony Search (cHS) Algorithm

There has been much interest from researchers and scientists from different fields exploiting the cellular automata (CA) in physics, biology, social science, computer science, and so on. The initial concepts of CA were developed by Neumann [28] and have been an effective research tool to be incorporated with a wide variety of disciplines.

The concepts of cellular automata (CA) are normally concerned with individual perspective. The main idea from CA is to provide a population of a particular structure formulated as a toroidal grid. The cell in the toroidal grid refers to an individual who communicates with his closest neighboring individuals so that all the individuals have exactly the same number of neighbors. This leads us to a kind of locality known as *isolation by distance*. Normally, the Manhattan distance is used to measure the distance between any individual and his neighbors. Note that the neighboring individuals have identical shapes and the same size [22].

There exist two-different kinds of cellular models based on how the breeding cycle is performed to the individuals. To put it differently, if the cycle is performed to the whole individuals at the same time, the cellular model is said to be synchronous, where the individuals of the next generation are simultaneously build. On the other hand, if the individuals of the population are sequentially updated with a particular order policy, an asynchronous cellular model is stated. For more discussion about the theory of cellular automata, relevant papers can be seen in [22, 29].

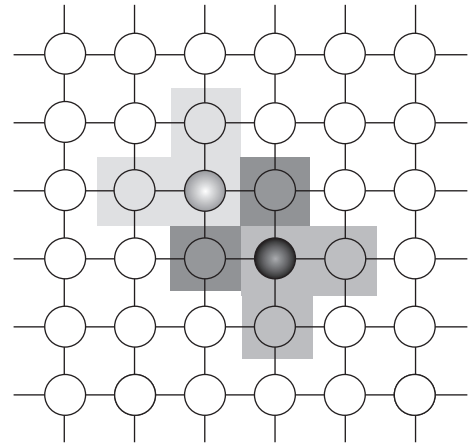


Figure 3: The population arrangement based on cellular structure where each cell is an individual in HM.

Cellular harmony search (cHS) algorithm can be considered as a new decentralized variation of HS which hinges on the structured HM. The individuals in the HM are arranged in the form of two-dimensional toroidal grid. This is meant to keep a high level of diversity during the breeding step and thus increases the chance to converge into global minima. This can be achieved by avoiding the genetic drift and providing a more suitable population diffusion during the search.

Some steps of cHS algorithm to the original version of HS algorithm presented in Section 2 have been adjusted. The adjustments are flowcharted in Figure 2. The breeding step of cHS solely interacts with the neighboring individuals of a randomly selected individual using "cellular memory consideration" operators. The update of HM step is adjusted to replace the worst individual amongst the neighboring individuals with a new individual, but not amongst the whole HM. Figure 3 shows the population on cellular structure (toroidal grid population), which inspires the concept of

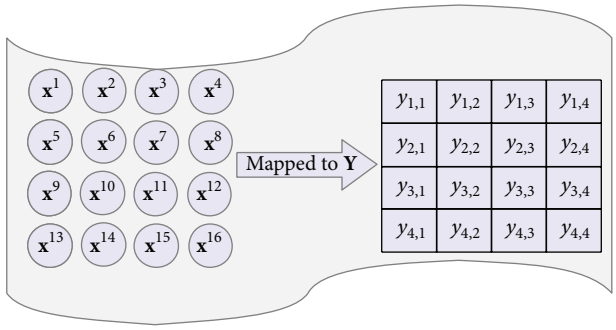


Figure 4: The individuals of HM mapped into the toroidal mesh **Y.**

small neighborhood regions in the cellular automata. Particularly, the overlap of the neighborhood provides implicit technique migration into population. Therefore, the best solutions are diffused smoothly in the whole population, where the diversity of the cellular harmony search is preserved throughout the search. The cellular HS model includes several components.

(1) Cell: the selected random individual in the population (the number of individual is HMS).

(2) Cell space: the set of the whole individuals in HM.

(3) Neighborhood: the set of potential mates of any individual.

(4) Neighborhood shapes: the way of selecting the neighborhoods of the cell as seen in Figure 5.

(5) Discrete time limit: the number of generations in HS algorithm which is normally determined by NI parameter.

The detailed steps of cHS are discussed in the steps below.

Step 1 (initialize cHS parameters and optimization problem). It is clear that the successful search of any metaheuristic method is based on skillful parameter setting. The parameters have different effects on optimization solutions. The parameters of cHS are harmony memory size (HMS), harmony memory consideration rate (HMCR), pitch adjusting rate (PAR), number of improvisation (NI), and the size of neighborhood (NH) determined by cellular structure (see Figure 5), where if NH = L9, this means the neighbors structure is length with 8 neighbors.

Step 2 (initialize the harmony memory (HM)). This step is the same as Step 2 in the original version of HS. Note that the individuals of the HM are arranged as a two-dimensional toroidal grid as shown in Figure 4.

Step 3 (initialization of neighborhood matrix (NM)). NM is a binary matrix of size HMS × HMS (see (4)). The binary values are assigned to each element $\mathrm{NM}_{i,j}$ based on NH parameter. This matrix is used during the breeding step to determine the neighboring individuals of any randomly selected individual. The **NM** matrix is filled by binary value as in (5). Consider

$$\mathbf{NM} = \begin{bmatrix} \mathrm{NM}_{1,1} & \cdots & \cdots & \mathrm{NM}_{1,\mathrm{HMS}} \\ \vdots & \ddots & \ddots & \vdots \\ \mathrm{NM}_{\mathrm{HMS},1} & \cdots & \cdots & \mathrm{NM}_{\mathrm{HMS},\mathrm{HMS}} \end{bmatrix}, \tag{4}$$

$$\mathrm{NM}_{i,j} \longleftarrow \begin{cases} 1 & \mathbf{x}^j \in \mathcal{N}\left(\mathbf{x}^i\right) \quad \forall i \wedge j \in \{1, \ldots, \mathrm{HMS}\}. \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The $\mathcal{N}(\mathbf{x}^{\mathbf{i}})$ is a set of all neighboring individuals of the individual $\mathbf{x}^i$ arranged in two-dimensional toroidal mesh. This set is determined based on a *neighborhood shape* as seen in Figure 5.

Figure 4 shows how the HM individuals are mapped to toroidal grid. Note that the element $Y_{1,1}$ reflects the index of the individual 1 in HM, while the $Y_{K,K}$ reflects the individual index HMS in HM. The HMS value is the square value of K (i.e., $K^2 = \mathrm{HMS}$).

To map the element $(Y_{i,j})$ in toroidal grid **Y** and the individual index r in HM, the following will be used:

$$r = \left((i-1) \times K + j\right). \tag{6}$$

To map the index of the individual r in HM to the element $(Y_{i,j})$ in the matrix NM, the following will be used:

$$i = \mathrm{int}\left(\frac{(r-1)}{K}\right) + 1, \tag{7}$$

$$j = ((r-1) \bmod K) + 1. \tag{8}$$

This mapping mechanism between the individual index of HM and the elements in **Y** is very useful to determine the neighboring of any individual in HM. As shown in Figure 5, there are several neighborhood shapes to determine the neighbors of any individual. For example, the L5 neighborhood shape takes the nearest neighbors of a given cell axial direction. Therefore, to determine the individual indexes in HM that belong to the neighbors of the individual index r using L5, the i and j should be calculated using (7) and (8). And the set of neighbors of individual r called $\vartheta = \{Y_{i-1,j}, Y_{i+1,j}, Y_{i,j-1}, Y_{i,j+1}\}$, then (8) is used to map the elements in **Y** to the corresponding individual indexes in HM. The same step is used if different neighborhood shapes (i.e., L9, C9,...) are used to determine the neighboring individuals of $\mathbf{x}^r$.

As can be noted, the individuals in the set ϑ will be assigned by 1 in **NM** for the individual $\mathbf{x}^r$ while others will be assigned by 0.

Step 4 (select random individual from HM). The selection of a random individual $\mathbf{x}^r$, $r \in \{1, \ldots, \mathrm{HMS}\}$, is done to determine the neighboring individuals taken from the matrix **NM** (i.e., $\xi_r = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^m\}$), where m is the number of the neighboring individuals; that is, $m \leq \mathrm{HMS}$. The neighboring individuals interact together in order to generate the new individual in the next step.

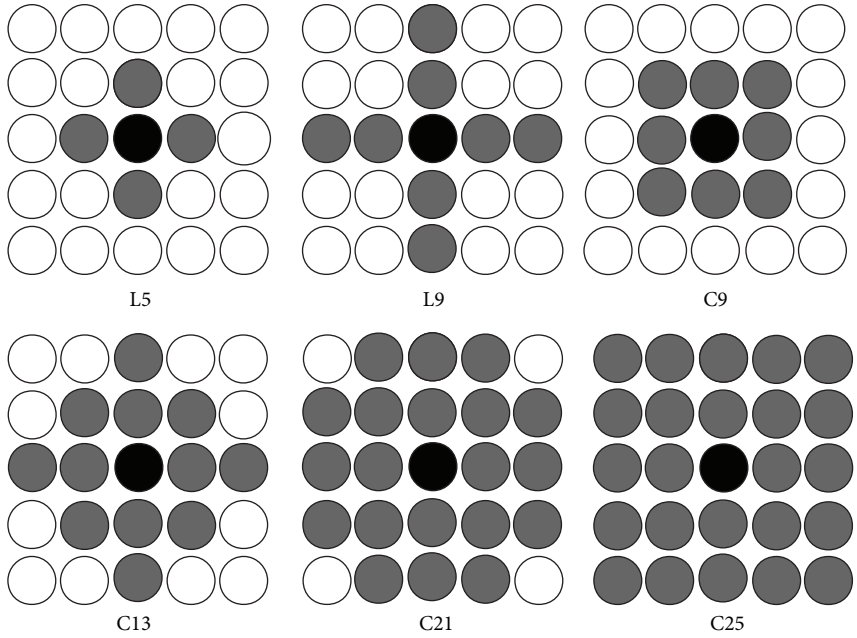


Figure 5: Neighboring shapes [22].

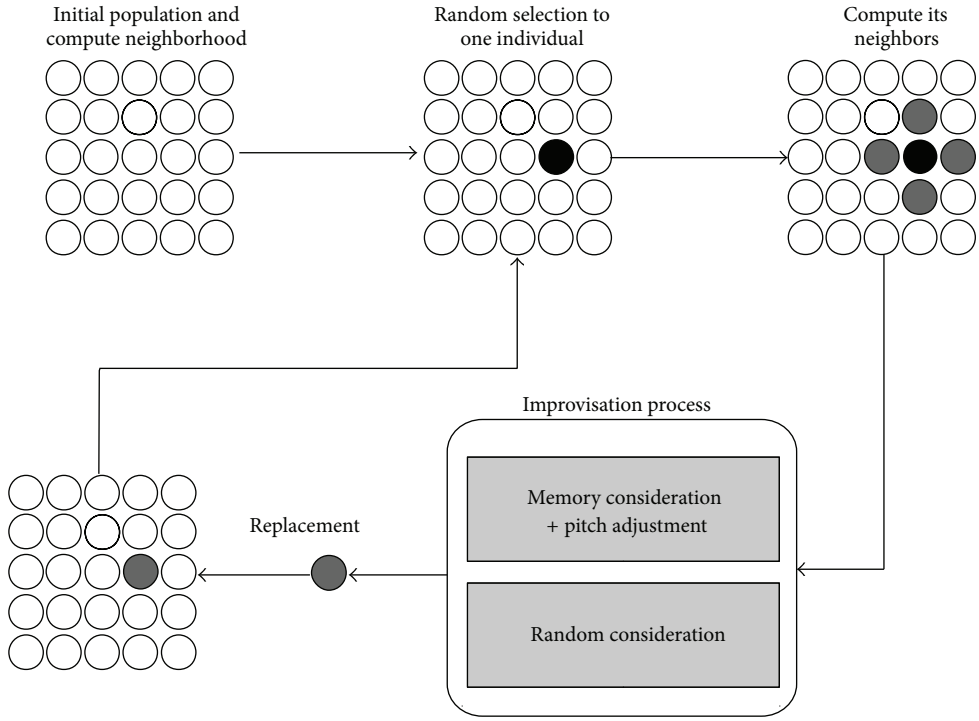
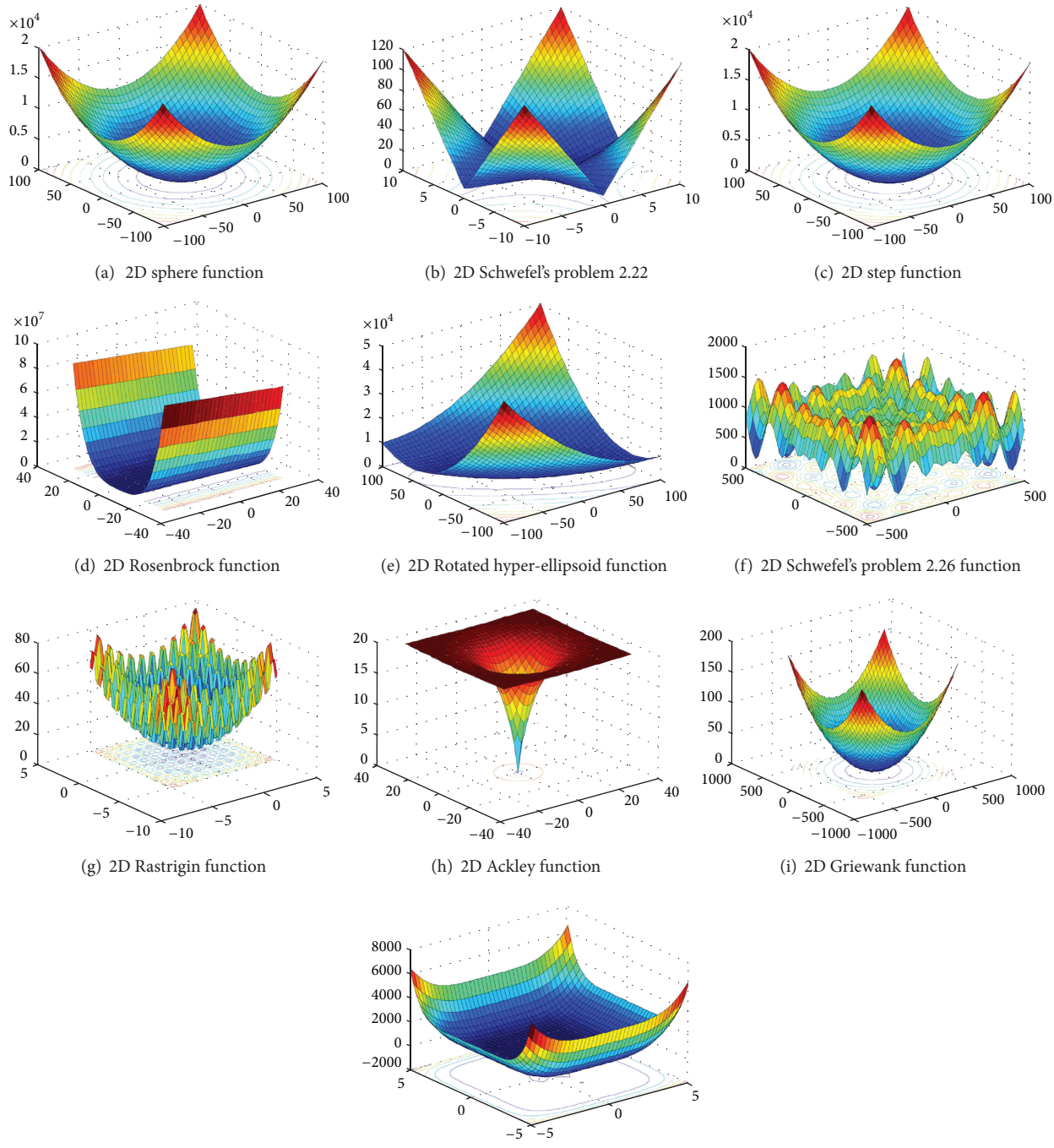


Figure 6: The processes fluency of cHS algorithm.

Step 5 (generate a new individual). In this step, a new individual, $\mathbf{x} = (\acute{x}_1, \acute{x}_2, \ldots, \acute{x}_N)$, is generated based on the three operators: (1) cellular memory consideration, (2) pitch adjustment, and (3) random consideration. The whole process is drawn in Figure 6.

The pitch adjustment and random consideration operators in cHS algorithm are the same as those in the original version of HS algorithm. However, the memory consideration is modified to be inline with the concepts of cellular automata as follows.

Cellular Memory Consideration. As in cellular GA, the cellular memory consideration solely interacts with the close individuals of $\mathbf{x}^r$ taken from the set of ξ_r. The other individuals are not used in the breeding step. The value of the first decision



(a) 2D sphere function
(b) 2D Schwefel's problem 2.22
(c) 2D step function
(d) 2D Rosenbrock function
(e) 2D Rotated hyper-ellipsoid function
(f) 2D Schwefel's problem 2.26 function
(g) 2D Rastrigin function
(h) 2D Ackley function
(i) 2D Griewank function
(j) Six-Hump Camel-Back function

FIGURE 7: The benchmark functions landscape where the value of $N = 2$ (2 dimensions).

variable $\acute{x}_1$ is randomly *selected* from the historical values, $\{x_1^1, x_1^2, \ldots, x_1^m\}$, stored individuals in the set ξ_r. The other decision variables, $(x_2', x_3', \ldots, x_N')$, are sequentially assigned in a similar way with a probability of HMCR, where HMCR $\in$ $(0, 1)$.

It is worth mentioning that cellular memory consideration is able to control the diffusion between the individuals in HM, and thus, it is able to preserve the cHS diversity as long as the search process is iterated. By this strategy, the population is structured and it is possible to improve the numerical behavior of the cHS algorithm.

Step 6 (update the harmony memory). This step is modified in cHS algorithm. The worst individual from the set of neighbors ξ_r, that is, $(\mathbf{x}^{\text{worst}} \mid \mathbf{x}^{\text{worst}} \in \xi_r \wedge f(\mathbf{x}^{\text{worst}}) \geq f(\mathbf{x}^i), \forall i \in \{1, \ldots, m\})$, is replaced by the new individual $\acute{x}$, if better. Note that the replacement process is done taking into account the neighboring individuals in the set ξ_r only.

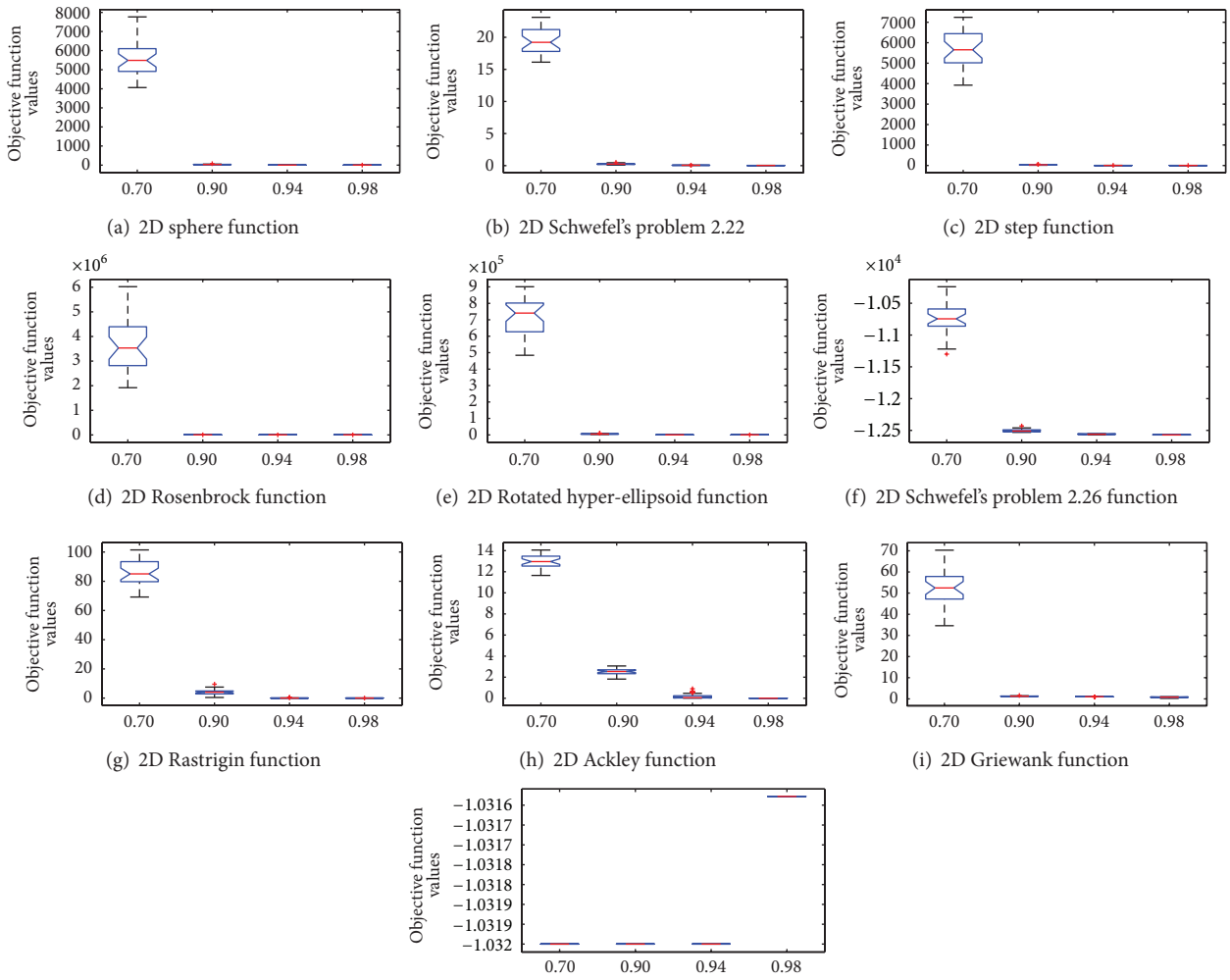


Figure 8: The box plots for showing the effect varying HMCR values using the ten global optimization functions.

Step 7 (check stopping criterion). In this step, the cHS will stop if the maximum number of the iteration (i.e., NI) is reached; otherwise the algorithm repeats Steps 4 to 6. The pseudocode of cHS is demonstrated in Algorithm 1.

4. Experimental and Comparative Evaluation

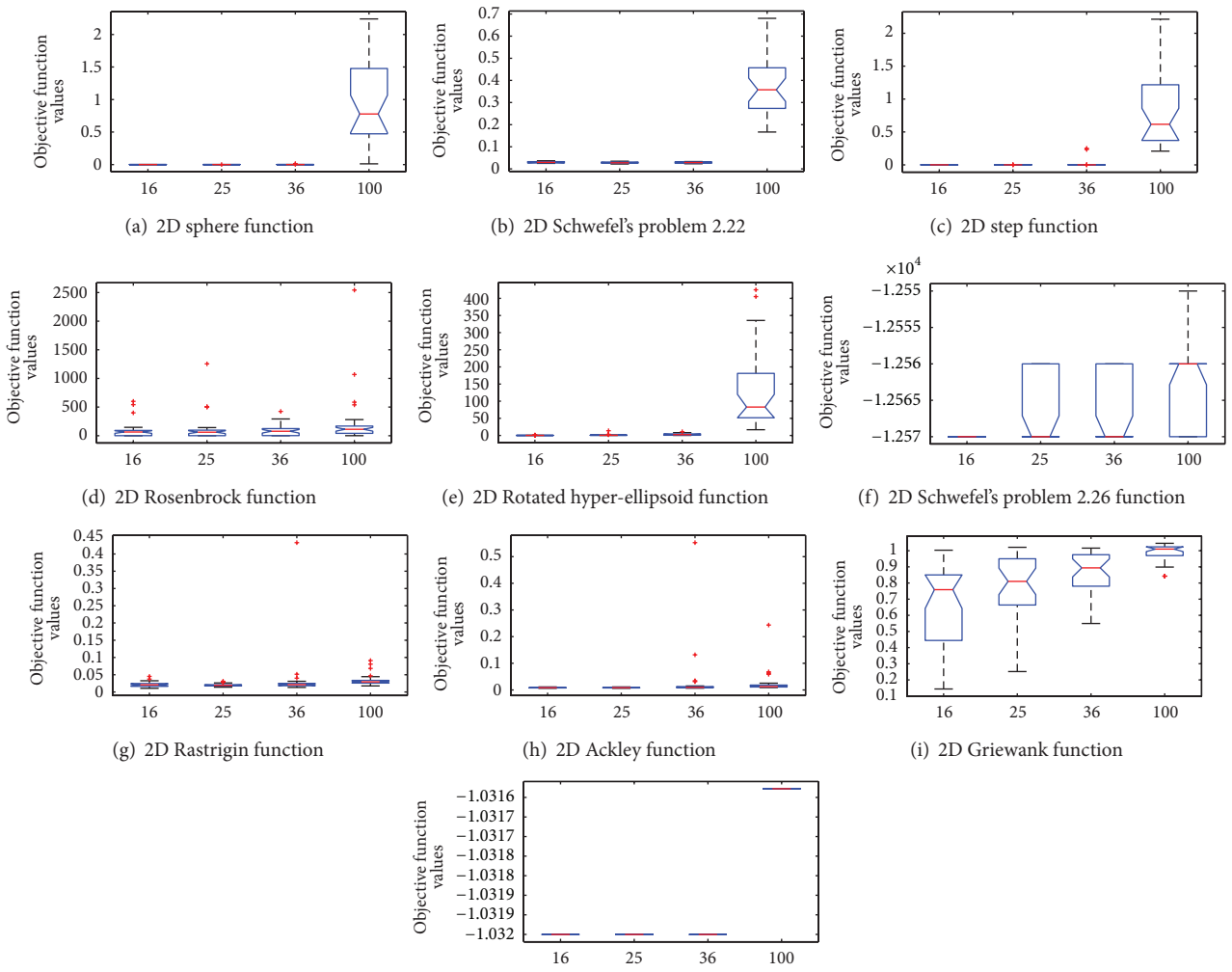
In this section, cHS algorithm is evaluated using benchmark functions circulated in the literature used to evaluate different variations of HS algorithm. The comparative evaluation is demonstrated while the sensitivity analysis of the control parameters of the proposed method is carried out.

4.1. Comparative Evaluation. In this section, a set of test functions designed for the special session on real-parameter optimization organized in the 2005 IEEE Congress on Evolutionary Computation (CEC 2005) [30], is used. The CEC 2005 comprises 25 test functions including 5 unimodals and 20 multimodals functions, as shown in Table 1. Note that a full discussion about these functions can be taken from Sunganthan et al. [30]. The CEC 2005 provides a suitable number of comparative methods in which the proposed method can be evaluated, as abbreviated in Table 2.

The experiments done followed the conditions of CEC 2005 [30], where the 25 repeated runs have been performed for each test function. The 25 runs have been summarized in terms of average of the error (AE) of the best individual (i.e., AE $= |f(\mathbf{x}^*) - f(\mathbf{x}^{\text{best}})|$). Note that $\mathbf{x}^*$ is a given optimal solution while the $\mathbf{x}^{\text{best}}$ is the average best solution obtained in 25 runs. The dimension $N = 10$ and the cHS is iterated 100,000 evaluations of the fitness function.

Notably, most of the winner comparative methods are hybrid versions of a particular EA, where their results are very efficient to the tested functions. It is appeared that the AE of HS and cHS are very close or sometimes better than those achieved by the comparative methods. In particular, the HS algorithm is able to achieve very powerful results for most test functions and excels some of the best results reported by the comparative methods. For example, HS has achieved the smallest AE for f_1, f_2, f_9, f_{14}, and f_{23}.

It is noted that the AE obtained by cHS is not the best in most cases. This is because the main idea of proposing



(a) 2D sphere function
(b) 2D Schwefel's problem 2.22
(c) 2D step function
(d) 2D Rosenbrock function
(e) 2D Rotated hyper-ellipsoid function
(f) 2D Schwefel's problem 2.26 function
(g) 2D Rastrigin function
(h) 2D Ackley function
(i) 2D Griewank function
(j) Six-Hump Camel-Back function

Figure 9: The box plots for showing the effect varying HMS values using the ten global optimization functions.

Table 2: Key to CEC'2005 comparative methods.

Key	Method name	Reference
BLXGL50	Hybrid real-coded genetic algorithms with female and male differentiation	[31]
BLX-MA	Adaptive local search parameters for real-coded memetic algorithms	[32]
CoEVO	Real-parameter optimization using the mutation step coevolution	[33]
DE	Real-parameter optimization with differential evolution	[34]
DMS-L-PSO	Dynamic multiswarm particle swarm optimizer with local search	[35]
EDA	Experimental results for the special session on real-parameter optimization at CEC'2005 : a simple, continuous EDA	[36]
K-PCX	A population-based, steady-state procedure for real-parameter optimization	[37]
G-CMA-ES	A restart CMA evolution strategy with increasing population size	[38]
L-CMA-ES	Performance evaluation of an advanced local search evolutionary algorithm	[39]
L-SaDE	Self-adaptive differential evolution algorithm	[40]
SPC-PNX	Real-parameter optimization performance study on the CEC-2005 benchmark with SPC-PNX	[41]

Table 3: Average error rate obtained in CEC'2005 special session in dimension 10.

Algorithm	$F1$	$F2$	$F3$	$F4$	$F5$	$F6$	$F7$	$F8$	$F9$
BLX-GL50	$1.00E-009$	$1.00E-009$	$5.71E+002$	$1.00E-009$	$1.00E-009$	$1.00E-009$	$1.17E-002$	$2.04E+001$	$1.15E+000$
BLX-MA	$1.00E-009$	$1.00E-009$	$4.77E+004$	$2.00E-008$	$2.12E-002$	$1.49E+000$	$1.97E-001$	$2.02E+001$	$4.38E-001$
COEVO	$1.00E-009$	$1.00E-009$	$1.00E-009$	$1.00E-009$	$2.13E+000$	$1.25E+001$	$3.71E-002$	$2.03E+001$	$1.92E+001$
DE	$1.00E-009$	$1.00E-009$	$1.94E-006$	$1.00E-009$	$1.00E-009$	$1.59E-001$	$1.46E-001$	$2.04E+001$	$9.55E-001$
DMS-L-PSO	$1.00E-009$	$1.00E-009$	$1.00E-009$	$1.89E-003$	$1.14E-006$	$6.89E-008$	$4.52E-002$	$2.00E+001$	$1.00E-009$
EDA	$1.00E-009$	$1.00E-009$	$2.12E+001$	$1.00E-009$	$1.00E-009$	$4.18E-002$	$4.20E-001$	$2.03E+001$	$5.42E+000$
IPOP-CMA-ES	$1.00E-009$	$1.00E-009$	$1.00E-009$	$1.00E-009$	$1.00E-009$	$1.00E-009$	$1.00E-009$	$2.00E+001$	$2.39E-001$
K-PCX	$1.00E-009$	$1.00E-009$	$4.15E-001$	$7.94E-007$	$4.85E+001$	$4.78E-001$	$2.31E-001$	$2.00E+001$	$1.19E-001$
LR-CMA-ES	$1.00E-009$	$1.00E-009$	$1.00E-009$	$1.76E+006$	$1.00E-009$	$1.00E-009$	$1.00E-009$	$2.00E+001$	$4.49E+001$
L-SADE	$1.00E-009$	$1.00E-009$	$1.67E-005$	$1.42E-005$	$1.23E-002$	$1.20E-008$	$1.99E-002$	$2.00E+001$	$1.00E-009$
SPC-PNX	$1.00E-009$	$1.00E-009$	$1.08E+005$	$1.00E-009$	$1.00E-009$	$1.89E+001$	$8.26E-002$	$2.10E+001$	$4.02E+000$
HS	$1.00E-009$	$1.00E-009$	$5.20E+004$	$9.63E+000$	$1.40E-003$	$5.42E-002$	$1.82E-001$	$2.02E+001$	$1.00E-009$
cHS	$1.00E-009$	$1.00E-009$	$1.49E+007$	$4.66E+003$	$1.33E+004$	$5.84E+007$	$7.85E+001$	$2.03E+001$	$2.69E+001$
Algorithm	$F10$	$F11$	$F12$	$F13$	$F14$	$F15$	$F16$	$F17$	$F18$
BLX-GL50	$4.97E+000$	$2.33E+000$	$4.07E+002$	$7.50E-001$	$2.17E+000$	$4.00E+002$	$9.35E+001$	$1.09E+002$	$4.20E+002$
BLX-MA	$5.64E+000$	$4.56E+000$	$7.43E+001$	$7.74E-001$	$2.03E+000$	$2.70E+002$	$1.02E+002$	$1.27E+002$	$8.03E+002$
COEVO	$2.68E+001$	$9.03E+000$	$6.05E+002$	$1.14E+000$	$3.71E+000$	$2.94E+002$	$1.77E+002$	$2.12E+002$	$9.02E+002$
DE	$1.25E+001$	$8.47E-001$	$3.17E+001$	$9.77E-001$	$3.45E+000$	$2.59E+002$	$1.13E+002$	$1.15E+002$	$4.00E+002$
DMS-L-PSO	$3.62E+000$	$4.62E+000$	$2.40E+000$	$3.69E-001$	$2.36E+000$	$4.85E+000$	$9.48E+001$	$1.10E+002$	$7.61E+002$
EDA	$5.29E+000$	$3.94E+000$	$4.42E+002$	$1.84E+000$	$2.63E+000$	$3.65E+002$	$1.44E+002$	$1.57E+002$	$4.83E+002$
IPOP-CMA-ES	$7.96E-002$	$9.34E-001$	$2.93E+001$	$6.96E-001$	$3.01E+000$	$2.28E+002$	$9.13E+001$	$1.23E+002$	$3.32E+002$
K-PCX	$2.39E-001$	$6.65E+000$	$1.49E+002$	$6.53E-001$	$2.35E+000$	$5.10E+002$	$9.59E+001$	$9.73E+001$	$7.52E+002$
LR-CMA-ES	$4.08E+001$	$3.65E+000$	$2.09E+002$	$4.94E-001$	$4.01E+000$	$2.11E+002$	$1.05E+002$	$5.49E+002$	$4.97E+002$
L-SADE	$4.97E+000$	$4.89E+000$	$4.50E-007$	$2.20E-001$	$2.92E+000$	$3.20E+001$	$1.01E+002$	$1.14E+002$	$7.19E+002$
SPC-PNX	$7.30E+000$	$1.91E+000$	$2.60E+002$	$8.38E-001$	$3.05E+000$	$2.54E+002$	$1.10E+002$	$1.19E+002$	$4.40E+002$
HS	$5.97E+000$	$1.47E+000$	$2.67E+005$	$5.93E-002$	$1.95E+000$	$1.00E-009$	$1.01E+002$	$1.06E+002$	$8.81E+002$
cHS	$6.17E+001$	$9.74E+000$	$1.28E+004$	$3.90E+000$	$3.97E+000$	$3.476E+02$	$2.73E+002$	$2.89E+002$	$8.85E+002$
Algorithm	$F19$	$F20$	$F21$	$F22$	$F23$	$F24$	$F25$		
BLX-GL50	$4.49E+002$	$4.46E+002$	$6.89E+002$	$7.59E+002$	$6.39E+002$	$2.00E+002$	$4.04E+002$		
BLX-MA	$7.63E+002$	$8.00E+002$	$7.22E+002$	$6.71E+002$	$9.27E+002$	$2.24E+002$	$3.96E+002$		
COEVO	$8.45E+002$	$8.63E+002$	$6.35E+002$	$7.79E+002$	$8.35E+002$	$3.14E+002$	$2.57E+002$		
DE	$4.20E+002$	$4.60E+002$	$4.92E+002$	$7.18E+002$	$5.72E+002$	$2.00E+002$	$9.23E+002$		
DMS-L-PSO	$7.14E+002$	$8.22E+002$	$5.36E+002$	$6.92E+002$	$7.30E+002$	$2.24E+002$	$3.66E+002$		
EDA	$5.64E+002$	$6.52E+002$	$4.84E+002$	$7.71E+002$	$6.41E+002$	$2.00E+002$	$3.73E+002$		
IPOP-CMA-ES	$3.26E+002$	$3.00E+002$	$5.00E+002$	$7.29E+002$	$5.59E+002$	$2.00E+002$	$3.74E+002$		
K-PCX	$7.51E+002$	$8.13E+002$	$1.05E+003$	$6.59E+002$	$1.06E+003$	$4.06E+002$	$4.06E+002$		
LR-CMA-ES	$5.16E+002$	$4.42E+002$	$4.04E+002$	$7.40E+002$	$7.91E+002$	$8.65E+002$	$4.42E+002$		
L-SADE	$7.05E+002$	$7.13E+002$	$4.64E+002$	$7.35E+002$	$6.64E+002$	$2.00E+002$	$3.76E+002$		
SPC-PNX	$3.80E+002$	$4.40E+002$	$6.80E+002$	$7.49E+002$	$5.76E+002$	$2.00E+002$	$4.06E+002$		
HS	$7.54E+002$	$8.00E+002$	$8.53E+002$	$7.42E+002$	$5.59E+002$	$2.36E+002$	$3.29E+002$		
cHS	$1.09E+003$	$1.08E+003$	$1.09E+003$	$9.00E+002$	$1.30E+003$	$1.06E+003$	$8.84E+002$		

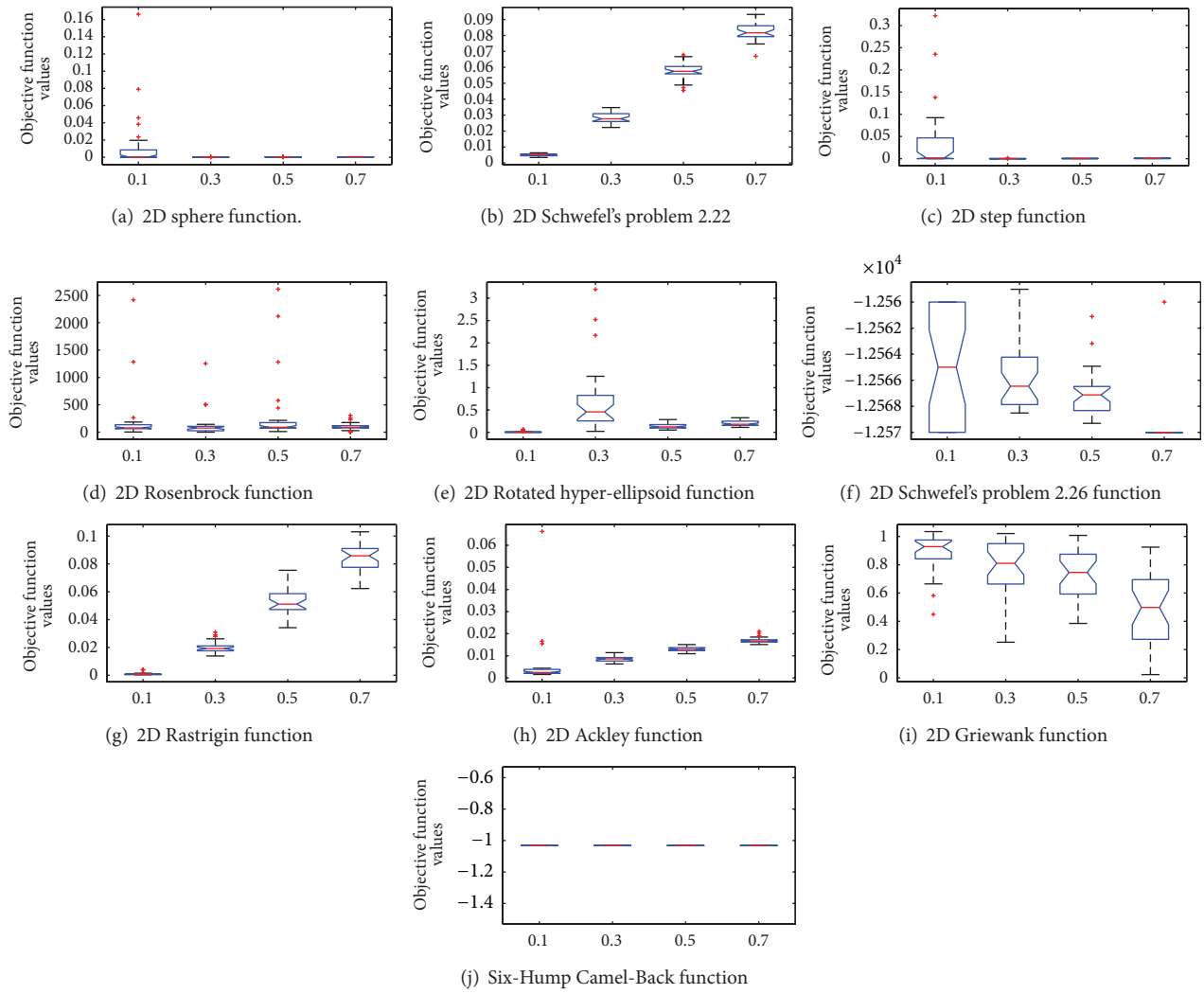


FIGURE 10: The box plots for showing the effect varying PAR values using the ten global optimization functions.

the cHS is to ensure a high-level of diversity, and thus, the cHS required a higher number of evaluations to hit better results. However, the results obtained by cHS are very close to the winner results.

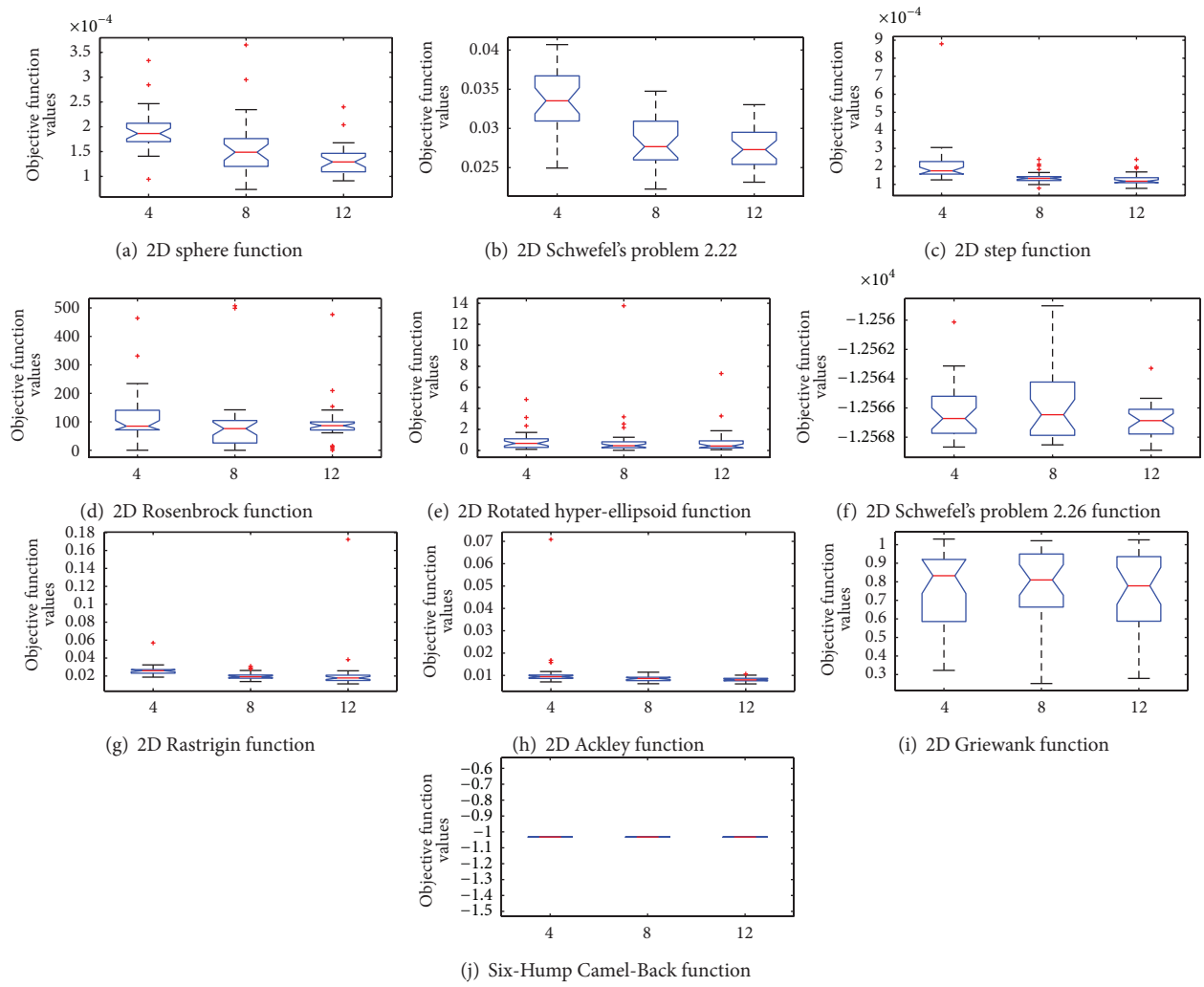
4.2. Sensitivity Analysis of cHS Parameters. All the experiments are run using a computer with 2.66 Intel Core 2 Quad with 4 GB of RAM. The operating system used is Microsoft windows Vista Enterprise Service Pack 1. The source code is implemented using MATLAB Version 7.6.0.324(R2008a).

The common parameters among all algorithms used in the experiments are set based on empirical guidelines [2]. For the sake of studying the effect of different parameter settings, in general, the parameters setting used for evaluating the cHS method are as follows: HMS = 25, HMCR = 0.98, PAR = 0.3, $N = 30$, NH = 9, and NI = 5×10^4. The neighborhood shape used is C9, presented in Figure 5.

All functions are implemented in 30 dimensions (30D). For the scalability study in Section 4.2.6, the functions are implemented in 10 dimensions (10D), 50 dimensions (50D), and 100 dimensions (100D). The stopping criterion used is the maximum number of improvisation (NI).

All the results in this section are presented in Tables 4 to 8, demonstrating the mean and standard deviation for thirty independent runs. The best solution has been highlighted in bold font. A comparative analysis between cHS algorithm and the original variant of HS for the common parameters is also conducted.

4.2.1. Benchmark Functions. The global minimization benchmark functions are used to study the sensitivity analysis of the parameters of the proposed method (cHS) against the original version of HS algorithm. Five functions are defined by Whitley et al. [42] and the other five were described by Yao et al. [43]. These functions provide a balance between unimodal and multimodal functions. These functions are commonly used to evaluate the state-of-the-art variations of harmony search algorithms [5, 44, 45].



(a) 2D sphere function
(b) 2D Schwefel's problem 2.22
(c) 2D step function
(d) 2D Rosenbrock function
(e) 2D Rotated hyper-ellipsoid function
(f) 2D Schwefel's problem 2.26 function
(g) 2D Rastrigin function
(h) 2D Ackley function
(i) 2D Griewank function
(j) Six-Hump Camel-Back function

FIGURE 11: The box plots for showing the effect varying NH values using the ten global optimization functions.

Most of the benchmark functions have standard solution space range of the objective function. Otherwise, unsymmetrical initialization ranges are used for these functions whose global optima are at the center of the solution space. These benchmark functions are as follows.

(1) *Sphere* function, defined as

$$f_1(x) = \sum_{i=1}^{N_d} x_i^2, \tag{9}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-100 \le x_i \le 100$) (Figure 7(a)).

(2) *Schwefel's* problem 2.22, defined as

$$f_2(x) = \sum_{i=1}^{N_d} |x_i| + \prod_{i=1}^{N_d} |x_i|, \tag{10}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-10 \le x_i \le 10$) (Figure 7(b)).

(3) *Step* function, defined as

$$f_3(x) = \sum_{i=1}^{N_d} (\lfloor x_i + 0.5 \rfloor)^2, \tag{11}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-100 \le x_i \le 100$) (Figure 7(c)).

(4) *Rosenbrock* function, defined as

$$f_4(x) = \sum_{i=1}^{N_d - 1} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right), \tag{12}$$

where global optimum $x^* = (1, 1, \ldots, 1)$ and $f(x^*) = 0$ (search range $-100 \le x_i \le 100$) (Figure 7(d)).

(5) *Rotated hyper-ellipsoid* function, defined as

$$f_5(x) = \sum_{i=1}^{N_d} \left(\sum_{j=1}^{i} x_j\right)^2, \tag{13}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-100 \le x_i \le 100$) (Figure 7(e)).

Set HMCR, PAR, NI, HMS, bw.
$x_i^j = LB_i + U(0,1) \times (UB_i - LB_i)$, $\forall i = 1, 2, \ldots, N$, $\forall j = 1, 2, \ldots,$ HMS, $U(0,1)$;
Calculate $(f(\acute{x}_i))$, $\forall j = 1, 2, \ldots,$ HMS;
Generate(**NM**) {generate the neighborhood matrix}
While !*StopCondition*() **do**
  select random individual $\mathbf{x}^r$
  $\acute{x} = \varnothing$;
  for $i = 1, \ldots, N$ **do**
    if $U(0,1) \leq$ HMS **then**
      $\acute{x}^i \in \xi_r$; {Memory Consideration}
      if $U(0,1) \leq$ PAR **then**
        $\acute{x}^i = \acute{x}^i + U(-1,1) \times bw$; {Pitch Adjustment}
      end if
    else
      $\acute{x}^i = LB_i + U(0,1) \times (UB_i - LB_i)$; {Random Consideration}
    end if
  end for
  find(x^{worst}) where $x^{\text{worst}} \in \xi_r$
  if $f(\acute{x}) < f(x^{\text{worst}})$ **then**
    include $\acute{x}$ to **HM**;
    exclude $\acute{x}^{\text{worst}}$ from **HM**;
  end if
end while

Algorithm 1: The pseudocode of cellular harmony search (cHS).

(6) *Generalized Schwefel's* problem 2.26, defined as

$$f_6(x) = -\sum_{i=1}^{N_d} \left(x_i \sin \sqrt{(|x_i|)} \right), \tag{14}$$

where global optimum $x^* = (420.9687, 420.9687, \ldots, \ldots, \ldots, 420.9687)$ and $f(x^*) = -12569.5$ (search range $-500 \leq x_i \leq 500$) (Figure 7(f)).

(7) *Rastrigin* function, defined as

$$f_7(x) = \sum_{i=1}^{N_d} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right), \tag{15}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-5.12 \leq x_i \leq 5.12$) (Figure 7(g)).

(8) *Ackley's* function, defined as

$$f_8(x) = -20\exp -0.2\sqrt{\left(\frac{1}{30}\sum_{i=1}^{N_d} x_i^2 \right)} - \exp\left(\frac{1}{30}\sum_{i=1}^{N_d} \cos(2\pi x_i) \right) + 20 + e, \tag{16}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-32 \leq x_i \leq 32$) (Figure 7(h)).

(9) *Griewank* function, defined as

$$f_9(x) = \frac{1}{4000}\sum_{i=1}^{N_d} x_i^2 - \prod_{i=1}^{N_d} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1, \tag{17}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-600 \leq x_i \leq 600$) (Figure 7(i)).

(10) *Six-Hump Camel-Back* function, defined as

$$f_{10}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \tag{18}$$

where global optimum $x = (-0.08983, 0.7126), (-0.08983, 0.7126)$ and $f(x^*) = 1.0316285$ (search range $-5 \leq x_i \leq 5$) (Figure 7(j)).

Figure 7 depicts the corresponding landscape of the search space of each function [46].

4.2.2. The Effect of HMCR. Table 4 summarizes the effect of HMCR on the performance of cHS and the original version of HS using four values of HMCR (i.e., 0.7, 0.9, 0.94, and 0.98). The best solution obtained at each corresponding value is highlighted in bold font. Figure 8 is the box plot visualizing the effect of various values of HMCR on the behavior of cHS algorithm using the ten global minimization functions.

The results show that increasing the HMCR value improves the performance of the cHS for all functions, except six-hump where the opposite function is true. Where a small value is used, HMCR increases the diversity and hence prevents the cHS from convergence (i.e., it results in random search). Thus, it is generally better to use a large value for the HMCR (i.e., ≤0.98).

The high value of HMCR means high probability of using the harmony memory that leads to less exploration of search space. Using a probability of HMCR close to 1 (high value) might lead the algorithm to fall into the local minima. Using less probability of HMCR allows more randomly generated

Table 4: The effect of varying the HMCR parameter on HS and cHS for ten functions (f_1–f_{10}).

Function	Alg.	0.7	0.9	0.94	0.98
Sphere	cHS	5.514E + 03	3.383E + 01	2.240E + 00	**1.323E − 04**
		(8.887E + 02)	(1.164E + 01)	(9.216E − 01)	(**3.285E − 05**)
	HS	3.750E + 03	1.663E + 01	1.152E + 00	6.086E − 03
		(6.455E + 02)	(4.716E + 00)	(7.069E − 01)	(1.406E − 03)
Schwefel's 2.22	cHS	1.939E + 01	2.584E − 01	5.397E − 02	**2.738E − 02**
		(1.966E + 00)	(1.006E − 01)	(8.303E − 03)	(**2.542E − 03**)
	HS	1.581E + 01	5.359E + 00	4.081E − 01	2.20087E − 01
		(2.380E + 00)	(1.036E + 00)	(4.914E − 01)	(2.49573E − 02)
Step	cHS	5.700E + 03	3.233E + 01	2.437E + 00	**1.288E − 04**
		(8.707E + 02)	(1.085E + 01)	(1.132E + 00)	(**3.421E − 05**)
	HS	3.949E + 03	1.605E + 01	8.041E + 00	6.510E − 03
		(6.671E + 02)	(4.439E + 00)	(5.844E − 01)	(1.019E − 03)
Rosenbrock	cHS	3.578E + 06	1.276E + 03	1.944E + 02	**9.892E + 01**
		(9.728E + 05)	(9.524E + 02)	(1.177E + 02)	(**8.330E + 01**)
	HS	2.143E + 06	5.444E + 02	1.449E + 02	1.489E + 02
		(5.423E + 05)	(2.240E + 02)	(7.499E + 01)	(1.22097E + 02)
Rotated hyper	cHS	7.206E + 05	5.588E + 03	4.250E + 01	**8.421E − 01**
		(1.122E + 05)	(2.071E + 03)	(2.364E + 02)	(**1.381E + 00**)
	HS	5.155E + 05	2.972E + 03	1.373E + 02	2.647E + 00
		(7.586E + 04)	(9.061E + 02)	(1.137E + 02)	(1.531E + 00)
Schwefel's 2.26	cHS	−1.075E + 04	−1.250E + 04	−1.256E + 04	**−1.257E + 04**
		(2.536E + 02)	(2.208E + 01)	(3.982E + 00)	(2.371E + 00)
	HS	−1.123E + 04	−1.253E + 04	−1.256E + 04	**−1.257E + 04**
		(2.643E + 02)	(1.389E + 01)	(2.465E + 00)	(**2.131E + 00**)
Rastrigin	cHS	8.581E + 01	4.029E + 00	7.068E − 02	2.380E − 02
		(8.587E + 00)	(1.780E + 00)	(9.444E − 02)	(2.85250E − 02)
	HS	6.612E + 01	1.548E + 00	1.289E − 01	**1.582E − 02**
		(8.693E + 00)	(1.198E + 00)	(3.037E − 01)	(**7.023E − 03**)
Ackley	cHS	1.295E + 01	2.522E + 00	1.891E − 01	**3.890E − 02**
		(6.804E − 01)	(3.160E − 01)	(2.418E − 01)	(**1.686E − 01**)
	HS	1.209E + 01	1.830E + 00	6.493E − 01	1.060E − 01
		(5.334E − 01)	(3.862E − 01)	(1.282E − 01)	(5.763E − 02)
Griewank	cHS	5.234E + 01	1.298E + 00	1.024E + 00	7.870E − 01
		(8.732E + 00)	(1.066E − 01)	(3.995E − 02)	(2.086E − 01)
	HS	3.895E + 01	1.166E + 00	1.013E + 00	**6.566E − 01**
		(6.677E + 00)	(5.312E − 02)	(1.920E − 02)	(**2.352E − 01**)
Six-Hump	cHS	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**
		(**2.882E − 10**)	(2.998E − 11)	(9.912E − 11)	(4.150E − 11)
	HS	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**
		(1.291E − 10)	(5.571E − 11)	(3.175E − 11)	(2.922E − 09)

solutions. Therefore, the diversity increases in a way that prevents the convergence. The results reveal that cHS and HS have identical sensitivity to the different values of HMCR for all functions, and in 0.98 probabilities to use HMCR they get the best results for the majority of optimization functions. Furthermore, the results produced by cHS are better than those produced by HS in almost all tested functions.

4.2.3. The Effect of HMS. Table 5 summarizes the effect of HMS on the performance of cHS and basic HS using four values of HMS (i.e., 16, 25, 36, and 100). The best solution obtained at each corresponding value is highlighted in bold font. Figure 9 is the box plot visualizing the effect of various values of HMS on the behavior of cHS algorithm using the ten global minimization functions.

It is revealed that both cHS and the basic HS are not sensitive to the HMS. For (f_4, f_6, f_{10}), cHS obtained the best results when the value of HMS is large (see Figure 9). Apparently, with different values of HMS, cHS performance is better for most of the functions because the overlap of

Table 5: The effect of varying the HMS parameter on HS and cHS for ten functions (f_1–f_{10}).

Function	Alg.	16	25	36	100
Sphere	cHS	$1.188E-04$	$1.323E-04$	$8.506E-04$	$6.950E-01$
		$(2.204E-05)$	$(3.285E-05)$	$(2.744E-03$	$(5.385E-01)$
	HS	$\mathbf{7.961E-05}$	$6.086E-03$	$3.934E-03$	$7.274E-01$
		$(\mathbf{1.511E-05})$	$(1.40654E-03)$	$(1.721E-02)$	$(5.510E-01)$
Schwefel's 2.22	cHS	$3.011E-02$	$2.738E-02$	$2.869E-02$	$3.850E-01$
		$(3.522E-03)$	$(2.542E-03)$	$(3.043E-03)$	$(1.391E-01)$
	HS	$\mathbf{2.208E-02}$	$2.200E-01$	$2.447E-02$	$3.210E-02$
		$(\mathbf{2.658E-03})$	$(2.495E-02)$	$(5.482E-03)$	$(3.840E-03)$
Step	cHS	$1.225E-04$	$1.288E-04$	$1.637E-02$	$7.698E-01$
		$(2.373E-05)$	$(3.421E-05)$	$(6.139E-02)$	$(5.098E-01)$
	HS	$\mathbf{7.988E-05}$	$6.510E-03$	$3.161E-03$	$8.057E-01$
		$(\mathbf{1.123E-05})$	$(1.019E-03)$	$(9.481E-03)$	$(4.692E-01)$
Rosenbrock	cHS	$1.106E+02$	$1.328E+02$	$1.025E+02$	$2.738E+02$
		$(1.449E+02)$	$(2.425E+02)$	$(8.530E+01)$	$(4.751E+02)$
	HS	$8.418E+02$	$\mathbf{9.892E+01}$	$1.148E+02$	$2.573E+02$
		$(7.497E+02)$	$(\mathbf{1.220E+02})$	$(1.278E+02)$	$(4.789E+02)$
Rotated hyper	cHS	$\mathbf{2.604E-02}$	$8.421E-01$	$2.883E+00$	$1.649295E+01$
		$(\mathbf{3.042E-02})$	$(1.381E+00)$	$(2.572E+00)$	$(2.234042E+01)$
	HS	$1.661E-01$	$2.647E+00$	$1.808E+00$	$1.259E+02$
		$(1.835E-01)$	$(1.531E+00)$	$(4.377E+00)$	$(1.060E+02)$
Schwefel's 2.26	cHS	$-1.075E+04$	$-1.256E+04$	$-1.256E+04$	$\mathbf{-1.257E+04}$
		$(1.163E+00)$	$(1.190E+00)$	$(2.122E+00)$	$(3.050E+00)$
	HS	$\mathbf{-1.257E+04}$	$\mathbf{-1.257E+04}$	$-1.256E+04$	$-1.256E+04$
		$(\mathbf{1.209E+00})$	$(1.300E+00)$	$(2.125E+00)$	$(2.263E+00)$
Rastrigin	cHS	$\mathbf{2.159E-02}$	$2.380E-02$	$3.625E-02$	$1.010E-01$
		$(\mathbf{7.348E-03})$	$(2.852E-02)$	$(7.533E-02)$	$(2.517E-01)$
	HS	$3.268E-02$	$1.418E+01$	$5.107E-02$	$2.503E-02$
		$(1.956E-03)$	$(2.376E+00)$	$(1.816E-01)$	$(8.612E-03)$
Ackley	cHS	$\mathbf{3.495E-03}$	$3.890E-02$	$3.369E-02$	$2.636E-02$
		$(\mathbf{2.452E-03})$	$(1.686E-01)$	$(1.006E-01)$	$(4.388E-02)$
	HS	$6.760E-03$	$1.060E-01$	$2.313E-02$	$8.757E-02$
		$(8.735E-04)$	$(5.763E-02)$	$(8.375E-02)$	$(1.395E-01)$
Griewank	cHS	$6.481E-01$	$7.437E-01$	$8.571E-01$	$9.878E-01$
		$(2.614E-01)$	$(2.086E-01)$	$(1.486E-01)$	$(5.487E-02)$
	HS	$\mathbf{5.010E-01}$	$6.566E-01$	$8.020E-01$	$9.907E-01$
		$(\mathbf{2.409E-01})$	$(2.352E-01)$	$(1.658E-01)$	$(5.565E-02)$
Six-Hump	cHS	$-1.031E+00$	$-1.031E+00$	$-1.031E+00$	$\mathbf{-1.032E+00}$
		$(5.325E-11)$	$(4.15028E-11)$	$(6.052E-11)$	$(\mathbf{2.001E-11})$
	HS	$-1.031E+00$	$-1.031E+00$	$-1.031E+00$	$-1.030E+00$
		$(1.897E-11)$	$(2.922E-09)$	$(4.793E-10)$	$(2.235E-10)$

the neighborhoods provides an implicit mechanism of migration to the cHS. Since the best solutions spread smoothly through the whole population, the cHS diversity in the structured population is preserved longer than in the classical version of HS algorithm.

HM is analogous to the short-term memory of a musician that is known to be small. A plausible interpretation may rely on the high number of similar harmonies within the HM when the HMS is large that leads to shortages of diversity and, hence, lead to falling into local minima. Therefore, cHS is likely to be capable of maintaining the diversity than HS with the cellular structure.

4.2.4. The Effect of PAR. Table 6 summarizes the effect of PAR on the performance of cHS and basic HS using four values of PAR (0.1, 0.3, 0.7, and 0.9). The best solution at each corresponding value is highlighted in bold font. Figure 10 is the box plot visualizing the effect of various values of PAR on the behavior of cHS algorithm using the ten global minimization functions.

Table 6: The effect of varying the PAR parameter on HS and cHS for ten functions (f_1–f_{10}).

Function	Alg.	0.1	0.3	0.7	0.9
Sphere	cHS	$1.343E-02$	**1.323E − 04**	$3.136E-04$	$4.713E-04$
		$(3.391E-02)$	**(3.285E − 05)**	$(4.066E-05)$	$(5.207E-05)$
	HS	$2.012E-02$	$6.086E-03$	$3.827E-04$	$5.497E-04$
		$(6.098E-02)$	$(1.406E-03)$	$(5.318E-05)$	$(7.251E-05)$
Schwefel's 2.22	cHS	$4.965E-03$	$2.738E-02$	$5.784E-02$	$2.827E-02$
		$(7.512E-04)$	$(2.542E-03)$	$(5.463E-03)$	$(3.017E-03)$
	HS	**3.518E − 03**	$2.200E-01$	$7.583E-02$	$9.548E-02$
		(5.837E − 04)	$(2.495E-02)$	$(5.482E-03)$	$(4.834E-03)$
Step	cHS	$5.526E-02$	**1.288E − 04**	$2.929E-04$	$4.606E-04$
		$(1.177E-01)$	**(3.421E − 05)**	$(3.776E-05)$	$(5.136268E-05)$
	HS	$1.682E-02$	$6.510E-03$	$3.753E-04$	$5.291E-04$
		$(3.939E-02)$	$(1.019E-03)$	$(5.417E-05)$	$(4.401E-05)$
Rosenbrock	cHS	$2.065E+02$	$9.892E+01$	$3.114E+02$	$1.030E+02$
		$(4.747E+02)$	$(8.330E+01)$	$(6.121E+02)$	$(6.798E+01)$
	HS	$1.813E+02$	$1.489E+02$	$1.312E+02$	**8.301E + 01**
		$(3.929E+02)$	$(1.220E+02)$	$(2.515E+02)$	**(5.748E + 01)**
Rotated hyper	cHS	**1.038E − 02**	$8.421E-01$	$1.323E-01$	$1.970E-01$
		(1.603E − 02)	$(1.381E+00)$	$(5.359E-02)$	$(5.845E-02)$
	HS	$5.404E+00$	$2.647E+00$	$4.726E-01$	$4.660E-01$
		$(1.507E+01)$	$(1.531E+00)$	$(3.248E-01)$	$(3.479E-01)$
Schwefel's 2.26	cHS	**−1.257E + 04**	$-1.256E+04$	$-1.256E+04$	**−1.257E + 04**
		$(2.536E+02)$	$(1.190E+00)$	$(3.982E+00)$	$(2.371E+00)$
	HS	**−1.257E + 04**	$-1.256E+04$	**−1.257E + 04**	**−1.257E + 04**
		$(2.092E+00)$	$(1.393E+00)$	**(1.082E + 00)**	$(1.151E+00)$
Rastrigin	cHS	**9.355E − 04**	$2.380E-02$	$5.248E-02$	$9.029E-02$
		(8.827E − 04)	$(2.852E-02)$	$(9.304E-03)$	$(3.178E-02)$
	HS	$3.468E-03$	$1.418E+01$	$7.952E-02$	$1.092E-01$
		$(1.763E-03)$	$(2.376E+00)$	$(1.720E-02)$	$(1.377E-02)$
Ackley	cHS	**5.655E − 03**	$3.890E-02$	$5.730E-02$	$3.827E-02$
		(1.197E − 02)	$(1.686E-01)$	$(1.815E-01)$	$(1.152E-03)$
	HS	$(1.878E-02)$	$1.060E-01$	$1.585E-02$	$1.804E-02$
		$(8.563E-02)$	$(5.763E-02)$	$(1.606E-03)$	$(1.168E-01)$
Griewank	cHS	$8.866E-01$	$7.437E-01$	$7.310E-01$	$4.982E-01$
		$(1.306E-01)$	$(2.086E-01)$	$(1.826E-01)$	$(2.712E-01)$
	HS	$8.542E-01$	$6.566E-01$	$4.912E-01$	**3.853E − 01**
		$(1.270E-01)$	$(2.352E-01)$	$(2.591E-01)$	**(2.413E − 01)**
Six-Hump	cHS	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**
		$(1.496E-10)$	$(4.150E-11)$	$(3.543E-11)$	**(3.197E − 11)**
	HS	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**
		$(1.897E-10)$	$(2.922E-09)$	$(4.793E-11)$	$(2.235E-10)$

It seems that using a relatively small value of PAR (i.e., ≤0.5) improves the performance of the cHA and HS. Most results at large value of PAR can increase the convergence speed of HS algorithm, while a small value of PAR increases diversity in HM. On the other hand, the small value of PAR allows more exploration of the search space, and the large value of PAR leads to a lower rate of exploration, where the diversity is reduced and then the algorithm might be trapped into the local optima. It is observed that cHS is able to get the best results than the original version of HS algorithm for

Table 7: The effect of varying the number of neighbors on cHS for ten functions ($f_1 - f_{10}$).

Function	NH = 4	NH = 8	NH = 12
Sphere	$5.338E-03$	$1.587E-04$	**1.32E − 04**
	$(2.819E-02)$	$(5.907E-05)$	(**3.29E − 05**)
Schwefel's 2.22	$3.331E-02$	$2.827E-02$	**2.74E − 02**
	$(3.789E-03)$	$(3.017E-03)$	(**2.54E − 03**)
Step	$2.074E-04$	$1.761E-04$	**1.29E − 04**
	$(1.339E-04)$	$(2.058E-04)$	(**3.42E − 05**)
Rosenbrock	$1.169E+02$	$1.329E+02$	**9.89E + 01**
	$(9.558E+01)$	$(2.425E+02)$	(**8.33E + 01**)
Rotated hyper-ellipsoid	$1.945E+00$	$1.119E+00$	**8.42E − 01**
	$(5.520E+00)$	$(2.500E+00)$	(**1.38E + 00**)
Schwefel's 2.26	**−1.257E + 04**	**−1.257E + 04**	$-1.26E+04$
	(**1.901E + 00**)	$(2.371E+00)$	$(1.19E+00)$
Rastrigin	$3.548E-02$	**2.021E − 02**	$2.38E-02$
	$(5.073E-02)$	(**4.141E − 03**)	$(2.85E-02)$
Ackley	$2.901E-02$	**8.698E − 03**	$3.89E-02$
	$(6.715E-02)$	(**1.152E − 03**)	$(1.69E-01)$
Griewank	$7.530E-01$	$7.870E-01$	**7.44E − 01**
	$(2.071E-01)$	$(1.903E-01)$	(**2.09E − 01**)
Six-Hump Camel-Back	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**
	$(5.473E-11)$	(**2.821E − 11**)	$(4.15E-11)$

some benchmark functions (i.e., f_4, f_2, f_8, f_9), but not so for the others.

4.2.5. The Effect Number of Neighbors NH. Table 7 summarizes the effect number of neighbors (NH) according to cellular structure on the performance of cHS. Table 7 also exhibits the effect of NH for (f_1 to f_{10}) using four values of NH (4, 8, 12), where the value of HMS = 25. The best solution at each corresponding value is highlighted in bold font. The best results of comparing cHS with original version HS are highlighted in bold. Figure 11 is the box plot visualizing the effect of various NH values on the behavior of cHS algorithm using the ten global minimization functions.

The results show that cHS obtained the best result when the number of neighbors is large, except (f_6) which obtained the best result when the number of neighbors is small. Generally, this leads to the conclusion that cHS often shows better performance at larger values of NH between 8 and 12. Figure 11 shows the box plots of the recorded results which reveal the distribution of the 30 tested runs against the various value of NH.

4.2.6. Scalability Study of N. In this section, the results produced by cHS and HS, when the dimension of the function is set to $N = 10$, $N = 30$, $N = 50$, and $N = 100$ as shown in Table 8, are recorded.

In general, decreasing the dimensionality leads to better results in cHS and HS. This comes in line with the previous theory. However, it is observed from results (Table 8) that the proposed cHS algorithm outperforms the HS when dimensionality is large for the majority of the benchmark optimization functions. This shows that increasing the dimensionality of the problem needs a better algorithm like cHS.

5. Conclusion and Future Work

In this paper, a new version of HS algorithm called cellular harmony search (cHS) algorithm is proposed. cHS is an HS algorithm embedded with cellular automata concepts. The main idea of proposing cHS algorithm is to provide a structured population that preserves a high level of diversity during the search. In cHS, the HM individuals are arranged as a two-dimensional toroidal grid, where each individual is generated and only interacts with its neighboring individuals. The operators of the original version of HS algorithm are adjusted to observe the cellular GA theory, where the concepts of cell and cell search space are employed.

Using ten global optimization functions circulated in the literature, the cHS is evaluated. The results support the theory of cellular automata, where in almost all cases the cHS outperforms HS algorithm. The sensitivity analysis of cHS parameters has suggested that the cHS is sensitive to the values of HMCR, PAR, N, and NH. The comparative evaluation is also conducted with two versions of HS algorithms proposed in [44, 45], where comparable results were obtained.

This is an initial investigation of using cellular automata concepts in the HS algorithm optimization framework.

Table 8: Mean and standard deviation of ten functions (f_1–f_{10}) when $N = (10, 30, 50, 100)$.

Function	Alg.	$N = 10$	$N = 30$	$N = 50$	$N = 100$
Sphere	cHS	$7.022E-08$	$1.323E-04$	$4.776E+02$	$2.036E+04$
		$(4.079E-08)$	$(3.285E-05)$	$(9.256E+01)$	$(2.584E+03)$
	HS	**3.033E − 08**	$6.086E-03$	$8.746E+02$	$1.591E+04$
		(1.842E − 08)	$(1.406E-03)$	$(1.484E+02)$	$(2.471E+03)$
Schwefel's 2.22	cHS	$3.548E-04$	$2.738E-02$	$8.810E+00$	$8.500E+01$
		$(1.327E-04)$	$(2.542E-03)$	$(9.866E-01)$	$(8.610E+00)$
	HS	**2.725E − 04**	$2.200E-01$	$1.050E+01$	$5.479E+01$
		(6.537E − 05)	$(2.495E-02)$	$(1.323E+00)$	$(3.951E+00)$
Step	cHS	$6.297E-08$	$1.288E-04$	$3.771E+02$	$1.960E+04$
		$(3.292E-08)$	$(3.42100E-05)$	$(1.029E+02)$	$(2.260E+03)$
	HS	**3.501E − 08**	$6.510E-03$	$7.214E+02$	$1.551E+04$
		(2.873E − 08)	$(1.019E-03)$	$(1.364E+02)$	$(1.169E+03)$
Rosenbrock	cHS	$3.758E+01$	$9.892E+01$	$2.090E+04$	$1.359E+07$
		$(7.344E+01)$	$(8.330E+01)$	$(9.488E+03)$	$(2.397E+06)$
	HS	**1.651E + 01**	$1.489E+02$	$9.093E+04$	$1.730E+07$
		(3.809E + 01)	$(1.220E+02)$	$(2.346E+04)$	$(2.195E+06)$
Rotated hyper-ellipsoid	cHS	$2.351E-06$	$8.421E-01$	$1.993E+05$	$3.097E+07$
		$(1.799E-06)$	$(1.381E+00)$	$(4.916E+04)$	$(3.408E+06)$
	HS	**8.554E − 07**	$2.647E+00$	$2.825E+05$	$2.723E+07$
		(7.059E − 07)	$(1.531E+00)$	$(4.035E+04)$	$(3.321E+06)$
Schwefel's 2.26	cHS	**−4.190E + 03**	$-1.256E+04$	$9.019E+02$	$8.874E+03$
		$(1.040E-01)$	$(1.190E+00)$	$(2.182E+02)$	$(8.167E+02)$
	HS	**−4.190E + 03**	$-1.256E+04$	$8.594E+02$	$5.119E+03$
		(3.861E − 03)	$(1.393E+00)$	$(1.645E+02)$	$(5.619E+02)$
Rastrigin	cHS	$8.538E-06$	$2.380E-02$	$3.594E+01$	$3.575E+02$
		$(4.752E-06)$	$(2.852E-02)$	$(4.537E+00)$	$(4.352E+01)$
	HS	**4.938E − 06**	$1.418E+01$	$4.766E+01$	$2.325E+02$
		(5.215E − 06)	$(2.376E+00)$	$(4.308E+00)$	$(1.703E+01)$
Ackley	cHS	**9.366E − 06**	$3.890E-02$	$5.168E+00$	$1.385E+01$
		(4.385E − 06)	$(1.686E-01)$	$(4.202E-01)$	$(3.524E-01)$
	HS	$2.101E-04$	$1.060E-01$	$6.197E+00$	$1.277E+01$
		$(1.146E-04)$	$(5.763E-02)$	$(3.790E-01)$	$(3.405E-01)$
Griewank	cHS	$9.958E-02$	$7.437E-01$	$5.828E+00$	$1.931E+02$
		$(4.819E-02)$	$(2.086E-01)$	$(1.398E+00)$	$(2.241E+01)$
	HS	**7.082E − 02**	$6.566E-01$	$8.641E+00$	$1.470E+02$
		(4.053E − 02)	$(2.352E-01)$	$(1.330E+00)$	$(1.583E+01)$
Six-Hump Camel-Back	cHS	**−1.032E + 00**	$-1.03163E+00$	**−1.032E + 00**	**−1.032E + 00**
		$(3.927E-11)$	$(4.150E-11)$	**(0)**	**(0)**
	HS	**−1.032E + 00**	$-1.03163E+00$	**−1.032E + 00**	**−1.032E + 00**
		$(6.526E-11)$	$(2.922E-09)$	**(0)**	**(0)**

The future, indeed, is pregnant with several research directions, such as

(i) analyzing the selection pressure and time complexity concepts of cHS algorithm,

(ii) studying the effect of the neighborhood shapes on the performance of cHS,

(iii) studying a new migration strategy to empower the interaction between the individuals and their neighbors.

Acknowledgment

The first author is grateful to be awarded Postdoctoral Research Fellowship from the School of Computer Sciences (USM). This paper is supported by Grant no. 203/PTS6728001 (Grant Type: LRGS).

References

[1] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[2] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[3] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[4] Z. W. Geem, "Novel derivative of harmony search algorithm for discrete design variables," *Applied Mathematics and Computation*, vol. 199, no. 1, pp. 223–230, 2008.

[5] M. A. Al-Betar, I. A. Doush, A. T. Khader, and M. A. Awadallah, "Novel selection schemes for harmony search," *Applied Mathematics and Computation*, vol. 218, no. 10, pp. 6095–6117, 2012.

[6] M. A. Al-Betar, A. T. Khader, and M. Zaman, "University course timetabling using a hybrid harmony search metaheuristic algorithm," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 42, pp. 664–681, 2012.

[7] M. A. Al-Betar and A. T. Khader, "A harmony search algorithm for university course timetabling," *Annals of Operations Research*, vol. 194, no. 1, pp. 3–31, 2012.

[8] M. Awadallah, A. Khader, M. Al-Betar, and A. Bolaji, "Nurse rostering using modied harmony search algorithm," in *Swarm, Evolutionary, and Memetic Computing*, B. Panigrahi, P. Suganthan, S. Das, and S. Satapathy, Eds., vol. 7077 of *Lecture Notes in Computer Science*, pp. 27–37, Springer, Berlin, Germany, 2011.

[9] M. Awadallah, A. Khader, M. Al-Betar, and P. Woon, "Office-space-allocation problem using harmony search algorithm," in *Neural Information Processing*, T. Huang, Z. Zeng, C. Li, and C. Leung, Eds., vol. 7664 of *Lecture Notes in Computer Science*, pp. 365–374, Springer, Berlin, Germany, 2012.

[10] L. Guo, "A novel hybrid bat algorithm with harmony search for global numerical optimization," *Journal of Applied Mathematics*, vol. 2013, Article ID 696491, 21 pages, 2013.

[11] L. Zhang, Y. Xu, and Y. Liu, "An elite decision making harmony search algorithm for optimization problem," *Journal of Applied Mathematics*, vol. 2012, Article ID 860681, 15 pages, 2012.

[12] Z. W. Geem, "Economic dispatch using parameter-setting-free harmony search," *Journal of Applied Mathematics*, vol. 2013, Article ID 427936, 5 pages, 2013.

[13] J. Fourie, R. Green, and Z. W. Geem, "Generalised adaptive harmony search: a comparative analysis of modern harmony search," *Journal of Applied Mathematics*, vol. 2013, Article ID 380985, 13 pages, 2013.

[14] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation*, vol. 198, no. 2, pp. 643–656, 2008.

[15] A. Mukhopadhyay, A. Roy, S. Das, S. Das, and A. Abraham, "Population-variance and explorative power of harmony search: an analysis," in *Proceedings of the 3rd International Conference on Digital Information Management (ICDIM '08)*, pp. 775–781, November 2008.

[16] X.-S. Yang, "Harmony search as a metaheuristic algorithm," in *Music-Inspired Harmony Search Algorithm*, Z. Geem, Ed., vol. 191 of *Studies in Computational Intelligence*, pp. 1–14, Springer, Berlin, Germany, 2009.

[17] M. A. Al-Betar, A. T. Khader, Z. W. Geem, I. A. Doush, and M. A. Awadallah, "An analysis of selection methods in memory consideration for harmony search," *Applied Mathematics and Computation*, vol. 219, no. 22, pp. 10753–10767, 2013.

[18] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 126–142, 2005.

[19] D. Whitley and T. Starkweather, "Genitor ii: a distributed genetic algorithm," *Journal of Experimental & Theoretical Articial Intelligence*, vol. 2, pp. 189–214, 1990.

[20] Y. Shi, H. Liu, L. Gao, and G. Zhang, "Cellular particle swarm optimization," *Information Sciences*, vol. 181, no. 20, pp. 4460–4493, 2011.

[21] T. Y. Lim, "Structured population genetic algorithms: a literature survey," *Artificial Intelligence Review*, 2012.

[22] E. Alba and B. Dorronsoro, *Cellular Genetic Algorithms*, vol. 42, Springer, Berlin, Germany, 2008.

[23] S. Janson, E. Alba, B. Dorronsoro, and M. Middendorf, "Hierarchical cellular genetic algorithm," in *Evolutionary Computation in Combinatorial Optimization*, J. Gottlieb and G. Raidl, Eds., vol. 3906 of *Lecture Notes in Computer Science*, pp. 111–122, Springer, Berlin, Germany, 2006.

[24] J. Błazewicz, M. Drozdowski, F. Guinand, and D. Trystram, "Scheduling a divisible task in a two-dimensional toroidal mesh," *Discrete Applied Mathematics*, vol. 94, no. 1–3, pp. 35–50, 1999.

[25] E. Cantu-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, vol. 1, Springer, London, UK, 2000.

[26] H. Mühlenbein, M. Schomisch, and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel Computing*, vol. 17, no. 6-7, pp. 619–632, 1991.

[27] C. Fernandes and A. Rosa, "A study on non-random mating and varying population size in genetic algorithms using a royal road function," in *Proceedings of the Congress on Evolutionary Computation*, pp. 60–66, May 2001.

[28] J. V. Neumann, *Theory of Self-Reproducing Automata*, University of Illinois Press, Urbana, Ill, USA, 1966.

[29] B. Schönfisch and A. de Roos, "Synchronous and asynchronous updating in cellular automata," *BioSystems*, vol. 51, no. 3, pp. 123–143, 1999.

[30] P. Suganthan, N. Hansen, J. Liang et al., "Problem denitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Tech. Rep., Nanyang Technological University, 2005.

[31] C. García-Martínez and M. Lozano, "Hybrid real-coded genetic algorithms with female and male differentiation," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 896–903, September 2005.

[32] D. Molina, F. Herrera, and M. Lozano, "Adaptive local search parameters for real-coded memetic algorithms," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 888–895, September 2005.

[33] P. Posik, "Real-parameter optimization using the mutation step coevolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 872–879, 2005.

[34] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 506–513, September 2005.

[35] 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.

[36] B. Yuan and M. Gallagher, "Experimental results for the special session on real-parameter optimization at CEC 2005: a simple, continuous EDA," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 1792–1799, September 2005.

[37] A. Sinha, S. Tiwari, and K. Deb, "A population-based, steady-state procedure for real-parameter optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 514–521, September 2005.

[38] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 1777–1784, September 2005.

[39] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 1769–1776, September 2005.

[40] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 1785–1791, September 2005.

[41] P. J. Ballester, J. Stephenson, J. N. Carter, and K. Gallagher, "Real-parameter optimization performance study on the CEC-2005 benchmark with SPC-PNX," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 498–505, September 2005.

[42] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias, "Evaluating evolutionary algorithms," *Artificial Intelligence*, vol. 85, no. 1-2, pp. 245–276, 1996.

[43] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[44] Y. M. Cheng, L. Li, T. Lansivaara, S. C. Chi, and Y. J. Sun, "An improved harmony search minimization algorithm using different slip surface generation methods for slope stability analysis," *Engineering Optimization*, vol. 40, no. 2, pp. 95–115, 2008.

[45] Q.-K. Pan, P. N. Suganthan, J. J. Liang, and M. F. Tasgetiren, "A local-best harmony search algorithm with dynamic subpopulations," *Engineering Optimization*, vol. 42, no. 2, pp. 101–117, 2010.

[46] M. A. Al-Betar, A. T. Khader, and F. Nadi, "Selection mechanisms in memory consideration for examination timetabling with harmony search," in *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference (GECCO '10)*, pp. 1203–1210, ACM, Portland, Ore, USA, July 2010.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 575414, 24 pages
http://dx.doi.org/10.1155/2013/575414

Research Article

Multilevel Thresholding Segmentation Based on Harmony Search Optimization

Diego Oliva,[1] Erik Cuevas,[2] Gonzalo Pajares,[1] Daniel Zaldivar,[2] and Marco Perez-Cisneros[2]

[1] *Departamento de Ingeniería del Software e Inteligencia Artificial, Facultad Informática, Universidad Complutense, 28040 Madrid, Spain*
[2] *Departamento de Ciencias Computacionales, Universidad de Guadalajara, CUCEI, CU-TONALA, Avenida Revolución 1500, C.P 44430, Guadalajara, JAL, Mexico*

Correspondence should be addressed to Erik Cuevas; erik.cuevas@cucei.udg.mx

Received 14 June 2013; Revised 17 August 2013; Accepted 20 August 2013

Academic Editor: Zong Woo Geem

In this paper, a multilevel thresholding (MT) algorithm based on the harmony search algorithm (HSA) is introduced. HSA is an evolutionary method which is inspired in musicians improvising new harmonies while playing. Different to other evolutionary algorithms, HSA exhibits interesting search capabilities still keeping a low computational overhead. The proposed algorithm encodes random samples from a feasible search space inside the image histogram as candidate solutions, whereas their quality is evaluated considering the objective functions that are employed by the Otsu's or Kapur's methods. Guided by these objective values, the set of candidate solutions are evolved through the HSA operators until an optimal solution is found. Experimental results demonstrate the high performance of the proposed method for the segmentation of digital images.

1. Introduction

Segmentation is one of the most important tasks in image processing that endeavors to identify whether a pixel intensity corresponds to a predefined class. Thresholding is the easiest method for segmentation as it works taking a threshold (th) value so that pixels whose intensity value is higher than th are labeled as first class while the rest correspond to a second class label. When the image is segmented into two classes, the task is called bilevel thresholding (BT) and requires only one th value. On the other hand, when pixels are separated into more than two classes, the task is named as MT and demands more than one th value [1].

In recent years image processing has been applied to different areas as engineering, medicine, agriculture, and so forth. Since most of such implementations use a TH methodology, several techniques had been studied. Generally, TH methods are divided into parametric and nonparametric [2–5]. Parametric approaches need to estimate values of a probability density function to model each class. The estimation process is time consuming and computationally expensive. On the other hand, the TH nonparametric employs several criteria such as the between-class variance, the entropy, and the error rate [6–8] in order to verify the quality of a th value. These metrics could also be used as optimization functions since they result as an attractive option due their robustness and accuracy.

There exist two classical thresholding methods. The first, proposed by Otsu in [6] that maximizes the variance between classes while the second method, submitted by Kapur et al. in [7], uses the maximization of the entropy to measure the homogeneity among classes. Their efficiency and accuracy have been already proved for a bilevel segmentation [4]. Although they can be expanded for MT, their computational complexity increases exponentially when a new threshold is incorporated [5].

As an alternative to classical methods, the MT problem has also been handled through evolutionary optimization methods. In general, they have demonstrated to deliver better results than those based on classical techniques in terms

of accuracy, speed, and robustness. Numerous evolutionary approaches have been reported in the literature.

Hammouche et al. in [9] provides an interesting survey of how different evolutionary algorithms are used to solve the Kaptur's and Otsu's problems. The study uses four classical evolutionary algorithms to test their efficiency in MT. Such methods include differential evolution (DE) [10], simulated annealing (SA) [11], and tabu search (TS) [12].

Genetic algorithms (GA) [13], inspired on the biological evolution, have been also used for solving segmentation tasks. One interesting example is presented in [14], where a GA-based algorithm is combined with Gaussian models for multilevel thresholding. Other similar works, such as that of Yin [15] proposes an improved GA for optimal thresholding. In the approach, it is used as a learning strategy to increase the speed of convergence.

Evolutionary approaches inspired on swarm behavior, such as particle swarm optimization (PSO) [16] and artificial bee colony (ABC) [17], have been employed to face the segmentation problem. In [18], both methods are used to find the optimal multilevel threshold points by using the Kapur's entropy as fitness function.

Finally, in [19], the optimal segmentation threshold values are determined by using the bacterial foraging algorithm (BFA). Such method aims to maximize the Kapur's and Otsu's objective functions by considering a set of operators that are based on the social foraging behavior of the bacteria Eschericha Colli.

On the other hand, the harmony search algorithm (HSA) introduced by Geem et al. [20] is an evolutionary optimization algorithm which is based on the metaphor of the improvisation process that occurs when a musician searches for a better state of harmony. The HSA generates a new candidate solution from all existing solutions. The solution vector is analogous to the harmony in music while the local and global search schemes are analogous to musician's improvisations. In comparison to other metaheuristics methods in the literature, HSA imposes fewer mathematical requirements as it can be easily adapted for solving several sorts of engineering optimization challenges [21]. Furthermore, numerical comparisons have demonstrated that the convergence for the HSA is faster than GA [22] which attracts further attention. It has been successfully applied to solve a wide range of practical optimization problems such as discrete and continuous structural optimization [23], parameter estimation of the nonlinear Muskingum model [24], design optimization of water distribution networks [25], vehicle routing [26], combined heat and power economic dispatch [27], design of steel frames [28], and image processing [29]. Although the standard HSA presents good search characteristics, several modifications to the original HSA have been proposed in the literature in order to enhance its own features [30].

In this paper, a novel multithresholding segmentation algorithm is introduced. The proposed approach, called the harmony search multithresholding algorithm (HSMA), combines the original harmony search algorithm (HSA) and the Otsu's and Kapur's methodologies. The proposed algorithm takes random samples from a feasible search space inside the image histogram. Such samples build each harmony (candidate solution) in the HSA context, whereas its quality is evaluated considering the objective function that is employed by the Otsu's or the Kapur's method. Guided by these objective values, the set of candidate solutions are evolved using the HSA operators until the optimal solution is found. The approach generates a multilevel segmentation algorithm which can effectively identify the threshold values of a digital image within a reduced number of iterations. Experimental results over several complex images have validated the efficiency of the proposed technique regarding accuracy, speed, and robustness.

The paper is organized as follows. In Section 2, the HSA is introduced. Section 3 gives a brief description of the Otsu's and Kapur's methods. Section 4 explains the implementation of the proposed algorithm. Section 5 discusses experimental results after testing the proposed method over a set benchmark images. Finally, some conclusions are discussed in Section 6.

2. Harmony Search Algorithm

2.1. The Harmony Search Algorithm. In the basic HSA, each solution is called a "harmony" and is represented by an n-dimension real vector. An initial population of harmony vectors are randomly generated and stored within a harmony memory (HM). A new candidate harmony is thus generated from the elements in the HM by using a memory consideration operation either by a random reinitialization or a pitch adjustment operation. Finally, the HM is updated by comparing the new candidate harmony and the worst harmony vector in the HM. The worst harmony vector is replaced by the new candidate vector when the latter delivers a better solution in the HM. The above process is repeated until a certain termination criterion is met. The basic HS algorithm consists of three main phases: HM initialization, improvisation of new harmony vectors, and updating the HM. The following discussion addresses details about each stage.

2.1.1. Initializing the Problem and the Algorithm Parameters. In general, the global optimization problem can be summarized as follows:

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}), \quad \mathbf{x} = (x(1), x(2), \ldots, x(n)) \in \mathbf{R}^n, \\ &\text{subject to: } x(j) \in [l(j), u(j)] \quad j = 1, 2, \ldots, n, \end{aligned} \tag{1}$$

where $f(\mathbf{x})$ is the objective function, $\mathbf{x} = (x(1), x(2), \ldots, x(n))$ is the set of design variables, n is the number of design variables, and $l(j)$ and $u(j)$ are the lower and upper bounds for the design variable $x(j)$, respectively. The parameters for HSA are the harmony memory size, that is, the number of solution vectors lying on the harmony memory (HM), the harmony-memory consideration rate (HMCR), the pitch adjusting rate (PAR), the distance bandwidth (BW), and the number of improvisations (NI) which represents the total number of iterations. The performance of HSA is strongly influenced by values assigned to such parameters, which in turn, depend on the application domain [31].

2.1.2. Harmony Memory Initialization. In this stage, initial vector components at HM, that is, HMS vectors, are configured. Let $\mathbf{x}_i = \{x_i(1), x_i(2), \ldots, x_i(n)\}$ represent the ith randomly-generated harmony vector: $x_i(j) = l(j) + (u(j) - l(j)) \cdot \text{rand}(0,1)$ for $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots,$ HMS, where rand(0,1) is a uniform random number between 0 and 1, the upper and lower bounds of the search space are defined by $l(j)$ and $u(j)$, respectively. Then, the HM matrix is filled with the HMS harmony vectors as follows:

$$\text{HM} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{\text{HMS}} \end{bmatrix}. \tag{2}$$

2.1.3. Improvisation of New Harmony Vectors. In this phase, a new harmony vector $\mathbf{x}_{\text{new}}$ is built by applying the following three operators: memory consideration, random reinitialization, and pitch adjustment. Generating a new harmony is known as "improvisation." In the memory consideration step, the value of the first decision variable $x_{\text{new}}(1)$ for the new vector is chosen randomly from any of the values already existing in the current HM, that is, from the set $\{x_1(1), x_2(1), \ldots, x_{\text{HMS}}(1)\}$. For this operation, a uniform random number r_1 is generated within the range $[0, 1]$. If r_1 is less than HMCR, the decision variable $x_{\text{new}}(1)$ is generated through memory considerations; otherwise, $x_{\text{new}}(1)$ is obtained from a random reinitialization between the search bounds $[l(1), u(1)]$. Values of the other decision variables $x_{\text{new}}(2), x_{\text{new}}(3), \ldots, x_{\text{new}}(n)$ are also chosen accordingly. Therefore, both operations, memory consideration and random reinitialization, can be modeled as follows:

$$x_{\text{new}}(j) = \begin{cases} x_i(j) \in \{x_1(j), x_2(j), \ldots, x_{\text{HMS}}(j)\}, \\ \qquad \text{with probability HMCR}, \\ l(j) + (u(j) - l(j)) \cdot \text{rand}(0,1), \\ \qquad \text{with probability } 1 - \text{HMCR}. \end{cases} \tag{3}$$

Every component obtained by memory consideration is further examined to determine whether it should be pitch-adjusted. For this operation, the pitch-adjusting rate (PAR) is defined as to assign the frequency of the adjustment and the bandwidth factor (BW) to control the local search around the selected elements of the HM. Hence, the pitch-adjusting decision is calculated as follows:

$$x_{\text{new}}(j) = \begin{cases} x_{\text{new}}(j) = x_{\text{new}}(j) \pm \text{rand}(0,1) \cdot \text{BW}, \\ \qquad \text{with probability PAR}, \\ x_{\text{new}}(j), \quad \text{with probability } (1 - \text{PAR}). \end{cases} \tag{4}$$

Pitch adjusting is responsible for generating new potential harmonies by slightly modifying original variable positions. Such operation can be considered similar to the mutation process in evolutionary algorithms. Therefore, the decision variable is either perturbed by a random number between 0 and BW or left unaltered. In order to protect the pitch adjusting operation, it is important to assure that points lying outside the feasible range $[l, u]$ must be reassigned, that is, truncated to the maximum or minimum values of the interval.

2.1.4. Updating the Harmony Memory. After a new harmony vector x_{new} is generated, the harmony memory is updated by the survival of the fit competition between x_{new} and the worst harmony vector x_w in the HM. Therefore x_{new} will replace x_w and become a new member of the HM in case the fitness value of x_{new} is better than the fitness value of x_w.

2.1.5. Computational Procedure. The computational procedure of the basic HSA can be summarized as in Algorithm 1 [20].

This procedure is implemented for minimization. If the intention is to maximize the objective function, a sign modification of step 4 ($\mathbf{x}_w = \mathbf{x}_{\text{new}}$ if $f(\mathbf{x}_{\text{new}}) > f(\mathbf{x}_w)$) is required. In this paper the HSA is used for maximization proposes.

3. Image Multilevel Thresholding (MT)

Thresholding is a process in which the pixels of a gray scale image are divided in sets or classes depending on their intensity level (L). For this classification it is necessary to select a threshold value (th) and follow the simple rule of

$$\begin{aligned} C_1 &\longleftarrow p \quad \text{if } 0 \le p < \text{th}, \\ C_2 &\longleftarrow p \quad \text{if th} \le p < L - 1, \end{aligned} \tag{5}$$

where p is one of the $m \times n$ pixels of the gray scale image I_g that can be represented in L gray scale levels $L = \{0, 1, 2, \ldots, L-1\}$. C_1 and C_2 are the classes in which the pixel p can be located, while th is the threshold. The rule in (5) corresponds to a bilevel thresholding and can be easily extended for multiple sets:

$$\begin{aligned} C_1 &\longleftarrow p \quad \text{if } 0 \le p < \text{th}_1, \\ C_2 &\longleftarrow p \quad \text{if th}_1 \le p < \text{th}_2, \\ C_i &\longleftarrow p \quad \text{if th}_i \le p < \text{th}_{i+1}, \\ C_n &\longleftarrow p \quad \text{if th}_n \le p < L - 1, \end{aligned} \tag{6}$$

where $\{\text{th}_1, \text{th}_2, \ldots, \text{th}_i, \text{th}_{i+1}, \text{th}_k\}$ represent different thresholds. The problem for both bilevel and MT is to select the th values that correctly identify the classes. Although, Otsu's and Kapur's methods are well-known approaches for determining such values, both propose a different objective function which must be maximized in order to find optimal threshold values, just as it is discussed below.

3.1. Between-Class Variance (Otsu's Method). This is a nonparametric technique for thresholding proposed by Otsu [6] that employs the maximum variance value of the different

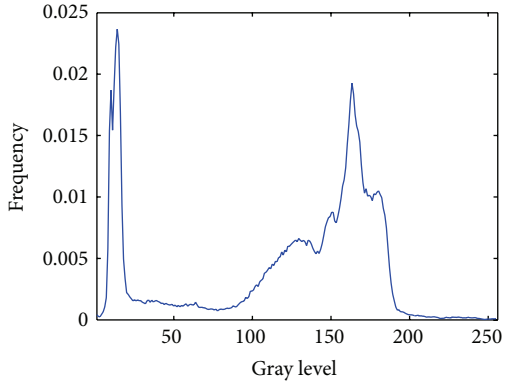
Step 1. Set the parameters HMS, HMCR, PAR, BW and NI.
Step 2. Initialize the HM and calculate the objective function value of each harmony vector.
Step 3. Improvise a new harmony $\mathbf{x}_{\text{new}}$ as follows:
for ($j = 1$ to n) do
if ($r_1 <$ HMCR) then
$x_{\text{new}}(j) = x_a(j)$ where $a \in (1, 2, \ldots, \text{HMS})$
if ($r_2 <$ PAR) then
$x_{\text{new}}(j) = x_{\text{new}}(j) \pm r_3 \cdot \text{BW}$ where $r_1, r_2, r_3 \in \text{rand}\ (0, 1)$
end if
if $x_{\text{new}}(j) < l(j)$
$x_{\text{new}}(j) = l(j)$
end if
if $x_{\text{new}}(j) > u(j)$
$x_{\text{new}}(j) = u(j)$
end if
else
$x_{\text{new}}(j) = l(j) + r \cdot (u(j) - l(j))$, where $r \in \text{rand}\ (0, 1)$
end if
end for
Step 4. Update the HM as $\mathbf{x}_w = \mathbf{x}_{\text{new}}$ if $f(\mathbf{x}_{\text{new}}) < f(\mathbf{x}_w)$
Step 5. If NI is completed, the best harmony vector $\mathbf{x}_b$ in the HM is returned; otherwise go back to Step3.

Algorithm 1

Step 1. Read the image I and if it is RGB separate it into I_R, I_G and I_B. If I is gray scale store it into I_{Gr}. $c = 1, 2, 3$ for RGB images or $c = 1$ for gray scale images.
Step 2. Obtain histograms: for RGB images h^R, h^G, h^B and for gray scale images h^{Gr}.
Step 3. Calculate the probability distribution using (7) and obtain the histograms.
Step 4. Initialize the HSA parameters: HMS, k, HMCR, PAR, BW, NI, and the limits l and u.
Step 5. Initialize a HM $\mathbf{x}_i^c$ of HMS random particles with k dimensions.
Step 6. Compute the values ω_i^c and μ_i^c. Evaluate each element of **HM** in the objective function $J(\mathbf{HM})$ (14) or (20) depending on the thresholding method (Otsu or Kapur respectively).
Step 7. Improvise a new harmony $\mathbf{x}_{\text{new}}^c$ as follows:
for $(j = 1$ to $n)$ do
if $(r_1 <$ HCMR$)$ then
$x_{\text{new}}^c(j) = x_a^c(j)$ where $a \in (1, 2, \ldots, \text{HMS})$
if $(r_2 <$ PAR$)$ then
$x_{\text{new}}^c(j) = x_a^c(j) \pm r_3 \cdot \text{BW}$ where $r_1, r_2, r_3 \in \text{rand}\,(0, 1)$
end if
if $x_{\text{new}}^c(j) < l(j)$
$x_{\text{new}}^c(j) = l(j)$
end if
if $x_{\text{new}}^c(j) > u(j)$
$x_{\text{new}}^c(j) = u(j)$
end if
else
$x_{\text{new}}^c(j) = l(j) + r \cdot (u(j) - l(j))$ whew $r \in \text{rand}\ (0, 1)$
end if
end for
Step 8. Update the HM as $\mathbf{x}_{\text{worst}}^c = \mathbf{x}_{\text{new}}^c$ if $f(\mathbf{x}_{\text{new}}^c) > f(\mathbf{x}_{\text{worst}}^c)$
Step 9. If NI is completed or the stop criteria is satisfied, then jump to Step 10; otherwise go back to Step 6.
Step 10. Select the harmony that has the best x_{best}^cobjective function value.
Step 11. Apply the thresholds values contained in x_{best}^c to the image I (6).

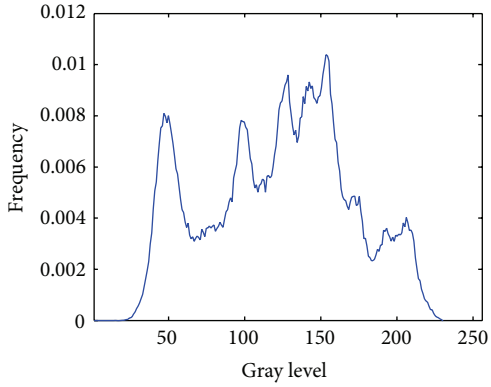
Algorithm 2

(a)

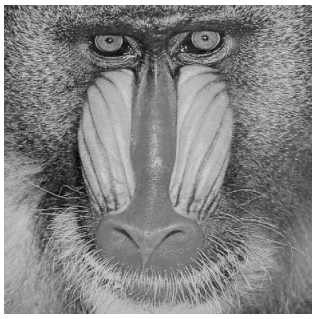


(b)

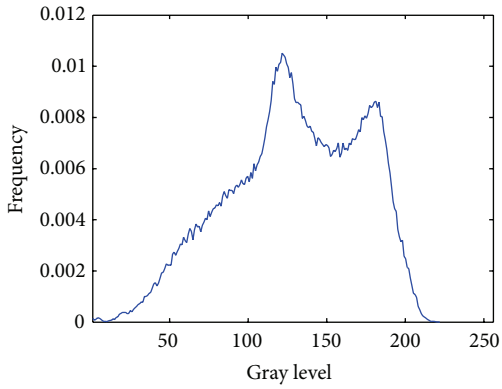
(c)



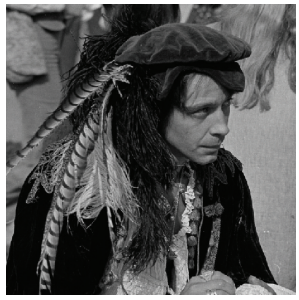
(d)



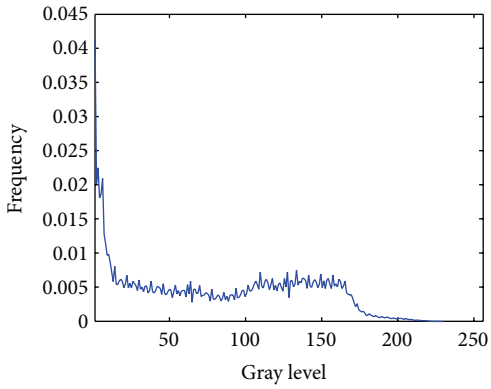
(e)



(f)

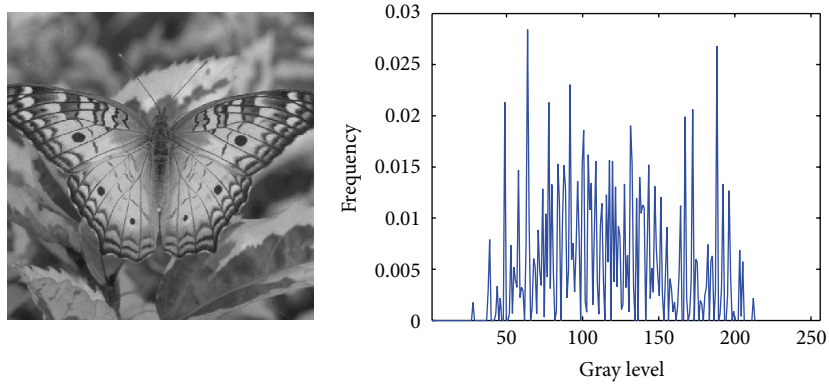


(g)



(h)

Figure 1: Continued.



(i) (j)

Figure 1: (a) Camera man, (c) Lena, (e) Baboon, (g) Hunter, and (i) Butterfly, the selected benchmark images. (b), (d), (f), (h), and (j) histograms of the images.

classes as a criterion to segment the image. Taking the L intensity levels from a gray scale image or from each component of a RGB (red, green, and blue) image, the probability distribution of the intensity values is computed as follows:

$$\mathrm{Ph}_i^c = \frac{h_i^c}{\mathrm{NP}}, \qquad \sum_{i=1}^{\mathrm{NP}} \mathrm{Ph}_i^c = 1,$$
$$c = \begin{cases} 1, 2, 3, & \text{if RGB Image,} \\ 1, & \text{if Gray scale Image,} \end{cases} \tag{7}$$

where i is a specific intensity level ($0 \le i \le L-1$), c is the component of the image which depends if the image is gray scale or RGB, whereas NP is the total number of pixels in the image. h_i^c (histogram) is the number of pixels that corresponds to the i intensity level in c. The histogram is normalized within a probability distribution Ph_i^c. For the simplest segmentation (bilevel) two classes are defined as

$$C_1 = \frac{\mathrm{Ph}_1^c}{\omega_0^c(\mathrm{th})}, \ldots, \frac{\mathrm{Ph}_{\mathrm{th}}^c}{\omega_0^c(\mathrm{th})}, \qquad C_2 = \frac{\mathrm{Ph}_{\mathrm{th}+1}^c}{\omega_1^c(\mathrm{th})}, \ldots, \frac{\mathrm{Ph}_L^c}{\omega_1^c(\mathrm{th})}, \tag{8}$$

where $\omega_0(\mathrm{th})$ and $\omega_1(\mathrm{th})$ are probabilities distributions for C_1 and C_2, as it is shown by

$$\omega_0^c(\mathrm{th}) = \sum_{i=1}^{\mathrm{th}} \mathrm{Ph}_i^c, \qquad \omega_1^c(\mathrm{th}) = \sum_{i=\mathrm{th}+1}^{L} \mathrm{Ph}_i^c. \tag{9}$$

It is necessary to compute mean levels μ_0^c and μ_1^c that define the classes using (10). Once those values are calculated, the Otsu variance between classes σ^{2^c} is calculated using (11) as follows:

$$\mu_0^c = \sum_{i=1}^{\mathrm{th}} \frac{i\mathrm{Ph}_i^c}{\omega_0^c(\mathrm{th})}, \qquad \mu_1^c = \sum_{i=\mathrm{th}+1}^{L} \frac{i\mathrm{Ph}_i^c}{\omega_1^c(\mathrm{th})}, \tag{10}$$

$$\sigma^{2^c} = \sigma_1^c + \sigma_2^c. \tag{11}$$

Notice that for both equations, (10) and (11), c depends on the type of image. In (11) the number two is part of the Otsu's variance operator and does not represent an exponent in the mathematical sense. Moreover σ_1^c and σ_2^c in (11) are the variances of C_1 and C_2 which are defined as

$$\sigma_1^c = \omega_0^c(\mu_0^c + \mu_T^c)^2, \qquad \sigma_2^c = \omega_1^c(\mu_1^c + \mu_T^c)^2, \tag{12}$$

where $\mu_T^c = \omega_0^c\mu_0^c + \omega_1^c\mu_1^c$ and $\omega_0^c + \omega_1^c = 1$. Based on the values σ_1^c and σ_2^c, (13) presents the objective function:

$$J(\mathrm{th}) = \max\left(\sigma^{2^c}(\mathrm{th})\right), \quad 0 \le \mathrm{th} \le L-1, \tag{13}$$

where $\sigma^{2^c}(\mathrm{th})$ is the Otsu's variance for a given th value. Therefore, the optimization problem is reduced to find the intensity level (th) that maximizes (13).

Otsu's method is applied for a single component of an image. In case of RGB images, it is necessary to apply separation into single component images. The previous description of such bilevel method can be extended for the identification of multiple thresholds. Considering k thresholds, it is possible separate the original image into k classes using (6); then it is necessary to compute the k variances and their respective elements. The objective function $J(\mathrm{th})$ in (13) can thus be rewritten for multiple thresholds as follows:

$$J(\mathbf{TH}) = \max\left(\sigma^{2^c}(\mathbf{TH})\right), \quad 0 \le \mathrm{th}_i \le L-1, \ i = 1, 2, \ldots, k, \tag{14}$$

where $\mathbf{TH} = [\mathrm{th}_1, \mathrm{th}_2, \ldots, \mathrm{th}_{k-1}]$, is a vector containing multiple thresholds and the variances are computed through

$$\sigma^{2^c} = \sum_{i=1}^{k} \sigma_i^c = \sum_{i=1}^{k} \omega_i^c(\mu_i^c - \mu_T^c)^2. \tag{15}$$

Here, i represents the i class, ω_i^c and μ_j^c are, respectively, the probability of occurrence and the mean of a class. In MT, such values are obtained as

$$\begin{aligned}\omega_0^c(\text{th}) &= \sum_{i=1}^{\text{th}_1} \text{Ph}_i^c,\\ \omega_1^c(\text{th}) &= \sum_{i=\text{th}_1+1}^{\text{th}_2} \text{Ph}_i^c,\\ &\vdots\\ \omega_{k-1}^c(\text{th}) &= \sum_{i=\text{th}_k+1}^{L} \text{Ph}_i^c.\end{aligned} \tag{16}$$

And, for the mean values

$$\begin{aligned}\mu_0^c &= \sum_{i=1}^{\text{th}_1} \frac{i\text{Ph}_i^c}{\omega_0^c(\text{th}_1)},\\ \mu_1^c &= \sum_{i=\text{th}_1+1}^{\text{th}_2} \frac{i\text{Ph}_i^c}{\omega_0^c(\text{th}_2)},\\ &\vdots\\ \mu_{k-1}^c &= \sum_{i=\text{th}_k+1}^{L} \frac{i\text{Ph}_i^c}{\omega_1^c(\text{th}_k)}.\end{aligned} \tag{17}$$

Similar to the bilevel case, for the MT using the Otsu's method, c corresponds to the image components, RGB $c = 1, 2, 3$, and gray scale $c = 1$.

3.2. Entropy Criterion Method (Kapur's Method). Another nonparametric method that is used to determine the optimal threshold values has been proposed by Kapur et al. [7]. It is based on the entropy and the probability distribution of the image histogram. The method aims to find the optimal th that maximizes the overall entropy. The entropy of an image measures the compactness and separability among classes. In this sense, when the optimal th value appropriately separates the classes, the entropy has the maximum value. For the bilevel example, the objective function of the Kapur's problem can be defined as

$$\begin{aligned}J(\text{th}) &= H_1^c + H_2^c,\\ c &= \begin{cases}1, 2, 3, & \text{if RGB Image,}\\ 1, & \text{if Gray scale Image,}\end{cases}\end{aligned} \tag{18}$$

where the entropies H_1 and H_2 are computed by the following model:

$$\begin{aligned}H_1^c &= \sum_{i=1}^{\text{th}} \frac{\text{Ph}_i^c}{\omega_0^c} \ln\left(\frac{\text{Ph}_i^c}{\omega_0^c}\right),\\ H_2^c &= \sum_{i=\text{th}+1}^{L} \frac{\text{Ph}_i^c}{\omega_1^c} \ln\left(\frac{\text{Ph}_i^c}{\omega_1^c}\right).\end{aligned} \tag{19}$$

Ph_i^c is the probability distribution of the intensity levels which is obtained using (7). $\omega_0(\text{th})$ and $\omega_1(\text{th})$ are probabilities distributions for C_1 and C_2. $\ln(\cdot)$ stands for the natural logarithm. Similar to the Otsu's method, the entropy-based approach can be extended for multiple threshold values; for such a case, it is necessary to divide the image into k classes using the similar number of thresholds. Under such conditions, the new objective function is defined as:

$$\begin{aligned}J(\mathbf{TH}) &= \max\left(\sum_{i=1}^{k} H_i^c\right),\\ c &= \begin{cases}1, 2, 3, & \text{if RGB Image,}\\ 1, & \text{if Gray scale Image,}\end{cases}\end{aligned} \tag{20}$$

where $\mathbf{TH} = [\text{th}_1, \text{th}_2, \ldots, \text{th}_{k-1}]$ is a vector that contains the multiple thresholds. Each entropy is computed separately with its respective th value, so (21) is expanded for k entropies:

$$\begin{aligned}H_1^c &= \sum_{i=1}^{\text{th}_1} \frac{\text{Ph}_i^c}{\omega_0^c} \ln\left(\frac{\text{Ph}_i^c}{\omega_0^c}\right),\\ H_2^c &= \sum_{i=\text{th}_1+1}^{\text{th}_2} \frac{\text{Ph}_i^c}{\omega_1^c} \ln\left(\frac{\text{Ph}_i^c}{\omega_1^c}\right),\\ &\vdots\\ H_k^c &= \sum_{i=\text{th}_k+1}^{L} \frac{\text{Ph}_i^c}{\omega_{k-1}^c} \ln\left(\frac{\text{Ph}_i^c}{\omega_{k-1}^c}\right).\end{aligned} \tag{21}$$

The values of the probability occurrence $(\omega_0^c, \omega_1^c, \ldots, \omega_{k-1}^c)$ of the k classes are obtained using (16) and the probability distribution Ph_i^c with (10). Finally, it is necessary to use (6) to separate the pixels into the corresponding classes.

4. Multilevel Thresholding Using Harmony Search Algorithm (HSMA)

4.1. Harmony Representation. Each harmony (candidate solution) uses k different elements as decision variables within the optimization algorithm. Such decision variables represent a different threshold point th that is used for the segmentation. Therefore, the complete population is represented as

$$\begin{aligned}\mathbf{HM} &= \left[\mathbf{x}_1^c, \mathbf{x}_2^c, \ldots, \mathbf{x}_{\text{HMS}}^c\right]^T,\\ \mathbf{x}_i^c &= \left[\text{th}_1^c, \text{th}_2^c, \ldots, \text{th}_k^c\right],\end{aligned} \tag{22}$$

where T refers to the transpose operator, HMS is the size of the harmony memory, $\mathbf{x}_i$ is the ith element of HM, and $c = 1, 2, 3$ is set for RGB images while $c = 1$ is chosen for gray scale images. For this problem, the boundaries of the search space are set to $l = 0$ and $u = 255$, which correspond to image intensity levels.

4.2. HMA Implementation. The proposed segmentation algorithm has been implemented considering two different objective functions: Otsu and Kapur. Therefore, the HSA has been

coupled with the Otsu and Kapur functions, producing two different segmentation algorithms. The implementation of both algorithms can be summarized as in Algorithm 2.

4.3. Parameter Setting. The performance of HSA is strongly influenced by values assigned to parameters HM, HMCR, PAR, BW, and NI. Determining the most appropriate parameter values for an arbitrary problem is a complex issue, since such parameters interact to each other in a highly nonlinear manner, and no mathematical models of such interaction currently exist. The common method to find the best set of parameter values is to fix each parameter value to a random number within the parameter limits and then HSA is executed. If the final result is not satisfactory; then a new set of parameter values is defined, and the evolutionary algorithm is executed again. Evidently, this process can be very expensive (in terms of computational time), since many different trials may be required before reaching a set of satisfactory parameter values. Additionally, the set of values chosen by the user are not necessarily the best possible, but only the best from the arbitrary number of trials performed by the user. In order to reduce the number of experiments in this paper, it has used the factorial design method proposed in [32, 33] to systematically identify the best parameters of HSA.

The factorial design method [34] is a statistical technique that evaluates at the same time all process variables in order to determine which ones really exert significant influence on the final response. All variables are called factors and the different values chosen to study the factors are called levels. The factors to be considered in the factorial design are the HSA parameters, the harmony memory (HM), the harmony-memory consideration rate (HMCR), the pitch adjusting rate (PAR), the distance bandwidth (BW), and the number of improvisations (NI), whereas the response is the best fitness value obtained as a consequence of the HSA execution. Table 1 show the levels of the quantitative factors used in the factorial design. The values of zero level (central point) are based on the suggestions of the literature [33].

Each experiment is conducted combining the two different levels that define each parameter considering as a problem an image histogram example. Since the factors are five, a 2^{5-1} fractional factorial design is chosen, requiring sixteen optimization experiments plus one optimization trial for the central point. The results obtained from seventeen runs were analyzed according to [32, 33] using a general linear form of analysis of variance (ANOVA) [34] considering a 95% of confidence. After such analysis, it was found that the best possible configuration of HSA is shown in Table 2. These results were consistent considering six replications using different image histograms and the Otsu (14) or Kapur (20) functions, indistinctly. For more information on how to build fractional factorial designs, the reader is referred to [32, 33].

Table 1: Levels of the factors used for the factorial design method.

HSA parameters (Factors)	(−) Level	Central point	(+) Level
HM	50	100	200
HMCR	0.5	0.75	0.9
PAR	0.1	0.5	0.9
BW	0.1	0.3	0.5
NI	200	250	300

Table 2: HSMA parameter values obtained by the factorial design method.

HM	HMCR	PAR	BW	NI
100	0.75	0.5	0.5	300

5. Experimental Results

The HSMA has been tested under a set of 11 benchmark images. Some of these images are widely used in the image processing literature to test different methods (Lena, Cameraman, Hunter, Baboon, etc.) [3, 9]. All the images have the same size (512 × 512 pixels), and they are in JPGE format. For the sake of representation, only five images which are presented in Figure 1 have been used to show the visual results; however, the numerical outcomes are analyzed considering the complete set.

Since HSMA is stochastic, it is necessary to employ an appropriate statistical metrics to measure its efficiency. Hence, the results have been reported executing the algorithm 35 times for each image. In order to maintain compatibility with similar works reported in the literature [14, 15, 18, 19], the number of thresholds points used in the test are th = 2, 3, 4, 5. In the experiments, the stop criterion is the number of times in which the best fitness values remain with no change. Therefore, if the fitness value for the best harmony remains unspoiled in 10% of the total number of iterations (NI), then the HSA is stopped.

To evaluate the stability and consistency, it has been computed the standard deviation (STD) from the results obtained in the 35 executions. Since the STD represents a measure about how the data are dispersed, the algorithm becomes more instable as the STD value increases [19]. Equation (23) shows the model used to calculate the STD value:

$$\mathrm{STD} = \sqrt{\sum_{i=1}^{\mathrm{NI}} \frac{(\mathrm{bf}_i - \mathrm{av})}{\mathrm{Ru}}}, \tag{23}$$

where bf_i is the best fitness of the ith iteration, av is the average value of bf, and Ru is the number of total executions (Ru = 35).

On the other hand, as an index of quality, the peak-to-signal ratio (PSNR) is used to assess the similarity of the segmented image against a reference image (original image) based on the produced mean square error (MSE) [18, 35]. Both PSNR and MSE are defined as

$$\begin{aligned} \mathrm{PSNR} &= 20\log_{10}\left(\frac{255}{\mathrm{RMSE}}\right), \quad (\mathrm{dB}) \\ \mathrm{RMSE} &= \sqrt{\frac{\sum_{i=1}^{\mathrm{ro}}\sum_{j=1}^{\mathrm{co}}\left(I_o^c(i,j) - I_{\mathrm{th}}^c(i,j)\right)}{\mathrm{ro}\times\mathrm{co}}}, \end{aligned} \tag{24}$$

Table 3: Result after appling the HSMA using Otsu's function to the set of benchmark images.

Image	k	Thresholds x^c_{best}	PSNR	STC
Camera man	2	70, 144	17.247	$2.30E-12$
	3	59, 119, 156	20.211	$1.55E-02$
	4	42, 95, 140, 170	21.533	$2.76E-12$
	5	36, 82, 122, 149, 173	23.282	$5.30E-03$
Lena	2	91, 150	15.401	$9.22E-13$
	3	79, 125, 170	17.427	$2.99E-02$
	4	73, 112, 144, 179	18.763	$2.77E-01$
	5	71, 107, 134, 158, 186	19.443	$3.04E-01$
Baboon	2	97, 149	15.422	$6.92E-13$
	3	85, 125, 161	17.709	$1.92E-02$
	4	71, 105, 136, 167	20.289	$5.82E-02$
	5	66, 97, 123, 147, 173	21.713	$4.40E-01$
Hunter	2	51, 116	17.875	$2.30E-12$
	3	36, 86, 135	20.350	$2.30E-12$
	4	27, 65, 104, 143	22.203	$1.22E-02$
	5	22, 53, 88, 112, 152	23.703	$1.84E-12$
Airplane	2	113, 173	15.029	$9.22E-13$
	3	92, 144, 190	18.854	$4.83E-01$
	4	84, 129, 172, 203	20.735	$7.24E-01$
	5	68, 106, 143, 180, 205	23.126	$8.38E-01$
Peppers	2	72, 138	16.299	$1.38E-12$
	3	65, 122, 169	18.359	$4.61E-13$
	4	50, 88, 128, 171	20.737	$4.61E-13$
	5	48, 85, 118, 150, 179	22.310	$1.84E-012$
Living room	2	87, 145	15.999	$1.15E-12$
	3	76, 123, 163	18.197	$6.92E-12$
	4	56, 97, 132, 168	20.673	$9.22E-12$
	5	49, 88, 120, 146, 178	22.225	$2.86E-02$
Blonde	2	106, 155	14.609	$6.92E-13$
	3	53, 112, 158	19.157	$9.23E-13$
	4	50, 103, 139, 168	20.964	$5.48E-01$
	5	49, 92, 121, 152, 172	22.409	$6.50E-01$
Bridge	2	91, 56	13.943	$4.61E-13$
	3	72, 120, 177	17.019	$7.10E-01$
	4	63, 103, 145, 193	18.872	$2.91E-01$
	5	56, 91, 124, 159, 201	20.539	$3.57E-01$
Butterfly	2	99, 151	13.934	$7.30E-02$
	3	82, 119, 160	16.932	$6.17E-01$
	4	71, 102, 130, 163	19.259	$3.07E+00$
	5	62, 77, 109, 137, 167	21.450	$3.87E+00$
Lake	2	85, 154	14.638	$4.61E-13$
	3	78, 140, 194	15.860	$1.84E-12$
	4	67, 110, 158, 198	17.629	$2.68E-01$
	5	57, 88, 127, 166, 200	19.416	$1.12E-01$

where I^c_o is the original image, I^c_{th} is the segmented image, $c = 1$ for gray scale, and $c = 3$ for RGB images, whereas ro, co are the total number of rows and columns of the image, respectively.

5.1. Otsu's Results. This section analyzes the results of HSMA after considering the variance among classes (14) as the objective function, just as it has been proposed by Otsu [6]. The approach is applied over the complete set of benchmark images, whereas the results are registered in Table 3. Such results present the best threshold values after testing the proposed method with four different threshold points th = 2, 3, 4, 5. The table also features the PSNR and the STD values. It is evident that the PSNR and STD values increase their magnitude as the number of threshold points also increases.

For the sake of representation, it has been selected only five images of the set to show (graphically) the segmentation results. Figure 1 presents the images selected from the benchmark set and their respective histograms which possess irregular distributions (see Figure 1(j) in particular). Under such circumstances, classical methods face great difficulties to find the best threshold values.

The processing results for the selected original images are presented in five tables: Tables 4, 5, 6, 7, and 8. Such results show the segmented images considering four different threshold points, th = 2, 3, 4, 5. The tables also show the evolution of the objective function during one execution.

5.2. Kapur's Results. This section analyzes the performance of HSMA after considering the entropy function (20) as objective function, as it has been proposed by Kapur et al. in [7]. Table 9 presents the experimental results after the application of HSMA over the entire set of benchmark images. The values listed are PSNR, STD, and the best threshold values of the last population (x^B_t). The same test procedure that was previously applied to the Otsu's method (Section 5.1) is used with the Kapur's method, also considering the same stop criterion and a similar HSA parameter configuration.

The results after appling the HSMA to the selected benchmark images are presented in Tables 10, 11, 12, 13, and 14. Four different threshold points have been employed: th = 2, 3, 4, 5. All tables exhibit the segmented image, the approximated histogram, and the evolution of the fitness value during the execution of the HSA method.

From the results of both Otsu's and Kapur's methods, it is possible to appreciate that the HSMA converges (stabilizes) after a determined number of iterations depending on the th value. For experimental purposes HSMA continues running still further, even though the stop criterion is achieved. In this way, the graphics show that convergence is often reached in the first iterations of the optimization process. The segmented images provide evidence that the outcome is better with th = 4 and th = 5; however, if the segmentation task does not require to be extremely accurate then it is possible to select th = 3.

5.3. Comparisons. In order to analyze the results of HSMA, two different comparisons are executed. The first one involves the comparison between the two versions of the proposed approach, one with the Otsu function and the other with

Table 4: Results after appling the HSMA using Otsu's over the Camera man image.

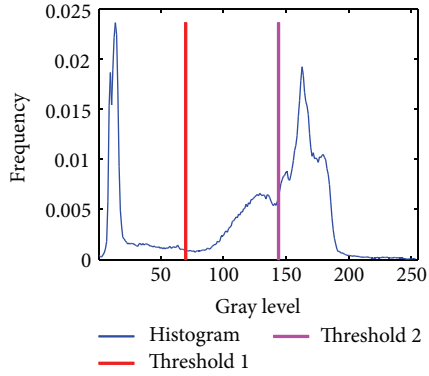
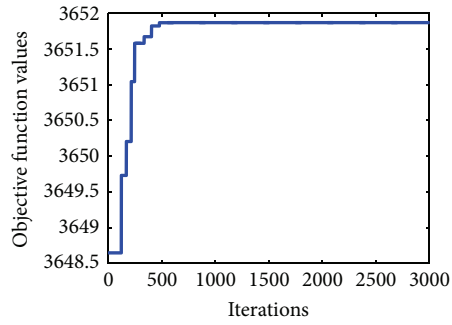
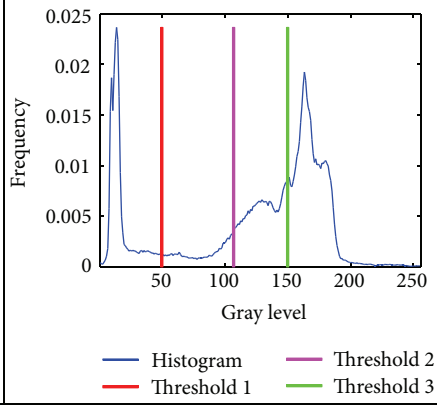
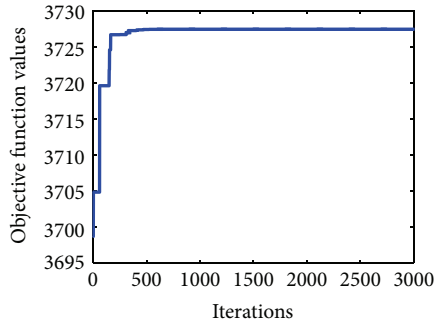
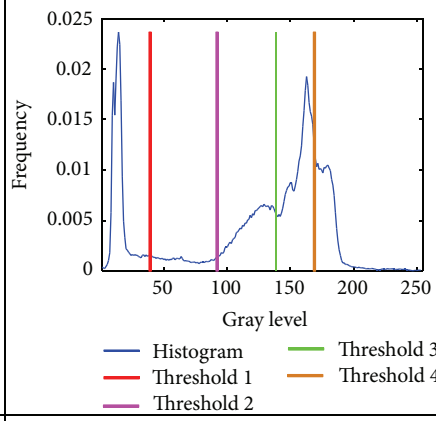
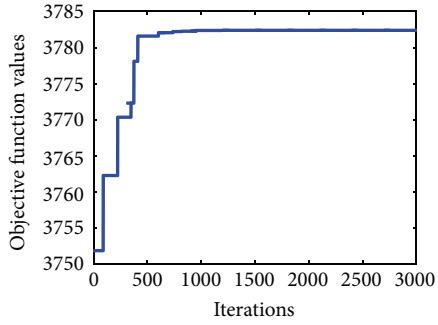
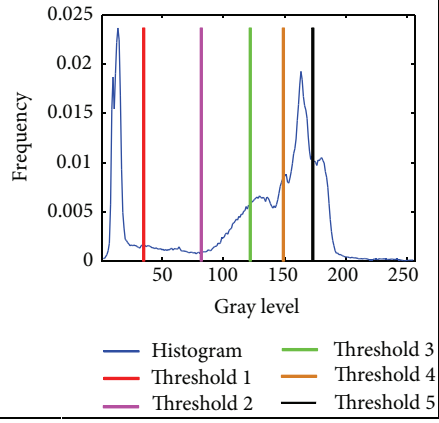
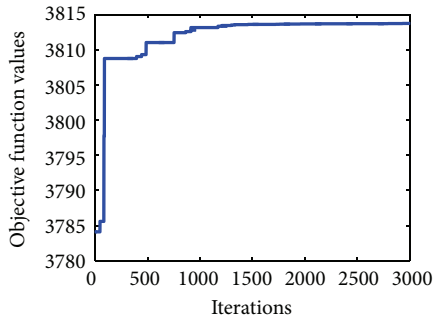
th = 2			
th = 3			
th = 4			
th = 5			

Table 5: Results after appling the HSMA using Otsu's over Lena image.

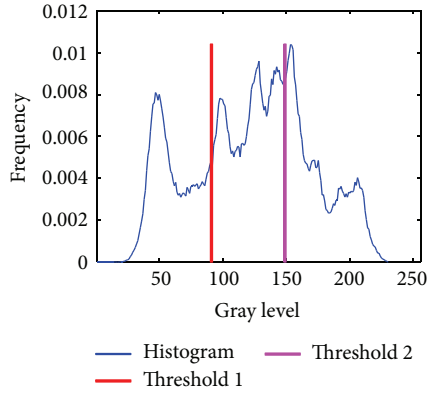
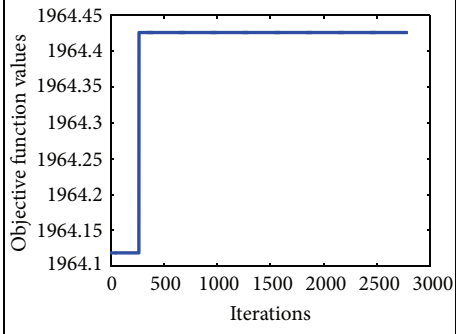
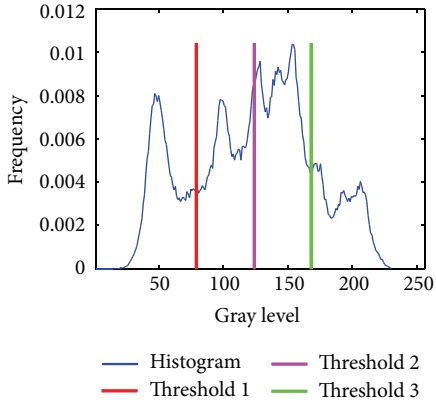
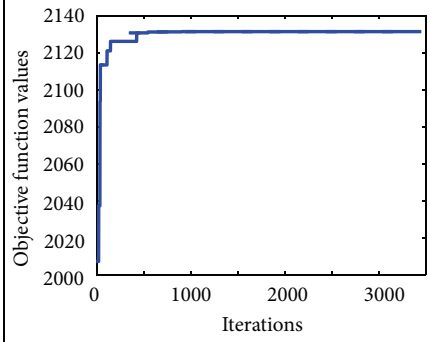
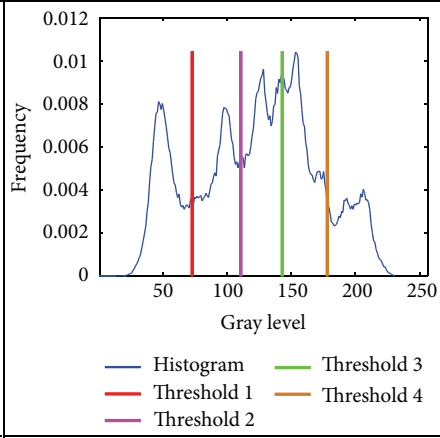
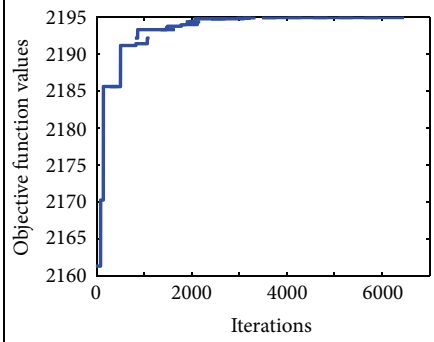
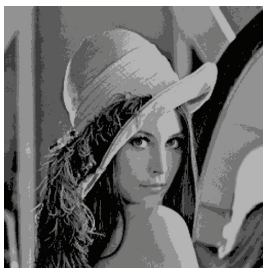
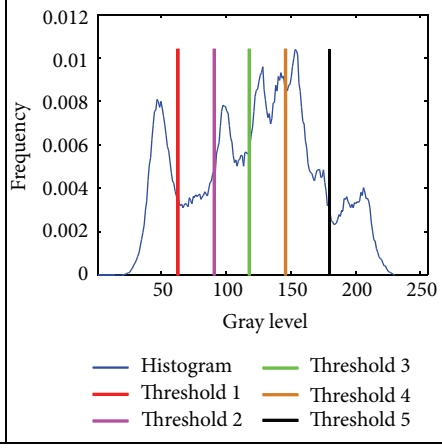
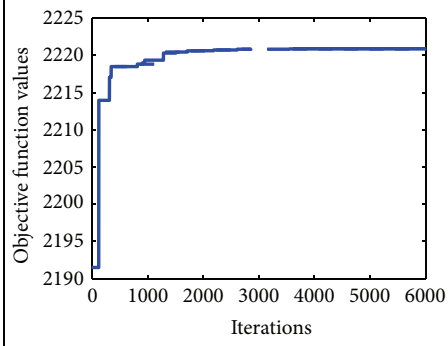
th = 2			
th = 3			
th = 4			
th = 5			

Table 6: Results after appling the HSMA using Otsu's over the Baboon image.

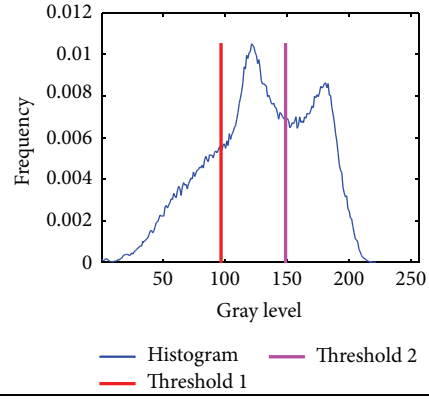
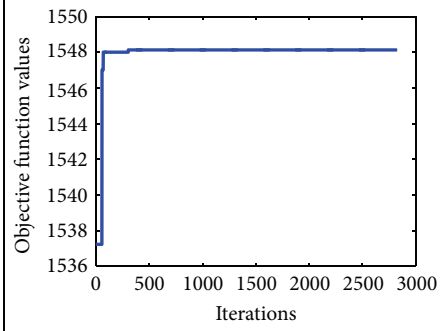
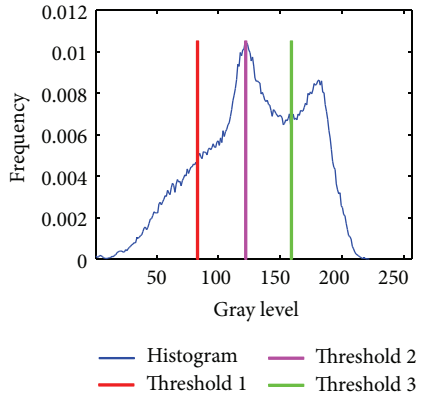
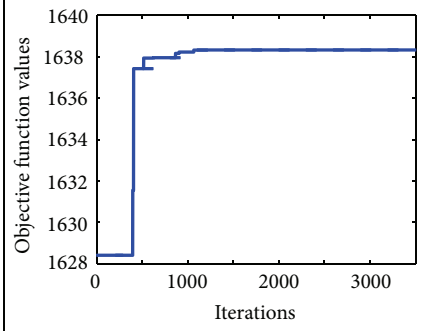
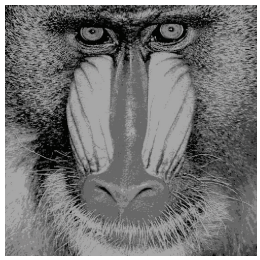
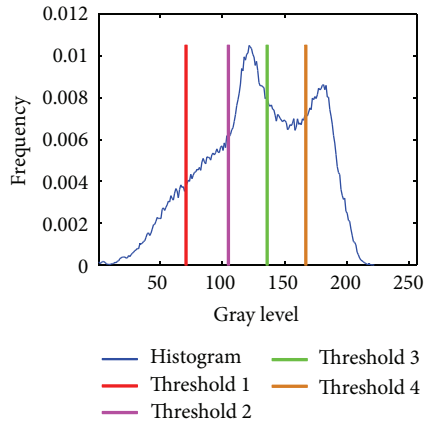
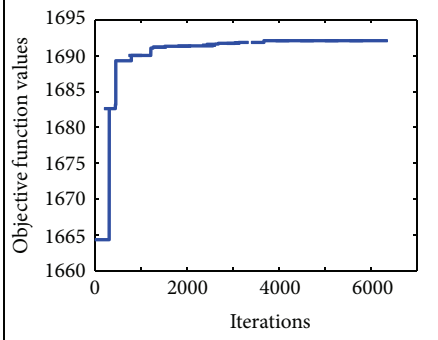
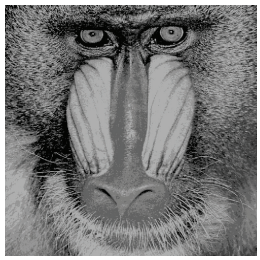
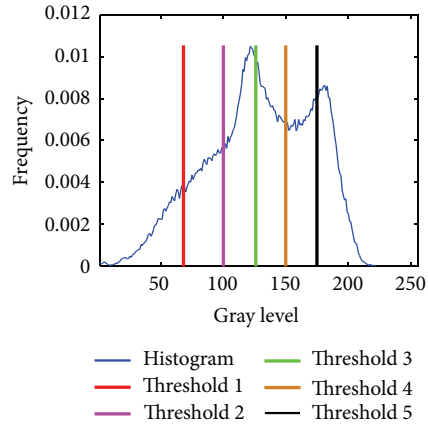
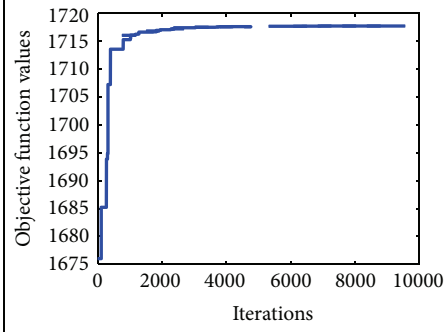
th = 2			
th = 3			
th = 4			
th = 5			

Table 7: Results after appling the HSMA using Otsu's over the Hunter image.

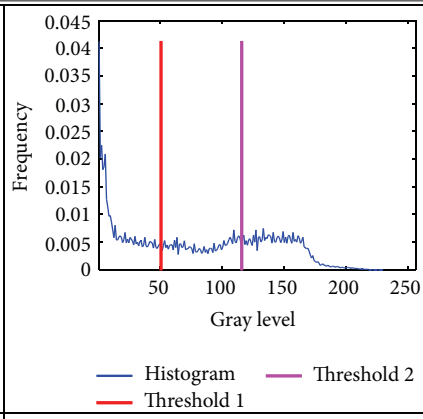
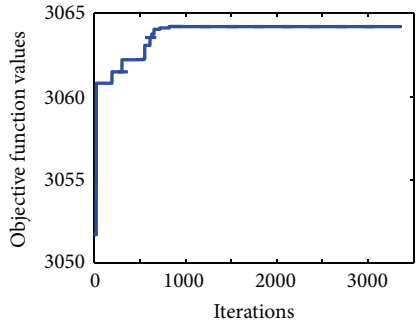
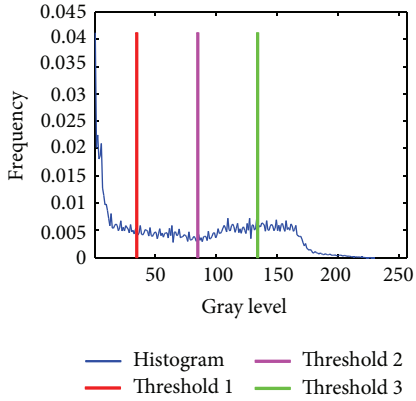
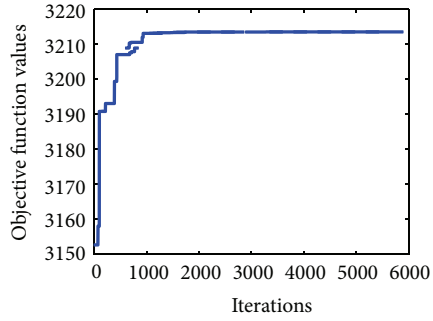
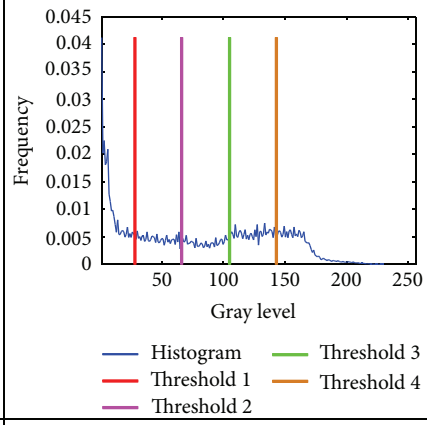
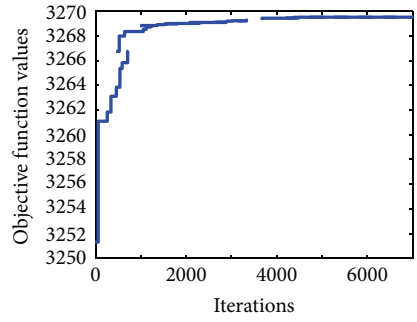
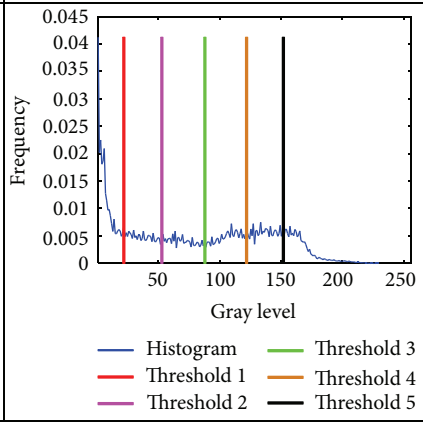
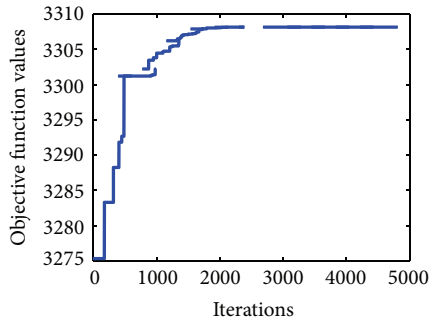
th = 2			
th = 3			
th = 4			
th = 5			

Table 8: Results after appling the HSMA using Otsu's over the Butterfly image.

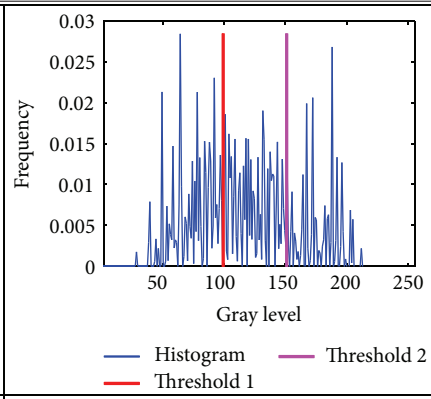
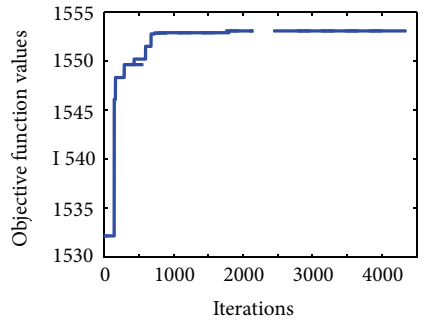
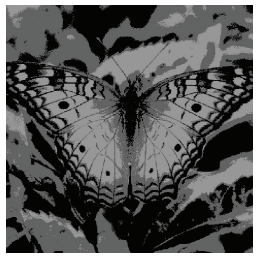
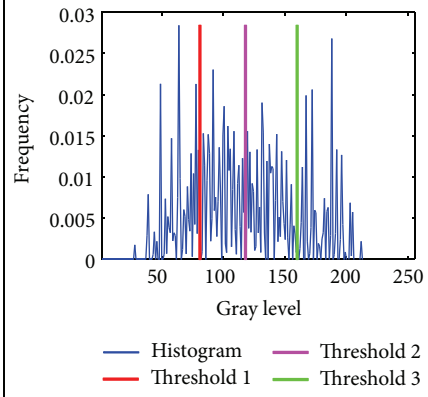
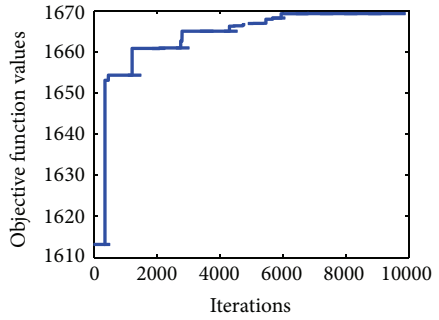
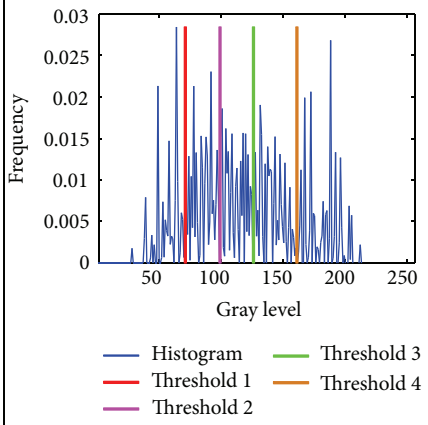
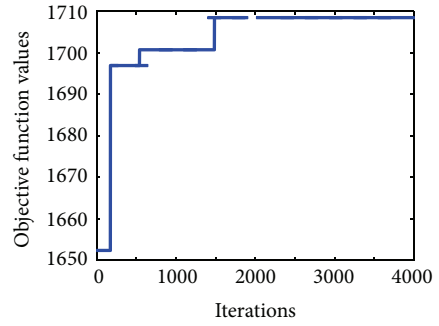
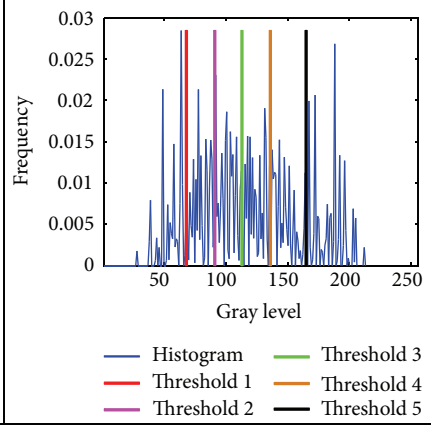
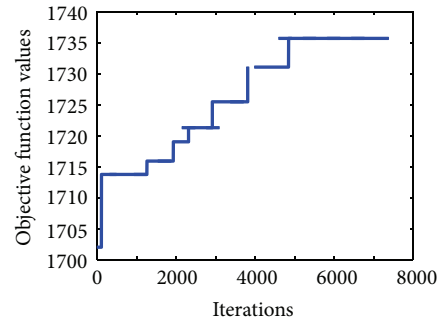
th = 2			
th = 3			
th = 4			
th = 5			

Table 9: Results after appling the HSMA using Kapur's function to the set of benchmark images.

Image	k	Thresholds x^c_{best}	PSNR	STD
Camera man	2	128, 196	13.626	$3.60E-15$
	3	44, 103, 196	14.460	$1.40E-03$
	4	44, 96, 146, 196	20.153	$1.20E-03$
	5	24, 60, 98, 146, 196	20.661	$2.75E-02$
Lena	2	96, 163	14.638	$3.60E-15$
	3	23, 96, 163	16.218	$7.66E-02$
	4	23, 80, 125, 173	19.287	$1.44E-14$
	5	23, 71, 109, 144, 180	21.047	$1.22E-02$
Baboon	2	79, 143	16.016	$1.08E-14$
	3	79, 143, 231	16.016	$7.19E-02$
	4	44, 98, 152, 231	18.485	$8.47E-02$
	5	33, 74, 114, 159, 231	20.507	$1.08E-14$
Hunter	2	92, 179	15.206	$1.44E-14$
	3	59, 117, 179	18.500	$4.16E-04$
	4	44, 89, 133, 179	21.065	$4.31E-04$
	5	44, 89, 133, 179, 222	21.086	$3.43E-02$
Airplane	2	70, 171	15.758	$3.30E-03$
	3	68, 126, 182	18.810	$1.08E-14$
	4	68, 126, 182, 232	18.810	$1.82E-01$
	5	64, 104, 143, 184, 232	20.321	$1.80E-01$
Peppers	2	66, 143	16.265	$7.21E-15$
	3	62, 112, 162	18.367	$1.80E-14$
	4	62, 112, 162, 227	18.376	$2.39E-02$
	5	48, 86, 127, 171, 227	18.827	$4.17E-04$
Living room	2	89, 170	14.631	$1.40E-03$
	3	47, 103, 175	17.146	$2.70E-03$
	4	47, 98, 149, 197	19.144	$1.34E-02$
	5	42, 85, 125, 162, 197	21.160	$1.89E-02$
Blonde	2	125, 203	12.244	$1.44E-14$
	3	65, 134, 203	16.878	$1.00E-03$
	4	65, 113, 155, 203	20.107	$5.50E-03$
	5	65, 113, 155, 203, 229	20.107	$4.48E-02$
Bridge	2	94, 171	13.529	$7.40E-03$
	3	65, 131, 195	16.806	$1.44E-02$
	4	53, 102, 151, 199	18.902	$2.47E-02$
	5	40, 85, 131, 171, 211	20.268	$2.12E-02$
Butterfly	2	27, 213	8.1930	$2.25E-02$
	3	27, 120, 213	13.415	$8.60E-04$
	4	27, 96, 144, 213	16.725	$3.80E-03$
	5	27, 83, 118, 152, 213	19.413	$3.90E-03$
Lake	2	91, 163	14.713	$1.44E-14$
	3	73, 120, 170	16.441	$3.05E-04$
	4	69, 112, 156, 195	17.455	$4.53E-02$
	5	62, 96, 131, 166, 198	18.774	$3.66E-02$

Table 10: Results after appling the HSMA using Kapur's over the Camera man image.

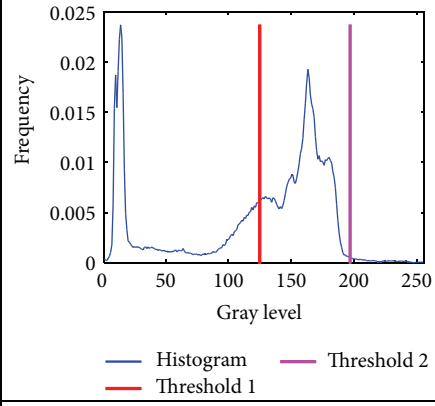
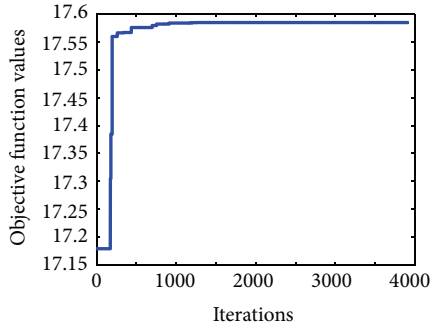
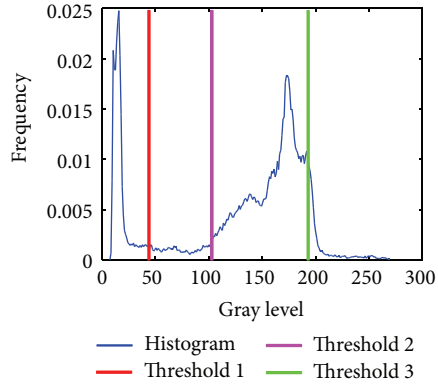
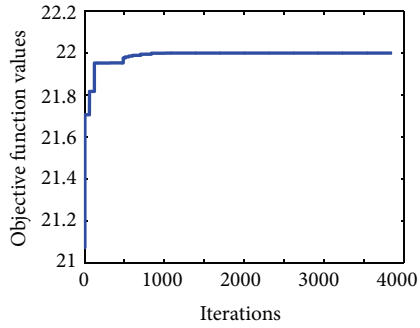
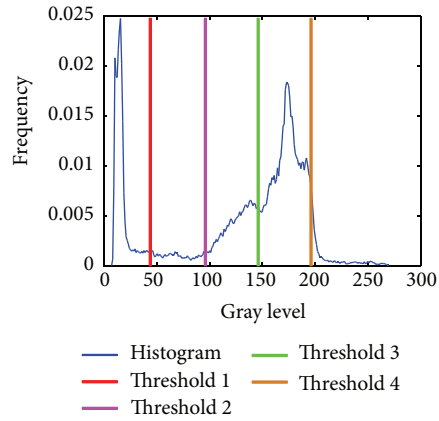
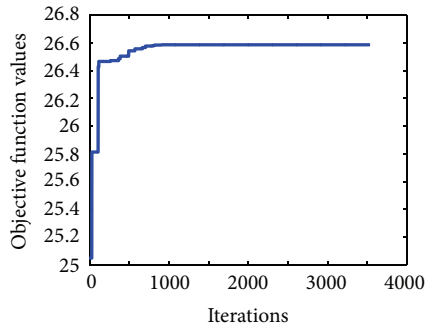
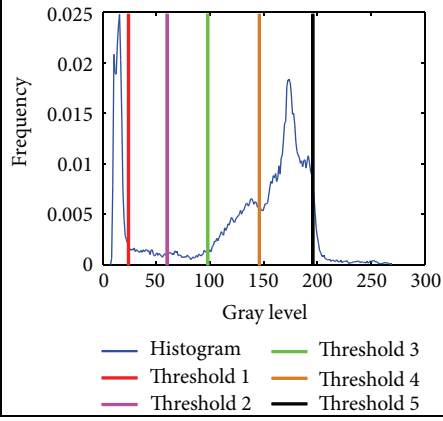
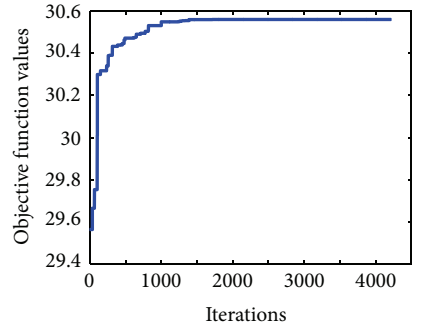
th = 2			
th = 3			
th = 4			
th = 5			

Table 11: Results after appling the HSMA using Kapur's over Lena's image.

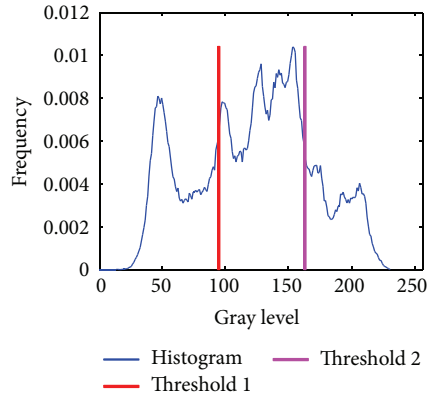
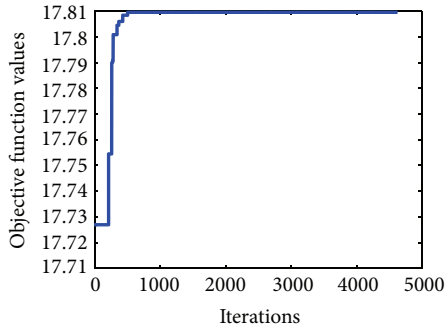
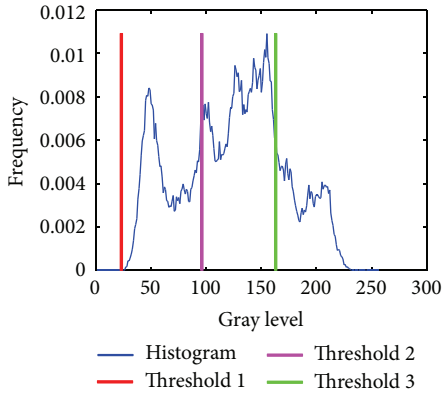
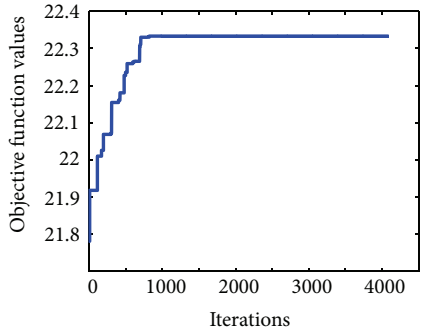
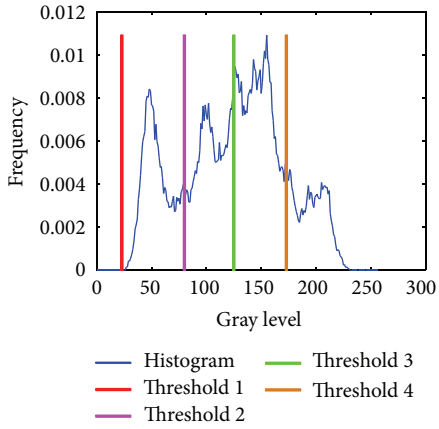
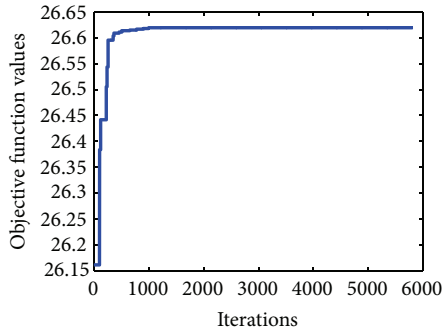
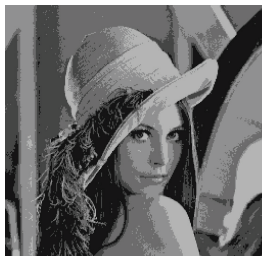
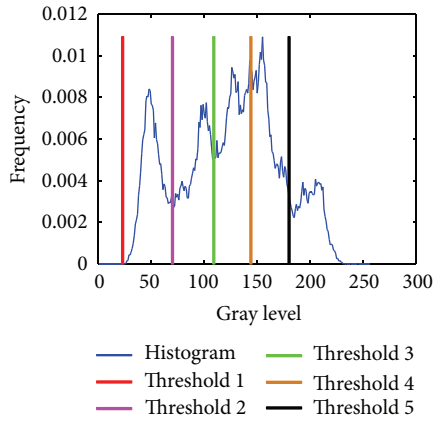
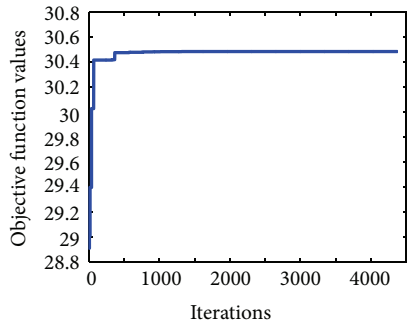
th = 2			
th = 3			
th = 4			
th = 5			

Table 12: Results after appling the HSMA using Kapur's over the Baboon image.

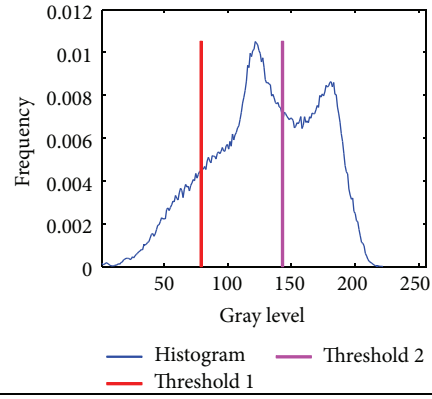
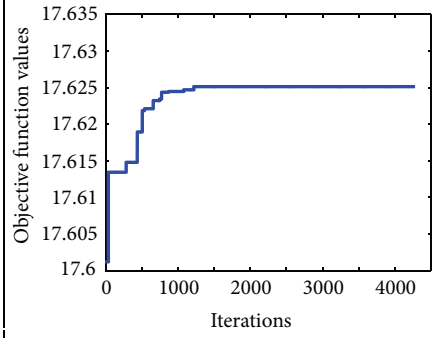
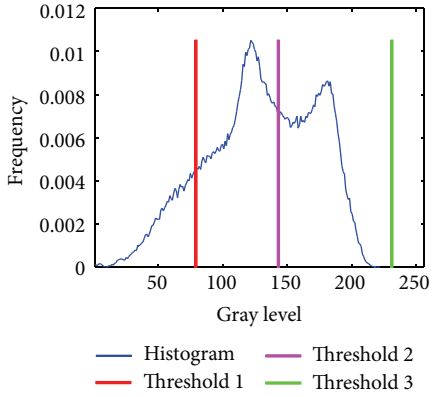
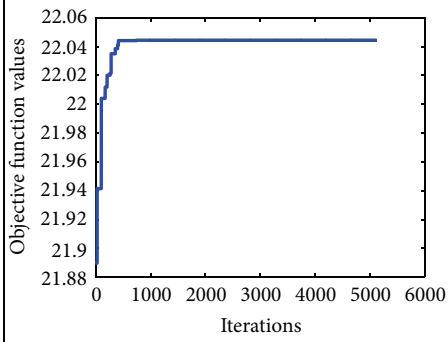
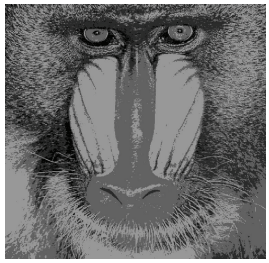
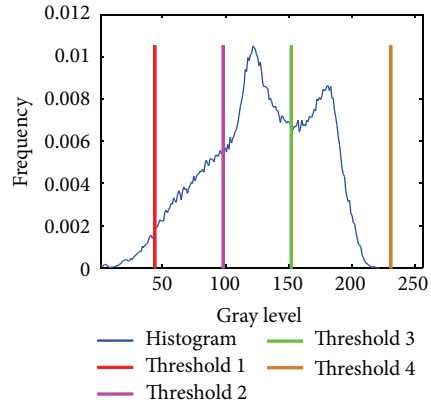
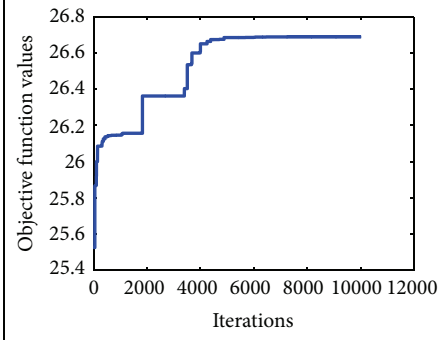
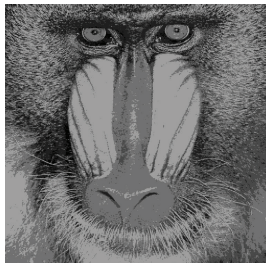
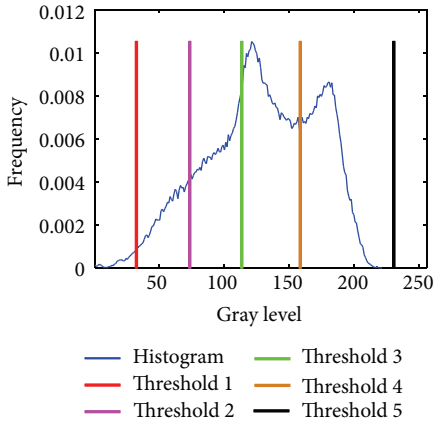
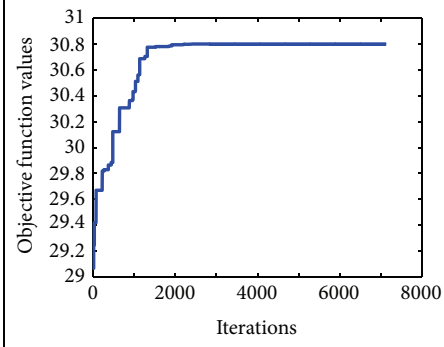
th = 2			
th = 3			
th = 4			
th = 5			

Table 13: Results after appling the HSMA using Kapur's over the Hunter image.

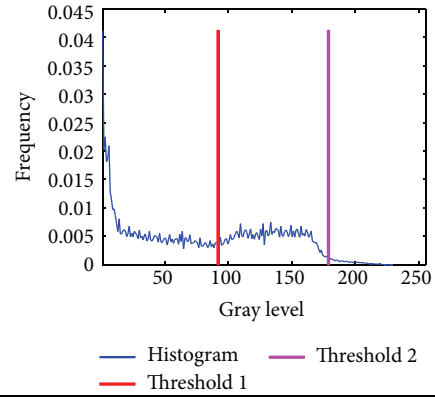
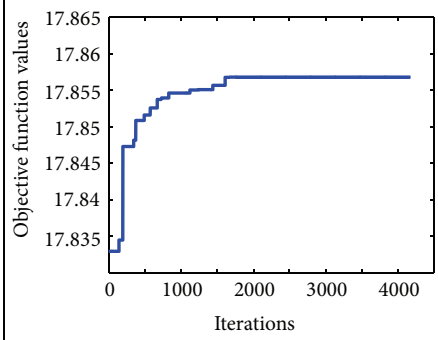
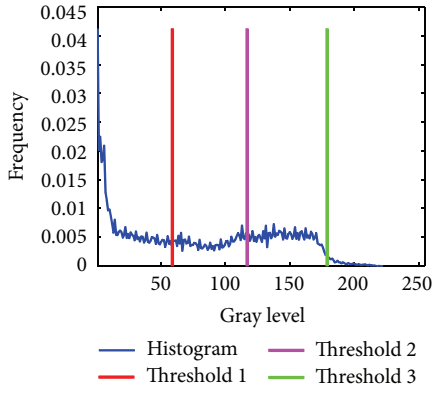
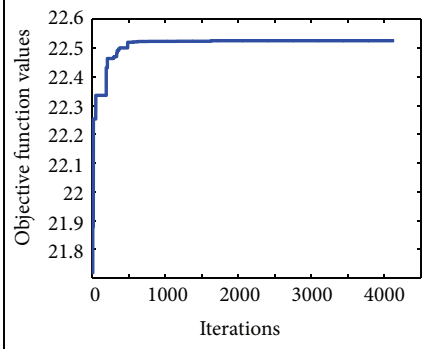
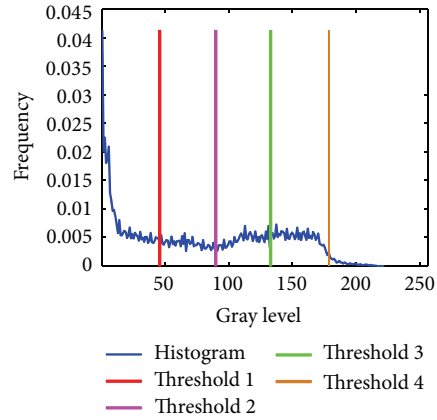
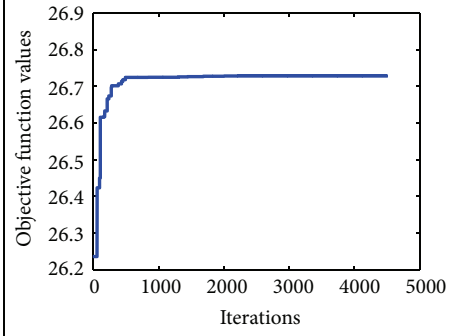
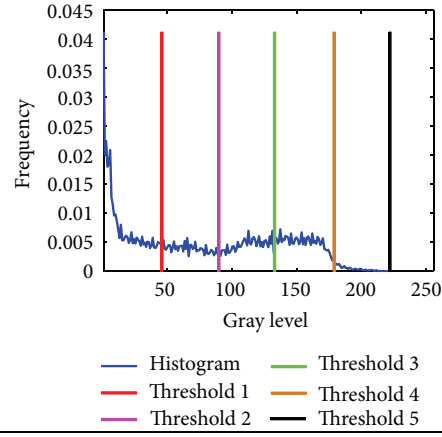
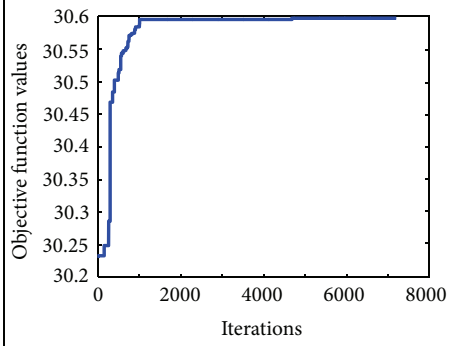
th = 2			
th = 3			
th = 4			
th = 5			

Table 14: Results after appling the HSMA using Kapur's over the Butterfly image.

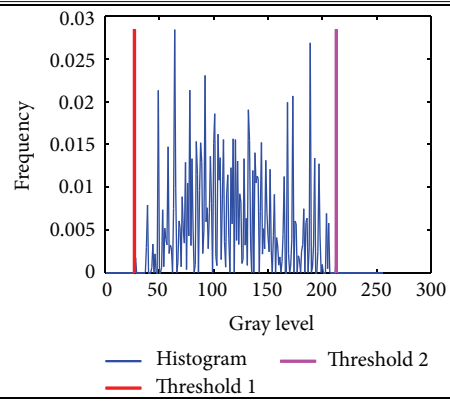
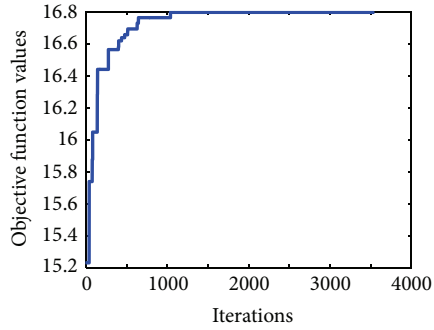
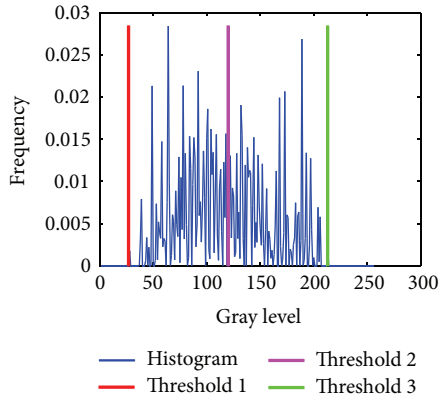
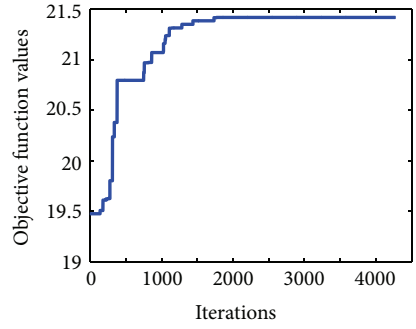
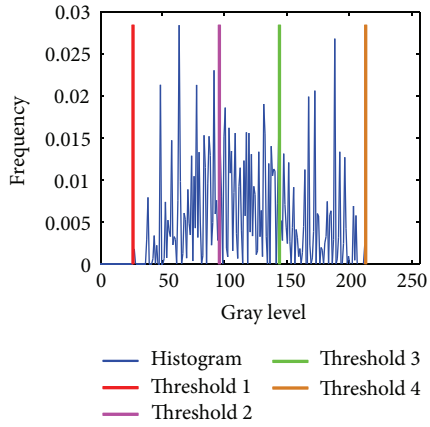
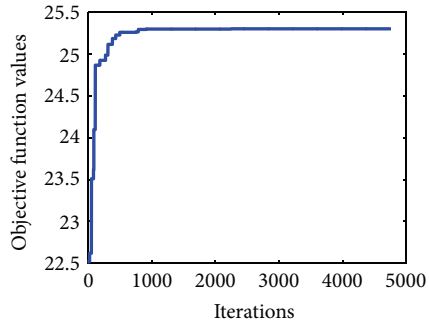
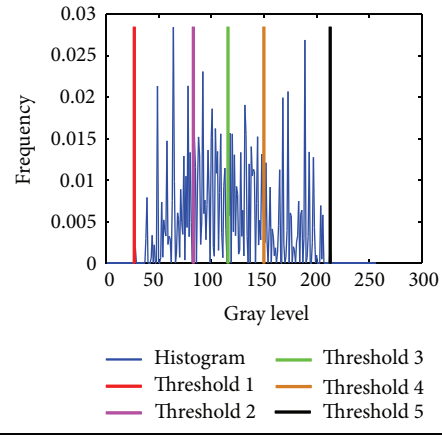
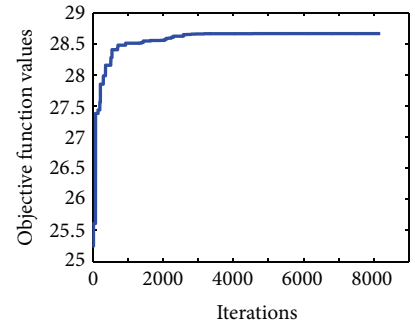
th = 2			
th = 3			
th = 4			
th = 5			

Table 15: P values produced by Wilcoxon's test comparing Otsu versus Kapur over the averaged PSNR from Tables 3 and 9.

Image	k	P-value Otsu versus Kapur
Camera man	2	$1.0425E-16$
	3	$2.1435E-15$
	4	$2.6067E-16$
	5	$6.2260E-16$
Lena	2	$1.0425E-16$
	3	$9.4577E-15$
	4	$9.7127E-15$
	5	$1.2356E-12$
Baboon	2	$1.0425E-16$
	3	$1.7500E-02$
	4	$5.3417E-14$
	5	$1.4013E-14$
Hunter	2	$1.0425E-16$
	3	$2.6067E-16$
	4	$6.6386E-14$
	5	$6.4677E-15$
Airplane	2	$1.0425E-16$
	3	$4.6500E-02$
	4	$1.7438E-13$
	5	$6.2475E-13$
Peppers	2	$1.0425E-16$
	3	$1.0425E-16$
	4	$6.0498E-15$
	5	$5.3194E-15$
Living room	2	$3.2405E-15$
	3	$1.4223E-15$
	4	$1.3175E-14$
	5	$6.2220E-14$
Blonde	2	$1.0425E-16$
	3	$2.6067E-16$
	4	$4.0480E-13$
	5	$7.8167E-04$
Bridge	2	$7.3588E-06$
	3	$1.1300E-04$
	4	$1.9400E-02$
	5	$2.1900E-02$
Butterfly	2	$1.1615E-14$
	3	$2.5697E-14$
	4	$3.7190E-13$
	5	$1.7941E-06$
Lake	2	$1.0425E-16$
	3	$5.6120E-16$
	4	$9.8174E-14$
	5	$2.2292E-14$

the Kapur criterion. The second analyses the comparison between the HSMA and other state-of-the-art approaches.

5.3.1. Comparison between Otsu and Kapur HSMA. In order to statistically compare the results from Tables 3 and 9, a nonparametric significance proof known as the Wilcoxon's rank test [36, 37] for 35 independent samples has been conducted. Such proof allows assessing result differences among two related methods. The analysis is performed considering a 5% significance level over the peak-to-signal ratio (PSNR) data corresponding to the five threshold points. Table 15 reports the P values produced by Wilcoxon's test for a pairwise comparison of the PSNR values between the Otsu and Kapur objective functions. As a null hypothesis, it is assumed that there is no difference between the values of the two objective functions. The alternative hypothesis considers an existent difference between the values of both approaches. All P values reported in Table 15 are less than 0.05 (5% significance level) which is a strong evidence against the null hypothesis, indicating that the Otsu PSNR mean values for the performance are statistically better and it has not occurred by chance.

5.3.2. Comparison among HSMA and Other MT Approaches. The results produced by HSMA have been compared with those generated by state-of-the-art thresholding methods such genetic algorithms (GA) [15], particle swarm optimization (PSO) [18], and bacterial foraging (BF) [19].

All the algorithms run 35 times over each selected image. The images at this test are the same as in Sections 5.2 and 5.1 (Camera man, Lena, Baboon, Hunter, and Butterfly). For each image, the PSNR, the STD, and the mean of the objective function values are calculated. Moreover, the entire test is performed using both Otsu's and Kapur's objective functions.

Table 16 presents the computed values for a reduced benchmark test (five images). It is clear that the HSMA delivers better performance than the others. Such values are computed using the Otsu's method as the objective function. On the other hand, the same experiment has been performed using the Kapur's method. Using the same criteria (as those described for the Otsu's method), the algorithm runs over 35 times at each image. The results of this experiment are presented in Table 17 and show that the proposed HSMA algorithm is better in comparison with the GA, PSO, and BF.

6. Conclusions

In this paper, a MT-based method on the original harmony search algorithm (HSA) is presented. The approach combines the good search capabilities of HSA algorithm and the use of some objective functions that have been proposed by the popular MT methods of Otsu and Kapur. In order to measure the performance of the proposed approach, the peak signal-to-noise ratio (PSNR) is used to assess the segmentation quality by considering the coincidences between the segmented and the original images. In this work, a simple HSA implementation without any modification is considered in order to demonstrate that it can be applied to image processing tasks.

The study explores the comparison between two versions of HSMA: one employs the Otsu objective function while the other uses the Kapur criterion. Results show that the

Table 16: Comparisons between HSMA, GA, PSO, and BF, applied over the selected test images using Otsu's method.

Image	k	HSMA			GA			PSO			BF		
		PSNR	STD	Mean	PSNR	STD	Mean	PSNR	STD	Mean	PSNR	STD	Mean
Camera man	2	17.247	$2.30E-12$	3651.9	17.048	0.0232	3604.5	17.033	0.0341	3598.3	17.058	0.0345	3590.9
	3	20.211	$1.55E-02$	3727.4	17.573	0.1455	3678.3	19.219	0.2345	3662.7	20.035	0.2459	3657.5
	4	21.533	$2.76E-12$	3782.4	20.523	0.2232	3781.5	21.254	0.3142	3777.4	21.209	0.4560	3761.4
	5	23.282	$5.30E-03$	3813.7	21.369	0.4589	3766.4	22.095	0.5089	3741.6	22.237	0.5089	3789.8
Lena	2	15.401	$9.22E-13$	1964.4	15.040	0.0049	1960.9	15.077	0.0033	1961.4	15.031	$2.99E-04$	1961.5
	3	17.427	$2.99E-02$	2131.4	17.304	0.1100	2126.4	17.276	0.0390	2127.7	17.401	0.0061	2128.0
	4	18.763	$2.77E-01$	2194.9	17.920	0.2594	2173.7	18.305	0.1810	2180.6	18.507	0.0081	2189.0
	5	19.443	$3.04E-01$	2218.7	18.402	0.3048	2196.2	18.770	0.2181	2212.5	19.001	0.0502	2215.6
Baboon	2	15.422	$6.92E-13$	1548.1	15.304	0.0031	1547.6	15.088	0.0077	1547.9	15.353	$8.88E-04$	1548.0
	3	17.709	$1.92E-02$	1638.3	17.505	0.1750	1633.5	17.603	0.0816	1635.3	17.074	0.0287	1637.0
	4	20.289	$5.82E-02$	1692.1	18.708	0.2707	1677.7	19.233	0.0853	1684.3	19.654	0.0336	1690.7
	5	21.713	$4.40E-01$	1717.5	20.203	0.3048	1712.9	20.526	0.1899	1712.9	21.160	0.1065	1716.7
Hunter	2	17.875	$2.30E-12$	3054.2	17.088	0.0470	3064.1	17.932	0.2534	3064.1	17.508	0.0322	3064.1
	3	20.350	$2.30E-12$	3213.4	20.045	0.1930	3212.9	19.940	0.9727	3212.4	20.350	0.9627	3213.4
	4	22.203	$1.22E-02$	3269.5	20.836	0.6478	3268.4	21.128	2.2936	3266.3	21.089	2.2936	3266.3
	5	23.703	$1.84E-12$	3308.1	21.284	1.6202	3305.6	22.026	4.1811	3276.3	22.804	3.6102	3291.1
Butterfly	2	13.934	$7.30E-02$	1553.0	13.007	0.0426	1553.0	13.092	0.0846	1553.0	13.890	0.0643	1553.0
	3	16.932	$6.17E-01$	1669.2	15.811	0.3586	1669.0	17.261	2.6268	1665.7	17.285	1.2113	1667.2
	4	19.259	$3.07E+00$	1708.3	17.104	0.6253	1709.9	17.005	3.7976	1702.9	17.128	2.2120	1707.0
	5	21.450	$3.87E+00$	1728.0	18.593	0.5968	1734.4	18.099	6.0747	1730.7	18.9061	3.5217	1733.0

Table 17: Comparison between HSMA GA, PSO, and BF, applied over selected test images using Kapur's method.

Image	k	HSMA			GA			PSO			BF		
		PSNR	STD	Mean	PSNR	STD	Mean	PSNR	STD	Mean	PSNR	STD	Mean
Camera man	2	13.626	$3.60E-15$	17.584	11.941	0.1270	15.341	12.259	0.1001	16.071	12.264	0.0041	16.768
	3	14.460	$1.40E-03$	22.007	14.827	0.2136	20.600	15.211	0.1107	21.125	15.250	0.0075	21.498
	4	20.153	$1.20E-03$	26.586	17.166	0.2857	24.267	18.000	0.2005	25.050	18.406	0.0081	25.093
	5	20.661	$2.75E-02$	30.553	19.795	0.3528	28.326	20.963	0.2734	28.365	21.211	0.0741	30.026
Lena	2	14.638	$3.60E-15$	17.809	12.334	0.0049	16.122	12.345	0.0033	16.916	12.345	$2.99E-4$	16.605
	3	16.218	$7.66E-02$	22.306	14.995	0.1100	20.920	15.133	0.0390	20.468	15.133	0.0061	20.812
	4	19.287	$1.44E-14$	26.619	17.089	0.2594	23.569	17.838	0.1810	24.449	17.089	0.0081	26.214
	5	21.047	$1.22E-02$	30.485	19.549	0.3043	27.213	20.442	0.2181	27.526	19.549	0.0502	28.046
Baboon	2	16.016	$1.08E-14$	17.625	12.184	0.0567	16.425	12.213	0.0077	16.811	12.216	$8.88E-4$	16.889
	3	16.016	$7.19E-02$	22.117	14.745	0.1580	21.069	15.008	0.0816	21.088	15.211	0.0287	21.630
	4	18.485	$8.47E-02$	26.671	16.935	0.1765	25.489	17.574	0.0853	24.375	17.999	0.0336	25.446
	5	20.507	$1.08E-14$	30.800	19.662	0.2775	29.601	20.224	0.1899	30.994	20.720	0.1065	30.887
Hunter	2	15.206	$1.44E-14$	17.856	12.349	0.0148	16.150	12.370	0.0068	15.580	12.373	0.0033	16.795
	3	18.500	$4.16E-04$	22.525	14.838	0.1741	21.026	15.128	0.0936	20.639	15.553	0.1155	21.860
	4	21.065	$4.31E-04$	26.728	17.218	0.2192	25.509	18.040	0.1560	27.085	18.381	0.0055	26.230
	5	21.086	$3.43E-02$	30.612	19.563	0.3466	29.042	20.533	0.2720	29.013	21.256	0.0028	28.856
Butterfly	2	8.1930	$2.25E-02$	16.791	10.470	0.0872	15.481	10.474	0.0025	14.098	10.474	0.0014	15.784
	3	13.415	$8.60E-04$	21.417	11.628	0.2021	20.042	12.313	0.1880	19.340	12.754	0.0118	21.308
	4	16.725	$3.80E-03$	25.292	13.314	0.2596	23.980	14.231	0.2473	25.190	14.877	0.0166	25.963
	5	19.413	$3.90E-03$	28.664	15.756	0.3977	27.411	16.337	0.2821	27.004	16.828	0.0877	27.980

Otsu function delivers better results than the Kapur criterion. Such conclusion has been statistically proved considering the Wilcoxon test.

The proposed approach has been compared to other techniques that implement different optimization algorithms like GA, PSO, and BF. The efficiency of the algorithm has been evaluated in terms of the PSNR index and the STD value. Experimental results provide evidence on the outstanding performance, accuracy, and convergence of the proposed algorithm in comparison to other methods. Although the results offer evidence to demonstrate that the standard HSA method can yield good results on complicated images, the aim of our paper is not to devise an MT algorithm that could beat all currently available methods, but to show that harmony search algorithms can be effectively considered as an attractive alternative for this purpose.

Acknowledgment

The proposed algorithm is part of the vision system used by a biped robot supported under the Grant CONACYT CB 181053. The first author acknowledges The National Council of Science and Technology of Mexico (CONACyT) for the doctoral Grant number 215517 and The Youth Institute of Jalisco (IJJ) for partially support this research.

References

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison Wesley, Reading, Mass, USA, 1992.

[2] R. Guo and S. M. Pandit, "Automatic threshold selection based on histogram modes and a discriminant criterion," *Machine Vision and Applications*, vol. 10, no. 5-6, pp. 331–338, 1998.

[3] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, no. 9, pp. 1277–1294, 1993.

[4] P. K. Sahoo, S. Soltani, and A. K. C. Wong, "A survey of thresholding techniques," *Computer Vision, Graphics and Image Processing*, vol. 41, no. 2, pp. 233–260, 1988.

[5] W. Snyder, G. Bilbro, A. Logenthiran, and S. Rajala, "Optimal thresholding: a new approach," *Pattern Recognition Letters*, vol. 11, no. 12, pp. 803–809, 1990.

[6] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[7] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Computer Vision, Graphics, & Image Processing*, vol. 29, no. 3, pp. 273–285, 1985.

[8] J. Kittler and J. Illingworth, "Minimum error thresholding," *Pattern Recognition*, vol. 19, no. 1, pp. 41–47, 1986.

[9] K. Hammouche, M. Diaf, and P. Siarry, "A comparative study of various meta-heuristic techniques applied to the multilevel thresholding problem," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 5, pp. 676–688, 2010.

[10] 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.

[11] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[12] F. Glover, "Tabu Search—part I," *ORSA Journal on Computing*, vol. 1, article 3, pp. 190–206, 1989.

[13] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[14] C. Lai and D. Tseng, "A hybrid approach using Gaussian smoothing and genetic algorithm for multilevel thresholding," *International Journal of Hybrid Intelligent Systems*, vol. 1, pp. 143–152, 2004.

[15] P.-Y. Yin, "A fast scheme for optimal thresholding using genetic algorithms," *Signal Processing*, vol. 72, no. 2, pp. 85–95, 1999.

[16] 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.

[17] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR06, Computer Engineering Department, Engineering Faculty, Erciyes University, 2005.

[18] B. Akay, "A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding," *Applied Soft Computing*, vol. 13, no. 6, pp. 3066–3091, 2012.

[19] P. D. Sathya and R. Kayalvizhi, "Optimal multilevel thresholding using bacterial foraging algorithm," *Expert Systems with Applications*, vol. 38, no. 12, pp. 15549–15564, 2011.

[20] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[21] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[22] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36-38, pp. 3902–3933, 2005.

[23] K. S. Lee, Z. W. Geem, S.-H. Lee, and K.-W. Bae, "The harmony search heuristic algorithm for discrete structural optimization," *Engineering Optimization*, vol. 37, no. 7, pp. 663–684, 2005.

[24] J. H. Kim, Z. W. Geem, and E. S. Kim, "Parameter estimation of the nonlinear Muskingum model using harmony search," *Journal of the American Water Resources Association*, vol. 37, no. 5, pp. 1131–1138, 2001.

[25] Z. W. Geem, "Optimal cost design of water distribution networks using harmony search," *Engineering Optimization*, vol. 38, no. 3, pp. 259–280, 2006.

[26] Z. W. Geem, K. S. Lee, and Y. J. Park, "Application of harmony search to vehicle routing," *American Journal of Applied Sciences*, vol. 2, pp. 1552–1557, 2005.

[27] A. Vasebi, M. Fesanghary, and S. M. T. Bathaee, "Combined heat and power economic dispatch by harmony search algorithm," *International Journal of Electrical Power and Energy Systems*, vol. 29, no. 10, pp. 713–719, 2007.

[28] S. O. Degertekin, "Optimum design of steel frames using harmony search algorithm," *Structural and Multidisciplinary Optimization*, vol. 36, no. 4, pp. 393–401, 2008.

[29] E. Cuevas, N. Ortega-Sánchez, D. Zaldivar, and M. Pérez-Cisneros, "Circle detection by harmony search optimization," *Journal of Intelligent and Robotic Systems*, pp. 1–18, 2011.

[30] O. M. Alia and R. Mandava, "The variants of the harmony search algorithm: an overview," *Artificial Intelligence Review*, vol. 36, no. 1, pp. 49–68, 2011.

[31] F. G. Lobo, C. F. Lima, and Z. Michalewicz, Eds., *Parameter Setting in Evolutionary Algorithms*, vol. 54 of *Studies in Computational Intelligence*, Springer, Berlin, Germany, 2007.

[32] C. B. B. Costa, M. R. W. MacIel, and R. M. Filho, "Factorial design technique applied to genetic algorithm parameters in a batch cooling crystallization optimisation," *Computers and Chemical Engineering*, vol. 29, no. 10, pp. 2229–2241, 2005.

[33] A. Khadwilard, P. Luangpaiboon, and P. Pongcharoen, "Full factorial experimental design for parameters selection of harmony search Algorithm," *The Journal of Industrial Technology*, vol. 8, no. 2, pp. 1–10, 2012.

[34] G. E. P. Box, W. G. Hunter, and J. S. Hunter, *Statistic for Experimenters—An Introduction to Design Data Analysis and Model Building*, John Wiley & Sons, New York, NY, USA, 1978.

[35] S. K. Pal, D. Bhandari, and M. K. Kundu, "Genetic algorithms for optimal image enhancement," *Pattern Recognition Letters*, vol. 15, no. 3, pp. 261–271, 1994.

[36] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, pp. 80–83, 1945.

[37] S. Garcia, D. Molina, M. Lozano, and F. Herrera, "Astudy on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special session on real parameter optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2008.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 237984, 9 pages
http://dx.doi.org/10.1155/2013/237984

Research Article

Firefly Algorithm for Polynomial Bézier Surface Parameterization

Akemi Gálvez[1] and Andrés Iglesias[1,2]

[1] *Department of Applied Mathematics and Computational Sciences, E.T.S.I. Caminos, Canales y Puertos, University of Cantabria, Avenida de los Castros, s/n, 39005 Santander, Spain*
[2] *Department of Information Science, Faculty of Sciences, Toho University, 2-2-1 Miyama, Funabashi 274-8510, Japan*

Correspondence should be addressed to Andrés Iglesias; iglesias@unican.es

Received 21 June 2013; Accepted 7 August 2013

Academic Editor: Xin-She Yang

A classical issue in many applied fields is to obtain an approximating surface to a given set of data points. This problem arises in Computer-Aided Design and Manufacturing (CAD/CAM), virtual reality, medical imaging, computer graphics, computer animation, and many others. Very often, the preferred approximating surface is polynomial, usually described in parametric form. This leads to the problem of determining suitable parametric values for the data points, the so-called surface parameterization. In real-world settings, data points are generally irregularly sampled and subjected to measurement noise, leading to a very difficult nonlinear continuous optimization problem, unsolvable with standard optimization techniques. This paper solves the parameterization problem for polynomial Bézier surfaces by applying the firefly algorithm, a powerful nature-inspired metaheuristic algorithm introduced recently to address difficult optimization problems. The method has been successfully applied to some illustrative examples of open and closed surfaces, including shapes with singularities. Our results show that the method performs very well, being able to yield the best approximating surface with a high degree of accuracy.

1. Introduction

Obtaining a curve or surface that approximates a given cloud of data points is a classical problem in several scientific and technological domains such as computer-aided design and manufacturing (CAD/CAM), virtual reality, medical imaging, computer graphics, computer animation, and many others. In real-world settings, data points come from real measurements of an existing geometric entity, as it typically happens in the construction of car bodies, ship hulls, airplane fuselage, and other free-form objects [1–8]. This process is also applied in the shoes industry, in archeology (reconstruction of archeological assets), in medicine (computed tomography), and in many other fields. The primary goal is to convert the real data from a physical object into a fully usable digital model, a process called reverse engineering. Such digital models are usually easier and cheaper to modify than their real counterparts, leading to a significant reduction of the costs associated with the processing and manufacturing time of the real goods they represent. Furthermore, due to their inherent digital nature, they become available anytime and anywhere, a very valuable feature in our current *digital-world* era.

Data points in reverse engineering are usually acquired through laser scanning and other digitizing methods (light digitizers, coordinate measuring machines, CT scanners, and tactile scanners) and are, therefore, subjected to measurement noise, irregular sampling, and other artifacts [7, 9]. Consequently, a good fitting of data is generally based on approximation schemes (where the curve/surface is expected to pass *near* the data points) rather than on interpolation (where the curve/surface is constrained to pass *through* all input data points). Because this is the typical case in many real-world industrial problems, in this paper we focus on the approximation scheme to a given set of irregularly sampled noisy data points.

There are two key components for a good approximation of data points: a proper choice of the approximating function

and a suitable parameter tuning. The usual models for data fitting in CAD/CAM and other industrial fields are free-form parametric entities, such as Bézier, B-spline, and NURBS, as they have a great flexibility and can represent well any smooth shape with only a few parameters, thus leading to substantial savings in terms of computer memory and storage capacity [10–17].

In this paper we focus particularly on the case of polynomial Bézier surfaces, a kind of free-form splines very popular in fields such as CAD/CAM and computer graphics. Bézier splines were developed independently in the early 60s by Paul de Casteljau and Pierre Bézier for the CAD systems of the French automotive companies Citröen and Renault, respectively. Mathematically, they are based on the Bernstein polynomials (see Section 4 for details), developed as early as 1912 but whose applicability to engineering design was unknown until the 60s. A Bézier curve is a linear combination of the Bernstein polynomials and vector coefficients called control points. The curve follows approximately the shape of its control polygon (the collection of segments joining the control points), and hence, it reacts to the movement of its control points by following a push-pull effect. This powerful feature was fundamental for the popularization of free-form curves and surfaces for interactive design. The generalization of this idea to surfaces leads to the Bézier surfaces, which are linear combinations of the control points (now arranged in a three-dimensional net) and the so-called tensor-product basis functions (given by the products of all possible combinations of univariate Bernstein polynomials in surface parameters u and v, resp.).

Although nowadays Bézier splines have been overtaken by the B-splines (developed during the 70s and of which the Bézier splines are a particular case), they played a key role in the current development of computer design. In addition to their historical value, they are still widely used today for different purposes, such as computer fonts (e.g., TrueType fonts, PostScript), computer animation (for simple movements of objects in programs such as Adobe Flash), and computer design (Adobe Photoshop, Corel Draw, Adobe illustrator). The reader is referred to [18–20] for further details about the subject. See also [21] for a nice historical approach written by some of the most prominent figures in the field.

Best approximation methods make commonly use of least-squares techniques [1, 8, 10, 13, 14, 22–28], where the goal is to obtain the relevant parameters of the polynomial approximating surface that fits the data points better in the least-squares sense. This problem is far from being trivial: because the surface is parametric, we are confronted with the problem of obtaining a suitable parameterization of the data points [18, 20]. As remarked in [29], the selection of an appropriate parameterization is essential for a good fitting. Unfortunately, it also becomes a very hard problem, specially for the cases of irregularly sampled noisy data points. In fact, it is well known that it leads to a very difficult overdetermined continuous nonlinear optimization problem. It is also multivariate, as it typically involves a large number of unknown variables for a large number of data points, a case that happens very often in real-world examples. Finally, it is usually a multimodal problem as well, because of the potential existence of several (global or local) optima of the objective function.

In this context, the present paper describes a new method to solve this challenging parameterization problem for free-form polynomial Bézier surfaces. Our method applies a powerful nature-inspired metaheuristic algorithm, called firefly algorithm, introduced recently by Professor Yang (Cambridge University) to solve difficult optimization problems. The trademark of the firefly algorithm is its search mechanism, inspired by the social behavior of the swarms of fireflies and the phenomenon of bioluminescent communication. The paper shows that this approach can be effectively applied to obtain an optimal approximating Bézier surface to a given set of noisy data points, provided that an adequate representation of the problem and a proper selection of the parameters are carried out. To check the performance of our approach, it has been applied to some illustrative examples of open and closed surfaces, including shapes with singularities. Our results show that the method performs very well, being able to yield the best approximating surface with a high degree of accuracy.

The structure of this paper is as follows: in Section 2 the previous work in the field is briefly reported. Then, the fundamentals and main ideas of the firefly algorithm, the method used in this paper, are briefly explained in Section 3. Our proposed firefly-based method for data fitting with Bézier surfaces is described in Section 4. The section begins with the description of the problem to be solved. Then, the application of the firefly algorithm to solve it is explained in detail. Some illustrative examples of its application to open and closed surfaces, including shapes with singularities, along with some implementation details are reported in Section 5. The paper closes with the main conclusions of this contribution and our plans for future work in the field.

2. Previous Work

The problem of data fitting through free-form parametric surfaces has been the subject of research for many years [1, 20, 30–35]. One of the most important problems regarding this issue is the surface parameterization, that is, the computation of suitable parametric values for the fitting surface to data points. In many practical situations, it is advisable to obtain a parameterization as similar as possible to the arc-length parameterization. The ultimate reason for this is that a constant step on the parametric domain automatically translates into a constant distance along an arc-length parameterized curve on the surface. In other words, for constant parameter intervals, the curve on the surface exhibits a point spacing that is as uniform as possible. Therefore, this parameterization is very convenient for surface interrogation issues, such as surface intersections or measuring distances on a surface [36, 37]. For instance, it has been traditionally applied in metrology for design and manufacturing, to collect measurement data from industrial parts of the designed and manufactured products. Many other industrial operations also require a uniform parameterization. For example, in computer controlled milling operations, the curve path followed by the milling machine must be parameterized such

that the cutter neither speeds up nor slows down along the path [9]. This property is only guaranteed when the curve path is parameterized with the arc-length parameterization. Consequently, this has been the preferred and most classical choice for surface parameterization.

Some recent papers have shown that the application of Artificial Intelligence techniques can achieve remarkable results regarding this parameterization problem [2, 5, 6, 38–40]. Most of these methods rely on some kind of neural networks, either standard neural networks [38], Kohonen's SOM (Self-Organizing Maps) nets [29, 39], or the Bernstein Basis Function (BBF) network [40]. In the case of surfaces, the network is used exclusively to order the data and create a grid of control vertices with quadrilateral topology [39]. After this preprocessing step, any standard surface reconstruction method (such as those referenced in the bibliography) has to be applied. In some other cases, the neural network approach is combined with partial differential equations [29] or other approaches. The generalization to functional networks (an extension of neural networks where the weights are replaced by functions) is also analyzed in [2, 5, 6, 41].

Due to their good behavior for complex optimization problems involving ambiguous and noisy data, there has recently been an increasing interest in applying nature-inspired optimization techniques (such as metaheuristics and evolutionary methods) to this problem. However, there are still few works reported in the literature. A previous paper in [42] describes the application of genetic algorithms and functional networks yielding pretty good results for both curves and surfaces. Other approaches are based on the application of metaheuristic techniques, which have been intensively applied to solve difficult optimization problems that cannot be tackled through traditional optimization algorithms. Recent schemes in this area are described in [4, 10] for particle swarm optimization (PSO), [3, 27, 28, 43] for genetic algorithms (GA), [44, 45] for artificial immune systems, [46] for estimation of distribution algorithms, and [11] for hybrid GA-PSO techniques. The method used in this paper also belongs to this category, as described in next section.

3. The Firefly Algorithm

The firefly algorithm is a nature-inspired metaheuristic algorithm introduced in 2008 by Yang to solve optimization problems [47, 48] (see also [49] for a recent modified version of this algorithm). The algorithm is based on the social flashing behavior of fireflies in nature. The key ingredients of the method are the variation of light intensity and formulation of attractiveness. In general, the attractiveness of an individual is assumed to be proportional to their brightness, which in turn is associated with the encoded objective function. The reader is kindly referred to [50] for a comprehensive review of the firefly algorithm and other nature-inspired metaheuristic approaches. See also [51] for a gentle introduction to metaheuristic applications in engineering optimization.

In the firefly algorithm, there are three particular idealized rules, which are based on some of the major flashing characteristics of real fireflies [47]. They are

(1) all fireflies are unisex, so that one firefly will be attracted to other fireflies regardless of their sex;

(2) the degree of attractiveness of a firefly is proportional to its brightness, which decreases as the distance from the other firefly increases due to the fact that the air absorbs light. For any two flashing fireflies, the less brighter one will move towards the brighter one. If there is not a brighter or more attractive firefly than a particular one, it will then move randomly;

(3) the brightness or light intensity of a firefly is determined by the value of the objective function of a given problem. For instance, for maximization problems, the light intensity can simply be proportional to the value of the objective function.

The distance between any two fireflies i and j, at positions $\mathbf{X}_i$ and $\mathbf{X}_j$, respectively, can be defined as a Cartesian or Euclidean distance as follows:

$$r_{ij} = \left\|\mathbf{X}_i - \mathbf{X}_j\right\| = \sqrt{\sum_{k=1}^{D}\left(x_{i,k} - x_{j,k}\right)^2}, \tag{1}$$

where $x_{i,k}$ is the k-th component of the spatial coordinate $\mathbf{X}_i$ of the i-th firefly and D is the number of dimensions.

In the firefly algorithm, as attractiveness function of a firefly j one should select any monotonically decreasing function of the distance to the chosen firefly, for example, the exponential function:

$$\beta = \beta_0 e^{-\gamma r_{ij}^{\mu}} \quad (\mu \geq 1), \tag{2}$$

where r_{ij} is the distance defined as in (1), β_0 is the initial attractiveness at $r = 0$, and γ is an absorption coefficient at the source which controls the decrease of the light intensity.

The movement of a firefly i which is attracted by a more attractive (i.e., brighter) firefly j is governed by the following evolution equation:

$$\mathbf{X}_i = \mathbf{X}_i + \beta_0 e^{-\gamma r_{ij}^{\mu}}\left(\mathbf{X}_j - \mathbf{X}_i\right) + \alpha\left(\sigma - \frac{1}{2}\right), \tag{3}$$

where the first term on the right-hand side is the current position of the firefly, the second term is used for considering the attractiveness of the firefly to light intensity seen by adjacent fireflies, and the third term is used for the random movement of a firefly in case there are not any brighter ones. The coefficient α is a randomization parameter determined by the problem of interest, while σ is a random number generator uniformly distributed in the space $[0, 1]$.

The method described in previous paragraphs corresponds to the original version of the firefly algorithm (FFA), as originally developed by its inventor. Since then, many different modifications and improvements on the original version have been developed, including the discrete FFA, multiobjective FFA, chaotic FFA, parallel FFA, elitist FFA, Lagrangian FFA, and many others, including its hybridization with other techniques. The interested reader is referred to the nice paper in [52] for a comprehensive, updated review and taxonomic classification of the firefly algorithms and all its variants and applications.

4. The Proposed Method

A free-form polynomial parametric surface is defined as [18, 19]:

$$\mathbf{S}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{ij}\phi_i(u)\,\varphi_j(v), \tag{4}$$

where $\{\mathbf{P}_{ij}\}_{i,j}$ are vector coefficients in $\mathbb{R}^3$ (usually referred to as the control points as they roughly control the shape of the surface), $\{\phi_i(u)\varphi_j(v)\}_{i,j}$ are the tensor-product functions obtained from two sets of basis functions (or *blending functions*) $\{\phi_i(u)\}_i$, and $\{\varphi_j(v)\}_j$, and (u,v) are the surface parameters, usually defined on a bounded rectangular domain $[\alpha_u,\beta_u]\times[\alpha_v,\beta_v]\subset\mathbb{R}^2$. Note that in this paper vectors are denoted in bold.

In this work we will focus on the particular case of free-form polynomial Bézier surfaces. In this case, (4) becomes

$$\mathbf{S}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u)\,\Psi_j^n(v), \tag{5}$$

where the blending functions $\Psi_k^d(\omega)$ are the Bernstein polynomials of index k and degree d, given by

$$\Psi_k^d(\omega) = \binom{d}{k}\omega^k(1-\omega)^{d-k}, \tag{6}$$

where

$$\binom{d}{k} = \frac{d!}{k!\,(d-k)!}, \tag{7}$$

and the surface parameters u, v are defined on the unit square $[0,1]\times[0,1]$. Note that, by convention, $0! = 1$.

Let us suppose now that we are given a set of data points $\{\mathbf{Q}_{k,l}\}_{k=1,\dots,p;l=1,\dots,q}$ in an ξ-dimensional space (usually $\xi = 2$ or $\xi = 3$). Our goal is to obtain the free-form polynomial Bézier surface $\mathbf{S}(u,v)$ that fits the data points better in the discrete least-squares sense. To do so, we have to compute the control points $\{\mathbf{P}_{i,j}\}_{i=0,\dots,m;j=0,\dots,n}$ of the approximating surface by minimizing the least-squares error, E, defined as the sum of squares of the residuals:

$$E = \sum_{k=1}^{p}\sum_{l=1}^{q}\left(\mathbf{Q}_{k,l} - \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u_k)\,\Psi_j^n(v_l)\right)^2. \tag{8}$$

In the case of irregularly sampled data points $\{\mathbf{Q}_r\}_{r=1,\dots,R}$, our method will work in a similar way by simply replacing the previous expression (8) by

$$E = \sum_{r=1}^{R}\left(\mathbf{Q}_r - \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u_r)\,\Psi_j^n(v_r)\right)^2. \tag{9}$$

The least-squares minimization of either (8) or (9) leads to the system of equations:

$$\langle\mathbf{Q}\rangle = \langle\mathbf{P}\rangle\cdot\boldsymbol{\Xi}, \tag{10}$$

where $\langle\mathbf{Q}\rangle$ corresponds to the vectorization of the set of data points $\{\mathbf{Q}_{k,l}\}_{k=1,\dots,p;l=1,\dots,q}$ (alternatively, $\{\mathbf{Q}_r\}_{r=1,\dots,R}$), $\langle\mathbf{P}\rangle$ corresponds to the vectorization of the set of control points $\{\mathbf{P}_{i,j}\}_{i=0,\dots,m;j=0,\dots,n}$, and $\boldsymbol{\Xi}$ is a matrix given by $\boldsymbol{\Xi}_{i,j} = \boldsymbol{\Psi}^n(v_j)\odot\Psi_0^m(\mathbf{u})$, with $\boldsymbol{\Psi}^d(\omega_k) = (\Psi_0^d(\omega_k),\dots,\Psi_D^d(\omega_k))$, $\Psi_k^d(\boldsymbol{\Theta}) = (\Psi_k^d(\theta_1),\dots,\Psi_k^d(\theta_K))$, for any $\boldsymbol{\Theta} = (\theta_1,\dots,\theta_K)$, and $\odot$ represents the tensor product of vectors. The indices in (10) vary in the ranges of values indicated throughout the section.

The algebraic solution of (10) is given by, $\mathbf{P} = \boldsymbol{\Xi}^+\cdot\mathbf{Q}$, where $\boldsymbol{\Xi}^+$ denotes the Moore-Penrose pseudoinverse of $\boldsymbol{\Xi}$. Due to the fact that the blending functions are nonlinear in u and v, the least-squares minimization of the errors is a strongly nonlinear problem, with a large number of unknowns for large sets of data points. Our strategy for solving the problem consists of applying the firefly algorithm to determine suitable parameter values for the least-squares minimization of functional E according to either (8) or (9). However, in order to do it, some previous steps must be carefully carried out.

(1) First of all, we need an adequate representation of the unknowns of the problem. Because of the tensor-product structure of the free-form Bézier surfaces, the fireflies in our method can be encoded as either strings of two sorted real-coded vectors on the interval $[0,1]$ of length p and q, respectively, for organized data points, or as sorted real-coded vectors of length R for the case of irregularly sampled data points. All fireflies are initialized with sorted uniformly distributed random numbers on the coordinate parametric domain.

(2) The objective function corresponds to the evaluation of the least-squares function given by either (8) or (9). Since this error function does not consider the number of data points, we also compute the RMSE (root-mean squared error), given by

$$\text{RMSE} = \sqrt{\frac{\sum_{k=1}^{p}\sum_{l=1}^{q}\left(\mathbf{Q}_{k,l} - \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u_k)\,\Psi_j^n(v_l)\right)^2}{p\cdot q}}, \tag{11}$$

for (8) or, alternatively by:

$$\text{RMSE} = \sqrt{\frac{\sum_{r=1}^{R}\left(\mathbf{Q}_r - \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u_r)\,\Psi_j^n(v_r)\right)^2}{R}}, \tag{12}$$

for (9) and report our results by using these error criteria.

(3) We also need to choose the degree of the approximating surface, which in turn depends on the number of control points. This value is chosen according to the complexity of the shape of the underlying function of data. In general, a small amount of control points is needed for simple, smooth shapes, while

a large number of control points must be selected for complicated, twisted, or irregular shapes. Since this number is unknown a priori, it is advisable to start with a low number of control points for each parametric coordinate and increase it until the error reaches values below a prescribed threshold, which generally depends on both the underlying surface and the application domain.

(4) Regarding the firefly algorithm, some control parameters should be set up. As usual when working with metaheuristic techniques, the choice of suitable control parameters is very important as it determines the performance of the method at large extent. It is also challenging, because it is strongly problem dependent. In this paper, our choice is based on a large collection of empirical results. These control parameters are

(a) the number of fireflies, n_f: this value is set up to $n_f = 100$ fireflies in all examples of this paper. We also tried larger populations of fireflies (up to 1000 individuals) but found that our results do not change significantly. Since larger populations mean larger computation times with no remarkable improvement at all, we found this value to be appropriate in our simulations;

(b) the number of iterations, n_{iter}: this number is another parameter of the method that has to be determined in order to run the algorithm until the convergence of the minimization of the error is achieved. In general, the firefly algorithm does not need a large number of iterations to reach the global optima. This also happens in this case. In all our simulations, we found that $n_{\text{iter}} = 10$ is a suitable value, as larger values for this parameter does not improve our results;

(c) the initial attractiveness, β_0: some theoretical results suggest that $\beta_0 = 1$ is a good choice for many optimization problems. We also take this value in this paper, with very good results, as it will be discussed in next section;

(d) the absorption coefficient, γ: it is set up to $\gamma = 0.5$ in this paper, as this value provides a quick convergence of the algorithm to the optimal solution;

(e) the potential coefficient, μ: although any positive value can be used for this parameter, the light intensity varies according to the inverse square law. Therefore, we choose $\mu = 2$ accordingly;

(f) the randomization parameter, α. This parameter varies on the interval $[0, 1]$ and allows us to determine the degree of randomization introduced in the algorithm. This stochastic component is necessary in order to allow new solutions appear and avoid getting stuck in a local minimum. However, larger values introduce large perturbations on the evolution of the firefly and, therefore, delay convergence to the global optima. Consequently, it is advisable to select values in between. In this work, we take $\alpha = 0.5$.

After the selection of those parameters, the firefly algorithm is performed iteratively for the given number of iterations. To remove the stochastic effects and avoid premature convergence, 20 independent executions have been carried out for each choice of the surface degree. Then, the firefly with the best (i.e., minimum) fitness value is selected as the best solution to the problem.

5. Experimental Results

To check the performance of our method described previously, it has been tested with a large collection of examples with excellent results in all cases. To keep the paper at manageable size, in this section we consider only three of them. They have been primarily chosen to reflect the diversity of situations to which the method can be applied. The examples correspond to both open and closed surfaces, including shapes with singularities. As the reader will see, they clearly show the good performance of our approach.

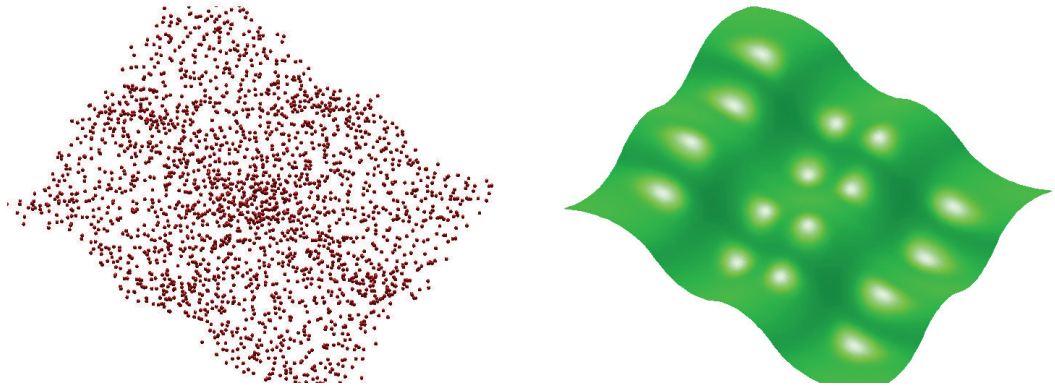
Examples in this paper are shown in Figures 1, 2, and 3. For each example, two different pictures are displayed: on the left, we show the original cloud of input data points, represented as small red points; on the right, the best approximating Bézier surface, as obtained with our firefly-based method, is displayed. Our input consists of sets of irregularly sampled data points (this fact can readily be seen from simple visual inspection of the point clouds on the left), which are also affected by measurement noise of low to medium intensity (signal-to-noise ratio of 15 : 1, 25 : 1, and 10 : 1, resp.). In all examples, no information about the data points parameterization is available at all. In fact, no information about the structure and properties of the underlying surface of data is either assumed or known beyond the data points.

Table 1 summarizes the main results of our computer simulations. The different examples are arranged in rows. For each example, the following data are arranged in columns: number of data points, E error value (according to (8) and (9)), the maximum of the E error (denoted by Max E and that provides a useful upper bound for that error), and RMSE error value (according to (11) and (12)). The error values are reported for each coordinate in all cases.

First observation is that, although our data points are irregularly sampled and affected by noise, the method yields very good fitting results in all cases. The RMSE is of order 10^{-3} in all cases, while the order of the least-squares E error is within the range 10^{-3}–10^{-2} and so is its maximum. Furthermore, these very small fitting errors are obtained for surfaces that are more complicated than it may seem at first sight. For instance, the surfaces of the first and third examples are apparently simple, flat, and height-map surfaces. However, a careful observation reveals that they oscillate several times, and hence, they exhibit a rich variety of hills and valleys, which have been highlighted by using an illumination model for the sake of clarity. On the other hand,

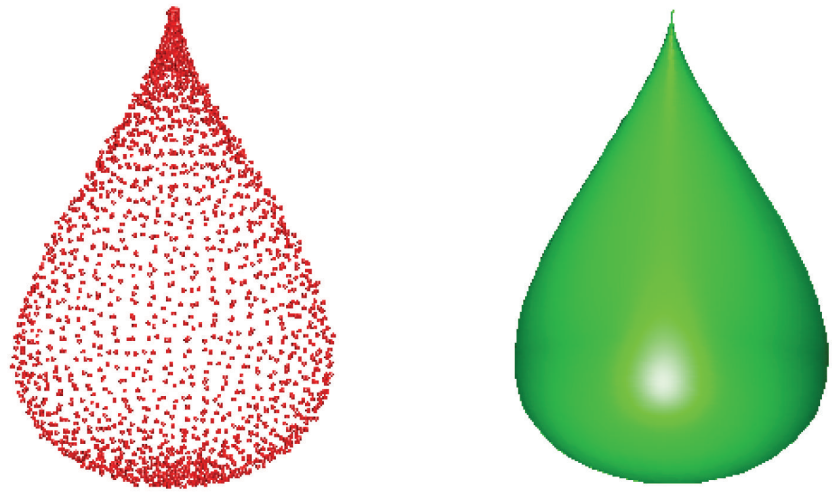
Table 1: Number of data points and error values (for each coordinate) of the three examples discussed in this paper.

Example	Number of data points	Error (E)	Error (Max E)	Error (RMSE)
Example 1	6572	x: 7.3652×10^{-2}	x: 9.5144×10^{-2}	x: 3.3476×10^{-3}
		y: 7.4303×10^{-2}	y: 9.8452×10^{-2}	y: 3.3624×10^{-3}
		z: 7.5126×10^{-2}	z: 9.9673×10^{-2}	z: 3.3811×10^{-3}
Example 2	3378	x: 5.2958×10^{-3}	x: 7.2446×10^{-3}	x: 1.2521×10^{-3}
		y: 5.1216×10^{-3}	y: 7.0237×10^{-3}	y: 1.2313×10^{-3}
		z: 5.2909×10^{-3}	z: 7.4532×10^{-3}	z: 1.2515×10^{-3}
Example 3	7312	x: 6.4191×10^{-2}	x: 8.4377×10^{-2}	x: 2.9629×10^{-3}
		y: 6.3774×10^{-2}	y: 8.3875×10^{-2}	y: 2.9532×10^{-3}
		z: 6.4746×10^{-2}	z: 9.3271×10^{-2}	z: 2.9756×10^{-3}



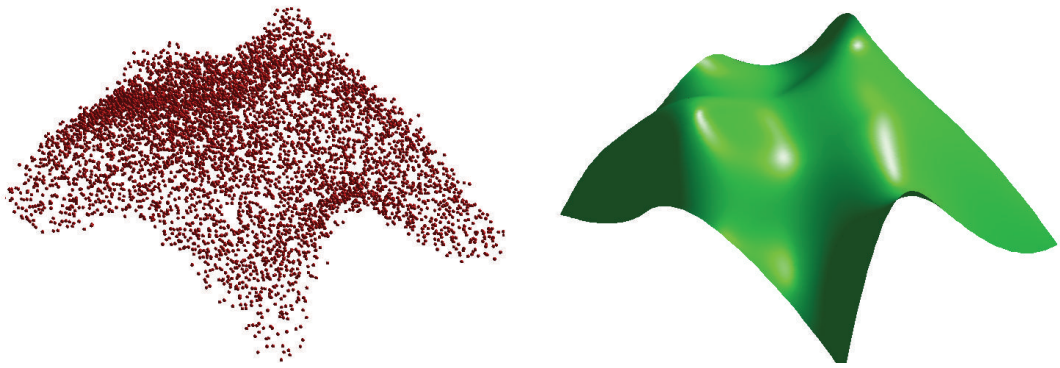
(a) (b)

Figure 1: Applying the firefly algorithm to Bézier surface approximation of data points: (a) original data points; (b) best approximating Bézier surface.



(a) (b)

Figure 2: Applying the firefly algorithm to Bézier surface approximation of data points: (a) original data points; (b) best approximating Bézier surface.



(a) (b)

Figure 3: Applying the firefly algorithm to Bézier surface approximation of data points: (a) original data points; (b) best approximating Bézier surface.

the second surface is a closed surface with a strong singularity at its uppermost part, where many data points concentrate in a very small volume. This is usually a very challenging feature for free-form parametric surfaces, which typically tend to distribute the control points by following a rectangular topology. Clearly, such a distribution is not adequate for this surface. To our delight, the proposed method identifies this situation automatically and rearranges the control points by itself to adapt to the underlying structure of data points. In opinion of the authors, this is a striking and very remarkable feature of this method and shows its ability to capture the real behavior of data points even under unfavorable conditions.

To summarize, a visual inspection of the three figures clearly shows that our method yields a very good approximating surface to data points in all cases. This fact is validated by the numerical results reported in Table 1, which confirm the good behavior of the method. From these examples and many other not reported here for the sake of brevity, we conclude that the presented method performs very well, with remarkable capability to provide a satisfactory, accurate solution to our parameterization problem with polynomial Bézier surfaces.

Regarding the implementation issues, all computations in this paper have been performed on a 2.9 GHz. Intel Core i7 processor with 8 GB of RAM. The source code has been implemented by the authors in the native programming language of the popular scientific program *Matlab*, version 2010b for Windows 8 operating system.

6. Conclusions and Future Work

This paper introduces a new method to address the surface parameterization problem, that is, to compute a suitable parameterization of a set of data points in order to construct the free-form parametric surface approximating such data points better in the least-squares sense. This is a challenging problem that appears recurrently in reverse engineering for computer design and manufacturing and in many other industrial fields. Very often, data points in real-world settings are irregularly sampled and subjected to measurement noise, leading to a very difficult nonlinear continuous optimization problem, which cannot be solved by using standard optimization techniques. To overcome this limitation, this paper proposes a new method based on a powerful nature-inspired metaheuristic algorithm called firefly algorithm, introduced recently to solve difficult optimization problems. The method has been successfully applied to solve the parameterization problem for polynomial Bézier surfaces. The paper discusses the main issues in this problem, such as the solution representation and the selection of suitable control parameters. To check the performance of our approach, it has been applied to some illustrative examples of open and closed surfaces, including shapes with singularities. Our results show that the method performs very well, being able to yield the best approximating surface with a high degree of accuracy.

As mentioned in Section 3, the original firefly algorithm has been improved and modified in many different ways. Some of its variants have shown to be more efficient than the original version, meaning that the presented approach can arguably be improved with new, optimized features for better performance. An illustrative example is given by a very recent version called memetic self-adaptive firefly algorithm [53], whose new capabilities (the use of self-adaptation strategies on the control parameters, a new population model based on elitism, and the hybridization with a local search heuristics) improve the original firefly algorithm significantly. The application of many of these variants to our parameterization problem along with a comparative analysis of their performance is part of our future work. We are also interested to extend this method to other families of surfaces, such as the B-splines and NURBS, where the existence of additional parameters (such as knots and weights) can modify our procedure significantly. The application of this method to some interesting real-world problems in industrial settings is also part of our plans for future work.

Conflict of Interests

The authors of this paper have no current or past, direct or indirect financial relationship with any commercial identity

mentioned in this paper that might lead to any conflict of interests. They are solely mentioned here for scientific purposes.

Acknowledgments

This research has been kindly supported by the Computer Science National Program of the Spanish Ministry of Economy and Competitiveness, Project Reference no. TIN2012-30768, Toho University (Funabashi, Japan), and the University of Cantabria (Santander, Spain). The authors are particularly grateful to the Department of Information Science of Toho University for all the facilities given to carry out this research work. Special thanks are due to the Editor and the anonymous reviewers for their useful comments and suggestions that allowed us to improve the final version of this paper.

References

[1] R. E. Barnhill, *Geometry Processing for Design and Manufacturing*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 1992.

[2] G. Echevarra, A. Iglesias, and A. Galvez, "Extending neural networks for *B*-spline surface reconstruction," in *Computational Science—ICCS 2002*, vol. 2330 of *Lecture Notes in Computer Science*, pp. 305–314, 2002.

[3] A. Gálvez, A. Iglesias, and J. Puig-Pey, "Iterative two-step genetic-algorithm-based method for efficient polynomial *B*-spline surface reconstruction," *Information Sciences*, vol. 182, pp. 56–76, 2012.

[4] A. Gálvez and A. Iglesias, "Particle swarm optimization for non-uniform rational *B*-spline surface reconstruction from clouds of 3D data points," *Information Sciences*, vol. 192, pp. 174–192, 2012.

[5] A. Iglesias and A. Galvez, "A new artificial intelligence paradigm for computer aided geometric design," in *Artificial Intelligence and Symbolic Computation*, vol. 1930, pp. 200–213, Lecture Notes in Computer Science, 2001.

[6] A. Iglesias, G. Echevarría, and A. Gálvez, "Functional networks for *B*-spline surface reconstruction," *Future Generation Computer Systems*, vol. 20, no. 8, pp. 1337–1353, 2004.

[7] H. Pottmann, S. Leopoldseder, M. Hofer, T. Steiner, and W. Wang, "Industrial geometry: recent advances and applications in CAD," *Computer Aided Design*, vol. 37, no. 7, pp. 751–766, 2005.

[8] T. Varady and R. Martin, "Reverse engineering," in *Handbook of Computer Aided Geometric Design*, pp. 651–681, North-Holland, Amsterdam, The Netherlands, 2002.

[9] N. M. Patrikalakis and T. Maekawa, *Shape Interrogation for Computer Aided Design and Manufacturing*, Springer, Berlin, Germany, 2002.

[10] A. Gálvez and A. Iglesias, "Efficient particle swarm optimization approach for data fitting with free knot *B*-splines," *Computer Aided Design*, vol. 43, no. 12, pp. 1683–1692, 2011.

[11] A. Galvez and A. Iglesias, "A new iterative mutually-coupled hybrid GA-PSO approach for curve fitting in manufacturing," *Applied Soft Computing*, vol. 13, no. 3, pp. 1491–1504, 2013.

[12] J. Ling and S. Li, "Fitting *B*-spline curves by least squares support vector machines," in *Proceedings of the International Conference on Neural Networks and Brain Proceedings (ICNNB '05)*, pp. 905–909, Beijing, China, October 2005.

[13] D. L. B. Jupp, "Approximation to data by splines with free knots," *SIAM Journal on Numerical Analysis*, vol. 15, no. 2, pp. 328–343, 1978.

[14] T. C. M. Lee, "On algorithms for ordinary least squares regression spline fitting: a comparative study," *Journal of Statistical Computation and Simulation*, vol. 72, no. 8, pp. 647–663, 2002.

[15] W. Li, S. Xu, G. Zhao, and L. P. Goh, "Adaptive knot placement in *B*-spline curve approximation," *Computer Aided Design*, vol. 37, no. 8, pp. 791–797, 2005.

[16] H. Park, "An error-bounded approximate method for representing planar curves in *B*-splines," *Computer Aided Geometric Design*, vol. 21, no. 5, pp. 479–497, 2004.

[17] H. Park and J. Lee, "*B*-spline curve fitting based on adaptive curve refinement using dominant points," *Computer Aided Design*, vol. 39, no. 6, pp. 439–451, 2007.

[18] G. Farin, *Curves and Surfaces for CAGD*, Morgan Kaufmann, San Francisco, Calif, USA, 5th edition, 2002.

[19] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A K Peters, Wellesley, Mass, USA, 1993.

[20] L. Piegl and W. Tiller, *The NURBS Book*, Springer, Berlin, Germany, 1997.

[21] D. F. Rogers, *An Introduction to NURBS: With His Historical Perspective*, Morgan Kaufmann, 2000.

[22] G. E. Hölzle, "Knot placement for piecewise polynomial approximation of curves," *Computer-Aided Design*, vol. 15, no. 5, pp. 295–296, 1983.

[23] W. Ma and J. Kruth, "Parameterization of randomly measured points for least squares fitting of *B*-spline curves and surfaces," *Computer-Aided Design*, vol. 27, no. 9, pp. 663–675, 1995.

[24] L. A. Piegl and W. Tiller, "Least-squares *B*-spline curve approximation with arbitrary end derivatives," *Engineering with Computers*, vol. 16, no. 2, pp. 109–116, 2000.

[25] T. Várady, R. R. Martin, and J. Cox, "Reverse engineering of geometric models—an introduction," *Computer Aided Design*, vol. 29, no. 4, pp. 255–268, 1997.

[26] W. P. Wang, H. Pottmann, and Y. Liu, "Fitting *B*-spline curves to point clouds by curvaturebased squared distance minimization," *ACM Transactions on Graphics*, vol. 25, no. 2, pp. 214–238, 2006.

[27] F. Yoshimoto, M. Moriyama, and T. Harada, "Automatic knot adjustment by a genetic algorithm for data fitting with a spline," in *Proceedings of the International Conference on Shape Modeling International and Applications*, pp. 162–169, IEEE Computer Society Press, 1999.

[28] F. Yoshimoto, T. Harada, and Y. Yoshimoto, "Data fitting with a spline using a real-coded genetic algorithm," *Computer Aided Design*, vol. 35, no. 8, pp. 751–760, 2003.

[29] J. Barhak and A. Fischer, "Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 1, pp. 1–16, 2001.

[30] M. Alhanaty and M. Bercovier, "Curve and surface fitting and design by optimal control methods," *Computer Aided Design*, vol. 33, no. 2, pp. 167–182, 2001.

[31] P. Dierckx, *Curve and Surface Fitting with Splines*, Oxford University Press, Oxford, Miss, USA, 1993.

[32] T. Lyche and K. Mørken, "Knot removal for parametric *B*-spline curves and surfaces," *Computer Aided Geometric Design*, vol. 4, no. 3, pp. 217–230, 1987.

[33] M. J. D. Powell, "Curve fitting by splines in one variable," in *Numerical Approximation to Functions and Data*, J. G. Hayes, Ed., Athlone Press, London, UK, 1970.

[34] J. R. Rice, *Numerical Methods, Software and Analysis*, Academic Press, New York, NY, USA, 2nd edition, 1993.

[35] H. Yang, W. Wang, and J. Sun, "Control point adjustment for *B*-spline curve approximation," *Computer Aided Design*, vol. 36, no. 7, pp. 639–652, 2004.

[36] E. Castillo and A. Iglesias, "Some characterizations of families of surfaces using functional equations," *ACM Transactions on Graphics*, vol. 16, no. 3, pp. 296–318, 1997.

[37] A. Gálvez, J. Puig-Pey, and A. Iglesias, "A differential method for parametric surface intersection," in *Computational science and its applications—ICCSA 2004*, vol. 3044 of *Lecture Notes in Computer Science*, pp. 651–660, Springer, Berlin, Germany, 2004.

[38] P. Gu and X. Yan, "Neural network approach to the reconstruction of freeform surfaces for reverse engineering," *Computer-Aided Design*, vol. 27, no. 1, pp. 59–64, 1995.

[39] M. Hoffmann, "Numerical control of Kohonen neural network for scattered data approximation," *Numerical Algorithms*, vol. 39, no. 1–3, pp. 175–186, 2005.

[40] G. K. Knopf and J. Kofman, "Free-form surface reconstruction using Bernstein basis function networks," in *Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE '99)*, vol. 9, pp. 797–802, ASME Press, November 1999.

[41] A. Iglesias and A. Galvez, "Applying functional networks to fit data points from *B*-spline surfaces," in *Proceedings of the Computer Graphics International (CGI '01)*, pp. 329–332, IEEE Computer Society Press, Hong Kong, China, 2001.

[42] A. Galvez, A. Iglesias, A. Cobo, J. Puig-Pey, and J. Espinola, "Bézier curve and surface fitting of 3D point clouds through genetic algorithms, functional networks and least-squares approximation," in *Computational Science and Its Applications—ICCSA 2007*, vol. 4706 of *Lecture Notes in Computer Science*, pp. 680–693, 2007.

[43] M. Sarfraz and S. A. Raza, "Capturing outline of fonts using genetic algorithms and splines," in *Proceedings of the 5th International Conference on Information Visualization (IV '01)*, pp. 738–743, IEEE Computer Society Press, 2001.

[44] A. Galvez, A. Iglesias, and A. Avila, "Immunological-based approach for accurate fitting of 3D noisy data points with Bezier surfaces," in *Proceedings of the International Conference on Computational Science (ICCS '13)*, vol. 18, pp. 50–59, Procedia Computer Science, 2013.

[45] E. Ülker and A. Arslan, "Automatic knot adjustment using an artificial immune system for *B*-spline curve approximation," *Information Sciences*, vol. 179, no. 10, pp. 1483–1494, 2009.

[46] X. Zhao, C. Zhang, B. Yang, and P. Li, "Adaptive knot placement using a GMM-based continuous optimization algorithm in *B*-spline curve approximation," *Computer Aided Design*, vol. 43, no. 6, pp. 598–604, 2011.

[47] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, vol. 5792 of *Lectures Notes in Computer Science*, pp. 169–178, Springer, Berlin, Germany, 2009.

[48] X. S. Yang, "Firey algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[49] S. L. Tilahun and H. C. Ong, "Modified firefly algorithm," *Journal of Applied Mathematics*, vol. 2012, Article ID 467631, 12 pages, 2012.

[50] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, Frome, UK, 2nd edition, 2010.

[51] X.-S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, Wiley & Sons, New Jersey, NJ, USA, 2010.

[52] I. Fister, I. Fister Jr., X. S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*. In press.

[53] I. Fister, X. S. Yang, J. Brest, and I. Fister Jr., "Memetic self-adaptive firefly algorithm," in *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, X. S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu, Eds., pp. 73–102, Elsevier, 2013.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 718015, 11 pages
http://dx.doi.org/10.1155/2013/718015

Research Article

Determining Optimal Link Capacity Expansions in Road Networks Using Cuckoo Search Algorithm with Lévy Flights

Ozgur Baskan

Department of Civil Engineering, Faculty of Engineering, Pamukkale University, 20070 Denizli, Turkey

Correspondence should be addressed to Ozgur Baskan; obaskan@pau.edu.tr

Received 28 June 2013; Accepted 31 July 2013

Academic Editor: Xin-She Yang

During the last two decades, Continuous Network Design Problem (CNDP) has received much more attention because of increasing trend of traffic congestion in road networks. In the CNDP, the problem is to find optimal link capacity expansions by minimizing the sum of total travel time and investment cost of capacity expansions in a road network. Considering both increasing traffic congestion and limited budgets of local authorities, the CNDP deserves to receive more attention in order to use available budget economically and to mitigate traffic congestion. The CNDP can generally be formulated as bilevel programming model in which the upper level deals with finding optimal link capacity expansions, whereas at the lower level, User Equilibrium (UE) link flows are determined by Wardrop's first principle. In this paper, cuckoo search (CS) algorithm with Lévy flights is introduced for finding optimal link capacity expansions because of its recent successful applications in solving such complex problems. CS is applied to the 16-link and Sioux Falls networks and compared with available methods in the literature. Results show the potential of CS for finding optimal or near optimal link capacity expansions in a given road network.

1. Introduction

The Continuous Network Design Problem (CNDP) deals with finding optimal capacity expansions for a set of selected links and the corresponding equilibrium link flows in a given road network. This well-known transportation problem has been studied for many years to improve the performance of transportation road networks and thus mitigate traffic congestion. Since the multiple objectives exist in the CNDP, it is considerably typical to formulate it as a bilevel model, which is difficult to solve. Due to the nonconvexity of the bilevel solution of the CNDP, it can be recognized as one of the most challenging problems in transportation field. The difficulty of the bilevel modeling arises from the evaluation of the upper level objective function that involves solving the lower level problem for each set of upper level decision variables. In the CNDP, upper level problem can be formulated by minimizing the sum of total travel time and total investment cost of link capacity expansions in a given road network, whilst the lower level can be solved as User Equilibrium (UE) traffic assignment model considering Wardrop's first principle [1]. In the CNDP, optimal capacity expansion plan at the upper level cannot be found without considering the reactions of the road users to that plan at the lower level. Although the upper and lower level problems consist of convex problems, the bilevel solution of the CNDP may be nonconvex due to both traffic assignment constraints and nonlinear travel time function. This nonconvexity may cause serious problems for deterministic algorithms [2].

The first continuous network design formulation was proposed by Abdulaal and LeBlanc [3]. They have formulated the network design problem with continuous variables subject to equilibrium assignment as a nonlinear optimization problem. Hooke-Jeeves (HJ) and Powell's methods were used in order to test the proposed model on a medium-sized network. In their study, the effect of type of investment function was also investigated. It has been found that the performance of two methods is approximately same for convex investment cost function, whilst the HJ is better than the Powell's method in the case of using concave investment function. After this first study, several variations of the CNDP have been studied and various solution methods have been proposed for solving this problem. Suwansirikul et al. [4] proposed Equilibrium Decomposition Optimization (EDO) method for finding an

approximate solution to the CNDP and tested this heuristic on several networks. Their results showed that the proposed heuristic is more efficient than the HJ algorithm. The efficiency of the method stems from decomposition of the original problem into a set of suboptimization problems. The other advantage of the EDO algorithm was reported that the computational cost of the proposed model does not depend on the number of links which are considered for capacity expansion. Marcotte [5] and Marcotte and Marquis [6] presented efficient implementations of heuristic methods in small-sized networks for solving the CNDP where road users follow Wardrop's first principle. In addition, a number of sensitivity-based heuristic algorithms were developed for the CNDP [7–10]. Furthermore, Friesz et al. [11] used Simulated Annealing (SA) approach to solve the CNDP for two different road networks and found that the proposed heuristic is more efficient than Iterative Optimization Assignment (IOA), HJ, and EDO algorithms. Afterwards, Friesz et al. [12] presented a model for continuous multiobjective optimal design of a transportation road network. Results showed that SA is ideally suited for solving multiobjective versions of the equilibrium network design problem. Unlike using classical lower level solution as in most studies, Stochastic User Equilibrium (SUE) assignment procedure was embedded to the CNDP in Davis [13]. The generalized reduced gradient and sequential quadratic programming methods were used to solve the CNDP. The proposed solution methods were tested on several example networks, and it has been found that the differentiable and tractable version of the CNDP could be created. In order to avoid the disadvantages for the use of bilevel formulation, Meng et al. [14] formulated the CNDP as a single level continuously differentiable optimization problem and applied the augmented Lagrangian method to solve this problem. Their results showed that the proposed method has the potential for application to large-scale problems. Chiou [15] used a bilevel programming model to solve the CNDP. Four variants of gradient-based methods are presented, and numerical comparisons are made with several test networks. Results showed that the proposed methods are more effective than the other compared algorithms when especially congested road networks are considered.

Similarly, Ban et al. [16] proposed a relaxation method to solve the CNDP when the lower level is a nonlinear complementary problem. They converted original bilevel model into a single level formulation by means of adding some constraints to the lower level problem, and a relaxation scheme was proposed to solve it. The proposed solution algorithm was tested on different test networks, and promising results were obtained. Karoonsoontawong and Waller [17] presented SA, Genetic Algorithm (GA), and random search techniques to solve the CNDP. Their study showed that GA is performed better than the others on the test networks in terms of solution quality and convergence. Moreover, they emphasized that the algorithm parameters should be calibrated to achieve best results for different road networks. Gao et al. [18] converted the bilevel solution of the CNDP into a single level convex model and proposed a globally convergent algorithm to solve this problem. They presented a numerical example to show the effectiveness of the proposed method against the other existing heuristic algorithms. Xu et al. [19] proposed SA and GA methods to achieve the optimal solutions of the CNDP. They tested the proposed methods on small-sized network for three demand scenarios and found that when demand is large, SA is more efficient than GA in solving the CNDP. In addition, much more computational time is needed for GA in comparison with SA in order to achieve same optimal solution. Unlike the study proposed by Xu et al. [19], Mathew and Sharma [20] reported that GA model is more efficient than other compared algorithms available in the literature for solving the CNDP. They applied the proposed model to three different road networks, one of which is considered as a real city network, and found that the GA is capable of finding high quality solution especially for large scale road network. Wang and Lo [21] formulated the CNDP as a single level optimization problem subject to equilibrium constraints. In order to overcome the nonconvexity of the CNDP, they transformed the equilibrium constraints into a set of mixed-integer constraints and linearized the travel time function. Their results showed that the proposed method is capable to find globally optimal solution of the CNDP. Li et al. [22] presented an applicable global optimization method for solving the CNDP and converted the CNDP into a sequence of single level concave programs. Their method has the potential to find global optimum of large network design problems. Baskan and Dell'Orco [23] applied artificial bee colony optimization algorithm to solve the CNDP. The proposed method is compared with SA and GA algorithms for small-sized road network and obtained good results in comparison with other methods in terms of objective function value and number of UE assignments. In addition, Baskan [2] and Baskan and Ceylan [24] attempted to solve the bilevel formulation of the CNDP using Harmony Search and Differential Evolution algorithms, respectively.

From the viewpoint of reserve capacity, Yang and Wang [25] compared the level of equivalence and effectiveness of two different objective functions for the CNDP, which are minimizing the total system cost under a budget constraint and maximization of network reserve capacity. Numerical results showed that a combined objective function by applying different weightings on two objectives can also be more effective. Following the study made by [25], Ziyou and Yifan [26] combined the concept of reserve capacity with the CNDP. A bilevel programming model and heuristic solution algorithm based on sensitivity analysis are proposed to solve the reserve capacity problem of optimal signal control with user equilibrium route choice. They concluded that it is crucial importance to combine the concept of reserve capacity with the CNDP in order to provide more realistic information for transportation planners. Similarly, Chiou [27] proposed a projected Quasi-Newton method for simultaneously solving the problem of finding the maximum possible increase in travel demand and determining optimal link capacity expansions. Numerical applications are made on signal-controlled networks where obtained results outperform than traditional methods.

The reviewed literature shows that heuristic methods play an important role for solving the various types of the CNDP. Therefore, the performance of recently developed

heuristic algorithms needs to be investigated in order to obtain probably better solutions to the CNDP. On the other hand, although some exact algorithms for the CNDP are available in the literature, they may not be suitable especially for large scale networks. Therefore, this paper deals with finding optimal link capacity expansions in a given road network using cuckoo search (CS) algorithm with Lévy flights. For this purpose, a bilevel model has been proposed in which the lower level problem is formulated as UE traffic assignment mode,l and Frank-Wolfe (FW) method is used to solve this problem.

The remainder of this paper is organized as follows. In Section 2, problem formulation for the CNDP is given. In the next section, the CS algorithm and its solution procedure on the CNDP are presented. In Section 4, numerical studies are conducted on two different test networks. Finally, concluding remarks and future study directions are given in Section 5.

2. Problem Formulation

The following notation is used for the problem formulation:

A : the set of links, $\forall a \in A$,

C_{rs}: the set of paths between O-D pair rs $\forall r \in R$, $s \in S$,

R: the set of origins,

S: the set of destinations,

D: the vector of O-D pair demands, $D = [D_{rs}]$ $\forall r \in R$, $s \in S$,

f: the vector of path flows, $f = [f_c^{rs}]$, $\forall r \in R$, $s \in S$, $c \in C_{rs}$,

g: the vector of investment costs, $g = [g_a(y_a)]$ $\forall a \in A$,

L: the step length,

t: the vector of link travel times, $t = [t_a(x_a, y_a)]$ $\forall a \in A$

h: the vector of upper bound for link capacity expansions, $h = [h_a]$, $\forall a \in A$,

x: the vector of equilibrium link flows, $x = [x_a]$, $\forall a \in A$,

y: the vector of link capacity expansions, $y = [y_a]$, $\forall a \in A$,

$\mathbf{P}$: the probability matrix,

$\mathbf{K}$: the matrix of local step size,

d_a: the cost coefficient, $\forall a \in A$,

n: the number of nests,

p: the discovering parameter,

Z: upper level objective function,

z: lower level objective function,

θ_a: the link capacity, $\forall a \in A$,

ρ: the conversion factor from investment cost to travel times,

$\delta_{a,c}^{rs}$: the link/path incidence matrix variables, $\forall r \in R, s \in S, c \in C_{rs}, a \in A$. $\delta_{a,c}^{rs} = 1$ if route c between O-D pair rs uses link a, and $\delta_{a,c}^{rs} = 0$ otherwise,

λ_a, φ_a: the parameters of link cost function, $\forall a \in A$,

u, v: the parameters of the step length L,

β: the scale parameter,

Γ: the gamma function,

α: the step size.

2.1. Upper Level Formulation. In the CNDP, the upper level deals with finding optimal capacity expansion plan for a set of selected links in a given road network by minimizing the total system cost and construction cost, while the lower level determines UE link flows considering given capacity expansion plan in the upper level. Therefore, the CNDP is recognized within the framework of a leader-follower, where the transportation planner is the leader and the user is the follower [28]. It is assumed that the leader as transportation planning manager can influence the users' route choice behavior but cannot be able to control it. The users make their route choice decision by minimizing their own travel costs under given service level of transportation road networks [18]. This interaction can be formulated as follows:

$$\begin{aligned} \min_{y} \quad & Z(x, y) = \sum_{a \in A} \left(t_a(x_a, y_a) x_a + \rho g_a(y_a) \right) \\ \text{s.t.} \quad & 0 \le y_a \le h_a, \quad \forall a \in A \\ & x = x(y), \end{aligned} \tag{1}$$

where $x(y)$ is UE link flow under given capacity expansion plan and obtained by solving the lower level problem. The constraint of (1) ensures that the investment cost of link $a \forall a \in A$ will not exceed the related budget. It is also the nonnegativity constraint of the upper level decision variables.

2.2. Lower Level Formulation. In the CNDP models, the user's route choice behavior is generally characterized by the UE assignment that is to find the equilibrium link flows. In this paper, Wardrop's first principle is followed, which states that the travel times of all used paths between the same Origin-Destination (O-D) pair are equal and less than any unused paths [21]. It is well-known that the increase in capacity of any link in a given road network without considering the response of network users may actually increase traffic congestion rather than reducing it. Due to well-known Braess' paradox, prediction of traffic flows is crucial importance to the CNDP. The UE assignment problem can be formulated as follows:

$$\begin{aligned} \min_{x} \quad & z = \sum_{a \in A} \int_0^{x_a} t_a(w, y_a)\, dw \\ \text{s.t.} \quad & \sum_{c \in C} f_c^{rs} = D_{rs}, \quad \forall r \in R, s \in S, c \in C_{rs} \\ & x_a = \sum_{rs} \sum_{c \in C_{rs}} f_c^{rs} \delta_{a,c}^{rs}, \quad \forall r \in R, s \in S, a \in A, c \in C_{rs} \\ & f_c^{rs} \ge 0, \quad \forall r \in R, s \in S, c \in C_{rs}, \end{aligned} \tag{2}$$

where the constraints of (2) are definitional, conservation of the flow constraints, and nonnegativity, respectively. Since UE traffic assignment is a convex optimization problem, it can be numerically solved by various methods. For this purpose, the most widely used solution method is the Frank-Wolfe (FW) algorithm [29]. It has been developed for solving quadratic optimization problems and is also highly convenient for determining equilibrium link flows in road networks [30].

3. Cuckoo Search Algorithm

3.1. Cuckoo Breeding Behavior. The CS is an optimization algorithm proposed by Yang and Deb [31, 32] and recently improved for multiobjective optimization [33]. Before the description of the CS algorithm, it may be helpful to briefly review the fascinating breed behavior of some cuckoo species. The CS is inspired by some cuckoo species by laying their eggs in the nests of host birds of other species. In this case, if a host bird realizes that the eggs are not their own, these alien eggs are either taken away or the nest is abandoned by host bird, and a new nest is built elsewhere. Some cuckoo species such as parasitic cuckoos have evolved in such a way that female cuckoos are very specialized in the mimicry in colours and pattern of the eggs of a few chosen host species. This behavior reduces the probability of their eggs being abandoned and thus increases their hatching probability [34]. Additionally, a cuckoo chick can imitate the call of host chicks to gain access to more feeding opportunities [35]. Furthermore, the timing of egg laying of some cuckoo species is also amazing. Parasitic cuckoos often choose a nest where the host bird has just laid its own eggs. In general, the cuckoo eggs hatch slightly earlier than the host eggs. Once the first cuckoo chick is hatched, the first instinctive action by cuckoo chicks is to evict the host eggs by blindly propelling the eggs out of the nest, which increases the cuckoo chick's share of food provided by the host bird [34]. As summarized previously, the CS algorithm idealizes cuckoo's breeding behavior and thus can be applied to various optimization problems.

3.2. Lévy Flights. As it is well-known, random searching is crucial importance in meta-heuristic algorithms. The Lévy flight is a random process which consists of taking a series of consecutive random steps [36]. From the mathematical point of view, two consecutive steps need to be performed to generate random numbers with Lévy flights: (i) the generation of steps and (ii) the choice of a random direction. To do this, one of the most efficient methods is to use the so-called Mantegna algorithm where the step length L can be determined as follows:

$$L = \frac{u}{|v|^{1/\beta}}, \tag{3}$$

where β is the scale parameter and its recommended range is $[1, 2]$. The β value is set to 1.5 in this study. u and v are obtained from normal distribution as

$$u \sim N\left(0, \sigma_u^2\right), \qquad v \sim N\left(0, \sigma_v^2\right), \tag{4}$$

where σ_u and σ_v are calculated using the following:

$$\sigma_u = \left\{ \frac{\Gamma\left(1+\beta\right)\sin(\pi\beta/2)}{\Gamma\left[(1+\beta)/2\right]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \qquad \sigma_v = 1, \tag{5}$$

where Γ denotes gamma function.

3.3. Cuckoo Search. The CS algorithm is inspired by some species of cuckoos because of their special lifestyle and fascinating breeding behavior [37]. These species tend to lay their eggs in the nests which belong to other host birds. Regarding this parasite behavior of some species of cuckoos, some of host birds either throw out these alien eggs or abandon their nests and build new nests elsewhere.

The following three rules are utilized in the CS algorithm: (i) selection of the best by keeping the best nests or solutions (ii) replacement of host eggs with respect to the quality of the new solutions or cuckoo eggs produced based randomization with Lévy flights and (iii) discovering of some cuckoo eggs by the host birds and replacing according to the quality of the local random walks [38]. The algorithm steps of CS are based on these rules and given in Algorithm 1.

One of the most important steps in the algorithm is the use of Lévy flights for random searching. The Lévy flight is a type of random walk and described by a series of instantaneous jumps chosen from a probability density function which has a power law tail [39, 40]. The step size α, which controlled random search process in Lévy flight, is generally selected between $[0, 1]$ interval. Setting $\alpha = 0.1$ may produce efficient results especially for small-sized optimization problems [40]. The other important parameter in the CS algorithm is p which is used by discovering of cuckoo eggs by the host birds. Besides Yang and Deb [32] emphasized that the convergence rate of the algorithm was not strongly affected by the p value; they suggested setting $p = 0.25$. The flowchart of the CS algorithm for the CNDP is given in Figure 1 and corresponding solution steps can be summarized as given in Figure 1.

Initialize the CS Parameters. The CS parameters, which are number of nests (n), step size (α), discovering probability (p), and maximum number of generations (MNG), are set to 10, 0.1, 0.25, and 1000, respectively. These values are selected in accordance with the recommendation by Yang and Deb [32].

Generation of Initial Population. At generation t, the nests, $\text{nest}^t = [\text{nest}_i^t]$, where $i \in \{1, 2, \ldots, n\}$, are filled with randomly generated capacity expansions for a set of selected links in a given road network by considering upper and lower bounds, and UE link flows are determined for each nest (i.e., set of link capacity expansions) by using (2). After that, their corresponding objective function values are calculated using (1).

Generate New Nests by Lévy Flights. The vector of new nest is generated from randomly selected ith nest by Lévy flights using the following:

$$\text{new\_nest}_i^t = \text{nest}_i^t + \alpha L\left(\text{nest}_i^t - \text{nest}_{\text{best}}^t\right), \tag{6}$$

Generate initial population of n nests $y_i, i = 1, 2, \ldots, n$
for $i = 1 : n$
  Calculate fitness value $F_i = f(y_i)$
end for
while (stopping criterion is not satisfied)
  Generate a cuckoo egg (y_j) by Lévy flights from random nest
  Calculatefitness value $F_j = f(y_j)$
  Choose a random nest i
  if $F_j > F_i$ **then**
  $y_i \leftarrow y_j$
  $F_i \leftarrow F_j$
  end if
  Abandon a fraction p_a of the worst nests
  Build new nests by Lévy flights
  Evaluate fitness of new nests and keep best nest
end while

Algorithm 1: Cuckoo search algorithm.

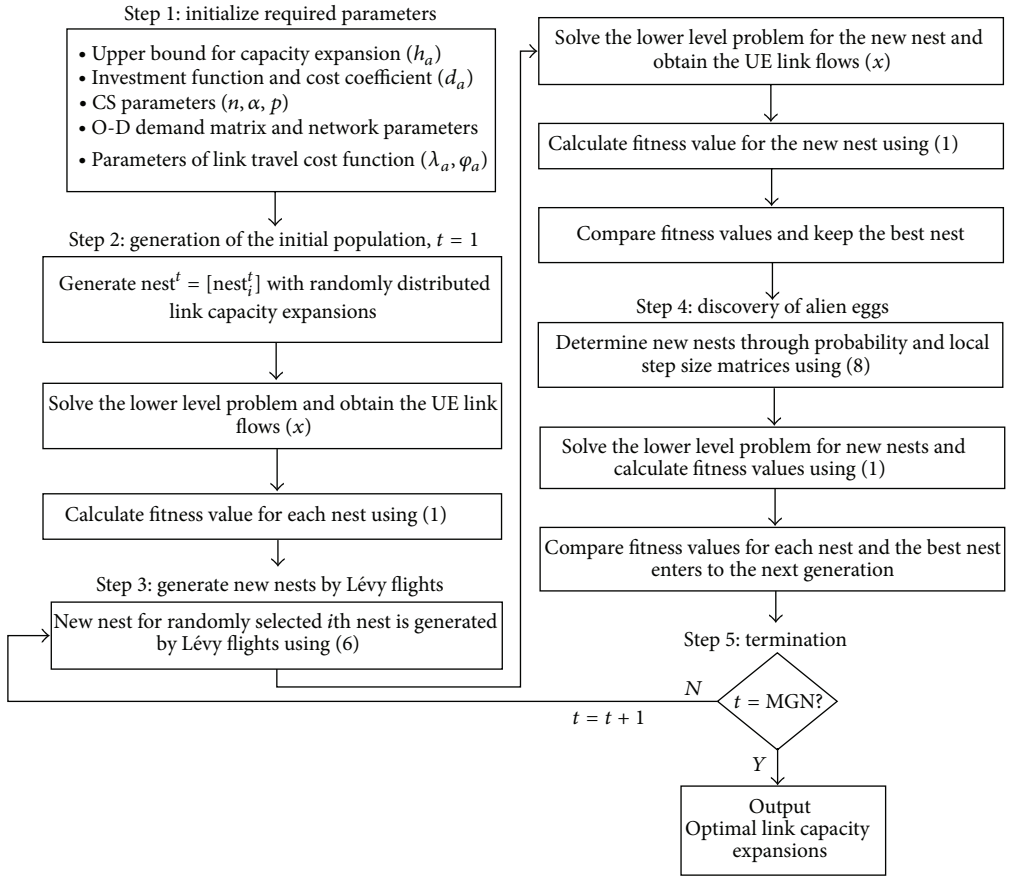


Figure 1: Flowchart of the CS for the CNDP.

where new_nest$_i^t$ is the new nest generated by Lévy flights; nest$_i^t$ is randomly selected nest from population; nest$_{\text{best}}^t$ is the best nest obtained so far; α is step size; and L is the Lévy flights vector or step length as in Mantegna's algorithm. After determining the new nest, the objective function values of two nests are calculated using (1), and the best nest is kept.

Discovery of Alien Eggs. The alien eggs discovery is performed for all of the eggs using the probability matrix. The probability matrix is produced as

$$p_{ij} = \begin{cases} 1, & \text{if rand}(0,1) < p \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

where p_{ij} is discovering probability for the jth variable of the ith nest. The value of p is compared with the output of a uniform random number generator, rand(0, 1), to determine whether local random walk is considered or not. After determining discovering probabilities, new nests are produced using the following:

$$\text{new\_nest}^t = \text{nest}^t + \mathbf{K} * \mathbf{P}, \tag{8}$$

where $\mathbf{P}$ is the probability matrix and $\mathbf{K}$ is the matrix of local step size, which is produced by using the following:

$$\begin{aligned}\mathbf{K} = \text{rand}(\,) * \big(&\text{nests}\left[\text{permute 1}\,[i]\right][j] \\ &- \text{nests}\left[\text{permute 2}\,[i]\right][j]\big),\end{aligned} \tag{9}$$

where rand( ) is random number generator in [0, 1] interval and permute 1 and permute 2 are different rows permutation functions applied to the nests matrix [41]. Finally, the existing and new objective function values are compared for each nest and the best nest enters to the next generation according to the simple rule as given in the following:

$$\text{nest}_i^{t+1} = \begin{cases}\text{nest}_i^t, & \text{if } F\left(\text{nest}_i^t\right) < F\left(\text{new\_nest}_i^t\right) \\ \text{new\_nest}_i^t, & \text{otherwise.}\end{cases} \tag{10}$$

Termination. The generation of new nests and the discovering of the alien eggs steps are repeated until a predetermined stopping criterion is satisfied or maximum number of generations is reached.

4. Numerical Studies

4.1. 16-Link Network. In order to test the performance of the CS algorithm in solving the CNDP, it is first applied to the 16-link network which is most widely used test network by many researchers. This network consists of 16 links and 6 nodes as shown in Figure 2. For this network, two demand scenarios are considered as given in Table 1. Results obtained by the CS algorithm for different demand cases are compared with the results produced by other methods available in the literature. The compared methods for all numerical applications are given in Table 2. The link travel time function is defined as given in (11). Link parameters and demand data are adopted from Suwansirikul et al. [4]. Consider the following:

$$t_a\left(x_a, y_a\right) = \lambda_a + \varphi_a\left(\frac{x_a}{\theta_a + y_a}\right)^4. \tag{11}$$

The upper level objective function for the 16-link network is defined as

$$\begin{aligned}&\min_y \quad Z(x, y) = \sum_{a \in A}\left(t_a\left(x_a, y_a\right)x_a + d_a y_a\right) \\ &\text{s.t.} \quad 0 \le y_a \le h_a, \quad \forall a \in A,\end{aligned} \tag{12}$$

where d_a is the cost coefficient. The upper bound (h_a) was set to 10 and 20 for scenarios 1-2 for fair comparison with other

Table 1: Travel demand scenarios for the 16-link network.

Scenario	D_{16}	D_{61}	Total demand
1	5	10	15
2	10	20	30

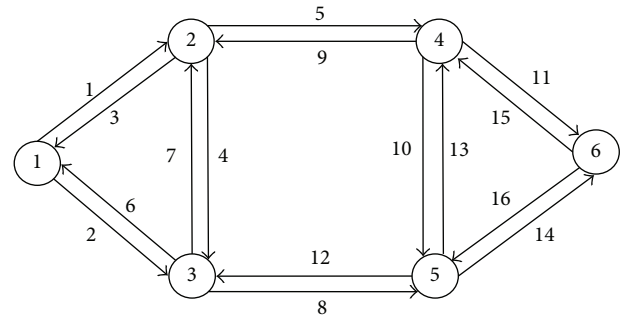


Figure 2: 16-link network.

algorithms. Results obtained by all algorithms for scenario 1 are presented in Tables 3 and 4.

Since the CS algorithm is a stochastic search method, the results obtained from this algorithm are selected as the best output of trials made by different random numbers. As can be seen from Table 4, the CS algorithm achieved to the value of 199.32 as its best output. Among all the compared algorithms, the SA produces the best solution but needs much more computational efforts than other algorithms in solving traffic assignment problem. It is clear that the CS produces better results with less computational efforts in comparison with other methods except SA, CG, and QNEW for scenario 1. However, they need much computational efforts than the CS algorithm in solving traffic assignment problem. Additionally, results show that the objective function values produced by all compared algorithms are very close to each other, but optimal capacity expansion vectors are not similarly consistent. This result shows us again that the CNDP has multiple optimal solutions.

In order to evaluate the sensitivity of the CS algorithm under different demand levels, scenario 2 is considered, and results are given in Tables 5 and 6. It can be clearly seen that the CS algorithm is able to produce the best solution among 14 algorithms, as well as with significant less computational efforts. In order to validate the obtained results for scenarios 1 and 2, the equilibrium link flows and travel times are given in Table 7.

4.2. Sioux Falls Network. In order to show the ability of the CS algorithm on realistic test network, the city of Sioux Falls is used which is probably the most used network for the CNDP. It consists of 24 nodes and 76 links. The link parameters of the network and travel demands between 552 O-D pairs are adopted from Suwansirikul et al. [4]. The link travel time function is used as given in (11). The dashed links 16, 17, 19, 20, 25, 26, 29, 39, 48, and 74 of Sioux Falls network are candidates for capacity expansion as shown in Figure 3. The upper level

Table 2: The compared algorithms on all test networks.

Methods	Sources
Hooke-Jeeves algorithm (HJ)	Abdulaal and LeBlanc [3]
Equilibrium Decomposed Optimization (EDO)	Suwansirikul et al. [4]
Modular In-core Nonlinear Optimization System (MINOS)	Suwansirikul et al. [4]
Genetic Algorithm (GA)	Mathew and Sharma [20]
Iterative Optimization Assignment algorithm (IOA)	Allsop [42]
Simulated Annealing algorithm (SA)	Friesz et al. [11]
Sensitivity Analysis Based algorithm (SAB)	Yang and Yagar [43]
Augmented Lagrangian algorithm (AL)	Meng et al. [14]
Path based Mixed Integer Linear Program (MILP)	Wang and Lo [21]
Link based Mixed Integer Linear Program (LMILP)	Luathep et al. [44]
Penalty with MultiCutting plane method (PMC)	Li et al. [22]
Gradient projection method (GP)	Chiou [15]
Conjugate gradient projection method (CG)	Chiou [15]
Quasi-Newton projection method (QNEW)	Chiou [15]
PARATAN version of gradient projection method (PT)	Chiou [15]
Cuckoo search (CS) algorithm with Lévy flights	This paper

Table 3: Comparison of results from solving the 16-link network for scenario 1.

	MINOS	HJ	EDO	IOA	SA	AL
y_1	0	0	0	0	0	0
y_2	0	0	0	0	0	0
y_3	0	1.2	0.13	0	0	0.0062
y_4	0	0	0	0	0	0
y_5	0	0	0	0	0	0
y_6	6.58	3.00	6.26	6.95	3.1639	5.2631
y_7	0	0	0	0	0	0.0032
y_8	0	0	0	0	0	0
y_9	0	0	0	0	0	0
y_{10}	0	0	0	0	0	0
y_{11}	0	0	0	0	0	0.0064
y_{12}	0	0	0	0	0	0
y_{13}	0	0	0	0	0	0
y_{14}	0	0	0	0	0	0
y_{15}	7.01	3.00	0.13	5.66	0	0.71701
y_{16}	0.22	2.80	6.26	1.79	6.7240	6.7561
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Z	211.25	215.08	201.84	210.86	198.10378	202.9913
⋮	⋮	⋮	⋮	⋮	⋮	⋮
#	—	54	10	9	18300	2700

Note: Z describes the objective function value, and # denotes the number of Frank-Wolfe iterations performed.

Table 4: Comparison of results from solving the 16-link network for scenario 1 (continued).

	SAB	GP	CG	QNEW	MILP	CS
y_1	0	0	0	0	0	0
y_2	0	0	0	0	0	0
y_3	0	0	0	0	0	0
y_4	0	0	0	0	0	0
y_5	0	0	0	0	0	0
y_6	5.8352	5.8302	6.1989	6.0021	4.41	5.1894
y_7	0	0	0	0	0	0
y_8	0	0	0	0	0	0
y_9	0	0	0	0	0	0
y_{10}	0	0	0	0	0	0
y_{11}	0	0	0	0	0	0
y_{12}	0	0	0	0	0	0
y_{13}	0	0	0	0	0	0
y_{14}	0	0	0	0	0	0
y_{15}	0.9739	0.87	0.0849	0.1846	0	0
y_{16}	6.1762	6.1090	7.5888	7.5438	7.70	7.6076
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Z	204.7	202.24	199.27	198.68	199.781	199.32
⋮	⋮	⋮	⋮	⋮	⋮	⋮
#	6	14	7	12	—	3

Note: Z describes the objective function value, and # denotes the number of Frank-Wolfe iterations performed.

objective function for the Sioux Falls network is formulated as in the following:

$$\begin{aligned} \min_{y} \quad & Z(x,y) = \sum_{a\in A} \left(t_a(x_a, y_a)\, x_a + 0.001 d_a y_a^2\right) \\ \text{s.t} \quad & 0 \le y_a \le h_a, \quad \forall a \in A. \end{aligned} \tag{13}$$

The results obtained by the CS algorithm on the Sioux Falls network are compared with those generated by other existing methods, and they are given in Tables 8 and 9. From these tables, it can be observed that the CS algorithm is able to produce the best solution among the compared major algorithms except SA. Although the SA slightly outperformed

Table 5: Comparison of results from solving the 16-link network for scenario 2.

	MINOS	HJ	EDO	IOA	SA	AL	SAB
y_1	0	0	0	0	0	0	0.0189
y_2	4.61	5.40	4.88	4.55	0	4.6153	2.2246
y_3	9.86	8.18	8.59	10.65	10.1740	9.8804	9.3394
y_4	0	0	0	0	0	0	0
y_5	0	0	0	0	0	0	0
y_6	7.71	8.10	7.48	6.43	5.7769	7.5995	9.0466
y_7	0	0	0.26	0	0	0.0016	0
y_8	0.59	0.90	0.85	0.59	0	0.6001	0.0175
y_9	0	0	0	0	0	0.001	0
y_{10}	0	0	0	0	0	0	0
y_{11}	0	0	0	0	0	0	0
y_{12}	0	0	0	0	0	0.1130	0.0816
y_{13}	0	0	0	0	0	0	0
y_{14}	1.32	3.90	1.54	1.32	0	1.3184	0.0198
y_{15}	19.14	8.10	0.26	19.36	0	2.7265	2.1429
y_{16}	0.85	8.40	12.52	0.78	17.2786	17.5774	18.9835
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Z	557.14	557.22	540.74	556.61	528.497	532.71	536.084
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
#	—	134	12	13	24300	4000	45

Note: Z describes the objective function value, and # denotes the number of Frank-Wolfe iterations performed.

Table 6: Comparison of results from solving the 16-link network for scenario 2 (continued).

	GP	CG	QNEW	MILP	LMILP	PMC	CS
y_1	0.1013	0.1022	0.0916	0	0	0	0
y_2	2.1818	2.1796	2.1521	4.41	2.722	4.6905	4.6144
y_3	9.3423	9.3425	9.1408	10.00	9.246	9.9778	9.9419
y_4	0	0	0	0	0	0	0
y_5	0	0	0	0	0	0	0
y_6	9.0443	9.0441	8.8503	7.42	8.538	7.5554	7.3821
y_7	0	0	0	0	0	0	0
y_8	0.008	0.0074	0.0114	0.54	0	0.6333	0.5922
y_9	0	0	0	0	0	0	0
y_{10}	0	0	0	0	0	0	0
y_{11}	0	0	0	0	0	0	0
y_{12}	0.0375	0.0358	0.0377	0	0	0	0
y_{13}	0	0	0	0	0	0	0
y_{14}	0.0089	0.0083	0.0129	1.18	0	1.7664	1.3152
y_{15}	1.9433	1.9483	1.9706	0	0	0	0
y_{16}	18.9859	18.986	18.575	19.50	20.000	19.6737	20
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Z	534.017	534.109	534.08	523.627	526.488	522.748	522.396
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
#	31	16	11	—	—	—	4

Note: Z describes the objective function value, and # denotes the number of Frank-Wolfe iterations performed.

than CS, the objective function values obtained by both algorithms are quite close. In addition, the CS algorithm produced good results with much less computational efforts in solving the traffic assignment problem in comparison with SA. It should be noted that AL, HJ, and GA algorithms have the potential to produce good results for solving the CNDP, but they require much more computational efforts in solving traffic assignment problem than CS.

As presented in the previous numerical application, the equilibrium link flows and travel times produced by the CS on the Sioux Falls network are given in Tables 10 and 11 in order to give an opportunity to the readers for validating the obtained result by the CS algorithm.

Table 7: The equilibrium link flows and travel times from solving 16-link network.

Scenario 1				Scenario 2			
t_1	1	x_1	0	t_1	1	x_1	0
t_2	2.3125	x_2	5	t_2	3.0961	x_2	10
t_3	3.5915	x_3	6.0287	t_3	4.291	x_3	15.3514
t_4	4	x_4	0	t_4	4	x_4	0
t_5	5	x_5	0	t_5	5	x_5	0
t_6	3.9215	x_6	3.9713	t_6	3.2153	x_6	4.6486
t_7	1	x_7	0	t_7	1	x_7	0
t_8	1.0625	x_8	5	t_8	1.7944	x_8	10
t_9	2.0025	x_9	6.0287	t_9	2.1081	x_9	15.3514
t_{10}	3	x_{10}	0	t_{10}	3	x_{10}	0
t_{11}	9	x_{11}	0	t_{11}	9	x_{11}	0
t_{12}	5.9805	x_{12}	3.9713	t_{12}	7.633	x_{12}	4.6486
t_{13}	4.0043	x_{13}	5.0101	t_{13}	4.2827	x_{13}	14.3194
t_{14}	2.1289	x_{14}	5	t_{14}	3.5986	x_{14}	10
t_{15}	9.0335	x_{15}	1.0186	t_{15}	9.8617	x_{15}	1.032
t_{16}	6.3125	x_{16}	8.9814	t_{16}	6.3622	x_{16}	18.968

5. Conclusions

In this paper, the CS algorithm with Lévy flights has been introduced to solve the CNDP, which is formulated as a bilevel programming model. In the upper level, the objective function is defined as the sum of total travel time and investment cost of link capacity expansions. The lower level is formulated as UE static traffic assignment problem, and Frank-Wolfe method is used to solve it.

The proposed model is first tested on the 16-link network, which is widely used network for solving the CNDP. Two scenarios are considered in order to evaluate the sensitivity of the CS algorithm to different demand levels. Results obtained by the proposed algorithm are compared with those generated by existing major methods in the literature. From the results, it has been found that the CS algorithm is able to produce good results for solving the CNDP, especially

Table 8: Comparison of results from solving the Sioux Falls network.

Initial value of y_a	HJ	EDO	SA	AL	IOA	SAB
	1.0	12.5	6.25	12.5	12.5	12.5
y_{16}	3.8	4.59	5.38	5.5728	4.6875	5.7392
y_{17}	3.6	1.52	2.26	1.6343	3.9063	5.7182
y_{19}	3.8	5.45	5.50	5.6228	1.2695	4.9591
y_{20}	2.4	2.33	2.01	1.6443	1.6599	4.9612
y_{25}	2.8	1.27	2.64	3.1437	2.3331	5.5066
y_{26}	1.4	2.33	2.47	3.2837	2.3438	5.5199
y_{29}	3.2	0.41	4.54	7.6519	5.5651	5.8024
y_{39}	4.0	4.59	4.45	3.8035	4.6862	5.5902
y_{48}	4.0	2.71	4.21	7.3820	5.4688	5.8439
y_{74}	4.0	2.71	4.67	3.6935	6.2500	5.8662
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Z	81.77	83.47	80.87	81.75	87.34	84.21
⋮	⋮	⋮	⋮	⋮	⋮	⋮
#	108	12	3900	2700	31	11

Note: the upper bound for y was set to 25 for CS. Z describes the objective function value, and # denotes the number of Frank-Wolfe iterations performed.

Table 9: Comparison of results from solving the Sioux Falls network (continued).

Initial value of y_a	GP	CG	QNEW	PT	GA	CS
	12.5	12.5	6.25	12.5	—	—
y_{16}	4.8693	4.7691	4.9776	5.0237	5.17	5.0916
y_{17}	4.8941	4.8605	5.0287	5.2158	2.94	1.3515
y_{19}	1.8694	3.0706	1.9412	1.8298	4.72	6.4903
y_{20}	1.5279	2.6836	2.1617	1.5747	1.76	2.2995
y_{25}	2.7168	2.8397	2.6333	2.7947	2.39	2.9074
y_{26}	2.7102	2.9754	2.7923	2.6639	2.91	2.0515
y_{29}	6.2455	5.6823	5.7462	6.1879	2.92	3.6725
y_{39}	5.0335	4.2726	5.6519	4.9624	5.99	5.2202
y_{48}	3.7597	4.4026	4.5738	4.0674	3.63	3.4230
y_{74}	3.5665	5.5183	4.1747	3.9199	4.43	4.8798
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Z	82.71	82.53	83.08	82.53	81.74	81.51
⋮	⋮	⋮	⋮	⋮	⋮	⋮
#	9	6	5	7	77	36

Note: the upper bound for y was set to 25 for CS. Z describes the objective function value, and # denotes the number of Frank-Wolfe iterations performed.

Table 10: The resulting link travel times from solving Sioux Falls network.

t_1	0.0600	t_{20}	0.0460	t_{39}	0.0775	t_{58}	0.0808
t_2	0.0402	t_{21}	0.1270	t_{40}	0.1180	t_{59}	0.1036
t_3	0.0600	t_{22}	0.1011	t_{41}	0.1295	t_{60}	0.0461
t_4	0.0884	t_{23}	0.0935	t_{42}	0.0889	t_{61}	0.1046
t_5	0.0402	t_{24}	0.1284	t_{43}	0.1127	t_{62}	0.0925
t_6	0.0486	t_{25}	0.0506	t_{44}	0.1304	t_{63}	0.0924
t_7	0.0411	t_{26}	0.0552	t_{45}	0.0512	t_{64}	0.0923
t_8	0.0484	t_{27}	0.1286	t_{46}	0.0933	t_{65}	0.0499
t_9	0.0281	t_{28}	0.1120	t_{47}	0.1053	t_{66}	0.1248
t_{10}	0.0847	t_{29}	0.1247	t_{48}	0.1301	t_{67}	0.0930
t_{11}	0.0280	t_{30}	0.1776	t_{49}	0.1016	t_{68}	0.0924
t_{12}	0.1087	t_{31}	0.0854	t_{50}	0.0348	t_{69}	0.0495
t_{13}	0.0974	t_{32}	0.1277	t_{51}	0.1775	t_{70}	0.1335
t_{14}	0.0947	t_{33}	0.1337	t_{52}	0.1023	t_{71}	0.0912
t_{15}	0.1169	t_{34}	0.1139	t_{53}	0.0802	t_{72}	0.1311
t_{16}	0.0588	t_{35}	0.0411	t_{54}	0.0210	t_{73}	0.0493
t_{17}	0.0559	t_{36}	0.1328	t_{55}	0.0347	t_{74}	0.0812
t_{18}	0.0210	t_{37}	0.0310	t_{56}	0.0461	t_{75}	0.1256
t_{19}	0.0457	t_{38}	0.0310	t_{57}	0.0512	t_{76}	0.0497

Table 11: The resulting equilibrium link flows from solving Sioux Falls network.

x_1	6.6117	x_{20}	13.9081	x_{39}	16.2801	x_{58}	10.2387
x_2	9.5834	x_{21}	5.8748	x_{40}	9.0660	x_{59}	8.9514
x_3	6.8855	x_{22}	8.1800	x_{41}	9.3401	x_{60}	23.5386
x_4	7.4205	x_{23}	15.3377	x_{42}	8.4098	x_{61}	8.9850
x_5	9.3097	x_{24}	5.9425	x_{43}	21.1721	x_{62}	6.9501
x_6	18.6187	x_{25}	24.4434	x_{44}	9.4240	x_{63}	7.8222
x_7	15.2883	x_{26}	24.3975	x_{45}	18.2917	x_{64}	6.9563
x_8	18.4954	x_{27}	18.1153	x_{46}	17.7595	x_{65}	9.3722
x_9	22.6678	x_{28}	21.1013	x_{47}	8.3726	x_{66}	10.6043
x_{10}	6.3151	x_{29}	15.6432	x_{48}	15.4477	x_{67}	17.7493
x_{11}	22.6053	x_{30}	8.4342	x_{49}	12.0336	x_{68}	7.8211
x_{12}	9.1783	x_{31}	6.3642	x_{50}	19.9374	x_{69}	9.3819
x_{13}	15.6613	x_{32}	18.0856	x_{51}	8.4327	x_{70}	9.6804
x_{14}	7.6942	x_{33}	8.4341	x_{52}	12.0139	x_{71}	8.4095
x_{15}	9.4394	x_{34}	8.9825	x_{53}	10.2598	x_{72}	9.6788
x_{16}	18.9802	x_{35}	15.1379	x_{54}	17.8759	x_{73}	8.9006
x_{17}	14.1828	x_{36}	8.4800	x_{55}	19.8542	x_{74}	16.1756
x_{18}	17.6012	x_{37}	17.9425	x_{56}	23.4572	x_{75}	10.6008
x_{19}	19.5151	x_{38}	17.9481	x_{57}	18.3465	x_{76}	8.8986

under heavier demand condition. Secondly, the performance of the CS algorithm is tested on the Sioux Falls city network. In comparison with the results obtained by the other major algorithms except SA, the CS algorithm achieved the best solution. Although the SA slightly outperformed than CS, it needs much more computational efforts in solving the traffic assignment problem. It is clear that the CS algorithm gives promising results in terms of objective function value and required computational effort and would be considered for large-scale road network applications in solving the CNDP.

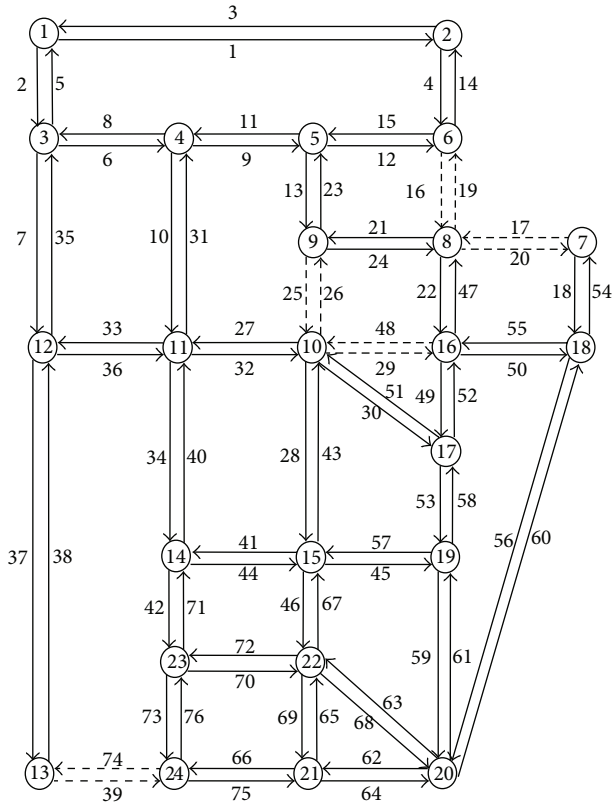


Figure 3: Sioux Falls network.

References

[1] J. G. Wardrop, "Some theoretical aspects of road traffic research," *Proceedings of the Institution of Civil Engineers Part II*, vol. 1, pp. 325–378, 1952.

[2] O. Baskan, "Harmony search algorithm for continuous network design problem with link capacity expansions," *KSCE Journal of Civil Engineering*, pp. 1–11, 2013.

[3] M. Abdulaal and L. J. LeBlanc, "Continuous equilibrium network design models," *Transportation Research B*, vol. 13, no. 1, pp. 19–32, 1979.

[4] C. Suwansirikul, T. L. Friesz, and R. L. Tobin, "Equilibrium decomposed optimisation: a heuristic for the continuous equilibrium network design problem," *Transportation Science*, vol. 21, no. 4, pp. 254–263, 1987.

[5] P. Marcotte, "Network optimization with continuous control parameters," *Transportation Science*, vol. 17, no. 2, pp. 181–197, 1983.

[6] P. Marcotte and G. Marquis, "Efficient implementation of heuristics for the continuous network design problem," *Annals of Operations Research*, vol. 34, no. 1, pp. 163–176, 1992.

[7] T. L. Friesz, R. L. Tobin, H.-J. Cho, and N. J. Mehta, "Sensitivity analysis based heuristic algorithms for mathematical programs with variational inequality constraints," *Mathematical Programming*, vol. 48, no. 2, pp. 265–284, 1990.

[8] H. J. Cho, *analysis of equilibrium network flows and its application to the development of solution methods for equilibrium network design problems [Ph.D. thesis]*, University of Pennsylvania, Philadelphia, Pa, USA, 1988.

[9] Y. Hai, "Sensitivity analysis for queuing equilibrium network flow and its application to traffic control," *Mathematical and Computer Modelling*, vol. 22, no. 4–7, pp. 247–258, 1995.

[10] H. Yang, "Sensitivity analysis for the elastic-demand network equilibrium problem with applications," *Transportation Research B*, vol. 31, no. 1, pp. 55–70, 1997.

[11] T. L. Friesz, H.-J. Cho, N. J. Mehta, R. L. Tobin, and G. Anandalingam, "Simulated annealing approach to the network design problem with variational inequality constraints," *Transportation Science*, vol. 26, no. 1, pp. 18–26, 1992.

[12] T. L. Friesz, G. Anandalingam, N. J. Mehta, K. Nam, S. J. Shah, and R. L. Tobin, "The multiobjective equilibrium network design problem revisited: a simulated annealing approach," *European Journal of Operational Research*, vol. 65, no. 1, pp. 44–57, 1993.

[13] G. A. Davis, "Exact local solution of the continuous network design problem via stochastic user equilibrium assignment," *Transportation Research B*, vol. 28, no. 1, pp. 61–75, 1994.

[14] Q. Meng, H. Yang, and M. G. H. Bell, "An equivalent continuously differentiable model and a locally convergent algorithm for the continuous network design problem," *Transportation Research B*, vol. 35, no. 1, pp. 83–105, 2001.

[15] S.-W. Chiou, "Bilevel programming for the continuous transport network design problem," *Transportation Research B*, vol. 39, no. 4, pp. 361–383, 2005.

[16] X. Ban, H. X. Liu, J. Lu, and M. C. Ferris, "Decomposition scheme for continuous network design problem with asymmetric user equilibria," *Transportation Research Record*, no. 1964, pp. 185–192, 2006.

[17] A. Karoonsoontawong and S. T. Waller, "Dynamic continuous network design problem: linear bilevel programming and metaheuristic approaches," *Transportation Research Record*, no. 1964, pp. 104–117, 2006.

[18] Z. Gao, H. Sun, and H. Zhang, "A globally convergent algorithm for transportation continuous network design problem," *Optimization and Engineering*, vol. 8, no. 3, pp. 241–257, 2007.

[19] T. Xu, H. Wei, and G. Hu, "Study on continuous network design problem using simulated annealing and genetic algorithm," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1322–1328, 2009.

[20] T. V. Mathew and S. Sharma, "Capacity expansion problem for large urban transportation networks," *Journal of Transportation Engineering*, vol. 135, no. 7, pp. 406–415, 2009.

[21] D. Z. W. Wang and H. K. Lo, "Global optimum of the linearized network design problem with equilibrium flows," *Transportation Research B*, vol. 44, no. 4, pp. 482–492, 2010.

[22] C. Li, H. Yang, D. Zhu, and Q. Meng, "A global optimization method for continuous network design problems," *Transportation Research B*, vol. 46, pp. 1144–1158, 2012.

[23] O. Baskan and M. Dell'Orco, "Artificial bee colony algorithm for continuous network design problem with link capacity expansions," in *Proceedings of the 10th International Congress on Advances in Civil Engineering*, Middle East Technical University, Ankara, Turkey, October 2012.

[24] O. Baskan and H. Ceylan, "Modified differential evolution algorithm for the continuous network design problem," in *Proceedings of the 16th Euro Working Group on Transportation*, Porto, Portugal, 2013.

[25] H. Yang and J. Y. T. Wang, "Travel time minimization versus reserve capacity maximization in the network design problem," *Transportation Research Record*, no. 1783, pp. 17–26, 2002.

[26] G. Ziyou and S. Yifan, "A reserve capacity model of optimal signal control with user-equilibrium route choice," *Transportation Research B*, vol. 36, no. 4, pp. 313–323, 2002.

[27] S.-W. Chiou, "A hybrid approach for optimal design of signalized road network," *Applied Mathematical Modelling*, vol. 32, no. 2, pp. 195–207, 2008.

[28] C. S. Fisk, "Optimal signal controls on congested networks," in *Proceedings of the 9th International Symposium on Transportation and Traffic Theory*, pp. 197–216, VNU Science Press, Utrecht, 1984.

[29] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, pp. 95–110, 1956.

[30] Y. Sheffi, *Urban Transport Networks: Equilibrium Analysis With Mathematical Programming Methods*, Prentice-Hall, New Jersey, NJ, USA, 1985.

[31] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the world congress on nature and biologically inspired computing (NaBIC '09)*, pp. 210–214, IEEE Press, Piscataway, NJ, USA, December 2009.

[32] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[33] X.-S. Yang and S. Deb, "Multiobjective cuckoo search for design optimization," *Computers & Operations Research*, vol. 40, no. 6, pp. 1616–1624, 2013.

[34] R. B. Payne, M. D. Sorenson, and K. Klitz, *The Cuckoos*, Oxford University Press, 2005.

[35] E. Valian, S. Mohanna, and S. Tavakoli, "Improved Cuckoo search algorithm for global optimization," *International Journal of Communications and Information Technology*, vol. 1, no. 1, pp. 31–44, 2011.

[36] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2nd edition, 2010.

[37] X. S. Yang and S. Deb, "Cuckoo search: recent advances and applications," *Neural Computing and Applications*, 2013.

[38] E. Valian and E. Valian, "A cuckoo search algorithm by Lévy flights for solving reliability redundancy allocation problems," *Engineering Optimization*, 2013.

[39] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *Journal of Computational Physics*, vol. 226, no. 2, pp. 1830–1844, 2007.

[40] S. Walton, O. Hassan, K. Morgan, and M. R. Brown, "Modified cuckoo search: a new gradient free optimisation algorithm," *Chaos, Solitons and Fractals*, vol. 44, no. 9, pp. 710–718, 2011.

[41] A. Kaveh and T. Bakhshpoori, "Optimum design of steel frames using Cuckoo Search algorithm with Lévy flights," *Structural Design of Tall and Special Buildings*, vol. 22, no. 13, pp. 1023–1036, 2013.

[42] R. E. Allsop, "possibilities for using traffic control to influence trip distribution and route choice," in *Proceedings of the 6th International Symposium on Transportation and Traffic Theory*, pp. 345–375, Elsevier, New York, NY, USA, 1974.

[43] H. Yang and S. Yagar, "Traffic assignment and signal control in saturated road networks," *Transportation Research A*, vol. 29, no. 2, pp. 125–139, 1995.

[44] P. Luathep, A. Sumalee, W. H. K. Lam, Z.-C. Li, and H. K. Lo, "Global optimization method for mixed transportation network design problem: a mixed-integer linear programming approach," *Transportation Research B*, vol. 45, no. 5, pp. 808–827, 2011.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 912056, 9 pages
http://dx.doi.org/10.1155/2013/912056

Research Article

A Cooperative Coevolutionary Cuckoo Search Algorithm for Optimization Problem

Hongqing Zheng[1] and Yongquan Zhou[1,2]

[1] *Guangxi Key Laboratory of Hybrid Computation and Integrated Circuit Design Analysis, Nanning, Guangxi 530006, China*
[2] *College of Information Science and Engineering, Guangxi University for Nationalities, Nanning, Guangxi 530006, China*

Correspondence should be addressed to Yongquan Zhou; yongquanzhou@126.com

Received 24 May 2013; Accepted 8 July 2013

Academic Editor: Xin-She Yang

Taking inspiration from an organizational evolutionary algorithm for numerical optimization, this paper designs a kind of dynamic population and combining evolutionary operators to form a novel algorithm, a cooperative coevolutionary cuckoo search algorithm (CCCS), for solving both unconstrained, constrained optimization and engineering problems. A population of this algorithm consists of organizations, and an organization consists of dynamic individuals. In experiments, fifteen unconstrained functions, eleven constrained functions, and two engineering design problems are used to validate the performance of CCCS, and thorough comparisons are made between the CCCS and the existing approaches. The results show that the CCCS obtains good performance in the solution quality. Moreover, for the constrained problems, the good performance is obtained by only incorporating a simple constraint handling technique into the CCCS. The results show that the CCCS is quite robust and easy to use.

1. Introduction

High dimension numerical optimization problems tend to be complex, and general basic intelligent algorithms are difficult to obtain the global optimal solution. In order to solve this problem, many improved methods are put forward, such as evolutionary programming made faster [1], orthogonal genetic algorithm [2], and good point set based genetic algorithm [3], and these methods have achieved good effect.

Recently, a novel heuristic search algorithm, called Cuckoo Search (CS) in [4], has been proposed by Yang and Deb in 2009. The CS is a search algorithm based on the interesting breeding behavior such as brood parasitism of certain species of cuckoos. Each nest within the swarm is represented by a vector in multidimensional search space; the CS algorithm also determines how to update the position of cuckoo laid egg. Each cuckoo updates it position of lay egg based on current step size via Lévy flights. It has been shown that this simple model has been applied successfully to continuous nonlinear function, engineering optimization problem in [5], and so forth. Intelligent algorithm is based on the selection of the fittest in biological systems which have evolved by natural selection over millions of years, between organisms in nature not only competition but also cooperation. Potter et al. earlier proposed a cooperative coevolutionary genetic algorithm to function optimization (CCGA) [6], and Bergh et al. apply this idea to the standard particle group Algorithm to construct a new collaborative model (CPSO-SK) [7]. Liu et al. proposed an organizational evolutionary algorithm for numerical optimization [8], Mu et al. put forward M-elite coevolutionary algorithm for numerical optimization [9], and Fister et al. proposed memetic artificial bee colony algorithm for large-scale global optimization [10]. They have obtained the good effect in the numerical optimization problem. Based on this idea and combining evolutionary operators, this paper proposes a new algorithm of solving high-dimensional unconstrained, constrained, and engineering optimization problem, namely, a cooperative coevolutionary cuckoo search algorithm (CCCS) algorithm. Population of the algorithm is divided into M groups, each group has a leader, by annexation and collaborative operation between different organizations, and uses the cuboids crossover operator, discrete crossover operator, flip crossover operator, and mutation operator to achieve the

exchange of information between individuals, to promote the evolution of the population. Simulation experiments show that CCCS optimization ability is very strong, can well solve the unconstrained optimization, constrained optimization, and engineering optimization problems and so on.

The remainder of this paper is organized as follows: Section 2 briefly introduces the original cuckoo search algorithm. This is followed in Section 3 by new cooperative coevolutionary implements of the CS algorithm. Section 4 describes the definition of a constrained optimization and a unconstrained optimization problem in the penalty function approach. The results can be found and discussed in Section 5. Finally, some directions for future research are discussed in Section 6.

2. Original CS

CS is a heuristic search algorithm which has been proposed recently by Yang and Deb. The algorithm is inspired by the reproduction strategy of cuckoos. At the most basic level, cuckoos lay their eggs in the nests of other host birds, which may be of different species. The host bird may discover that the eggs are not it's own and either destroy the egg or abandon the nest all together. This has resulted in the evolution of cuckoo eggs which mimic the eggs of local host birds. To apply this as an optimization tool, Yang and Deb [4] used three ideal rules.

(1) Each cuckoo lays one egg, which represents a set of solution coordinates, at a time, and dumps it in a random nest.

(2) A fraction of the nests containing the best eggs, or solutions, will carry over to the next generation.

(3) The number of nests is fixed, and there is a probability that a host can discover an alien egg. If this happens, the host can either discard the egg or the nest, and this results in building a new nest in a new location.

This algorithm uses a balanced combination of a local random walk and the global explorative random walk, controlled by a switching parameter p_a. The local random walk can be written as

$$x_i^{t+1} = x_i^t + \partial s \otimes H\left(p_a - \varepsilon\right) \otimes \left(x_j^t - x_k^t\right), \tag{1}$$

where x_j^t and x_k^t are two different solutions selected randomly by random permutation, $H(u)$ is a Heaviside function, ε is a random number drawn from a uniform distribution, and s is the step size. On the other hand, the global random walk is carried out by using Lévy flights

$$x_i^{t+1} = x_i^t + \partial L\left(s, \lambda\right), \tag{2}$$

where

$$L\left(s, \lambda\right) = \frac{\lambda\Gamma\left(\lambda\right)\sin\left(\pi\lambda/2\right)}{\pi}\frac{1}{s^{1+\lambda}}, \qquad \left(s \gg s_0 > 0\right). \tag{3}$$

Here $\partial > 0$ is the step-size-scaling factor, which should be related to the scales of the problem of interests. In most cases, we can use $\partial = O(L/10)$, and $\partial = O(L/100)$ can be more effective and can avoid flying too far.

3. A Cooperative Coevolutionary Cuckoo Search Algorithm

In section, taking inspiration from an organizational evolutionary algorithm, we present a cooperative coevolutionary cuckoo search algorithm (CCCS) which integers annexing operator and cooperating operator, in the core the cuckoo search algorithm. This proposed model will focus on enhancing diversity and the performance of the cuckoo search algorithm.

3.1. Splitting Operator. When a size is too large usually it is split into several small organizations; let $\mathrm{Max}_{\mathrm{or}}$ be the parameter controlling the maximum size of organization.

3.2. Annexing Operator. Two organizations, $\mathrm{org}_{p1} = \{x_1, x_2, \ldots, x_M\}$ and $\mathrm{org}_{p2} = \{y_1, y_2, \ldots, y_N\}$, are randomly selected from the current generation. Choose their leaders using CS algorithm. If the org_{p1} is the winner; thus org_{p1} will annex org_{p2} to generate a new organization, $\mathrm{org}_c = \{z_1, z_2, \ldots, z_M, z_{M+1}, z_{M+2}, \ldots z_{M+N}\}$, where $z_i = x_i$, $i = 1, 2, \ldots, M$. Let AS be a predefined parameter. Then, if rand < AS, z_j, $j = M+1, M+2, \ldots, M+N$ are generated by (4). Otherwise they are generated by (5). x_k is the leader of an organization and r_{jk} are new member

$$r_{jk} = \begin{cases} \underline{x_k} & \beta_k < \underline{x_k}, \\ \overline{x_k} & \beta_k > \overline{x_k},\ \beta_k = x_k + \partial_k \times \left(x_k - y_{jk}\right), \\ & k = 1, 2, \ldots, n, \\ \beta_k & \text{otherwise}, \end{cases} \tag{4}$$

$$r_{jk} = \begin{cases} \underline{x_k} + \partial \times \left(\overline{x_k} - \underline{x_k}\right) & \beta_k\left(0,1\right) < \dfrac{1}{n}, \\ x_k & \text{otherwise.} \end{cases} \tag{5}$$

After r_j, $j = 1, 2, \ldots N$ are generated, z_{j+M}, $j = 1, 2, \ldots, M$ are determined in (6):

$$z_{j+M} = \begin{cases} r_j & r_j < y_j, \\ r_j & \left(y_j > r_j\right), \\ & \left\{U_j\left(0,1\right)\right\} < \exp\left(f\left(r_j\right) - f\left(y\right)\right), \\ y_j & \text{otherwise.} \end{cases} \tag{6}$$

3.3. Cooperating Operator. Two organizations, $\mathrm{org}_{p1} = \{x_1, x_2, \ldots, x_M\}$ and $\mathrm{org}_{p2} = \{y_1, y_2, \ldots, y_N\}$, are randomly selected from the current generation. Let $\mathrm{CS} \in (0,1)$ be a predefined parameter; if rand < CS, the child organization is generated in (7). x_k, y_k are the leader of organization respectively. Otherwise use (8); i is random integer and uses

```
Begin
    Initializing population p_0 with n_0 organizations, and each organization has one member;
        t ← 0;
    While (the termination criteria are not reached) do
    Begin
        For each organization in p_t, if the number of it more than 20, performing the splitting
        operator on it, deleting it from p_t, and adding the child organizations into p_{t+1};
        While (the number of organizations in is p_t greater than 1) do
        Begin
            Randomly selecting two parent organizations org_{p1} and org_{p2} from p_t;
            Performing the CS and selecting their leaders;
            If rand < 0.5
                Annexing operator;
            Else
                Cooperating operator;
                If f(w) < f(x) and f(z) < f(y)
                    x = w;
                    y = z;
                End
                adding the child organizations into p_{t+1};
            End
            Deleting org_{p1} and org_{p2} form p_t;
        End
        Deleting the u organizations form p_{t+1};
        % u is the child number of join organizations
        p_t ← p_{(t+1)};
        t ← t + 1;
    End
    output the best solution in p_t
End
```

Algorithm 1: A cooperative coevolutionary cuckoo search algorithm (minimum).

flip operator;

$$q_k = \partial_k \times x_k + (1 - \partial_k) \times y_k,$$
$$r_k = (1 - \partial_k) \times x_k + \partial_k \times y_k, \quad (7)$$
$$k = 1, 2, \ldots, n,$$

$$x = (x_1, x_2, \ldots, \boxed{x_i, \ldots, x_n}), \quad y = (y_1, y_2, \ldots, \boxed{y_i, \ldots, y_n}).$$
$$\downarrow \qquad\qquad \downarrow$$
$$w = (x_1, x_2, \ldots, \boxed{y_n, \ldots, y_i}), \quad z = (y_1, y_2, \ldots, \boxed{x_n, \ldots, x_i}). \quad (8)$$

3.4. The Pseudo Code of the Proposed Algorithm is Shown in Algorithm 1. In the initialization, each organization has only one member, and the population has total n_0 organizations. During the evolutionary process, the number of the organizations changes; this is just to maintain the diversity of the population. The main difference between the CCCS and the OEA is that populations changes during the optimization process, in OEA, the number of populations in the optimization process is the same. In contrast, the number of the population in CCCS is changing. In addition, cooperating operators of CCCS and OEA are also different.

4. Problem Definition

A unconstrained optimization problems (UCOPs) are formulated as solving the objective function

$$\text{minimize} \quad f(x), \quad x = (x_1, x_2, \ldots, x_n) \in s, \quad (9)$$

where $s \subseteq \mathfrak{R}^n$ defines the search space which is an n-dimensional space bounded by the parametric constraints $\underline{x_i} \le x_i \le \overline{x_i}$, $i = 1, 2, \ldots, n$. Thus, $s = [\underline{x}, \overline{x}]$, where $\underline{x} = (\underline{x_1}, \underline{x_2}, \ldots, \underline{x_n})$ and $\overline{x} = (\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$.

A constrained optimization problems (COPs) can be formulated as solving objective function

$$\text{minimize} \quad f(x), \quad x = (x_1, x_2, \ldots, x_n) \in s \cap \chi, \quad (10)$$

where s is the same with that of (9), and the feasible region χ is

$$\chi = \left\{ x \in \mathfrak{R}^n \mid g_j(x) \le 0,\ j = 1, 2, \ldots, m \right\}, \quad (11)$$

where $g_j(x)$, $j = 1, 2, \ldots, m$ are constraints.

4.1. Constraint Handling. In the penalty function approach, nonlinear constraints can be collapsed with the cost function into a response functional. By doing this, the constrained optimization problem is transformed into an unconstrained

Table 1: Experimental result of CCCS on 15 unconstrained benchmark functions over 50 trails.

f	$f_{\min}$	Best function value	Mean function value	Standard deviation	Worst function value
F_{01}	0	$8.804e-242$	$2.011e-217$	0	$2.011e-216$
F_{02}	0	$4.271e-143$	$1.628e-130$	$4.142e-130$	$1.323e-129$
F_{03}	0	$6.280e-192$	$5.464e-154$	$1.729e-153$	$5.464e-153$
F_{04}	0	$1.085e-121$	$1.333e-098$	$4.080e-098$	$1.294e-097$
F_{05}	0	25.0366	25.4182	0.3013	25.8868
F_{06}	0	0	0	0	0
F_{07}	0	$5.309e-005$	$3.080e-004$	$2.391e-004$	$8.047e-004$
F_{08}	−12569.5	$-1.25694870e+004$	$-1.25694867e+004$	$4.830e-004$	$-1.25694860e+004$
F_{09}	0	0	0	0	0
F_{10}	0	$1.655e-7$	$3.663e-5$	$2.654e-9$	$3.188e-4$
F_{11}	0	0	0	0	0
F_{12}	0	$6.845e-011$	$9.518e-007$	$2.780e-006$	$8.635e-006$
F_{13}	0	$2.073e-013$	$1.723e-007$	$2.657e-007$	$6.775e-007$
F_{14}	−99.60	−96.6008	−83.7328	6.1910	−75.6993
F_{15}	−78.33236	−78.33233	−78.33232	$8.654e-006$	−78.33230

Table 2: Comparison between MECA, OEA, and CCCS over 50 trials.

f	$f_{\min}$	Mean function value			Standard deviation		
		CCCS	MECA	OEA	CCCS	MECA	OEA
F_{01}	0	**$2.011e-217$**	$4.228e-183$	$2.481e-30$	**0**	0	$1.128e-29$
F_{02}	0	**$1.628e-130$**	$1.845e-110$	$2.068e-13$	**$4.142e-130$**	$3.113e-110$	$1.440e-12$
F_{03}	0	**$5.464e-154$**	$3.274e-95$	$1.883e-9$	**$1.729e-153$**	$2.313e-94$	$3.726e-9$
F_{04}	0	**$1.333e-098$**	$5.124e-2$	$8.821e-2$	**$4.080e-098$**	$9.732e-2$	$2.356e-2$
F_{05}	0	**25.4182**	$7.973e-2$	0.227	**0.3013**	$5.638e-1$	0.941
F_{06}	0	**0**	0	0	**0**	0	0
F_{07}	0	**$3.080e-004$**	$4.084e-4$	$3.297e-3$	**$2.391e-004$**	$3.800e-4$	$1.096e-3$
F_{08}	−12569.5	**−12569.4867**	−12569.4866	−12569.4866	**$4.830e-004$**	$7.350e-12$	$5.555e-12$
F_{09}	0	**0**	0	$5.430e-17$	**0**	0	$1.683e-16$
F_{10}	0	**$3.663e-5$**	0	$5.336e-14$	**$2.654e-9$**	0	$2.945e-13$
F_{11}	0	**0**	$3.844e-3$	$1.317e-2$	**0**	$7.130e-3$	$1.561e-2$
F_{12}	0	**$9.518e-007$**	$1.571e-32$	$9.207e-30$	**$2.780e-006$**	$5.529e-48$	$6.436e-31$
F_{13}	0	**$1.723e-007$**	$1.350e-32$	$4.323e-19$	**$2.657e-007$**	$1.106e-47$	$2.219e-18$
F_{14}	−99.60	**−83.7328**	−98.7094891	−99.5024042	**6.1910**	$1.450e-1$	$2.526e-2$
F_{15}	−78.33236	**−78.33232**	−78.3323314	−78.3323314	**$8.654e-6$**	$1.005e-13$	$2.804e-11$

optimization problem simpler to solve [35]. For example, if there are some nonlinear equality constraints ϕ_i and some inequality constraints φ_j, the response functional PI can be defined as follows:

$$\prod\left(x,\mu_i,\nu_j\right)=f\left(x\right)+\sum_{i=1}^{M}\mu_i\phi_i^2\left(x\right)+\sum_{j=1}^{N}\nu_j\varphi_j^2\left(x\right), \quad (12)$$

where $1 \le \mu_i$ and $0 \le \nu_i$. The coefficients of penalty terms should be large enough; their values may depend on the specific optimization problem. The contribution of any equality constraints function to the response functional $\prod$ is null but increases significantly as soon as the constraint is violated. The same applies to inequality constraints when they become critical.

5. Implementation in Optimization Problems

All computational experiments are conducted with Matlab R2010a and run on Celeron(R) Dual-core CPU T3100, 1.90 GHZ with 2 GB memory capacity under windows7.

5.1. Experimental Studies on Unconstrained Optimization Problems. In this section, 15 benchmark functions (F_{01}–F_{15}) are used to test the performance of CCCS in solving UCOPS. F_{01}–F_{13} are f_1–f_{13} in [1] and F_{14}, F_{15} are f_7, f_9 in [36]. The problem dimension is set to 30 for F_{01}–F_{13} and 100 for F_{14} and F_{15}. In this manner, these functions have so many local

Table 3: Comparison between MECA, CCGA, CPSO, and CCCS over 50 trials.

f	CCCS	MECA	CCGA	CPSO
F_{05} (f_0)	**25.0366**	6.43e − 1	3.80	1.94e − 1
F_{03} (f_1)	**5.464e − 154**	1.23e − 64	1.38e + 2	2.55e − 128
F_{10} (f_2)	**1.655e − 7**	0	9.51e − 2	2.78e − 14
F_{09} (f_3)	**0**	0	1.22	0
F_{11} (f_4)	**0**	3.94e − 3	2.20e − 1	1.86e − 2

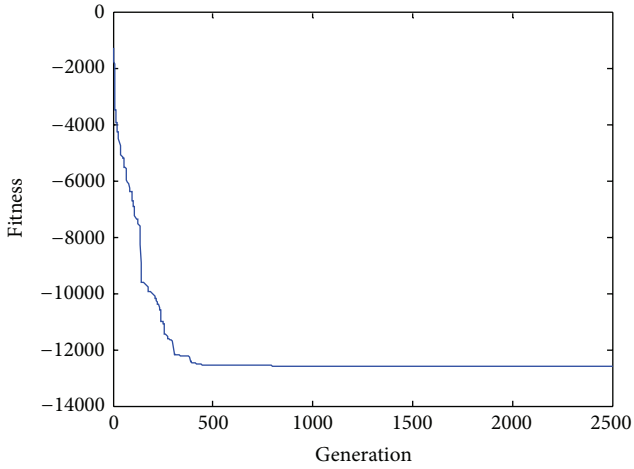


Figure 1: Convergence curve of F_{08}.

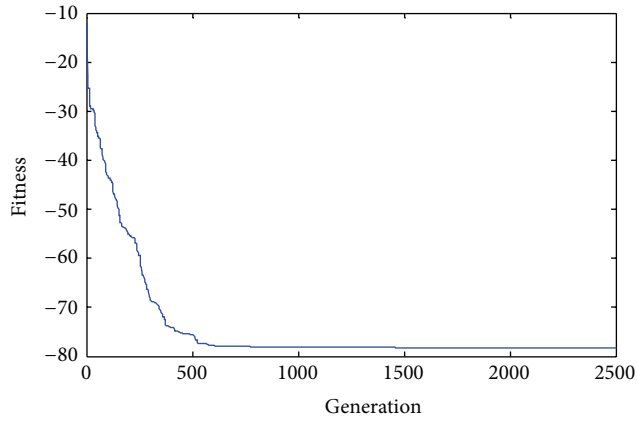


Figure 2: Convergence curve of F_{15}.

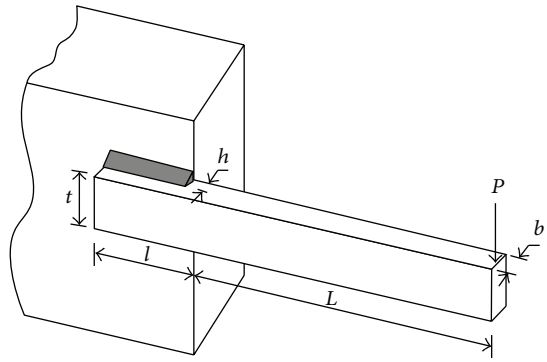


Figure 3: Schematic of the welded beam design problem.

minima that they are challenging enough for performance evaluation. The parameters of CCCS are set as follows: the number of organization is 10, others refer to [8], and the number of iterations is 2500.

5.1.1. Experimental Results of CCCS. Table 1 summarizes the experimental results of CCCS, which include the best, the mean, the standard deviation, and the worst function values found. As can be seen, besides F_{05}, F_{14}, other function values reached the theoretical value or are very close to the theoretical value. What should be noted is that the global optimum for F_{11} is 0 every time, but both MECA (M-elite co-evolutionary algorithm for numerical optimization) and OEA (An organizational evolutionary algorithm for numerical optimization) cannot find the global optimal solution. Convergence curve of F_{08} and F_{15} as shown in Figures 1 and 2 respectively, other convergence curve figure has been omitted.

5.1.2. Comparison between CCCS, MECA, and OEA. Table 2 shows statistical results of the CCCS optimization. As can be seen, as for functions F_{01}, F_{02}, F_{03}, F_{04}, F_{07}, F_{08}, and F_{11}, the mean function values of CCCS is better than MECA; as for functions F_{06} and F_{09}, both CCCS and MECA can find the global optimal solution. It is a pity that the results of these functions (F_{05}, F_{10}, F_{12}, F_{13}, and F_{14}) is worse than MECA.

5.1.3. Comparison between MECA, CCGA, CPSO, and CCCS. Table 3 summarizes the experimental results of comparison between MECA, CCGA, CPSO, and CCCS. Data of CCGA and CPSO algorithm results from the literature [9], MECA, CCGA, and CPSO requite 200000 function evaluations. But CCCS requires 2500 function evaluations to complete the optimization process. As a whole, the results of CCCS are better than those of MECA, OEA.

5.2. Experimental Studies on Constrained Optimization Problems. In this section, 11 benchmark functions (G_{01}–G_{11}) and 2 engineering design problems (welded beam design, pressure vessel design) are used to validate the performance of CCCS in solving constrained optimization problems. These functions are described in [37]. The equality constraints have been converted into inequality constraints, $|g(x) - \delta| \leq 0$, using the degree of violation $\delta = 0.00001$, the same with that of [37] (see Figures 3 and 5).

Table 4: The comparison between OEA, SMES, and CCCS.

f	$f_{\min}$	Method	Best FV	Mean FV	St. dev	Worst FV
G_{01}	−15	**CCCS**	**−15**	**−15**	**0**	**−15**
		OEA CHp	−15	−15	0	−15
		OEA CHc	−15	−14.78	7.637e − 1	−12
		SMES	−15.000	−15.000	0	−15.000
G_{02}	−0.803619	**CCCS**	**−0.7947**	**−0.7826**	**0.0093**	**−0.7694**
		OEA CHp	−0.803605	−0.782518	1.483e − 2	−0.746283
		OEA CHc	−0.803589	−0.782676	1.512e − 2	−0.743149
		SMES	−0.803601	−0.795238	1.67e − 2	−0.751322
G_{03}	−1	**CCCS**	**−1.000**	**−1.000**	**0**	**−1.000**
		OEA CHp	−1.000	−1.000	8.470e − 6	−1.000
		OEA CHc	−1.000	−1.000	1.131e − 5	−1.000
		SMES	−1.000	−1.000	2.09e − 4	−1.000
G_{04}	−30665.539	**CCCS**	**−3.0665539e + 004**	**−3.0665539e + 004**	**3.8348e − 012**	**−3.0665539e + 004**
		OEA CHp	−30665.539	−30665.539	1.735e − 11	−30665.539
		OEA CHc	−30665.539	−30665.539	1.837e − 11	−30665.539
		SMES	−30665.539	−30665.539	0	−30665.539
G_{05}	5126.498	**CCCS**	**5.1264981e + 003**	**5.1264981e + 003**	**9.5869e − 013**	**5.1264981e + 003**
		OEA CHp	5126.497	5127.048	7.071e − 1	5130.051
		OEA CHc	5126.532	5315.975	145.473	5900.26
		SMES	5126.599	5174.492	50.06	5304.167
G_{06}	−6961.814	**CCCS**	**−6.9618139e + 003**	**−6.9618139e + 003**	**9.5869e − 013**	**−6.9618139e + 003**
		OEA CHp	−6961.814	−6961.814	5.512e − 12	−6961.814
		OEA CHc	−6961.814	−6961.814	5.512e − 12	−6961.814
		SMES	−6961.814	−6861.284	1.85	−6952.482
G_{07}	24.306	**CCCS**	**2.4306209e + 001**	**2.4306209e + 001**	**4.2164e − 007**	**2.4306209e + 001**
		OEA CHp	24.308	24.373	7.615e − 2	24.655
		OEA CHc	24.307	24.392	1.178e − 1	24.973
		SMES	24.327	24.475	1.32e − 1	24.843
G_{08}	−0.095825	**CCCS**	**−9.5825041e − 002**	**−9.5825041e − 002**	**0**	**−9.5825041e − 002**
		OEA CHp	−0.095825	−0.095825	0	−0.095825
		OEA CHc	−0.095825	−0.095825	0	−0.095825
		SMES	−0.095825	−0.095825	0	−0.095825
G_{09}	680.630	**CCCS**	**6.8063006e + 002**	**6.8063006e + 002**	**1.1984e − 013**	**6.8063006e + 002**
		OEA CHp	680.630	680.632	1.718e − 3	680.638
		OEA CHc	680.630	680.632	2.163e − 3	680.641
		SMES	680.632	680.643	1.55e − 2	680.719
G_{10}	7049.331	**CCCS**	**7.0492e + 003**	**7.0492e + 003**	**4.7796e − 004**	**7.0492e + 003**
		OEA CHp	7052.236	7219.011	60.737	7326.032
		OEA CHc	7100.030	7231.357	86.409	7469.047
		SMES	7051.903	7253.603	136.02	7638.366
G_{11}	0.750	**CCCS**	**0.7500**	**0.7500**	**0**	**0.7500**
		OEA CHp	0.750	0.750	5.993e − 5	0.750
		OEA CHc	0.750	0.750	1.881e − 7	0.750
		SMES	0.75	0.75	1.52e − 4	0.75

The parameters of CCCS are set as follows: the number of iterations is 2500. However, the others of OEA are 24000. The experimental results of OEA are obtained over 50 independent trials. The running environment is the same as the previously. Table 4 shows the comparison results between OEA, SMES, and CCCS. As can be seen, besides G_{02}, other function values reached the theoretical value or are very close to the theoretical value. On the whole, the result of CCCS is better than those of SMES and CCCS.

Table 5: Welded beam problem: comparison of CS results with the literature.

Researcher(s)	Method	h	l	t	b	Cost	No. of evaluation
Coello [11]	GA	0.2088	3.4205	8.9975	0.2100	1.7483	N.A.
Leite and Topping [12]	GA	0.2489	6.1097	8.2484	0.2485	2.4000	6273
Deb [13]	GA	0.2489	6.1730	8.1789	0.2533	2.4331	320,080
Lemonge and Barbosa [14]	GA	0.2443	6.2117	8.3015	0.2443	2.3816	320,000
Bernardino et al. [15]	AISa-GA	0.2444	6.2183	8.2912	0.2444	2.3812	320,000
Atiqullah and Rao [16]	SAb	0.2471	6.1451	8.2721	0.2495	2.4148	N.A.
Hedar and Fukushima [17]	SA	0.2444	6.2175	8.2915	0.2444	2.3810	N.A.
Liu [18]	SA-GA	0.2231	1.5815	12.8468	0.2245	2.2500	26,466
Hwang and He [19]	SA-DSc	0.2444	6.2158	8.2939	0.2444	2.3811	56,243
Parsopoulos and Vrahatis [20]	PSO	N.A.	N.A.	N.A.	N.A.	1.9220	100,000
He et al. [21]	PSO	0.2444	6.2175	8.2915	0.2444	2.3810	30,000
Zhang et al. [22]	EAd	0.2443	6.2201	8.2940	0.2444	2.3816	28,897
Coello [23]	EA	N.A.	N.A.	N.A.	N.A.	1.8245	N.A.
Lee and Geem [24]	HSe	0.2442	6.2231	8.2915	0.2443	2.381	110,000
Mahdavi et al. [25]	HS	0.2057	3.4705	9.0366	0.2057	1.7248	200,000
Fesanghary et al. [26]	HS-SQP	0.2057	3.4706	9.0368	0.2057	1.7248	90,000
Siddall [27]	RAg	0.2444	6.2819	8.2915	0.2444	2.3815	N.A.
Akhtar et al. [28]	SBMh	0.2407	6.4851	8.2399	0.2497	2.4426	19,259
Ray and Liew [29]	SCAi	0.2444	6.2380	8.2886	0.2446	2.3854	33,095
Montes and Oca [30]	BFOj	0.2536	7.1410	7.1044	0.2536	2.3398	N.A.
Zhang et al. [31]	DEk	0.2444	6.2175	8.2915	0.2444	2.3810	24,000
Gandomi et al. [32]	FA	0.2015	3.562	9.0414	0.2057	1.7312	50,000
Present study	**CCCS**	**0.3312**	**10.0000**	**2.4095**	**0.3456**	**2.1730**	**25,000**

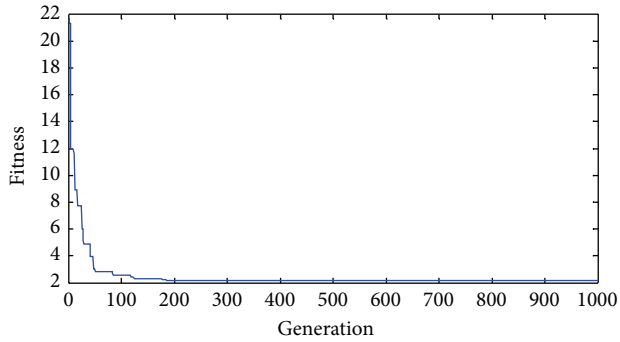


Figure 4: Convergence curve of Case I.

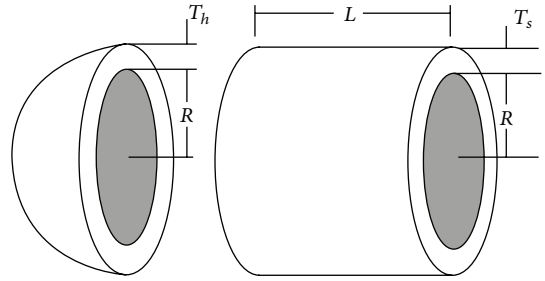


Figure 5: Schematic of the pressure vessel design problem.

5.3. Implementation in Structural Optimization Problems

Case I (welded beam design). The objective function and parameters of Case I are refers to in [32]. With 25 nests, CCCS found the global optimum requiring 1000 iterations per optimization run. Table 5 compares the optimization results found by CCCS with similar data reported in the literature. CCCS obtained the best design overall of 2.1730. Mahdavi et al. [25] and Fesanghary et al. [26] found a better desgin. "But for the continuous optimization problem equal to 1.7248" is deleted. In addition, CCCS requires only 25,000 function evaluations to complete the optimization process, hence much less than the literature. Convergence curve of Case I as shown in Figure 4.

Case II (pressure vessel design). The objective function and parameters of the Case II are refers to in [33]. With 25 nests, CCCS found the global optimum requiring 1000 iterations per optimization run. Table 6 compares the optimization results found by CCCS with similar data reported in the literature. CCCS obtained the best design overall of 5885.3. In addition, CCCS requires only 25,000 function evaluations to complete the optimization process, hence much less than the literature. Here, N/A represents no records in the literature. Convergence curve of Case II is shown in Figure 6.

6. Conclusions

Taking inspiration from the OEA, a new numerical optimization algorithm, CCCS, has been proposed in this paper. The experimental results in Tables 1–6 indicate that CCCS

Table 6: Pressure vessel design: comparison of CCCS results with the literature.

Reference method	R	L	T_s	T_h	$f_{\min}$	$\text{iter}_{\max}$
CCCS	**40.3196**	**200.0000**	**0.7782**	**0.3846**	**5885.3**	**1000**
ALPSO [33]	41.35	200	0.798	0.395	6234	7590
PSOA [34]	N/A	N/A	N/A	N/A	6292	6506
PSOSTR [34]	N/A	N/A	N/A	N/A	6272	3723
He et al. [21]	N/A	N/A	N/A	N/A	6290	30000
Akhtar et al. [28]	N/A	N/A	N/A	N/A	6335	12630

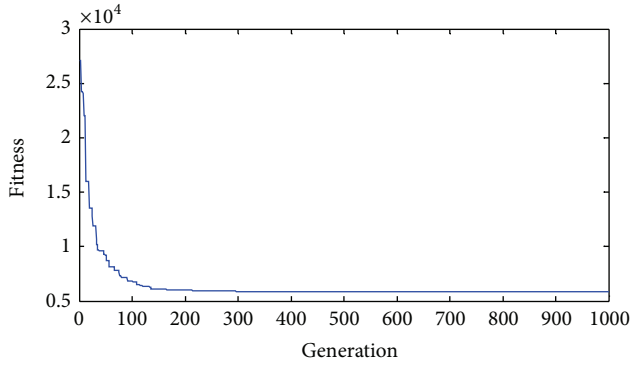


Figure 6: Convergence curve of Case II.

outperforms the MECA and OEA. On the whole, CCCS obtains a good performance for the unconstrained functions, constrained functions, and 2 engineering design problems. These benefits mainly from the following three aspects. One is the dynamics population, and the other is three evolutionary operators. The third aspect is a combination of CS algorithm. 28 experiments illustrate that CCCS has an effective searching mechanism. However, the number of dynamic populations, is difficult to control, which spends lots of computational cost. How to control the number of dynamic population is the future research work.

Acknowledgments

This work is supported by the National Science Foundation of China under Grant no. 61165015, Key Project of Guangxi Science Foundation under Grant no. 2012GXNSFDA053028, Key Project of Guangxi High School Science Foundation under Grant no. 20121ZD008, Open Research Fund Program of Key Lab of Intelligent Perception and Image Understanding of Ministry of Education of China under Grant no. IPIU01201100 and the Innovation Project of Guangxi Graduate Education under Grant no. YCSZ2012063.

References

[1] X. Yao, Y. Liu, and G. M. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[2] Y. W. Leung and Y. P. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 41–53, 2001.

[3] L. Zhang and B. Zhang, "Good point set based genetic algorithm," *Chinese Journal of Computers*, vol. 24, no. 9, pp. 917–922, 2001.

[4] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceeding of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, Coimbatore, India, December 2009.

[5] X. S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[6] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature—PPSN III*, Y. Davidor, H. P. Schwefel, and R. Mnner, Eds., vol. 866 of *Lecture Notes in Computer Science*, pp. 249–257, Springer, Berlin, Germany, 1994.

[7] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[8] J. Liu, W. C. Zhong, and L. C. Jiao, "An organizational evolutionary algorithm for numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 37, no. 4, pp. 1052–1064, 2007.

[9] C. Mu, L. Jiao, and Y. Liu, "M-elite coevolutionary algorithm for numerical optimization," *Ruan Jian Xue Bao/Journal of Software*, vol. 20, no. 11, pp. 2925–2938, 2009.

[10] I. Fister, I. Fister Jr., J. Brest, and V. Žumer, "Memetic artificial bee colony algorithm for large-scale global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–8, Brisbane, Australia, 2012.

[11] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, no. 2, pp. 113–127, 2000.

[12] J. P. B. Leite and B. H. V. Topping, "Improved genetic operators for structural engineering optimization," *Advances in Engineering Software*, vol. 29, no. 7–9, pp. 529–562, 1998.

[13] K. Deb, "Optimal design of a welded beam via genetic algorithms," *AIAA Journal*, vol. 29, no. 11, pp. 2013–2015, 1991.

[14] A. C. C. Lemonge and H. J. C. Barbosa, "An adaptive penalty scheme for genetic algorithms in structural optimization," *International Journal for Numerical Methods in Engineering*, vol. 59, no. 5, pp. 703–736, 2004.

[15] H. S. Bernardino, H. J. C. Barbosa, and A. C. C. Lemonge, "A hybrid genetic algorithm for constrained optimization problems in mechanical engineering," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 646–653, Singapore, September 2007.

[16] M. M. Atiqullah and S. S. Rao, "Simulated annealing and parallel processing: an implementation for constrained global design

optimization," *Engineering Optimization*, vol. 32, no. 5, pp. 659–685, 2000.

[17] A. R. Hedar and M. Fukushima, "Derivative-free filter simulated annealing method for constrained continuous global optimization," *Journal of Global Optimization*, vol. 35, no. 4, pp. 521–649, 2006.

[18] J. L. Liu, "Novel orthogonal simulated annealing with fractional factorial analysis to solve global optimization problems," *Engineering Optimization*, vol. 37, no. 5, pp. 499–519, 2005.

[19] S. F. Hwang and R. S. He, "A hybrid real-parameter genetic algorithm for function optimization," *Advanced Engineering Informatics*, vol. 20, no. 1, pp. 7–21, 2006.

[20] K. E. Parsopoulos and M. N. Vrahatis, "Unified particle swarm optimization for solving constrained engineering optimization problems," in *Advances in Natural Computation*, vol. 3612 of *Lecture Notes in Computer Science*, pp. 582–591, 2005.

[21] S. He, E. Prempain, and Q. H. Wu, "An improved particle swarm optimizer for mechanical design optimization problems," *Engineering Optimization*, vol. 36, no. 5, pp. 585–605, 2004.

[22] J. Zhang, C. Liang, Y. Huang, J. Wu, and S. Yang, "An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization," *Applied Mathematics and Computation*, vol. 211, no. 2, pp. 392–416, 2009.

[23] C. A. C. Coello, "Constraint-handling using an evolutionary multiobjective optimization technique," *Civil Engineering and Environmental Systems*, vol. 17, no. 4, pp. 319–346, 2000.

[24] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36–38, pp. 3902–3933, 2005.

[25] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[26] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, and Y. Alizadeh, "Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 33–40, pp. 3080–3091, 2008.

[27] J. N. Siddall, *Analytical Decision-Making in Engineering Design*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1972.

[28] S. Akhtar, K. Tai, and T. Ray, "A socio-behavioral simulation model for engineering design optimization," *Engineering Optimization*, vol. 34, no. 4, pp. 341–354, 2002.

[29] T. Ray and K. M. Liew, "Society and civilization: an optimization algorithm based on the simulation of social behavior," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 386–396, 2003.

[30] E. M. Montes and B. H. Oca, "Modified bacterial foraging optimization for engineering design," in *Proceedings of the Intelligent Engineering Systems through Artificial Neural Networks (ANNIE '2009)*, C. H. Dagli, K. M. Bryden, and S. M. Corns, Eds., vol. 19 of *ASME Press Series*, pp. 357–364.

[31] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Information Sciences*, vol. 178, no. 15, pp. 3043–3074, 2008.

[32] A. H. Gandomi, X. S. Yang, and A. H. Alavi, "Mixed variable structural optimization using Firefly algorithm," *Computers and Structures*, vol. 89, no. 23-24, pp. 2325–2336, 2011.

[33] Y. Yu, X. Yu, and Y. Li, "Solving engineering optimization problem by Augmented Lagrange particle swarm optimization," *Journal of Mechanical Engineering*, vol. 45, no. 12, pp. 167–172, 2009.

[34] G. G. Dimopoulos, "Mixed-variable engineering optimization based on evolutionary and social metaphors," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 4–6, pp. 803–817, 2007.

[35] X. S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, John Wiley & Sons, New York, NY, USA, 2010.

[36] Y. W. Leung and Y. P. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 41–53, 2001.

[37] 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.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 742653, 8 pages
http://dx.doi.org/10.1155/2013/742653

Research Article

A Hybrid Metaheuristic for Multiple Runways Aircraft Landing Problem Based on Bat Algorithm

Jian Xie,[1] Yongquan Zhou,[1,2] and Hongqing Zheng[1]

[1] *College of Information Science and Engineering, Guangxi University for Nationalities, Nanning, Guangxi 530006, China*
[2] *Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning, Guangxi 530006, China*

Correspondence should be addressed to Yongquan Zhou; yongquanzhou@126.com

Received 17 May 2013; Accepted 11 July 2013

Academic Editor: Xin-She Yang

The aircraft landing problem (ALP) is an NP-hard problem; the aim of ALP is to minimize the total cost of landing deviation from predefined target time under the condition of safe landing. In this paper, the multiple runways case of the static ALP is considered and a hybrid metaheuristic based on bat algorithm is presented to solve it. Moreover, four types of landing time assignment strategies are applied to allocate the scheduling time, and a constructed initialization is used to speed up the convergence rate. The computational results show that the proposed algorithm can obtain the high-quality and comparable solutions for instances up to 500 aircrafts, and also it is capable of finding the optimal solutions for many instances in a short time.

1. Introduction

Airport runway scheduling optimization is an ongoing challenge for air traffic controllers. The increasing demand is a challenge, which leads to many airports and routes are congested. Although investment in infrastructure may increase capacity at airports, it is an effective solution to improve planning and scheduling on current infrastructure. The aircraft landing Problem (ALP) is a kind of typical NP-hard problem in airport runway scheduling optimization [1]. The ALP consists of determining an optimal schedule of landing aircrafts on runways and assigning the landing time of each arriving aircraft. The objective is to minimize the total cost of landing deviation from predefined target time under the condition of safe landing. A predefined time window and separation time requirements with other aircraft must meet.

The first-come first-served (FCFS) is one of the most common solving methods for ALP, especially, in the single runway situation. A detailed review of published work addressing the ALP can be found in [1–3]. Generally speaking, for ALP, those methods can be broadly classified into three categories: exact methods (e.g., dynamic programming [4], branch-and-bound [1], and branch-and-price [5]), queueing theory [6], and heuristic or metaheuristic. Metaheuristic includes genetic algorithms [7], ant colony optimization [8], simulated annealing [9], scatter search and bionomic algorithms [2], cellular automata optimization [10], and other hybrid metaheuristics [9, 11]. Briskorn and Stolletz resent integer programming models an aircraft landing problems with aircraft classes [12].

Recently, more and more metaheuristics inspired by nature or social phenomenon are proposed and these algorithms are increasingly applied to different fields. The bat algorithm (BA) is one of the most popular algorithms, which is inspired by the intelligent echolocation behavior of microbats when they are foraging [13]. Many researchers applied BA to solve various optimization problems. For example, Gandomi et al. focus on solving constrained optimization tasks [14]. Yang and Gandomi apply BA to solve many global engineering optimizations [15]. Mishra at al. use BA to update the weights of a functional link artificial neural network classifier, a model proposed for classification [16]. Meanwhile, some researchers have improved BA and applied it to various optimization problems. Xie et al. proposed a bat algorithm based on differential operator and the Lévy flights trajectory (DLBA) to solve function optimization and nonlinear equations [17]. Wang et al. proposed a new bat algorithm with mutation (BAM) to solve the uninhabited

combat air vehicle (UCAV) path planning problem [18]. In this paper, the multiple runways case of the static ALP is considered; a hybrid metaheuristic based on bat algorithm (HBA, for short) is presented to solve it.

The rest of this paper is organized as follows. Section 2 presents the mathematical model of ALP and original bat algorithm. Hybrid metaheuristic based on bat algorithm is proposed in detail to solve multiple runways aircraft landing problem in Section 3. The experimental results of the HBA and comparisons with other previous algorithms are shown in Section 4. In the last section, we conclude this paper and point out some future work.

2. Problem Definitions and Bat Algorithm

2.1. The Mathematical Formulation of ALP. The ALP aims at finding the best arrangement of sequences, runway, and corresponding landing time for a given set of landing aircraft to minimize total cost by following separation requirements. To illustrate the mathematical formulation, some notations and decision variables are defined as follows.

Notations:

n = the number of planes;

m = the number of runways;

S_{ij} = the separation time (≥ 0) between plane i landing and plane j landing, (where the planes i and j land on the same runways), $i \neq j \in \{1, 2, \ldots, n\}$;

s_{ij} = the separation time ($0 \leq s_{ij} \leq S_{ij}$) between plane i landing and plane j landing, (where the planes i and j land on the different runways), $i \neq j \in \{1, 2, \ldots, n\}$;

T_i = the target landing time (target time) of plane i, $i \in \{1, 2, \ldots, n\}$;

E_i = the earliest landing time of plane i, $i \in \{1, 2, \ldots, n\}$;

L_i = the latest landing time of plane i, $i \in \{1, 2, \ldots, n\}$;

$\vec{c}_i$ = the cost of late landing of plane i, $i \in \{1, 2, \ldots, n\}$;

$\overleftarrow{c}_i$ = the cost of early landing of plane i, $i \in \{1, 2, \ldots, n\}$.

Decision variables:

t_i = the scheduled landing time of plane $i, i \in \{1, 2, \ldots, n\}$;

$$y_{ij} = \begin{cases} 1, & \text{if plane } i \text{ lands before} \\ & \text{plane } j \qquad i \neq j \in \{1, 2, \ldots, n\}; \\ 0, & \text{otherwise,} \end{cases}$$

$$\gamma_{ir} = \begin{cases} 1, & \text{if plane } i \text{ lands} \\ & \text{on runway } r \qquad i \in \{1, \ldots, n\}, r \in \{1, \ldots, m\}; \\ 0, & \text{otherwise,} \end{cases}$$

$$\delta_{ij} = \begin{cases} 1, & \text{if planes } i \text{ and } j \\ & \text{land on the same runway} \qquad i \neq j \in \{1, 2, \ldots, n\}; \\ 0, & \text{otherwise,} \end{cases}$$

$\vec{\tau}_i$ = the delay of landing plane i
(landing after the target time);
thus $\vec{\tau}_i = \max(0, t_i - T_i)$;

$\overleftarrow{\tau}_i$ = the earliness of landing plane i
(landing before the target time);
thus $\overleftarrow{\tau}_i = \max(0, T_i - t_i)$. (1)

The mathematical formulation of a mixed-integer programming for this problem is as follows [1, 2, 9]:

$$\min \quad Z = \sum_{i=1}^{n} \left(\vec{\tau}_i \vec{c}_i + \overleftarrow{\tau}_i \overleftarrow{c}_i \right) \tag{2}$$

$$\text{s.t.} \quad E_i \leq t_i \leq T_i, \; i = 1, 2, \ldots n; \tag{3}$$

$$\gamma_{ij} + \gamma_{ji} = 1, \; \forall i, j \in \{1, 2, \ldots, n\}, i \neq j; \tag{4}$$

$$t_j \geq t_i + S_{ij}\delta_{ij} + s_{ij}\left(1 - \delta_{ij}\right) - M\delta_{ij};$$
$$M = \left(L_i + \max\left(S_{ij}, s_{ij}\right) - E_j\right), \; \forall i \neq j \in \{1, 2, \ldots, n\}; \tag{5}$$

$$\sum_{r=1}^{m} \gamma_{ir} = 1, \; \forall i \in \{1, 2, \ldots, n\}; \tag{6}$$

$$\delta_{ij} \geq \gamma_{ir} + \gamma_{jr} - 1, i \neq j \in \{1, 2, \ldots, n\}, \; r \in \{1, 2, \ldots, m\}; \tag{7}$$

$$t_i - T_i = \vec{\tau}_i - \overleftarrow{\tau}_i, \; i = 1, 2, \ldots, n;$$
$$0 \leq \vec{\tau}_i \leq T_i - E_i, \; \vec{\tau}_i \geq T_i - t_i, \; i = 1, 2, \ldots, n; \tag{8}$$
$$0 \leq \overleftarrow{\tau}_i \leq L_i - T_i, \; \overleftarrow{\tau}_i \geq t_i - T, \; i = 1, 2, \ldots, n;$$

$$y_{ij}, \gamma_{ir}, \delta_{ij} \in \{0, 1\}, i \neq j \in \{1, 2, \ldots, n\}, \; r \in \{1, 2, \ldots, m\};$$
$$t_i, \vec{\tau}_i, \overleftarrow{\tau}_i \geq 0, \; i = 1, 2, \ldots n. \tag{9}$$

Objective function (2) minimizes the total cost of landing deviation from target time. The constraints (3) ensure that the scheduled landing time of each aircraft lies within its time window; constraints (4) consider the landing order; either aircraft i or j must land first; separation constraints must be ensured by constraints (5), where the role of M is to ensure that the equation is redundant if j lands before i; the constraints (6) ensure that each aircraft should land on only one runway; when aircrafts i and j are assigned to land on the same runway, constraints (7) ensure that the runways assigned to aircrafts i and j are identical; additional constraints (8) are introduced in order to link $\vec{\tau}_i$ and $\overleftarrow{\tau}_i$ to the decision variable t_i; constraints (9) ensure that the decision variables y_{ij}, γ_{ir}, and δ_{ij} are binary and the decision variables t_i, $\vec{\tau}_i$, and $\overleftarrow{\tau}_i$ are nonnegative, respectively.

2.2. Bat Algorithm. The basic bat algorithm (BA) is a metaheuristic proposed by Yang in 2010 [13]. Under several ideal rules, the BA has the following steps.

Step 1. Initialize the bat population and other parameters, and these initial individuals are evaluated.

Step 2. Each bat individual randomly selects a certain frequency f_i of sonic pulse, and the position of bat individual is updated according to their selected frequency. The formulas are as follows:

$$f_i = f_{\min} + \left(f_{\max} - f_{\min}\right)\beta,$$
$$v_i^t = v_i^{t-1} + \left(x_i^t - x_*\right) f_i, \tag{10}$$
$$x_i^t = x_i^{t-1} + v_i^t,$$

where x_i^t and v_i^t represent the positions and velocities of individual i in a D-dimensional search space at generation t, $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution, and x_* is the current global best location (solution) which is located after comparing all the solutions among all the n bats. Meanwhile, these new individuals are evaluated.

Step 3. If a random number is greater than its pulse emission rate R, then a new position is generated around the current global best position for each individual, which is the equal to local search. The updating formula of local search adopts $x = x_* + \varepsilon \times L_t$, where $\varepsilon \in [-1, 1]$ is a random number and $L_t = \langle L_i^t \rangle$ is the average loudness of all the bats at current generation t.

Step 4. If the local search is effective and its loudness L is greater than a random number, then the new position is accepted and its pulse emission rate R and loudness L are updated, where pulse emission rate is increased and loudness is decreased. L_i and R_i are updated by $L_i^{t+1} = \alpha \times L_i^t$, $R_i^{t+1} = R_i^0 \times [1 - \exp(-\gamma \times t)]$, where α, γ are constants.

Step 5. If the termination criterion is met, then the algorithm stops; otherwise repeat algorithm (go to Step 2).

In general, the bat algorithm has three procedures, position updating, local search, and decreasing the probability of local search. For the details of BA refer to [13].

3. Hybrid Metaheuristic Based on Bat Algorithm for ALP

Basic bat algorithm is a continuous optimization algorithm, which is successfully applied to solve real optimization problem [15, 17]. However, the standard continuous encoding scheme of BA cannot be used to solve ALP directly. Therefore, in order to solve aircraft landing problem effectively, HBA is proposed.

3.1. Solution Representation in HBA. In order to apply BA to ALP, the first step is to devise a suitable representation of the candidate solutions for this particular problem. Each individual is a sequence S ($S = (s_1, s_2, \ldots, s_n)$) with integer number s_i ($s_i \in \{1, \ldots, m\}$), where the integer number represents the runway and the length of this sequence S is the number of aircrafts. For example, if we have three runways and ten aircrafts, the coded individual with integer is $1 \to 3 \to 2 \to 2 \to 3 \to 2 \to 1 \to 2 \to 1 \to 3$ Figure 1 shows that the aircrafts 1, 7, and 9 land on runway number 1, the aircrafts 3, 4, 6 and 8 land on runway number 2, the aircrafts 2, 5, and 10 land on runway number 3.

Aircraft:	1	2	3	4	5	6	7	8	9	10
Runway:	1	3	2	2	3	2	1	2	1	3

Figure 1: The representation of the candidate solutions.

3.2. Landing Time Assignment. The assigned landing time (ALT) of each aircraft is the very important step in the ALP; the purpose is to reduce the total cost of penalty caused by all aircrafts. In this paper, the landing time is assigned based on the target landing time T_i assigned to each aircraft. There are four types of assignment strategies carried out in the proposed algorithm: forward assignment strategy (FAS), backward assignment strategy (BAS), random forward assignment strategy (RFAS), and random backward assignment strategy (RBAS). For each kind of assignment strategy, firstly, all aircrafts are found out on each runway, and the target landing time T of these aircrafts is sorted in ascending order, namely, $T_{i_1} \le T_{i_2} \le \cdots \le T_{i_{p-1}} \le T_{i_p}$, where p is the number of aircrafts on this runway.

(i) FAS: for each two sorted aircrafts i_k and i_{k+1} on a runway, the ALT is $\vec{t}_{i_k}$ and $\vec{t}_{i_{k+1}}$, separately. If $T_{i_{k+1}} < \vec{t}_{i_k} + S_{i_k i_{k+1}}$; then $\vec{t}_{i_{k+1}} = \vec{t}_{i_k} + S_{i_k i_{k+1}}$, otherwise $\vec{t}_{i_{k+1}} = T_{i_{k+1}}$.

(ii) BAS: for each two sorted aircrafts i_k and i_{k+1} on a runway, the ALT is $\overleftarrow{t}_{i_k}$ and $\overleftarrow{t}_{i_{k+1}}$, separately. If $T_{i_k} > \overleftarrow{t}_{i_{k+1}} - S_{i_k i_{k+1}}$; then $\overleftarrow{t}_{i_k} = \overleftarrow{t}_{i_{k+1}} - S_{i_k i_{k+1}}$, otherwise $\overleftarrow{t}_{i_k} = T_{i_k}$.

(iii) RFAS: for each two sorted aircrafts i_k and i_{k+1} on a runway, the ALT is $\widehat{t}_{i_k}$ and $\widehat{t}_{i_{k+1}}$, separately. $\widehat{t}_{i_k}$ is rounded down to the nearest integer between $\overleftarrow{t}_{i_k}$ and $\vec{t}_{i_k}$. If $\widehat{t}_{i_{k+1}} < \widehat{t}_{i_k} + S_{i_k i_{k+1}}$, then $\widehat{t}_{i_{k+1}} = \widehat{t}_{i_k} + S_{i_k i_{k+1}}$.

(iv) BFAS: for each two sorted aircrafts i_k and i_{k+1} on a runway, the ALT is $\breve{t}_{i_k}$ and $\breve{t}_{i_{k+1}}$, separately. $\breve{t}_{i_k}$ is rounded down to the nearest integer between $\overleftarrow{t}_{i_k}$ and $\vec{t}_{i_k}$. If $\breve{t}_{i_k} < \breve{t}_{i_{k+1}} - S_{i_k i_{k+1}}$, then $\breve{t}_{i_k} = \breve{t}_{i_{k+1}} - S_{i_k i_{k+1}}$.

The assigned landing time is effective just while the all constraints (3)–(9) are satisfied, and the four kinds of total cost are compared; the best total cost is used as the objective function value.

3.3. Initialization Construction. The initialization of a population is usually performed by randomly selecting a runway from the available runways for each aircraft. However, the target landing time of aircraft is ordered, and the time window must ensure considering the safety. So, a preprocessing is performed to improve the performance of HBA in the initialization construction.

In initialization, *ps* individuals are generated, where *ps* is population size. For each individual, all target landing time T_i $(i = 1, 2, \ldots, n)$ is sorted in an ascending order; namely, $T_{i_1} \le T_{i_2} \le \cdots \le T_{i_{n-1}} \le T_{i_n}$. The first aircraft i_1 lands on a runway randomly, for each two sorted aircrafts i_k and i_{k+1}, if $T_{i_{k+1}} < T_{i_1} + S_{i_k i_{k+1}}$, and then aircraft i_{k+1} is allocated to other runway, namely, aircrafts i_k and i_{k+1} on different runways. Otherwise, the aircraft i_{k+1} is allocated to the runway that aircraft i_k lands on. The initialization repeats until the *ps* individuals are generated.

According to the experiments performed, this initialization construction can develop an initial schedule of landing aircrafts which has a very good quality. However, to derive near-optimal solutions, this initial schedule should be improved using the bat algorithm provided in Section 3.4.

3.4. Hybrid Bat Algorithm. In original bat algorithm framework, the idea is that, firstly, the bat individual randomly selects a certain frequency of sonic pulse, and the position of bat individual is updated according to its selected frequency; secondly, if a random number is greater than its pulse emission rate R, then a new position is generated around the current global best position for each individual, which is equal to local search; at last, if the local search is effective and its loudness L is greater than a random number, then the new position is accepted, and its pulse emission rate R and loudness L are updated, where pulse emission rate is increased and loudness is decreased. In general, the bat algorithm has three procedures: position updating, local search, and decreasing the probability of local search.

In this paper, the frequency f is a runway, $f \in \{1, \ldots, m\}$. The position updating is different from continuous bat algorithm. The position updating is used to assign the landing sequence, which is performed as follows:

(i) select a frequency f randomly, namely; select a runway randomly;

(ii) select an aircraft randomly on selected runway; then assign this aircraft to other runway.

There is an example used to illustrate the procedure in Figure 2. If the frequency $f = 2$ and the second aircraft is selected, then, this aircraft is assigned to runway number 3.

For the local search part, this procedure is controlled by pulse emission rate R. The R is equivalent to the probability of performing local search, and the R is updated by

$$R(t) = \left(1 + \exp\left(-\frac{5}{t_{\max}} \times \left(t - \frac{t_{\max}}{2}\right)\right)\right)^{-1}, \quad (11)$$

where t denotes the tth generation $t_{\max}$ is the maximal generation. The rate R is similar to sigmoid function. The purpose of the local search is to enhance the solution generated, and the operation is performed on the current global best individual in bat algorithm. In this paper, two types of local search are presented: the swap and loop subsequence inserting (LSI).

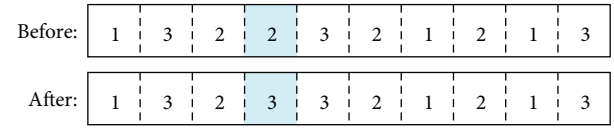


Figure 2: An individual with ten aircrafts before and after position updating.

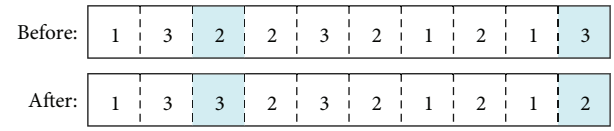


Figure 3: An individual with ten aircrafts before and after swap local search.

The purpose of swap is mildly mutating the current global best individual, so that an improved solution can be found out around the current optimal solution. The swap is illustrated in Figure 3, where the third and tenth components are randomly chosen to be exchanged; note that the two selected aircrafts land on different runways. Thus, the value of the third component is changed from 2 to 3, while the value of the tenth component is switched from 3 to 2.

The LSI is a variant of inserting operation; the purpose is to mutate the current global best individual in a large extent, so that the diversity of population can be ensured. It can effectively avoid prematurity and greatly improve efficiency of global search. The LSI is illustrated in Figure 4; a start point is randomly chosen (ninth component) and a random length of subsequence is determined (5), so the sub-sequence can be determined $(1 \to 3 \to 1 \to 3)$. A random insert point is chosen in remainder sub-sequence (third, fifth components); then the selected sub-sequence is inserted into remainder sub-sequence before insert point.

The local search systematically explores different neighborhood structures. The swap and LSI are preformed according to the pulse emission rate R. In other words, if a random number is greater than the R, the swap is performed; otherwise, the LSI is performed.

The loudness L_i of bat individual i determines the accepted probability a solution generated by local search in original bat algorithm. Meanwhile, it also dominates the updating of pulse emission rate R and loudness L in continuous bat algorithm. However, in this paper, a runway balance (RB) operation is performed according to the value of loudness L for each individual.

The intention of RB operation is to balance the load of each runway. Firstly, the aircrafts are counted on each runway; an aircraft selected randomly on runway with maximum aircrafts is assigned to a runway with minimum aircrafts. If the amount of aircrafts on runway (maximum aircrafts or minimum aircrafts) is equal, then a runway is selected

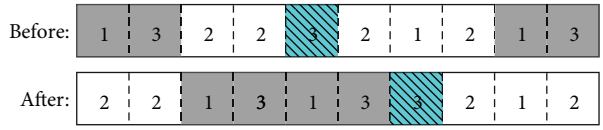


Figure 4: An individual with ten aircrafts before and after LSI local search.

randomly. This process is stopped until the difference of aircrafts amount on each runway is not more than one.

The loudness L_i is updated by (12)

$$L_i^{t+1} = \alpha \times L_i^t, \tag{12}$$

where α is a constant and initial value of $L_i^1 \in (1, 2)$. If a random number is less than its loudness L_i^t, then the RB operation is performed; otherwise, each aircraft is assigned to a random runway.

The iterative process is repeated until the termination criterion is met; Algorithm 1 shows the pseudo-code of HBA based on the framework of bat algorithm for ALP. The lines 1–3, the bat population and other parameters are initialized (the initialization of population uses a method described in Section 3.3), and these initial individuals are evaluated (Section 3.2 detailedly described this procedure). In lines 5–9, the individuals are updated by selecting frequency (this part corresponds to the position updating of BA; this method is described in Section 3.4), and these new individuals are evaluated. Lines 10–15 show the local search and these solutions generated by local search are evaluated. In lines 16–24, the runway balance operation is carried out. Lines 25-26 are the judgment of termination criterion and the output of results.

4. Simulation Results and Comparisons

The simulation experiment is extensively investigated by a large number of benchmark instances; these well-studied problems are taken from the web OR-Library (last update: June 2012, http://people.brunel.ac.uk/~mastjjb/jeb/info.html), a reference site which contains detailed information regarding a large number of benchmark instances. In this paper, the whole 13 instances from OR-Library are selected; these instances have been widely used as benchmarks to certify the performance of algorithms by many researchers [2, 9].

All computational experiments are conducted with MATLAB 2012a on a 3.0 GHz Athlon PC with 2.0 GB memory. There are two kinds of termination criterion for different instances. For instances where the value of optimal (exact) solution (V_{opt}) is known, the algorithm is repeated until the objective function value is equal to the V_{opt}; if the objective function value is greater than the V_{opt} when the maximum generation $t_{\max}$(= 200) is met, then the algorithm also is terminated. For instances where the value of V_{opt} is not known, the termination criterion is set as maximum generation $t_{\max}$ = 1000.

4.1. Sensitivity Analysis. Since performance is affected by the settings of the parameter values used in metaheuristics, a sensitivity analysis is conducted to examine the effect of different parameter values on the proposed HBA. Two parameters, that is, population size *ps* and parameter α, are used to investigate the performance with respect to different values, which are based on the average results obtained from instances involving from 10 to 50 aircrafts, and each instance run ten times. The principal figures of merit for comparison of different parameter values are the average percentage gap G_{avg} (%) associated with the best solution found and the average computational time CT_{avg} in seconds for exact solution is reached. The percentage gap G(%) is measurement criteria referenced [2]. Figures 5 and 6 show the statistical result, where G_{avg} is shown on the left y-axis and CT_{avg} is shown on the right y-axis and different parameter values (*ps* and α) are listed in on the horizontal x-axis.

The performance of the HBA in terms of different population numbers is shown in Figure 5. It shows that better results can be obtained with a larger population. However, it does not improve significantly when the value of *ps* is equal to or greater than 10. On the other hand, the computational time increases steadily when the population rises.

Figure 6 illustrates the performance effect of the HBA with a decreasing α value. It indicates that the average solution quality deteriorates when the value of α decreases gradually. In some cases, increasing the value above 0.9 may actually worsen the average objective value. This is probably because the search is extremely random and RB operation is carried out rarely.

4.2. Comparisons of Results. Based on the results of the sensitivity analysis, the parameter values are set in the proposed HBA (i.e., *ps*= 10, α= 0.9) for comparison with other algorithms for solving the ALP with multiple runways that were proposed and tested to be valid in previous studies. The computational results obtained are shown in Tables 1 and 2. In Tables 1 and 2, for each problem, the instance (Ins); the number of aircrafts (n); the number of runways (m); the value of optimal solution (V_{opt}); the value of the best-known solution (V_{best}) if the optimal solution is not known; the objective function value (O) obtained and the percentage gap (G%) associated with the best solution found over the 15 replications and the average execution time (T) in seconds for 15 replications. Note that in order to compare the results, the s_{ij} is set zero in accordance with the previous literature, where G is calculated on the basis of V_{opt} or V_{best}; $G = 100 * (O - V_{\text{opt}})/V_{\text{opt}}$. One complication is that for some problem where V_{opt} or V_{best} is zero, the percentage gap is defined as zero if and only if the best solution found as zero is also zero, otherwise it is undefined (nd).

Table 1 presents results for a set of instances involving from 10 to 50 aircrafts. SS is scatter search, the bionomic algorithm is marked as BA1 [2], and IACA is improved ant colony algorithm [8]; heuristic is an effective heuristic algorithm in [1], FCFS is first-come first-served, and the result is reference [2]. According to Table 1, the percentage gap G of SS and BA1 is better than HBA; however, the average

```
(1) Initialize the ps, t = 1, bat population and other parameters;
(2) Construct initial bat population;     // (3.3 Initialization construction)
(3) Assign landing time and evaluate each individual;     // (3.2 Landing time assignment)
(4) repeat
(5)     for i = 1:ps do
(6)         Determine frequency f;
(7)         Update each bat individual;
(8)     end
(9)     Assign landing time and evaluate each individual;     // (3.2 Landing time assignment)
(10)    if rand > R(t) then
(11)        Carry out swap local search operation;
(12)    else
(13)        Carry out LSI local search operation;
(14)    end
(15)    Assign landing time and evaluate each individual;     // (3.2 Landing time assignment)
(16)    Compute loudness of each individual by (12);
(17)    for i = 1:ps do
(18)        if rand < L_i then
(19)            Carry out RB operation;     // Runway balance operation
(20)        else
(21)            Assign each aircraft to a random runway;
(22)        end
(23)    end
(24)    Assign landing time and evaluate each individual;     // (3.2 Landing time assignment)
(25) until t = t_max
(26) Output result and plot
```

Algorithm 1: The pseudo-code of HBA for ALP.

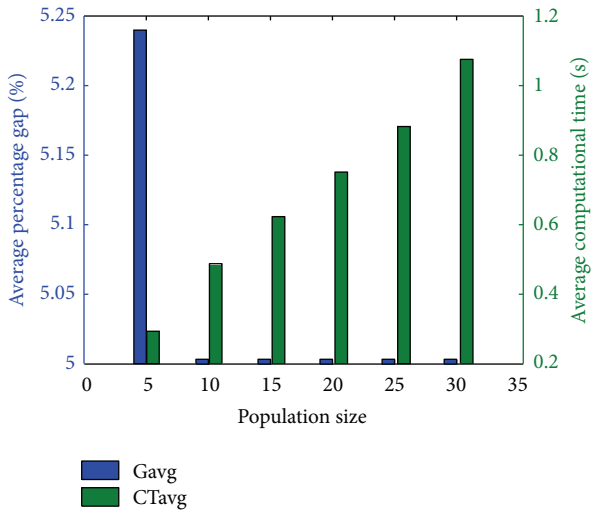


Figure 5: The effect of the value of population size *ps* with respect to relative computational time and average percentage gap.

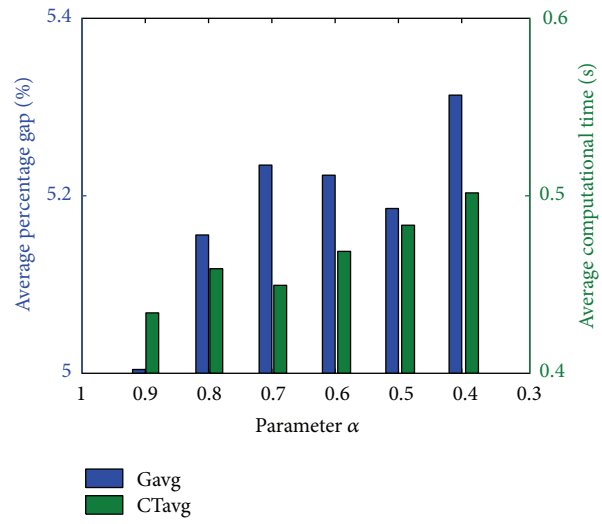


Figure 6: The effect of the value of α with respect to relative computational time and average percentage gap.

execution time T of SS and BA1 is longer than the T of HBA and the average execution time of HBA only expends 0.43 seconds; by contrast, the SS and BA1 expend 6.4 and 7.7 seconds, respectively. The comparison of results between HBA and heuristic shows the solutions obtained by HBA are superior to the solutions obtained by heuristic. Meanwhile, the average execution time is approximate between HBA and heuristic. Comparing with IACA and FCFS, it is evidently shown that the HBA is effective.

Table 2 presents results for a set of instances involving from 100 to 500 aircrafts. The termination criterion is set as maximum generation $t_{\max} = 1000$. The IACA and heuristic did not select these instances for testing. From Table 2, even for the larger instances, the HBA found that optimal solutions

Table 1: The comparisons of computational results for first group instances.

Ins	n	m	V_{opt}	HBA			SS		BA1		IACA		Heuristic		FCFS
				O	G	T	G	T	G	T	G	T	G	T	G
Airland1	10	2	90	90	0	0.08	0	2.4	0	4.5	0	0.1	0	0.1	0
		3	0	0	0	0.11	0	3.9	0	3.4	0	0.0	0	0.1	0
Airland2	15	2	210	210	0	0.09	0	4.5	0	4.9	0	0.6	0	0.1	9.52
		3	0	0	0	0.10	0	4.6	0	4.3	0	0.4	0	0.1	0
Airland3	20	2	60	60	0	0.09	0	4.8	0	5.8	0	11.3	0	0.1	116.67
		3	0	0	0	0.10	0	6.2	0	6.3	0	9.3	0	0.2	nd
Airland4	20	2	640	640	0	0.55	0	5.2	0	5.5	0	7.9	0	0.1	0
		3	130	130	0	0.14	0	4.6	0	5.7	0	6.4	0	0.1	0
		4	0	0	0	0.14	0	5.6	0	5.2	0	5.8	0	0.2	0
Airland5	20	2	650	890	36.92	1.44	0	5.0	3.08	6.1	12.31	2.4	64.62	0.1	16.92
		3	170	170	0	0.16	0	5.4	0	4.3	0	7.0	41.18	0.1	5.88
		4	0	0	0	0.21	0	5.6	0	6.8	0	3.2	0	0.2	nd
Airland6	30	2	554	636	14.80	1.61	0	7.0	3.61	10.1	51.08	3.8	59.21	0.1	59.21
		3	0	0	0	0.30	0	5.4	0	8.7	0	12.2	0	0.2	0
Airland7	44	2	0	0	0	0.09	0	11.8	0	12.4	0	55.2	0	0.2	0
Airland8	50	2	135	180	33.33	2.01	0	12.1	0	19.6	22.22	168.1	88.89	0.2	425.93
		3	0	0	0	0.16	0	13.9	0	18.1	nd	108.2	0	0.6	nd
Average					5.00	0.43	0	6.4	0.39	7.7	5.35	23.6	14.93	0.2	45.30

Table 2: The comparisons of computational results for second group instances.

Ins	n	m	V_{best}	HBA			SS		BA1		FCFS
				O	G	T	G	T	G	T	G
Airland9	100	2	452.92	499.49	10.28	16.0	5.67	24.3	54.73	48.7	172.60
		3	75.75	77.03	1.69	16.5	0	39.0	87.46	46.6	342.81
		4	0	0	0	16.6	0	33.6	nd	43.9	nd
Airland10	150	2	1288.73	1407.45	9.21	18.6	7.87	60.8	25.95	84.5	103.47
		3	220.79	224.13	1.51	21.0	8.88	66.8	195.88	80.3	552.16
		4	34.22	34.22	0	20.6	16.74	64.7	292.40	78.8	3473.49
		5	0	0	0	23.1	0	60.7	nd	76.2	nd
Airland11	200	2	1540.84	1673.95	8.64	23.2	9.19	95.9	38.54	128.7	129.80
		3	280.82	280.64	−0.06	26.1	21.59	102.1	290.09	120.3	764.25
		4	54.53	54.53	0	27.4	2.77	99.3	474.47	116.8	3947.88
		5	0	0	0	27.2	0	95.6	nd	115.8	nd
Airland12	250	2	1961.39	2482.26	26.56	28.3	18.80	126.6	50.18	183.5	137.42
		3	290.04	243.79	−15.95	31.5	17.48	145.4	198.01	171.0	903.97
		4	3.49	2.44	−30.09	33.3	271.63	144.5	13216.91	168.8	70752.44
		5	0	0	0	34.6	0	138.6	nd	166.2	nd
Airland13	500	2	5501.96	5184.06	−5.78	58.0	3.72	383.6	37.47	537.9	56.91
		3	1108.51	755.15	−31.88	60.7	1.98	456.0	182.69	515.8	462.60
		4	188.46	90.03	−52.23	63.7	22.98	441.3	1186.81	497.7	2027.94
		5	7.35	0	−100	65.9	0	442.1	22308.44	488.7	52628.71
Average					−9.37	34.0	21.54	159.0	2576.00	193.2	9097.10

in several cases are prominent. We can clearly find that the percentage deviation G from the best-known solutions is negative, which indicates the solutions found by HBA are better than the best-known solutions. The average percentage gap G of HBA is −9.37% for this group of instances; however, the average percentage gap G of SS, BA1, and FCFS is positive and is much greater than the G of HBA. On the other hand, the average execution time T expended by HBA is much lesser than the average execution time T of SS and BA1 expended.

Figure 7 illustrates the performance of the HBA, the SS, the BA1, the IACA, and the heuristic with respect to the computational time. HBA and heuristic perform almost

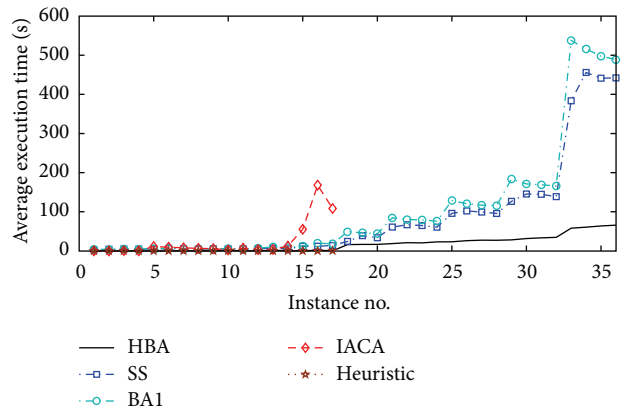


Figure 7: The computational time of each test case.

the same on instances up to 50 aircrafts. For the larger instances up to 500 aircrafts, the computational time of HBA is shortened observably, which also demonstrates that the HBA has a faster convergence rate.

5. Conclusions

In this paper, we considered the multiple runways aircraft landing problem with the objective of minimizing the total deviation of landing time from the target time. In order to solve the larger instances involving up to 500 aircraft and multiple runways, a hybrid metaheuristic based on bat algorithm (HBA, for short) has been implemented. The HBA includes a problem-dependent initialization construction, and several local search operations are integrated into the framework of bat algorithm. The computational results of the HBA show that the proposed algorithm is very effective and competitive and can obtain solutions with high quality for instances up to 500 aircrafts in a short time. Moreover, the landing time assignment of each aircraft is a key for solving ALP; several excellent assignment strategies need to be presented in our further work; meanwhile, the aircraft take-off problem (ATP) in airport runway scheduling problem also is our future work.

Acknowledgment

This work is supported by National Science Foundation of China under Grant no. 61165015, Key Project of Guangxi Science Foundation under Grant no. 2012GXNSFDA053028, Key Project of Guangxi High School Science Foundation under Grant no. 20121ZD008, the Funded by Open Research Fund Program of Key Lab of Intelligent Perception and Image Understanding of Ministry of Education of China under Grant no. IPIU01201100, and the Innovation Project of Guangxi Graduate Education under Grant no. YCSZ2012063.

References

[1] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha, and D. Abramson, "Scheduling aircraft landings—the static case," *Transportation Science*, vol. 34, no. 2, pp. 180–197, 2000.

[2] H. Pinol and J. E. Beasley, "Scatter search and bionomic algorithms for the aircraft landing problem," *European Journal of Operational Research*, vol. 171, no. 2, pp. 439–462, 2006.

[3] J. A. Bennell, M. Mesgarpour, and C. N. Potts, "Airport runway scheduling," *Annals of Operations Research*, vol. 204, pp. 249–270, 2013.

[4] A. R. Brentnall and R. C. H. Cheng, "Some effects of aircraft arrival sequence algorithms," *Journal of the Operational Research Society*, vol. 60, no. 7, pp. 962–972, 2009.

[5] M. Wen, *Algorithms of Scheduling Aircraft Landing Problem*, Technical University of Denmark, DTU, Lyngby, Denmark, 2005.

[6] N. Bäuerle, O. Engelhardt-Funke, and M. Kolonko, "On the waiting time of arriving aircrafts and the capacity of airports with one or two runways," *European Journal of Operational Research*, vol. 177, no. 2, pp. 1180–1196, 2006.

[7] X.-B. Hu and E. Di Paolo, "An efficient genetic algorithm with uniform crossover for air traffic control," *Computers and Operations Research*, vol. 36, no. 1, pp. 245–259, 2009.

[8] G. Bencheikh, J. Boukachour, and A. E. H. Alaoui, "Improved ant colony algorithm to solve the aircraft landing problem," *International Journal of Computer Theory and Engineering*, vol. 3, no. 2, pp. 224–233, 2011.

[9] A. Salehipour, M. Modarres, and N. L. Moslemi, "An efficient hybrid meta-heuristic for aircraft landing problem," *Computers & Operations Research*, vol. 40, no. 1, pp. 207–213, 2013.

[10] S.-P. Yu, X.-B. Cao, and J. Zhang, "A real-time schedule method for aircraft landing scheduling problem based on cellular automation," *Applied Soft Computing Journal*, vol. 11, no. 4, pp. 3485–3493, 2011.

[11] G. Hancerliogullari, G. Rabadi, A. H. Al-Salem, and M. Kharbeche, "Greedy algorithms and metaheuristics for a multiple runway combined arrival-departure aircraft sequencing problem," *Journal of Air Transport Management*, vol. 32, pp. 39–48, 2013.

[12] D. Briskorn and R. Stolletz, "Aircraft landing problems with aircraft classes," *Journal of Scheduling*, pp. 1–15, 2013.

[13] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO '10)*, pp. 65–74, Springer, Berlin, Germany, 2010.

[14] A. H. Gandomi, X. S. Yang, A. H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing and Applications*, vol. 22, no. 6, pp. 1239–1255, 2013.

[15] X. S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.

[16] S. Mishra, K. Shaw, and D. Mishra, "A new meta-heuristic bat inspired classification approach for microarray data," *Procedia Technology*, vol. 4, pp. 802–806, 2012.

[17] J. Xie, Y. Zhou, and H. Chen, "A novel bat algorithm based on differential operator and Lévy-flights trajectory," *Computational Intelligence and Neuroscience*, vol. 2013, Article ID 453812, 13 pages, 2013.

[18] G. Wang, L. Guo, H. Duan, L. Liu, and H. Wang, "A bat algorithm with mutation for UCAV path planning," *The Scientific World Journal*, vol. 2012, Article ID 418946, 15 pages, 2012.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 362619, 8 pages
http://dx.doi.org/10.1155/2013/362619

Research Article

Parameter Estimation of Photovoltaic Models via Cuckoo Search

Jieming Ma,[1,2] T. O. Ting,[2] Ka Lok Man,[2] Nan Zhang,[2] Sheng-Uei Guan,[2] and Prudence W. H. Wong[1]

[1] *Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool, L69 3BX, UK*
[2] *Xi'an Jiaotong-Liverpool University, 111 Ren'ai Road, Suzhou Dushu Lake, HET, Jiangsu Province 215123, China*

Correspondence should be addressed to T. O. Ting; toting@xjtlu.edu.cn

Received 27 June 2013; Accepted 11 July 2013

Academic Editor: Xin-She Yang

Since conventional methods are incapable of estimating the parameters of Photovoltaic (PV) models with high accuracy, bioinspired algorithms have attracted significant attention in the last decade. Cuckoo Search (CS) is invented based on the inspiration of brood parasitic behavior of some cuckoo species in combination with the Lévy flight behavior. In this paper, a CS-based parameter estimation method is proposed to extract the parameters of single-diode models for commercial PV generators. Simulation results and experimental data show that the CS algorithm is capable of obtaining all the parameters with extremely high accuracy, depicted by a low Root-Mean-Squared-Error (RMSE) value. The proposed method outperforms other algorithms applied in this study.

1. Introduction

Photovoltaic (PV) cells, normally assembled into modules or arrays on mounting systems, are capable of producing electrons when photons strike its surface. Taking the advantages of many promising features like renewability, less pollution, and ease of installation, PV generators are envisaged to be an important energy source for the future.

Due to the high initial cost of a PV-supplied system, predictive performance tools are widely used by engineers to optimize the system performance [1, 2]. PV manufacturers normally provide limited tabular data measured under the Standard Test Conditions (STCs), which correspond to a cell temperature of 25°C and an irradiance of 1000 W/m^2 at 1.5 air mass spectral distributions. As reported in [3], PV generators always operate under environments far from the STCs. Due to this reason, the data available in the datasheet usually fail to fulfill the engineering requirements.

PV model, with the ability to predict I-V characteristics of PV generators under an operating environment other than the STCs, is a predictive performance tool that allows consumers to maximize the cost effectiveness of the system before installation [2]. They are generally analytical equations based on a physical description that formulate PV generated current (I) with the most crucial technical characteristics and the environmental variables, such as the operating voltage (V), the ambient temperature (T), and the irradiance (G). Over the years, significant research efforts have been contributing to the development of the behavioral models [4–8]. Among numerous modeling approaches, the Single-Diode Model (SDM) is the most widely utilized PV model in the literature. A general SDM includes five parameters, namely, photocurrent (I_{pv}), saturation current (I_o), diode ideality constant (n), series resistance (R_s), and shunt resistance (R_p). In order to adapt PV model behavior to different operating conditions, de Blas et al. [9] suggested to apply the procedure described in the International Standard IEC 891 that relates current and voltage of the PV characteristics at given values of T and G with the corresponding values at different operating environments. The Improved Single Diode Model (ISDM) presented by De Soto et al. [5] includes the dependence of the PV parameters on operating conditions. The normal parameters at the STCs are necessary to be determined in this model. Both SDM and ISDM are adopted in this study of parameter estimation.

Analytical methods [5, 10–12] are common approaches in estimating the parameters by mathematical equations. Although having the merit of simplicity, it is hard to further reduce the errors of the estimated values. Furthermore, analytical methods utilize the I-V curve features or

semiconductor parameters that are unavailable in the datasheet. This problem often reduces its feasibility. Recently, PV parameter estimation is deemed as a multidimensional optimization problem. Several computational intelligence methods, such as Genetic Algorithms (GA) [13], Chaos Particle Swarm Optimization (CPSO) [14], Firefly [15], and Pattern Search (PS) [16], were proposed in the literature. These algorithms usually extract relevant parameters by minimizing the Root Mean Square Error (RMSE) as the objective function in the optimization process. Askarzadeh and Rezazadeh [17] reported that the optimization methods produce better results than analytical methods.

Cuckoo Search (CS) is a nature-inspired optimization algorithm based on the fascinating breeding behavior such as brood parasitism of certain species of cuckoos. In [18, 19], Yang and Deb reported that the CS algorithm outperforms Particle Swarm Optimization (PSO) and GA algorithms for various standard test functions. In this paper, a CS-based parameter estimation method for the SDM and ISDM is presented. Simulation and experimental results show superior accuracy and feasibility of the proposed parameter estimation method.

The rest of the paper is organized as follows. Section 2 explains both PV models (SDM and ISDM) used in this work. The objective function formulation is given in Section 3. This is followed by results and discussions in Section 4. The results comparison is also available here, and finally the conclusions are derived in Section 5.

2. PV Modeling

2.1. Single Diode PV Model (SDM). PV cells are made of a variety of semiconductor materials using different manufacturing processes. The working principle of PV cells is essentially on the basis of the PV effect, which refers to the generation of a potential difference at the *P-N* junction in response to visible or other radiation. When a PV cell is exposed to light, the semiconductor materials absorb photons, and accordingly charged carriers are generated. Potential difference and current in the external circuit lead to the separation of carriers in the internal electric field created by the *P-N* junction and collection at the electrodes. The photogenerated charge carriers can be subsequently captured in the form of an electric current, that is, electricity I_{pv}. Eliminated the PV effect, a PV cell behaves like a conventional diode that does not depend on any light parameters. The Shockley diode equation is generally used to describe the current flowing through the diode (I_d):

$$I_d = I_o\left(e^{V_d/nV_t} - 1\right). \tag{1}$$

In (1), I_o is the normal diode current, and V_d represents the electrical potential difference between the two ends of the diode. The ideality factor n is assumed to be independent of the environment variables T and G. V_t denotes the thermal voltage of the PV, and its value can be written as a function of T:

$$V_t = \frac{kT}{q}, \tag{2}$$

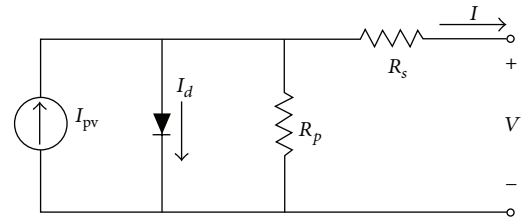


Figure 1: The equivalent circuit of the SDM.

where k and q represent the Boltzmann constant (1.380650×10^{-23} J/K) and the electron charge (1.602176×10^{-19} C), respectively.

SDM assumes that the superposition principle holds; that is, the total characteristic is the sum of the dark and illuminated characteristics [3–5]. As expressed in (3) below, the terminal current I is therefore equal to I_{pv} subtracting the current diverting through the diode and R_s. The equivalent circuit of the SDM is shown in Figure 1.

$$I = I_{\text{pv}} - I_o\left(e^{(V+IR_s)/nV_t} - 1\right) - \frac{V + IR_s}{R_p}. \tag{3}$$

PV module is a particular case of the PV cells connected in series. If the number of the connected cells is up to N_s, V_t is scaled to N_s times. The model equation is then rewritten as

$$I = I_{\text{pv}} - I_o\left(e^{(V+IR_s)/nN_sV_t} - 1\right) - \frac{V + IR_s}{R_p}. \tag{4}$$

In this sense, I_{pv}, I_o, R_s, and R_p are the corresponding parameters of a PV module.

2.2. Improved Single Diode Model (ISDM). The traditional SDM ignores the operating conditions effect on these parameters. However, some studies have shown that the parameters, such as I_{pv} and I_o, vary under different environmental conditions. These are due to changes of temperature T and irradiance G. Aiming to evaluate the PV behavior at environmental conditions other than the normal values T_n and G_n, the relations between the operating parameters and the normal parameters are studied by numerous researchers.

In [4], the value of light-generated I_{pv} is reported to be linearly dependent on the solar irradiation under the influence of temperature:

$$I_{\text{pv}} = \frac{G}{G_n}\left(I_{\text{pv}n} - K_i\Delta T\right), \tag{5}$$

where $I_{\text{pv}n}$ is the light-generated current at the STCs. K_i, the short-circuit current coefficient, is one of the ISDM parameters. The difference between the standard test temperature T_n and T is denoted by ΔT.

Based on the diode theory, Messenger and Ventre [20] presented an approximate linear expression for the diode saturation current I_o, which can be expressed as

$$I_o = I_{on}\left(\frac{T}{T_n}\right)e^{[(qE_g/nk)(1/T_n - 1/T)]}, \tag{6}$$

where E_g is the material band gap. Usually, E_g is set at a reasonable level depending on the semiconductor materials (1.12 eV for crystalline silicon and 1.75 eV for amorphous silicon) in simulation and design tools [21]. De Soto et al. [5] present an estimation method for E_g in a wide temperature range:

$$E_g = E_{gn}\left(1 - 0.0002677\Delta T\right), \tag{7}$$

where E_{gn} is a normal value at the STCs ($E_{gn} = 1.12$ eV for silicon cells and $E_{gn} = 1.6$ eV for the triple junction amorphous cells).

In [3], Lo Brano et al. study how the series and shunt resistances are affected by the solar irradiance. On the basis of the experimental data, the values of R_s and R_p are observed varying in inverse linear modes with G:

$$\begin{aligned} R_s &= \frac{G_n}{G} R_{sn}, \\ R_p &= \frac{G_n}{G} R_{pn}, \end{aligned} \tag{8}$$

where the values of the resistances R_{sn} and R_{pn} are evaluated under the STCs.

By using the aforementioned relations, the ISDM described in [5] is able to analytically describe the I-V characteristic of a PV generator for each generic condition of operative temperature and solar irradiance.

3. Parameter Estimation

3.1. Formulation of Parameter Estimation Problem. PV parameter estimation is a process that minimizes the difference between the measured data and the calculated current by adjusting the normal PV parameters. When the number of experimental data is up to N, the objective function can be formulated by RMSE as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(f_i\left(V, I, x\right)\right)^2}, \tag{9}$$

where $x = [I_{\text{pv}}, I_o n, R_s, R_p]$ for SDM and $x = [I_{\text{pvn}}, I_{on}, n, R_{sn}, R_{pn}, K_i, E_g]$ for ISDM. $f(V, I, x)$ is the homogeneous form of (4) which expresses the I-V characteristics of the SDM:

$$f\left(V, I, x\right) = I_{\text{pv}} - I_o\left(e^{(V+IR_s)/nN_sV_t} - 1\right) - \frac{V + IR_s}{R_p} - I. \tag{10}$$

For the case of ISDM, I_{pv}, I_o, n, R, and R_p satisfy the relational expressions discussed in the previous subsection, namely, (5)–(8).

3.2. Cuckoo Search. The CS algorithm [18, 19], proposed by Yang and Deb, is a nature-inspired stochastic global search algorithm that follows three idealized behavior rules.

(i) A cuckoo lays an egg and dumps it randomly into other bird species' nests.

(ii) The best nests with high quality eggs will be carried forward to the next generation.

(iii) There are a fixed number of available host nests. If a host bird discovers that the eggs are not its own, it will either throw these alien eggs away, or it may abandon the nest and build a brand new nest at a nearby location.

```
Cuckoo Search via Lévy Flights
Initialization of n host nests (population)
  while within the stopping criterion,
      Choose a cuckoo egg by Lévy flights and evaluate
      its fitness (F_i);
      Choose an egg in other's nest randomly and
       calculate its fitness (F_j);
      If F_i > F_j, replace jth egg by ith egg;
      A fraction (p_a) of worse nests are demolished and
       replaced by new ones;
      Preserve good nests (best solutions).
  end while
```

Pseudocode 1: Pseudocode of the Cuckoo Search (CS) [19].

Based on the three rules, the basic steps of CS can be summarized by the pseudocode shown in Pseudocode 1. In the CS algorithm, a pattern corresponds to a nest while each individual attribute of the pattern corresponds to an egg laid by the cuckoo. On the basis of random-walk algorithms, the general system equation of the CS algorithm is given in:

$$X_{g+1;i} = X_{g;i} + \alpha \otimes \text{Lévy}\left(\beta\right), \tag{11}$$

where g and i denote the generation number ($g = 1, 2, 3, \ldots, \text{MaxGen}$) and the pattern number ($i = 1, 2, \ldots, n$), respectively. The product $\otimes$ means entry-wise multiplications. Here $\alpha > 0$ is the step size scaling factor which should be related to the scales of the problem of interest [19]. The jth attributes of the ith pattern is initiated by using (12),

$$X_{g=0;j,i} = \text{rand} \cdot \left(Ub_i - Lb_i\right) + Lb_i, \tag{12}$$

where Ub_i and Lb_i are the upper and lower bounds of the jth attributes, respectively. In each computation step, the CS algorithm checks whether the value of an attribute exceeds the allowed search range. If this happens, the value of the related attribute will be updated with the corresponding boundary value.

Before the searching process, the CS algorithm detects the most successful pattern as x_{best} pattern. Among the existing algorithms exist for generating Lévy flights in the literature, Yang and Deb [18, 19] reported that Mantegna's algorithm [22] works well in most of the optimization problems. Accordingly the evolution phase of the pattern initialized with the detection step of ϕ, which is given by (13) [23]:

$$\phi = \left(\frac{\Gamma(1+\beta)\cdot\sin\left(\pi\cdot\beta/2\right)}{\Gamma\left(\left(\left(1+\beta\right)/2\right)\cdot\beta\cdot 2^{(\beta-1)/2}\right)}\right)^{1/\beta}, \tag{13}$$

where β is 1.5 in the standard software implementation of the CS algorithm [30]. Γ denotes the gamma function.

After initialization, the evolution phase of the x_i pattern starts by defining the donor vector v, where $v = x_i$. The required step size of the jth attributes can be calculated by the following equation:

$$s_j = 0.01 \cdot \left(\frac{u_j}{v_j}\right)^{1/\beta} \cdot \left(v - x_{\text{best}}\right), \tag{14}$$

where $u = \phi \cdot \text{rand}n[D]$ and $v = \text{rand}n[D]$. The $\text{rand}n[D]$ function generates a uniform integer between $[1, D]$ [25]. The donor pattern v is then randomly adjusted by

$$v = v + s_j \cdot \text{rand}n\,[D]\,. \tag{15}$$

The CS algorithm will evaluate the fitness of the random pattern. If a better solution is caught, the x_{best} pattern will be updated. The unfeasible patterns are revised by the crossover operator given in (16) as follows:

$$v_i = \begin{cases} x_i + \text{rand} \cdot \left(x_{r1} - x_{r2}\right), & \text{rand}_i > p_0, \\ x_i, & \text{others}, \end{cases} \tag{16}$$

where p_0 is the mutation probability value ($p_0 = 0.25$ in the standard software implementation [30]). The final step of a generation is to check if the revised infeasible patterns deliver a better solution.

4. Results and Discussions

With the aim of providing a thorough evaluation of the CS algorithm in estimating the PV parameters, both SDM and ISDM are considered in this paper. Two case studies are designed to estimate the CS algorithm in model parameters estimation:

(i) a commercial 57 mm diameter solar cell (R.T.C. France [26]) operating at the standard irradiance level;

(ii) a PV module (KC200GT Multicrystal Photovoltaic Module) operating under varied environment conditions.

During the parameter extraction process, the objective function $f(V, I, x)$ is minimized with respect to the parameters range. In theory, the value of $I_{\text{pv}n}$ is slightly larger than that of I_{scn}. E_{gn} is in a loose range from 1 eV to 2 eV. K_i is around the value provided by the datasheet (normally less than 0.02%/°C). The I_{on} is usually less than 50 μA. As stated in [27], the ideality factor ranges between 1 and 2. PV modules produced by most manufacturers have R_s less than 0.5 Ω and R_p between 5 and 170 Ω [8, 28]. As for PV cell, the ranges of R_s and R_p can be scaled by simply dividing N_s [29].

Statistical analysis is performed to evaluate the quality of the fitted models to the experimental data. Besides RMSE, other two fundamental measures, namely, Individual Absolute Error (IAE) and the Mean Absolute Error (MAE), are applied to evaluate in this paper. Equations (17) and (18) preset the IAE and MAE, respectively:

$$\text{IAE} = \left|I_{\text{calculated}} - I_{\text{measured}}\right|, \tag{17}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} \text{IAE}_i. \tag{18}$$

Table 1: A comparison between the parameter results obtained by the CS algorithm and that of other algorithms from the SDM.

	CS	CPSO [14]	GA [13]	PS [16]
I_{pv}	0.7608	0.7607	0.7619	0.7617
I_o	3.23E–07	4.00E–07	8.09E–07	9.98E–07
n	1.4812	1.5033	1.5751	1.6
R_s	0.0364	0.0354	0.0299	0.0313
R_p	53.7185	59.012	42.3729	61.1026
RMSE	0.0010	0.0014	0.0191	0.0149

The optimization algorithms applied in this paper are programmed in MATLAB. Similar simulation conditions, including population size, maximum generation number, and search ranges, are set to ensure a fair evaluation (population size = 25; maximum generation number = 5000).

4.1. Case Study 1: Parameter Estimation for a PV Cell at the Certain Irradiance Level. Table 1 lists the model parameters of the R.T.C France PV cell at 33°C, which are extracted from the experimental data in [26]. The parameters obtained from the CS algorithm are compared with three different parameter estimation approaches: CPSO [14], GA [13], and PS [16]. From the RMSEs of these methods, which are listed in the last row of Table 1, the CS algorithm [30] outperforms the other three optimization methods. CS obtained slightly lower RMSE, recording 0.0010 in numerical value.

During the parameter estimation process for the SDM, the values of the objective function in different optimization algorithms are shown in Figure 2. The function "ga" in MATLAB [31], whose crossover rate $P_c = 0.8$ and mutation rate $P_m = 0.2$, is utilized for the convergence process test. As for PSO implementation [24], the algorithm parameters are set as learning factors $c_1 = c_2 = 2$, inertia factors $w_{\max} = 0.9$, $w_{\min} = 0.4$, and velocity clamping factor $V_{\max} = 0.5$. In Figure 2, no further improvement by GA is observed after 500 iterations. On the contrary, the CS algorithm showed continuous improvement until the maximum generation. The CS algorithm, whose convergence speed is slightly faster than PSO, shows the best accuracy result in the minimization task after 5000 iterations.

Table 2 lists the parameters of the ISDM obtained by the CS algorithm. In order to evaluate the accuracy of the CS-based estimation, these parameters are substituted into the ISDM. Since the I-V demonstrates nonlinear characteristics, the PV terminal current I is solved by the Newton-Raphson method [32] in this paper. In Table 3, the calculated results I_{ISDM} are compared with the experimental data I_{measured} to observe the agreement between them. The notations IAE_{SDM} and IAE_{ISDM} denote the IAE for SDM and ISDM, respectively.

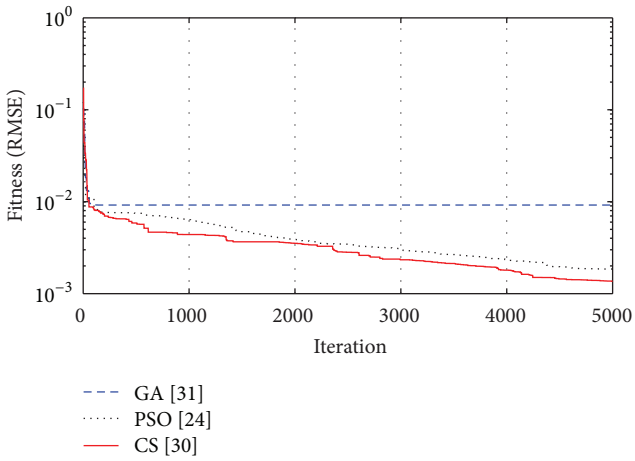


Figure 2: Convergence process of different optimization algorithms during the parameter estimation process of the SDM.

Table 2: Parameters of the ISDM obtained by the CS algorithm.

I_{pvn}	I_{on}	n	R_{sn}	R_{pn}	K_i	E_{gn}
0.7361	1.84E–07	1.5009	0.0355	57.8394	0.0031	1.0020

Although the RMSE of the ISDM is less than that of CPSO, GA, and PS, it is similar to the RMSE of the conventional SDM under a certain environmental condition.

4.2. Case Study 2: Parameter Estimation for a PV Module under Different Environment Conditions. In this section, the validity of the CS algorithm is evaluated using KC200GT PV module operating under different environment conditions. The estimated parameters, both in the SDM and ISDM, are shown in Table 4. As illustrated in Section 1, the main application of the parameter extraction is to predict the I-V characteristics for design purpose. It is worth pointing out that the SDM parameters can only be extracted by the experimental data measured under a certain test condition. Significant errors may occur as the experimental data are measured under varying operating conditions. In the commercial simulation tool like PSIM [21], the PV parameters of the SDM are firstly estimated at the STCs, then the equations (given in the appendix) are applied to calculate the electrical characteristics of different operating conditions. The ISDM-based parameter estimation, however, can be performed by the data measured under any conditions.

Figure 3 shows the I-V curves generated using the parameters obtained by the CS algorithm. The simulated results are compared with the experimental data, which are collected at five different irradiance levels (1000 W/m^2, 800 W/m^2, 600 W/m^2, 400 W/m^2, and 200 W/m^2) and three different temperature levels (25°C, 50°C, and 75°C). It can be seen that the I-V curves of the ISDM fit the whole range of the experimental dataset. On the other hand, the errors of SDM seem larger at lower irradiance and higher temperature levels. With the experimental data, the RMSE of the current I in SDM is calculated as 0.2837, while the RMSE of I in ISDM is only 0.0776.

Figure 4 shows the absolute current errors of different performance predicting methods under different operating conditions. The curves denoted by the label "analytical SDM" are obtained from the analytical SDM model [4]. Ignoring the effect of incidence angle and air mass, the curves labeled by "analytical ISDM" denote the I-V curves from De Soto's

Table 3: A comparison between the errors of ISDM and SDM. The parameters are extracted by the CS algorithm.

No	$V_{measured}$	$I_{measured}$	I_{ISDM}	IAE_{ISDM}	IAE_{SDM}
1	−0.2057	0.7640	0.7639	0.0001	0.0001
2	−0.1291	0.7620	0.7626	0.0006	0.0007
3	−0.0588	0.7605	0.7614	0.0009	0.0009
4	0.0057	0.7605	0.7602	0.0003	0.0003
5	0.0646	0.7600	0.7592	0.0008	0.0009
6	0.1185	0.7590	0.7583	0.0007	0.0010
7	0.1678	0.7570	0.7574	0.0004	0.0001
8	0.2132	0.7570	0.7565	0.0005	0.0009
9	0.2545	0.7555	0.7555	0.0000	0.0004
10	0.2924	0.7540	0.7540	0.0000	0.0003
11	0.3269	0.7505	0.7517	0.0012	0.0009
12	0.3585	0.7465	0.7476	0.0011	0.0009
13	0.3873	0.7385	0.7402	0.0017	0.0016
14	0.4137	0.7280	0.7273	0.0007	0.0006
15	0.4373	0.7065	0.7066	0.0001	0.0005
16	0.4590	0.6755	0.6748	0.0007	0.0002
17	0.4784	0.6320	0.6304	0.0016	0.0011
18	0.4960	0.5730	0.5717	0.0013	0.0009
19	0.5119	0.4990	0.4994	0.0004	0.0005
20	0.5265	0.4130	0.4137	0.0007	0.0005
21	0.5398	0.3165	0.3176	0.0011	0.0007
22	0.5521	0.2120	0.2127	0.0007	0.0001
23	0.5633	0.1035	0.1033	0.0002	0.0008
24	0.5736	−0.0100	−0.0089	0.0011	0.0008
25	0.5833	−0.1230	−0.1244	0.0014	0.0014
26	0.5900	−0.2100	−0.2095	0.0005	0.0009
MAE				0.0007	0.0007
RMSE				0.0010	0.0010

Table 4: Parameters of the KC200GT PV module obtained by the CS algorithm.

(a) SDM parameters (extracted by the CS algorithm)

I_{pv}	I_o	n	R_s	R_p
8.1729	4.23E–10	1.0090	0.2665	140.4875

(b) ISDM parameters (extracted by the CS algorithm)

I_{pvn}	I_{on}	n	R_{sn}	R_{pn}	K_i	E_{gn}
8.1847	5.12E–10	1.0170	0.2574	117.9224	0.0028	1.2474

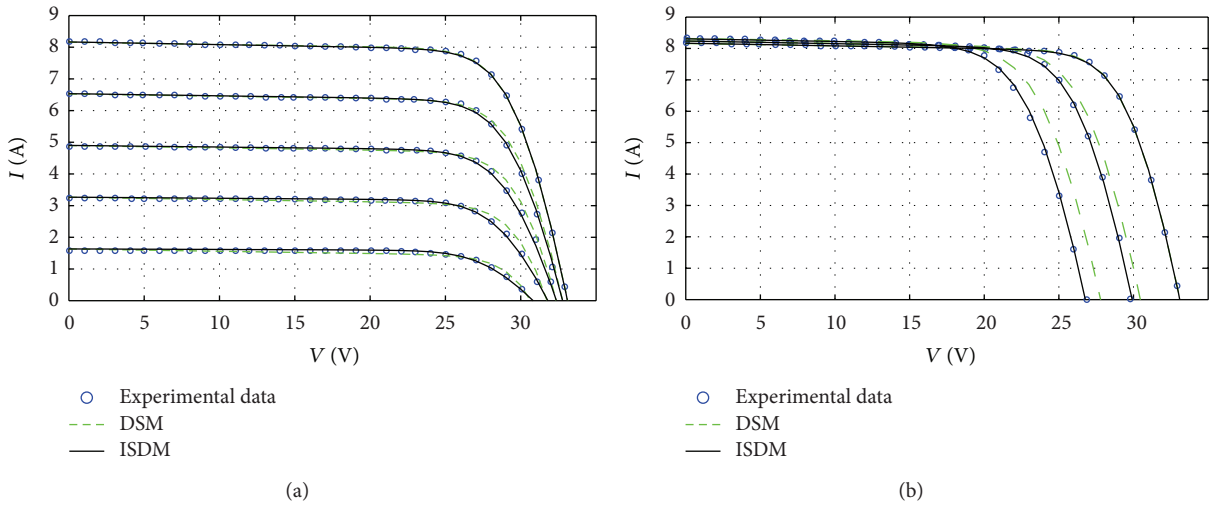


FIGURE 3: The simulated I-V characteristic curves of the KC200GT PV module: (a) under different irradiance levels; (b) under different temperature levels.

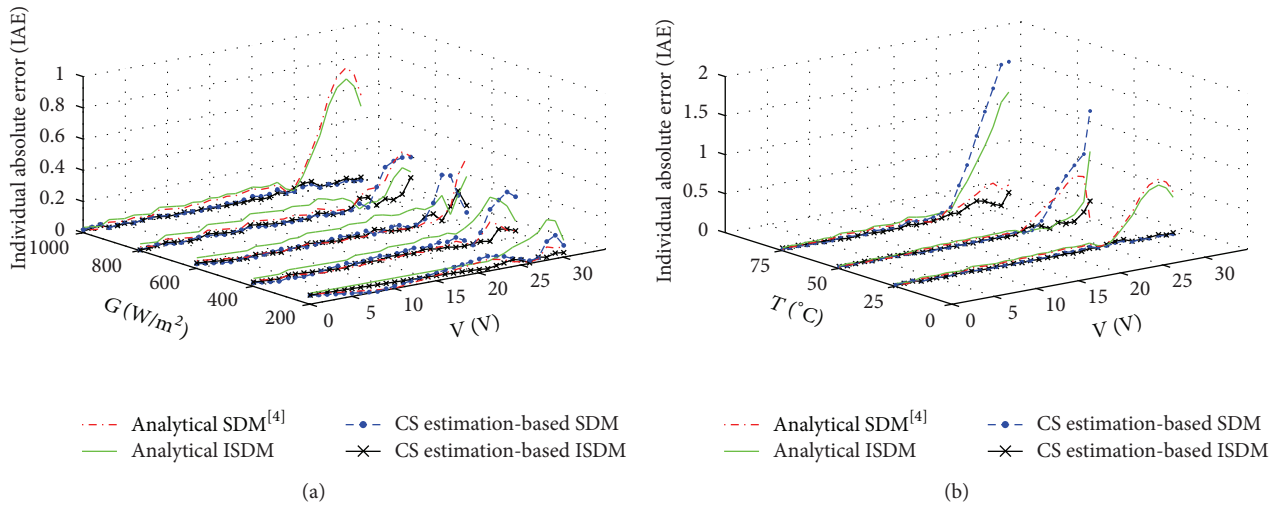


FIGURE 4: A comparison of the individual absolute errors among different PV modeling methods: (a) under different irradiance levels; (b) under different temperature levels.

analytical ISDM model [5]. It is evident the ISDM with the parameters extracted by the CS algorithm is more accurate than the analytical model. As for the SDM, the CS algorithm is capable of extracting a set of PV parameters with a good fit for the experimental data at the STCs. However, the SDM with the equations in the appendix does not exhibit a good prediction performance under other operating conditions.

To further validate the accuracy of the CS algorithm, the extracted parameters are compared to the ones obtained using GA in Figure 5. In general, the CS algorithm gives the better performance than GA for all cases. The Maximum Power Point (MPP), usually locating around 74% of the open circuit voltage, is an important technical data in PV modeling. However, a negative point of the GA-based ISDM is that the errors in the high voltage range are relatively high. The maximum absolute error of the GA-based ISDM is up to about 0.8 A, while the absolute error of the CS is kept below 0.2 A.

5. Conclusion

In this work, the Cuckoo Search (CS) algorithm is applied to estimate the parameters of two PV models, namely, Single Diode Model (SDM) and its improved version (ISDM). The feasibility of the proposed method has been validated by estimating the parameters of two commercial PV generators. The simulation and experimental results showed that the CS algorithm is capable of not only extracting all the parameters of the SDM under a certain condition but also successfully

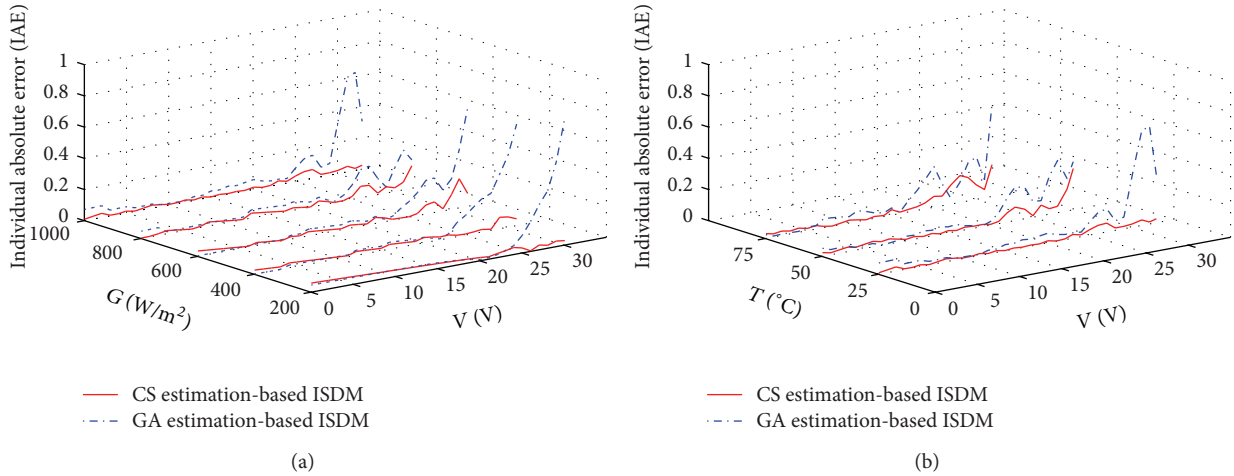


FIGURE 5: A comparison of the individual absolute errors between CS- and GA- based ISDM: (a) under different irradiance levels; (b) under different temperature levels.

estimating all the parameters of ISDM under different operating conditions. In statistical analysis, CS algorithm recorded the lowest RMSE value compared to other algorithms such as GA, PSO and PS.

Appendix

PV Physical Model Adopted in PSIM

By using the parameters extracted at the STCs, the I-V characteristics of a PV generator under nonstandard operating conditions can be calculated via the following equations:

$$
\begin{aligned}
I &= I_{\text{pv}} - I_d - I_R,\\
I_{\text{pv}} &= I_{scn} \cdot \frac{G}{G_n} - K_i \cdot (T - T_n),\\
I_d &= I_o \cdot \left(e^{qV_d/nkT} - 1\right),\\
I_o &= I_{cn} \cdot \left(\frac{G}{G_n}\right)^3 \cdot e^{(qE_g/nk)\cdot(1/T - 1/T_n)},\\
I_R &= \frac{V_d}{R_p},\\
V_d &= \frac{V}{N_s} + I \cdot R_s.
\end{aligned}
\tag{A.1}
$$

Acknowledgments

The authors are grateful to Professor Xin-She Yang for the sharing of Cuckoo Search source code online. Without his generosity, this work would not be possible. This research is supported by the National Natural Science Foundation of China under Grant 61070085.

References

[1] B. Amrouche, A. Guessoum, and M. Belhamel, "A simple behavioural model for solar module electric characteristics based on the first order system step response for MPPT study and comparison," *Applied Energy*, vol. 91, no. 1, pp. 395–404, 2012.

[2] A. Orioli and A. di Gangi, "A procedure to calculate the five-parameter model of crystalline silicon photovoltaic modules on the basis of the tabular performance data," *Applied Energy*, vol. 102, pp. 1160–1177, 2013.

[3] V. Lo Brano, A. Orioli, G. Ciulla, and A. di Gangi, "An improved five-parameter model for photovoltaic modules," *Solar Energy Materials and Solar Cells*, vol. 94, no. 8, pp. 1358–1370, 2010.

[4] M. G. Villalva, J. R. Gazoli, and E. R. Filho, "Comprehensive approach to modeling and simulation of photovoltaic arrays," *IEEE Transactions on Power Electronics*, vol. 24, no. 5, pp. 1198–1208, 2009.

[5] W. De Soto, S. A. Klein, and W. A. Beckman, "Improvement and validation of a model for photovoltaic array performance," *Solar Energy*, vol. 80, no. 1, pp. 78–88, 2006.

[6] K. Ishaque, Z. Salam, and H. Taheri, "Simple, fast and accurate two-diode model for photovoltaic modules," *Solar Energy Materials and Solar Cells*, vol. 95, no. 2, pp. 586–594, 2011.

[7] V. Lo Brano, A. Orioli, and G. Ciulla, "On the experimental validation of an improved five-parameter model for silicon photovoltaic modules," *Solar Energy Materials and Solar Cells*, vol. 105, pp. 27–39, 2012.

[8] A. N. Celik and N. Acikgoz, "Modelling and experimental verification of the operating current of mono-crystalline photovoltaic modules using four- and five-parameter models," *Applied Energy*, vol. 84, no. 1, pp. 1–15, 2007.

[9] M. A. de Blas, J. L. Torres, E. Prieto, and A. García, "Selecting a suitable model for characterizing photovoltaic devices," *Renewable Energy*, vol. 25, no. 3, pp. 371–380, 2002.

[10] J. P. Charles, G. Bordure, A. Khoury, and P. Mialhe, "Consistency of the double exponential model with physical mechanisms of conduction for a solar cell under illumination," *Journal of Physics D*, vol. 18, no. 11, pp. 2261–2268, 1985.

[11] D. S. H. Chan and J. C. H. Phang, "Analytical methods for the extraction of solar-cell single- and double-diode model

parameters from I-V characteristics," *IEEE Transactions on Electron Devices*, vol. 34, no. 2, pp. 286–293, 1987.

[12] J. C. H. Phang, D. S. H. Chan, and J. R. Phillips, "Accurate analytical method for the extraction of solar cell model parameters," *Electronics Letters*, vol. 20, no. 10, pp. 406–408, 1984.

[13] A. J. Joseph, B. Hadj, and A. L. Ali, "Solar cell parameter extraction using genetic algorithms," *Measurement Science and Technology*, vol. 12, no. 11, pp. 1922–1925, 2001.

[14] W. Huang, C. Jiang, L. Xue, and D. Song, "Extracting solar cell model parameters based on chaos particle swarm algorithm," in *Proceedings of the International Conference on Electric Information and Control Engineering (ICEICE '11)*, pp. 398–402, April 2011.

[15] I. Fister, I. Fister Jr, X. S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, 2013.

[16] M. F. AlHajri, K. M. El-Naggar, M. R. AlRashidi, and A. K. Al-Othman, "Optimal extraction of solar cell parameters using pattern search," *Renewable Energy*, vol. 44, pp. 238–245, 2012.

[17] A. Askarzadeh and A. Rezazadeh, "Parameter identification for solar cell models using harmony search-based algorithms," *Solar Energy*, vol. 86, pp. 3241–3249, 2012.

[18] X. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[19] X. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, December 2009.

[20] R. A. Messenger and J. Ventre, *Photovoltaic Systems Engineering*, CRC Press, New York, NY, USA, 2nd edition, 2004.

[21] *PSIM User Manual*, Powersim, Woburn, Mass, USA, 2001.

[22] R. N. Mantegna, "Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes," *Physical Review E*, vol. 49, no. 5, pp. 4677–4683, 1994.

[23] A. H. Gandomi, X. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, pp. 17–35, 2013.

[24] B. Birge, Particle Swarm Optimization Toolbox, http://www.mathworks.com/matlabcentral/fileexchange/7506-particle-swarm-optimization-toolbox.

[25] 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. 315–346, 2013.

[26] T. Easwarakhanthan, J. Bottin, I. Bouhouch, and C. Boutrit, "Nonlinear minimization algorithm for determining the solar cell parameters with microcomputers," *International Journal of Solar Energy*, vol. 4, pp. 1–12, 1986.

[27] M. Bashahu and P. Nkundabakura, "Review and tests of methods for the determination of the solar cell junction ideality factors," *Solar Energy*, vol. 81, no. 7, pp. 856–863, 2007.

[28] S. J. Jun and L. Kay-Soon, "Photovoltaic model identification using particle swarm optimization with inverse barrier constraint," *IEEE Transactions on Power Electronics*, vol. 27, pp. 3975–3983, 2012.

[29] K. Ishaque, Z. Salam, S. Mekhilef, and A. Shamsudin, "Parameter extraction of solar photovoltaic modules using penalty-based differential evolution," *Applied Energy*, vol. 99, pp. 297–308, 2012.

[30] X. S. Yang, Cuckoo Search Algorithm (Source Code), http://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm.

[31] Optimization Toolbox, The MathWorks Inc., http://www.mathworks.com/products/optimization/index.html.

[32] R. L. Burden and J. D. Faires, *Numerical Analysis*, Cengage Learning, Singapore, 2010.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 750819, 12 pages
http://dx.doi.org/10.1155/2013/750819

Research Article

Harmony Search Based Parameter Ensemble Adaptation for Differential Evolution

Rammohan Mallipeddi

School of Electronics Engineering, Kyungpook National University, Taegu 702 701, Republic of Korea

Correspondence should be addressed to Rammohan Mallipeddi; mallipeddi.ram@gmail.com

Received 28 June 2013; Accepted 8 July 2013

Academic Editor: Zong Woo Geem

In differential evolution (DE) algorithm, depending on the characteristics of the problem at hand and the available computational resources, different strategies combined with a different set of parameters may be effective. In addition, a single, well-tuned combination of strategies and parameters may not guarantee optimal performance because different strategies combined with different parameter settings can be appropriate during different stages of the evolution. Therefore, various adaptive/self-adaptive techniques have been proposed to adapt the DE strategies and parameters during the course of evolution. In this paper, we propose a new parameter adaptation technique for DE based on ensemble approach and harmony search algorithm (HS). In the proposed method, an ensemble of parameters is randomly sampled which form the initial harmony memory. The parameter ensemble evolves during the course of the optimization process by HS algorithm. Each parameter combination in the harmony memory is evaluated by testing them on the DE population. The performance of the proposed adaptation method is evaluated using two recently proposed strategies (DE/current-to-*p*best/bin and DE/current-to-gr_best/bin) as basic DE frameworks. Numerical results demonstrate the effectiveness of the proposed adaptation technique compared to the state-of-the-art DE based algorithms on a set of challenging test problems (CEC 2005).

1. Introduction

During the last decade, evolutionary algorithms (EAs) inspired by Darwinian theory of evolution are becoming increasingly popular because of their ability to handle nonlinear and complex optimization problems. Unlike, the conventional numerical optimization methods, EAs are population-based metaheuristic algorithms and require the objective function values, while properties such as differentiability and continuity are not necessary. However, EAs performance depends on the encoding schemes, evolutionary operators, and parameter settings such as population size, mutation scale factor, and crossover rate. In addition, an appropriate parameter selection is a problem dependent and requires a time-consuming trial-and-error parameter tuning process. The trail-and-error based parameter selection is ineffective if the optimization is required in an automated environment or if the user has no experience in the fine art of the control parameter tuning. To overcome this, different parameter adaptation schemes have been presented [1–6]. Among the different parameter adaptation techniques adaptive and self-adaptive techniques are popular due to their ability to adjust the parameter during the course of the evolution with minimal or no intervention from the user. In other words, in adaptive and self-adaptive techniques, the parameter adaptation is done based on the feedback from the search process. Self-adaptive techniques are based on the assumption that the most appropriate parameter values produce better offspring which are more likely to survive and propagate the better parameter values [7]. Therefore, in self-adaptive methods the parameters are directly encoded into the individuals and are evolved together with the encoded solutions.

Differential evolution (DE) [8] is a fast and simple technique that has been successfully applied in diverse fields [9–12]. Like most EAs, the performance of DE [13] is sensitive to population size (NP), mutation and crossover strategies, and their associated control parameters such as scale factor (F) and crossover rate (CR). In other words, the best combination

of strategies and their control parameter settings can be different for different optimization problems. In addition, for a given optimization the best combination of strategies and parameter values differ based on the available computational resources and required accuracy. Therefore, to successfully solve a specific optimization problem, it is generally necessary to perform a time-consuming trial-and-error search for the most appropriate strategies and their associated parameter values. However, in DE during the evolution process, the population traverses through different regions in the search space, within which different strategies [14] with different parameter settings may be more effective than a well-tuned, single combination of strategies and parameters. In DE literature, different partial adaptation schemes have been proposed [7, 14–17] to overcome the time-consuming trial-and-error procedure.

In [18], the authors proposed an adaptive DE algorithm referred to as JADE. In JADE, the authors implemented a new mutation strategy "DE/current-to-*p*best/1" and the control parameters (F and CR) are self-adapted. "DE/current-to-*p*best/1" is a generalized version of "DE/current-to-best/1." JADE uses the conventional binomial crossover strategy. In JADE, the self-adaptation of the control parameters avoids the requirement of prior knowledge about parameter settings and works well without user interaction. Motivated by JADE, the authors in [19] proposed another adaptive DE algorithm referred to as MDE-*p*BX (Modified DE with *p*-best crossover). MDE-*p*BX uses a new mutation strategy "DE/current-to-gr_best/1" which is a modified version of "DE/current-to-best/1." Unlike JADE, MDE-*p*BX uses a more exploitative "*p*-best binomial crossover" strategy. In [20], the authors proposed an ensemble approach for parameter adaptation of DE, where each parameter has a pool of values competing to produce future offspring based on their success in the past generations.

Harmony search (HS) is also population-based metaheuristic optimization algorithm which mimics the music improvisation process. Recently, HS is gaining significance as an efficient optimization algorithm and is used in variety of applications. In HS, the generation of a new vector or solution is based on the consideration of all the existing vectors, rather than considering only two vectors as in DE (parent and mutant vector) [21]. This characteristic of HS makes it more explorative compared to the DE algorithm.

During the past decade, hybridization of EAs has gained significance, due to ability to complement each other's strengths and overcome the drawbacks of the individual algorithms. To enhance the exploitation ability in HS [22], memory consideration is generally employed where new individuals are generated based on the historical search experience. In addition, HS employs random selection approach to explore and sample new solutions from the search space. In HS, the random selection aids the exploration ability but not as efficient as the DE mutation strategy and results in slow convergence characteristics. However, a new solution formed using a set of few randomly selected individuals may limit the exploration ability in DE when the population diversity is low [23]. In [23], the authors propose a hybrid algorithm referred to as differential harmony search (DHS) by fusing the HS and DE mechanisms. The hybridized DHS algorithm could reasonably balance the exploration and exploitation abilities.

In this paper, we propose a DE parameter adaptation technique based on HS algorithm. In the proposed adaptation method, a group of DE control parameter combinations are randomly initialized. The randomly initialized DE parameter combinations form the initial harmony memory (HM) of the HS algorithm. Each combination of the parameters present in the HM is evaluated by testing on the DE population during the evolution. Based on the effectiveness of the DE parameter combinations present in HM, the HS algorithm evolves the parameter combinations. At any given point of time during the evolution of the DE population, the HM contains an ensemble of DE parameters that suits the evolution process of the DE population.

The rest of the paper is organized as follows. Section 2 provides a brief literature review on different adaptive DE and HS algorithms. Section 3 presents the proposed algorithm where the DE parameters are adapted using a HS algorithm. Section 4 presents the experimental results while Section 5 presents the conclusions with some future directions.

2. Literature Review

2.1. Classical Differential Evolution. Differential evolution (DE) is a simple real parameter optimization algorithm that belongs to the class of evolutionary algorithms (EAs) and involves the continuous application of operators such as mutation, crossover, and selection. DE starts with NP, D-dimensional parameter vectors, referred to as population, where each individual is a candidate solution to the problem at hand as shown in

$$\mathbf{X}_{i,G} = \left\{x^1_{i,G}, \ldots, x^D_{i,G}\right\}, \quad i = 1, \ldots, NP. \tag{1}$$

The initial population is uniformly sampled from the search space constrained by the prescribed minimum and maximum parameter bounds $\mathbf{X}_{\min} = \{x^1_{\min}, \ldots, x^D_{\min}\}$ and $\mathbf{X}_{\max} = \{x^1_{\max}, \ldots, x^D_{\max}\}$.

After initialization, corresponding to each target vector $\mathbf{X}_{i,G}$ in the population at the generation G, a mutant vector $\mathbf{V}_{i,G} = \{v^1_{i,G}, v^2_{i,G}, \ldots, v^D_{i,G}\}$ can be generated via a mutation strategy. The most commonly used mutation strategy in DE is given by:

"DE/rand/1" [24]:

$$\mathbf{V}_{i,G} = \mathbf{X}_{r^i_1,G} + F \cdot \left(\mathbf{X}_{r^i_2,G} - \mathbf{X}_{r^i_3,G}\right). \tag{2}$$

The indices r^i_1, r^i_2, r^i_3 are randomly generated mutually exclusive integers in the range $[1, NP]$. The indices are randomly generated once for each mutant vector and are also different from the index i. The scale factor F is a positive control parameter for scaling the difference vector.

After the mutation, crossover operation is applied to each pair of the target vector $\mathbf{X}_{i,G}$ and its corresponding mutant vector $\mathbf{V}_{i,G}$ to generate a trial vector: $\mathbf{U}_{i,G}$ =

$\{u_{i,G}^1, u_{i,G}^2, \ldots, u_{i,G}^D\}$. In the basic version, DE employs the binomial (uniform) crossover defined as follows [25]:

$$u_{i,G}^j = \begin{cases} v_{i,G}^j & \text{if } \left(\text{rand}_j\ [0,1] \le CR\right) \\ & \text{or } \quad \left(j = j_{\text{rand}}\right) \quad j = 1, 2, \ldots, D \\ x_{i,G}^j & \text{otherwise.} \end{cases} \tag{3}$$

In (3), the crossover rate CR is a user-specified constant within the range [0,1], which controls the fraction of parameter values copied from the mutant vector. j_{rand} is a randomly chosen integer in the range $[1, D]$. In DE, there exists another type of crossover operator called exponential crossover which is functionally equivalent to the circular two-point crossover operator [14].

After crossover, the generated trial vectors are evaluated using the objective function and a selection operation is performed as shown in

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if } f\left(\mathbf{U}_{i,G}\right) \le f\left(\mathbf{X}_{i,G}\right), \\ \mathbf{X}_{i,G}, & \text{otherwise.} \end{cases} \tag{4}$$

In (4), $f(\mathbf{U}_{i,G})$ and $f(\mathbf{X}_{i,G})$ correspond to the objective values of the trial and target vectors.

As mentioned above, the mutation, crossover, and selection steps are repeated generation after generation until a termination criterion (reaching the maximum number of function evaluations set) is satisfied. The algorithmic description of the DE is summarized in Algorithm 1.

2.2. Parameter Adaptation in Differential Evolution. Although, DE has attracted much attention recently as a global optimizer over continuous spaces [25], the performance of the conventional DE algorithm depends on the chosen mutation and crossover strategies and the associated control parameters. Depending on the complexity of the problem, the performance of DE becomes more sensitive to the strategies and the associated parameter values [26] and inappropriate choice may lead to premature convergence, stagnation, or wastage of computational resources [16, 26–29]. In other words, due to the complex interaction of control parameters with the DE's performance [7], choosing an appropriate mutation and crossover strategies and control parameters require some expertise. Since DE was proposed, various empirical guidelines were suggested for choosing the population size (NP) [24–26, 29, 30], mutation and crossover strategies [12, 24–26, 28, 31, 32], and their associated control parameter settings: scale factor (F) [24–26, 29, 30, 33, 34] and crossover rate (CR) [24–26, 28–30, 35, 36].

To some extent, the guidelines are useful for selecting the individual parameters of DE. However, the performance of DE is more sensitive to the combination of the mutation strategy and its associated parameters. For a mutation strategy, [7] a particular value of CR makes the parameter F sensitive while some other values of CR make the same F robust. Hence, the manual parameter tuning of DE is not easy and requires a good expertise. To overcome the burden of tuning the DE parameters by trial-and-error, various adaptive techniques have been proposed [14–16, 37–39]. The most popular adaptive DE variants are as follows [40].

(1) SaDE [14]: in SaDE, the trail vector generation strategies and the associated control parameter values are self-adapted based on their previous experiences of generating promising solutions.

(2) jDE [7]: the control parameters F and CR are encoded into the individuals and are adjusted based on the parameters τ_1 and τ_2. The initial values of F and CR of each population individual of DE were selected as 0.5 and 0.9, respectively. Then, based on a random number (rand) which is uniformly generated in the range of $[0, 1]$, the values of F and CR are reinitialized if rand $< \tau_1$ and rand $< \tau_2$, respectively. F and CR are reinitialized to a new value randomly generated in the ranges [0.1, 1.0] and [0, 1], respectively.

(3) JADE [18]: JADE employs a new mutation strategy "DE/current-to-$p$best" and updates the control parameters in an adaptive manner. "DE/current-to-$p$best" is a generalized version of "DE/current-to-best" and helps in diversifying the population and improves the convergence performance. In JADE, the parameter adaptation is done automatically and does not need any prior knowledge regarding relationship between the parameter settings and the characteristics of optimization problems. In JADE, the F and CR values corresponding to each population member are sampled from the mean values of F and CR. The mean values of F and CR are updated by the individual F and CR values which are successful in generating better trail vectors compared to the target vectors.

(4) EPSDE [20]: while solving a specific problem, different mutation strategies with different parameter settings may be better during different stages of the evolution than a single mutation strategy with unique parameter settings as in the conventional DE. Motivated by these observations an ensemble of mutation strategies and parameter values for DE (EPSDE) was proposed in which a pool of mutation strategies, along with a pool of values corresponding to each associated parameter competes to produce successful offspring population. In EPSDE, the candidate pool of mutation strategies and parameters should be restrictive to avoid the unfavorable influences of less effective mutation strategies and parameters [14].

(5) MDE-pBX [19]: motivated by JADE, MDE-pBX employs a new mutation strategy "DE/current-to-gr_best/1" and the control parameters are self-adapted. According to the new mutation strategy, the algorithm uses the best individual of a group (whose size is q% of the population size) of randomly selected solutions from current generation to perturb the parent (target) vector. In addition, unlike JADE, MDE-pBX uses a modified binomial crossover operation referred to as "p-best crossover." According to the modified crossover operation, a biased parent selection scheme has been incorporated by letting

STEP 1: Randomly initialize a population of *NP*, *D*-dimensional parameter vectors. Set the generation number $G = 0$.
STEP 2: **WHILE** stopping criterion is not satisfied
DO
Mutation—Equation (2)
Crossover—Equation (3)
Selection—Equation (4)
Increment the generation count $G = G + 1$
STEP 3: **END WHILE**

Algorithm 1: Standard differential evolution algorithm.

STEP 1: Initialize the HM with HMS randomly generated solutions. Set generation count $G = 0$.
STEP 2: **WHILE** stopping criterion is not satisfied
/* *Generate a new solution** /
FOR each decision variable **DO**
IF $rand_1$ < HMCR
Pick the value from one of the solutions in HM
IF $rand_2$ < PAR
Perturb the value picked /* *New solution generated** /
END IF
END IF
END FOR
IF new solution better than the worst solution in HM (in terms of fitness)
Replace the worst solution in HM with new solution
END IF
Increment the generation count $G = G + 1$
STEP 3: **END WHILE**

Algorithm 2: Standard harmony search algorithm.

each mutant undergo the usual binomial crossover with one of the p top-ranked individuals from the current population and not with the target vector with the same index as used in all variants of DE.

2.3. Harmony Search Algorithm. Unlike most EAs, which simulate natural selection and biological evolution, HS is a population-based metaheuristic optimization algorithm which mimics the music improvisation process where musicians improvise their instruments' pitch by searching for a perfect state of harmony. Some of the characteristics of HS that distinguish it from other metaheuristics such as DE are as follows [21]: (1) considering all the existing solution vectors while generating a new vector, rather than considering only two vectors as in DE (target vector and trail vector); and (2) independent consideration for each decision variable in a solution vector. An overview of the standard HS algorithm is presented in Algorithm 2.

In HS the improvisation operators, memory consideration, pitch adjustment, and random consideration play a major role in achieving the desired balance between the exploitation and exploration during the optimization process [41]. Essentially, both pitch adjustment and random consideration are the key components of achieving the desired diversification in HS. In random consideration, the new vector's components are generated at random and have the same level of efficiency as in other algorithms that handle randomization. The random consideration of HS allows the exploration of new regions that may not have been visited in the search space. In HS, pitch adjustment enhances diversification by tuning the components of a new vector's within a given bandwidth by adding/subtracting a small random amount to an existing component stored in HM. Further to that, pitch adjustment operator can also be considered as a mechanism to support the intensification of HS through controlling the probability of PAR. The intensification in the HS algorithm is represented by the third HS operator, memory consideration. A high harmony acceptance rate means that good solutions from the history/memory are more likely to be selected or inherited. This is equivalent to a certain degree of elitism. Obviously, if the acceptance rate is too low, solutions will converge more slowly.

Recently, HS algorithm garnered a lot of attention from the research community and is successfully applied in solving many optimization problems in engineering and computer science. Consequently, the interest in HS has led to the improvement and development of its performance in line with the requirements of problems that are solved. The improvements proposed by different researchers related to HS can be categorized as follows [42]: (1) HS improvement

STEP 1: Initialize a population of *NP*, *D*-dimensional individuals as the population of DE
STEP 2: Initialize HM with HMS randomly selected individuals.
STEP 3: **WHILE** stopping criterion is not satisfied
DO
Generate a new vector based on HS
FOR 1: HMS + 1 /* Evaluate each parameter combination of the HM */
Mutation
Crossover
Selection
Evaluate the objective value of each HM vector
Increment the generation count $G = G + 1$
END FOR
Update HM
STEP 4: **END WHILE**

Algorithm 3: Harmony search based parameter ensemble adaptation for DE (HSPEADE).

by appropriate parameters setting; and (2) improvement of HS by hybridizing with other metaheuristic algorithms.

3. Harmony Search Based Parameter Ensemble Adaptation for DE (HSPEADE)

As highlighted in the previous section, depending on the nature of problem (unimodal or multimodal) and available computation resources, different optimization problems require different mutation and crossover strategies combined with different parameter values to obtain optimal performance. In addition, to solve a specific problem, different mutation and crossover strategies with different parameter settings may be better during different stages of the evolution than a single set of strategies with unique parameter settings as in the conventional DE. Motivated by these observations, the authors in [20] proposed an ensemble approach (EPSDE) in which a pool of mutation and crossover strategies, along with a pool of values corresponding to each associated parameter competes to produce successful offspring population.

In EPSDE, each member in the DE population is randomly assigned a mutation and crossover strategies and the associated parameter values taken from the respective pools. The population members (target vectors) produce offspring (trial vectors) using the assigned strategies and parameter values. If the generated trial vector is able to enter the next generation of the evolution, then combination of the strategies and the parameter values that produced trail vector are stored. If trial vector fails to enter the next generation, then the strategies and parameter values associated with that target vector are randomly reinitialized from the respective pools or from the successful combinations stored with equal probability.

To have an optimal performance based on the ensemble approach, the candidate pool of strategies and parameters should be restrictive to avoid the unfavorable influences of less effective strategies and parameters [14]. In other words, the strategies and the parameters present in the respective pools should have diverse characteristics, so that they can exhibit distinct performance characteristics during different stages of the evolution, when dealing with a particular problem.

In EPSDE, since the strategy and parameter pools are restrictive, most of the individuals in the pools may become obsolete during the course of the evolution of DE population. Therefore, it would be apt if the strategy and the parameter pools can evolve with the DE population. Based on this motivation, we propose an HS based parameter ensemble adaptation for DE (HSPEADE). The overall view of the proposed HSPEADE is presented in Algorithm 3.

As shown in Algorithm 3, after the initialization of DE population, the HM of the HS algorithm is initialized with HMS number of randomly generated vectors. The members of the HM are the parameter combinations (F and CR values) corresponding to the mutation and crossover strategies used. Using the members in the HM, a new parameter combination vector is generated using the HS algorithm described in Algorithm 2. Each of the HMS + 1 parameter combinations is evaluated by testing them on the DE population during the evolution. After evaluating all the members of the HM and the newly generated parameter combination, the HM is updated as in HS algorithm. The generation of new parameter combination and the updating of the HM are performed throughout the evolution process of the DE algorithm.

To obtain optimal performance based on the ensemble approach, it is obvious that the parameter combinations in HM should be diverse during initial generations of the DE population evolution and should converge to the optimal combination towards the end of the evolution. During the course of the experimentation, we observed that HS is more suitable to evolve the parameter ensemble due to its characteristics such as the following: (1) HS generates a single vector every generation and replaces the worst performing vector; (2) it can randomly generate new solution vectors thus enabling diversity if needed and (3) it considers all the solution vectors in the memory to generate a new solution.

In HSPEADE, to facilitate the diversity in parameter ensemble in the initial stages and to allow the HS to converge

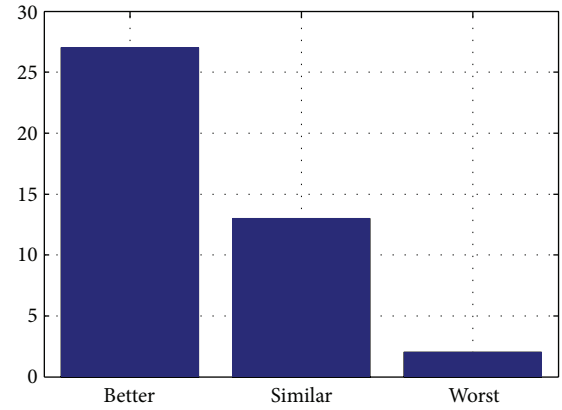


Figure 1: Performance comparison of JADE and HSPEADE1.

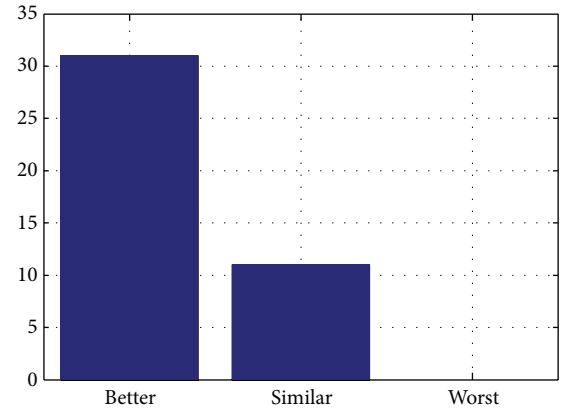


Figure 2: Performance comparison of EPDE1 and HSPEADE1.

to the optimal combination, we made some modifications in the HS algorithm. In HS algorithm shown in Algorithm 2, the parameters HMCR and PAR are deterministically changed. HMCR is linearly increased from 0 to 1 while PAR is decreased linearly from 1 to 0 with the increase in the generation count.

4. Experimental Setup and Results

In this section, we evaluate the performance of the proposed parameter adaptation technique for DE. The details regarding the test problems, experimental environment, and algorithms used for comparison are given below.

4.1. Problem Set. The performance of the proposed method is evaluated using a selected set of standard test functions from the special session on real-parameter optimization of the IEEE Congress on Evolutionary Computation (CEC 2005) [43]. In this work, we use the first 14 functions of CEC 2005 out of which functions 1–5 are unimodal, functions 6–12 are multimodal, and functions 13-14 are hybrid composition functions. The details about the problems such as parameter ranges, location of the optimal solution, and the optimal objective values can be found in [43]. In the present work, to

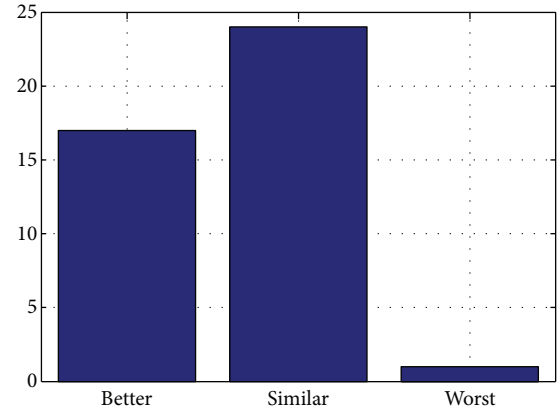


Figure 3: Performance comparison of MDE and HSPEADE2.

evaluate the scalability of the algorithms 30-, 50-, and 100-dimensional versions of the test problems are considered. The number of function evaluation used for 30-, 50-, and 100-dimensional problems are 100000, 500000, and 1000000, respectively. On each of the test problems, every algorithm under consideration is run 30 times.

4.2. Setup for Algorithmic Comparison. The proposed HSPEADE being a general idea can be applied with any frame work. In this work, the experiments are designed as follows.

(1) We consider a single crossover strategy which is binomial crossover. We selected binomial crossover because the two recent adaptive DE algorithms (JADE [18] and MDE-pBX [19]) which show significant performance on the problem set considered employ binomial crossover. It is to be noted that MDE-pBX uses a modified "p-best binomial crossover" operator. However, in this work we consider the classical binomial crossover only.

(2) We consider two mutation strategies "DE/current-to-$p$best" and "DE/current-to-gr_best"

The algorithmic comparison is divided into two sets as follows. SET 1 uses the "DE/current-to-$p$best" strategy while SET 2 uses the "DE/current-to-gr_best" strategy. EPSDE algorithm mentioned above is referred to as EPDE below because in the current work the strategies are fixed. MDE-pBX algorithm is referred to as MDE below because in the present work we use a simple binomial crossover instead of the greedy "p-best crossover."

SET 1:

JADE: "DE/current-to-$p$best" strategy, binomial crossover strategy, $c = 0.1$, $p = 0.05$, F and CR are adapted [18].

EPDE1: "DE/current-to-$p$best" strategy, binomial crossover, $c = 0.1$, $p = 0.05$, $F = \{0.5, 0.7, 0.9\}$, $CR = \{0.1, 0.5, 0.9\}$.

Table 1: Performance of JADE, EPDE1, and HSEPDE1 on 30D problems.

	JADE		EPDE1		HSPEADE1	
	Mean	Std.	Mean	Std.	Mean	Std.
1	0	0	0	0	0	0
2	*8.59E − 28*	*4.19E − 28*	7.89E − 10	7.89E − 10	5.83E − 14	1.84E − 13
3	**1.31E + 04**	**5.75E + 04**	2.47E + 05	9.21E + 04	3.09E + 04	1.27E + 04
4	2.45E − 02	8.40E − 02	*8.16E − 05*	*1.70E − 04*	**6.71E − 06**	**6.39E − 06**
5	7.53E + 02	3.68E + 02	*1.29E − 02*	*2.81E − 02*	**3.19E − 04**	**3.46E − 04**
6	1.03E + 01	2.72E + 01	*1.26E + 00*	*1.88E + 00*	**5.69E − 15**	**5.20E − 15**
7	1.56E − 02	1.51E − 02	1.53E − 02	1.19E − 02	**0**	**0**
8	2.08E + 01	2.46E − 01	2.09E + 01	2.97E − 02	2.09E + 01	2.83E − 02
9	**0**	**0**	8.61E + 00	2.04E + 00	**0**	**0**
10	*2.73E + 01*	*5.70E + 00*	1.26E + 02	1.03E + 01	2.51E + 01	5.41E + 00
11	*2.68E + 01*	*2.03E + 00*	3.36E + 01	1.29E + 00	2.69E + 01	1.56E + 00
12	*4.82E + 03*	*3.97E + 03*	3.52E + 04	3.59E + 04	6.03E + 03	6.01E + 03
13	*1.67E + 00*	*3.04E − 02*	5.36E + 00	2.90E − 01	**1.48E + 00**	**9.91E − 02**
14	*1.24E + 01*	*3.27E − 01*	1.31E + 01	9.57E − 02	**1.28E + 01**	**2.47E − 01**

Table 2: Performance of JADE, EPDE1, and HSEPDE1 on 50D problems.

	JADE		EPDE1		HSPEADE1	
	Mean	Std.	Mean	Std.	Mean	Std.
1	6.347E − 15	2.32E − 15	*0*	*0*	**0**	**0**
2	5.63E − 04	7.82E − 06	*2.47E − 05*	*2.16E − 05*	**3.49E − 08**	**6.85E − 08**
3	**8.75E + 04**	**2.96E + 04**	7.84E + 05	2.33E + 05	1.34E + 05	4.20E + 04
4	1.66E + 03	5.13E − 01	2.16E + 02	3.54E + 02	**6.16E + 00**	**3.77E + 00**
5	3.16E + 03	5.29E + 02	2.20E + 03	6.49E + 02	**1.19E + 03**	**4.51E + 02**
6	*1.54E + 01*	*1.06E + 01*	3.26E + 01	2.89E + 01	**6.97E − 09**	**2.16E − 08**
7	9.83E − 03	1.38E − 02	8.85E − 03	1.18E + 02	**0**	**0**
8	2.11E + 01	3.25E − 02	2.11E + 01	3.34E − 02	2.11E + 01	5.22E − 02
9	**0**	**0**	5.35E + 01	6.45E + 00	**0**	**0**
10	*1.94E + 02*	*2.06E + 01*	2.74E + 02	1.20E + 01	**6.67E + 01**	**2.16E + 01**
11	*6.21E + 01*	*1.74E + 00*	6.47E + 01	1.49E + 00	**5.33E + 01**	**2.26E + 00**
12	1.77E + 05	7.11E + 04	**1.71E + 04**	**1.28E + 04**	**1.73E + 04**	**9.14E + 03**
13	2.31E + 01	4.78E − 01	*1.33E + 01*	*6.41E − 01*	**2.66E + 00**	**7.21E − 02**
14	2.28E + 01	2.55E − 01	2.29E + 01	1.79E − 01	**2.23E + 01**	**4.09E − 01**

HSPEADE1: "DE/current-to-*p*best" strategy, binomial crossover, F, CR and p are encoded into the HS algorithm for adaptation. F ranges from 0.2 to 1.2, CR ranges from 0 to 1, and p ranges from 0.05 to 2.50.

SET 2:

MDE: "DE/current-to-gr_best" strategy, binomial crossover strategy, $q = 0.15$, F and CR are adapted [19].

EPDE2: "DE/current-to-gr_best" strategy, binomial crossover strategy, $q = 0.15$, $F = \{0.5, 0.7, 0.9\}$, $CR = \{0.1, 0.5, 0.9\}$.

HEPEADE2: "DE/current-to-gr_best" strategy, binomial crossover, F, CR, and q are encoded in to the HS algorithm for adaptation. F ranges from 0.2 to 1.2, CR ranges from 0 to 1, and p ranges from 0.05 to 2.50.

4.3. Statistical Tests. To compare the performance of different algorithms, we employ two types of statistical tests, namely, t-test and Wilcoxon rank sum test. The t-test being a parametric method can be used to compare the performance of two algorithms on a single problem. When the performances of two algorithms are to be compared on multiple problems t-test is not valid as the normality assumption fails [44]. Therefore, to compare the performance of two algorithms over a set of different problems, we can use a nonparametric test such as the Wilcoxon rank sum test [44].

4.4. Experimental Results. The experimental results (mean and standard deviations) corresponding to algorithms JADE, EPDE1, and HSEPDE1 (SET 1) on 30-, 50-, and 100-dimensional problems are presented in Tables 1, 2, and 3, respectively. The experimental results (mean and standard deviations) corresponding to algorithms MDE, EPDE2, and

Table 3: Performance of JADE, EPDE1, and HSEPDE1 on 100D problems.

	JADE		EPDE1		HSPEADE1	
	Mean	Std.	Mean	Std.	Mean	Std.
1	6.48$E-10$	4.02$E-09$	*1.57E − 28*	*2.49E − 28*	**5.05E − 30**	**1.59E − 29**
2	3.49$E+01$	6.47$E+01$	*5.61E + 00*	*1.77E + 01*	**5.99E − 01**	**3.82E − 01**
3	3.01$E+06$	6.21$E+04$	*1.53E + 06*	*601E + 05*	**1.02E + 06**	**3.39E + 05**
4	5.03$E+04$	6.98$E-03$	2.03$E+04$	8.35$E+03$	**1.03E + 04**	**2.49E + 03**
5	7.53$E+05$	3.95$E+03$	*5.89E + 03*	*1.33E + 03*	**3.39E + 03**	**7.82E + 02**
6	6.35$E+03$	7.37$E+01$	*1.85E + 02*	*4.13E + 01*	**5.49E − 01**	**1.66E + 00**
7	8.13$E-03$	5.69$E-03$	4.19$E-03$	6.77$E-03$	**1.23E − 03**	**3.89E − 03**
8	2.19$E+01$	9.47$E+01$	2.13$E+01$	1.49$E-02$	2.13$E+01$	2.61$E-02$
9	*3.55E − 12*	*1.12E − 12*	2.64$E+02$	1.19$E+01$	1.06$E-15$	1.91$E-15$
10	*5.94E + 02*	*8.88E + 01*	7.13$E+02$	1.53$E+01$	**2.39E + 02**	**8.14E + 01**
11	*1.28E + 02*	*3.57E + 00*	1.51$E+02$	3.16$E+00$	**1.27E + 02**	**4.82E + 00**
12	6.77$E+05$	2.11$E+04$	*1.02E + 05*	*6.39E + 04*	**9.39E + 04**	**2.96E + 04**
13	*6.96E + 00*	*7.09E − 01*	4.15$E+01$	1.65$E+00$	**6.17E + 00**	**6.72E − 01**
14	*4.58E + 01*	*3.98E − 01*	4.74$E+01$	1.85$E-01$	4.67$E+01$	3.35$E-01$

Table 4: Performance of MDE, EPDE2, and HSEPDE2 on 30D problems.

	MDE		EPDE2		HSPEADE2	
	Mean	Std.	Mean	Std.	Mean	Std.
1	0	0	0	0	0	0
2	2.19$E-12$	5.17$E-12$	6.29$E-10$	6.59$E-10$	**2.33E − 18**	**5.67E − 18**
3	**2.45E + 04**	**1.31E + 04**	3.30$E+05$	1.52$E+05$	2.83$E+04$	1.69$E+04$
4	5.96$E-05$	1.33$E-04$	*7.30E − 06*	*9.85E − 06*	**3.88E − 06**	**6.79E − 06**
5	6.48$E+02$	3.81$E+02$	*1.04E − 01*	*2.72E − 01*	**1.93E − 04**	**1.56E − 04**
6	2.49$E+01$	2.67$E+01$	*8.02E − 01*	*1.68E + 00*	**2.02E − 15**	**2.17E − 15**
7	1.79$E-02$	1.29$E-02$	*9.59E − 03*	*9.58E − 03*	**0**	**0**
8	2.08$E+01$	2.75$E-01$	2.09$E+01$	5.49$E-02$	2.09$E+01$	5.56$E-02$
9	**0**	**0**	8.62$E+00$	1.96$E+00$	**0**	**0**
10	*2.72E + 01*	*5.27E + 00*	1.19$E+02$	1.01$E+01$	**2.31E + 01**	**7.51E + 00**
11	*2.72E + 01*	*1.68E + 00*	3.25$E+01$	1.84$E+00$	**2.55E + 01**	**3.50E + 00**
12	**2.81E + 03**	**2.01E + 03**	3.97$E+04$	2.42$E+04$	4.19$E+03$	4.61$E+03$
13	**1.44E + 00**	**1.02E − 01**	5.31$E+00$	2.79$E-01$	1.45$E+00$	1.15$E-01$
14	**1.22E + 01**	**5.39E − 01**	1.31$E+01$	2.47$E-01$	1.28$E+01$	1.52$E-01$

HSEPDE2 (SET 2) on 30-, 50-, and 100-dimensional problems are presented in Tables 4, 5, and 6, respectively. In Tables 1–6, the mean and standard deviation (std.) values are reported for every algorithm on each test problem.

The t-test and Wilcoxon rank sum test results comparing the performance of algorithms in SET 1 and SET 2 are presented in Tables 7 and 8, respectively. In Tables 7 and 8, the t-test results comparing two algorithms on each problem are presented and the last row presents the Wilcoxon rank sum test results. For each of the two tests, +1, 0, −1 in A versus B comparison indicates B is better than A, B is equal to A, and B is worse than A, respectively. For example, in JADE versus EPDE1 comparison in Table 7 (30D) EPSDE1 is better, equal and worst on test problems 4, 7, and 9, respectively.

4.5. Analysis of Experimental Results

SET 1. In Tables 1, 2, and 3, for each test problem, the best performing algorithms among the JADE and EPDE1 are highlighted using italic while the overall best among JADE, EPDE1, and HSPEADE1 are highlighted using bold.

From the experimental results, it can be observed that JADE performs better than EPDE1 on 30-dimensional versions of the problems. However, as the dimensionality of the test problems increases, the performance of the EPDE1 becomes better compared to JADE algorithm. The improved performance of EPDE1 can be attributed to the ensemble approach as different combinations of strategies and parameters can be effective during different stages of the evolution process [20].

Table 5: Performance of MDE, EPDE2, and HSEPDE2 on 50D problems.

	MDE		EPDE2		HSPEADE2	
	Mean	Std.	Mean	Std.	Mean	Std.
1	0	0	0	0	0	0
2	*4.19E − 07*	*4.17E − 07*	2.30*E* − 05	1.72*E* − 05	**1.08E − 08**	**3.36E − 08**
3	2.77*E* + 05	7.29*E* + 04	**1.19E + 05**	**4.68E + 04**	6.29*E* + 05	1.81*E* + 05
4	2.03*E* + 02	2.19*E* + 02	*1.30E + 02*	*1.31E + 02*	**3.56E + 00**	**2.87E + 00**
5	4.27*E* + 03	5.51*E* + 02	*2.23E + 03*	*5.09E + 02*	**8.73E + 02**	**6.51E + 02**
6	9.78*E* + 01	3.82*E* + 01	*7.77E + 00*	*6.97E + 00*	**3.99E − 01**	**1.26E + 00**
7	*2.22E − 03*	*4.71E − 03*	1.10*E* − 02	1.65*E* − 02	3.45*E* − 03	5.59*E* − 03
8	2.11*E* + 01	2.98*E* − 02	2.12*E* + 01	2.77*E* − 02	2.11*E* + 01	4.06*E* − 02
9	*5.33E − 16*	*1.19E − 15*	5.41*E* + 01	3.67*E* + 00	**0**	**0**
10	**6.34E + 01**	**6.99E + 00**	2.78*E* + 02	1.14*E* + 01	7.54*E* + 01	1.77*E* + 01
11	*5.37E + 01*	*3.07E + 00*	6.44*E* + 01	2.01*E* + 00	**5.15E + 01**	**6.29E + 00**
12	2.14*E* + 04	1.92*E* + 04	*1.49E + 04*	*1.23E + 04*	**1.46E + 04**	**1.37E + 04**
13	*2.87E + 00*	*3.64E − 01*	1.29*E* + 01	8.07*E* − 01	**2.84E + 00**	**2.79E − 01**
14	*2.18E + 01*	*3.04E − 01*	2.28*E* + 01	3.29*E* − 01	2.23*E* + 01	2.79*E* − 01

Table 6: Performance of MDE, EPDE2, and HSEPDE2 on 100D problems.

	MDE		EPDE2		HSPEADE2	
	Mean	Std.	Mean	Std.	Mean	Std.
1	*5.05E − 30*	*1.59E − 29*	2.68*E* − 28	2.13*E* − 28	**6.31E − 30**	**1.60E − 29**
2	*3.20E − 01*	*4.12E − 01*	1.07*E* + 00	6.92*E* − 01	**1.19E − 02**	**1.30E − 02**
3	3.16*E* + 06	4.85*E* + 05	*1.31E + 06*	*3.51E + 05*	**1.22E + 06**	**3.91E + 05**
4	*1.78E + 04*	*7.20E + 03*	1.93*E* + 04	7.89*E* + 03	**8.88E + 03**	**2.70E + 03**
5	1.38*E* + 04	1.50*E* + 03	*6.02E + 03*	*1.03E + 03*	**2.89E + 03**	**6.91E + 02**
6	2.04*E* + 02	4.41*E* + 01	*1.39E + 02*	*3.69E + 01*	**1.17E + 00**	**2.14E + 00**
7	**8.04E − 04**	**2.34E − 03**	6.65*E* − 03	6.57*E* − 03	5.17*E* − 03	7.01*E* − 03
8	2.13*E* + 01	2.21*E* − 02	2.13*E* + 01	1.99*E* − 02	2.13*E* + 01	2.19*E* − 02
9	*1.94E − 14*	*7.83E − 15*	2.61*E* + 02	1.64*E* + 01	**0**	**0**
10	*2.69E + 02*	*4.02E + 01*	6.99*E* + 02	2.80*E* + 01	**2.17E + 02**	**5.19E + 01**
11	*1.33E + 02*	*3.85E + 00*	1.49*E* + 02	2.97*E* + 00	**1.25E + 02**	**4.06E + 00**
12	8.35*E* + 04	4.31*E* + 04	8.47*E* + 04	3.92*E* + 04	**6.63E + 04**	**5.09E + 04**
13	*8.99E + 00*	*1.42E + 00*	4.20*E* + 01	1.24*E* + 00	**6.32E + 00**	**1.31E + 00**
14	**4.55E + 01**	**5.08E + 01**	4.74*E* + 01	1.78*E* − 01	4.63*E* + 01	3.53*E* − 01

From Tables 1–3, it can be observed that HSPEADE1 outperforms JADE and EPDE1 in most of the test problems. On 30-dimensional problems HSPEADE1 is better than or equal to JADE in 11 cases while JADE is better than HSPEADE1 in 3 cases. As the dimensionality of the test problems increases, the performance of HSPEADE1 gets better than JADE.

From the results, it is clear that the performance of HSPEADE1 is always better than or equal to EPDE1. This confirms our assumption that evolving parameter ensemble is better than fixed combination of parameter ensemble.

SET 2. In Tables 4, 5, and 6, for each test problem, the best performing algorithms among the MDE and EPDE2 are highlighted using italic while the overall best among MDE, EPDE2, and HSPEADE2 are highlighted using bold.

From the experimental results in Tables 4–6 a similar observation to the above can be made. In 30-, 50-, and 100-dimensional problems, the performance of MDE and EPDE2 is distributed. Unlike EPDE1 which dominates JADE as the dimensionality increases, EPDE2 is unable to dominate MDE. This may be due to the explorative ability of "DE/current-to-gr_best" strategy employed in MDE. However, as the dimensionality of the test problems increases the performance of HSPEADE2 becomes better compared to MDE.

From the Wilcoxon rank sum test results (bottom row of Tables 7 and 8), it can be observed that HSPEADE (HSPEADE1 and HSPEADE2) is always better than the algorithms under comparison. In both the experimental setups (SET 1 and SET 2), the statistical t-test results present in Tables 7 and 8 are summarized in Figures 1 to 4 to present a better view. For example in Figure 1, the bar plots indicate the number of test problems (30D, 50D, and 100D) on which HSPEADE1 is better, similar, and worst compared

Table 7: Statistical test results to compare the performance of JADE, EPDE1, and HSEPDE1.

	JADE versus EPDE1			JADE versus HSPEADE1			EPDE1 versus HSPEADE1		
	30D	50D	100D	30D	50D	100D	30D	50D	100D
1	0	+1	+1	0	+1	+1	0	0	0
2	−1	+1	+1	0	+1	+1	+1	+1	+1
3	−1	−1	+1	−1	−1	+1	+1	+1	+1
4	+1	0	0	+1	+1	+1	0	+1	+1
5	+1	0	+1	+1	+1	+1	0	+1	+1
6	+1	−1	+1	+1	+1	+1	+1	+1	+1
7	0	0	0	+1	+1	+1	+1	+1	0
8	0	0	0	0	0	0	0	0	0
9	−1	−1	−1	0	0	0	+1	+1	+1
10	−1	−1	−1	0	+1	+1	+1	+1	+1
11	−1	−1	−1	0	+1	0	+1	+1	+1
12	−1	+1	+1	0	+1	+1	+1	0	+1
13	−1	+1	−1	+1	+1	+1	+1	+1	+1
14	−1	0	−1	+1	+1	0	+1	+1	0
Wilcoxon test	−1	−1	+1	+1	+1	+1	+1	+1	+1

Table 8: Statistical test results to compare the performance of MDE, EPDE2, and HSEPDE2.

	MDE versus EPDE2			MDE versus HSPEADE2			EPDE2 versus HSPEADE2		
	30D	50D	100D	30D	50D	100D	30D	50D	100D
1	0	0	−1	0	0	0	0	0	+1
2	0	−1	−1	+1	+1	+1	+1	+1	+1
3	−1	+1	+1	0	0	+1	+1	−1	0
4	+1	0	0	0	+1	+1	0	+1	+1
5	+1	+1	+1	+1	+1	+1	+1	+1	+1
6	+1	+1	+1	+1	+1	+1	+1	+1	+1
7	+1	0	−1	+1	0	0	+1	0	0
8	0	0	0	0	0	0	0	0	0
9	−1	−1	−1	0	0	+1	+1	+1	+1
10	−1	−1	−1	0	0	+1	+1	+1	+1
11	−1	−1	−1	0	0	+1	+1	+1	+1
12	−1	0	0	0	0	0	+1	0	0
13	−1	−1	−1	0	0	+1	+1	+1	+1
14	−1	−1	−1	0	0	−1	+1	+1	+1
Wilcoxon test	0	0	0	+1	+1	+1	+1	+1	+1

to JADE. From Figures 1, 2, 3, and 4, it is clear that the performance of HSPEADE1 is better than JADE and EPDE1 while HSPEADE2 is better than MDE and EPDE2.

5. Conclusions and Future Work

To improve the performance of DE, different adaptation techniques have been proposed. In this paper, we propose a new parameter adaptation technique for DE based on ensemble approach and HS algorithm and is referred to as HSPEADE. In HSPEADE, an ensemble of parameters is randomly sampled and forms the initial harmony memory. The parameter ensemble evolves during the course of the optimization process by HS algorithm. Each parameter combination in the harmony memory is evaluated by testing them on the DE population. During the initial stages of the evolution the DE parameter combinations in the harmony memory of HS are diverse and facilitate exploration for the better parameter combination. However, during the end of the evolution process fine tuning of the parameter combinations occurs and facilitates exploitation.

The performance of HSPEADE is evaluated by using two recently proposed DE strategies (DE/current-to-*p*best/bin and DE/current-to-gr_best/bin) and the numerical results show that the proposed adaptation technique significant improvement compared to the state-of-the-art adaptive DE

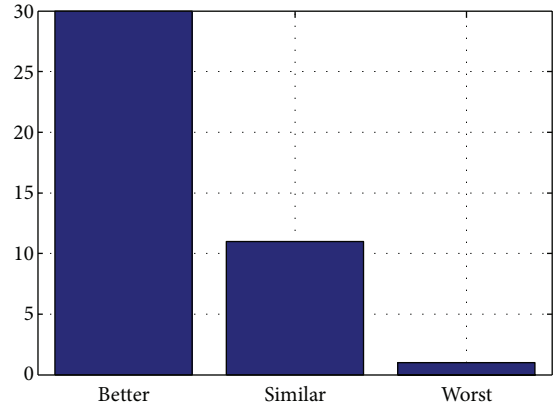


Figure 4: Performance comparison of EPDE2 and HSPEADE2.

algorithms. From the experimental results, it can be observed that the proposed adaptation technique can handle the scalability issues better compared to the other adaptive techniques.

In the present work, we only consider the evolution of the parameter ensemble using the HS framework. As a future work, we would like to incorporate the ensemble of mutation and crossover strategies into the HS framework.

Acknowledgment

This research was supported by Kyungpook National University Research Fund, 2012.

References

[1] P. J. Angeline, M. Palaniswami, Y. Attikiouzel, R. J. Marks, D. Fogel, and T. Fukuda, "Adaptive and self-adaptive evolutionary computation," *Computational Intelligence*, pp. 152–161, 1995.

[2] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[3] J. Gomez, D. Dasgupta, and F. Gonazalez, "Using adaptive operators in genetic search," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03)*, Chicago, Ill, USA, 2003.

[4] B. R. Julstrom, "What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm," in *Proceedings of the 6th International Conference on Genetic Algorithms*, Pittsburgh, Pa, USA, 1995.

[5] J. E. Smith and T. C. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft Computing*, vol. 1, pp. 81–87, 1997.

[6] A. Tuson and P. Ross, "Adapting operator settings in genetic algorithms," *Evolutionary Computation*, vol. 6, no. 2, pp. 161–184, 1998.

[7] 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.

[8] R. Storn and K. Price, "Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep. TR-95-012, ICSI, http://ftp.icsi.berkeley.edu/ftp/pub/techreports/1995/tr-95-012.pdf.

[9] R. Joshi and A. C. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics Part A*, vol. 29, no. 1, pp. 63–76, 1999.

[10] T. Rogalsky, R. W. Derksen, and S. Kocabiyik, "Differential evolution in aerodynamic optimization," in *Proceedings of 46th Annual Conference of Canadian Aeronautics and Space Institute*, pp. 29–36, Montreal, Quebec, 1999.

[11] M. K. Venu, R. Mallipeddi, and P. N. Suganthan, "Fiber Bragg grating sensor array interrogation using differential evolution," *Optoelectronics and Advanced Materials, Rapid Communications*, vol. 2, no. 11, pp. 682–685, 2008.

[12] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[13] J. Liu and J. Lampinen, "On setting the control parameter of the differential evolution method," in *Proceedings of 8th International Conference on Soft Computing (MENDEL '02)*, pp. 11–18, 2002.

[14] 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.

[15] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, "Self-adaptive differential evolution," in *Proceedings of the Computational Intelligence and Security (LNCS '05)*, pp. 192–199, 2005.

[16] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in *Proceedings of the 9th International Conference on Soft Computing*, Brno, Czech Republic, 2003.

[17] J. Tvrdík, "Adaptation in differential evolution: a numerical comparison," *Applied Soft Computing Journal*, vol. 9, no. 3, pp. 1149–1155, 2009.

[18] Z. Jingqiao and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 945–958, 2009.

[19] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 42, no. 2, pp. 482–500, 2012.

[20] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1679–1696, 2011.

[21] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[22] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation*, vol. 198, no. 2, pp. 643–656, 2008.

[23] L.-P. Li and L. Wang, "Hybrid algorithms based on harmony search and differential evolution for global optimization," in *Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation, (GEC '09)*, pp. 271–278, Shanghai, China, June 2009.

[24] R. Storn, "On the usage of differential evolution for function optimization," in *Proceedings of the Biennial Conference of the*

North American Fuzzy Information Processing Society (NAFIPS '96), Berkeley, Calif, USA, June 1996.

[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] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Proceedings of the Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, Interlaken, Switzerland, 2002.

[27] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 991–998, June 2005.

[28] J. Lampinen and I. Zelinka, "On stagnation of the differential evolution algorithm," in *Proceedings of 6th International Mendel Conference on Soft Computing (MENDEL '00)*, pp. 76–83, 2000.

[29] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution : A Practical Approach to Global Optimization*, Natural Computing Series, Springer, Berlin, Germany, 2005.

[30] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, pp. 506–513, Edinburgh, Scotland, September 2005.

[31] K. V. Price, "An introduction to differential evolution," in *New Ideas in Optimization*, D. Corne, M. Dorgio, and F. Glover, Eds., pp. 79–108, McGraw-Hill, London, UK, 1999.

[32] A. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in *Proceedings of the Australian Conference on Artificial Intelligence*, Cairns, Australia, 2004.

[33] D. Zaharie, "Critical values for the control parameters of differential evolution," in *Proceedings of 18th International Conference on Soft Computing (MENDEL '02)*, pp. 62–67, Brno, Czech Republic, 2002.

[34] J. Rönkkönen and J. Lampinen, "On using normally distributed mutation step length for the differential evolution algorithm," in *Proceedings of 19th International MENDEL Confernce on Soft Computing (MENDEL '03)*, pp. 11–18, Brno, Czech Republic, 2003.

[35] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Applied Soft Computing Journal*, vol. 9, no. 3, pp. 1126–1138, 2009.

[36] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "Modified differential evolution for constrained optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 25–32, July 2006.

[37] H. A. Abbass, "The self-adaptive pareto differential evolution algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '02)*, vol. 1, pp. 831–836, 2002.

[38] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.

[39] D. Zaharie and D. Petcu, "Adaptive Pareto differential evolution and its parallelization," in *Proceedings of 5th International Conference on Parallel Processing and Applied Mathematics*, pp. 261–268, Czestochowa, Poland, 2003.

[40] 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.

[41] X.-S. Yang, "Harmony search as a metaheuristic algorithm," in *Music-Inspired Harmony Search Algorithm*, Z. Geem, Ed., vol. 191, pp. 1–14, Springer, Berlin, Germany, 2009.

[42] O. M. Alia and R. Mandava, "The variants of the harmony search algorithm: an overview," *Artificial Intelligence Review*, vol. 36, no. 1, pp. 49–68, 2011.

[43] P. N. Suganthan, N. N. Hansen, J. J. Liang et al., *Problem Definitions and Evaluation Criteria For the CEC, 2005 Special Session on Real-Parameter Optimization*, Nanyang Technological University; Singapore and KanGAL; IIT, Kanpur, India, 2005.

[44] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 252808, 5 pages
http://dx.doi.org/10.1155/2013/252808

Research Article

Determination of Pavement Rehabilitation Activities through a Permutation Algorithm

Sangyum Lee,[1] Sungho Mun,[2] and Hyungchul Moon[3]

[1] *Road Management Division, Seoul Metropolitan Government, Republic of Korea*
[2] *Department of Civil Engineering, Seoul National University of Science & Technology, Republic of Korea*
[3] *ROADTECH Corporation, Ansan-si, Gyunggi-do, Republic of Korea*

Correspondence should be addressed to Sungho Mun; smun@seoultech.ac.kr

Received 15 April 2013; Accepted 29 May 2013

Academic Editor: Zong Woo Geem

This paper presents a mathematical programming model for optimal pavement rehabilitation planning. The model maximized the rehabilitation area through a newly developed permutation algorithm, based on the procedures outlined in the harmony search (HS) algorithm. Additionally, the proposed algorithm was based on an optimal solution method for the problem of multilocation rehabilitation activities on pavement structure, using empirical deterioration and rehabilitation effectiveness models, according to a limited maintenance budget. Thus, nonlinear pavement performance and rehabilitation activity decision models were used to maximize the objective functions of the rehabilitation area within a limited budget, through the permutation algorithm. Our results showed that the heuristic permutation algorithm provided a good optimum in terms of maximizing the rehabilitation area, compared with a method of the worst-first maintenance currently used in Seoul.

1. Introduction

This study involves the objective function of the rehabilitation area to be maximized, using the evaluation model of pavement performance, which is the structural and functional behavior of a road section or length of pavement; furthermore, using budget limitations for pavement rehabilitation, the more deteriorated areas are fixed, based on solving a maximizing problem in this paper. Regression analysis was used to characterize the structural performance under rutting and fatigue deterioration conditions (Figure 1).

The functional behavior of the pavement is based on the passenger's perception of the level of service of the pavement and is related to the comfort quality (i.e., smoothness and safety) of the ride. Pavement riding quality was determined in terms of the international roughness index (IRI), as shown in Figure 2. The measurement of riding serviceability of the pavement used the quarter-car simulation model, which accounts for the sprung mass of the vehicle body, the suspension spring and damper constants, the unsprung mass of the suspension, tire, and wheel, and the spring constant of the tire [1]. For the calculation of IRI, the road profile was filtered using the quarter-car simulation, with specific parameter values that corresponded to a simulation speed of 80 km h^{-1} (Figure 2) [2].

In this study, we created an efficient optimization method, based on the heuristic permutation algorithm, to determine the optimum pavement rehabilitation area for a specified budget. The proposed algorithm, based on the harmony search algorithm [3–6], was developed from field measurement data obtained from a road survey vehicle and quarter-car simulations. Section 2 presents a review of the relevant literature of structural and functional deterioration models. Section 3 describes the determination of the optimum rehabilitation area, using the permutation algorithm. Finally, Section 4 presents the application of a case study.

2. Structural and Functional Deterioration Models

Much attention has been paid to the deterioration model of pavement performance in terms of fatigue cracking and rut depth as permanent deformation [5]. In this approach,

(a) (b)

Figure 1: Pavement distress: (a) rutting and (b) fatigue cracking.

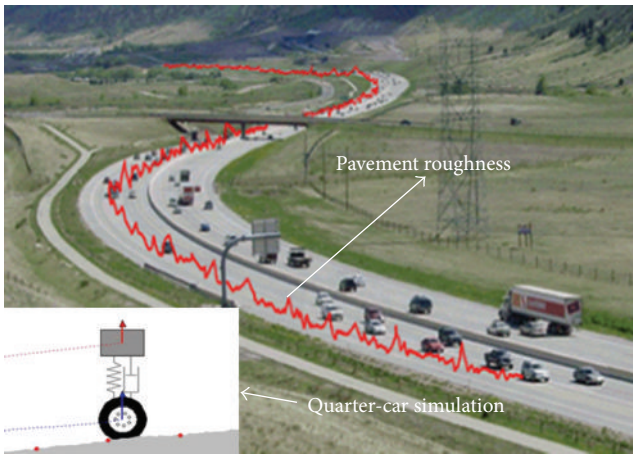


Figure 2: International roughness index (IRI) determination through measurement of pavement roughness and quarter-car simulation.

the initial value of fatigue cracking is assigned a value of 10 (i.e., undamaged status), which decreases as described in (1):

$$\mathrm{SPI}_{\mathrm{crack}} = 10 - 1.667 \cdot \mathrm{Cr}^{0.38}, \quad (1)$$

where $\mathrm{SPI}_{\mathrm{crack}}$ is the Seoul pavement index (SPI) of the crack deterioration model for Seoul city, and Cr is the percentage of crack area of the pavement surface. The $\mathrm{SPI}_{\mathrm{crack}}$ decreases as the crack area increases.

The deterioration model related to rut depth is based on the averaged measurement of permanent deformation in specific pavement sections and is given as

$$\mathrm{SPI}_{\mathrm{RD}} = 10 - 0.267 \cdot \mathrm{RD}, \quad (2)$$

where $\mathrm{SPI}_{\mathrm{RD}}$ is the SPI of the rutting deterioration model in Seoul city, and RD is the rut depth (mm). The intact status of 10 (the initial value) decreases as the rut depth increases.

The functional pavement condition of the IRI value has a similar form as the structural deterioration models specified in (1) and (2):

$$\mathrm{SPI}_{\mathrm{IRI}} = 10 - 0.727 \cdot \mathrm{IRI}, \quad (3)$$

where $\mathrm{SPI}_{\mathrm{IRI}}$ is the functional pavement deterioration index of the IRI ($\mathrm{m\,km^{-1}}$) value, related to ride quality, and IRI is the international roughness index. The unit for the IRI of meters per kilometer ($\mathrm{m\,km^{-1}}$) is based on the slope measurement of the pavement roughness.

Finally, the combination model for determining pavement integrity considers both structural and functional deterioration:

$$\mathrm{SPI} = 10 - \left[\left(10 - \mathrm{SPI}_{\mathrm{crack}}\right)^5 + \left(10 - \mathrm{SPI}_{\mathrm{RD}}\right)^5 + \left(10 - \mathrm{SPI}_{\mathrm{IRI}}\right)^5\right]^{1/5}, \quad (4)$$

where SPI is the integral SPI consisting of a combination of the structural and functional SPIs, including fatigue cracking, rutting, and IRI conditions, for surface deterioration of the pavement in Seoul. The threshold SPI value for pavement rehabilitation is 5.

Figure 3 shows three cases; each case is dependent on one of the indices, $\mathrm{SPI}_{\mathrm{crack}}$, $\mathrm{SPI}_{\mathrm{RD}}$, or $\mathrm{SPI}_{\mathrm{IRI}}$, while the other two indices remain constant. For example, the distress curve of the integral SPI can be calculated for a fixed value of 3 mm for the rut depth and $2\,\mathrm{m\,km^{-1}}$ for the IRI value, as shown in Figure 3(a). The distress curves of the integral SPIs shown in Figures 3(b) and 3(c) are obtained from fixed values of 1% fatigue cracking and $2\,\mathrm{m\,km^{-1}}$ IRI and 1% fatigue cracking and 3 mm rutting, respectively.

If the SPI integral is <5, then the rut depth, cracking area, and IRI are checked to determine the appropriate rehabilitation technique. The procedure is given as follows: (1) check whether the rut depth is ≥18 mm; if so, a modified asphalt overlay of 15 cm thickness is applied; (2) check whether the crack area on the pavement surface is ≥30%; if so, a modified asphalt overlay of 10 cm thickness is applied; if 17% ≤ crack area <30%, then a modified asphalt overlay of 5-cm thickness is applied; (3) check whether the IRI is $\geq 7.3\,\mathrm{m\,km^{-1}}$; if so, a modified asphalt overlay of 5 cm thickness is applied; (4) for all other cases, a traditional asphalt overlay of 5 cm thickness is applied. The costs of the above rehabilitation overlays are shown in Table 1.

3. Permutation Algorithm

The permutation algorithm conceptualizes a behavioral phenomenon of randomly generated permutations, in which a randomly selected permutation set continues to improve the optimization solution, similar to the search for a better state of harmony [6, 7]. This section describes the proposed permutation algorithm, based on the heuristic algorithm,

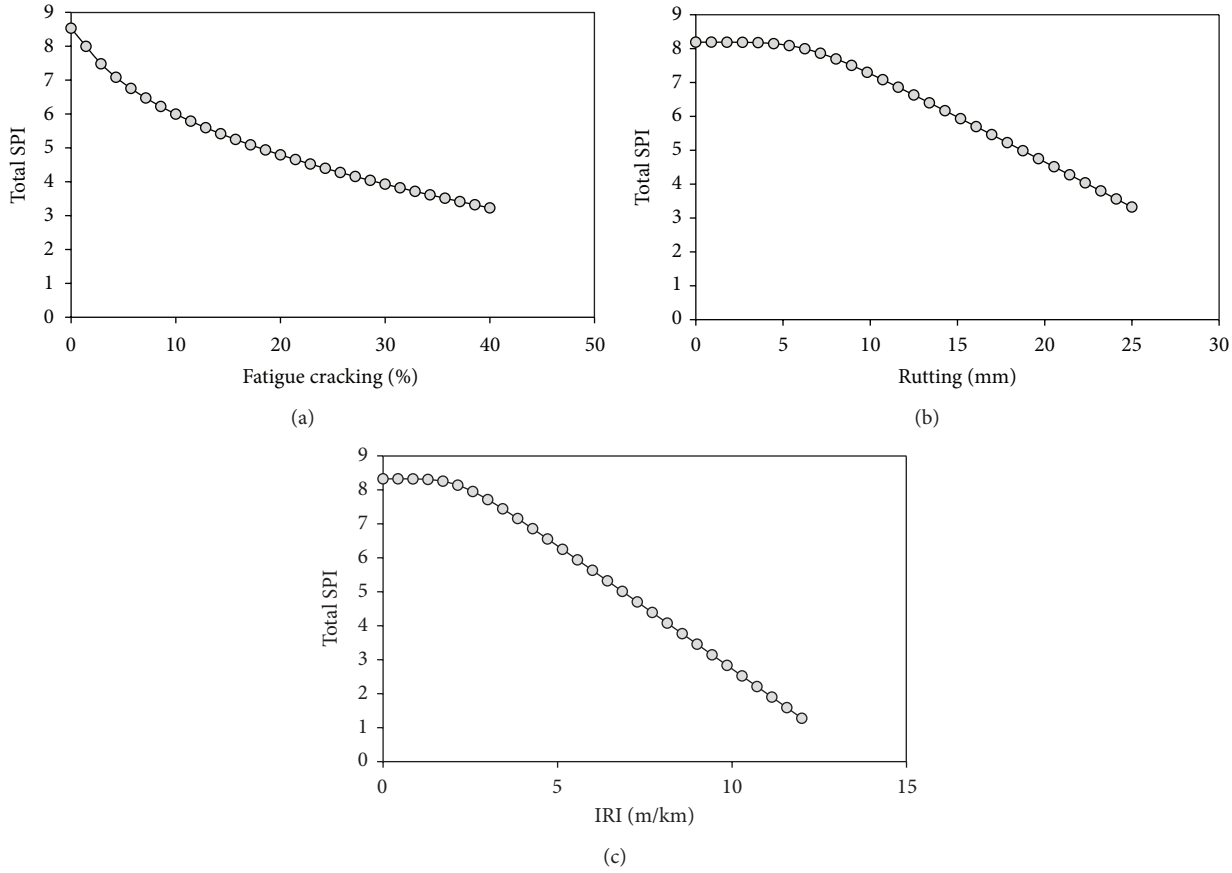


Figure 3: Distress models: (a) fatigue cracking, (b) rutting, and (c) IRI.

Table 1: Costs of the rehabilitation overlays.

Rehabilitation method	Modified asphalt overlay of 15 cm	Modified asphalt overlay of 10 cm	Modified asphalt overlay of 5 cm	General asphalt overlay of 5 cm
Cost (million won/are*)	5.6	4.2	2.5	2.2

*Are is 100 m^2.

which searches for an optimized solution. First, we present an application overview of the permutation algorithm, used to formulate the solution vectors for optimization. The object function is then evaluated. Finally, the application setup of the permutation algorithm is explained in terms of pavement rehabilitation.

3.1. Algorithm Procedure. Budget limitations for pavement rehabilitation in Seoul city require that the worst deteriorated areas are fixed first (worst-first maintenance) with an asphalt overlay, followed by subsequent areas with slightly less wear. The worst-first maintenance is currently used in Seoul city. However, this rehabilitation procedure has not been evaluated to determine whether or not it provides an increase in the overall averaged SPI value, an indicator of the overall pavement surface condition for Seoul city. The more deteriorated pavement surface areas should be rehabilitated through the permutation algorithm to optimize resources within a limited budget, compared with the current rehabilitation method, such as worst-first maintenance.

The procedure for the permutation algorithm consists of Steps 1 through 4. Algorithm preparation is set up in Step 1, as follows: the sectional SPI values of (4) are calculated, regarding the number of deteriorated sections; these values are then checked to determine whether or not they are below the threshold value of 5, to decide the appropriate rehabilitation method; the deteriorated sections below an SPI value of 5 are then grouped; the deteriorated sections are sorted in ascending order, in terms of their SPI values; and finally, the worst-first maintenance sequence is set up.

In Step 2, the optimization problem is specified as follows:

$$\begin{aligned} &\text{Maximize} \quad \text{Area}_T = \sum_{i=1}^{N} \text{DeArea}_i \\ &\text{subject to} \quad L_{\text{Budget}}, \end{aligned} \tag{5}$$

Table 2: Setup of the measured field data used for the heuristic permutation algorithm.

Section number	Road width (m)	Road length (m)	IRI (m/km)	Rutting (mm)	Fatigue cracking (%)
1	4	200	10.55	7.76	4.69
2	23	380	10.02	13.27	8.02
3	23	600	9.29	6.80	3.43
⋮	⋮	⋮	⋮	⋮	⋮
150	18	320	5.92	4.33	5.64

where DeArea_i is the ith deteriorated area, which is computed by multiplying the length of the area by its width. For example, the worst deteriorated area, which shows the lowest value of SPI, can be found in DeArea_1 (e.g., $i = 1$). L_{Budget} is the specified budget limitation.

In Step 3, a sorted column vector of the worst-first sequence in Step 1 is filled in the first row of a permutation memory (PM) matrix. The second row of the PM matrix is filled with a randomly generated permutation vector, consisting of the columns of the rehabilitation sections covered by the limited budget, as well as the corresponding total area, Area_T, as shown in

$$\text{PM} = \left[\begin{array}{cccccc|c} x_1^1 & x_2^1 & x_3^1 & \cdots & \cdots & x_{N_T}^1 & \text{Area}_T^1 \\ x_{11}^2 & x_{N_T}^2 & x_3^2 & \cdots & x_7^2 & \cdots & \text{Area}_T^2 \end{array}\right], \quad (6)$$

where the first row represents the sorted column vector of the worst-first maintenance (e.g., x_2^1 is the first row (superscript) and the second worst deterioration area (subscript)); the second row represents the randomly generated permutation vector; N_T is the total number of deteriorated sections which show an SPI value <5, based on satisfying the limited city budget; and Area_T is the total deteriorated area. Some of the deteriorated sections will not be rehabilitated due to budget constraints.

In Step 4, a comparison process is carried out between the first and second rows in (6). Firstly, the second row in the PM matrix is randomly permutated until Area_T^2 is larger than Area_T^1 in (6). Secondly, a new permutation vector is randomly generated until the new permutation set provides a larger rehabilitation area in comparison with the second row, as given in

$$X^{\text{New}} = \left(x_{32}^{\text{New}}, x_7^{\text{New}}, x_{19}^{\text{New}}, \ldots, x_8^{\text{New}}, \ldots\right). \quad (7)$$

If the new permutation set satisfies the above criterion of the larger rehabilitation area then, it can be added to the third row in the PM matrix, as shown in (7) and

$$\text{PM} = \left[\begin{array}{cccccc|c} x_1^1 & x_2^1 & x_3^1 & \cdots & \cdots & x_{N_T}^1 & \text{Area}_T^1 \\ x_{11}^2 & x_{N_T}^2 & x_3^2 & \cdots & x_7^2 & \cdots & \text{Area}_T^2 \\ x_{32}^3 & x_7^3 & x_{19}^3 & \cdots & x_8^3 & \cdots & \text{Area}_T^3 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{array}\right]. \quad (8)$$

Additionally, based on an evaluation of the objective function value of Area_T in (5), another new permutation set is randomly searched to update the fourth row of the PM matrix, consisting of a larger rehabilitation area, compared with the third row. Finally, if the stopping criterion (i.e., the maximum number of iterations) is satisfied, the procedure of adding better permutation sets into the PM matrix is terminated. Otherwise, Step 4 is repeated.

3.2. Application Setup for the Maximized Rehabilitation Area of Deteriorated Pavement Sections. In terms of applying the permutation algorithm to the area maximization problem of pavement rehabilitation, the costs of rehabilitation overlays shown in Table 1 are considered, as well as the specified budget limitations of this study. The optimization problem can be solved using the permutation algorithm, based on the setup shown in Table 2. For the simplicity of applying the permutation algorithm, the section numbers shown in Table 2 are sequentially assigned in ascending order, in terms of the integral SPI values.

4. Results

Our permutation algorithm was used to evaluate 150 pavement sections being considered for rehabilitation, as shown in Table 2. According to Step 3, the worst-first sequence was stored in the PM matrix, and the second row was obtained by randomly permutated selection, as shown in (9).

Permutation memory (PM) matrix generation for worst-first maintenance, along with the first permutation, is as follows:

$$\text{PM} = \left[\begin{array}{ccccccccc|c} 1 & 2 & 3 & \cdots & \cdots & 27 & 28 & 29 & 30 & 262860 \\ 12 & 50 & 45 & \cdots & 24 & 0 & 0 & 0 & 0 & 257020 \end{array}\right]. \quad (9)$$

There were 50 sections that had an integral SPI value of <5. The 30 sections in the first row can be rehabilitated, using a limited budget of seven billion won (Korean currency), as shown in (9). The second row was updated until the second row chosen by the permutation algorithm provided more rehabilitated areas, compared with the first row of the worst-first sequence. The rehabilitated sections were limited to 25 sections, which provided more rehabilitated area than the approach using worst-first maintenance, as shown in (10).

PM matrix consisting of a better solution in the second row, compared with the worst-first maintenance, is as follows:

$$\text{PM} = \left[\begin{array}{ccccccccc|c} 1 & 2 & \cdots & 25 & 26 & 27 & 28 & 29 & 30 & 262860 \\ 50 & 26 & \cdots & 27 & 0 & 0 & 0 & 0 & 0 & 275580 \end{array}\right]. \quad (10)$$

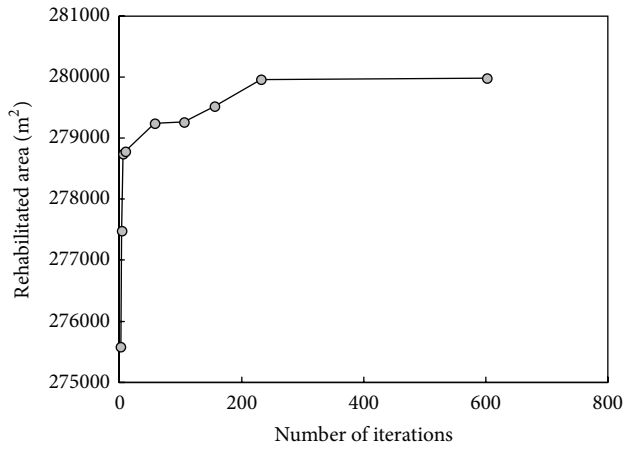


Figure 4: Rehabilitated area as a function of the number of iterations.

The columns above 25th in the second row of the PM matrix were assigned a zero value, as shown in (10). Additionally, a third row was added to the PM matrix when the randomly generated third row resulted in better selection than the second row group, in terms of rehabilitation area covered by the limited budget, as shown in (11). Equation (12) and Figure 4 show the best selection resulting in the largest rehabilitation area, after ~1000 iterations.

PM matrix updating procedure

$$\mathrm{PM} = \left[\begin{array}{cccccccccccc|c} 1 & 2 & 3 & \cdots & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 262860 \\ 50 & 26 & 23 & \cdots & 19 & 27 & 0 & 0 & 0 & 0 & 0 & 275580 \\ 29 & 17 & 47 & \cdots & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 277480 \end{array}\right]. \tag{11}$$

Final PM matrix obtained using the permutation algorithm

$$\mathrm{PM} = \left[\begin{array}{cccccccccccc|c} 1 & 2 & 3 & \cdots & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 262860 \\ 50 & 26 & 23 & \cdots & 19 & 27 & 0 & 0 & 0 & 0 & 0 & 275580 \\ 29 & 17 & 47 & \cdots & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 277480 \\ 53 & 14 & 58 & \cdots & 16 & 30 & 25 & 0 & 0 & 0 & 0 & 278740 \\ 14 & 19 & 11 & \cdots & 49 & 58 & 2 & 46 & 41 & 12 & 31 & 278780 \\ 5 & 9 & 12 & \cdots & 58 & 0 & 0 & 0 & 0 & 0 & 0 & 279240 \\ 18 & 20 & 33 & \cdots & 9 & 45 & 36 & 26 & 43 & 27 & 0 & 279260 \\ 37 & 26 & 41 & \cdots & 8 & 21 & 11 & 3 & 0 & 0 & 0 & 279520 \\ 30 & 2 & 50 & \cdots & 26 & 21 & 9 & 0 & 0 & 0 & 0 & 279960 \\ 38 & 27 & 46 & \cdots & 16 & 23 & 58 & 42 & 28 & 47 & 0 & 279980 \end{array}\right]. \tag{12}$$

5. Conclusion

A permutation algorithm has been described to determine the optimal pavement rehabilitation planning which maximizes the rehabilitation area, using the empirical models of fatigue and rutting deterioration, as well as the functional model of IRI. To evaluate the suggested permutation algorithm, the worst-first maintenance method used in the current rehabilitation technique was compared. Our results indicated that the developed permutation algorithm, described in this study, provided a better solution in terms of covering a larger deteriorated area within the limited city budget, when compared with the worst-first maintenance, which is currently used in Seoul city.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MEST) (no. 2011-0030848).

References

[1] R. Hass, W. R. Hudson, and J. Zaniewski, *Modern Pavement Engineering*, Krieger Publishing Company, Malabar, Fla, USA, 1994.

[2] M. W. Sayers and S. M. Karamihas, "Interpretation of road roughness profile data," Final Report Contract DTFH 61-92-C00143, Federal Highway Administration, 1996.

[3] S. Mun and Z. W. Geem, "Determination of viscoelastic and damage properties of hot mix asphalt concrete using a harmony search algorithm," *Mechanics of Materials*, vol. 41, no. 3, pp. 339–353, 2009.

[4] S. Mun and Z. W. Geem, "Determination of individual sound power levels of noise sources using a harmony search algorithm," *International Journal of Industrial Ergonomics*, vol. 39, no. 2, pp. 366–370, 2009.

[5] Y. Suh, S. Mun, and I. Yeo, "Fatigue life prediction of asphalt concrete pavement using a harmony search algorithm," *KSCE Journal of Civil Engineering*, vol. 14, no. 5, pp. 725–730, 2010.

[6] Z. W. Geem, "Particle-swarm harmony search for water network design," *Engineering Optimization*, vol. 41, no. 4, pp. 297–311, 2009.

[7] J. Fourie, R. Green, and Z. W. Geem, "Generalized adaptive harmony search: a comparative analysis of modern harmony search," *Journal of Applied Mathematics*, vol. 2013, Article ID 380985, 13 pages, 2013.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 427936, 5 pages
http://dx.doi.org/10.1155/2013/427936

Research Article

Economic Dispatch Using Parameter-Setting-Free Harmony Search

Zong Woo Geem

Department of Energy and Information Technology, Gachon University, Seongnam 461-701, Republic of Korea

Correspondence should be addressed to Zong Woo Geem; geem@gachon.ac.kr

Received 5 February 2013; Revised 26 March 2013; Accepted 8 April 2013

Academic Editor: Xin-She Yang

Economic dispatch is one of the popular energy system optimization problems. Recently, it has been solved by various phenomenon-mimicking metaheuristic algorithms such as genetic algorithm, tabu search, evolutionary programming, particle swarm optimization, harmony search, honey bee mating optimization, and firefly algorithm. However, those phenomenon-mimicking problems require a tedious and troublesome process of algorithm parameter value setting. Without a proper parameter setting, good results cannot be guaranteed. Thus, this study adopts a newly developed parameter-setting-free technique combined with the harmony search algorithm and applies it to the economic dispatch problem for the first time, obtaining good results. Hopefully more researchers in energy system fields will adopt this user-friendly technique in their own problems in the future.

1. Introduction

Economic dispatch (ED) is defined in the US Energy Policy Act of 2005 as the operation of electrical generation facilities to produce energy at the least cost to reliably serve consumers while satisfying any operational limits of generation and transmission facilities. ED became a popular optimization problem in energy system field, which has been tackled by various optimization techniques such as genetic algorithm (GA) [1], tabu search (TS) [2], evolutionary programming (EP) [3], particle swarm optimization (PSO) [4], harmony search (HS) [5], honey bee mating optimization (HBMO) [6], and firefly algorithm (FA) [7].

As observed in the literature, better results have been obtained by phenomenon-mimicking metaheuristic algorithms rather than gradient-based mathematical techniques. Indeed, the metaheuristic algorithm has advantages over the mathematical technique in terms of several factors: (1) the former does not require complex derivative functions; (2) the former does not require a feasible starting solution vector which is sensitive to the final solution quality; and (3) the former has more chance to find the global optimum.

However, the metaheuristic algorithm also has the weakness in the sense that it requires "proper and appropriate" value setting for algorithm parameters [8]. For example, in GA, only carefully chosen values for crossover and mutation rates can guarantee good final solution quality, which is not an easy task for algorithm users in practical fields who seldom know how the algorithm exactly works.

In order to overcome this troublesome parameter setting process, researchers have proposed adaptive GA techniques [9], which adjust crossover and mutation rates adaptively, instead of using fixed rates, to find good solutions without manually setting the algorithm parameters. This adaptive technique has been applied to various technical applications such as environmental treatment [10], structural design [11], and sewer network design [12].

In energy system field, the adaptive GA was also applied to a reactive power dispatch optimization as early as 1998 [13]. Afterwards, however, there have been seldom applications in major research databases using the adaptive technique. Thus, this study intends to apply a newly developed adaptive parameter-setting-free (PSF) technique [8], which is combined with the HS algorithm, to the economic dispatch problem for the first time.

2. Economic Dispatch Problem

The economic dispatch problem can be optimally formulated. The objective function can be as follows:

$$\text{Min } z = \sum_i C_i\left(P_i\right), \tag{1}$$

where $C_i(\cdot)$ is generation cost for generator i and P_i is electrical power generated by generator i. Here, $C_i(\cdot)$ can be further expressed as follows:

$$C_i\left(P_i\right) = a_i + b_i P_i + c_i P_i^2 + \left|e_i \times \sin\left(f_i \times \left(P_i^{\min} - P_i\right)\right)\right|, \tag{2}$$

where a_i, b_i, c_i, e_i, and f_i are cost coefficients for generator i. The fourth term in the right-hand side of (2) represents valve-point effects.

The above objective function is to be minimized while satisfying the following equality constraint:

$$\sum_i P_i = D, \tag{3}$$

where D is total load demand. Also, each generator should generate power between minimum and maximum limits as the following inequality constraint:

$$P_i^{\min} \le P_i \le P_i^{\max}. \tag{4}$$

3. Parameter-Setting-Free Technique

The parameter-setting-free harmony search (PSF-HS) algorithm was first proposed for optimizing the discrete-variable problems such as structural design [14], water network design [15], and recreational magic square [8]. PSF-HS was also applied to a continuous-variable problem such as hydrologic parameter calibration [16].

However, it was never applied to a continuous-variable problem with technical constraints. Thus, this study first applies PSF-HS to the ED problem, whose type is the continuous-variable problem with a technical constraint, because its decision variable P_i has the continuous value and it has the equality constraint of total power demand as expressed in (3). Here, the inequality constraint in (4) can be simply considered as value ranges without using any penalty method.

The basic HS algorithm manages a memory matrix, named harmony memory, as follows:

$$\mathbf{HM} = \left[\begin{array}{cccc|c} P_1^1 & P_2^1 & \cdots & P_n^1 & z\left(\mathbf{P}^1\right) \\ P_1^2 & P_2^2 & \cdots & P_n^2 & z\left(\mathbf{P}^2\right) \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ P_1^{\text{HMS}} & P_2^{\text{HMS}} & \cdots & P_n^{\text{HMS}} & z\left(\mathbf{P}^{\text{HMS}}\right) \end{array}\right]. \tag{5}$$

Once this **HM** is fully filled with randomly generated vectors ($\mathbf{P}^1, \ldots, \mathbf{P}^{\text{HMS}}$), a new vector $\mathbf{P}^{\text{New}}$ is generated as follows:

$$P_i^{\text{New}} \longleftarrow \begin{cases} P_i^{\min} \le P_i \le P_i^{\max} & \text{w.p. } R_{\text{Random}} \\ P_i(k) \in \left\{P_i^1, P_i^2, \ldots, P_i^{\text{HMS}}\right\} & \text{w.p. } R_{\text{Memory}} \\ P_i(k) + \Delta & \text{w.p. } R_{\text{Pitch}}, \end{cases} \tag{6}$$

where R_{Random} is random selection rate, R_{Memory} is pure memory consideration rate, R_{Pitch} is pure pitch adjustment rate, and Δ is pitch adjustment amount.

If the newly generated vector $\mathbf{P}^{\text{New}}$ is better than the worst vector $\mathbf{P}^{\text{Worst}}$ in **HM**, those two vectors are swapped as follows:

$$\mathbf{P}^{\text{New}} \in \mathbf{HM} \wedge \mathbf{P}^{\text{Worst}} \notin \mathbf{HM}. \tag{7}$$

The basic HS algorithm performs (6) and (7) until a termination criterion is satisfied.

For PSF-HS, one additional matrix, named operation type matrix (OTM), is also managed as follows:

$$\left[\begin{array}{cccc} o_1^1 = \text{Random} & o_2^1 = \text{Pitch} & \cdots & o_n^1 = \text{Memory} \\ o_1^2 = \text{Memory} & o_2^2 = \text{Memory} & \cdots & o_n^2 = \text{Pitch} \\ \vdots & \cdots & \cdots & \cdots \\ o_1^{\text{HMS}} = \text{Memory} & o_2^{\text{HMS}} = \text{Random} & \cdots & o_n^{\text{HMS}} = \text{Memory} \end{array}\right]. \tag{8}$$

OTM memorizes which operation (random selection, memory consideration, and pitch adjustment) each value comes from. For example, if the value of P_2^2 in **HM** comes from memory consideration operation, the value of o_2^2 in OTM is also set as "Memory." This process happens when initial vectors are populated or when a new vector is inserted into **HM**.

Thus, instead of using fixed algorithm parameter values, PSF-HS can utilize adaptive parameter values by calculating them at each iteration as follows:

$$\begin{aligned} R_{i,\text{Random}} &= \frac{ct\left(o_i^j = \text{Random},\ j = 1, 2, \ldots, \text{HMS}\right)}{\text{HMS}}, \quad i = 1, 2, \ldots, n, \\ R_{i,\text{Memory}} &= \frac{ct\left(o_i^j = \text{Memory},\ j = 1, 2, \ldots, \text{HMS}\right)}{\text{HMS}}, \quad i = 1, 2, \ldots, n, \\ R_{i,\text{Pitch}} &= \frac{ct\left(o_i^j = \text{Pitch},\ j = 1, 2, \ldots, \text{HMS}\right)}{\text{HMS}}, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{9}$$

where $ct(\cdot)$ is a function which counts specific elements that satisfy the condition.

4. Numerical Example

The PSF-HS is applied to a popular bench-mark ED problem with three generators. The input data for the three-generator problem is shown in Table 1.

When the total system demand is set to 850 MW, the optimal solution is known as \$8234.07 [2–4], which was replicated by using a popular gradient-based technique (generalized reduced gradient (GRG) method), which has

Table 1: Data for three-generator example with valve-point loading.

Generator	$P_i^{\min}$	$P_i^{\max}$	a_i	b_i	c_i	e_i	f_i
1	100	600	0.001562	7.92	561	300	0.0315
2	50	200	0.00482	7.97	78	150	0.063
3	100	400	0.00194	7.85	310	200	0.042

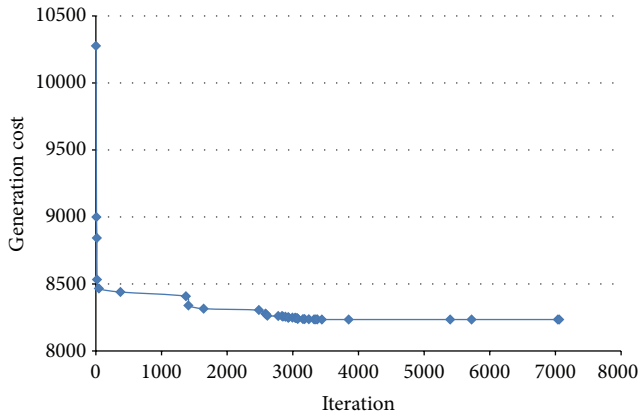


Figure 1: Convergence History of Generation Cost.

been also successfully applied to other energy optimization problems such as building chiller loading [17], combined heat and power ED [18], and hybrid renewable energy system design [19]. However, the GRG method was able to obtain the identical best solution only when it started with a vector (P_1 = 300; P_2 = 150; P_3 = 400). Instead, when, different starting vector (P_1 = 600, P_2 = 200, P_3 = 400) was used, solution quality was worsened as \$8241.41.

When PSF-HS was also applied to the problem, it obtained a near-optimal solution of \$8234.47 after 100 runs, which has small discrepancy from the optimal solution (\$8234.07) by 0.005%. For the results from 100 runs, maximum and mean solutions are \$8429.74 (2.4% discrepancy) and \$8292.88 (0.7% discrepancy), respectively. Here, PSF-HS was performed using MS-Excel VBA environment with Intel CPU 3.3 GHz. Each run takes only one second in this computing environment.

Figure 1 shows the convergence history of power generation cost for the case of the near-optimal solution \$8234.47. As seen in the figure, PSF-HS closely approached to the near-optimal solution in early iterations.

Table 2 shows the final **HM** with HMS = 30. As observed in the table, there are many similar vectors in **HM** because PSF-HS tried local search, instead of global search, in late stage of computation.

Figure 2 shows the history of random selection rate R_{Random}. As observed in the figure, all three parameters ($R_{1,\text{Random}}$, $R_{2,\text{Random}}$, and $R_{3,\text{Random}}$) started with higher values (0.5). In less than 1,000 iterations, $R_{1,\text{Random}}$ went up to around 0.4, $R_{2,\text{Random}}$ to around 0.5, and $R_{3,\text{Random}}$ to around 0.8. Then, they abruptly wend down to less than 0.1 after 3,000 iterations.

Figure 3 shows the history of pure memory consideration rate R_{Memory}. As observed in the figure, all three parameters

Table 2: Values of final HM.

Number	P_1	P_2	P_3	$\sum_i P_i$	$\sum_i C_i(P_i)$
1	300.944	149.782	399.274	850.000	8234.472
2	300.944	149.782	399.274	850.001	8234.477
3	300.944	149.782	399.274	850.001	8234.479
4	300.973	149.754	399.274	850.002	8234.481
5	301.006	149.751	399.244	850.001	8234.482
6	300.974	149.782	399.244	850.000	8234.483
7	300.974	149.754	399.274	850.002	8234.487
8	300.977	149.779	399.244	850.001	8234.489
9	300.977	149.751	399.274	850.003	8234.496
10	300.945	149.782	399.274	850.002	8234.497
11	300.912	149.815	399.274	850.001	8234.501
12	300.934	149.822	399.244	850.000	8234.509
13	300.973	149.784	399.244	850.002	8234.510
14	300.944	149.784	399.274	850.002	8234.511
15	300.912	149.815	399.274	850.002	8234.511
16	300.934	149.794	399.274	850.002	8234.511
17	300.905	149.822	399.274	850.001	8234.514
18	300.974	149.784	399.244	850.002	8234.516
19	300.944	149.784	399.274	850.003	8234.517
20	301.013	149.786	399.202	850.001	8234.520
21	301.013	149.786	399.202	850.002	8234.535
22	300.945	149.784	399.274	850.004	8234.535
23	301.009	149.751	399.244	850.004	8234.536
24	301.006	149.754	399.244	850.004	8234.538
25	300.973	149.786	399.244	850.003	8234.542
26	300.977	149.782	399.244	850.003	8234.542
27	300.944	149.786	399.274	850.004	8234.542
28	301.006	149.794	399.202	850.002	8234.543
29	300.977	149.782	399.244	850.004	8234.547
30	300.974	149.786	399.244	850.004	8234.548

($R_{1,\text{Memory}}$, $R_{2,\text{Memory}}$, and $R_{3,\text{Memory}}$) abruptly went up from the starting point of 0.25. After 4,000 iterations, they became more than 0.8 and stayed.

Figure 4 shows the history of pure pitch adjustment rate R_{Pitch}. As observed in the figure, all three parameters ($R_{1,\text{Pitch}}$, $R_{2,\text{Pitch}}$, and $R_{3,\text{Pitch}}$), from the starting point of 0.25, monotonically stayed less than 0.3 except for one situation when $R_{3,\text{Pitch}}$ spiked near 3,000 iterations.

Furthermore, the sensitivity analysis of initial parameter values was performed. While the original parameter set (R_{Random} = 0.5, R_{Memory} = 0.25, and R_{Pitch} = 0.25) resulted in minimal solution of \$8,243.56 and average solution of \$8,287.69 after 10 runs, equal-valued parameter set (R_{Random} = 0.33, R_{Memory} = 0.33, and R_{Pitch} = 0.33) resulted in minimal solution of \$8,242.12 and average solution of \$8,322.11; memory-consideration-oriented parameter set (R_{Random} = 0.1, R_{Memory} = 0.7, and R_{Pitch} = 0.2) resulted in minimal solution of \$8,241.34 and average solution of \$8,314.45; random-selection-oriented parameter set (R_{Random} = 0.8, R_{Memory} = 0.1, and R_{Pitch} = 0.1) resulted in minimal solution of \$8,241.29 and average solution of

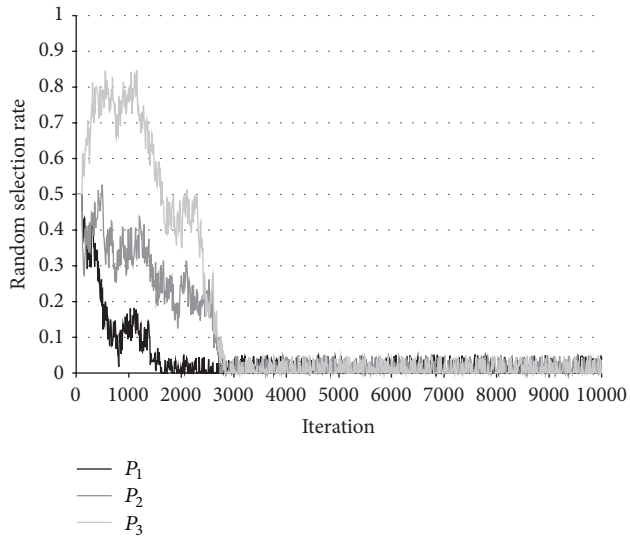


Figure 2: History of Random Selection Rate.

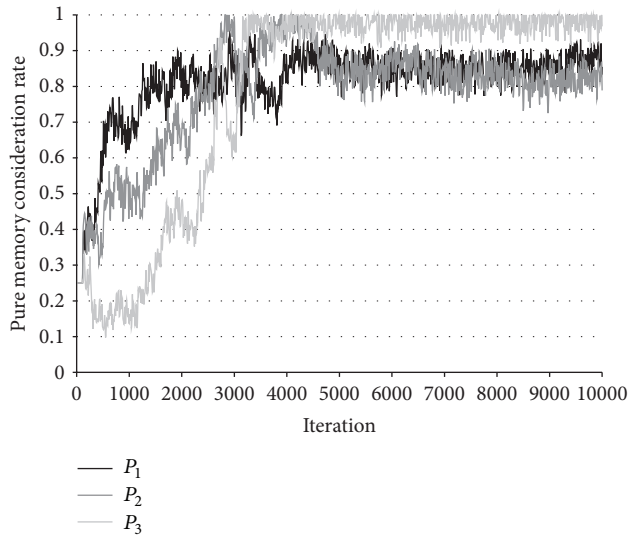


Figure 3: History of Pure Memory Consideration Rate.

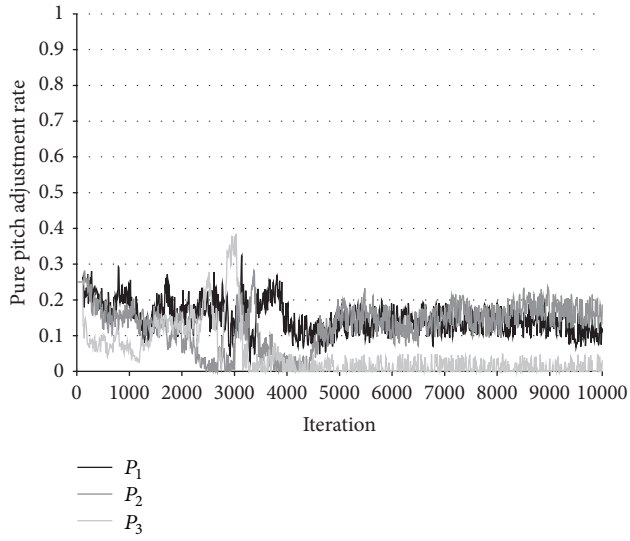


Figure 4: History of Pure Pitch Adjustment Rate.

\$8,272.40. It appeared that the initial parameter values are not very sensitive to final solution quality.

Especially, when the results from memory-consideration-oriented parameter set ($R_{\text{Random}} = 0.1$, $R_{\text{Memory}} = 0.7$, and $R_{\text{Pitch}} = 0.2$) and those from random-selection-oriented parameter set ($R_{\text{Random}} = 0.8$, $R_{\text{Memory}} = 0.1$, and $R_{\text{Pitch}} = 0.1$) were statistically compared, although their variances are different based on F-test ($p = 0.04$), their averages are not significantly different based on t-test ($p = 0.16$).

5. Conclusions

This study applied PSF-HS to the ED problem for the first time, obtaining a good solution which is very close to the best solution ever found. While existing metaheuristic algorithms require carefully chosen algorithm parameters, PSF-HS did not require that tedious process. Thus, there surely exists a tradeoff between original HS and PSF-HS. Also, it should be noted that PSF-HS respectively considers individual algorithm parameters for each variable, which is more efficient way than using lumped parameters for all variables.

For future study, the structure of PSF-HS should be improved to do better performance. Also, it can be applied to large-scale real-world problems to test scalability. Also, other researchers are expected to apply this novel technique to their own energy-related problems.

Acknowledgment

This work was supported by the Gachon University Research Fund of 2013 (GCU-2013-R114).

References

[1] D. C. Walters and G. B. Sheble, "Genetic algorithm solution of economic dispatch with value point loading," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1325–1332, 1993.

[2] W.-M. Lin, F.-S. Cheng, and M.-T. Tsay, "An improved tabu search for economic dispatch with multiple minima," *IEEE Transactions on Power Systems*, vol. 17, no. 1, pp. 108–112, 2002.

[3] N. Sinha, R. Chakrabarti, and P. K. Chattopadhyay, "Evolutionary programming techniques for economic load dispatch," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 1, pp. 83–94, 2003.

[4] J.-B. Park, K.-S. Lee, J.-R. Shin, and K. Y. Lee, "A particle swarm optimization for economic dispatch with nonsmooth cost functions," *IEEE Transactions on Power Systems*, vol. 20, no. 1, pp. 34–42, 2005.

[5] B. K. Panigrahi, V. R. Pandi, S. Das, Z. Cui, and R. Sharma, "Economic load dispatch using population-variance harmony search algorithm," *Transactions of the Institute of Measurement and Control*, vol. 34, no. 6, pp. 746–754, 2012.

[6] T. Niknam, H. D. Mojarrad, H. Z. Meymand, and B. B. Firouzi, "A new honey bee mating optimization algorithm for non-smooth economic dispatch," *Energy*, vol. 36, no. 2, pp. 896–908, 2011.

[7] X.-S. Yang, S. S. S. Hosseini, and A. H. Gandomi, "Firefly Algorithm for solving non-convex economic dispatch problems with valve loading effect," *Applied Soft Computing*, vol. 12, no. 3, pp. 1180–1186, 2012.

[8] Z. W. Geem and K.-B. Sim, "Parameter-setting-free harmony search algorithm," *Applied Mathematics and Computation*, vol. 217, no. 8, pp. 3881–3889, 2010.

[9] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.

[10] M. S. Gibbs, H. R. Maier, and G. C. Dandy, "Comparison of genetic algorithm parameter setting methods for chlorine injection optimization," *Journal of Water Resources Planning and Management*, vol. 136, no. 2, pp. 288–291, 2010.

[11] S. Bekiroğlu, T. Dede, and Y. Ayvaz, "Implementation of different encoding types on structural optimization based on adaptive genetic algorithm," *Finite Elements in Analysis and Design*, vol. 45, no. 11, pp. 826–835, 2009.

[12] A. Haghighi and A. E. Bakhshipour, "Optimization of sewer networks using an adaptive genetic algorithm," *Water Resources Management*, vol. 26, no. 12, pp. 3441–3456, 2012.

[13] Q. H. Wu, Y. J. Cao, and J. Y. Wen, "Optimal reactive power dispatch using an adaptive genetic algorithm," *International Journal of Electrical Power and Energy Systems*, vol. 20, no. 8, pp. 563–569, 1998.

[14] O. Hasaṇebi, F. Erdal, and M. P. Saka, "Adaptive harmony search method for structural optimization," *Journal of Structural Engineering*, vol. 136, no. 4, pp. 419–431, 2010.

[15] Z. W. Geem and Y. H. Cho, "Optimal design of water distribution networks using parameter-setting-free harmony search for two major parameters," *Journal of Water Resources Planning and Management*, vol. 137, no. 4, pp. 377–380, 2011.

[16] Z. W. Geem, "Parameter estimation of the nonlinear muskingum model using parameter-setting-free harmony search," *Journal of Hydrologic Engineering*, vol. 16, no. 8, pp. 684–688, 2011.

[17] Z. W. Geem, "Solution quality improvement in chiller loading optimization," *Applied Thermal Engineering*, vol. 31, no. 10, pp. 1848–1851, 2011.

[18] Z. W. Geem and Y. H. Cho, "Handling non-convex heat-power feasible region in combined heat and power economic dispatch," *International Journal of Electrical Power & Energy Systems*, vol. 34, no. 1, pp. 171–173, 2012.

[19] Z. W. Geem, "Size optimization for a hybrid photovoltaic-wind energy system," *International Journal of Electrical Power & Energy Systems*, vol. 42, no. 1, pp. 448–451, 2012.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 380985, 13 pages
http://dx.doi.org/10.1155/2013/380985

Research Article

Generalised Adaptive Harmony Search: A Comparative Analysis of Modern Harmony Search

Jaco Fourie,[1] Richard Green,[1] and Zong Woo Geem[2]

[1] *Department of Computer Science and Software Engineering, University of Canterbury, Christchurch 8140, New Zealand*
[2] *Department of Energy and Information Technology, Gachon University, Seongnam 461-701, Republic of Korea*

Correspondence should be addressed to Zong Woo Geem; geem@gachon.ac.kr

Received 30 January 2013; Accepted 22 March 2013

Academic Editor: Xin-She Yang

Harmony search (HS) was introduced in 2001 as a heuristic population-based optimisation algorithm. Since then HS has become a popular alternative to other heuristic algorithms like simulated annealing and particle swarm optimisation. However, some flaws, like the need for parameter tuning, were identified and have been a topic of study for much research over the last 10 years. Many variants of HS were developed to address some of these flaws, and most of them have made substantial improvements. In this paper we compare the performance of three recent HS variants: exploratory harmony search, self-adaptive harmony search, and dynamic local-best harmony search. We compare the accuracy of these algorithms, using a set of well-known optimisation benchmark functions that include both unimodal and multimodal problems. Observations from this comparison led us to design a novel hybrid that combines the best attributes of these modern variants into a single optimiser called generalised adaptive harmony search.

1. Introduction

Harmony search (HS) is a relatively new *metaheuristic* optimisation algorithm first introduced in 2001 [1]. In this context, metaheuristic means that, unlike general heuristic algorithms that use trial-and-error methods to solve a specific problem, it uses higher level techniques to solve a general class of problems efficiently. HS fits into the category of population-based evolutionary algorithms together with genetic algorithms (GAs) and the particle swarm optimisation (PSO) algorithm.

As is often the case with evolutionary metaheuristics, HS was inspired by a natural phenomenon. In this case, the methods used by professional musicians to collaboratively improvise new harmonies were taken as the inspiration for HS. An analogy between improvisation and optimisation was constructed, and an algorithm was designed to mimic the way a musician uses short-term memory and past experiences to lead her to the note that results in the most pleasing harmony when played together with the other musicians. Harmony search is easy to implement and can easily be applied and adapted to solve almost any problem that can be modelled as the minimisation or maximisation of an objective function. The objective function itself can be continuous or discrete, and a smooth gradient is not required. No initial solutions or carefully chosen starting points are required. In fact, the objective function is considered a black box by harmony search. Any procedure that takes a solution vector as input and gives a fitness score as output can be used as an objective function.

These properties make harmony search attractive. It has been successfully used in a wide range of disciplines, including computer vision, vehicle routing, music composition, Sudoku puzzle, and various engineering disciplines [2–8]. However, it was soon realised that many aspects of HS can be improved and, in particular, that many of the HS parameters that are often difficult to set can be set and adjusted automatically.

An early approach to this automatic adjustment of the HS parameters was the improved harmony search (IHS) algorithm developed by Mahdavi et al. [9]. They noticed that the pitch adjustment rate (PAR) and fret width (FW) (The fret width (FW) was formally known as the bandwidth (BW), and it is still sometimes referred to as such. The terminology has since been updated to better reflect the musical analogy.) parameters are important to the fine tuning of optimised

solution vectors, and they suggested a method of adapting PAR and FW to the relative progress of the optimiser instead of keeping these parameters constant through all iterations. Their algorithm alleviates the problem of choosing an exact value for the PAR and FW, but it still requires that a range of values specified by a minimum and maximum value be given.

Another early approach to improving HS's performance and solving the parameter setting problem is the global-best harmony search (GHS) algorithm [10]. It was inspired by the swarm intelligence concept used in the particle swarm optimisation algorithm (PSO) [11]. Instead of using the FW parameter to adjust possible solutions closer to an optimum, GHS moves component values toward the value of the current best solution in the population.

Both IHS and GHS improved on the performance of HS and alleviated the parameter setting problem to some degree. However, just in the last few years, many more HS variants were developed that have shown to be superior to these early variants, and some require even less manual parameter setting. Most of these new variants are inspired by other evolutionary algorithms, and they combine the best aspects of the simplex algorithm, simulated annealing, differential evolution, sequential quadratic programming, and many other aspects with HS to form new HS variants [12–16]. For a thorough summery of the latest research in HS variants see the review article by Alia and Mandava [17].

In this paper, we concentrate on the comparison of three of the most recent and promising HS variants: exploratory harmony search (EHS), self-adaptive harmony search (SAHS), and dynamic local-best harmony search (DLHS). All three improve on both the performance of the original HS and many of the earlier HS variants. We also chose these variants because they were not developed as hybrid algorithms by combining HS with other optimisation algorithms. Instead, the focus was on investigating the steps of HS and adding novel modifications to key areas of the original algorithm.

In the section that follows, we briefly overview the HS algorithm and explain how the optimiser parameters influence the performance of the optimisation process. This provides the necessary background to introduce the three HS variants in the sections that follow. In Section 3, we compare the performance of three HS variants using a set of well-known optimisation benchmark functions. The results from these tests are then interpreted and discussed in Section 4. These observations then lead us to develop a novel hybrid algorithm called generalised adaptive harmony search (GAHS) that we introduce in Section 5. We conclude in Section 6 with final remarks and our thoughts on future research in the development of harmony search-based optimisers.

2. Harmony Search and Optimisers Based on Harmony Search

The harmony search algorithm is a metaheuristic population-based optimisation algorithm inspired by the way musicians in a band discover new harmonies through cooperative improvisation [1]. The analogy between the optimisation of an objective function and the improvisation of pleasing harmonies is explained by the actions of the musicians in the band. Each musician corresponds to a decision variable in the solution vector of the problem and is also a dimension in the search space. Each musician (decision variable) has a different instrument whose pitch range corresponds to a decision variable's value range. A solution vector, also called an improvisation, at a certain iteration corresponds to the musical harmony at a certain time, and the objective function corresponds to the audience's aesthetics. New improvisations are based on previously remembered good ones represented by a data structure called the harmony memory (HM).

This analogy is common to all the HS variants that we will investigate. HS variants differ in the way solution vectors (improvisations) are generated and how the HM is updated at the end of each iteration. We start our explanation of how this is done differently in each variant with an overview of the original HS algorithm.

2.1. An Overview of Harmony Search. The core data structure of HS is a matrix of the best solution vectors called the harmony memory (HM). The number of vectors that are simultaneously processed is known as the harmony memory size (HMS). It is one of the algorithm's parameters that has to be set manually. Memory is organised as a matrix with each row representing a solution vector and the final column representing the vector's fitness. In an N-dimensional problem, the HM would be represented as

$$\begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^N & | & w_1 \\ x_2^1 & x_2^2 & \cdots & x_2^N & | & w_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{\text{HMS}}^1 & x_{\text{HMS}}^2 & \cdots & x_{\text{HMS}}^N & | & w_{\text{HMS}} \end{bmatrix}, \tag{1}$$

where each row consists of N decision variables and the fitness score w ($[x^1, x^2, \ldots, x^N, w]$). Before optimisation starts, the HM is initialised with HMS randomly generated solution vectors. Depending on the problem, these vectors can also be randomly chosen around a seed point that may represent an area in the search space where the optimum is most likely to be found [3].

The step after initialisation is called improvisation. A new solution is improvised by using three rules: memory consideration, pitch adjustment, and random selection. Each decision variable is improvised separately, and any one of the three rules can be used for any variable. The harmony memory consideration rate (HMCR) is also one of the HS parameters that must be manually chosen. It controls how often the memory (HM) is taken into consideration during improvisation. For standard HS memory consideration means that the decisions variable's value is chosen directly from one of the solution vectors in the HM. A random number is generated for each decision variable. If it is smaller than the HMCR, the memory is taken into consideration; otherwise, a value is randomly chosen from the range of possible values for that dimension.

If the HM is taken into consideration, the improvised value is chosen randomly from one of the values in the HM. The pitch adjustment rate (PAR) is set during initialisation, and it controls the amount of *pitch adjustment* done when memory consideration is used. Another random number is generated. If it is smaller than the PAR, the improvised value is pitch adjusted using

$$x'_{\text{new}} = x_{\text{new}} + \text{rand}() \cdot \text{FW}, \tag{2}$$

where x'_{new} is the new pitch-adjusted value, x_{new} is the old value chosen using memory consideration, rand() is a random value between −1 and 1, and FW is the fret width parameter. The terminology has since been updated to better reflect the musical analogy [5] that controls the maximum variation in pitch adjustment and is one of the parameters that must be manually set.

Once a new value has been improvised, the memory is updated by comparing the new improvisation with the vector in the memory with the lowest fitness. If the new improvisation has a higher fitness, it replaces the vector with the lowest fitness. This process of improvisation and update continues iteratively until some stopping criterion is fulfilled, or the maximum number of iterations are reached.

To summarise, the main steps of the HS are as follows.

(1) Initialise the HM with possible solutions from the entire search space of the function to be optimised.

(2) Improvise a new solution vector using memory consideration, pitch adjustment, and random selection

(3) If the new improvisation is better than the worst solution in the HM, replace the worst solution with the new improvisation.

(4) Check the stopping criteria. If it has not been met, repeat Steps (2) and (3), or else proceed to Step (5).

(5) Return the best solution (highest fitness) in the HM as the optimal solution vector of the function.

This is a brief overview of HS. The interested reader is referred to [1, 5] for a more detailed explanation.

2.2. Exploratory Harmony Search. The exploratory harmony search (EHS) algorithm was the result of improving the performance of HS by focussing on the evolution of the population variance in the HM during optimisation. Das et al. realised that the exploratory power of HS can be maximised by dynamically adjusting the FW to be proportional to the standard deviation of the HM population [18].

This realisation comes from the following lemma that is proven in [18].

Lemma 1. *If the HMCR is chosen to be high (i.e., very near to 1), and the FW is chosen to be proportional to the standard deviation of the HM (i.e., FW* $\propto \sigma(HM) = \sqrt{\text{Var}(HM)}$*), then the expected population variance (without updating the HM) can grow exponentially over iterations.*

This lemma led them to redefine the FW as a dynamically adjusting parameter defined by $\text{FW} = k\sqrt{\text{Var}(\text{HM})}$, where k is a proportionality constant. The FW is therefore recalculated at each iteration based on the current HM variance, instead of using a constant value that must be chosen manually.

To ensure exponential variance growth, the PAR, HMCR, and k must also be chosen such that $((\text{HMS} - 1)/\text{HMS}) \cdot \text{HMCR} \cdot [1 + (1/3)k^2 \cdot \text{PAR}]$ is greater than 1. The proof of this statement can also be found in [18].

This simple change to HS results in an algorithm that is less likely to get stuck in local optima, requires less iterations to reach a desired accuracy, and frees the user from determining the best value for the FW parameter. However, one could argue that the FW parameter was simply replaced by the proportionality constant k. In their article, Das et al. investigate the effect that variations in k have on the performance of EHS. They determined that EHS is not sensitive to the choice of k and used a constant value ($k = 1.17$) in all their experiments. This contrasts with the choice of the FW in HS that is usually chosen to fit a particular objective function and can have a significant impact on performance if chosen poorly.

2.3. Self-Adaptive Harmony Search. In 2010, Wang and Huang developed a variant of HS that we refer to as the self-adaptive harmony search (SAHS) algorithm [19]. Their main focus was to alleviate the problem of choosing the best HS parameter values for a specific problem. SAHS does not require the FW and PAR parameters that need to be specified when using HS. However, SAHS like IHS requires that a minimum and maximum value for PAR be specified. Specifying a minimum and maximum for PAR is typically much easier than specifying an exact value, and it is more reliable since the performance is not as sensitive to changes in the range of PAR values as it is for a specific constant value.

SAHS is somewhat similar to IHS in that it also replaces the PAR with a range of PAR values. Like IHS, it also recalculates the current value of the PAR by linearly adjusting it between the minimum and maximum values based on the iteration count. However, it differs in that it *decreases* the PAR by starting at the maximum value and linearly decreasing it until the minimum is reached at the final iteration instead of *increasing* it like IHS does.

Unlike IHS, the FW parameter is not replaced by a range of FW values. Instead, it is replaced by a novel method of pitch adjustment that does not require an FW parameter. When pitch adjustment of an improvised decision variable is required, it is randomly adjusted for that decision variable between the minimum and maximum values found in the current HM, using the following equations:

$$\begin{aligned} x'_i &\longleftarrow x'_i + \left[\max\left(\text{HM}^i\right) - x'_i\right] \cdot r, \\ x'_i &\longleftarrow x'_i - \left[x'_i - \min\left(\text{HM}^i\right)\right] \cdot r, \end{aligned} \tag{3}$$

where x'_i is the new improvisation, $\max(\text{HM}^i)$ and $\min(\text{HM}^i)$ are the maximum and minimum values in the HM for the ith decision variables and r is a uniformly generated random number between 0 and 1. Each time pitch adjustment is performed, the improvisation is updated by randomly applying, with equal probability, one of these two equations.

This approach causes progressively smaller changes to be made to the new improvisation as $\max(\mathrm{HM}^i)$ and $\min(\mathrm{HM}^i)$ converge closer together with increasing iterations. Therefore, pitch adjustment begins as a rough operator making larger changes to favour exploration and then becomes a fine operator favouring exploitation as the optimiser converges closer to the optimum.

Another important difference between SAHS and HS is in the initialisation of the HM. In HS, the HM is initialised one decision variable at a time by randomly picking a value from a uniform distribution of values in the allowable range. However, this uniform distribution is only an approximation created by scaling the output from a pseudorandom number generator to the correct range of values. The resulting distribution is often unevenly distributed causing slow convergence or convergence to local optima. Instead of using a pseudorandom number generator, SAHS uses a low discrepancy sequence [20] to initialise the HM. Low discrepancy sequences are more evenly spread out in the search space than pseudorandom ones, causing the initial vectors in the HM to better represent the search space of the problem.

2.4. Dynamic Local-Best Harmony Search. dynamic local-best harmony search (DLHS) was developed by Pan et al. as an improvement to IHS and GHS that does not require the HMCR and PAR parameters [21]. The focus of DLHS can be roughly divided into three categories. First, the improvisation step was improved by basing new improvisations on the current best solution vector in the HM instead of a randomly chosen one. Second, it divides the HM into multiple independently optimised sub-HMs in an attempt to maintain greater diversity throughout the search process. Third, DLHS uses a self-learning parameter set list (PSL) to evolve the HMCR and PAR parameters into the optimal values without any user specification. The FW parameter is dynamically adjusted in a similar way to the method that IHS uses.

The modification of the improvisation process is inspired by GHS. Like GHS, the random solution vector used in memory consideration is replaced by the best solution vector in the HM. This tends to concentrate future improvisations around the current best solution which is likely a local optimum, and it may be the global optimum. To prevent possible convergence around local optima, the random solution vector that was discarded in memory consideration is instead used during pitch adjustment. So unlike the pitch adjustment of (2), DLHS uses the following equation:

$$x'_{\mathrm{new}} = x_{\mathrm{rand}} + \mathrm{rand}\,() \cdot \mathrm{FW}, \tag{4}$$

where x_{rand} is randomly chosen from the current values in the HM for a particular decision variable.

Before improvisation starts, the HM is randomly divided into m equally sized sub-HMs. Each sub-HM then uses its own small subset of solution vectors to independently converge onto a local (or possibly global) optimum. Due to the loss of diversity that comes with restricting the size of the individual HMs by splitting them up into sub-HMs, it is likely that convergence will not be towards the global optimum but rather toward a local one. To prevent this loss of diversity, information is allowed to exchange between sub-HMs with a frequency controlled by the regrouping schedule R. The regrouping schedule determines how often information exchange is allowed between sub-HMs by randomly regrouping the individual solution vectors of all sub-HMs into m new sub-HM configurations every R iteration. This regrouping simultaneously increases the diversity of all sub-HMs and allows the best solutions from the entire HM to be shared among sub-HMs.

Once the global optimum is identified, the injection of diversity through the regrouping operation does not further optimise, but it can result in inaccurate convergence. For this reason, DLHS enters the final phase, of the optimisation process after 90% of the iterations have been made. In this final phase the regrouping operation is halted, and a new HM is formed by combining the best three solution vectors from all sub-HMs into a single new HM. The new HM is then exclusively processed until the maximum number of iterations has been reached.

The HMCR and PAR parameter values are dynamically determined by selecting from a self-learning parameter set list (PSL). This process starts with the initialisation of the PSL by filling it with randomly generated HMCR and PAR values. HMCR values are generated from a uniform distribution where $\mathrm{HMCR} \in [0.9, 1.0]$ and PAR values are generated from a uniform distribution where $\mathrm{PAR} \in [0.0, 1.0]$. At the start of each iteration, one pair of HMCR and PAR values is removed from the PSL and used for that iteration. If the current iteration's improvisation resulted in the HM being updated, the current parameter pair is saved in the winning parameter set list (WPSL). This process continues until the PSL becomes empty. The PSL is then refilled by randomly selecting a pair from the WPSL 75% of the time and randomly generating a new pair 25% of the time. To prevent memory effects in the WPSL, the WPSL is emptied each time the PSL is refilled. The result of this is that the best parameter set is gradually learned, and it is specific to the objective function under consideration. The size of the PSL is set to 200 pairs. It was found that the effect that the size of the PSL has on performance is insignificant [21].

Unlike the HMCR and the PAR, the FW parameter is dynamically adjusted in a way that is similar to the way IHS adjusts the PAR. Like IHS, DLHS favours a large FW during the early iterations to encourage exploration of the search space, while a small FW is favoured during the final iterations to better exploit good solutions in the HM. The FW is therefore linearly decreased with increasing iterations using the following equation:

$$\mathrm{FW}(i) = \begin{cases} \mathrm{FW}_{\max} - \dfrac{\mathrm{FW}_{\max} - \mathrm{FW}_{\min}}{\mathrm{MI}} 2i & \text{if } i < \dfrac{\mathrm{MI}}{2} \\ \mathrm{FW}_{\min} & \text{otherwise,} \end{cases} \tag{5}$$

where $\mathrm{FW}_{\max}$ and $\mathrm{FW}_{\min}$ are the maximum and minimum values of the FW, i is the iteration number, and MI is the maximum number of iterations. Like IHS, this means that a minimum and maximum value for the FW has to be set before optimisation starts, but this is again assumed to be much easier than deciding on a single value for the FW.

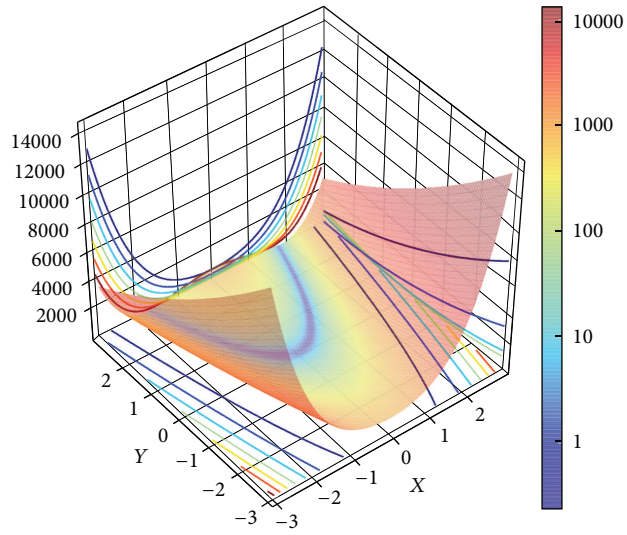


Figure 1: Rosenbrock's valley is illustrated here in two dimensions. Notice the parabolic valley that includes the global optimum at $[1, 1]$.

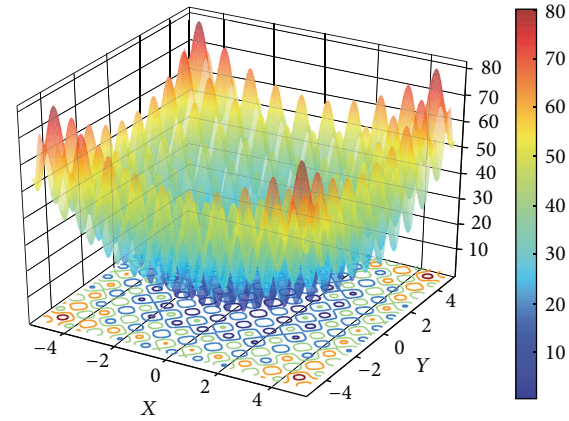


Figure 2: This illustration of Rastrigin's function shows its highly multimodal nature, and it is difficult to visually identify the global optima at $[0, 0]$.

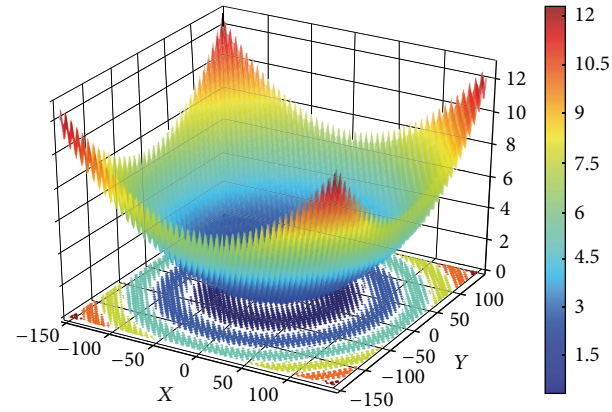


Figure 3: This is a two-dimensional surface plot of Griewangk's function. The global optima are at $[0, 0]$, where the function evaluates to 0.

3. Performance Analysis Using Five Benchmark Functions

3.1. Benchmark Functions. We start our performance analysis with a brief description of the five benchmark functions that we will be using to compare performance. All five of these functions were designed to be challenging to optimise for different reasons, and all have previously been used for benchmarking harmony search-based algorithms.

We start with a classic parametric optimisation function called Rosenbrock's valley [22]. This multidimensional problem has a global optimum inside a long narrow parabolic-shaped flat valley. The valley itself is easy to find since there are no other local minima anywhere in the search space, but converging onto the global optimum is difficult. Rosenbrock's valley is defined by the following equation:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} 100\left(x_{i+1} - x_i^2\right) + \left(1 - x_i\right)^2 \qquad -2.048 \le x_i \le 2.048. \tag{6}$$

Rosenbrock's valley has a global optimum at $x_i = 1\ \forall i$, where the function evaluates to 0. An illustration of Rosenbrock's valley implemented in two dimensions is seen in Figure 1.

Rastrigin's function uses cosine modulation to create a highly multimodal function that often causes optimisation algorithms to get stuck in the local optima without ever finding the global optimum [23]. However, it does have a single global optimum at $x_i = 0\ \forall i$, where the function evaluates to 0. Rastrigin's function is defined by

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}\left(x_i^2 - 10\cos\left(2\pi x_i\right)\right) \qquad -5.12 \le x_i \le 5.12. \tag{7}$$

An illustration of this function in two dimensions is seen in Figure 2. One can clearly see the multiple local minima in the contours overlaid on the floor of the surface plot.

Griewangk's function is similar to Rastrigin's function in that it also uses cosine modulation to create a highly multimodal function [23]. However, Griewangk's function is optimised over a larger search space and has different properties as one gets closer to the global optima situated at $x_i = 0\ \forall i$.

When one considers the entire search space, the function looks like a simple multidimensional unimodal parabola which could easily be minimised using gradient descent. As one moves closer to the optimum and starts to consider smaller areas, it becomes clear that the function is not as smooth as first thought, and it is in fact highly multimodal. This is clearly illustrated in Figure 3 where a surface plot of a two-dimensional implementation of Griewangk's function

is shown. The multimodal nature of Griewangk's function is also apparent from its defining equation which is as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad (8)$$
$$-600 \leq x_i \leq 600.$$

Ackley's Path function is another multimodal test function that is widely used to benchmark optimisation algorithms [24]. It uses a combination of exponentials and cosine modulation to create a search space with many local optima and a single global optimum at $x_i = 0\ \forall i$, where the function evaluates to 0. Ackley's Path function is illustrated in Figure 4 and is defined by the following equation:

$$f(\mathbf{x}) = 20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e \qquad (9)$$
$$-32.768 \leq x_i \leq 32.768.$$

The last test function that we consider is the Goldstein-Price function. This eighth-order polynomial is defined in two variables only and has global minima at $[0.0, -1.0]$ where the function evaluates to 3.0. This function has four local minima that lie close together, and the search space is usually limited to the $-2 \leq x_1, x_2 \leq 2$ cube. The Goldstein-Price function is illustrated in Figure 5 and is defined by the following polynomial:

$$f(\mathbf{x}) = \left(1 + (x_1 + x_2 + 1)^2 \times \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right) \times \left(30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right)$$
$$-32.768 \leq x_i \leq 32.768. \qquad (10)$$

3.2. Test Results. We compared the performance of the three HS variants together with the original HS algorithm using a series of test runs over all five benchmark functions. We implemented all the benchmark functions except for the Goldstein-Price function in both 30 and 100 dimensions to measure performance in both low- and high-dimensional problems. In functions with 100 dimensions, the algorithms were allowed to optimise for 500,000 function evaluations, while those in 30 dimensions were allowed to run for 100,000. The Goldstein-Price polynomial is only defined in two dimensions and therefore requires considerably less iterations to optimise. We allowed only 10,000 function evaluations when the Goldstein-Price polynomial was optimised.

In our first experiment, all HM variants were implemented using the parameters suggested by the original

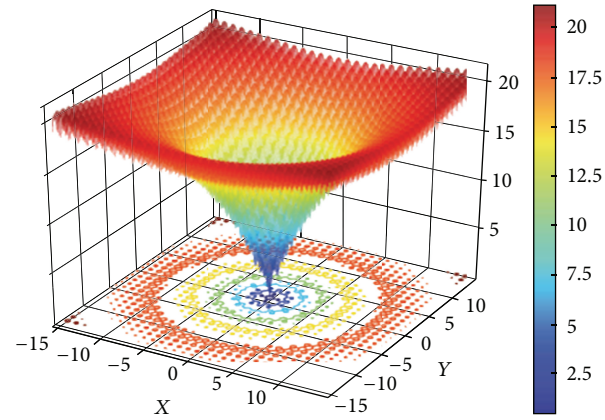


Figure 4: This is a two-dimensional surface plot of Ackley's function. The global optima is at $[0, 0]$ where the function evaluates to 0.

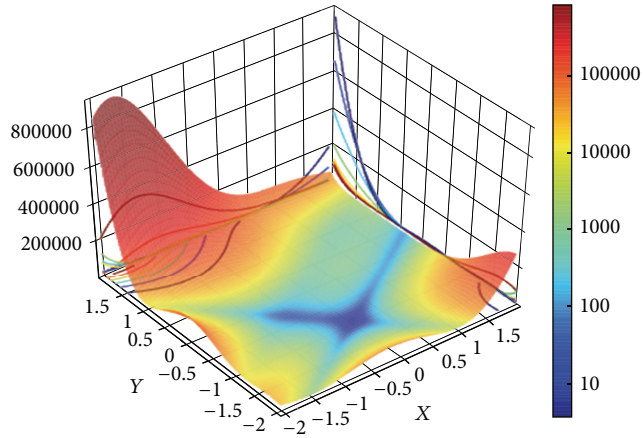


Figure 5: The Goldstein-Price polynomial is only defined in two variables and is shown here evaluated in the $-2 \leq x_1, x_2 \leq 2$ cube.

authors. The only exceptions were the parameter sets used in the original harmony search algorithm and that used for DLHS. Many different *optimal* parameter sets have been suggested for HS by researchers, and we chose to find our own set based on the best overall results after simulating all of five benchmark functions. For DLHS, the authors suggest 3 sub-HMs of size 3 which is much smaller than the 50 element HMs suggested by all the other HM variants [21]. We therefore opted to use 5 sub-HMs of size 10 instead, both because this resulted in better results for DLHS and also for a more fair comparison with the other variants. The parameter sets that we used in all our experiments are indicated in Table 1.

Each of the five benchmark functions were minimised using these parameters. The results for the 30 dimensional problems (together with the Goldstein-Price problem) are shown in Table 2. In each experiment, three values are given. The first is the average score calculated over 50 independently initialised runs. This is followed by the standard deviation and

Table 1: Parameter sets as suggested by the original authors.

	HS	EHS	DLHS	AHS
HMS	50.00	50.00	50.00	50.00
HMCR	0.99	0.99	—	0.99
PAR	0.33	0.33	—	—
PAR_{min}	—	—	—	0.00
PAR_{max}	—	—	—	1.00
FW	0.01	—	—	—
FW_{min}	—	—	0.0001	—
FW_{max}	—	—	(UB − LB)/200†	—
k (EHS)	—	1.17	—	—
R (DLHS)	—	—	50.00	—
m (DLHS)			5.00	

†UB and LB refer to the lower and upper bound of the component values.

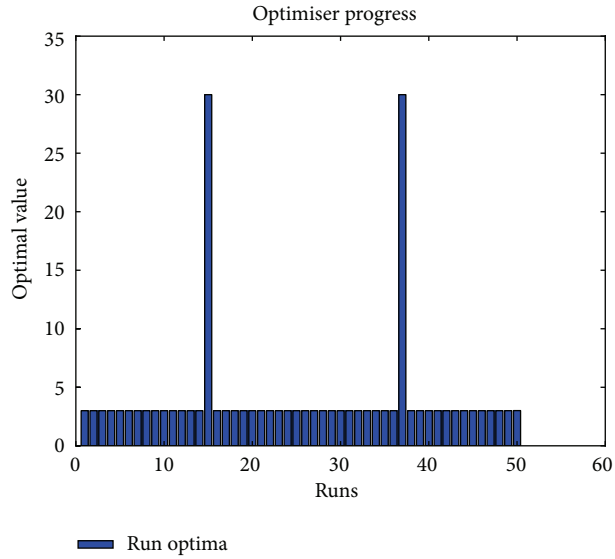


Figure 6: This bar graph shows the result of 50 independent runs of EHS on the Goldstein-Price polynomial. Notice that 48 of the 50 runs converged to the correct global minimum, while two converged to one of the local minima that is 10 times larger than the global minimum.

then the success rate defined as the number of successful hits on the global minimum within a 0.01 tolerance.

Notice that in many cases, more than one algorithm achieved 100% success rate (shown as 50 successful hits) on the global minimum. In these cases, we define the best results as the algorithm with both a 100% success rate and an average score closest to the global minimum. We repeat this experiment using the same parameter values on 100-dimensional problems. The average scores from the second experiment are shown in Table 3.

Some of these values seem surprising, and it is worth investigating why this is so. One would expect that the 2-dimensional Goldstein-Price function would be the easiest to minimise and that all the average scores should be very close to 3.0 which is the global optimum. However, as seen in Table 2, this is not the case for HS, DLHS, and EHS. These algorithms are not necessarily so much worse than AHS, but this indicates that some of the 50 runs that contributed to the average score converged to a local minimum that was far different from the global one. Since these local minima may be much larger than the global minimum, it can have a large effect on the average score. We see this clearly happening in the graph of Figure 6. The graph shows that only 2 of the 50 runs failed to converge to the exact global minimum, but because of these 2 the average score is significantly different than the global minimum.

A quick comparison of the results presented in Tables 2 and 3 suggests that the AHS algorithm is the overall best optimiser. In the 30-dimensional experiment, it performed the best on two of the five benchmark functions and was the only optimiser to achieve a 100% success rate on the Goldstein-Price polynomial. Its performance in optimising Ackley's, Griewank's and Rosenbrock's functions is very similar to that of EHS, and there are no significant differences that might indicate that one is clearly better than the other when only these three functions are considered. However, both completely failed at minimising Rosenbrock's function. It is intersting that the only successful run from all algorithms when optimising Rosenbrock's function was achieved by the original unmodified HS algorithm. This shows that the original HS algorithm can perform equally well and sometimes even better than modern variants when an optimised parameter set is used.

Surprising results are also seen in Rastrigin's column of Table 2. Like with Rosenbrock function, both EHS and AHS failed to optimise this function, but unlike Rosenbrock's the DLHS optimiser performs well here and has a much better hit ratio than any of the other variants. In the section that follows, we suggest a possible explanation for this result and investigate how one can use this to design an improved HS variant.

The results from the 100-dimensional problem summarised in Table 3 are very similar to that of the 30-dimensional problem. For Ackley's, Griewank's and Rosenbrock's functions, the performance of EHS and AHS is again the best and is not significantly different. Like in the 30-dimensional problem, both EHS and AHS failed in optimising Rastrigin's function which is again optimised much more successfully by DLHS.

4. Interpretation of Results

In this section, results from Tables 2 and 3 are used to show the best attributes of the HS variants and where they fail. In the previous section, we saw that EHS and AHS generally perform the best, but both failed to optimise Rastrigin's function. DLHS performs much better on Rastrigin's function but tends to get stuck in local optima as seen in the results from 30-dimensional Ackley's and the Goldstein-Price functions. The question is then why does DLHS perform so much better

Table 2: Average scores and standard deviations over 50 runs for 30-dimensional problems. All optimisers were allowed to run for 100,000 function evaluations, except when the Goldstein-Price polynomial was being minimised. In that case, 10,000 evaluations were allowed. The best results from each benchmark function are indicated in bold.

	Ackley	Griewank	Rosenbrock	Rastrigin	Goldstein-Price[†]
HS					
Best	0.0066	0.0322	**30.7287**	0.0142	6.8464
Standard deviation	0.0006	0.0334	**22.6459**	0.0027	9.3353
Hits	50	48	**1**	2	42
DLHS					
Best	0.8464	0.0170	32.7372	**0.2003**	6.2444
Standard deviation	0.5173	0.0180	22.8117	**0.4200**	8.7724
Hits	2	50	0	**31**	44
EHS					
Best	0.0008	**0.0006**	28.9183	15.4119	4.0809
Standard deviation	0.0024	**0.0022**	10.7493	10.8267	5.2910
Hits	50	**50**	0	5	48
AHS					
Best	**0.0003**	0.0042	26.5041	1.4819	**3**
Standard deviation	**0.0020**	0.0111	0.5624	0.9055	**0**
Hits	**50**	50	0	1	**50**

[†]Goldstein-Price was implemented in two dimensions as it is only defined in two.

Table 3: Average scores and standard deviations over 50 runs for 100-dimensional problems. The optimisers were allowed to run for 500,000 function evaluations.

	Ackley	Griewank	Rosenbrock	Rastrigin[†]
HS				
Best	0.0142	0.0050	162.5829	0.2280
Standard deviation	0.0004	0.0080	49.5829	0.0130
Hits	50	34	0	0
DLHS				
Best	0.2281	0.0004	142.4515	**1.3140**
Standard deviation	0.0130	0.0070	39.5126	**3.1364**
Hits	50	50	0	**32**
EHS				
Best	0.0009	0.0001	**97.2555**	388.2894
Standard deviation	0.0006	0.0007	**7.4711**	20.5949
Hits	50	50	**0**	0
AHS				
Best	**0.0003**	**0**	96.4333	6.7471
Standard deviation	**0.0006**	**0**	0.2943	2.3279
Hits	**50**	**50**	0	0

[†]The accuracy tolerance for awarding a successful hit on the global minimum was decreased to 0.1 for 100-dimensional Rastrigin's function.

than EHS and AHS on Rastrigin when it is worse in all other examples?

We start to answer that question by investigating what novel improvement these HS variants contribute to the final result. The main contribution of both EHS and AHS is the dynamic adjustment of the aggressiveness of the pitch adjustment operator. EHS does this by dynamically adjusting the value of the FW based on the variance of the values currently in the HM. AHS also determines the amount of pitch adjustment by analysing the values in the HM, but it does so without needing to directly calculate the variance. In both cases the effect is that pitch adjustment is more

aggressive at the beginning of the optimisation process and continues to make smaller and smaller adjustments as the variance in the HM decreases due to convergence.

The advantage that each of these approaches brings is that the pitch adjustment operator dynamically shifts from an aggressive mode that favours exploration of the search space at the start of optimisation to a less aggressive mode that favours exploitation of possible minima that were discovered during the aggressive mode. However, this also means that when the optimiser converges to local minima, the drop in variance of values in the HM causes the pitch adjustment to favour exploitation, making it unlikely that the optimiser will escape the local minima and find the global one. Premature convergence to local optima is therefore a weakness of these methods and may explain why even the original HS scored better than EHS and AHS on the optimisation of Rastrigin's function.

AHS attempts to minimise the effect of this weakness using the adjustment of the PAR. By linearly decreasing the PAR from a very high to a very low value, the exploration phase is lengthened due to the aggressive and frequent pitch adjustment caused by a high PAR and a high variance in the HM at the start of optimisation. In practice this does have a positive effect on the results and may explain the advantage AHS had over EHS in the Goldstein-Price experiment. It was, however, still not enough to prevent the poor performance when optimising Rastrigin's function.

DLHS uses a completely different approach, first maximising the HM diversity by dividing it up into sub-HMs and then by dynamically adjusting the PAR, FW, and HMCR values to fit the function being optimised and the optimisation progress. The sub-HM idea has been used successfully before to maximise diversity in harmony search (see [4, 25]). This idea of separating the population of candidate solutions into groups that converge independently is also used successfully in other evolutionary algorithms, and it was originally used in genetic algorithms in what became known as island model parallel genetic algorithms [26, 27].

The weakness of this approach, however, is often slow convergence due to function evaluations that are wasted in subpopulations (sub-HMs) and end up converging to local optima or not converging at all. Since the subpopulations converge independently, it is likely that they will converge to different optima. This is actually the desired behaviour since the motivation of this approach was the maximisation of diversity in the candidate solutions. However, the question then becomes when to consolidate the independent results from each subpopulation so the best candidates can be exploited for more accurate convergence.

The answer to this question lies in the classic tradeoff between maximising the convergence speed and minimising the probability of premature convergence to local optima. If sub-populations converge independently for too long, iterations are wasted on results that will finally be discarded. On the other hand, when results are consolidated too quickly diversity among the sub-populations is lost, and this increases the risk of premature convergence.

In DLHS, this tradeoff is left as a parameter, namely the regrouping schedule, and it is up to the user to decide how aggressively the sub-HMs should be consolidated. It is difficult to choose the best value for this parameter, and the optimum value is likely problem specific. This means that a single good choice is unlikely. The original authors of DLHS suggested a value of 50 iterations for the regrouping schedule. Rastrigin's function was one of the functions they used to test their algorithm [21]. This may explain the good results that DLHS produced on Rastrigin's function.

Another major contributing factor to the DLHS results is the dynamic adjustment of the PAR, FW and HMCR parameters. Like EHS and AHS, DLHS starts with a large FW, and decreases it as the optimiser progresses. However, unlike EHS and AHS, it does not take the variance in the current HM into account but instead linearly decreases it proportionally to the iteration count. This is similar to the way the IHS algorithm operates (see [9]) and to the way AHS linearly decreases the PAR. We believe that not taking the variance in the HM into account is a weakness of DLHS, especially since the consolidation of the sub-HMs can have a dramatic effect on the HM causing changes in the variance that do not follow a linearly decreasing trend.

However, the PAR and HMCR are adjusted in a novel way that potentially has the largest impact on the quality of the final results. To our knowledge, DLHS is also the only HM variant to dynamically adjust the HMCR. The HMCR is a parameter that has previously been shown by several researchers, including the authors of AHS and EHS, to have a large effect on the quality of the final results [10, 18, 19].

In our own analysis (see Section 2.4) of the values that end in the WPSL after several thousand iterations, we noticed that the self-adapting process that controls the PAR and HMCR values tends to favour very small PARs (<0.01) and large HMCRs (>0.99). This seems to verify our choice of optimal values for the HMCR and the PAR, but it also suggests that the maximum PAR used in AHS should optimally be much smaller than 1.0.

Finally, one should also be very aware of the fact that because the HM is divided into sub-HMs, the actual HM improvisation steps are performed using a much smaller HMS than the 50 candidates that were chosen for the other HM variants. Like the HMCR, HMS also makes a large difference to the quality of the results, but it is one of the hardest parameters to choose [10, 19]. Researchers initially recommended a small HMS (<15), but the authors of AHS showed that AHS performs better with a larger HMS which was also the conclusion made by the authors of EHS. However, in our own experiments, we found that AHS performs much better on Rastrigin's function, even to the point of being better than DLHS, when the HMS is decreased to 10 instead of the suggested 50. However, this caused a decrease in performance when optimising Griewank's and Ackley's functions. This then suggests that the HMS should be chosen to fit the problem, or it should be dynamically adjusted like the PAR and FW.

It is currently not known how the HMS should be adjusted or modelled to a specific problem. The practical effects that this would have on the HM would be much greater than simply changing the HMCR or the PAR since candidates will need to be added or removed from the HM. A further

problem would then be determining which candidate should be removed when the HMS is decreased, and how should a new candidate be created when the HMS is increased. If we are to develop an HM variant that performs well on all our benchmark functions without requiring a fine-tuned parameter set unique to each problem, this is the type of question that will need to be answered.

5. Generalised Adaptive Harmony Search

Our approach to designing an HS variant that performs well on all our benchmark functions is based on the AHS algorithm. We call it the generalised adaptive harmony search (GAHS) algorithm. We chose to base GAHS on AHS because of its simplicity and its good performance on all but Rastrigin's function. As we have pointed out, AHS can be made to perform well on Rastrigin given the right parameter set, but no single parameter set seems to perform well on all the benchmark functions.

We therefore need to either find a way to dynamically adjust the parameters during optimisation to fit the particular problem (particularly the HMS) or augment the improvisation process some other way that would allow for the same benefits. Our first attempt at addressing this was to split the HM into sub-HMs in the same way that DLHS does. It was our hope that this would both decrease the effect of HMS, which we already know has a positive effect on the results for Rastrigin's function and increase diversity in the HM enough to compensate for AHS's tendency for premature convergence. We used the AHS pitch adjustment operator and linearly decreased the PAR with the intention that this would compensate for DLHS's slow convergence.

Our initial results showed that using sub-HMs in AHS improved the performance when minimising Rastrigin's function but the convergence rate also slowed considerably, resulting in poor performance when minimising Griewank and Ackley. Several parameter sets were tried which indicated that a longer regrouping period was needed to compensate for AHS's tendency for premature convergence. Larger sub-HMs resulted in good performance in Griewank and Ackley, while Rastrigin needed small sub-HMs like those DLHS uses. However, no single parameter set that performed well on all our benchmark functions could be found.

A better approach was needed, and it was clear that using sub-HMs will either result in slow convergence or the need to fine-tune a parameter set to a specific problem. The focus should be on the HMS parameter as this parameter has the greatest effect on the performance of AHS when Rastrigin's function is minimised.

We propose to find a small collection of parameter sets, in particular HMS and HMCR values, that includes at least one set that will result in good performance on any of our benchmark functions. An instance of AHS will then be started for each parameter set and run for a fraction of the total function evaluations allowed. Once this has been done for each parameter set, the best results from each HM is compared, and the instance with the worst result is dropped. The process then repeats until only the best instance with the most optimal parameter set remains. This instance is then left to converge until the remainder of the allowable function evaluations have been reached. All AHS instances are initialised using a low discrepancy sequence (see Section 2.3), and regroupings of candidates between separate instances are never done. Each instance therefore converges independently.

This approach requires the addition of two important parameters. The first, called the period length (PL), is the number of iterations that a particular instance is allowed to run before it is stopped and evaluated to determine whether more function evaluations are spent on it or whether it is abandoned. The second is the *parameter set collection size* (PSCS) which is the number of parameter sets in the collection and is therefore also the number of AHS instances that are compared. These two parameters are important because together they determine the percentage of the total function evaluations wasted on finding the optimal parameter set and how many are left for convergence to the global optimum. Since one function evaluation is done during each iteration of AHS, the total number of iterations spent on instances that do not contribute to the final result (wasted iterations) can be calculated using the following equation:

$$\text{Wasted iterations} = \text{PL} \sum_{i=2}^{\text{PSCS}} (i-1)\,. \tag{11}$$

To maximise the number of useful iterations, the PL and the PSCS is kept as low as possible. However, if the PSCS are too small, an important parameter set may be overlooked. If the PL is too small, some of the AHS instances may be discarded before an accurate estimate of their quality can be measured.

Through experimentation, we determined that an optimal parameter set collection contains three parameter sets. All three parameter sets are equal to that suggested by the original authors of AHS and are identical except for the HMS. We use three different values for the HMS, namely, 10, 30, and 50. A good value for the PL was found to be 10% of the total number of iterations. This means according to (11) that 30% of the total iterations is spent on finding the optimal parameter set and 70% for convergence to the global optimum.

The performance of GAHS as defined in the previous paragraphs is compared with the other HS variants in Table 4. The benchmark functions were again implemented in 100 dimensions. We repeat the results from Table 3 for comparison.

It is seen from this comparison that GAHS performs even better than DLHS on Rastrigin's function as well as AHS on the other benchmark functions. It still performs poorly on Rosenbrock's function. Even when we used a fine-tuned parameter set specific to Rosenbrock's function, we could not improve the results. We suspect that since EHS also converges to this point that large local optima exist in that area, and that the nature of this function is such that escaping from this local optimum is very difficult for harmony search-based optimisers.

It is also clear that some weaknesses that hinder GAHS still remain. Since GAHS is only a generalised form of AHS,

Table 4: In this table, the performance of GAHS is compared with the other HS variants. All scores are averaged over 50 runs, and all functions except Goldstein-Price are implemented in 100 dimensions. The optimisers were allowed to run for 500,000 function evaluations except for Goldstein-Price which was allowed only 10,000 evaluations. The best results are again indicated in bold.

	Ackley	Griewank	Rosenbrock	Rastrigin[†]	Goldstein-Price[%]
HS					
Best	0.0142	0.0050	162.5829	0.2280	6.8464
Standard deviation	0.0004	0.0080	49.5829	0.0130	9.3353
Hits	50	34	0	0	42
DLHS					
Best	0.2281	0.0004	142.4515	1.3140	6.2444
Standard deviation	0.0130	0.0070	39.5126	3.1364	8.7724
Hits	50	50	0	32	44
EHS					
Best	0.0009	0.0001	97.2555	388.2894	4.0809
Standard deviation	0.0006	0.0007	7.4711	20.5949	5.2910
Hits	50	50	0	0	48
AHS					
Best	**0.0003**	**0**	96.4333	6.7471	**3**
Standard deviation	**0.0006**	**0**	0.2943	2.3279	**0**
Hits	**50**	**50**	0	0	**50**
GAHS					
Best	0.0263	0.0049	**96.4034**	**0.0633**	**3**
Standard deviation	0.0125	0.0021	**0.7833**	**0.0337**	**0**
Hits	50	49	**0**	**44**	**50**

[†]The accuracy tolerance for awarding a successful hit on the global minimum was decreased to 0.1 for 100-dimensional Rastrigin's function.
[%]Goldstein-Price was implemented in two dimensions as it is only defined in two.

one would expect that it would perform at least as well as AHS in all examples. However, as we see in Table 4, with Ackley's and Griewank's functions that this is not true. The reason for this is that AHS enters a slow convergence phase once the basin of attraction around local optima is found because of the drop in HM variance that causes the pitch adjustment operator to make smaller and smaller adjustments. This means that AHS requires several thousand iterations to find the exact optimum once it has found the basin of attraction. It therefore uses all the available iterations and keeps converging until the very end. If 30% of its available iterations is removed, as is done in GAHS, the final result will not be as accurate as it would have been otherwise.

Another weakness that affects the convergence speed is in the choice of the period length. The period length has to be long enough so that an accurate estimate can be made of the performance of the parameter set. This can only be an estimate since the true performance may only become apparent once an instance has fully converged, and we already know that AHS keeps converging even in the last few iterations. This means that our method of measuring the quality of a parameter set is inherently biased to those sets that perform well at the beginning of the optimisation process. This means that those sets with a small HMS and a large HMCR will frequently get chosen, causing valuable parameter sets that may later produce the best results to be discarded during the first round of eliminations.

An illustration of this effect is shown in Figure 7. The graph traces the convergence of GAHS over 500,000 iterations as it minimises Ackley's function. The fitness score of the best and the worst candidates that the optimiser currently has in the HM is recorded at each iteration and shown as two separate lines on the graph. We see a downward trend indicating convergence followed by discontinuous jumps that indicate the abandonment of one instance of AHS and the start of another. Note that there are three distinct convergence paths present in the first 150,000 iterations. These represent the three AHS instances that correspond to three parameter sets. The first one that clearly converges the quickest over the first 50,000 iterations corresponds to the parameter set where the HMS = 10. The two that follow are those that correspond to the HMS = 40 and HMS = 50 parameter sets. Since we know from the previous experiments that AHS performs better than GAHS and uses an HMS of 50, we know that GAHS would finally give the best results if the third instance was kept through both elimination rounds. However, due to its slow rate of convergence during the first 50,000 iterations, it is eliminated first, leaving a weaker parameter set to finally converge.

6. Conclusions

After investigating the results from comparing some of the best performing HS variants available today, we noticed that

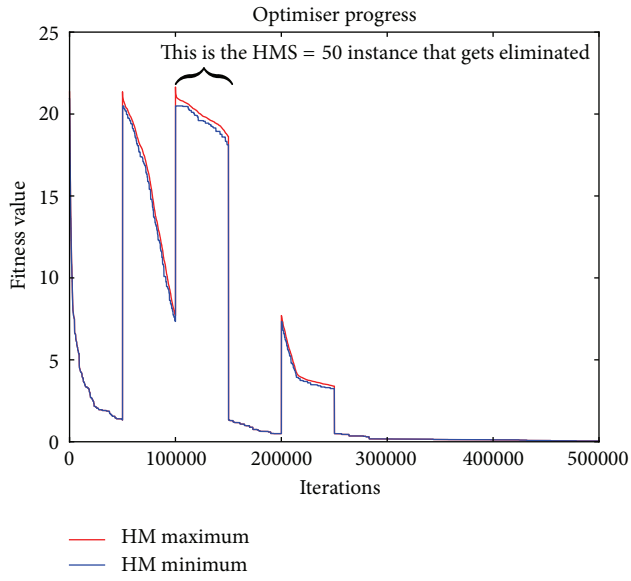


Figure 7: This graph traces the convergence of the GAHS algorithm over 500,000 iterations as it minimises Ackley's function. The fitness value, which is simply the candidate evaluated using Ackley's function, of the best and worst candidates in the HM is indidcated at the end of each iteration. The best candiate is the HM minimum indicated by the blue line; the worst candidate is the HM maximum indicated by the red line.

the overall quality of these algorithms was similar. We also noticed several weaknesses that prevent any one of them from being the best choice. Authors of these HS variants suggested parameter sets that are nearly optimal; however, some problems still required the parameters to be fine-tuned to a specific problem if good results were to be expected.

We suggested possible ways that some of these weaknesses can be addressed, and we proposed the generalised adaptive harmony search (GAHS) algorithm as a generalised version of AHS. GAHS performs well over a larger range of problems. By using 5 benchmark functions, we demonstrated that GAHS performs as well as AHS on most problems, and it can also produce good results on problems where AHS fails to find the global optimum.

However, some open questions still remain as a topic of future research. Our experiments clearly pointed out that there are categories of optimisation problems and that a parameter set that works well for one might fail completely for another one. For example, what makes Rastrigin's function so difficult to minimise using a parameter set that works well for all the other benchmark functions? What is special about Rastrigin's function that causes it to require a small HM to effectively optimise, and can this special attribute be detected before optimisation is attempted? A similar question can be asked about Rosenbrock's function. Why can none of the HM variants find the global optimum even when using a fine-tuned parameter set? Is there a weakness that is inherent in harmony search-based optimisers that make them poor optimisers for that category of problems to which Rosenbrock's function belongs, or can one augment harmony search in some way to address this weakness? Harmony search is still a relatively recent addition to the family of evolutionary algorithms, and finding answers to these type of questions is needed if harmony search is to become fully established as a leading heuristic optimiser.

Acknowledgment

This work was supported by the Gachon University research fund of 2013 (GCU-2013-R084).

References

[1] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[2] O. M. Alia, R. Mandava, D. Ramachandram, and M. E. Aziz, "Dynamic fuzzy clustering using harmony search with application to image segmentation," in *Proceedings of the 9th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT '09)*, pp. 538–543, December 2009.

[3] J. Fourie, S. Mills, and R. Green, "Harmony filter: a robust visual tracking system using the improved harmony search algorithm," *Image and Vision Computing*, vol. 28, no. 12, pp. 1702–1716, 2010.

[4] J. Fourie, R. Green, and S. Mills, "Counterpoint harmony search: an accurate algorithm for the blind deconvolution of binary images," in *Proceedings of the International Conference on Audio, Language and Image Processing (ICALIP '10)*, pp. 1117–1122, Shanghai, China, November 2010.

[5] Z. W. Geem, *Recent Advances in Harmony Search Algorithm*, vol. 270 of *Studies in Computational Intelligence*, Springer, Berlin, Germany, 1st edition, 2010.

[6] Z. W. Geem, "Harmony search algorithm for solving Sudoku," in *Knowledge-Based Intelligent Information and Engineering Systems*, B. Apolloni, R. Howlett, and L. Jain, Eds., Lecture Notes in Computer Science, pp. 371–378, Springer, Berlin, Germany, 2010.

[7] Z. W. Geem, K. S. Lee, and Y. Park, "Application of harmony search to vehicle routing," *American Journal of Applied Sciences*, vol. 2, no. 12, pp. 1552–1557, 2005.

[8] Z. W. Geem and J.-Y. Choi, "Music composition using harmony search algorithm," in *Proceedings of the EvoWorkshops*, pp. 593–600, Springer, Berlin, Germany.

[9] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[10] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation*, vol. 198, no. 2, pp. 643–656, 2008.

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

[12] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, and Y. Alizadeh, "Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 33–40, pp. 3080–3091, 2008.

[13] Z. W. Geem, "Particle-swarm harmony search for water network design," *Engineering Optimization*, vol. 41, no. 4, pp. 297–311, 2009.

[14] H. Jiang, Y. Liu, and L. Zheng, "Design and simulation of simulated annealing algorithm with harmony search," in *Advances in Swarm Intelligence*, Y. Tan, Y. Shi, and K. Tan, Eds., Lecture Notes in Computer Science, pp. 454–460, Springer, Berlin, Germany, 2010.

[15] W. S. Jang, H. I. Kang, and B. H. Lee, "Hybrid simplex-harmony search method for optimization problems," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 4157–4164, June 2008.

[16] L. P. Li and L. Wang, "Hybrid algorithms based on harmony search and differential evolution for global optimization," in *Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC '09)*, pp. 271–278, ACM, New York, NY, USA, June 2009.

[17] O. M. Alia and R. Mandava, "The variants of the harmony search algorithm: an overview," *Artificial Intelligence Review*, vol. 36, no. 1, pp. 49–68, 2011.

[18] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, and B. K. Panigrahi, "Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 41, no. 1, pp. 89–106, 2011.

[19] C. M. Wang and Y. F. Huang, "Self-adaptive harmony search algorithm for optimization," *Expert Systems with Applications*, vol. 37, no. 4, pp. 2826–2837, 2010.

[20] H. Faure, "Good permutations for extreme discrepancy," *Journal of Number Theory*, vol. 42, no. 1, pp. 47–56, 1992.

[21] Q. K. Pan, P. N. Suganthan, J. J. Liang, and M. F. Tasgetiren, "A local-best harmony search algorithm with dynamic subpopulations," *Engineering Optimization*, vol. 42, no. 2, pp. 101–117, 2010.

[22] H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, 1960.

[23] A. Törn and A. Pilinskas, *Global Optimization*, Lecture Notes in Computer Science 350, Springer, Berlin, Germany, 1989.

[24] D. H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*, vol. SECS28 of *The Kluwer International Series in Engineering and Computer Science*, Kluwer Academic Publishers, Boston, Mass, USA, 1987.

[25] J. Fourie, S. Mills, and R. Green, "Visual tracking using the harmony search algorithm," in *Proceedings of the 23rd International Conference Image and Vision Computing New Zealand (IVCNZ '08)*, Queenstown, New Zealand, November 2008.

[26] T. Niwa and M. Tanaka, "Analysis on the island model parallel genetic algorithms for the genetic drifts," in *Selected papers from the Second Asia-Pacific Conference on Simulated Evolution and Learning on Simulated Evolution and Learning (SEAL'98)*, vol. 98, pp. 349–356, Springer, London, UK, 1999.

[27] B. Artyushenko, "Analysis of global exploration of island model genetic algorithm," in *Proceedings of the 10th International Conference on Experience of Designing and Application of CAD Systems in Microelectronics (CADSM '09)*, pp. 280–281, February 2009.