Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 497514, 20 pages
http://dx.doi.org/10.1155/2014/497514

Research Article

A Cuckoo Search Algorithm for Multimodal Optimization

Erik Cuevas and Adolfo Reyna-Orta

Departamento de Electronica, Universidad de Guadalajara, CUCEI, Avenida Revolución 1500, 44430 Guadalajara, JAL, Mexico

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

Received 23 April 2014; Accepted 5 May 2014; Published 22 July 2014

Academic Editor: Xin-She Yang

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

Interest in multimodal optimization is expanding rapidly, since many practical engineering problems demand the localization of multiple optima within a search space. On the other hand, the cuckoo search (CS) algorithm is a simple and effective global optimization algorithm which can not be directly applied to solve multimodal optimization problems. This paper proposes a new multimodal optimization algorithm called the multimodal cuckoo search (MCS). Under MCS, the original CS is enhanced with multimodal capacities by means of (1) the incorporation of a memory mechanism to efficiently register potential local optima according to their fitness value and the distance to other potential solutions, (2) the modification of the original CS individual selection strategy to accelerate the detection process of new local minima, and (3) the inclusion of a depuration procedure to cyclically eliminate duplicated memory elements. The performance of the proposed approach is compared to several state-of-the-art multimodal optimization algorithms considering a benchmark suite of fourteen multimodal problems. Experimental results indicate that the proposed strategy is capable of providing better and even a more consistent performance over existing well-known multimodal algorithms for the majority of test problems yet avoiding any serious computational deterioration.

1. Introduction

Optimization is a field with applications in many areas of science, engineering, economics, and others, where mathematical modelling is used [1]. In general, the goal is to find an acceptable solution of an objective function defined over a given search space. Optimization algorithms are usually broadly divided into deterministic and stochastic ones [2]. Since deterministic methods only provide a theoretical guarantee of locating a local minimum for the objective function, they often face great difficulties in solving optimization problems [3]. On the other hand, stochastic methods are usually faster in locating a global optimum [4]. Moreover, they adapt easily to black-box formulations and extremely ill-behaved functions, whereas deterministic methods usually rest on at least some theoretical assumptions about the problem formulation and its analytical properties (such as Lipschitz continuity) [5].

Evolutionary algorithms (EA), which are considered to be members of the stochastic group, have been developed by a combination of rules and randomness that mimics several natural phenomena. Such phenomena include evolutionary processes such as the evolutionary algorithm (EA) proposed by Fogel et al. [6], de Jong [7], and Koza [8]; the genetic algorithm (GA) proposed by Holland [9] and Goldberg [10]; the artificial immune system proposed by de Castro and von Zuben [11]; and the differential evolution algorithm (DE) proposed by Storn and Price [12]. Some other methods which are based on physical processes include simulated annealing proposed by Kirkpatrick et al. [13], the electromagnetism-like algorithm proposed by Birbil and Fang [14], and the gravitational search algorithm proposed by Rashedi et al. [15]. Also, there are other methods based on the animal-behavior phenomena such as the particle swarm optimization (PSO) algorithm proposed by Kennedy and Eberhart [16] and the ant colony optimization (ACO) algorithm proposed by Dorigo et al. [17].

Most of research work on EA aims for locating the global optimum [18]. Despite its best performance, a global optimum may be integrated by parameter values that are considered impractical or prohibitively expensive, limiting their adoption into a real-world application. Therefore, from a practical point of view, it is desirable to have access to not only the global optimum but also as many local optima as possible (ideally all of them). Under such circumstances, a local optimum with acceptable performance quality and

modest cost may be preferred over a costly global solution with marginally better performance [19]. The process of finding the global optimum and multiple local optima is known as multimodal optimization.

EA perform well for locating a single optimum but fail to provide multiple solutions [18]. Several methods have been introduced into the EA scheme to achieve multimodal optimization, such as fitness sharing [20–22], deterministic crowding [23], probabilistic crowding [22, 24], clustering based niching [25], clearing procedure [26], species conserving genetic algorithm [27], and elitist-population strategies [28]. However, most of these methods have difficulties that need to be overcome before they can be employed successfully to multimodal applications. Some identified problems include difficulties in tuning some niching parameters, difficulties in maintaining discovered solutions in a run, extra computational overheads, and poor scalability when dimensionality is high. An additional problem represents the fact that such methods are devised for extending the search capacities of popular EA such as GA and PSO, which fail in finding a balance between exploration and exploitation, mainly for multimodal functions [29]. Furthermore, they do not explore the whole region effectively and often suffer premature convergence or loss of diversity.

As alternative approaches, other researchers have employed artificial immune systems (AIS) to solve multimodal optimization problems. Some examples are the clonal selection algorithm [30] and the artificial immune network (AiNet) [31, 32]. Both approaches use operators and structures which attempt to find multiple solutions by mimicking the natural immune system's behavior.

Every EA needs to address the issue of exploration and exploitation of the search space [33]. Exploration is the process of visiting entirely new points of a search space, whilst exploitation is the process of refining those points within the neighborhood of previously visited locations in order to improve their solution quality. Pure exploration degrades the precision of the evolutionary process but increases its capacity to find new potential solutions. On the other hand, pure exploitation allows refining existent solutions but adversely driving the process to local optimal solutions.

Multimodal optimization requires a sufficient amount of exploration of the population agents in hyperspace so that all the local and global attractors can be successfully and quickly detected [34, 35]. However, an efficient multimodal optimization algorithm should exhibit not only good exploration tendency but also good exploitative power, especially during the last stages of the search, because it must ensure accurately a distributed convergence to different optima in the landscape. Therefore, the ability of an EA to find multiple solutions depends on its capacity to reach a good balance between the exploitation of found-so-far elements and the exploration of the search space [36]. So far, the exploration-exploitation dilemma has been an unsolved issue within the framework of EA.

Recently, a novel nature-inspired algorithm, called the cuckoo search (CS) algorithm [37], has been proposed for solving complex optimization problems. The CS algorithm is based on the obligate brood-parasitic strategy of some cuckoo species. One of the most powerful features of CS is the use of Lévy flights to generate new candidate solutions. Under this approach, candidate solutions are modified by employing many small changes and occasionally large jumps. As a result, CS can substantially improve the relationship between exploration and exploitation, still enhancing its search capabilities [38]. Recent studies show that CS is potentially far more efficient than PSO and GA [39]. Such characteristics have motivated the use of CS to solve different sorts of engineering problems such as mesh generation [40], embedded systems [41], steel frame design [42], scheduling problems [43], thermodynamics [44], and distribution networks [45].

This paper presents a new multimodal optimization algorithm called the multimodal cuckoo search (MCS). The method combines the CS algorithm with a new memory mechanism which allows an efficient registering of potential local optima according to their fitness value and the distance to other potential solutions. The original CS selection strategy is mainly conducted by an elitist decision where the best individuals prevail. In order to accelerate the detection process of potential local minima in our method, the selection strategy is modified to be influenced by individuals that are contained in the memory mechanism. During each generation, eggs (individuals) that exhibit different positions are included in the memory. Since such individuals could represent to the same local optimum, a depuration procedure is also incorporated to cyclically eliminate duplicated memory elements. The performance of the proposed approach is compared to several state-of-the-art multimodal optimization algorithms considering a benchmark suite of 14 multimodal problems. Experimental results indicate that the proposed strategy is capable of providing better and even more consistent performance over the existing well-known multimodal algorithms for the majority of test problems avoiding any serious computational deterioration.

The paper is organized as follows. Section 2 explains the cuckoo search (CS) algorithm, while Section 3 presents the proposed MCS approach. Section 4 exhibits the experimental set and its performance results. Finally, Section 5 establishes some concluding remarks.

2. Cuckoo Search (CS) Method

CS is one of the latest nature-inspired algorithms, developed by Yang and Deb [37]. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights [46], rather than by simple isotropic random walks. Recent studies show that CS is potentially far more efficient than PSO and GA [39].

Cuckoo birds lay their eggs in the nests of other host birds (usually other species) with amazing abilities such as selecting nests containing recently laid eggs and removing existing eggs to increase the hatching probability of their own eggs. Some of the host birds are able to combat this parasitic behavior of cuckoos and throw out the discovered alien eggs or build a new nest in a distinct location. This cuckoo breeding analogy is used to develop the CS algorithm. Natural systems are complex, and therefore they cannot be modeled exactly by a computer algorithm in its basic form. Simplification of

natural systems is necessary for successful implementation in computer algorithms. Yang and Deb [39] simplified the cuckoo reproduction process into three idealized rules.

(1) An egg represents a solution and is stored in a nest. An artificial cuckoo can lay only one egg at a time.

(2) The cuckoo bird searches for the most suitable nest to lay the eggs in (solution) to maximize its eggs' survival rate. An elitist selection strategy is applied, so that only high-quality eggs (best solutions near the optimal value) which are more similar to the host bird's eggs have the opportunity to develop (next generation) and become mature cuckoos.

(3) The number of host nests (population) is fixed. The host bird can discover the alien egg (worse solutions away from the optimal value) with a probability of $p_a \in [0,1]$, and these eggs are thrown away or the nest is abandoned and a completely new nest is built in a new location. Otherwise, the egg matures and lives to the next generation. New eggs (solutions) laid by a cuckoo choose the nest by Lévy flights around the current best solutions.

From the implementation point of view, in the CS operation, a population, $\mathbf{E}^k$ ($\{\mathbf{e}_1^k, \mathbf{e}_2^k, \ldots, \mathbf{e}_N^k\}$), of N eggs (individuals) is evolved from the initial point ($k = 0$) to a total gen number iterations ($k = 2 \cdot \text{gen}$). Each egg, $\mathbf{e}_i^k$ ($i \in [1, \ldots, N]$), represents an n-dimensional vector, $\{e_{i,1}^k, e_{i,2}^k, \ldots, e_{i,n}^k\}$, where each dimension corresponds to a decision variable of the optimization problem to be solved. The quality of each egg, $\mathbf{e}_i^k$ (candidate solution), is evaluated by using an objective function, $f(\mathbf{e}_i^k)$, whose final result represents the fitness value of $\mathbf{e}_i^k$.Three different operators define the evolution process of CS: (A) Lévy flight, (B) replacement of some nests by constructing new solutions, and (C) elitist selection strategy.

2.1. Lévy Flight (A). One of the most powerful features of cuckoo search is the use of Lévy flights to generate new candidate solutions (eggs). Under this approach, a new candidate solution, $\mathbf{e}_i^{k+1}$ ($i \in [1, \ldots, N]$), is produced by perturbing the current $\mathbf{e}_i^k$ with a change of position $\mathbf{c}_i$. In order to obtain $\mathbf{c}_i$, a random step, $\mathbf{s}_i$, is generated by a symmetric Lévy distribution. For producing $\mathbf{s}_i$, Mantegna's algorithm [47] is employed as follows:

$$\mathbf{s}_i = \frac{\mathbf{u}}{|\mathbf{v}|^{1/\beta}}, \tag{1}$$

where $\mathbf{u}$ ($\{u_1, \ldots, u_n\}$) and $\mathbf{v}$ ($\{v_1, \ldots, v_n\}$) are n-dimensional vectors and $\beta = 3/2$. Each element of $\mathbf{u}$ and $\mathbf{v}$ is calculated by considering the following normal distributions:

$$u \sim N\left(0, \sigma_u^2\right), \qquad v \sim N\left(0, \sigma_v^2\right),$$
$$\sigma_u = \left(\frac{\Gamma\left(1+\beta\right) \cdot \sin\left(\pi \cdot \beta/2\right)}{\Gamma\left(\left(1+\beta\right)/2\right) \cdot \beta \cdot 2^{(\beta-1)/2}}\right)^{1/\beta}, \qquad \sigma_v = 1, \tag{2}$$

where $\Gamma(\cdot)$ represents the gamma distribution. Once $\mathbf{s}_i$ has been calculated, the required change of position $\mathbf{c}_i$ is computed as follows:

$$\mathbf{c}_i = 0.01 \cdot \mathbf{s}_i \oplus \left(\mathbf{e}_i^k - \mathbf{e}^{\text{best}}\right), \tag{3}$$

where the product $\oplus$ denotes entrywise multiplications whereas $\mathbf{e}^{\text{best}}$ is the best solution (egg) seen so far in terms of its fitness value. Finally, the new candidate solution, $\mathbf{e}_i^{k+1}$, is calculated by using

$$\mathbf{e}_i^{k+1} = \mathbf{e}_i^k + \mathbf{c}_i. \tag{4}$$

2.2. Replacement of Some Nests by Constructing New Solutions (B). Under this operation, a set of individuals (eggs) are probabilistically selected and replaced with a new value. Each individual, $\mathbf{e}_i^k$ ($i \in [1, \ldots, N]$), can be selected with a probability of $p_a \in [0,1]$. In order to implement this operation, a uniform random number, r_1, is generated within the range $[0,1]$. If r_1 is less than p_a, the individual $\mathbf{e}_i^k$ is selected and modified according to (5). Otherwise, $\mathbf{e}_i^k$ remains without change. This operation can be resumed by the following model:

$$\mathbf{e}_i^{k+1} = \begin{cases} \mathbf{e}_i^k + \text{rand} \cdot \left(\mathbf{e}_{d_1}^k - \mathbf{e}_{d_2}^k\right), & \text{with probability } p_a, \\ \mathbf{e}_i^k, & \text{with probability } \left(1 - p_a\right), \end{cases} \tag{5}$$

where rand is a random number normally distributed, whereas d_1 and d_2 are random integers from 1 to N.

2.3. Elitist Selection Strategy (C). After producing $\mathbf{e}_i^{k+1}$ either by operator A or by operator B, it must be compared with its past value $\mathbf{e}_i^k$. If the fitness value of $\mathbf{e}_i^{k+1}$ is better than $\mathbf{e}_i^k$, then $\mathbf{e}_i^{k+1}$ is accepted as the final solution. Otherwise, $\mathbf{e}_i^k$ is retained. This procedure can be resumed by the following statement:

$$\mathbf{e}_i^{k+1} = \begin{cases} \mathbf{e}_i^{k+1}, & \text{if } f\left(\mathbf{e}_i^{k+1}\right) < f\left(\mathbf{e}_i^k\right), \\ \mathbf{e}_i^k, & \text{otherwise.} \end{cases} \tag{6}$$

This elitist selection strategy denotes that only high-quality eggs (best solutions near the optimal value) which are more similar to the host bird's eggs have the opportunity to develop (next generation) and become mature cuckoos.

2.4. Complete CS Algorithm. CS is a relatively simple algorithm with only three adjustable parameters: p_a, the population size, N, and the number of generations gen. According to Yang and Deb [39], the convergence rate of the algorithm is not strongly affected by the value of p_a and it is suggested to use $p_a = 0.25$. The operation of CS is divided in two parts: initialization and the evolution process. In the initialization ($k = 0$), the first population, $\mathbf{E}^0$ ($\{\mathbf{e}_1^0, \mathbf{e}_2^0, \ldots, \mathbf{e}_N^0\}$), is produced. The values, $\{e_{i,1}^0, e_{i,2}^0, \ldots, e_{i,n}^0\}$, of each individual, $\mathbf{e}_i^k$, are randomly and uniformly distributed between the prespecified

(1) **Input:** p_a, N and gen
(2) Initialize $\mathbf{E}^0(k = 0)$
(3) **until** $(k = 2 \cdot \text{gen})$
(5) $\mathbf{E}^{k+1} \leftarrow \text{OperatorA}(\mathbf{E}^k)$ Section 2.1
(6) $\mathbf{E}^{k+1} \leftarrow \text{OperatorC}(\mathbf{E}^k, \mathbf{E}^{k+1})$ Section 2.3
(7) $\mathbf{E}^{k+2} \leftarrow \text{OperatorB}(\mathbf{E}^{k+1})$ Section 2.2
(8) $\mathbf{E}^{k+1} \leftarrow \text{OperatorC}(\mathbf{E}^{k+1}, \mathbf{E}^{k+2})$ Section 2.3
(9) **end until**

Algorithm 1: Cuckoo search (CS) algorithm.

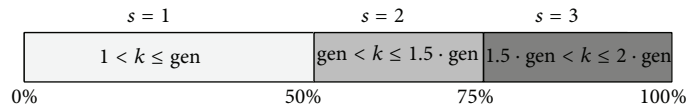


Figure 1: Division of the evolution process according to MCS.

lower initial parameter bound, b_j^{low}, and the upper initial parameter bound, b_j^{high}. One has

$$e_{i,j}^0 = b_j^{\text{low}} + \text{rand} \cdot \left(b_j^{\text{high}} - b_j^{\text{low}}\right); \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, n. \tag{7}$$

In the evolution process, operators A (Lévy flight), B (replacement of some nests by constructing new solutions), and C (elitist selection strategy) are iteratively applied until the number of iterations $k = 2 \cdot \text{gen}$ has been reached. The complete CS procedure is illustrated in Algorithm 1.

From Algorithm 1, it is important to remark that the elitist selection strategy (C) is used two times, just after operator A or operator B is executed.

3. The Multimodal Cuckoo Search (MCS)

In CS, individuals emulate eggs which interact in a biological system by using evolutionary operations based on the breeding behavior of some cuckoo species. One of the most powerful features of CS is the use of Lévy flights to generate new candidate solutions. Under this approach, candidate solutions are modified by employing many small changes and occasionally large jumps. As a result, CS can substantially improve the relationship between exploration and exploitation, still enhancing its search capabilities. Despite such characteristics, the CS method still fails to provide multiple solutions in a single execution. In the proposed MCS approach, the original CS is adapted to include multimodal capacities. In particular, this adaptation contemplates (1) the incorporation of a memory mechanism to efficiently register potential local optima according to their fitness value and the distance to other potential solutions, (2) the modification of the original CS individual selection strategy to accelerate the detection process of new local minima, and (3) the inclusion of a depuration procedure to cyclically eliminate duplicated memory elements.

In order to implement these modifications, the proposed MCS divides the evolution process in three asymmetric states. The first state ($s = 1$) includes 0 to 50% of the evolution process. The second state ($s = 2$) involves 50 to 75%. Finally, the third state ($s = 3$) lasts from 75 to 100%. The idea of this division is that the algorithm can react in a different manner depending on the current state. Therefore, in the beginning of the evolutionary process, exploration can be privileged, while, at the end of the optimization process, exploitation can be favored. Figure 1 illustrates the division of the evolution process according to MCS.

The next sections examine the operators suggested by MCS as adaptations of CS to provide multimodal capacities. These operators are (D) the memory mechanism, (E) new selection strategy, and (F) depuration procedure.

3.1. Memory Mechanism (D). In the MCS evolution process, a population, $\mathbf{E}^k$ ($\{\mathbf{e}_1^k, \mathbf{e}_2^k, \ldots, \mathbf{e}_N^k\}$), of N eggs (individuals) is evolved from the initial point ($k = 0$) to a total gen number iterations ($k = 2 \cdot \text{gen}$). Each egg, $\mathbf{e}_i^k$ ($i \in [1, \ldots, N]$), represents an n-dimensional vector, $\{e_{i,1}^k, e_{i,2}^k, \ldots, e_{i,n}^k\}$, where each dimension corresponds to a decision variable of the optimization problem to be solved. The quality of each egg, $\mathbf{e}_i^k$ (candidate solution), is evaluated by using an objective function, $f(\mathbf{e}_i^k)$, whose final result represents the fitness value of $\mathbf{e}_i^k$. During the evolution process, MCS maintains also the best, $\mathbf{e}^{\text{best},k}$, and the worst, $\mathbf{e}^{\text{worst},k}$, eggs seen so far, such that

$$\mathbf{e}^{\text{best},k} = \underset{i \in \{1,2,\ldots,N\}, a \in \{1,2,\ldots,k\}}{\arg\min} \left(f\left(\mathbf{e}_i^a\right)\right), \qquad \mathbf{e}^{\text{worst},k} = \underset{i \in \{1,2,\ldots,N\}, a \in \{1,2,\ldots,k\}}{\arg\min} \left(f\left(\mathbf{e}_i^a\right)\right). \tag{8}$$

Global and local optima possess two important characteristics: (1) they have a significant good fitness value and (2)

they represent the best fitness value inside a determined neighborhood. Therefore, the memory mechanism allows efficiently registering potential global and local optima during the evolution process, involving a memory array, $\mathbf{M}$, and a storage procedure. $\mathbf{M}$ stores the potential global and local optima, $\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_T\}$, during the evolution process, with T being the number of elements so far that are contained in the memory $\mathbf{M}$. On the other hand, the storage procedure indicates the rules that the eggs, $\{\mathbf{e}_1^k, \mathbf{e}_2^k, \ldots, \mathbf{e}_N^k\}$, must fulfill in order to be captured as memory elements. The memory mechanism operates in two phases: initialization and capture.

3.1.1. Initialization Phase. This phase is applied only once within the optimization process. Such an operation is achieved in the null iteration ($k = 0$) of the evolution process. In the initialization phase, the best egg, $\mathbf{e}_B$, of $\mathbf{E}^0$, in terms of its fitness value, is stored in the memory $\mathbf{M}$ ($\mathbf{m}_1$=$\mathbf{e}_B$), where $\mathbf{e}_B = \arg\min_{i\in\{1,2,\ldots,N\}}(f(\mathbf{e}_i^0))$, for a minimization problem.

3.1.2. Capture Phase. This phase is applied from the first ($k = 1$) iteration to the last iteration ($k = 2, 3, \ldots, 2 \cdot \text{gen}$), at the end of each operator (A and B). At this stage, eggs, $\{\mathbf{e}_1^k, \mathbf{e}_2^k, \ldots, \mathbf{e}_N^k\}$, corresponding to potential global and local optima are efficiently registered as memory elements, $\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_T\}$, according to their fitness value and the distance to other potential solutions. In the operation, each egg, $\mathbf{e}_i^k$, of $\mathbf{E}^k$ is tested in order to evaluate if it must be captured as a memory element. The test considers two rules: (1) significant fitness value rule and (2) nonsignificant fitness value rule.

Significant Fitness Value Rule. Under this rule, the solution quality of $\mathbf{e}_i^k$ is evaluated according to the worst element, $\mathbf{m}^{\text{worst}}$, that is contained in the memory $\mathbf{M}$, where $\mathbf{m}^{\text{worst}} = \arg\max_{i\in\{1,2,\ldots,T\}}(f(\mathbf{m}_i))$, in case of a minimization problem. If the fitness value of $\mathbf{e}_i^k$ is better than $\mathbf{m}^{\text{worst}}(f(\mathbf{e}_i^k) < f(\mathbf{m}^{\text{worst}}))$, $\mathbf{e}_i^k$ is considered potential global and local optima. The next step is to decide whether $\mathbf{e}_i^k$ represents a new optimum or it is very similar to an existent memory element, $\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_T\}$ (if it is already contained in the memory $\mathbf{M}$). Such a decision is specified by an acceptance probability function, $\Pr(\delta_{i,u}, s)$, that depends, on one side, on the distances $\delta_{i,u}$ from $\mathbf{e}_i^k$ to the nearest memory element $\mathbf{m}_u$ and, on the other side, on the current state s of the evolution process (1, 2, and 3). Under $\Pr(\delta_{i,u}, s)$, the probability that $\mathbf{e}_i^k$ would be part of $\mathbf{M}$ increases as the distance $\delta_{i,u}$ enlarges. Similarly, the probability that $\mathbf{e}_i^k$ would be similar to an existent memory element $\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_T\}$ increases as $\delta_{i,u}$ decreases. On the other hand, the indicator s that relates a numeric value with the state of the evolution process is gradually modified during the algorithm to reduce the likelihood of accepting inferior solutions. The idea is that in the beginning of the evolutionary process (exploration), large distance differences can be considered, while only small distance differences are tolerated at the end of the optimization process.

In order to implement this procedure, the normalized distance $\delta_{i,q}$ ($q \in [1, \ldots, T]$) is calculated from $\mathbf{e}_i^k$ to all the elements of the memory $\mathbf{M}$ $\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_T\}$. $\delta_{i,q}$ is computed as follows:

$$\delta_{i,q} = \sqrt{\left(\frac{e_{i,1}^k - m_{q,1}}{b_1^{\text{high}} - b_1^{\text{low}}}\right)^2 + \left(\frac{e_{i,2}^k - m_{q,2}}{b_2^{\text{high}} - b_2^{\text{low}}}\right)^2 + \cdots + \left(\frac{e_{i,n}^k - m_{q,n}}{b_n^{\text{high}} - b_n^{\text{low}}}\right)^2}, \tag{9}$$

where $\{m_{q,1}, m_{q,2}, \ldots, m_{q,n}\}$ represent the n components of the memory element $\mathbf{m}_q$, whereas b_j^{high} and b_j^{low} indicate the low j parameter bound and the upper j parameter bound ($j \in [1, \ldots, n]$), respectively. One important property of the normalized distance $\delta_{i,q}$ is that its values fall into the interval $[0, 1]$.

By using the normalized distance $\delta_{i,q}$ the nearest memory element $\mathbf{m}_u$ to $\mathbf{e}_i^k$ is defined, with $\mathbf{m}_u = \arg\min_{j\in\{1,2,\ldots,T\}}(\delta_{i,j})$. Then, the acceptance probability function $\Pr(\delta_{i,u}, s)$ is calculated by using the following expression:

$$\Pr(\delta_{i,u}, s) = (\delta_{i,u})^s. \tag{10}$$

In order to decide whether $\mathbf{e}_i^k$ represents a new optimum or it is very similar to an existent memory element, a uniform random number r_1 is generated within the range $[0, 1]$. If r_1 is less than $\Pr(\delta_{i,u}, s)$, the egg $\mathbf{e}_i^k$ is included in the memory $\mathbf{M}$ as a new optimum. Otherwise, it is considered that $\mathbf{e}_i^k$ is similar to $\mathbf{m}_u$. Under such circumstances, the memory $\mathbf{M}$ is updated by the competition between $\mathbf{e}_i^k$ and $\mathbf{m}_u$, according to their corresponding fitness values. Therefore, $\mathbf{e}_i^k$ would replace $\mathbf{m}_u$ in case $f(\mathbf{e}_i^k)$ is better than $f(\mathbf{m}_u)$. On the other hand, if $f(\mathbf{m}_u)$ is better than $f(\mathbf{e}_i^k)$, $\mathbf{m}_u$ remains with no change. The complete procedure of the significant fitness value rule can be resumed by the following statement:

$$\mathbf{M} = \begin{cases} \mathbf{m}_{T+1} = \mathbf{e}_i^k, \\ \quad \text{with probability } \Pr(\delta_{i,u}, s), \\ \mathbf{m}_u = \mathbf{e}_i^k \quad \text{if } f(\mathbf{e}_i^k) < f(\mathbf{m}_u), \\ \quad \text{with probability } 1 - \Pr(\delta_{i,u}, s). \end{cases} \tag{11}$$

In order to demonstrate the significant fitness value rule process, Figure 2 illustrates a simple minimization problem that involves a two-dimensional function, $f(\mathbf{x})$ ($\mathbf{x} = \{x_1, x_2\}$). As an example, it assumed a population, $\mathbf{E}^k$, of two different particles ($\mathbf{e}_1^k$,$\mathbf{e}_2^k$), a memory with two memory

(1) **Input:** $\mathbf{e}_i^k, \mathbf{e}^{\text{best},k}, \mathbf{e}^{\text{worst},k}$
(2) Calculate $p(\mathbf{e}_i^k, \mathbf{e}^{\text{best},k}, \mathbf{e}^{\text{worst},k}) = 1 - \left(f\left(\mathbf{e}_i^k\right) - f\left(\mathbf{e}^{\text{best},k}\right)\right) / \left(f\left(\mathbf{e}^{\text{worst},k}\right) - f\left(\mathbf{e}^{\text{best},k}\right)\right)$
(3) Calculate $P(p) = \begin{cases} p & 0.5 \le p \le 1 \\ 0 & 0 \le p < 0.5 \end{cases}$
(5) **if** $(\text{rand}(0,1) \le P)$ **then**
(6) $\quad \mathbf{e}_i^k$ is considered a local optimum — With probability P
(7) **else**
(8) $\quad \mathbf{e}_i^k$ is ignored — With probability $1 - P$
(9) **end if**

Algorithm 2: Nonsignificant fitness value rule procedure.

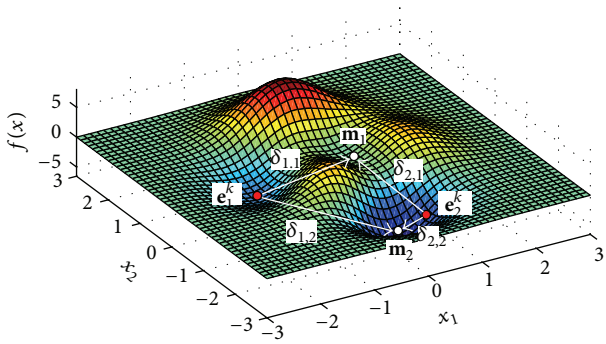


Figure 2: Graphical illustration of the significant fitness value rule process.

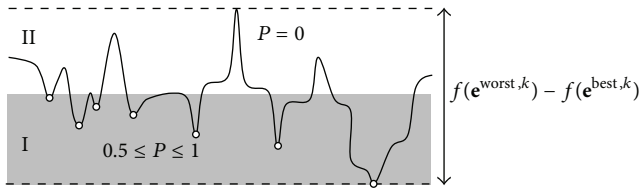


Figure 3: Effect of the probability function P in a simple example.

elements ($\mathbf{m}_1$,$\mathbf{m}_2$), and the execution of the first state ($s = 1$). According to Figure 2, both particles $\mathbf{e}_1^k$ and $\mathbf{e}_2^k$ maintain a better fitness value than $\mathbf{m}_1$, which possesses the worst fitness value of the memory elements. Under such conditions, the significant fitness value rule must be applied to both particles. In case of $\mathbf{e}_1^k$, the first step is to calculate the correspondent distances $\delta_{1,1}$ and $\delta_{1,2}$. $\mathbf{m}_1$ represents the nearest memory element to $\mathbf{e}_1^k$. Then, the acceptance probability function $\Pr(\delta_{1,1}, 1)$ is calculated by using (10). Since the value of $\Pr(\delta_{1,1}, 1)$ is high, there exists a great probability that $\mathbf{e}_1^k$ becomes the next memory element ($\mathbf{m}_3 = \mathbf{e}_1^k$). On the other hand, for $\mathbf{e}_2^k$, $\mathbf{m}_2$ represents the nearest memory element. As $\Pr(\delta_{2,2}, 1)$ is very low, there exists a great probability that $\mathbf{e}_2^k$ competes with $\mathbf{m}_2$ for a place within $\mathbf{M}$. In such a case, $\mathbf{m}_2$ remains with no change considering that $f(\mathbf{m}_2) < f(\mathbf{e}_2^k)$.

Nonsignificant Fitness Value Rule. Different to the significant fitness value rule, the nonsignificant fitness value rule allows capturing local optima with low fitness values. It operates if the fitness value of $\mathbf{e}_i^k$ is worse than $\mathbf{m}^{\text{worst}}$ ($f(\mathbf{e}_i^k) \ge f(\mathbf{m}^{\text{worst}})$). Under such conditions, it is necessary, as a first step, to test which particles could represent local optima and which must be ignored as a consequence of their very low fitness value. Then, if the particle represents a possible local optimum, its inclusion inside the memory $\mathbf{M}$ is explored.

The decision on whether $\mathbf{e}_i^k$ represents a new local optimum or not is specified by a probability function, P, which is based on the relationship between $f(\mathbf{e}_i^k)$ and the so far valid fitness value interval ($f(\mathbf{e}^{\text{worst},k}) - f(\mathbf{e}^{\text{best},k})$). Therefore, the probability function P is defined as follows:

$$p\left(\mathbf{e}_i^k, \mathbf{e}^{\text{best},k}, \mathbf{e}^{\text{worst},k}\right) = 1 - \frac{f\left(\mathbf{e}_i^k\right) - f\left(\mathbf{e}^{\text{best},k}\right)}{f\left(\mathbf{e}^{\text{worst},k}\right) - f\left(\mathbf{e}^{\text{best},k}\right)},$$
$$P(p) = \begin{cases} p, & 0.5 \le p \le 1, \\ 0, & 0 \le p < 0.5, \end{cases} \tag{12}$$

where $\mathbf{e}^{\text{best},k}$ and $\mathbf{e}^{\text{worst},k}$ represent the best and worst eggs seen so far, respectively. In order to decide whether $\mathbf{p}_i^k$ represents a new local optimum or it must be ignored, a uniform random number, r_2, is generated within the range $[0, 1]$. If r_2 is less than P, the egg $\mathbf{e}_i^k$ is considered to be a new local optimum. Otherwise, it must be ignored. Under P, the so far valid fitness value interval ($f(\mathbf{e}^{\text{worst},k}) - f(\mathbf{e}^{\text{best},k})$) is divided into two sections: I and II (see Figure 3). Considering this division, the function P assigns a valid probability (greater than zero) only to those eggs that fall into the zone of the best individuals (part I) in terms of their fitness value. Such a probability value increases as the fitness value improves. The complete procedure can be reviewed in Algorithm 2.

If the particle represents a possible local optimum, its inclusion inside the memory $\mathbf{M}$ is explored. In order to consider if $\mathbf{e}_i^k$ could represent a new memory element, another procedure that is similar to the significant fitness value rule process is applied. Therefore, the normalized distance $\delta_{i,q}$ ($q \in [1, \ldots, T]$) is calculated from $\mathbf{p}_i^k$ to all the elements of the memory $\mathbf{M}$ $\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_T\}$, according to (9). Afterwards, the nearest distance $\delta_{i,u}$ to $\mathbf{e}_i^k$ is determined.

Then, by using $\Pr(\delta_{i,u}, s)$ (10), the following rule can be thus applied:

$$\mathbf{M} = \begin{cases} \mathbf{m}_{T+1} = \mathbf{e}_i^k, & \text{with probability } \Pr\left(\delta_{i,u}, s\right), \\ \text{no change}, & \text{with probability } 1 - \Pr\left(\delta_{i,u}, s\right). \end{cases} \tag{13}$$

Under this rule, a uniform random number, r_3, is generated within the range $[0, 1]$. If r_3 is less than $\Pr(\delta_{i,u}, s)$, the egg $\mathbf{e}_i^k$ is included in the memory $\mathbf{M}$ as a new optimum. Otherwise, the memory does not change.

3.2. New Selection Strategy (E). The original CS selection strategy is mainly conducted by an elitist decision where the best individuals in the current population prevail. Such an operation, defined in this paper as operator C (Section 2.3), is executed two times, just after operators A and B in the original CS method. This effect allows incorporating interesting convergence properties to CS when the objective considers only one optimum. However, in case of multiple-optimum detection, such a strategy is not appropriate. Therefore, in order to accelerate the detection process of potential local minima in our method, the selection strategy is modified to be influenced by the individuals contained in the memory $\mathbf{M}$.

Under the new selection procedure (operator E), the final population $\mathbf{E}^{k+1}$ is built by considering the first N element from the memory $\mathbf{M}$ instead of using the best individuals between the currents $\mathbf{E}^{k+1}$ and $\mathbf{E}^k$. In case of the number of elements in $\mathbf{M}$ is less than N, the rest of the individuals are completed by considering the best elements from the current $\mathbf{E}^{k+1}$.

3.3. Depuration Procedure (F). During the evolution process, the memory $\mathbf{M}$ stores several individuals (eggs). Since such individuals could represent the same local optimum, a depuration procedure is incorporated at the end of each state s (1, 2, and 3) to eliminate similar memory elements. The inclusion of this procedure allows (a) reducing the computational overhead during each state and (b) improving the search strategy by considering only significant memory elements.

Memory elements tend to concentrate on optimal points (good fitness values), whereas element concentrations are enclosed by areas holding bad fitness values. The main idea in the depuration procedure is to find the distances among concentrations. Such distances, considered as depuration ratios, are later employed to delete all elements inside them, except for the best element in terms of their fitness values.

The method used by the depuration procedure in order to determine the distance between two concentrations is based on the element comparison between the concentration corresponding to the best element and the concentration of the nearest optimum in the memory. In the process, the best element $\mathbf{m}^{\text{best}}$ in the memory is compared to a memory element, $\mathbf{m}_b$, which belongs to one of both concentrations (where $\mathbf{m}^{\text{best}} = \arg\min_{i\in\{1,2,\ldots,T\}}(f(\mathbf{m}_i))$). If the fitness value of the medium point, $f((\mathbf{m}^{\text{best}} + \mathbf{m}_b)/2)$, between both is not worse than both, $(f(\mathbf{m}^{\text{best}}), f(\mathbf{m}_b))$, the element $\mathbf{m}_b$ is part of the same concentration of $\mathbf{m}^{\text{best}}$. However, if $f((\mathbf{m}^{\text{best}}+\mathbf{m}_b)/2)$ is worse than both, the element $\mathbf{m}_b$ is considered as part of the nearest concentration. Therefore, if $\mathbf{m}_b$ and $\mathbf{m}^{\text{best}}$ belong to different concentrations, the Euclidian distance between $\mathbf{m}_b$ and $\mathbf{m}^{\text{best}}$ can be considered as a depuration ratio. In order to avoid the unintentional deletion of elements in the nearest concentration, the depuration ratio D_R is lightly shortened. Thus, the depuration ratio r is defined as follows:

$$D_R = 0.85 \cdot \left\| \mathbf{m}^{\text{best}} - \mathbf{m}_b \right\|. \tag{14}$$

The proposed depuration procedure only considers the depuration ratio r between the concentration of the best element and the nearest concentration. In order to determine all ratios, preprocessing and postprocessing methods must be incorporated and iteratively executed.

The preprocessing method must (1) obtain the best element $\mathbf{m}^{\text{best}}$ from the memory in terms of its fitness value, (2) calculate the Euclidian distances from the best element to the rest of the elements in the memory, and (3) sort the distances according to their magnitude. This set of tasks allows identification of both concentrations: the one belonging to the best element and that belonging to the nearest optimum, so they must be executed before the depuration ratio D_R calculation. Such concentrations are represented by the elements with the shortest distances to $\mathbf{m}^{\text{best}}$. Once D_R has been calculated, it is necessary to remove all the elements belonging to the concentration of the best element. This task is executed as a postprocessing method in order to configure the memory for the next step. Therefore, the complete depuration procedure can be represented as an iterative process that at each step determines the distance of the concentration of the best element with regard to the concentration of the nearest optimum.

A special case can be considered when only one concentration is contained within the memory. This case can happen because the optimization problem has only one optimum or because all the other concentrations have been already detected. Under such circumstances, the condition where $f((\mathbf{m}^{\text{best}} + \mathbf{m}_b)/2)$ is worse than $f(\mathbf{m}^{\text{best}})$ and $f(\mathbf{m}_b)$ would be never fulfilled.

In order to find the distances among concentrations, the depuration procedure is conducted in Procedure 1.

At the end of the above procedure, the vector $\mathbf{Y}$ will contain the depurated memory which would be used in the next state or as a final result of the multimodal problem.

In order to illustrate the depuration procedure, Figure 4 shows a simple minimization problem that involves two different optimal points (concentrations). As an example, it assumed a memory, $\mathbf{M}$, with six memory elements whose positions are shown in Figure 4(a). According to the depuration procedure, the first step is (1) to build the vector $\mathbf{Z}$ and (2) to calculate the corresponding distance $\Delta_{1,j}^a$ among the elements. Following such operation, the vector $\mathbf{Z}$ and the set of distances are configured as $\mathbf{Z} = \{\mathbf{m}_5, \mathbf{m}_1, \mathbf{m}_3, \mathbf{m}_4, \mathbf{m}_6, \mathbf{m}_2\}$ and $\{\Delta_{1,2}^1, \Delta_{1,3}^2, \Delta_{1,5}^3, \Delta_{1,4}^4, \Delta_{1,6}^5\}$, respectively. Figure 4(b) shows the configuration of $\mathbf{X}$ where, for sake of easiness, only the two distances $\Delta_{1,2}^1$ and $\Delta_{1,5}^3$ have been represented. Then,

(1) Define two new temporal vectors **Z** and **Y**. The vector **Z** will hold the results of the iterative operations whereas **Y** will contain the final memory configuration. The vector **Z** is initialized with the elements of **M** that have been sorted according to their fitness values, so that the first element represents the best one. On other hand, **Y** is initialized empty.
(2) Store the best element $\mathbf{z}_1$ of the current **Z** in **Y**.
(3) Calculate the Euclidian distances $\Delta_{1,j}$ between $\mathbf{z}_1$ and the rest of elements from **Z** $(j \in \{2, \ldots, |\mathbf{Z}|\})$, where $|\mathbf{Z}|$ represents the number of elements in **Z**.
(4) Sort the distances $\Delta_{1,j}$ according to their magnitude. Therefore, a new index a is incorporated to each distance $\Delta^a_{1,j}$, where a indicate the place of $\Delta_{1,j}$ after the sorting operation. ($a = 1$ represents the shortest distance).
(5) Calculate the depuration ratio D_R:

for $q = 1$ **to** $|\mathbf{Z}| - 1$
Obtain the element $\mathbf{z}_j$ corresponding to the distance $\Delta^q_{1,j}$
Compute $f((\mathbf{z}_1 + \mathbf{z}_j)/2)$
if $(f((\mathbf{z}_1 + \mathbf{z}_j)/2) > f(\mathbf{z}_1)$ and $f((\mathbf{z}_1 + \mathbf{z}_j)/2) > f(\mathbf{z}_j))$
$D_R = 0.85 \cdot \|\mathbf{x}_1 - \mathbf{x}_j\|$
break
end if
if $q = |\mathbf{Z}| - 1$
There is only one concentration
end if
end for

(6) Remove all elements inside D_R from **Z**.
(7) Sort the elements of **Z** according to their fitness values.
(8) Stop, if there are more concentrations, otherwise return to Step 2.

Procedure 1

(1) **Input:** p_a, N and gen
(2) Initialize $\mathbf{E}^0 (k = 0)$
(3) **until** $(k = 2 \cdot \text{gen})$
(5) $\mathbf{E}^{k+1} \leftarrow \text{OperatorA}(\mathbf{E}^k)$ Section 2.1
(6) $\mathbf{M} \leftarrow \text{OperatorD}(\mathbf{E}^{k+1})$ Section 3.1
(7) $\mathbf{E}^{k+1} \leftarrow \text{OperatorE}(\mathbf{M}, \mathbf{E}^{k+1})$ Section 3.2
(8) $\mathbf{E}^{k+2} \leftarrow \text{OperatorB}(\mathbf{E}^{k+1})$ Section 2.2
(9) $\mathbf{M} \leftarrow \text{OperatorD}(\mathbf{E}^{k+2})$ Section 3.1
(10) $\mathbf{E}^{k+2} \leftarrow \text{OperatorE}(\mathbf{M}, \mathbf{E}^{k+2})$ Section 3.2
(11) **if** (s has changed)
(12) $\mathbf{M} \leftarrow \text{OperatorF}(\mathbf{M})$ Section 3.3
(13) **end if**
(14) **end until**

Algorithm 3: Multimodal cuckoo search (MCS) algorithm.

the depuration ratio R is calculated. This process is an iterative computation that begins with the shortest distance $\Delta^1_{1,2}$. The distance $\Delta^1_{1,2}$ (see Figure 4(c)), corresponding to $\mathbf{z}_1$ and $\mathbf{z}_2$, produces the evaluation of their medium point u $((\mathbf{z}_1+\mathbf{z}_2)/2)$. Since $f(u)$ is worse than $f(\mathbf{z}_1)$ but not worse than $f(\mathbf{z}_2)$, the element $\mathbf{z}_2$ is considered to be part of the same concentration as $\mathbf{z}_1$. The same conclusion is obtained for $\Delta^2_{1,3}$ in case of $\mathbf{z}_3$, after considering the point v. For $\Delta^3_{1,5}$, the point w is produced. Since $f(w)$ is worse than $f(\mathbf{z}_1)$ and $f(\mathbf{z}_5)$, the element $\mathbf{z}_5$ is considered to be part of the concentration corresponding to the next optimum. The iterative process ends here, after assuming that the same result is produced with $\Delta^4_{1,4}$ and $\Delta^5_{1,6}$, for $\mathbf{z}_4$ and $\mathbf{z}_6$, respectively. Therefore, the depuration ratio D_R is calculated as 85% of the distances $\Delta^3_{1,5}$. Once the elements inside of D_R have been removed from **Z**, the same process is applied to the new **Z**. As a result, the final configuration of the memory is shown in Figure 4(d).

3.4. Complete MCS Algorithm. Once the new operators (D) memory mechanism, (E) new selection strategy, and (F) depuration procedure have been defined, the proposed MCS algorithm can be summarized by Algorithm 3. The new algorithm combines operators defined in the original CS with the new ones. Despite these new operators, the MCS maintains the same three adjustable parameters (p_a, N, and gen) compared to the original CS method.

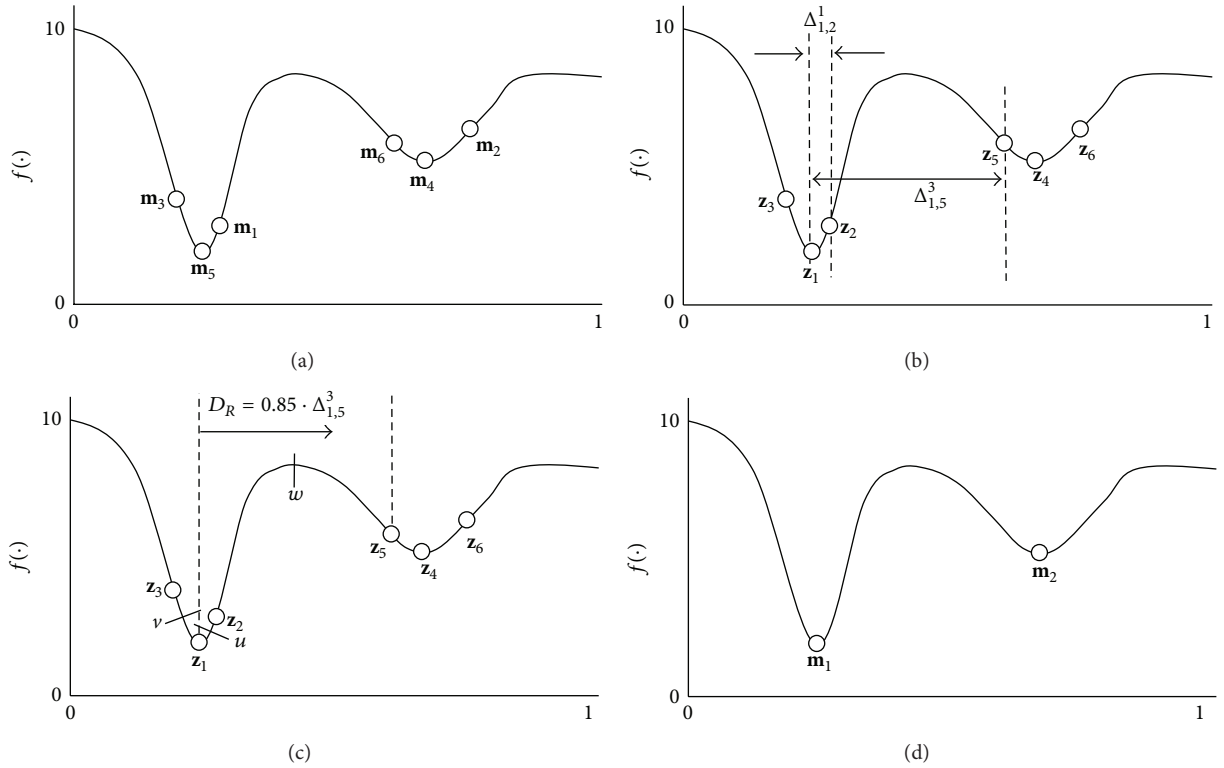


Figure 4: Depuration procedure. (a) Initial memory configuration, (b) vector $\mathbf{Z}$ and distances $\Delta^a_{1,j}$, (c) the determination of the depuration ratio R, and (d) the final memory configuration.

4. Experimental Results

This section presents the performance of the proposed algorithm beginning from Section 4.1 that describes the experimental methodology. For the sake of clarity, results are divided into two sections, Section 4.2 and Section 4.3, which report the comparison between the MCS experimental results and the outcomes produced by other multimodal metaheuristic algorithms.

4.1. Experimental Methodology. This section examines the performance of the proposed MCS by using a test suite of fourteen benchmark functions with different complexities. Table 3 in the appendix presents the benchmark functions used in our experimental study. In the table, **NO** indicates the number of optimal points in the function and S indicates the search space (subset of R^2). The experimental suite contains some representative, complicated, and multimodal functions with several local optima. Such functions are considered complex entities to be optimized, as they are particularly challenging to the applicability and efficiency of multimodal metaheuristic algorithms. A detailed description of each function is given in the appendix.

In the study, five performance indexes are compared: the effective peak number (EPN), the maximum peak ratio (MPR), the peak accuracy (PA), the distance accuracy (DA), and the number of function evaluations (NFE). The first four indexes assess the accuracy of the solution, whereas the last measures the computational cost.

The effective peak number (EPN) expresses the amount of detected peaks. An optimum $\mathbf{o}_j$ is considered as detected if the distance between the identified solution $\mathbf{z}_j$ and the optimum $\mathbf{o}_j$ is less than 0.01 ($\|\mathbf{o}_j - \mathbf{z}_j\| < 0.01$). The maximum peak ratio (MPR) is used to evaluate the quality and the number of identified optima. It is defined as follows:

$$\text{MPR} = \frac{\sum_{i=1}^{t} f(\mathbf{z}_i)}{\sum_{j=1}^{q} f(\mathbf{o}_j)}, \tag{15}$$

where t represents the number of identified solutions (identified optima) for the algorithm under testing and q represesnts the number of true optima contained in the function. The peak accuracy (PA) specifies the total error produced between the identified solutions and the true optima. Therefore, PA is calculated as follows:

$$PA = \sum_{j=1}^{q} \left| f(\mathbf{o}_j) - f(\mathbf{z}_j) \right|. \tag{16}$$

Peak accuracy (PA) may lead to erroneous results, mainly if the peaks are close to each other or hold an identical height. Under such circumstances, the distance accuracy (DA) is used to avoid such error. DA is computed as PA, but fitness values are replaced by the Euclidian distance. DA is thus defined by the following model:

$$\text{DA} = \sum_{j=1}^{q} \left\| \mathbf{o}_j - \mathbf{z}_j \right\|. \tag{17}$$

The number of function evaluations (NFE) indicates the total number of function computations that have been calculated by the algorithm under testing, through the overall optimization process.

The experiments compare the performance of MCS against the crowding differential evolution (CDE) [22], the fitness sharing differential evolution (SDE) [21, 22], the clearing procedure (CP) [26], the elitist-population strategy (AEGA) [28], the clonal selection algorithm (CSA) [30], and the artificial immune network (AiNet) [31].

Since the approach solves real-valued multimodal functions and a fair comparison must be assured, we have used for the GA approaches a consistent real coding variable representation and uniform operators for crossover and mutation. The crossover probability of $P_c = 0.8$ and the mutation probability of $P_m = 0.1$ have been used. We have employed the standard tournament selection operator with tournament size = 2 for implementing the sequential fitness sharing, the clearing procedure, and the elitist-population strategy (AEGA). On the other hand, the parameter values for the AiNet algorithm have been defined as suggested in [31], with the mutation strength of $\beta = 100$, the suppression threshold of $\sigma_{s(\text{aiNet})} = 0.2$, and the update rate of $d = 40\%$. Algorithms based on DE use a scaling factor of $F = 0.5$ and a crossover probability of $P_c = 0.9$. The crowding DE employs a crowding factor of CF = 50 and the sharing DE considers $\alpha = 1.0$ with a share radius of $\sigma_{\text{share}} = 0.1$.

In case of the MCS algorithm, the parameters are set to $p_a = 0.25$, the population size is $N = 50$, and the number of generations is gen = 500. Once they have been all experimentally determined, they are kept for all the test functions through all experiments.

To avoid relating the optimization results to the choice of a particular initial population and to conduct fair comparisons, we perform each test 50 times, starting from various randomly selected points in the search domain as it is commonly done in the literature.

All algorithms have been tested in MatLAB© over the same Dell Optiplex GX520 computer with a Pentium-4 2.66G-HZ processor, running Windows XP operating system over 1 Gb of memory. The sections below present experimental results for multimodal optimization problems which have been divided into two groups. The first one considers functions f_1–f_7, while the second gathers functions f_8–f_{14}.

4.2. Comparing MCS Performance for Functions f_1–f_7. This section presents a performance comparison for different algorithms solving the multimodal problems f_1–f_7 that are shown in Table 3. The aim is to determine whether MCS is more efficient and effective than other existing algorithms for finding all multiple optima of f_1–f_7. All the algorithms employ a population size of 50 individuals using 500 successive generations.

Table 1 provides a summarized performance comparison among several algorithms in terms of the effective peak number (EPN), the maximum peak ratio (MPR), the peak accuracy (PA), the distance accuracy (DA), and the number of function evaluations (NFE). The results are averaged by considering 50 different executions.

Considering the EPN index, in all functions f_1–f_7, MCS always finds better or equally optimal solutions. Analyzing results of function f_1, the CDE, AEGA, and the MCS algorithms reach all optima. In case of function f_2, only CSA and AiNet have not been able to detect all the optima values each time. Considering function f_3, only MCS can detect all optima at each run. In case of function f_4, most of the algorithms detect only half of the total optima but MCS can recognize most of them. Analyzing results of function f_5, CDE, CP, CSA, and AiNet present a similar performance whereas SDE, AEGA, and MCS obtain the best EPN values. In case of f_6, almost all algorithms present a similar performance; however, only the CDE, CP, and MCS algorithms have been able to detect all optima. Considering function f_7, the MCS algorithm is able to detect most of the optima whereas the rest of the methods reach different performance levels.

By analyzing the MPR index in Table 1, MCS has reached the best performance for all the multimodal problems. On the other hand, the rest of the algorithms present different accuracy levels, with CDE and SDE being the most consistent.

Considering thePA index, MCS presents the best performance. Since PA evaluates the accumulative differences of fitness values, it could drastically change when one or several peaks are not detected (function f_3) or when the function under testing presents peaks with high values (function f_4). For the case of the DA index in Table 1, it is evident that the MCS algorithm presents the best performance providing the shortest distances among the detected optima.

Analyzing the NFE measure in Table 1, it is clear that CSA and AiNet need fewer function evaluations than other algorithms considering the same termination criterion. This fact is explained by considering that both algorithms do not implement any additional process in order to detect multiple optima. On the other hand, the MCS method maintains a slightly higher number of function evaluations than CSA and AiNet due to the inclusion of the depuration procedure. The rest of the algorithms present a considerable higher NFE value.

It can be easily deduced from such results that the MCS algorithm is able to produce better search locations (i.e., a better compromise between exploration and exploitation) in a more efficient and effective way than other multimodal search strategies by using an acceptable number of function evaluations.

4.3. Comparing MCS Performance for Functions f_8–f_{14}. This section presents a performance comparison for different algorithms solving the multimodal problems f_8–f_{14} that are shown in Table 3. The aim is to determine whether MCS is more efficient and effective than its competitors for finding multiple optima in f_8–f_{14}. All the algorithms employ a population size of 50 individuals using 500 successive generations. Table 2 provides a summarized performance comparison among several algorithms in terms of the effective peak number (EPN), the maximum peak ratio (MPR), the peak accuracy (PA), the distance accuracy (DA), and the number

Table 1: Performance comparison among multimodal optimization algorithms for the test functions f_1–f_7. For all the parameters, numbers in parentheses are the standard deviations.

Function	Algorithm	EPN	MPR	PA	DA	NFE
	CDE	3 (0)	0.9996 (0.0004)	0.0995 (0.1343)	0.0305 (0.0169)	27432 (1432)
	SDE	2.96 (0.18)	0.9863 (0.0523)	1.3053 (0.8843)	0.1343 (0.0483)	31435 (2342)
	CP	2.93 (0.25)	0.9725 (0.0894)	1.3776 (1.0120)	0.1432 (0.0445)	34267 (4345)
f_1	AEGA	3 (0)	0.9932 (0.0054)	0.0991 (0.2133)	0.1031 (0.0065)	30323 (2316)
	CSA	2.91 (0.20)	0.9127 (0.0587)	1.4211 (1.0624)	0.2188 (0.0072)	25050 (0)
	AiNet	2.94 (0.20)	0.9002 (0.0901)	1.3839 (1.0214)	0.1760 (0.0067)	25050 (0)
	MCS	3 (0)	1 (0)	0.0005 (0.0001)	0.0007 (0.0002)	25433 (54)
	CDE	12 (0)	1 (0)	0.0015 (0.0010)	0.2993 (0.0804)	26321 (1934)
	SDE	12 (0)	1 (0)	0.0018 (0.0006)	0.3883 (0.0657)	32563 (1453)
	CP	12 (0)	1 (0)	0.0009 (0.0003)	0.2694 (0.0506)	30324 (3521)
f_2	AEGA	12 (0)	0.9987 (0.0011)	0.0988 (0.0097)	0.3225 (0.0058)	29954 (1987)
	CSA	11.92 (0.41)	0.9011 (0.0091)	0.1055 (0.0121)	0.4257 (0.0096)	25050 (0)
	AiNet	11.96 (0.30)	0.9256 (0.0074)	0.0996 (0.0105)	0.3239 (0.0081)	25050 (0)
	MCS	12 (0)	1 (0)	0.0001 (0.0001)	0.0073 (0.0002)	25188 (42)
	CDE	23.03 (1.77)	0.8780 (0.0956)	180.47 (265.54)	9.3611 (6.4667)	28654 (2050)
	SDE	20.06 (2.59)	0.6980 (0.1552)	155.52 (184.59)	14.892 (7.5935)	31432 (1017)
	CP	21.03 (1.90)	0.7586 (0.1125)	192.32 (146.21)	11.195 (3.1490)	32843 (2070)
f_3	AEGA	20.45 (1.21)	0.7128 (0.1493)	134.47 (157.37)	16.176 (8.0751)	30965 (2154)
	CSA	18.02 (2.41)	0.5875 (0.1641)	185.64 (104.24)	21.057 (10.105)	25050 (0)
	AiNet	19.24 (2.01)	0.6123 (0.1247)	179.35 (164.37)	18.180 (9.1112)	25050 (0)
	MCS	24.66 (1.01)	0.9634 (0.0397)	2.9408 (4.3888)	15.521 (8.0834)	25211 (37)
	CDE	3.46 (1.00)	0.4929 (0.1419)	395.46 (305.01)	210.940 (72.99)	29473 (3021)
	SDE	3.73 (0.86)	0.5301 (0.1268)	544.48 (124.11)	206.65 (160.84)	33421 (1342)
	CP	3.26 (0.63)	0.4622 (0.0869)	192.32 (146.21)	199.41 (68.434)	29342 (1543)
f_4	AEGA	3.51 (0.52)	0.5031 (0.0754)	188.23 (101.54)	187.21 (33.211)	32756 (1759)
	CSA	3.12 (0.11)	0.4187 (0.0464)	257.54 (157.18)	278.14 (47.120)	25050 (0)
	AiNet	3.20 (0.47)	0.5164 (0.0357)	197.24 (86.21)	178.23 (29.191)	25050 (0)
	MCS	6.26 (0.82)	0.8919 (0.1214)	41.864 (16.63)	39.938 (12.962)	25361 (81)
	CDE	22.96 (2.25)	0.4953 (0.0496)	0.2348 (0.0269)	17.83 (7.1214)	28543 (1345)
	SDE	31.40 (2.35)	0.6775 (0.0503)	0.7005 (0.0849)	3.9430 (0.9270)	30543 (1576)
	CP	21.33 (2.00)	0.4599 (0.0436)	1.3189 (0.5179)	10.766 (1.9245)	28743 (2001)
f_5	AEGA	30.11 (2.01)	0.6557 (0.0127)	0.8674 (0.0296)	2.870 (1.6781)	29765 (1911)
	CSA	24.79 (3.14)	0.5107 (0.0308)	0.2121 (0.0187)	8.7451 (3.470)	25050 (0)
	AiNet	26.57 (2.35)	0.5005 (0.0471)	0.2087 (0.0324)	6.472 (2.4187)	25050 (0)
	MCS	33.03 (2.07)	0.8535 (0.0251)	0.1617 (0.0283)	4.6012 (1.4206)	25159 (49)
	CDE	6 (0)	0.9786 (0.0157)	0.1280 (0.0942)	0.1231 (0.0182)	30234 (2410)
	SDE	5.86 (0.43)	0.9185 (0.0685)	0.3842 (0.1049)	0.1701 (0.0222)	31453 (1154)
	CP	6 (0)	0.9423 (0.0123)	0.3460 (0.0741)	0.1633 (0.0149)	30231 (832)
f_6	AEGA	5.11 (0.64)	0.8945 (0.0387)	0.4004 (0.0879)	0.1224 (0.0101)	31932 (943)
	CSA	4.97 (0.24)	0.8174 (0.0631)	0.4797 (0.0257)	0.1295 (0.0054)	25050 (0)
	AiNet	5.23 (1)	0.9012 (0.0197)	0.3974 (0.0702)	0.1197 (0.0054)	25050 (0)
	MCS	6 (0)	0.9993 (0.0002)	0.0037 (0.0014)	0.0006 (0.0002)	25463 (37)
	CDE	30.36 (2.77)	0.6200 (0.0566)	2.2053 (1.8321)	330.51 (47.531)	33423 (1021)
	SDE	35.06 (5.15)	0.7162 (0.1051)	1.9537 (0.9290)	243.39 (140.04)	32832 (995)
	CP	35.06 (3.98)	0.7164 (0.0812)	2.4810 (1.4355)	250.11 (78.194)	31923 (834)
f_7	AEGA	32.51 (2.59)	0.7004 (0.0692)	2.0751 (0.9561)	278.78 (46.225)	33821 (1032)
	CSA	31.78 (1.14)	0.6764 (0.4100)	1.9408 (0.9471)	347.21 (38.147)	25050 (0)
	AiNet	34.42 (1.80)	0.7237 (0.0257)	1.8632 (0.0754)	261.27 (61.217)	25050 (0)
	MCS	38.86 (1.54)	0.8014 (0.0313)	0.2290 (0.0166)	49.53 (7.1533)	25643 (97)

Table 2: Performance comparison among multimodal optimization algorithms for the test functions f_8–f_{14}. For all the parameters, numbers in parentheses are the standard deviations.

Function	Algorithm	EPN	MPR	PA	DA	NFE
	CDE	24.16 (2.77)	0.9682 (0.0318)	2.4873 (2.4891)	0.8291 (0.8296)	28453 (2345)
	SDE	18.56 (2.51)	0.4655 (0.0636)	30.21 (43.132)	2.1162 (0.6697)	31328 (945)
	CP	8.80 (1.95)	0.2222 (0.0509)	60.52 (56.056)	6.9411 (0.9500)	30743 (1032)
f_8	AEGA	15.67 (2.21)	0.3934 (0.0534)	40.56 (10.111)	3.2132 (0.2313)	32045 (684)
	CSA	14.54 (3.12)	0.3323 (0.0431)	48.34 (8.343)	3.8232 (0.4521)	25050 (0)
	AiNet	16.78 (2.63)	0.4264 (0.0321)	37.32 (10.432)	2.9832 (0.5493)	25050 (0)
	MCS	24.73 (0.49)	0.9898 (0.0170)	0.900 (1.4771)	0.2584 (0.1275)	25582 (74)
	CDE	2.1 (0.20)	0.7833 (0.0211)	23.235 (7.348)	2.9354 (0.3521)	30074 (1621)
	SDE	2.3 (0.31)	0.8245 (0.0145)	20.745 (8.012)	2.6731 (0.8621)	31497 (245)
	CP	2.4 (0.25)	0.8753 (0.0301)	18.563 (5.865)	2.3031 (0.7732)	29746 (1114)
f_9	AEGA	2.1 (0.10)	0.7879 (0.0174)	22.349 (6.231)	3.0021 (0.6431)	30986 (1027)
	CSA	2 (0)	0.7098 (0.0025)	32.859 (8.659)	3.1432 (0.5431)	25050 (0)
	AiNet	2 (0)	0.7165 (0.0076)	31.655 (6.087)	3.2265 (0.3467)	25050 (0)
	MCS	4.74 (0.25)	0.9154 (0.0163)	2.3515 (2.511)	0.0109 (0.0428)	26043 (112)
	CDE	4.12 (0.78)	0.7285 (0.0342)	3.546 (1.034)	3.0132 (0.5321)	29597 (1034)
	SDE	4.64 (0.54)	0.7893 (0.0532)	3.054 (1.127)	2.864 (0.3271)	32743 (964)
	CP	4 (0)	0.7092 (0.0298)	3.881 (1.154)	3.3412 (0.4829)	28463 (1142)
f_{10}	AEGA	3.43 (0.33)	0.6734 (0.0745)	4.212 (1.312)	3.9121 (0.8732)	29172 (1044)
	CSA	3.76 (0.51)	0.6975 (0.0828)	4.002 (1.197)	3.5821 (0.7498)	25050 (0)
	AiNet	4 (0)	0.7085 (0.0385)	3.797 (1.002)	3.3002 (0.6496)	25050 (0)
	MCS	6.82 (0.75)	0.9274 (0.0137)	0.423 (0.064)	0.6842 (0.0598)	25873 (88)
	CDE	10.36 (1.60)	0.8572 (0.1344)	1.859 (0.952)	0.5237 (0.0321)	34156 (2321)
	SDE	10.36 (2.04)	0.8573 (0.1702)	1.268 (0.581)	0.6927 (0.0921)	32132 (975)
	CP	9.16 (1.76)	0.7577 (0.1462)	2.536 (0.890)	0.6550 (0.0440)	30863 (1002)
f_{11}	AEGA	8.34 (1.32)	0.6954 (0.1021)	4.432 (1.232)	0.7021 (0.0231)	31534 (852)
	CSA	8 (0)	0.6532 (0.1378)	4.892 (1.003)	0.7832 (0.0432)	25050 (0)
	AiNet	8 (0)	0.6438 (0.2172)	4.921 (1.102)	0.7753 (0.0326)	25050 (0)
	MCS	12 (0)	0.9998 (0.0003)	0.011 (0.008)	0.0060 (0.0012)	25789 (121)
	CDE	6.21 (1.54)	0.6986 (0.1893)	4.029 (1.401)	5.1514 (1.0351)	31456 (975)
	SDE	5.34 (2.03)	0.5812 (0.1992)	5.075 (1.071)	6.0117 (1.1517)	32481 (1002)
	CP	6.04 (0.61)	0.6312 (0.1771)	4.657 (1.321)	5.3177 (1.7517)	33123 (563)
f_{12}	AEGA	4 (0)	0.4112 (0.0343)	6.341 (1.034)	7.8751 (1.652)	32634 (843)
	CSA	4 (0)	0.3998 (0.0212)	6.151 (1.121)	7.7976 (1.0043)	25050 (0)
	AiNet	4 (0)	0.4034 (0.0973)	6.003 (1.735)	7.6613 (1.1219)	25050 (0)
	MCS	9.65 (1.45)	0.9411 (0.0087)	0.015 (0.009)	0.1043 (0.0864)	25832 (65)
	CDE	13 (0)	1 (0)	0.010 (0.003)	0.031 (0.0098)	31572 (962)
	SDE	13 (0)	1 (0)	0.008 (0.004)	0.021 (0.0065)	33435 (1201)
	CP	13 (0)	1 (0)	0.015 (0.002)	0.037 (0.0065)	31834 (799)
f_{13}	AEGA	10.66 (1.21)	0.8323 (0.0343)	0.088 (0.033)	0.096 (0.0098)	32845 (1182)
	CSA	8.94 (2.34)	0.7998 (0.0564)	0.110 (0.088)	0.113 (0.0104)	25050 (0)
	AiNet	10.32 (1.52)	0.8297 (0.0206)	0.098 (0.075)	0.087 (0.0086)	25050 (0)
	MCS	13 (0)	0.9997 (0.0134)	0.011 (0.007)	0.023 (0.0016)	25740 (101)
	CDE	3.04 (1.34)	0.6675 (0.0754)	0.809 (0.101)	176.54 (21.23)	32273 (1004)
	SDE	3.55 (0.56)	0.7017 (0.0487)	0.675 (0.079)	115.43 (34.21)	30372 (965)
	CP	2.87 (1.23)	0.6123 (0.0861)	1.081 (0.201)	202.65 (42.81)	31534 (1298)
f_{14}	AEGA	3 (0)	0.6686 (0.0542)	0.894 (0.076)	150.32 (57.31)	29985 (1745)
	CSA	3 (0)	0.6691 (0.0231)	0.897 (0.045)	161.57 (27.92)	25050 (0)
	AiNet	3.50 (0.25)	0.7001 (0.0765)	0.668 (0.097)	121.43 (43.12)	25050 (0)
	MCS	7.13 (0.81)	0.9859 (0.0094)	0.023 (0.010)	17.62 (4.13)	25786 (92)

Table 3: Test functions used in the experimental study.

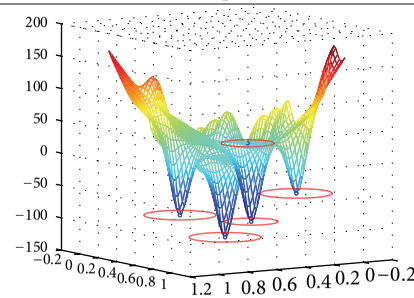
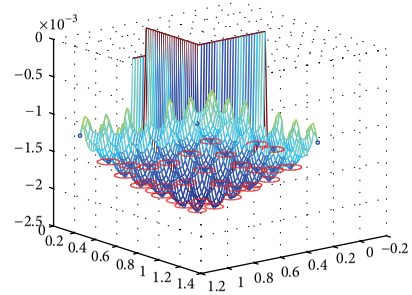
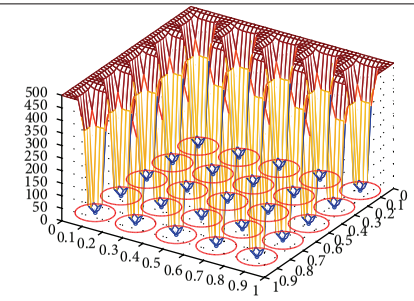
$f(x)$ $(x = \{x_1, x_2\})$	S	NO	Graph
Bird $f_1(\mathbf{x}) = \sin(x_1) \cdot e^{(1-\cos(x_2))^2} + \cos(x_2) \cdot e^{(1-\sin(x_1))^2} + (x_1 - x_2)^2$	$[-2\pi, 2\pi]$	3	
Cross in tray $f_2(\mathbf{x}) = -0.0001 \cdot \left(\left\lvert \sin(x_1) \cdot \sin(x_2) \cdot e^{\lvert 100 - (\sqrt{x_1^2 + x_2^2}/\pi) \rvert} \right\rvert + 1 \right)^{0.1}$	$[-10, 10]$	12	
DeJongs5 $f_3(\mathbf{x}) = \left\{ 0.002 + \sum_{i=-2}^{2} \sum_{j=-2}^{2} \left[5(i+1) + j + 3 + (x_1 - 16j)^6 + (x_2 - 16i)^6 \right]^{-1} \right\}^{-1}$	$[-40, 40]$	25	

Table 3: Continued.

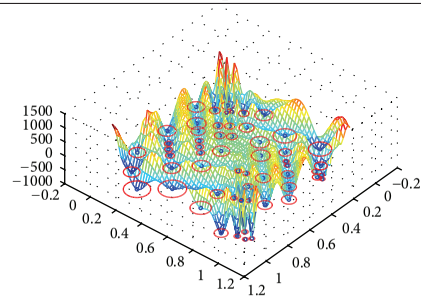
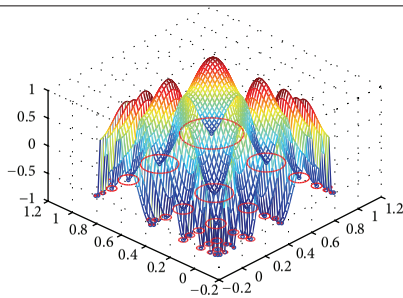
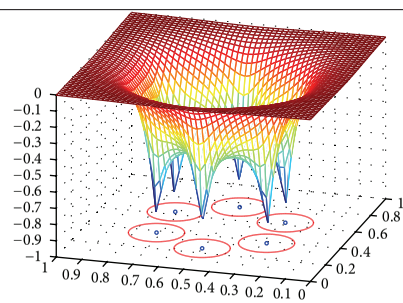
$f(x)$ $(x = \{x_1, x_2\})$	S	NO	Graph
Eggholder $f_4(\mathbf{x}) = -(x_2 + 47)\sin\left(\sqrt{\left\lvert x_2 + \frac{x_1}{2} + 47\right\rvert}\right) - x_1 \sin\left(\sqrt{\left\lvert x_1 + \frac{x_2}{2} + 47\right\rvert}\right)$	$[-512, 512]$	7	
Vincent $f_5(\mathbf{x}) = -\sum_{i=1}^{n} \sin(10 \cdot \log(x_i))$	$[0.25, 10]$	36	
Roots $f_6(\mathbf{x}) = -\left(1 + \left\lvert (x_1 + x_2 i)^6 - 1\right\rvert\right)^{-1}$	$[-2, 2]$	6	

Table 3: Continued.

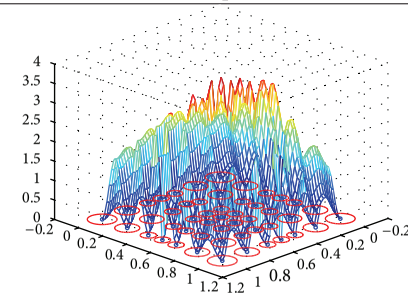
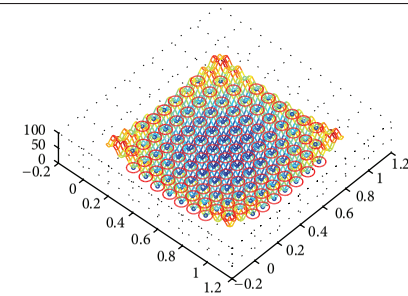
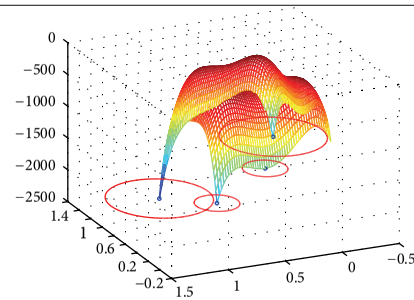
$f(x)$ $(x = \{x_1, x_2\})$	S	NO	Graph
Hilly $f_7(\mathbf{x}) = 10\left[e^{-(\lvert x_1\rvert/50)}\left(1-\cos\left(\frac{6}{100^{3/4}}\pi\lvert x_1\rvert^{3/4}\right)\right)+e^{-(\lvert x_2\rvert/250)}\left(1-\cos\left(\frac{6}{100^{3/4}}\pi\lvert x_2\rvert^{3/4}\right)\right)+2\left(e^{-((b-x_1)^2+(b-x_2)^2)/50}\right)\right]$ with $b = \left((5/6)\cdot 100^{3/4}\right)^{4/3}$	$[-100, 100]$	48	
Rastrigin $f_8(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i)$	$[-5.12, 5.12]$	25	
Himmemlblau $f_9(\mathbf{x}) = -\left(x_1^2 + x_2 - 11\right)^2 - \left(x_1 + x_2^2 - 7\right)^2$	$[-6, 6]$	5	

Table 3: Continued.

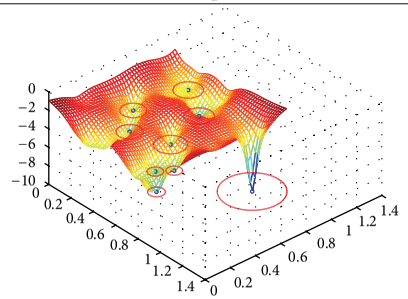
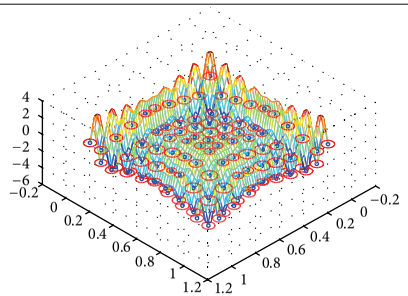
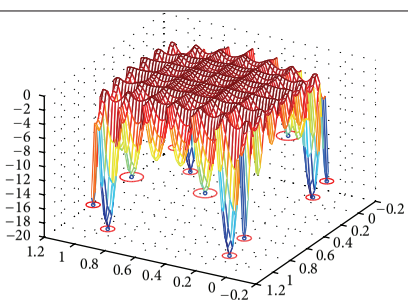
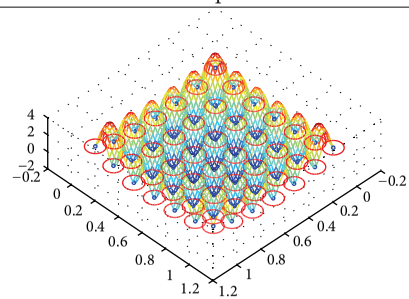
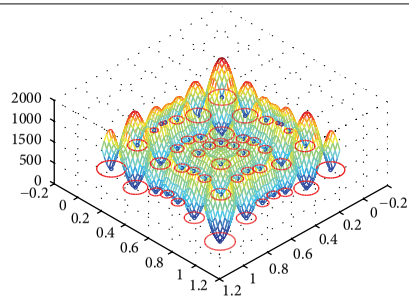
$f(x)$ $(x = \{x_1, x_2\})$	S	NO	Graph
Foxholes $f_{10}(\mathbf{x}) = -\sum_{i=1}^{30}\left(\sum_{j=1}^{n}\left[\left(x_j - a_{ij}\right)^2 + c_j\right]\right)^{-1}$	$[0, 10]$	8	
Guichi_f4 $f_{11}(\mathbf{x}) = -\left(x_1 \sin\left(4\pi x_1\right) - x_2 \sin\left(4\pi x_2 + \pi\right)\right)$	$[-2, 2]$	12	
Holder Table $f_{12}(\mathbf{x}) = -\sin\left(x_1\right)\cos\left(x_2\right) e^{\left\lvert 1-\left(\sqrt{{x_1}^2+{x_2}^2}/\pi\right)\right\rvert}$	$[-10, 10]$	12	

Table 3: Continued.

$f(x)$ $(x = \{x_1, x_2\})$	S	NO	Graph
Rastriguin_49m $f_{13}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 - 18\cos(2\pi x_i)$	$[-1, 1]$	13	
Schwefel $f_{14}(\mathbf{x}) = 418.9829 \cdot n + \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]$	8	

of function evaluations (NFE). The results are averaged by considering 50 different executions.

The goal of multimodal optimizers is to find as many as possible global optima and good local optima. The main objective in these experiments is to determine whether MCS is able to find not only optima with prominent fitness value, but also optima with low fitness values. Table 2 provides a summary of the performance comparison among the different algorithms.

Considering the EPN measure, it is observed that MCS finds more optimal solutions for the multimodal problems f_8–f_{14} than the other methods. Analyzing function f_8, only MCS can detect all optima whereas CP, AEGA, CSA, and AiNet exhibit the worst EPN performance.

Functions f_9–f_{12} represent a set of special cases which contain a few prominent optima (with good fitness value). However, such functions present also several optima with bad fitness values. In these functions, MCS is able to detect the highest number of optimum points. On the contrary, the rest of algorithms can find only prominent optima.

For function f_{13}, four algorithms (CDE, SDE, CP, and MCS) can recognize all optima for each execution. In case of function f_{14}, numerous optima are featured with different fitness values. However, MCS still can detect most of the optima.

In terms of number of the maximum peak ratios (MPR), MCS has obtained the best score for all the multimodal problems. On the other hand, the rest of the algorithms present different accuracy levels.

A close inspection of Table 2 also reveals that the proposed MCS approach is able to achieve the smallest PA and DA values in comparison to all other methods.

Similar conclusions to those in Section 4.2 can be established regarding the number of function evaluations (NFE). All results demonstrate that MCS achieves the overall best balance in comparison to other algorithms, in terms of both the detection accuracy and the number of function evaluations.

5. Conclusions

The cuckoo search (CS) algorithm has been recently presented as a new heuristic algorithm with good results in real-valued optimization problems. In CS, individuals emulate eggs (contained in nests) which interact in a biological system by using evolutionary operations based on the breeding behavior of some cuckoo species. One of the most powerful features of CS is the use of Lévy flights to generate new candidate solutions. Under this approach, candidate solutions are modified by employing many small changes and occasionally large jumps. As a result, CS can substantially improve the relationship between exploration and exploitation, still enhancing its search capabilities. Despite such characteristics, the CS method still fails to provide multiple solutions in a single execution. In order to overcome such inconvenience, this paper proposes a new multimodal optimization algorithm called the multimodal cuckoo search (MCS). Under MCS, the original CS is enhanced with multimodal capacities by means of (1) incorporation of a memory mechanism to efficiently register potential local optima according to their fitness value and the distance to other potential solutions, (2) modification of the original CS individual selection strategy to accelerate the detection process of new local minima, and (3) inclusion of a depuration procedure to cyclically eliminate duplicated memory elements.

MCS has been experimentally evaluated over a test suite of the fourteen benchmark multimodal functions. The performance of MCS has been compared to some other existing algorithms including the crowding differential evolution (CDE) [22], the fitness sharing differential evolution (SDE) [21, 22], the clearing procedure (CP) [26], the elitist-population strategy (AEGA) [28], the clonal selection algorithm (CSA) [30], and the artificial immune network (AiNet) [31]. All experiments have demonstrated that MCS generally outperforms all other multimodal metaheuristic algorithms in terms of both the detection accuracy and the number of function evaluations. The remarkable performance of MCS is explained by two different features: (i) operators (such as Lévy flight) allow a better exploration of the search space, increasing the capacity to find multiple optima, and (ii) the diversity of solutions contained in the memory **M** in the context of multimodal optimization is maintained and further improved through an efficient mechanism.

Appendix

For list of benchmark functions, see Table 3.

Conflict of Interests

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

Acknowledgment

The proposed algorithm is part of the optimization system used by a biped robot supported under the Grant CONACYT CB 181053.

References

[1] P. M. Pardalos, H. E. Romeijn, and H. Tuy, "Recent developments and trends in global optimization," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 209–228, 2000.

[2] C. A. Floudas, I. G. Akrotirianakis, S. Caratzoulas, C. A. Meyer, and J. Kallrath, "Global optimization in the 21st century: advances and challenges," *Computers & Chemical Engineering*, vol. 29, no. 6, pp. 1185–1202, 2005.

[3] Y. Ji, K.-C. Zhang, and S.-J. Qu, "A deterministic global optimization algorithm," *Applied Mathematics and Computation*, vol. 185, no. 1, pp. 382–387, 2007.

[4] A. Georgieva and I. Jordanov, "Global optimization based on novel heuristics, low-discrepancy sequences and genetic algorithms," *European Journal of Operational Research*, vol. 196, no. 2, pp. 413–422, 2009.

[5] D. Lera and Y. D. Sergeyev, "Lipschitz and Hölder global optimization using space-filling curves," *Applied Numerical Mathematics*, vol. 60, no. 1-2, pp. 115–129, 2010.

[6] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, John Wiley & Sons, Chichester, UK, 1966.

[7] K. de Jong, *Analysis of the behavior of a class of genetic adaptive systems [Ph.D. thesis]*, University of Michigan, Ann Arbor, Mich, USA, 1975.

[8] J. R. Koza, "Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems," Tech. Rep. STAN-CS-90-1314, Stanford University, Stanford, Calif, USA, 1990.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

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

[11] L. N. de Castro and F. J. von Zuben, "Artificial immune systems—part I: basic theory and applications," Tech. Rep. TR-DCA 01/99, 1999.

[12] R. Storn and K. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimisation over continuous spaces," Tech. Rep. TR-95-012, ICSI, Berkeley, Calif, USA, 1995.

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

[14] S. I. Birbil and S.-C. Fang, "An electromagnetism-like mechanism for global optimization," *Journal of Global Optimization*, vol. 25, no. 3, pp. 263–282, 2003.

[15] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Filter modeling using gravitational search algorithm," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 117–122, 2011.

[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] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Tech. Rep. 91-016, Politecnico di Milano, Milano, Italy, 1991.

[18] S. Dasa, S. Maity, B.-Y. Qu, and P. N. Suganthan, "Real-parameter evolutionary multimodal optimization—a survey of the state-of-the-art," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 71–78, 2011.

[19] K.-C. Wong, C.-H. Wu, R. K. P. Mok, C. Peng, and Z. Zhang, "Evolutionary multimodal optimization using the principle of locality," *Information Sciences*, vol. 194, pp. 138–170, 2012.

[20] D. Beasley, D. R. Bull, and R. R. Matin, "A sequential niche technique for multimodal function optimization," *Evolutionary Computation*, vol. 1, no. 2, pp. 101–125, 1993.

[21] B. L. Miller and M. J. Shaw, "Genetic algorithms with dynamic niche sharing for multimodal function optimization," in *Proceedings of the 3rd IEEE International Conference on Evolutionary Computation*, pp. 786–791, May 1996.

[22] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, pp. 1382–1389, June 2004.

[23] S. W. Mahfoud, *Niching methods for genetic algorithms [Ph.D. thesis]*, Illinois Genetic Algorithm Laboratory, University of Illinois, Urbana, Ill, USA, 1995.

[24] O. J. Mengshoel and D. E. Goldberg, "Probability crowding: deterministic crowding with probabilistic replacement," in *Proceedings of the International Genetic and Evolutionary Computation Conference*, W. Banzhaf, Ed., pp. 409–416, Orlando, Fla, USA, 1999.

[25] X. Yin and N. Germay, "A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization," in *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, pp. 450–457, 1993.

[26] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pp. 798–803, Nagoya, Japan, May 1996.

[27] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 207–234, 2002.

[28] Y. Liang and K.-S. Leung, "Genetic Algorithm with adaptive elitist-population strategies for multimodal function optimization," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 2017–2034, 2011.

[29] G. Chen, C. P. Low, and Z. Yang, "Preserving and exploiting genetic diversity in evolutionary programming algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 661–673, 2009.

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

[31] L. N. Castro and J. Timmis, "An artificial immune network for multimodal function optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 699–704, Honolulu, Hawaii, USA, 2002.

[32] Q. Xu, L. Wang, and J. Si, "Predication based immune network for multimodal function optimization," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 495–504, 2010.

[33] K. C. Tan, S. C. Chiam, A. A. Mamun, and C. K. Goh, "Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization," *European Journal of Operational Research*, vol. 197, no. 2, pp. 701–713, 2009.

[34] S. Roy, S. M. Islam, S. Das, S. Ghosh, and A. V. Vasilakos, "A simulated weed colony system with subregional differential evolution for multimodal optimization," *Engineering Optimization*, vol. 45, no. 4, pp. 459–481, 2013.

[35] F. Yahyaie and S. Filizadeh, "A surrogate-model based multimodal optimization algorithm," *Engineering Optimization*, vol. 43, no. 7, pp. 779–799, 2011.

[36] S. Yazdani, H. Nezamabadi-pour, and S. Kamyab, "A gravitational search algorithm for multimodal optimization," *Swarm and Evolutionary Computation*, vol. 14, pp. 1–14, 2014.

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

[38] S. Walton, O. Hassan, K. Morgan, and M. R. Brown, "A review of the development and applications of the Cuckoo search 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. 257–271, Elsevier, San Diego, Calif, USA, 2013.

[39] 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.

[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. Kumar and S. Chakarverty, "Design optimization for reliable embedded system using Cuckoo search," in *Proceedings of the 3rd International Conference on Electronics Computer Technology (ICECT '11)*, pp. 264–268, April 2011.

[42] A. Kaveh and T. Bakhshpoori, "Optimum design of steel frames using Cuckoo search algorithm with Lévy flights," *The Structural Design of Tall and Special Buildings*, vol. 22, no. 13, pp. 1023–1036, 2013.

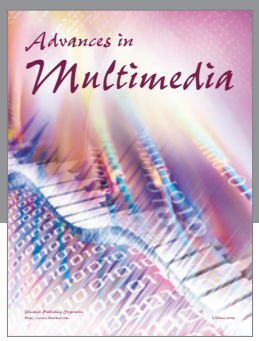
[43] L. H. Tein and R. Ramli, "Recent advancements of nurse scheduling models and a potential path," in *Proceedings of 6th IMT-GT Conference on Mathematics, Statistics and Its Applications (ICMSA '10)*, pp. 395–409, 2010.

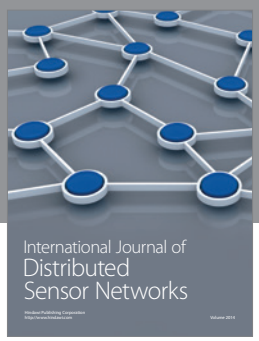
[44] V. Bhargava, S. E. K. Fateen, and A. Bonilla-Petriciolet, "Cuckoo search: a new nature-inspired optimization method for phase equilibrium calculations," *Fluid Phase Equilibria*, vol. 337, pp. 191–200, 2013.

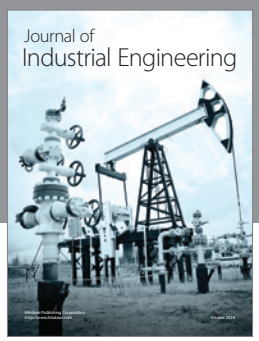
[45] Z. Moravej and A. Akhlaghi, "A novel approach based on cuckoo search for DG allocation in distribution network," *International Journal of Electrical Power and Energy Systems*, vol. 44, no. 1, pp. 672–679, 2013.

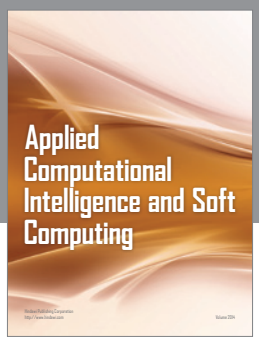
[46] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *Journal of Computational Physics*, vol. 226, no. 2, pp. 1830–1844, 2007.

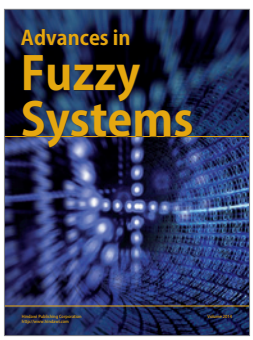
[47] R. N. Mantegna, "Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes," *Physical Review E*, vol. 49, no. 4, pp. 4677–4683, 1994.











Hindawi

Submit your manuscripts at
http://www.hindawi.com