Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 948303, 22 pages
http://dx.doi.org/10.1155/2013/948303

Research Article

A Swarm Optimization Algorithm for Multimodal Functions and Its Application in Multicircle Detection

Erik Cuevas, Daniel Zaldívar, and Marco Pérez-Cisneros

Departamento de Ciencias Computacionales, Universidad de Guadalajara, CUCEI, Avenida Revolución 1500, C.P 44430, Guadalajara, Jal, Mexico

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

Received 3 September 2012; Accepted 25 December 2012

Academic Editor: Baozhen Yao

In engineering problems due to physical and cost constraints, the best results, obtained by a global optimization algorithm, cannot be realized always. Under such conditions, if multiple solutions (local and global) are known, the implementation can be quickly switched to another solution without much interrupting the design process. This paper presents a new swarm multimodal optimization algorithm named as the collective animal behavior (CAB). Animal groups, such as schools of fish, flocks of birds, swarms of locusts, and herds of wildebeest, exhibit a variety of behaviors including swarming about a food source, milling around a central location, or migrating over large distances in aligned groups. These collective behaviors are often advantageous to groups, allowing them to increase their harvesting efficiency to follow better migration routes, to improve their aerodynamic, and to avoid predation. In the proposed algorithm, searcher agents emulate a group of animals which interact with each other based on simple biological laws that are modeled as evolutionary operators. Numerical experiments are conducted to compare the proposed method with the state-of-the-art methods on benchmark functions. The proposed algorithm has been also applied to the engineering problem of multi-circle detection, achieving satisfactory results.

1. Introduction

A large number of real-world problems can be considered as multimodal function optimization subjects. An objective function may have several global optima, that is, several points holding objective function values which are equal to the global optimum. Moreover, it may exhibit some other local optima points whose objective function values lay nearby a global optimum. Since the mathematical formulation of a real-world problem often produces a multimodal optimization issue, finding all global or even these local optima would provide to the decision makers multiple options to choose from [1].

Several methods have recently been proposed for solving the multimodal optimization problem. They can be divided into two main categories: deterministic and stochastic (metaheuristic) methods. When facing complex multimodal optimization problems, deterministic methods, such as gradient descent method, the quasi-Newton method, and the Nelder-Mead's simplex method, may get easily trapped into the local optimum as a result of deficiently exploiting local information. They strongly depend on a priori information about the objective function, yielding few reliable results.

Metaheuristic algorithms have been developed combining rules and randomness mimicking several phenomena. These phenomena include evolutionary processes (e.g., the evolutionary algorithm proposed by Fogel et al. [2], de Jong [3], and Koza [4] and the genetic algorithms (GAs) proposed by Holland [5] and Goldberg [6]), immunological systems (e.g., the artificial immune systems proposed by de Castro et al. [7]), physical processes (e.g., simulated annealing proposed by Kirkpatrick et al. [8], electromagnetism-like proposed by Birbil et al. [9], and the gravitational search algorithm proposed by Rashedi et al. [10]), and the musical process of searching for a perfect state of harmony (proposed by Geem et al. [11], Lee and Geem [12], Geem [13], and Gao et al. [14]).

Traditional GAs perform well for locating a single optimum but fail to provide multiple solutions. Several methods have been introduced into the GA's scheme to achieve

multimodal function optimization, such as sequential fitness sharing [15, 16], deterministic crowding [17], probabilistic crowding [18], clustering-based niching [19], clearing procedure [20], species conserving genetic algorithm [21], and elitist-population strategies [22]. However, algorithms based on the GAs do not guarantee convergence to global optima because of their poor exploitation capability. GAs exhibit other drawbacks such as the premature convergence which results from the loss of diversity in the population and becomes a common problem when the search continues for several generations. Such drawbacks [23] prevent the GAs from practical interest for several applications.

Using a different metaphor, other researchers have employed artificial immune systems (AIS) to solve the multimodal optimization problems. Some examples are the clonal selection algorithm [24] and the artificial immune network (AiNet) [25, 26]. Both approaches use some operators and structures which attempt to algorithmically mimic the natural immune system's behavior of human beings and animals.

Several studies have been inspired by animal behavior phenomena in order to develop optimization techniques such as the particle swarm optimization (PSO) algorithm which models the social behavior of bird flocking or fish schooling [27]. In recent years, there have been several attempts to apply the PSO to multimodal function optimization problems [28, 29]. However, the performance of such approaches presents several flaws when it is compared to the other multi-modal metaheuristic counterparts [26].

Recently, the concept of individual organization [30, 31] has been widely used to understand collective behavior of animals. The central principle of individual organization is that simple repeated interactions between individuals can produce complex behavioral patterns at group level [30, 32, 33]. Such inspiration comes from behavioral patterns seen in several animal groups, such as ant pheromone trail networks, aggregation of cockroaches, and the migration of fish schools, which can be accurately described in terms of individuals following simple sets of rules [34]. Some examples of these rules [33, 35] include keeping current position (or location) for best individuals, local attraction or repulsion, random movements, and competition for the space inside of a determined distance. On the other hand, new studies have also shown the existence of collective memory in animal groups [36–38]. The presence of such memory establishes that the previous history, of group structure, influences the collective behavior exhibited in future stages. Therefore, according to these new developments, it is possible to model complex collective behaviors by using simple individual rules and configuring a general memory.

On the other hand, the problem of detecting circular features holds paramount importance in several engineering applications. The circle detection in digital images has been commonly solved through the circular Hough transform (CHT) [39]. Unfortunately, this approach requires a large storage space that augments the computational complexity and yields a low processing speed. In order to overcome this problem, several approaches which modify the original CHT have been proposed. One well-known example is the randomized Hough transform (RHT) [40]. As an alternative to Hough-transform-based techniques, the problem of shape recognition has also been handled through optimization methods. In general, they have demonstrated to deliver better results than those based on the HT considering accuracy, speed, and robustness [41]. Such approaches have produced several robust circle detectors using different optimization algorithms such as genetic algorithms (GAs) [41], harmony search (HSA) [42], electromagnetism-like (EMO) [43], differential evolution (DE) [44], and bacterial foraging optimization (BFOA) [45]. Since such evolutionary algorithms are global optimizers, they detect only the global optimum (only one circle) of an objective function that is defined over a given search space. However, extracting multiple-circle primitives falls into the category of multi-modal optimization, where each circle represents an optimum which must be detected within a feasible solution space. The quality for such optima is characterized by the properties of their geometric primitives. Big and well-drawn circles normally represent points in the search space with high fitness values (possible global maximum) whereas small and dashed circles describe points with fitness values which account for local maxima. Likewise, circles holding similar geometric properties, such as radius and size, tend to represent locations with similar fitness values. Therefore, a multi-modal method must be applied in order to appropriately solve the problem of multi-shape detection. In this paper, a new multimodal optimization algorithm based on the collective animal behavior is proposed and also applied to multicircle detection.

This paper proposes a new optimization algorithm inspired by the collective animal behavior. In this algorithm, the searcher agents emulate a group of animals that interact with each other based on simple behavioral rules which are modeled as evolutionary operators. Such operations are applied to each agent considering that the complete group has a memory which stores its own best positions seen so far by applying a competition principle. Numerical experiments have been conducted to compare the proposed method with the state-of-the-art methods on multi-modal benchmark functions. Besides, the proposed algorithm is also applied to the engineering problem of multicircle detection, achieving satisfactory results.

This paper is organized as follows. Section 2 introduces the basic biological aspects of the algorithm. In Section 3, the proposed algorithm and its characteristics are described. A numerical study on different multi-modal benchmark functions is presented in Section 4. Section 5 presents the application of the proposed algorithm to multi-circle detection whereas Section 6 shows the obtained results. Finally, in Section 7 the conclusions are discussed.

2. Biological Fundaments

The remarkable collective behavior of organisms such as swarming ants, schooling fish, and flocking birds has long captivated the attention of naturalists and scientists. Despite a long history of scientific investigation, just recently we are beginning to decipher the relationship between individuals and group-level properties [46]. Grouping individuals often have to make rapid decisions about where to move

or what behavior to perform, in uncertain and dangerous environments. However, each individual typically has only relatively local sensing ability [47]. Groups are, therefore, often composed of individuals that differ with respect to their informational status, and individuals are usually not aware of the informational state of others [48], such as whether they are knowledgeable about a pertinent resource or of a threat.

Animal groups are based on a hierarchic structure [49] which differentiates individuals according to a fitness principle known as dominance [50]. Such concept represents the domain of some individuals within a group and occurs when competition for resources leads to confrontation. Several studies [51, 52] have found that such animal behavior leads to stable groups with better cohesion properties among individuals.

Recent studies have illustrated how repeated interactions among grouping animals scale to collective behavior. They have also remarkably revealed that collective decision-making mechanisms across a wide range of animal group types, ranging from insects to birds (and even among humans in certain circumstances), seem to share similar functional characteristics [30, 34, 53]. Furthermore, at a certain level of description, collective decision-making in organisms shares essential common features such as general memory. Although some differences may arise, there are good reasons to increase communication between researchers working in collective animal behavior and those involved in cognitive science [33].

Despite the variety of behaviors and motions of animal groups, it is possible that many of the different collective behavioral patterns are generated by simple rules followed by individual group members. Some authors have developed different models, such as the self-propelled particle (SPP) model which attempts to capture the collective behavior of animal groups in terms of interactions between group members following a diffusion process [54–57].

On other hand, following a biological approach, Couzin et al. [33, 34] have proposed a model in which individual animals follow simple rules of thumb: (1) keep the position of best individuals, (2) move from or to nearby neighbors (local attraction or repulsion), (3) move randomly, and (4) compete for the space inside of a determined distance. Each individual thus admits three different movements: attraction, repulsion, or random, while holding two kinds of states: preserve the position or compete for a determined position. In the model, the movement experimented by each individual is decided randomly (according to an internal motivation); meanwhile the states are assumed according to fixed criteria.

The dynamical spatial structure of an animal group can be explained in terms of its history [54]. Despite this, the majority of the studies have failed in considering the existence of memory in behavioral models. However, recent researches [36, 58] have also shown the existence of collective memory in animal groups. The presence of such memory establishes that the previous history of the group structure influences the collective behavior exhibited in future stages. Such memory can contain the position of special group members (the dominant individuals) or the averaged movements produced by the group.

According to these new developments, it is possible to model complex collective behaviors by using simple individual rules and setting a general memory. In this work, the behavioral model of animal groups is employed for defining the evolutionary operators through the proposed metaheuristic algorithm. A memory is incorporated to store best animal positions (best solutions) considering a competition-dominance mechanism.

3. Collective Animal Behaviour Algorithm (CAB)

The CAB algorithm assumes the existence of a set of operations that resembles the interaction rules that model the collective animal behavior. In the approach, each solution within the search space represents an animal position. The "fitness value" refers to the animal dominance with respect to the group. The complete process mimics the collective animal behavior.

The approach in this paper implements a memory for storing best solutions (animal positions) mimicking the aforementioned biologic process. Such memory is divided into two different elements, one for maintaining the best found positions in each generation ($\mathbf{M}_g$) and the other for storing best history positions during the complete evolutionary process ($\mathbf{M}_h$).

3.1. Description of the CAB Algorithm. Like other metaheuristic approaches, the CAB algorithm is also an iterative process. It starts by initializing the population randomly, that is, generating random solutions or animal positions. The following four operations are thus applied until the termination criterion is met, that is, the iteration number NI is reached as follows.

(1) Keep the position of the best individuals.

(2) Move from or nearby neighbors (local attraction and repulsion).

(3) Move randomly.

(4) Compete for the space inside of a determined distance (updating the memory).

3.1.1. Initializing the Population. The algorithm begins by initializing a set $\mathbf{A}$ of N_p animal positions ($\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{N_p}\}$). Each animal position $\mathbf{a}_i$ is a D-dimensional vector containing the parameter values to be optimized, which are randomly and uniformly distributed between the prespecified lower initial parameter bound a_j^{low} and the upper initial parameter bound a_j^{high}:

$$a_{j,i} = a_j^{\text{low}} + \text{rand}\,(0,1) \cdot \left(a_j^{\text{high}} - a_j^{\text{low}}\right), \quad j = 1, 2, \ldots, D;\ i = 1, 2, \ldots, N_p. \tag{1}$$

with j and i being the parameter and individual indexes, respectively. Hence, $a_{j,i}$ is the jth parameter of the ith individual.

All the initial positions $\mathbf{A}$ are sorted according to the fitness function (dominance) to form a new individual set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_p}\}$, so that we can choose the best B positions and store them in the memory $\mathbf{M}_g$ and $\mathbf{M}_h$. The fact that both memories share the same information is only allowed at this initial stage.

3.1.2. Keep the Position of the Best Individuals. Analogously to the biological metaphor, this behavioral rule, typical in animal groups, is implemented as an evolutionary operation in our approach. In this operation, the first B elements of the new animal position set $\mathbf{A}(\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_B\})$ are generated. Such positions are computed by the values contained in the historic memory $\mathbf{M}_h$ considering a slight random perturbation around them. This operation can be modelled as follows:

$$\mathbf{a}_l = \mathbf{m}_h^l + \mathbf{v}, \tag{2}$$

where $l \in \{1, 2, \ldots, B\}$ while $\mathbf{m}_h^l$ represents the l-element of the historic memory $\mathbf{M}_h$ and $\mathbf{v}$ is a random vector holding an appropriate small length.

3.1.3. Move from or to Nearby Neighbours. From the biological inspiration, where animals experiment a random local attraction or repulsion according to an internal motivation, we implement the evolutionary operators that mimic them. For this operation, a uniform random number r_m is generated within the range $[0, 1]$. If r_m is less than a threshold H, a determined individual position is moved (attracted or repelled) considering the nearest best historical value of the group (the nearest position contained in $\mathbf{M}_h$); otherwise it is considered the nearest best value in the group of the current generation (the nearest position contained in $\mathbf{M}_g$). Therefore, such operation can be modeled as follows:

$$\mathbf{a}_i = \begin{cases} \mathbf{x}_i \pm r \cdot \left(\mathbf{m}_h^{\text{nearest}} - \mathbf{x}_i\right) & \text{with probability } H \\ \mathbf{x}_i \pm r \cdot \left(\mathbf{m}_g^{\text{nearest}} - \mathbf{x}_i\right) & \text{with probability } (1 - H), \end{cases} \tag{3}$$

where $i \in \{B+1, B+2, \ldots, N_p\}$, $\mathbf{m}_h^{\text{nearest}}$ and $\mathbf{m}_g^{\text{nearest}}$ represent the nearest elements of $\mathbf{M}_h$ and $\mathbf{M}_g$ to $\mathbf{x}_i$, while r is a random number between $[-1, 1]$. Therefore, if $r > 0$, the individual position $\mathbf{x}_i$ is attracted to the position $\mathbf{m}_h^{\text{nearest}}$ or $\mathbf{m}_g^{\text{nearest}}$; otherwise such movement is considered as a repulsion.

3.1.4. Move Randomly. Following the biological model, under some probability P an animal randomly changes its position. Such behavioral rule is implemented considering the next expression:

$$\mathbf{a}_i = \begin{cases} \mathbf{r} & \text{with probability } P \\ \mathbf{x}_i & \text{with probability } (1 - P), \end{cases} \tag{4}$$

where $i \in \{B + 1, B + 2, \ldots, N_p\}$ and $\mathbf{r}$ is a random vector defined within the search space. This operator is similar to reinitialize the particle in a random position as it is done by (1).

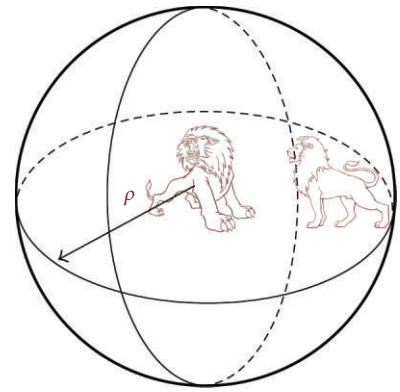


Figure 1: Dominance concept, presented when two animals confront each other inside of a ρ distance.

3.1.5. Compete for the Space Inside of a Determined Distance (Updating the Memory). Once the operations to preserve the position of the best individuals, to move from or to nearby neighbors and to move randomly, have all been applied to all the N_p animal positions, generating N_p new positions, it is necessary to update the memory $\mathbf{M}_h$.

In order to update the memory $\mathbf{M}_h$, the concept of dominance is used. Animals that interact in a group keep a minimum distance among them. Such distance ρ depends on how aggressive the animal behaves [50, 58]. Hence, when two animals confront each other inside of such distance, the most dominant individual prevails as the other withdraws. Figure 1 shows this process.

In the proposed algorithm, the historic memory $\mathbf{M}_h$ is updated considering the following procedure.

(1) The elements of $\mathbf{M}_h$ and $\mathbf{M}_g$ are merged into $\mathbf{M}_U (\mathbf{M}_U = \mathbf{M}_h \cup \mathbf{M}_g)$.

(2) Each element $\mathbf{m}_U^i$ of the memory $\mathbf{M}_U$ is compared pairwise with the remainder memory elements $(\{\mathbf{m}_U^1, \mathbf{m}_U^2, \ldots, \mathbf{m}_U^{2B-1}\})$. If the distance between both elements is less than ρ, the element holding a better performance in the fitness function will prevail; meanwhile the other will be removed.

(3) From the resulting elements of $\mathbf{M}_U$ (as they are obtained in Step 2), the B best value is selected to integrate the new $\mathbf{M}_h$.

Unsuitable values of ρ result in a lower convergence rate, longer computation time, larger function evaluation number, convergence to a local maximum, or unreliability of solutions. The ρ value is computed considering the following equation:

$$\rho = \frac{\prod_{j=1}^{D} \left(a_j^{\text{high}} - a_j^{\text{low}}\right)}{10 \cdot D}, \tag{5}$$

where a_j^{low} and a_j^{high} represent the prespecified lower bound and the upper bound of the j-parameter, respectively, within a D-dimensional space.

3.1.6. Computational Procedure. The computational procedure for the proposed algorithm can be summarized as follows.

Step 1. Set the parameters N_p, B, H, P, and NI.

Step 2. Generate randomly the position set $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{N_p}\}$ using (1).

Step 3. Sort $\mathbf{A}$, according to the objective function (dominance), building $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_p}\}$.

Step 4. Choose the first B positions of $\mathbf{X}$ and store them into the memory $\mathbf{M}_g$.

Step 5. Update $\mathbf{M}_h$ according to Section 3.1.5 (for the first iteration $\mathbf{M}_h = \mathbf{M}_g$).

Step 6. Generate the first B positions of the new solution set $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_B\}$. Such positions correspond to elements of $\mathbf{M}_h$ making a slight random perturbation around them: $\mathbf{a}_l = \mathbf{m}_h^l + \mathbf{v}$, $\mathbf{v}$ being a random vector holding an appropriate small length.

Step 7. Generate the rest of the $\mathbf{A}$ elements using the attraction, repulsion, and random movements:

for $i = B + 1 : N_p$
  if $(r_1 < 1 - P)$ then
  attraction and repulsion movement
    {if $(r_2 < H)$ then
      $\mathbf{a}_i = \mathbf{x}_i \pm r \cdot (\mathbf{m}_h^{\text{nearest}} - \mathbf{x}_i)$
      else if
      $\mathbf{a}_i = \mathbf{x}_i \pm r \cdot (\mathbf{m}_g^{\text{nearest}} - \mathbf{x}_i)$
    }
  else if
  random movement
    {
      $\mathbf{a}_i = \mathbf{r}$
    }
end for
where $r_1, r_2, r \in \text{rand}(0, 1)$.

Step 8. If NI is completed, the process is thus completed; otherwise go back to Step 3.

3.1.7. Optima Determination. Just after the optimization process has finished, an analysis of the final $\mathbf{M}_h$ memory is executed in order to find the global and significant local minima. For it, a threshold value Th is defined to decide which elements will be considered as a significant local minimum. Such threshold is thus computed as

$$\text{Th} = \frac{\max_{\text{fitness}}(\mathbf{M}_h)}{6}, \tag{6}$$

where $\max_{\text{fitness}}(\mathbf{M}_h)$ represents the best fitness value among $\mathbf{M}_h$ elements. Therefore, memory elements whose fitness values are greater than Th will be considered as global and local optima as other elements are discarded.

3.1.8. Capacities of CAB and Differences with PSO. Evolutionary algorithms (EAs) have been widely employed for solving complex optimization problems. These methods are found to be more powerful than conventional methods based on formal logics or mathematical programming [59]. Exploitation and exploration are two main features of the EA [60]. The exploitation phase searches around the current best solutions and selects the best candidates or solutions. The exploration phase ensures that the algorithm seeks the search space more efficiently in order to analyze potential unexplored areas.

The EAs do not have limitations in using different sources of inspiration (e.g., music-inspired [11] or physic-inspired charged system search [9]). However, nature is a principal inspiration for proposing new metaheuristic approaches, and the nature-inspired algorithms have been widely used in developing systems and solving problems [61]. Biologically inspired algorithms are one of the main categories of the nature-inspired metaheuristic algorithms. The efficiency of the bio inspired algorithms is due to their significant ability to imitate the best features in nature. More specifically, these algorithms are based on the selection of the most suitable elements in biological systems which have evolved by natural selection.

Particle swarm optimization (PSO) is undoubtedly one of the most employed EA methods that use biologically inspired concepts in the optimization procedure. Unfortunately, like other stochastic algorithms, PSO also suffers from the premature convergence [62], particularly in multi modal problems. Premature convergence, in PSO, is produced by the strong influence of the best particle in the evolution process. Such particle is used by the PSO movement equations as a main individual in order to attract other particles. Under such conditions, the exploitation phase is privileged by allowing the evaluation of new search position around the best individual. However, the exploration process is seriously damaged, avoiding searching in unexplored areas.

As an alternative to PSO, the proposed scheme modifies some evolution operators for allowing not only attracting but also repelling movements among particles. Likewise, instead of considering the best position as reference, our algorithm uses a set of neighboring elements that are contained in an incorporated memory. Such improvements allow increasing the algorithm's capacity to explore and to exploit the set of solutions which are operated during the evolving process.

In the proposed approach, in order to improve the balance between exploitation and exploration, we have introduced three new concepts. The first one is the "attracting and repelling movement", which outlines that one particle cannot be only attracted but also repelled. The application of this concept to the evolution operators (3) increases the capacity of the proposed algorithm to satisfactorily explore the search space. Since the process of attraction or repulsion of each particle is randomly determined, the possibility of

premature convergence is very low, even for cases that hold an exaggerated number of local minima (excessive number of multimodal functions).

The second concept is the use of the main individual. In the approach, the main individual, that is considered as pivot in the equations (in order to generate attracting and repulsive movements), is not the best (as in PSO) but one element ($\mathbf{m}_h^{\text{nearest}}$ or $\mathbf{m}_g^{\text{nearest}}$) of a set which is contained in memories that store the best individual seen so far. Such pivot is the nearest element in memory with regard to the individual whose position is necessary to evolve. Under such conditions, the points considered to prompt the movement of a new individual are multiple. Such fact allows to maintain a balance between exploring new positions and exploiting the best positions seen so-far.

Finally, the third concept is the use of an incorporated memory which stores the best individuals seen so far. As it has been discussed in Section 3.1.5, each candidate individual to be stored in the memory must compete with elements already contained in the memory in order to demonstrate that such new point is relevant. For the competition, the distance between each individual and the elements in the memory is used to decide pair-wise which individuals are actually considered. Then, the individual with better fitness value prevails whereas its pair is discarded. The incorporation of such concept allows simultaneous registering and refining of the best-individual set seen so far. This fact guarantees a high precision for final solutions of the multi-modal landscape through an extensive exploitation of the solution set.

3.1.9. Numerical Example. In order to demonstrate the algorithm's step-by-step operation, a numerical example has been set by applying the proposed method to optimize a simple function which is defined as follows:

$$f(x_1, x_2) = e^{-((x_1-4)^2-(x_2-4)^2)} + e^{-((x_1+4)^2-(x_2-4)^2)} + 2 \cdot e^{-((x_1)^2+(x_2)^2)} + 2 \cdot e^{-((x_1)^2-(x_2+4)^2)}. \tag{7}$$

Considering the interval of $-5 \leq x_1, x_2 \leq 5$, the function possesses two global maxima of value 2 at $(x_1, x_2) = (0, 0)$ and $(0, -4)$. Likewise, it holds two local minima of value 1 at $(-4, 4)$ and $(4, 4)$. Figure 2(a) shows the 3D plot of this function. The parameters for the CAB algorithm are set as $N_p = 10$, $B = 4$, $H = 0.8$, $P = 0.1$, $\rho = 3$, and $NI = 30$.

Like all evolutionary approaches, CAB is a population-based optimizer that attacks the starting point problem by sampling the objective function at multiple, randomly chosen, initial points. Therefore, after setting parameter bounds that define the problem domain, 10 (N_p) individuals ($\mathbf{i}_1$, $\mathbf{i}_2, \ldots, \mathbf{i}_{10}$) are generated using (1). Following an evaluation of each individual through the objective function (7), all are sorted decreasingly in order to build vector $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{10})$. Figure 2(b) depicts the initial individual distribution in the search space. Then, both memories $\mathbf{M}_g$ ($\mathbf{m}_g^1, \ldots, \mathbf{m}_g^4$) and $\mathbf{M}_h$ ($\mathbf{m}_h^1, \ldots, \mathbf{m}_h^4$) are filled with the first four (B) elements present in $\mathbf{X}$. Such memory elements are represented by solid points in Figure 2(c).

The new 10 individuals ($\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{10}$) are evolved at each iteration following three different steps: (1) keep the position of best individuals, (2) move from or nearby neighbors, and (3) move randomly. The first new four elements ($\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4$) are generated considering the first step (keeping the position of best individuals). Following such step, new individual positions are calculated as perturbed versions of all the elements which are contained in the $\mathbf{M}_h$ memory (that represent the best individuals known so far). Such perturbation is done by using $\mathbf{a}_l = \mathbf{m}_h^l + \mathbf{v}$ ($l \in 1, \ldots, 4$). Figure 2(d) shows a comparative view between the memory element positions and the perturbed values of ($\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4$).

The remaining 6 new positions ($\mathbf{a}_5, \ldots, \mathbf{a}_{10}$) are individually computed according to Steps 2 and 3 of the numerical example. For such operation, a uniform random number r_1 is generated within the range $[0, 1]$. If r_1 is less than $1 - P$, the new position $\mathbf{a}_j$ ($j \in 5, \ldots, 10$) is generated through Step 2; otherwise, $\mathbf{a}_j$ is obtained from a random reinitialization (Step 3) between search bounds.

In order to calculate a new position $\mathbf{a}_j$ at Step 2, a decision must be made on whether it should be generated by using the elements of $\mathbf{M}_h$ or $\mathbf{M}_g$. For such decision, a uniform random number r_2 is generated within the range $[0, 1]$. If r_2 is less than H, the new position $\mathbf{a}_j$ is generated by using $\mathbf{x}_j \pm r \cdot (\mathbf{m}_h^{\text{nearest}} - \mathbf{x}_j)$; otherwise, $\mathbf{a}_j$ is obtained by considering $\mathbf{x}_j \pm r \cdot (\mathbf{m}_g^{\text{nearest}} - \mathbf{x}_j)$, where $\mathbf{m}_h^{\text{nearest}}$ and $\mathbf{m}_g^{\text{nearest}}$ represent the closest elements to $\mathbf{x}_j$ in memory $\mathbf{M}_h$ and $\mathbf{M}_g$, respectively. In the first iteration, since there is not available information from previous steps, both memories $\mathbf{M}_h$ and $\mathbf{M}_g$ share the same information which is only allowed at this initial stage. Figure 2(e) shows graphically the whole procedure employed by Step 2 in order to calculate the new individual position $\mathbf{a}_8$ whereas Figure 2(f) presents the positions of all new individuals ($\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{10}$).

Finally, after all new positions ($\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{10}$) have been calculated, memories $\mathbf{M}_h$ and $\mathbf{M}_g$ must be updated. In order to update $\mathbf{M}_h$, new calculated positions ($\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{10}$) are arranged according to their fitness values by building vector $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{10})$. Then, the elements of $\mathbf{M}_h$ are replaced by the first four elements in $\mathbf{X}$ (the best individuals of its generation). In order to calculate the new elements of $\mathbf{M}_h$, current elements of $\mathbf{M}_h$ (the present values) and $\mathbf{M}_g$ (the updated values) are merged into $\mathbf{M}_U$. Then, by using the dominance concept (explained in Section 3.1.5) over $\mathbf{M}_U$, the best four values are selected to replace the elements in $\mathbf{M}_g$. Figures 2(g) and 2(h) show the updating procedure for both memories. Applying the dominance (see Figure 2(g)), since the distances $a = \text{dist}(\mathbf{m}_h^3, \mathbf{m}_g^4)$, $b = \text{dist}(\mathbf{m}_h^2, \mathbf{m}_g^3)$, and $c = \text{dist}(\mathbf{m}_h^1, \mathbf{m}_g^1)$ are less than $\rho = 3$, elements with better fitness evaluation will build the new memory $\mathbf{M}_h$. Figure 2(h) depicts final memory configurations. The circles and solid circles points represent the elements of $\mathbf{M}_g$ and $\mathbf{M}_h$, respectively, whereas the bold squares perform as elements shared by both memories. Therefore, if the complete procedure is repeated over 30 iterations, the memory $\mathbf{M}_h$ will contain the 4 global and local maxima as elements. Figure 2(i) depicts the final configuration after 30 iterations.

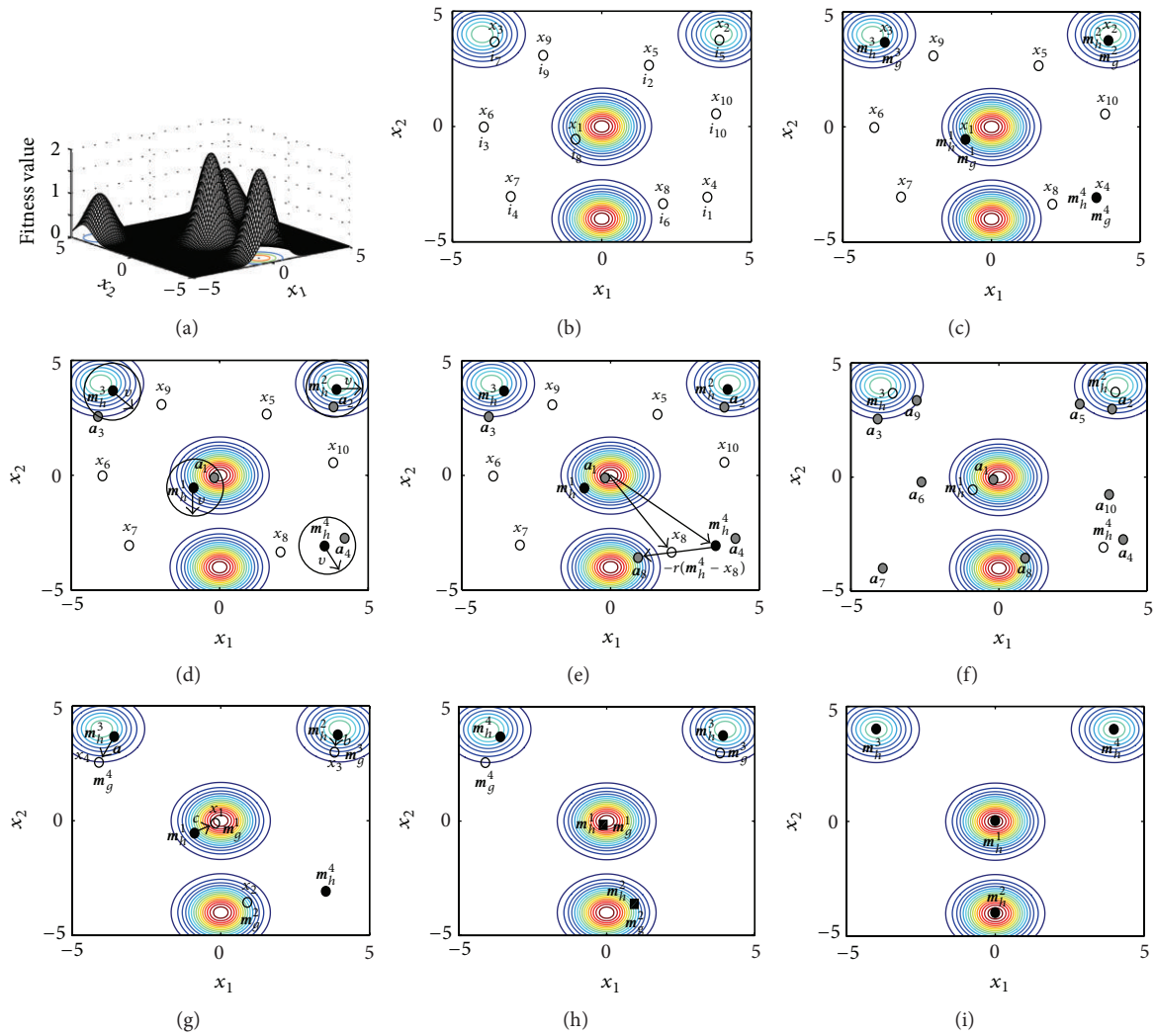


FIGURE 2: CAB numerical example. (a) 3D plot of the function used as example. (b) Initial individual distribution. (c) Initial configuration of memories $\mathbf{M}_g$ and $\mathbf{M}_h$. (d) The computation of the first four individuals ($\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4$). (e) It shows the procedure employed by Step 2 in order to calculate the new individual position $\mathbf{a}_8$. (f) Positions of all new individuals ($\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{10}$). (g) Application of the dominance concept over elements of $\mathbf{M}_g$ and $\mathbf{M}_h$. (h) Final memory configurations of $\mathbf{M}_g$ and $\mathbf{M}_h$ after the first iteration. (i) Final memory configuration of $\mathbf{M}_h$ after 30 iterations.

4. Results on Multimodal Benchmark Functions

In this section, the performance of the proposed algorithm is tested. Section 4.1 describes the experiment methodology. Sections 4.2 and 4.3 report on a comparison between the CAB experimental results and other multimodal metaheuristic algorithms for different kinds of optimization problems.

4.1. Experiment Methodology. In this section, we will examine the search performance of the proposed CAB by using a test suite of 8 benchmark functions with different complexities. They are listed in Tables 1 and 2. The suite mainly contains some representative, complicated, and multimodal functions with several local optima. These functions are normally regarded as difficult to be optimized as they are particularly challenging to the applicability and efficiency of multimodal metaheuristic algorithms. The performance measurements considered at each experiment are the following:

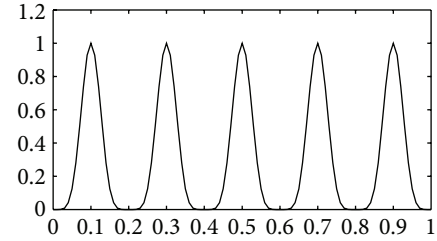
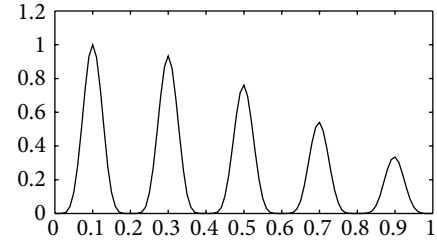
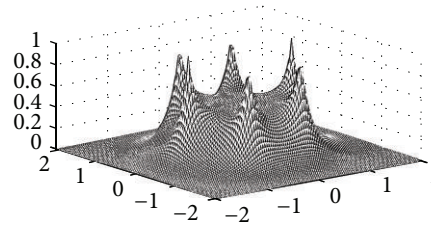
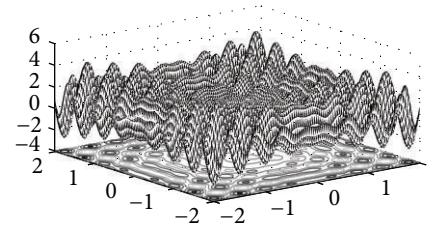
(i) the consistency of locating all known optima;

(ii) the averaged number of objective function evaluations that are required to find such optima (or the running time under the same condition).

The experiments compare the performance of CAB against the deterministic crowding [17], the probabilistic crowding [18], the sequential fitness sharing [15], the clearing procedure [20], the clustering based niching (CBN) [19], the species conserving genetic algorithm (SCGA) [21], the elitist-population strategy (AEGA) [22], the clonal selection algorithm [24], and the artificial immune network (AiNet) [25].

Since the approach solves real-valued multimodal functions, we have used, in the GA approaches, consistent real coding variable representation, uniform crossover, and

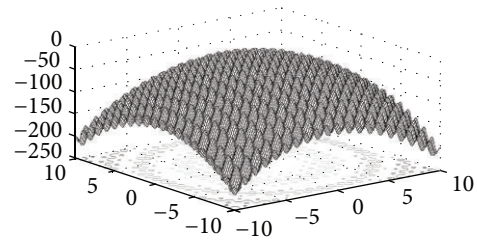
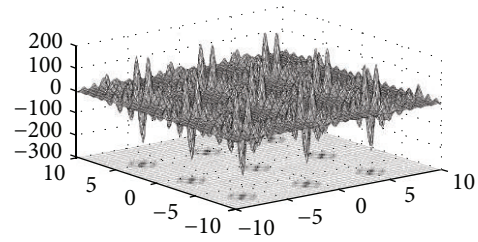
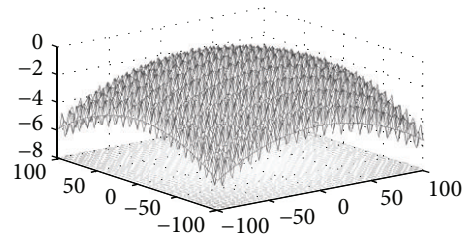
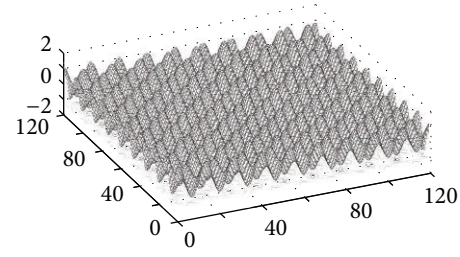
Table 1: The test suite of multimodal functions for Experiment 4.2.

Function	Search space	Sketch
Deb's function 5 optima		
$f_1 = \sin^6(5\pi x)$	$x \in [0, 1]$	
Deb's decreasing function 5 optima		
$f_2(x) = 2^{-2((x-0.1)/0.9)^2} \cdot \sin(5\pi x)$	$x \in [0, 1]$	
Roots function 6 optima		
$f_3(z) = \dfrac{1}{1 + \lvert z^6 + 1 \rvert}$	$z \in C, z = x_1 + ix_2$ $x_1, x_2 \in [-2, 2]$	
Two dimensional multi-modal function 100 optima		
$f_4(x_1, x_2) = x_1 \sin(4\pi x_1) - x_2 \sin(4\pi x_2 + \pi) + 1$	$x_1, x_2 \in [-2, 2]$	

mutation operators for each algorithm seeking a fair comparison. The crossover probability $P_c = 0.8$ and the mutation probability $P_m = 0.1$ have been used. We use the standard tournament selection operator with a tournament size of 2 in our implementation of sequential fitness sharing, clearing procedure, CBN, clonal selection algorithm, and SCGA. On the other hand, the parameter values for the AiNet algorithm have been defined as suggested in [25], with the mutation strength $\beta = 100$, the suppression threshold $\sigma_{s(aiNet)} = 0.2$, and the update rate $d = 40\%$.

In the case of the CAB algorithm, the parameters are set to $N_p = 200$, $B = 100$, $P = 0.8$, and $H = 0.6$. Once they have

Table 2: The test suite of multimodal functions used in the Experiment 4.3.

Function	Search space	Sketch
Rastringin's function 100 optima $f_5(x_1, x_2) = -(20 + x_1^2 + x_2^2 - 10(\cos(2\pi x_1) + \cos(2\pi x_2)))$	$x_1, x_2 \in [-10, 10]$	
Shubert function 18 optima $f_6(x_1, x_2) = -\prod_{i=1}^{2}\sum_{j=1}^{5}\cos((j+1)x_i + j)$	$x_1, x_2 \in [-10, 10]$	
Griewank function 100 optima $f_7(x_1, x_2) = \frac{1}{4000}\sum_{i=1}^{2}x_i^2 - \prod_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{2}}\right) + 1$	$x_1, x_2 \in [-100, 100]$	
Modified Griewank function 100 optima $f_8(x_1, x_2) = \frac{\cos(0.5x_1) + \cos(0.5x_2)}{4000} + \cos(10x_1)\cos(10x_2)$	$x_1, x_2 \in [0, 120]$	

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 given in the literature. An optimum o_j is considered as found if $\exists x_i \in \text{Pop}(k = T) \mid d(x_i, o_j) < 0.005$, where $\text{Pop}(k = T)$ is the complete population at the end of the run T and x_i is an individual in $\text{Pop}(k = T)$.

All algorithms have been tested in MATLAB over the same Dell OptiPlex GX260 computer with a Pentium 4 2.66 G HZ processor, running Windows XP operating system over 1 Gb of memory. Next sections present experimental results for multimodal optimization problems which have been divided into two groups with different purposes. The first one consists of functions with smooth landscapes and well-defined optima (local and global values), while the second gathers functions holding rough landscapes and complex location optima.

4.2. Comparing CAB Performance for Smooth Landscapes Functions. This section presents a performance comparison for different algorithms solving multimodal problems f_1–f_4 in Table 1. The aim is to determine whether CAB is more efficient and effective than other existing algorithms for finding all multiple optima of f_1–f_4. The stopping criterion analyzes if the number-identified optima cannot be further increased over 10 successive generations after the first 100 generations; then the execution will be stopped. Four measurements have been employed to evaluate the performance:

(i) the average of optima found within the final population (NO);

(ii) the average distance between multiple optima detected by the algorithm and their closest individuals in the final population (DO);

(iii) the average of function evaluations (FE);

(iv) the average of execution time in seconds (ET).

Table 3 provides a summarized performance comparison among several algorithms. Best results have been bold faced. From the NO measure, CAB always finds better or equally optimal solutions for the multimodal problems f_1–f_4. It is evident that each algorithm can find all optima of f_1. For function f_2, only AEGA, clonal selection algorithm, aiNet, and CAB can eventually find all optima each time. For function f_3, clearing procedure, SCGA, AEGA, and CAB can get all optima at each run. For function f_4, deterministic crowding leads to premature convergence and all other algorithms cannot get any better results, but CAB yet can find all multiple optima 48 times in 50 runs and its average successful rate for each run is higher than 99%. By analyzing the DO measure in Table 3, CAB has obtained the best score for all the multimodal problems except for f_3. In the case of f_3, the solution precision of CAB is only worse than that of clearing procedure. On the other hand, CAB has smaller standard deviations in the NO and DO measures than all other algorithms and hence its solution is more stable.

From the FE measure in Table 3, it is clear that CAB needs fewer function evaluations than other algorithms considering the same termination criterion. Recall that all algorithms use the same conventional crossover and mutation operators. It can be easily deduced from results that the CAB algorithm is able to produce better search positions (better compromise between exploration and exploitation), in a more efficient and effective way than other multimodal search strategies.

To validate that CAB improvement over other algorithms occurs as a result of CAB producing better search positions over iterations, Figure 3 shows the comparison of CAB and other multimodal algorithms for f_4. The initial populations for all algorithms have 200 individuals. In the final population of CAB, the 100 individuals belonging to the $\mathbf{M}_h$ memory correspond to the 100 multiple optima, while, on the contrary, the final population of the other nine algorithms fail consistently in finding all optima, despite that they have superimposed several times over some previously found optima.

When comparing the execution time (ET) in Table 3, CAB uses significantly less time to finish than other algorithms. The situation can be registered by the reduction of the redundancy in the $\mathbf{M}_h$ memory due to competition (dominance) criterion. All these comparisons show that CAB generally outperforms all other multimodal algorithms regarding efficacy and efficiency.

4.3. Comparing CAB Performance in Rough Landscapes Functions. This section presents the performance comparison among different algorithms solving multimodal optimization problems which are listed in Table 2. Such problems hold lots of local optima and very rugged landscapes. The goal of multimodal optimizers is to find as many global optima as possible and possibly good local optima. Rastrigin's function f_5 and Griewank's function f_7 have 1 and 18 global optima, respectively, becoming practical as to test whether a multimodal algorithm can find a global optimum and at least 80 higher fitness local optima to validate the algorithms' performance.

Our main objective in these experiments is to determine whether CAB is more efficient and effective than other existing algorithms for finding the multiple high fitness optima of functions f_5–f_8. In the experiments, the initial population size for all algorithms has been set to 1000. For sequential fitness sharing, clearing procedure, CBN, clonal selection, SCGA, and AEGA, we have set the distance threshold σ_s to 5. The algorithms' stopping criterion checks whenever the number of optima found cannot be further increased in 50 successive generations after the first 500 generations. If such condition prevails, then the algorithm is halted. We still evaluate the performance of all algorithms using the aforementioned four measures NO, DO, FE, and ET.

Table 4 provides a summary of the performance comparison among different algorithms. From the NO measure, we observe that CAB could always find more optimal solutions for the multimodal problems f_5–f_8. For Rastrigin's function f_5, only CAB can find all multiple high fitness optima 49 times out of 50 runs and its average successful rate for each run is higher than 97%. On the contrary, other algorithms cannot find all multiple higher fitness optima for any run. For f_6, 5 algorithms (clearing procedure, SCGA, AEGA, clonal selection algorithm, AiNet, and CAB) can get all multiple higher fitness maxima for each run, respectively. For Griewank's function (f_7), only CAB can get all multiple higher fitness optima for each run. In case of the modified Griewank's function (f_8), it has numerous optima whose value is always

Table 3: Performance comparison among the multimodal optimization algorithms for the test functions f_1–f_4. The standard unit in the column ET is seconds. (For all the parameters, numbers in parentheses are the standard deviations.) Bold faced letters represent best obtained results.

Function	Algorithm	NO	DO	FE	ET
f_1	Deterministic crowding	**5** (**0**)	1.52×10^{-4} (1.38×10^{-4})	7,153 (358)	0.091 (0.013)
	Probabilistic crowding	**5** (**0**)	3.63×10^{-4} (6.45×10^{-5})	10,304 (487)	0.163 (0.011)
	Sequential fitness sharing	**5** (**0**)	4.76×10^{-4} (6.82×10^{-5})	9,927 (691)	0.166 (0.028)
	Clearing procedure	**5** (**0**)	1.27×10^{-4} (2.13×10^{-5})	5,860 (623)	0.128 (0.021)
	CBN	**5** (**0**)	2.94×10^{-4} (4.21×10^{-5})	10,781 (527)	0.237 (0.019)
	SCGA	**5** (**0**)	1.16×10^{-4} (3.11×10^{-5})	6,792 (352)	0.131 (0.009)
	AEGA	**5** (**0**)	4.6×10^{-5} (1.35×10^{-5})	2,591 (278)	0.039 (0.007)
	Clonal selection algorithm	**5** (**0**)	1.99×10^{-4} (8.25×10^{-5})	15,803 (381)	0.359 (0.015)
	AiNet	**5** (**0**)	1.28×10^{-4} (3.88×10^{-5})	12,369 (429)	0.421 (0.021)
	CAB	**5** (**0**)	$\mathbf{1.69 \times 10^{-5}}$ ($\mathbf{5.2 \times 10^{-6}}$)	**1,776** (**125**)	**0.020** (**0.009**)
f_2	Deterministic crowding	3.53 (0.73)	3.61×10^{-3} (6.88×10^{-4})	6,026 (832)	0.271 (0.06)
	Probabilistic crowding	4.73 (0.64)	2.82×10^{-3} (8.52×10^{-4})	10,940 (9517)	0.392 (0.07)
	Sequential fitness sharing	4.77 (0.57)	2.33×10^{-3} (4.36×10^{-4})	12,796 (1,430)	0.473 (0.11)
	Clearing procedure	4.73 (0.58)	4.21×10^{-3} (1.24×10^{-3})	8,465 (773)	0.326 (0.05)
	CBN	4.70 (0.53)	2.19×10^{-3} (4.53×10^{-4})	14,120 (2,187)	0.581 (0.14)
	SCGA	4.83 (0.38)	3.15×10^{-3} (4.71×10^{-4})	10,548 (1,382)	0.374 (0.09)
	AEGA	**5** (**0**)	1.38×10^{-4} (2.32×10^{-5})	3,605 (426)	0.102 (0.04)
	Clonal selection algorithm	**5** (**0**)	1.37×10^{-3} (6.87×10^{-4})	21,922 (746)	0.728 (0.06)
	AiNet	**5** (**0**)	1.22×10^{-3} (5.12×10^{-4})	18,251 (829)	0.664 (0.08)
	CAB	**5** (**0**)	$\mathbf{4.5 \times 10^{-5}}$ ($\mathbf{8.56 \times 10^{-6}}$)	**2,065** (**92**)	**0.08** (**0.007**)
f_3	Deterministic crowding	4.23 (1.17)	7.79×10^{-4} (4.76×10^{-4})	11,009 (1,137)	1.07 (0.13)
	Probabilistic crowding	4.97 (0.64)	2.35×10^{-3} (7.14×10^{-4})	16,391 (1,204)	1.72 (0.12)
	Sequential fitness sharing	4.87 (0.57)	2.56×10^{-3} (2.58×10^{-3})	14,424 (2,045)	1.84 (0.26)
	Clearing procedure	**6** (**0**)	$\mathbf{7.43 \times 10^{-5}}$ ($\mathbf{4.07 \times 10^{-5}}$)	12,684 (1,729)	1.59 (0.19)
	CBN	4.73 (1.14)	1.85×10^{-3} (5.42×10^{-4})	18,755 (2,404)	2.03 (0.31)
	SCGA	**6** (**0**)	3.27×10^{-4} (7.46×10^{-5})	13,814 (1,382)	1.75 (0.21)
	AEGA	**6** (**0**)	1.21×10^{-4} (8.63×10^{-5})	6,218 (935)	0.53 (0.07)
	Clonal selection algorithm	5.50 (0.51)	4.95×10^{-3} (1.39×10^{-3})	25,953 (2,918)	2.55 (0.33)
	AiNet	4.8 (0.33)	3.89×10^{-3} (4.11×10^{-4})	20,335 (1,022)	2.15 (0.10)
	CAB	**6** (**0**)	9.87×10^{-5} (1.69×10^{-5})	**4,359** (**75**)	**0.11** (**0.023**)
f_4	Deterministic crowding	76.3 (11.4)	4.52×10^{-3} (4.17×10^{-3})	1,861,707 (329,254)	21.63 (2.01)
	Probabilistic crowding	92.8 (3.46)	3.46×10^{-3} (9.75×10^{-4})	2,638,581 (597,658)	31.24 (5.32)
	Sequential fitness sharing	89.9 (5.19)	2.75×10^{-3} (6.89×10^{-4})	2,498,257 (374,804)	28.47 (3.51)
	Clearing procedure	89.5 (5.61)	3.83×10^{-3} (9.22×10^{-4})	2,257,964 (742,569)	25.31 (6.24)
	CBN	90.8 (6.50)	4.26×10^{-3} (1.14×10^{-3})	2,978,385 (872,050)	35.27 (8.41)
	SCGA	91.4 (3.04)	3.73×10^{-3} (2.29×10^{-3})	2,845,789 (432,117)	32.15 (4.85)
	AEGA	95.8 (1.64)	1.44×10^{-4} (2.82×10^{-5})	1,202,318 (784,114)	12.17 (2.29)
	Clonal selection algorithm	92.1 (4.63)	4.08×10^{-3} (8.25×10^{-3})	3,752,136 (191,849)	45.95 (1.56)
	AiNet	93.2 (7.12)	3.74×10^{-3} (5.41×10^{-4})	2,745,967 (328,176)	38.18 (3.77)
	CAB	**100** (**2**)	$\mathbf{2.31 \times 10^{-5}}$ ($\mathbf{5.87 \times 10^{-6}}$)	**697,578** (**57,089**)	**5.78** (**1.26**)

the same. However, CAB still can find all global optima with an effectiveness rate of 95%.

From the FE and ET measures in Table 4, we can clearly observe that CAB uses significantly fewer function evaluations and a shorter running time than all other algorithms under the same termination criterion. Moreover, deterministic crowding leads to premature convergence as CAB is at least 2.5, 3.8, 4, 3.1, 4.1, 3.7, 1.4, 7.9, and 4.9 times faster than all others, respectively, according to Table 4 for functions f_5–f_8.

5. Application of CAB in Multicircle Detection

5.1. Individual Representation. In order to detect circle shapes, candidate images must be preprocessed first by the well-known Canny algorithm which yields a single-pixel

Table 4: Performance comparison among multimodal optimization algorithms for the test functions f_5–f_8. The standard unit of the column ET is seconds (numbers in parentheses are standard deviations). Bold faced letters represent best results.

Function	Algorithm	NO	DO	FE	ET
f_5	Deterministic crowding	62.4 (14.3)	4.72×10^{-3} (4.59×10^{-3})	1,760,199 (254,341)	14.62 (2.83)
	Probabilistic crowding	84.7 (5.48)	1.50×10^{-3} (9.38×10^{-4})	2,631,627 (443,522)	34.39 (5.20)
	Sequential fitness sharing	76.3 (7.08)	3.51×10^{-3} (1.66×10^{-3})	2,726,394 (562,723)	36.55 (7.13)
	Clearing procedure	93.6 (2.31)	2.78×10^{-3} (1.20×10^{-3})	2,107,962 (462,622)	28.61 (6.47)
	CBN	87.9 (7.78)	4.33×10^{-3} (2.82×10^{-3})	2,835,119 (638,195)	37.05 (8.23)
	SCGA	97.4 (4.80)	1.34×10^{-3} (8.72×10^{-4})	2,518,301 (643,129)	30.27 (7.04)
	AEGA	99.4 (1.39)	6.77×10^{-4} (3.18×10^{-4})	978,435 (71,135)	10.56 (4.81)
	Clonal selection algorithm	90.6 (9.95)	3.15×10^{-3} (1.47×10^{-3})	5,075,208 (194,376)	58.02 (2.19)
	AiNet	93.8 (7.8)	2.11×10^{-3} (3.2×10^{-3})	3,342,864 (549,452)	51.65 (6.91)
	CAB	**100** (**2**)	$2.22 \times \mathbf{10^{-4}}$ $(3.1 \times \mathbf{10^{-5}})$	**680,211** (**12,547**)	**7.33** (**1.84**)
f_6	Deterministic crowding	9.37 (1.91)	3.26×10^{-3} (5.34×10^{-4})	832,546 (75,413)	4.58 (0.57)
	Probabilistic crowding	15.17 (2.43)	2.87×10^{-3} (5.98×10^{-4})	1,823,774 (265,387)	12.92 (2.01)
	Sequential fitness sharing	15.29 (2.14)	1.42×10^{-3} (5.29×10^{-4})	1,767,562 (528,317)	14.12 (3.51)
	Clearing procedure	**18** (**0**)	1.19×10^{-3} (6.05×10^{-4})	1,875,729 (265,173)	11.20 (2.69)
	CBN	14.84 (2.70)	4.39×10^{-3} (2.86×10^{-3})	2,049,225 (465,098)	18.26 (4.41)
	SCGA	4.83 (0.38)	1.58×10^{-3} (4.12×10^{-4})	2,261,469 (315,727)	13.71 (1.84)
	AEGA	**18** (**0**)	3.34×10^{-4} (1.27×10^{-4})	656,639 (84,213)	3.12 (1.12)
	Clonal selection algorithm	**18** (**0**)	3.42×10^{-3} (1.58×10^{-3})	4,989,856 (618,759)	33.85 (5.36)
	AiNet	**18** (**0**)	2.11×10^{-3} (3.31×10^{-3})	3,012,435 (332,561)	26.32 (2.54)
	CAB	**18** (**0**)	$\mathbf{1.02 \times 10^{-4}}$ $(\mathbf{4.27 \times 10^{-5}})$	**431,412** (**21,034**)	**2.21** (**0.51**)
f_7	Deterministic crowding	52.6 (8.86)	3.71×10^{-3} (1.54×10^{-3})	2,386,960 (221,982)	19.10 (2.26)
	Probabilistic crowding	79.2 (4.94)	3.48×10^{-3} (3.79×10^{-3})	3,861,904 (457,862)	43.53 (4.38)
	Sequential fitness sharing	63.0 (5.49)	4.76×10^{-3} (3.55×10^{-3})	3,619,057 (565,392)	42.98 (6.35)
	Clearing procedure	79.4 (4.31)	2.95×10^{-3} (1.64×10^{-3})	3,746,325 (594,758)	45.42 (7.64)
	CBN	71.3 (9.26)	3.29×10^{-3} (4.11×10^{-3})	4,155,209 (465,613)	48.23 (5.42)
	SCGA	94.9 (8.18)	2.63×10^{-3} (1.81×10^{-3})	3,629,461 (373,382)	47.84 (0.21)
	AEGA	98 (2)	1.31×10^{-3} (8.76×10^{-4})	1,723,342 (121,043)	12,54 (1.31)
	Clonal selection algorithm	89.2 (5.44)	3.02×10^{-3} (1.63×10^{-3})	5,423,739 (231,004)	47.84 (6.09)
	AiNet	92.7 (3.21)	2.79×10^{-3} (3.19×10^{-4})	4,329,783 (167,932)	41.64 (2.65)
	CAB	**100** (**1**)	$\mathbf{3.32 \times 10^{-4}}$ $(\mathbf{5.25 \times 10^{-5}})$	**953,832** (**9,345**)	**8.82** (**1.51**)
f_8	Deterministic crowding	44.2 (7.93)	4.45×10^{-3} (3.63×10^{-3})	2,843,452 (353,529)	23.14 (3.85)
	Probabilistic crowding	70.1 (8.36)	2.52×10^{-3} (1.47×10^{-3})	4,325,469 (574,368)	49.51 (6.72)
	Sequential fitness sharing	58.2 (9.48)	4.14×10^{-3} (3.31×10^{-3})	4,416,150 (642,415)	54.43 (12.6)
	Clearing procedure	67.5 (10.11)	2.31×10^{-3} (1.43×10^{-3})	4,172,462 (413,537)	52.39 (7.21)
	CBN	53.1 (7.58)	4.36×10^{-3} (3.53×10^{-3})	4,711,925 (584,396)	61.07 (8.14)
	SCGA	87.3 (9.61)	3.15×10^{-3} (2.07×10^{-3})	3,964,491 (432,117)	53.87 (8.46)
	AEGA	90.6 (1.65)	2.55×10^{-3} (9.55×10^{-4})	2,213,754 (412,538)	16.21 (3.19)
	Clonal selection algorithm	74.4 (7.32)	3.52×10^{-3} (2.19×10^{-3})	5,835,452 (498,033)	74.26 (5.47)
	AiNet	83.2 (6.23)	3.11×10^{-3} (2.41×10^{-4})	4,123,342 (213,864)	60.38 (5.21)
	CAB	**97** (**2**)	$\mathbf{1.54 \times 10^{-3}}$ $(\mathbf{4.51 \times 10^{-4}})$	**1,121,523** (**51,732**)	**12.21** (**2.66**)

edge-only image. Then, the (x_i, y_i) coordinates for each edge pixel p_i are stored inside the edge vector $P = \{p_1, p_2, \ldots, p_{N_p}\}$, with N_p being the total number of edge pixels. Each circle C uses three edge points as individuals in the optimization algorithm. In order to construct such individuals, three indexes p_i, p_j, and p_k are selected from vector P, considering the circle's contour that connects them. Therefore, the circle $C = \{p_i, p_j, p_k\}$ that crosses over such points may be considered as a potential solution for the detection problem. Considering the configuration of the edge points shown by Figure 4, the circle center (x_0, y_0) and the radius r of C can be computed as follows:

$$(x - x_0)^2 + (y - y_0)^2 = r^2. \quad (8)$$

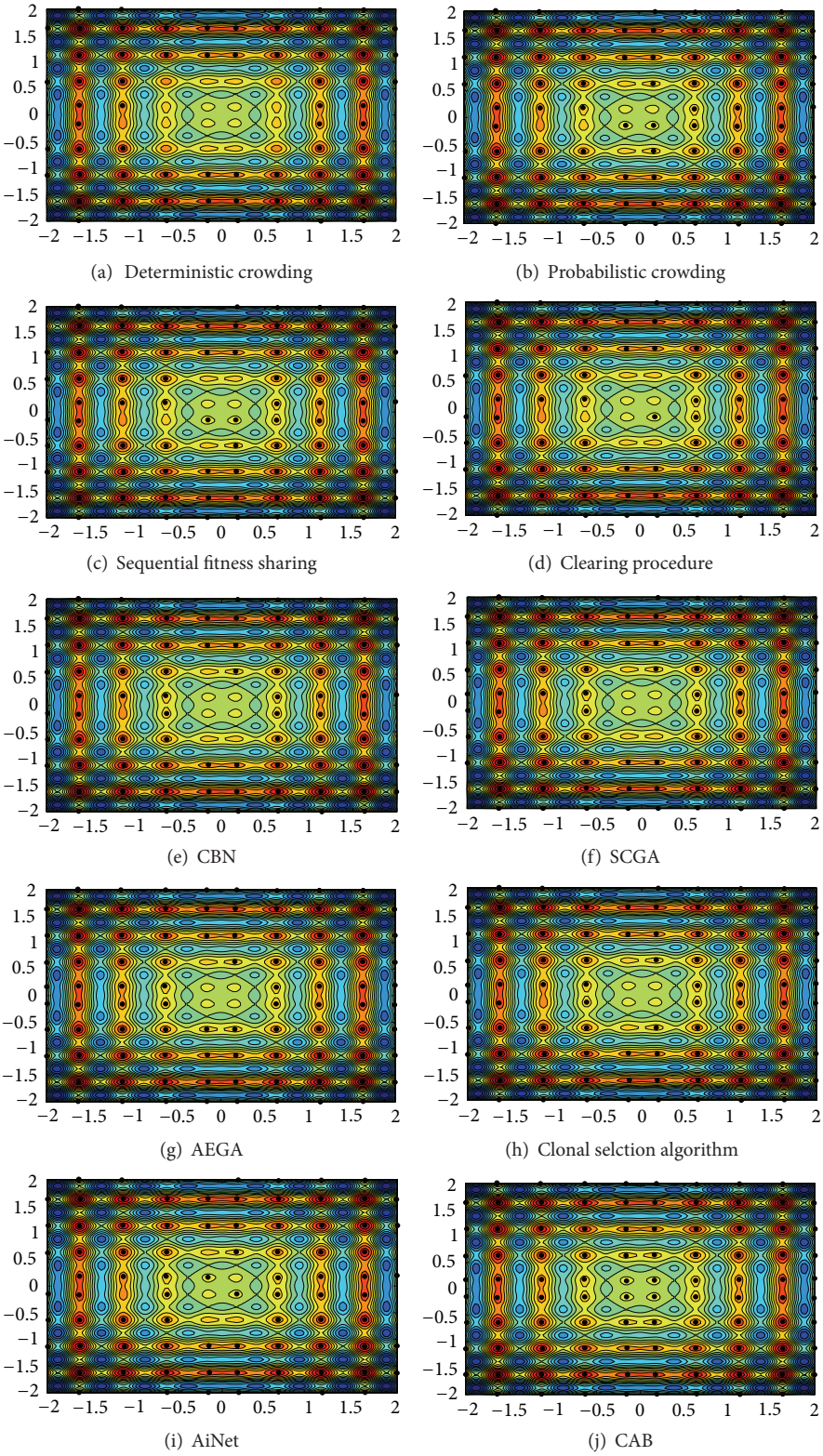


Figure 3: Typical results of the maximization of f_4. (a)–(j) Local and global optima located by all ten algorithms in the performance comparison.

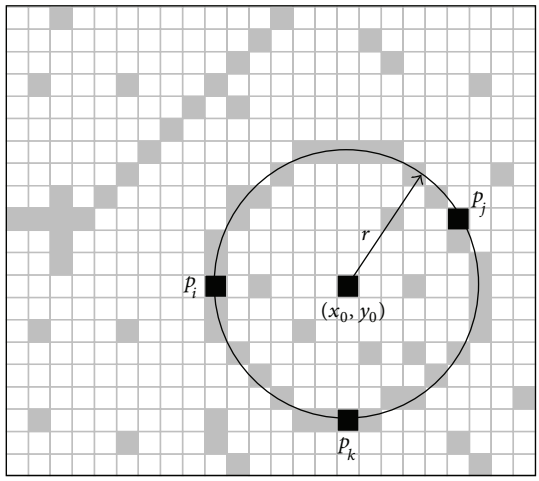


Figure 4: Circle candidate (individual) built from the combination of points p_i, p_j, and p_k.

Consider

$$\mathbf{A} = \begin{bmatrix} x_j^2 + y_j^2 - \left(x_i^2 + y_i^2\right) & 2 \cdot \left(y_j - y_i\right) \\ x_k^2 + y_k^2 - \left(x_i^2 + y_i^2\right) & 2 \cdot \left(y_k - y_i\right) \end{bmatrix},$$
$$\mathbf{B} = \begin{bmatrix} 2 \cdot \left(x_j - x_i\right) & x_j^2 + y_j^2 - \left(x_i^2 + y_i^2\right) \\ 2 \cdot \left(x_k - x_i\right) & x_k^2 + y_k^2 - \left(x_i^2 + y_i^2\right) \end{bmatrix}, \tag{9}$$

$$x_0 = \frac{\det(\mathbf{A})}{4\left(\left(x_j - x_i\right)\left(y_k - y_i\right) - \left(x_k - x_i\right)\left(y_j - y_i\right)\right)},$$
$$y_0 = \frac{\det(\mathbf{B})}{4\left(\left(x_j - x_i\right)\left(y_k - y_i\right) - \left(x_k - x_i\right)\left(y_j - y_i\right)\right)}, \tag{10}$$

$$r = \sqrt{\left(x_0 - x_d\right)^2 + \left(y_0 - y_d\right)^2}, \tag{11}$$

with $\det(\cdot)$ being the determinant and $d \in \{i, j, k\}$. Figure 2 illustrates the parameters defined by (8) to (11).

5.2. Objective Function. In order to calculate the error produced by a candidate solution C, a set of test points is calculated as a virtual shape which, in turn, must be validated, that is if it really exists in the edge image. The test set is represented by $S = \{s_1, s_2, \ldots, s_{N_s}\}$, where N_s is the number of points over which the existence of an edge point, corresponding to C, should be validated. In our approach, the set S is generated by the midpoint circle algorithm (MCA) [63]. The MCA is a searching method which seeks the required points for drawing a circle digitally. Therefore MCA calculates the necessary number of test points N_s to totally draw the complete circle. Such a method is considered the fastest because MCA avoids computing square-root calculations by comparing the pixel separation distances among them.

The objective function $J(C)$ represents the matching error produced between the pixels S of the circle candidate C (animal position) and the pixels that actually exist in the edge image, yielding

$$J(C) = 1 - \frac{\sum_{v=1}^{N_s} E\left(x_v, y_v\right)}{N_s}, \tag{12}$$

where $E(x_i, y_i)$ is a function that verifies the pixel existence in (x_v, y_v), with $(x_v, y_v) \in S$, and N_s being the number of pixels lying on the perimeter corresponding to C currently under testing. Hence, function $E(x_v, y_v)$ is defined as

$$E\left(x_v, y_v\right) = \begin{cases} 1 & \text{if the pixel } \left(x_v, y_v\right) \text{ is an edge point} \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

A value near zero of $J(C)$ implies a better response from the "circularity" operator. Figure 5 shows the procedure to evaluate a candidate solution C with its representation as a virtual shape S. In Figure 5(b), the virtual shape is compared to the original image, point by point, in order to find coincidences between virtual and edge points. The virtual shape is built from points p_i, p_j, and p_k shown by Figure 5(a). The virtual shape S gathers 56 points ($N_s = 56$) with only 18 of such points existing in both images (shown as blue points plus red points in Figure 5(c)) yielding $\sum_{v=1}^{N_s} E(x_v, y_v) = 18$ and therefore $J(C) \approx 0.67$.

5.3. The Multiple-Circle Detection Procedure. In order to detect multiple circles, most detectors simply apply a one-minimum optimization algorithm, which is able to detect only one circle at a time, repeating the same process several times as previously detected primitives are removed from the image. Such algorithms iterate until there are no more candidates left in the image.

On the other hand, the method in this paper is able to detect single or multiples circles through only one optimization step. The multidetection procedure can be summarized as follows: guided by the values of a matching function, the whole group of encoded candidate circles is evolved through the set of evolutionary operators. The best circle candidate (global optimum) is considered to be the first detected circle over the edge-only image. An analysis of the historical memory $\mathbf{M}_h$ is thus executed in order to identify other local optima (other circles).

In order to find other possible circles contained in the image, the historical memory $\mathbf{M}_h$ is carefully examined. The approach aims to explore all elements, one at a time, assessing which of them represents an actual circle in the image. Since several elements can represent the same circle (i.e., circles slightly shifted or holding small deviations), a distinctiveness factor $D_{A,B}$ is required to measure the mismatch between two given circles (A and B). Such distinctiveness factor is defined as follows:

$$D_{A,B} = \left|x_A - x_B\right| + \left|y_A - y_B\right| + \left|r_A - r_B\right|, \tag{14}$$

with (x_A, y_A) and r_A being the central coordinates and radius of the circle C_A, respectively, while (x_B, y_B) and r_B represent

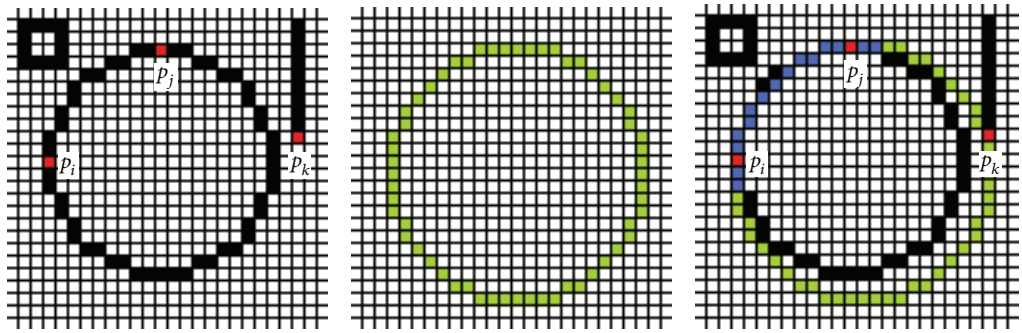


FIGURE 5: Evaluation of candidate solutions C: the image in (a) shows the original image while (b) presents the virtual shape generated including points p_i, p_j, and p_k. The image in (c) shows coincidences between both images marked by blue or red pixels while the virtual shape is also depicted in green.

the corresponding parameters of the circle C_B. One threshold value $E_{s_{\text{Th}}}$ is also calculated to decide whether two circles must be considered different or not. Th is computed as:

$$\text{Th} = \frac{r_{\max} - r_{\min}}{d}, \tag{15}$$

where $[r_{\min}, r_{\max}]$ is the feasible radii's range and d is a sensitivity parameter. By using a high d value, two very similar circles would be considered different while a smaller value for d would consider them as similar shapes. In this work, after several experiments, the d value has been set to 2.

Thus, since the historical memory $\mathbf{M}_h$ $\{C_1^{\mathbf{M}}, C_2^{\mathbf{M}}, \ldots, C_B^{\mathbf{M}}\}$ groups the elements in descending order according to their fitness values, the first element $C_1^{\mathbf{M}}$, whose fitness value represents the best value $J(C_1^{\mathbf{M}})$, is assigned to the first circle. Then, the distinctiveness factor $(D_{C_1^{\mathbf{M}}, C_2^{\mathbf{M}}})$ over the next element $C_2^{\mathbf{M}}$ is evaluated with respect to the prior $C_1^{\mathbf{M}}$. If $D_{C_1^{\mathbf{M}}, C_2^{\mathbf{M}}} > \text{Th}$, then $C_2^{\mathbf{M}}$ is considered as a new circle; otherwise the next element $C_3^{\mathbf{M}}$ is selected. This process is repeated until the fitness value $J(C_i^{\mathbf{M}})$ reaches a minimum threshold J_{Th}. According to such threshold, other values above J_{Th} represent individuals (circles) that are considered significant while other values lying below such boundary are considered as false circles and hence they are not contained in the image. After several experiments the value of J_{Th} is set to $(J(C_1^{\mathbf{M}})/10)$.

The fitness value of each detected circle is characterized by its geometric properties. Big and well-drawn circles normally represent points in the search space with higher fitness values whereas small and dashed circles describe points with lower fitness values. Likewise, circles with similar geometric properties, such as radius and size tend to represent locations holding similar fitness values. Considering that the historical memory $\mathbf{M}_h$ groups the elements in descending order according to their fitness values, the proposed procedure allows the cancelling of those circles which belong to the same circle and hold a similar fitness value.

5.4. Implementation of CAB Strategy for Circle Detection. The implementation of the proposed algorithm can be summarized in the following steps.

Step 1. Adjust the algorithm parameters N_p, B, H, P, NI, and d.

Step 2. Randomly generate a set of N_p candidate circles (position of each animal) $\mathbf{C} = \{C_1, C_2, \ldots, C_{N_p}\}$ set using (1).

Step 3. Sort $\mathbf{C}$ according to the objective function (dominance) to build $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_p}\}$.

Step 4. Choose the first B positions of $\mathbf{X}$ and store them into the memory $\mathbf{M}_g$.

Step 5. Update $\mathbf{M}_h$ according to Section 3.1.5. (during the first iteration: $\mathbf{M}_h = \mathbf{M}_g$).

Step 6. Generate the first B positions of the new solution set $\mathbf{C}(\{C_1, C_2, \ldots, C_B\})$. Such positions correspond to the elements of $\mathbf{M}_h$ making a slight random perturbation around them:

$$C_l = \mathbf{m}_h^l + \mathbf{v}, \quad \text{with being } \mathbf{v} \text{ a random vector of a small enough length.} \tag{16}$$

Step 7. Generate the rest of the $\mathbf{C}$ elements using the attraction, repulsion, and random movements:

for $i = B + 1 : N_p$
  if $(r_1 < P)$ then
  attraction and repulsion movement
    {if $(r_2 < H)$ then
      $C_i = \mathbf{x}_i \pm r \cdot (\mathbf{m}_h^{\text{nearest}} - \mathbf{x}_i)$
      else if
      $C_i = \mathbf{x}_i \pm r \cdot (\mathbf{m}_g^{\text{nearest}} - \mathbf{x}_i)$
    }
  else if
  random movement
    {
      $C_i = \mathbf{r}$

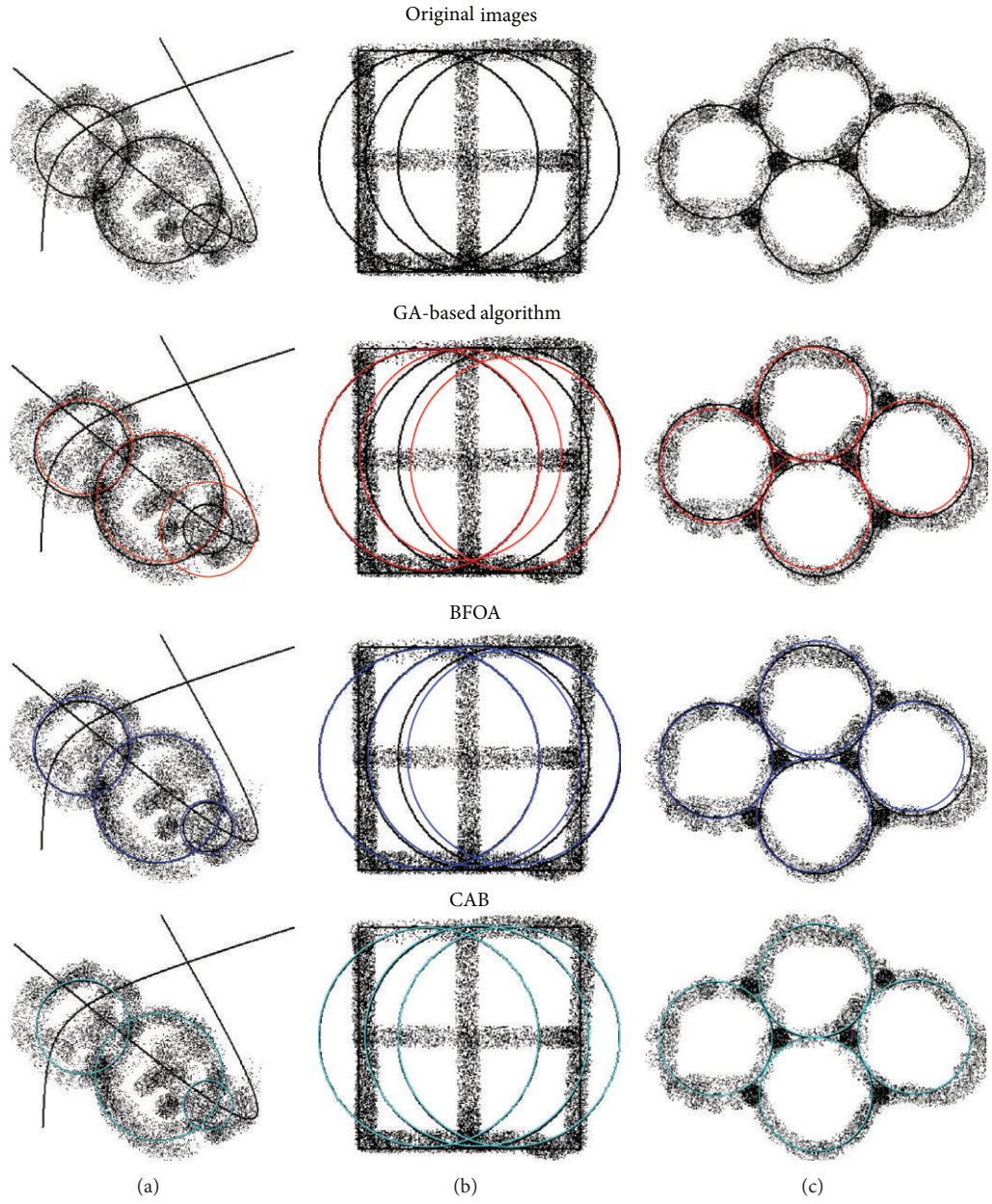


Figure 6: Synthetic images and their detected circles for GA-based algorithm, the BFOA method, and the proposed CAB algorithm.

}

end for where $r_1, r_2, r \in \text{rand}(0, 1)$.

Step 8. If NI is not completed, the process goes back to Step 3. Otherwise, the best values in $\mathbf{M}_h$ $\{C_1^{\mathbf{M}}, C_2^{\mathbf{M}}, \ldots, C_B^{\mathbf{M}}\}$ represent the best solutions (the best found circles).

Step 9. The element with the highest fitness value $J(C_1^{\mathbf{M}})$ is identified as the first circle C_1.

Step 10. The distinctiveness factor $D_{C_m^{\mathbf{M}}, C_{m-1}^{\mathbf{M}}}$ of circle $C_m^{\mathbf{M}}$ (element m) with the next highest probability is evaluated with respect to $C_{m-1}^{\mathbf{M}}$. If $D_{C_m^{\mathbf{M}}, C_{m-1}^{\mathbf{M}}} > \text{Th}$, then $C_m^{\mathbf{M}}$ is considered as a new circle; otherwise the next action is evaluated.

Step 11. Step 10 is repeated until the element's fitness value reaches $(J(C_1^{\mathbf{M}})/10)$.

The number of candidate circles N_p is set considering a balance between the number of local minima to be detected and the computational complexity. In general terms, a large value of N_p suggests the detection of a great amount of circles at the cost of excessive computer time. After exhaustive experimentation, it has been found that a value of $N_p = 30$ represents the best tradeoff between computational overhead

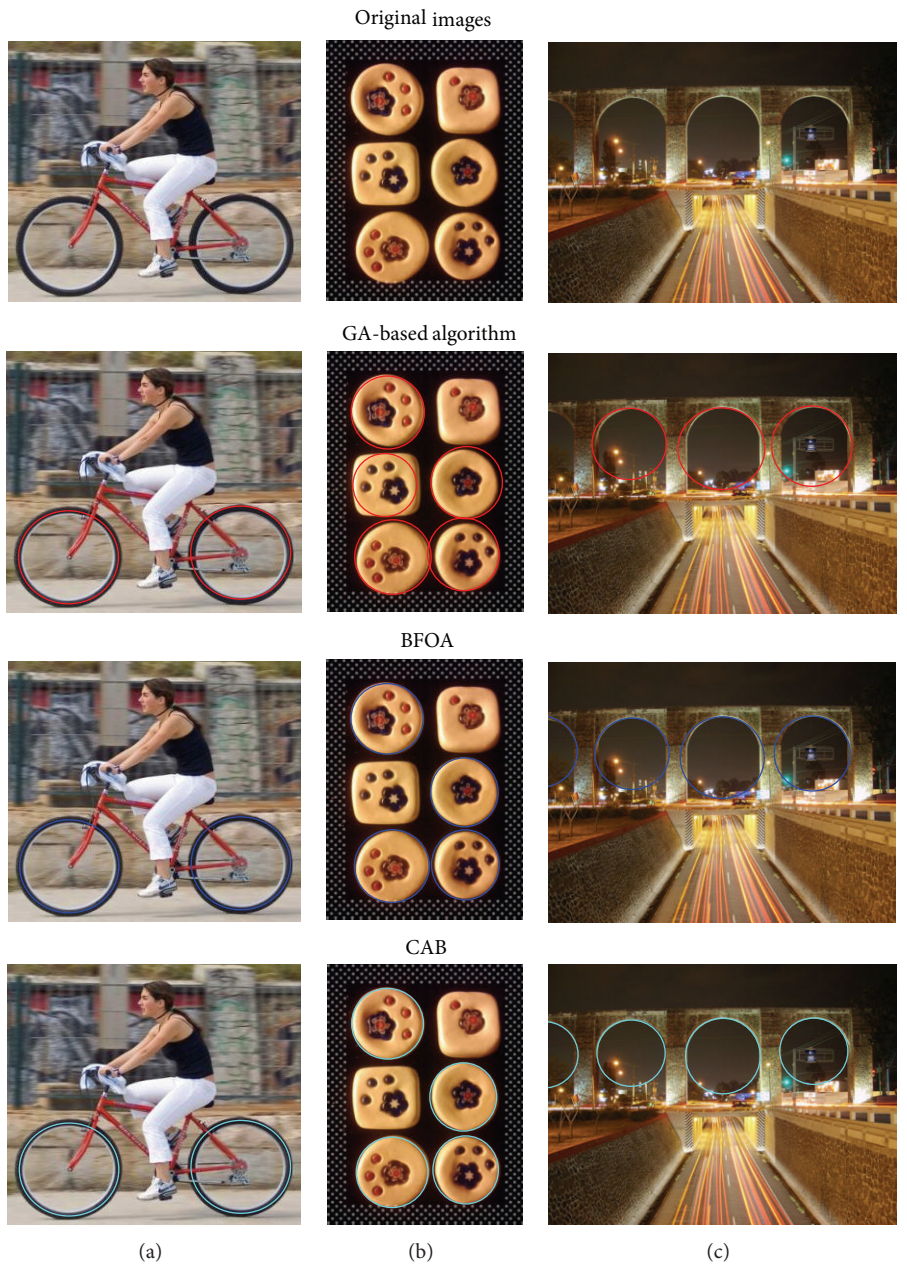


Figure 7: Real-life images and their detected circles for GA-based algorithm, the BFOA method, and the proposed CAB algorithm.

and accuracy and therefore such value is used throughout the study.

6. Results on Multicircle Detection

In order to achieve the performance analysis, the proposed approach is compared to the BFAO detector, the GA-based algorithm, and the RHT method over an image set.

The GA-based algorithm follows the proposal of Ayala-Ramirez et al. [41], which considers the population size as 70, the crossover probability as 0.55, the mutation probability as 0.10, and the number of elite individuals as 2. The roulette wheel selection and the 1-point crossover operator are both applied. The parameter setup and the fitness function follow the configuration suggested in [41]. The BFAO algorithm follows the implementation from [45] considering the experimental parameters as $S = 50$, $N_c = 350$, $N_s = 4$, $N_{ed} = 1$, $P_{ed} = 0.25$, $d_{attract} = 0.1$, $w_{attract} = 0.2$, $w_{repellant} = 10$, $h_{repellant} = 0.1$, $\lambda = 400$, and $\psi = 6$. Such values are found to be the best configuration set according to [45]. Both, the GA-based algorithm and the BAFO method use the same objective function that is defined by (12). Likewise, the RHT method has been implemented as it is described in [40]. Finally, Table 5 presents the parameters for the CAB algorithm used in this work. They have been kept for all test images after being experimentally defined.

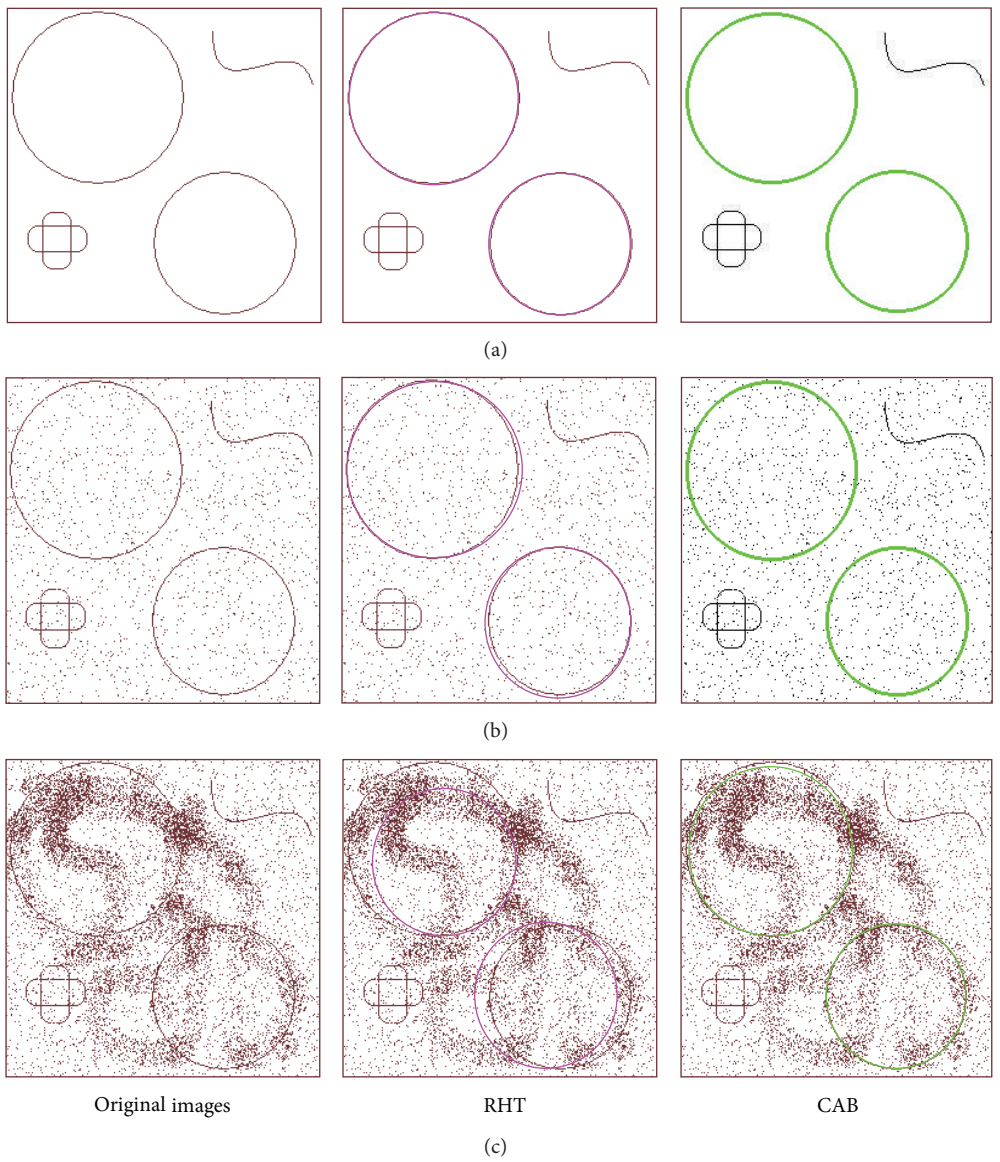


Figure 8: Relative performance of the RHT and the CAB.

Table 5: CAB detector parameters.

N_p	H	P	B	NI
30	0.5	0.1	12	200

Images rarely contain perfectly shaped circles. Therefore, with the purpose of testing accuracy for a single circle, the detection is challenged by a ground-truth circle which is determined from the original edge map. The parameters $(x_{\text{true}}, y_{\text{true}}, r_{\text{true}})$ representing the testing circle are computed using (6)–(9) for three circumference points over the manually drawn circle. Considering the center and the radius of the detected circle are defined as (x_D, y_D) and r_D, the error score (Es) can be accordingly calculated as

$$\text{Es} = \eta \cdot \left(\left|x_{\text{true}} - x_D\right| + \left|y_{\text{true}} - y_D\right|\right) + \mu \cdot \left|r_{\text{true}} - r_D\right|. \quad (17)$$

The central point difference $(|x_{\text{true}} - x_D| + |y_{\text{true}} - y_D|)$ represents the center shift for the detected circle as it is compared to a benchmark circle. The radio mismatch $(|r_{\text{true}} - r_D|)$ accounts for the difference between their radii. η and μ represent two weighting parameters which are to be applied separately to the central point difference and to the radio mismatch for the final error Es. At this work, they are chosen as $\eta = 0.05$ and $\mu = 0.1$. Such particular choice ensures that the radii difference would be strongly weighted in comparison to the difference of central circular positions between the manually detected and the machine-detected circles. Here we assume that if Es is found to be less than 1, then the algorithm gets a success; otherwise, we say that it has failed to detect the edge circle. Note that for $\eta = 0.05$ and $\mu = 0.1$, Es < 1 means that the maximum difference of radius tolerated is 10 while the maximum mismatch in the location of the center can be 20 (in number of pixels). In order to appropriately compare

Table 6: The averaged execution time, detection rate, and the averaged multiple error for the GA-based algorithm, the BFOA method, and the proposed CAB algorithm, considering six test images (shown by Figures 6 and 7).

Image	Averaged execution time ± standard deviation (s)			Success rate (DR) (%)			Averaged ME ± standard deviation		
	GA	BFOA	CAB	GA	BFOA	CAB	GA	BFOA	CAB
Synthetic images									
(a)	2.23 ± (0.41)	1.71 ± (0.51)	**0.21 ± (0.22)**	88	99	**100**	0.41 ± (0.044)	0.33 ± (0.052)	**0.22 ± (0.033)**
(b)	3.15 ± (0.39)	2.80 ± (0.65)	**0.36 ± (0.24)**	79	92	**99**	0.51 ± (0.038)	0.37 ± (0.032)	**0.26 ± (0.041)**
(c)	4.21 ± (0.11)	3.18 ± (0.36)	**0.20 ± (0.19)**	74	88	**100**	0.48 ± (0.029)	0.41 ± (0.051)	**0.15 ± (0.036)**
Natural images									
(a)	5.11 ± (0.43)	3.45 ± (0.52)	**1.10 ± (0.24)**	90	96	**100**	0.45 ± (0.051)	0.41 ± (0.029)	**0.25 ± (0.037)**
(b)	6.33 ± (0.34)	4.11 ± (0.14)	**1.61 ± (0.17)**	83	89	**100**	0.81 ± (0.042)	0.77 ± (0.051)	**0.37 ± (0.055)**
(c)	7.62 ± (0.97)	5.36 ± (0.17)	**1.95 ± (0.41)**	84	92	**99**	0.92 ± (0.075)	0.88 ± (0.081)	**0.41 ± (0.066)**

Table 7: P values produced by Wilcoxons test comparing CAB to GA and BFOA over the averaged ME from Table 2.

Image	P value	
	CAB versus GA	CAB versus BFOA
Synthetic images		
(a)	$1.8061e-004$	$1.8288e-004$
(b)	$1.7454e-004$	$1.9011e-004$
(c)	$1.7981e-004$	$1.8922e-004$
Natural images		
(a)	$1.7788e-004$	$1.8698e-004$
(b)	$1.6989e-004$	$1.9124e-004$
(c)	$1.7012e-004$	$1.9081e-004$

the detection results, the detection rate (DR) is introduced as a performance index. DR is defined as the percentage of reaching detection success after a certain number of trials. For "success" it does mean that the compared algorithm is able to detect all circles contained in the image, under the restriction that each circle must hold the condition Es < 1. Therefore, if at least one circle does not fulfil the condition of Es < 1, the complete detection procedure is considered as a failure.

In order to use an error metric for multiple-circle detection, the averaged Es produced from each circle in the image is considered. Such criterion, defined as the multiple error (ME), is calculated as follows:

$$\mathrm{ME} = \left(\frac{1}{NC}\right) \cdot \sum_{R=1}^{NC} \mathrm{Es}_R, \tag{18}$$

where NC represents the number of circles within the image according to a human expert.

Figure 6 shows three synthetic images and the resulting images after applying the GA-based algorithm [41], the BFOA method [45], and the proposed approach. Figure 7 presents experimental results considering three natural images. The performance is analyzed by considering 35 different executions for each algorithm. Table 6 shows the averaged execution time, the detection rate in percentage, and the averaged multiple error (ME), considering six test images (shown by Figures 6 and 7). The best entries are boldfaced in Table 6. Close inspection reveals that the proposed method is able to achieve the highest success rate keeping the smallest error, still requiring less computational time for most cases.

In order to statistically analyze the results in Table 6, a nonparametric significance proof known as the Wilcoxon's rank test [64–66] 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 multiple error (ME) data. Table 7 reports the P values produced by Wilcoxon's test for a pairwise comparison of the multiple error (ME), considering two groups gathered as CAB versus GA and CAB versus BFOA. As a null hypothesis, it is assumed that there is no difference between the values of the two algorithms. The alternative hypothesis considers an existent difference between the values of both approaches. All P values reported in Table 7 are less than 0.05 (5% significance level) which is a strong evidence against the null hypothesis, indicating that the best CAB mean values for the performance are statistically significant which has not occurred by chance.

Figure 8 demonstrates the relative performance of CAB in comparison with the RHT algorithm as it is described in [40]. All images belonging to the test are complicated and contain different noise conditions. The performance analysis is achieved by considering 35 different executions for each algorithm over the three images. The results, exhibited in Figure 8, present the median-run solution (when the runs were ranked according to their final ME value) obtained throughout the 35 runs. On the other hand, Table 4 reports the corresponding averaged execution time, detection rate (in %) and average multiple error (using (10)) for CAB and RHT algorithms over the set of images (the best results are boldfaced). Table 8 shows a decrease in performance of the RHT algorithm as noise conditions change. Yet the CAB algorithm holds its performance under the same circumstances.

7. Conclusions

In recent years, several metaheuristic optimization methods have been inspired from nature-like phenomena. In this paper, a new multimodal optimization algorithm known as the collective animal behavior algorithm (CAB) has been

Table 8: Average time, detection rate, and averaged error for CAB and HT, considering three test images.

Image	Average time ± standard deviation (s)		Success rate (DR) (%)		Average ME ± standard deviation	
	RHT	CAB	RHT	CAB	RHT	CAB
(a)	7.82 ± (0.34)	**0.30 ± (0.10)**	**100**	**100**	0.19 ± (0.041)	**0.11 ± (0.017)**
(b)	8.65 ± (0.48)	**0.22 ± (0.13)**	64	**100**	0.47 ± (0.037)	**0.13 ± (0.019)**
(c)	10.65 ± (0.48)	**0.25 ± (0.12)**	11	**100**	1.21 ± (0.033)	**0.15 ± (0.014)**

introduced. In CAB, the searcher agents emulate a group of animals that interact with each other depending on simple behavioral rules which are modeled as mathematical operators. Such operations are applied to each agent considering that the complete group hold a memory to store its own best positions seen so far, using a competition principle.

CAB has been experimentally evaluated over a test suite consisting of 8 benchmark multimodal functions for optimization. The performance of CAB has been compared to some other existing algorithms including deterministic crowding [17], probabilistic crowding [18], sequential fitness sharing [15], clearing procedure [20], clustering-based niching (CBN) [19], species conserving genetic algorithm (SCGA) [21], elitist-population strategies (AEGA) [22], clonal selection algorithm [24], and the artificial immune network (aiNet) [25]. All experiments have demonstrated that CAB generally outperforms all other multimodal metaheuristic algorithms regarding efficiency and solution quality, typically showing significant efficiency speedups. The remarkable performance of CAB is due to two different features: (i) operators allow a better exploration of the search space, increasing the capacity to find multiple optima; (ii) the diversity of solutions contained in the $\mathbf{M}_h$ memory in the context of multimodal optimization is maintained and even improved through of the use of a competition principle (dominance concept).

The proposed algorithm is also applied to the engineering problem of multicircle detection. Such a process is faced as a multimodal optimization problem. In contrast to other heuristic methods that employ an iterative procedure, the proposed CAB method is able to detect single or multiple circles over a digital image by running only one optimization cycle. The CAB algorithm searches the entire edge map for circular shapes by using a combination of three noncollinear edge points as candidate circles (animal positions) in the edge-only image. A matching function (objective function) is used to measure the existence of a candidate circle over the edge map. Guided by the values of such matching function, the set of encoded candidate circles is evolved using the CAB algorithm so that the best candidate can be fitted into an actual circle. After the optimization has been completed, an analysis of the embedded memory is executed in order to find the significant local minima (remaining circles). The overall approach generates a fast subpixel detector which can effectively identify multiple circles in real images despite that some circular objects exhibit a significant occluded portion.

In order to test the circle detection performance, both speed and accuracy have been compared. Score functions are defined by (17) and (18) in order to measure accuracy and effectively evaluate the mismatch between manually detected and machine-detected circles. We have demonstrated that the CAB method outperforms both the GA (as described in [41]) and the BFOA (as described in [45]) within a statistically significant framework (Wilcoxon test). In contrast to the CAB method, the RHT algorithm [40] shows a decrease in performance under noisy conditions. Yet the CAB algorithm holds its performance under the same circumstances. Finally, Table 6 indicates that the CAB method can yield better results on complicated and noisy images compared with the GA and the BFOA methods.

References

[1] A. Ahrari, M. Shariat-Panahi, and A. A. Atai, "GEM: a novel evolutionary optimization method with improved neighborhood search," *Applied Mathematics and Computation*, vol. 210, no. 2, pp. 376–386, 2009.

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

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

[4] 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, Palo Alto, Calif, USA, 1990.

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

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

[7] 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.

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

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

[10] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[11] 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.

[12] 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.

[13] 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.

[14] X. Z. Gao, X. Wang, and S. J. Ovaska, "Uni-modal and multi-modal optimization using modified Harmony Search methods," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 10, pp. 2985–2996, 2009.

[15] 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.

[16] 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 (ICEC '96)*, pp. 786–791, May 1996.

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

[18] O. J. Mengshoel and D. E. Goldberg, "Probability crowding: deterministic crowding with probabilistic replacement," in *Proceedings of the International Conference on Genetic and Evolutionary Computation Conferenc (GECCO '99)*, W. Banzhaf, Ed., pp. 409–416, Orlando, Fla, USA, 1999.

[19] 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.

[20] 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, IEEE Press, Nagoya, Japan, May 1996.

[21] 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.

[22] 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.

[23] L. Y. Wei and M. Zhao, "A niche hybrid genetic algorithm for global optimization of continuous multimodal functions," *Applied Mathematics and Computation*, vol. 160, no. 3, pp. 649–661, 2005.

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

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

[26] 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.

[27] 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.

[28] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[29] D. B. Chen and C. X. Zhao, "Particle swarm optimization with adaptive population size and its application," *Applied Soft Computing Journal*, vol. 9, no. 1, pp. 39–48, 2009.

[30] D. Sumper, "The principles of collective animal behaviour," *Philosophical Transactions of the Royal Society B*, vol. 361, no. 1465, pp. 5–22, 2006.

[31] O. Petit and R. Bon, "Decision-making processes: the case of collective movements," *Behavioural Processes*, vol. 84, no. 3, pp. 635–647, 2010.

[32] A. Kolpas, J. Moehlis, T. A. Frewen, and I. G. Kevrekidis, "Coarse analysis of collective motion with different communication mechanisms," *Mathematical Biosciences*, vol. 214, no. 1-2, pp. 49–57, 2008.

[33] I. D. Couzin, "Collective cognition in animal groups," *Trends in Cognitive Sciences*, vol. 13, no. 1, pp. 36–43, 2008.

[34] I. D. Couzin and J. Krause, "Self-organization and collective behavior in vertebrates," *Advances in the Study of Behavior*, vol. 32, pp. 1–75, 2003.

[35] N. W. F. Bode, D. W. Franks, and A. Jamie Wood, "Making noise: emergent stochasticity in collective motion," *Journal of Theoretical Biology*, vol. 267, no. 3, pp. 292–299, 2010.

[36] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective memory and spatial sorting in animal groups," *Journal of Theoretical Biology*, vol. 218, no. 1, pp. 1–11, 2002.

[37] I. D. Couzin, "Collective minds," *Nature*, vol. 445, no. 7129, pp. 715–728, 2007.

[38] S. Bazazi, J. Buhl, J. J. Hale et al., "Collective motion and cannibalism in locust migratory bands," *Current Biology*, vol. 18, no. 10, pp. 735–739, 2008.

[39] T. J. Atherton and D. J. Kerbyson, "Using phase to represent radius in the coherent circle Hough transform," in *Proceedings of the IEE Colloquium on the Hough Transform*, IEE, London, UK, 1993.

[40] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: randomized Hough transform (RHT)," *Pattern Recognition Letters*, vol. 11, no. 5, pp. 331–338, 1990.

[41] V. Ayala-Ramirez, C. H. Garcia-Capulin, A. Perez-Garcia, and R. E. Sanchez-Yanez, "Circle detection on images using genetic algorithms," *Pattern Recognition Letters*, vol. 27, no. 6, pp. 652–657, 2006.

[42] E. Cuevas, N. Ortega-Sánchez, D. Zaldivar, and M. Pérez-Cisneros, "Circle detection by harmony search optimization," *Journal of Intelligent and Robotic Systems*, vol. 66, no. 3, pp. 359–376, 2012.

[43] E. Cuevas, D. Oliva, D. Zaldivar, M. Pérez-Cisneros, and H. Sossa, "Circle detection using electro-magnetism optimization," *Information Sciences*, vol. 182, no. 1, pp. 40–55, 2012.

[44] E. Cuevas, D. Zaldivar, M. Pérez-Cisneros, and M. Ramírez-Ortegón, "Circle detection using discrete differential evolution optimization," *Pattern Analysis and Applications*, vol. 14, no. 1, pp. 93–107, 2011.

[45] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "Automatic circle detection on digital images with an adaptive bacterial foraging algorithm," *Soft Computing*, vol. 14, no. 11, pp. 1151–1164, 2010.

[46] N. W. F. Bode, A. J. Wood, and D. W. Franks, "The impact of social networks on animal collective motion," *Animal Behaviour*, vol. 82, no. 1, pp. 29–38, 2011.

[47] B. H. Lemasson, J. J. Anderson, and R. A. Goodwin, "Collective motion in animal groups from a neurobiological perspective: the adaptive benefits of dynamic sensory loads and selective attention," *Journal of Theoretical Biology*, vol. 261, no. 4, pp. 501–510, 2009.

[48] M. Bourjade, B. Thierry, M. Maumy, and O. Petit, "Decision-making in przewalski horses (equus ferus przewalskii) is driven by the ecological contexts of collective movements," *Ethology*, vol. 115, no. 4, pp. 321–330, 2009.

[49] A. Bang, S. Deshpande, A. Sumana, and R. Gadagkar, "Choosing an appropriate index to construct dominance hierarchies in animal societies: a comparison of three indices," *Animal Behaviour*, vol. 79, no. 3, pp. 631–636, 2010.

[50] Y. Hsu, R. L. Earley, and L. L. Wolf, "Modulation of aggressive behaviour by fighting experience: mechanisms and contest outcomes," *Biological Reviews of the Cambridge Philosophical Society*, vol. 81, no. 1, pp. 33–74, 2006.

[51] M. Broom, A. Koenig, and C. Borries, "Variation in dominance hierarchies among group-living animals: modeling stability and the likelihood of coalitions," *Behavioral Ecology*, vol. 20, no. 4, pp. 844–855, 2009.

[52] K. L. Bayly, C. S. Evans, and A. Taylor, "Measuring social structure: a comparison of eight dominance indices," *Behavioural Processes*, vol. 73, no. 1, pp. 1–12, 2006.

[53] L. Conradt and T. J. Roper, "Consensus decision making in animals," *Trends in Ecology and Evolution*, vol. 20, no. 8, pp. 449–456, 2005.

[54] A. Okubo, "Dynamical aspects of animal grouping," *Advances in Biophysics*, vol. 22, pp. 1–94, 1986.

[55] C. W. Reynolds, "Flocks, herds and schools: a distributed behavioural model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–33, 1987.

[56] S. Gueron, S. A. Levin, and D. I. Rubenstein, "The dynamics of herds: from individuals to aggregations," *Journal of Theoretical Biology*, vol. 182, no. 1, pp. 85–98, 1996.

[57] A. Czirók and T. Vicsek, "Collective behavior of interacting self-propelled particles," *Physica A*, vol. 281, no. 1, pp. 17–29, 2000.

[58] M. Ballerini, "Interaction ruling collective animal behavior depends on topological rather than metric distance: evidence from a field study," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, pp. 1232–1237, 2008.

[59] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, Beckington, UK, 2008.

[60] A. H. Gandomi, X. S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, no. 1, pp. 17–35, 2013.

[61] H. Zang, S. Zhang, and K. Hapeshi, "A review of nature-inspired algorithms," *Journal of Bionic Engineering*, vol. 7, pp. S232–S237, 2010.

[62] A. Gandomi and A. Alavi, "Krill herd: a new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, pp. 4831–4845, 2012.

[63] J. E. Bresenham, "A linear algorithm for incremental digital sisplay of circular arcs," *Communications of the ACM*, vol. 20, no. 2, pp. 100–106, 1977.

[64] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[65] S. Garcia, 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," vol. 15, no. 6, pp. 617–644, 2009.

[66] J. Santamaría, O. Cordón, S. Damas, J. M. García-Torres, and A. Quirin, "Performance evaluation of memetic approaches in 3D reconstruction of forensic objects," *Soft Computing*, vol. 13, no. 8-9, pp. 883–904, 2009.