Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2012, Article ID 638275, 24 pages
doi:10.1155/2012/638275

Research Article

An Algorithm for Global Optimization Inspired by Collective Animal Behavior

Erik Cuevas, Mauricio González, Daniel Zaldivar, Marco Pérez-Cisneros, and Guillermo García

CUCEI Departamento de Electrónica, Universidad de Guadalajara, Avenida Revolución 1500, 44100 Guadalajara, JAL, Mexico

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

Received 21 September 2011; Revised 15 November 2011; Accepted 16 November 2011

Academic Editor: Carlo Piccardi

A metaheuristic algorithm for global optimization called the collective animal behavior (CAB) is introduced. 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 locations, 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, the searcher agents emulate a group of animals which interact with each other based on the biological laws of collective motion. The proposed method has been compared to other well-known optimization algorithms. The results show good performance of the proposed method when searching for a global optimum of several benchmark functions.

1. Introduction

Global optimization (GO) is a field with applications in many areas of science, engineering, economics, and others, where mathematical modelling is used [1]. In general, the goal is to find a global optimum of an objective function defined in a given search space. Global optimization algorithms are usually broadly divided into deterministic and metaheuristic [2]. Since deterministic methods only provide a theoretical guarantee of locating a local minimum of the objective function, they often face great difficulties in solving global optimization problems [3]. On the other hand, metaheuristic methods are usually faster in locating a global optimum than deterministic ones [4]. Moreover, metaheuristic methods adapt better to black-box formulations and extremely ill-behaved functions whereas deterministic methods

usually rest on at least some theoretical assumptions about the problem formulation and its analytical properties (such as Lipschitz continuity) [5].

Several metaheuristic algorithms have been developed by a combination of rules and randomness mimicking several phenomena. Such phenomena include evolutionary processes, for example, the evolutionary algorithm proposed by Fogel et al. [6], De Jong [7], and Koza [8], the genetic algorithm (GA) proposed by Holland [9] and Goldberg [10] and the artificial immune systems proposed by de Castro and Von Zuben [11]. On the other hand, physical processes consider the simulated annealing proposed by Kirkpatrick et al. [12], the electromagnetism-like algorithm proposed by İlker et al. [13], the gravitational search algorithm proposed by Rashedi et al. [14], and the musical process of searching for a perfect state of harmony, which has been proposed by Geem et al. [15], Lee and Geem [16], and Geem [17].

Many studies have been inspired by animal behavior phenomena for developing optimization techniques. For instance, the particle swarm optimization (PSO) algorithm which models the social behavior of bird flocking or fish schooling [18]. PSO consists of a swarm of particles which move towards best positions, seen so far, within a searchable space of possible solutions. Another behavior-inspired approach is the ant colony optimization (ACO) algorithm proposed by Dorigo et al. [19], which simulates the behavior of real ant colonies. Main features of the ACO algorithm are the distributed computation, the positive feedback, and the constructive greedy search. Recently, a new metaheuristic approach which is based on the animal behavior while hunting has been proposed in [20]. Such algorithm considers hunters as search positions and preys as potential solutions.

Just recently, the concept of individual-organization [21, 22] has been widely referenced to understand collective behavior of animals. The central principle of individual-organization is that simple repeating interactions between individuals can produce complex behavioral patterns at group level [21, 23, 24]. Such inspiration comes from behavioral patterns previously seen in several animal groups. Examples include ant pheromone trail networks, aggregation of cockroaches, and the migration of fish schools, all of which can be accurately described in terms of individuals following simple sets of rules [25]. Some examples of these rules [24, 26] are keeping the current position (or location) for best individuals, local attraction or repulsion, random movements, and competition for the space within a determined distance.

On the other hand, new studies [27–29] 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. According to such principle, it is possible to model complex collective behaviors by using simple individual rules and configuring a general memory.

In this paper, a new optimization algorithm inspired by the collective animal behavior is proposed. 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 mathematical operators. Such operations are applied to each agent considering that the complete group has a memory storing their own best positions seen so far, by using a competition principle. The proposed approach has been compared to other well-known optimization methods. The results confirm a high performance of the proposed method for solving various benchmark functions.

This paper is organized as follows. In Section 2, we introduce basic biological aspects of the algorithm. In Section 3, the novel CAB algorithm and its characteristics are both described. Section 4 presents the experimental results and the comparative study. Finally, in Section 5, conclusions are given.

2. Biologic Fundamentals

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 research, the relationship between individuals and group-level properties has just recently begun to be deciphered [30].

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 a relatively local sensing ability [31]. 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 [32], such as whether they are knowledgeable about a pertinent resource or about a threat.

Animal groups are based on a hierarchic structure [33] which considers different individuals according to a fitness principle called dominance [34] which is the domain of some individuals within a group that occurs when competition for resources leads to confrontation. Several studies [35, 36] have found that such animal behavior lead to more stable groups with better cohesion properties among individuals.

Recent studies have begun to elucidate how repeated interactions among grouping animals scale to collective behavior. They have remarkably revealed that collective decision-making mechanisms across a wide range of animal group types, from insects to birds (and even among humans in certain circumstances) seem to share similar functional characteristics [21, 25, 37]. Furthermore, at a certain level of description, collective decision-making by organisms shares essential common features such as a 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 [24].

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, one of them, known as the self-propelled particle (SPP) model, attempts to capture the collective behavior of animal groups in terms of interactions between group members which follow a diffusion process [38–41].

On the other hand, following a biological approach, Couzin and krauze [24, 25] have proposed a model in which individual animals follow simple rules of thumb: (1) keep the current position (or location) for best individuals, (2) move from or to nearby neighbors (local attraction or repulsion), (3) move randomly, and (4) compete for the space within of a determined distance. Each individual thus admits three different movements: attraction, repulsión, or random and holds two kinds of states: preserve the position or compete for a determined position. In the model, the movement, which is executed by each individual, is decided randomly (according to an internal motivation). On the other hand, the states follow a fixed criteria set.

The dynamical spatial structure of an animal group can be explained in terms of its history [36]. Despite such a fact, the majority of studies have failed in considering the existence of memory in behavioral models. However, recent research [27, 42] 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 which is exhibited in future stages. Such memory can contain the location 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 inspires the definition of novel evolutionary operators which outline the CAB algorithm. A memory is incorporated to store best animal positions (best solutions) considering a competition-dominance mechanism.

3. Collective Animal Behavior 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 locations at each generation ($\mathbf{M}_g$) and the other for storing the best historical positions during the complete evolutionary process ($\mathbf{M}_h$).

3.1. Description of the CAB Algorithm

Following other metaheuristic approaches, the CAB algorithm is an iterative process that starts by initializing the population randomly (generated random solutions or animal positions). Then, the following four operations are applied until a termination criterion is met (i.e., the iteration number NI).

(1) Keep the position of the best individuals.

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

(3) Move randomly.

(4) Compete for the space within a determined distance (update 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 parameter values to be optimized. Such values are randomly and uniformly distributed between the prespecified lower initial parameter bound a_j^{low} and the upper initial parameter bound a_j^{high},

$$a_{j,i} = a_j^{\text{low}} + \text{rand}(0,1) \cdot \left(a_j^{\text{high}} - a_j^{\text{low}}\right); \quad j = 1, 2, \ldots, D;\ i = 1, 2, \ldots, N_p, \tag{3.1}$$

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

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

$$\mathbf{a}_l = \mathbf{m}_h^l + \mathbf{v}, \tag{3.2}$$

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

3.1.3. Move from or to Nearby Neighbors

From the biological inspiration, animals experiment a random local attraction or repulsion according to an internal motivation. Therefore, we have implemented new evolutionary operators that mimic such biological pattern. 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 attracted/repelled considering the nearest best historical position within the group (i.e., the nearest position in $\mathbf{M}_h$); otherwise, it is attracted/repelled to/from the nearest best location within the group for the current generation (i.e., the nearest position in $\mathbf{M}_g$). Therefore such operation can be modeled as follows:

$$\mathbf{a}_i = \begin{cases} \mathbf{x}_i \pm r \cdot \left(\mathbf{m}_h^{\text{nearest}} - \mathbf{x}_i\right) & \text{with probability } H \\ \mathbf{x}_i \pm r \cdot \left(\mathbf{m}_g^{\text{nearest}} - \mathbf{x}_i\right) & \text{with probability } (1 - H), \end{cases} \tag{3.3}$$

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, one animal randomly changes its position. Such behavioral rule is implemented considering the next expression:

$$\mathbf{a}_i = \begin{cases} \mathbf{r} & \text{with probability } P \\ \mathbf{x}_i & \text{with probability } (1 - P), \end{cases} \tag{3.4}$$

with $i \in \{B+1, B+2, \ldots, N_p\}$and $\mathbf{r}$ a random vector defined in the search space. This operator is similar to reinitializing the particle in a random position, as it is done by (3.1).

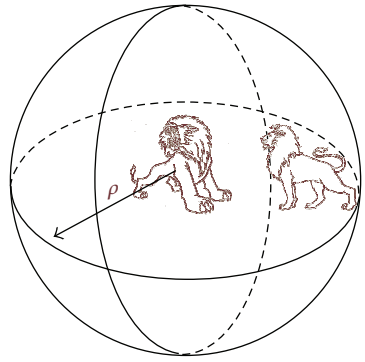

Figure 1: Dominance concept as it is presented when two animals confront each other inside of a ρ distance.

3.1.5. Compete for the Space within a Determined Distance (Update the Memory)

Once the operations to keep the position of the best individuals, such as moving from or to nearby neighbors and moving randomly, have been applied to all N_p animal positions, generating N_p new positions, it is necessary to update the memory $\mathbf{M}_h$.

In order to update memory $\mathbf{M}_h$, the concept of dominance is used. Animals that interact within the group maintain a minimum distance among them. Such distance, which is defined as ρ in the context of the CAB algorithm, depends on how aggressive the animal behaves [34, 42]. Hence, when two animals confront each other inside such distance, the most dominant individual prevails meanwhile other withdraw. Figure 1 depicts the process.

In the proposed algorithm, the historical 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 to the remaining 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 getting a better performance in the fitness function prevails meanwhile the other is removed.

(3) From the resulting elements of $\mathbf{M}_U$ (from Step 2), it is selected the B best value to build the new $\mathbf{M}_h$.

The use of the dominance principle in CAB allows considering as memory elements those solutions that hold the best fitness value within the region which has been defined by the ρ distance.

The procedure improves the exploration ability by incorporating information regarding previously found potential solutions during the algorithm's evolution. In general, the value of ρ depends on the size of the search space. A big value of ρ improves the exploration ability of the algorithm although it yields a lower convergence rate.

In order to calculate the ρ value, an empirical model has been developed after considering several conducted experiments. Such model is defined by following equation:

$$\rho = \frac{\prod_{j=1}^{D}\left(a_j^{\text{high}} - a_j^{\text{low}}\right)}{10 \cdot D}, \tag{3.5}$$

where a_j^{low} and a_j^{high} represent the prespecified lower and upper bound of the j-parameter respectively, within an 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 (3.1).

Step 3. Sort $\mathbf{A}$ 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 $\mathbf{A}(\{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_B}\})$.Such positions correspond to the elements of $\mathbf{M}_h$ making a slight random perturbation around them,

$$\mathbf{a}_l = \mathbf{m}_h^l + \mathbf{v}, \tag{3.6}$$

being $\mathbf{v}$ a random vector of a small enough 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 < P) then
   attraction and repulsion movement
      if (r_2 < H) then
      a_i = x_i ± r · (m_h^nearest − x_i)
      else if
      a_i = x_i ± r · (m_g^nearest − x_i)
   }
   else
   random movement
   {
      a_i = r
   }
end
```

where $r_1, r_2 \in r$ and $(0, 1)$ and $r \in [-1, 1]$

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

The best value in $\mathbf{M}_h$ represents the global solution for the optimization problem.

Table 1: Results of CAB with variant values of parameter P over 5 typical functions, with $H = 0.8$.

Function	n	$P = 0.5$, $\mu(\sigma^2)$	$P = 0.6$, $\mu(\sigma^2)$	$P = 0.7$, $\mu(\sigma^2)$	$P = 0.8$, $\mu(\sigma^2)$	$P = 0.9$, $\mu(\sigma^2)$
f_1	30	2.63×10^{-11} (2.13×10^{-12})	1.98×10^{-17} (6.51×10^{-18})	1.28×10^{-23} (3.54×10^{-24})	$\mathbf{2.33 \times 10^{-29}}$ ($\mathbf{4.41 \times 10^{-30}}$)	4.53×10^{-23} (5.12×10^{-24})
f_3	30	5.71×10^{-13} (1.11×10^{-14})	7.78×10^{-19} (1.52×10^{-20})	4.47×10^{-27} (3.6×10^{-28})	$\mathbf{7.62 \times 10^{-31}}$ ($\mathbf{4.23 \times 10^{-32}}$)	3.42×10^{-26} (3.54×10^{-27})
f_5	30	5.68×10^{-11} (2.21×10^{-12})	1.54×10^{-17} (1.68×10^{-18})	5.11×10^{-22} (4.42×10^{-23})	$\mathbf{9.02 \times 10^{-28}}$ ($\mathbf{6.77 \times 10^{-29}}$)	4.77×10^{-20} (1.94×10^{-21})
f_{10}	30	3.50×10^{-5} (3.22×10^{-6})	2.88×10^{-9} (3.28×10^{-10})	2.22×10^{-12} (4.21×10^{-13})	$\mathbf{8.88 \times 10^{-16}}$ ($\mathbf{3.49 \times 10^{-17}}$)	1.68×10^{-11} (5.31×10^{-12})
f_{11}	30	1.57×10^{-2} (1.25×10^{-3})	1.14×10^{-6} (3.71×10^{-7})	2.81×10^{-8} (5.21×10^{-9})	$\mathbf{4.21 \times 10^{-10}}$ ($\mathbf{4.87 \times 10^{-11}}$)	4.58×10^{-4} (6.92×10^{-5})

Table 2: Results of CAB with variant values of parameter H over 5 typical functions, with $P = 0.8$.

Function	n	$H = 0.5$, $\mu(\sigma^2)$	$H = 0.6$, $\mu(\sigma^2)$	$H = 0.7$, $\mu(\sigma^2)$	$H = 0.8$, $\mu(\sigma^2)$	$H = 0.9$, $\mu(\sigma^2)$
f_1	30	2.23×10^{-10} (8.92×10^{-11})	3.35×10^{-18} (3.21×10^{-19})	3.85×10^{-22} (6.78×10^{-23})	$\mathbf{2.33 \times 10^{-29}}$ ($\mathbf{4.41 \times 10^{-30}}$)	4.72×10^{-21} (6.29×10^{-22})
f_3	30	5.71×10^{-10} (5.12×10^{-11})	3.24×10^{-18} (1.32×10^{-19})	6.29×10^{-27} (8.26×10^{-23})	$\mathbf{7.62 \times 10^{-31}}$ ($\mathbf{4.23 \times 10^{-32}}$)	5.41×10^{-22} (5.28×10^{-23})
f_5	30	8.80×10^{-9} (5.55×10^{-10})	6.72×10^{-21} (1.11×10^{-22})	1.69×10^{-23} (1.34×10^{-24})	$\mathbf{9.02 \times 10^{-28}}$ ($\mathbf{6.77 \times 10^{-29}}$)	7.39×10^{-21} (4.41×10^{-22})
f_{10}	30	2.88×10^{-4} (3.11×10^{-5})	3.22×10^{-10} (2.18×10^{-12})	1.23×10^{-14} (4.65×10^{-15})	$\mathbf{8.88 \times 10^{-16}}$ ($\mathbf{3.49 \times 10^{-17}}$)	5.92×10^{-7} (3.17×10^{-9})
f_{11}	30	1.81×10^{-4} (2.16×10^{-5})	2.89×10^{-6} (6.43×10^{-7})	2.36×10^{-7} (3.75×10^{-4})	$\mathbf{4.21 \times 10^{-10}}$ ($\mathbf{4.87 \times 10^{-11}}$)	3.02×10^{-4} (4.37×10^{-6})

4. Experimental Results

4.1. Test Suite and Experimental Setup

A comprehensive set of 31 functions that have been collected from [43–54], they are used to test the performance of the proposed approach. Tables 12–17 in the appendix present the benchmark functions used in our experimental study. Such functions are classified into four different categories: unimodal test functions (Table 12), multimodal test functions (Table 13), multimodal test functions with fixed dimensions (Tables 14 and 15), and GKLS test functions (Tables 16 and 17). In such tables, n is the dimension of function, f_{opt} is the minimum value of the function, and S is a subset of R^n. The optimum location ($\mathbf{x}_{\text{opt}}$) for functions in Tables 12 and 13 fall into $[0]^n$, except for f_5, f_{12} and f_{13} with $\mathbf{x}_{\text{opt}}$ falling into $[1]^n$ and f_8 in $[420.96]^n$. A detailed description of all functions is given in the appendix.

To study the impact of parameters P and H (described in Sections 3.1.3 and 3.1.4) over the performance of CAB, different values have been tested on 5 typical functions. The maximum number of iterations is set to 1000. N_p and B are fixed to 50 and 10, respectively. The mean best function values (μ) and the standard deviations (σ^2) of CAB, averaged over 30 runs, for the different values of P and H are listed in Tables 1 and 2, respectively. The results suggest that a proper combination of different parameter values can improve the performance of CAB and the quality of solutions. Table 1 shows the results of an experiment which consist in fixing $H = 0.8$ and varying P from 0.5 to 0.9. On a second test, the experimental setup is swapped, that is, $P = 0.8$ and H varies from 0.5 to 0.9. The best results

in the experiments are highlighted in both tables. After the best value in parameters P and H has been experimentally determined (with a value of 0.8), it is kept for all tests throughout the paper.

In order to demonstrate that the CAB algorithm provides a better performance, it has been compared to other optimization approaches such as metaheuristic algorithms (Section 4.2) and continuous methods (Section 4.3). The results of such comparisons are explained in the following sections.

4.2. Performance Comparison with Other Metaheuristic Approaches

We have applied CAB to 31 test functions in order to compare its performance to other well-known metaheuristic algorithms such as the real genetic algorithm (RGA) [55], the PSO [18], the gravitational search algorithm (GSA) [56], and the differential evolution method (DE) [57]. In all cases, population size is set to 50. The maximum iteration number is 1000 for functions in Tables 12 and 13, and 500 for functions in Table 14 and 16. Such stop criteria have been chosen as to keep compatibility to similar works which are reported in [14] and [58].

Parameter settings for each algorithm in the comparison are described as follows.

(1) RGA: according to [55], the approach uses arithmetic crossover, Gaussian mutation, and roulette wheel selection. The crossover and mutation probabilities have been set to 0.3 and 0.1, respectively.

(2) PSO: In the algorithm, $c_1 = c_2 = 2$ while the inertia factor (ω) is decreasing linearly from 0.9 to 0.2.

(3) In GSA, G_0 is set to 100 and α is set to 20; T is the total number of iterations (set to 1000 for functions f_1–f_{13} and to 500 for functions f_{14}–f_{31}). Besides, K_0 is set to 50 (total number of agents) and is decreased linearly to 1. Such values have been found as the best configuration set according to [56].

(4) DE: the DE/Rand/1 scheme is employed. The parameter settings follow the instructions in [57]. The crossover probability is CR = 0.9 and the weighting factor is $F = 0.8$.

Several experimental tests have been developed for comparing the performance of the CAB algorithm against other metaheuristic algorithms. The experiments have been developed considering the following function types.

(1) Unimodal test functions (Table 12).

(2) Multimodal test functions (Table 13).

(3) Multimodal test functions with fixed dimensions (Tables 14 and 15).

(4) GKLS test functions (Tables 16 and 17).

4.2.1. Unimodal Test Functions

In this test, the performance of the CAB algorithm is compared to RGA, PSO, GSA and DE, considering functions with only one minimum/maximum. Such function type is represented by functions f_1 to f_7 in Table 12. The results, over 30 runs, are reported in Table 3 considering the following performance indexes: the average best-so-far solution, the average mean fitness

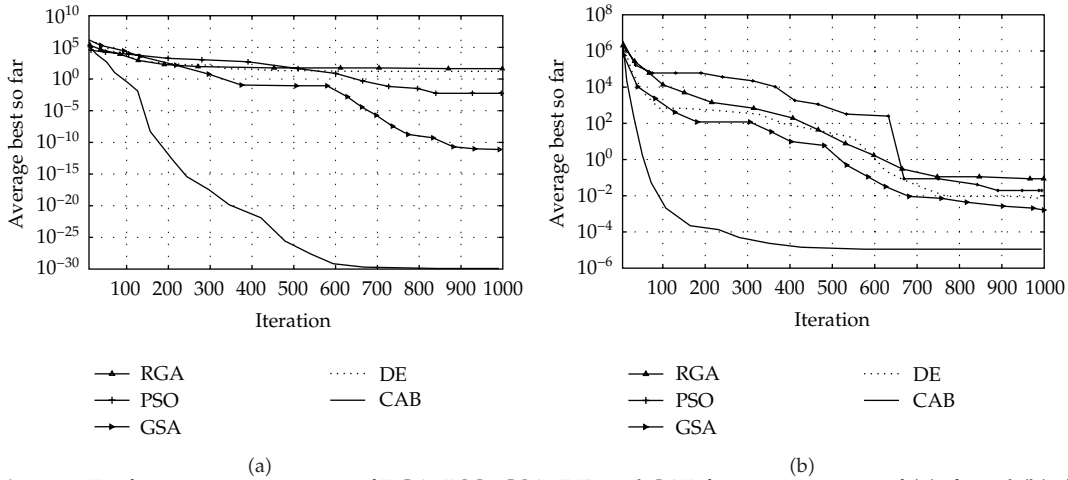

Figure 2: Performance comparison of RGA, PSO, GSA, DE, and CAB for minimization of (a) f_1 and (b) f_7 considering $n = 30$.

Table 3: Minimization result of benchmark functions in Table 12 with $n = 30$. Maximum number of iterations = 1000.

		RGA	PSO	GSA	DE	CAB
f_1	Average best sofar	23.13	1.8×10^{-3}	7.3×10^{-11}	11.21	$\mathbf{2.3 \times 10^{-29}}$
	Median best sofar	21.87	1.2×10^{-3}	7.1×10^{-11}	13.21	$\mathbf{1.1 \times 10^{-20}}$
	Average mean fitness	23.45	1.2×10^{-2}	2.1×10^{-10}	11.78	$\mathbf{1.2 \times 10^{-10}}$
f_2	Average best sofar	1.07	2.0	4.03×10^{-5}	0.95	$\mathbf{5.28 \times 10^{-20}}$
	Median best sofar	1.13	1.9×10^{-3}	4.07×10^{-5}	1.05	$\mathbf{2.88 \times 10^{-11}}$
	Average mean fitness	1.07	2.0	6.9×10^{-5}	0.90	$\mathbf{1.43 \times 10^{-9}}$
f_3	Average best sofar	5.6×10^{3}	4.1×10^{3}	0.16×10^{3}	0.12	$\mathbf{7.62 \times 10^{-31}}$
	Median best sofar	5.6×10^{3}	2.2×10^{3}	0.15×10^{3}	0.09	$\mathbf{1.28 \times 10^{-19}}$
	Average mean fitness	5.6×10^{3}	2.9×10^{3}	0.16×10^{3}	0.11	$\mathbf{3.51 \times 10^{-12}}$
f_4	Average best sofar	11.78	8.1	3.7×10^{-6}	0.012	$\mathbf{2.17 \times 10^{-17}}$
	Median best sofar	11.94	7.4	3.7×10^{-6}	0.058	$\mathbf{5.65 \times 10^{-12}}$
	Average mean fitness	11.78	23.6	8.5×10^{-6}	0.013	$\mathbf{4.96 \times 10^{-10}}$
f_5	Average best sofar	1.1×10^{3}	3.6×10^{4}	25.16	0.25	$\mathbf{9.025 \times 10^{-28}}$
	Median best sofar	1.0×10^{3}	1.7×10^{3}	25.18	0.31	$\mathbf{3.10 \times 10^{-18}}$
	Average mean fitness	1.1×10^{3}	3.7×10^{4}	25.16	0.24	$\mathbf{6.04 \times 10^{-14}}$
f_6	Average best sofar	24.01	1.0×10^{-3}	8.3×10^{-11}	1.25×10^{-3}	$\mathbf{4.47 \times 10^{-29}}$
	Median best sofar	24.55	6.6×10^{-3}	7.7×10^{-11}	3.33×10^{-3}	$\mathbf{4.26 \times 10^{-21}}$
	Average mean fitness	24.52	0.02	2.6×10^{-10}	1.27×10^{-3}	$\mathbf{1.03 \times 10^{-12}}$
f_7	Average best sofar	0.06	0.04	0.018	6.87×10^{-3}	$\mathbf{3.45 \times 10^{-5}}$
	Median best sofar	0.06	0.04	0.015	4.72×10^{-3}	$\mathbf{7.39 \times 10^{-4}}$
	Average mean fitness	0.56	1.04	0.533	1.28×10^{-2}	$\mathbf{8.75 \times 10^{-4}}$

function, and the median of the best solution in the last iteration. The best result for each function is boldfaced. According to this table, CAB provides better results than RGA, PSO, GSA, and DE for all functions. In particular, the results show considerable precision differences which are directly related to different local operators at each metaheuristic algorithm. Moreover, the good convergence rate of CAB can be observed from Figure 2. According to this figure, CAB tends to find the global optimum faster than other algorithms and yet offer the highest convergence rate.

Table 4: P values produced by Wilcoxon's test comparing CAB versus RGA, PSO, GSA, and DE over the "average best-so-far" values from Table 3.

CAB versus	RGA	PSO	GSA	DE
f_1	1.21×10^{-6}	3.94×10^{-5}	7.39×10^{-4}	1.04×10^{-6}
f_2	2.53×10^{-6}	5.62×10^{-5}	4.92×10^{-4}	2.21×10^{-6}
f_3	8.34×10^{-8}	6.42×10^{-8}	7.11×10^{-7}	1.02×10^{-4}
f_4	3.81×10^{-8}	1.91×10^{-8}	7.39×10^{-4}	1.27×10^{-6}
f_5	4.58×10^{-8}	9.77×10^{-9}	4.75×10^{-7}	0.23×10^{-4}
f_6	8.11×10^{-8}	1.98×10^{-6}	5.92×10^{-4}	2.88×10^{-5}
f_7	5.12×10^{-7}	4.77×10^{-7}	8.93×10^{-6}	1.01×10^{-4}

In order to statistically analyze the results in Table 3, a non-parametric significance proof known as the Wilcoxon's rank test has been conducted [59, 60], which allows assessing result differences among two related methods. The analysis is performed considering a 5% significance level over the "average best-so-far" data. Table 4 reports the P values produced by Wilcoxon's test for the pairwise comparison of the "average best so-far" of four groups. Such groups are formed by CAB versus RGA, CAB versus PSO, CAB versus GSA, and CAB versus DE. As a null hypothesis, it is assumed that there is no significant difference between mean values of the two algorithms. The alternative hypothesis considers a significant difference between the "average best-so-far" values of both approaches. All P values reported in the table are less than 0.05 (5% significance level) which is a strong evidence against the null hypothesis, indicating that the CAB results are statistically significant and that it has not occurred by coincidence (i.e., due to the normal noise contained in the process).

4.2.2. Multimodal Test Functions

Multimodal functions, in contrast to unimodal, have many local minima/maxima which are, in general, more difficult to optimize. In this section the performance of the CAB algorithm is compared to other metaheuristic algorithms considering multimodal functions. Such comparison reflects the algorithm's ability to escape from poor local optima and to locate a near-global optimum. We have done experiments on f_8 to f_{13} of Table 13 where the number of local minima increases exponentially as the dimension of the function increases. The dimension of these functions is set to 30. The results are averaged over 30 runs, reporting the performance indexes in Table 5 as follows: the average best-so-far solution, the average mean fitness function and, the median of the best solution in the last iteration (the best result for each function is highlighted). Likewise, P values of the Wilcoxon signed-rank test of 30 independent runs are listed in Table 6.

For f_9, f_{10}, f_{11}, and f_{12}, CAB yields a much better solution than the others. However, for functions f_8 and f_{13}, CAB produces similar results to RGA and GSA, respectively. The Wilcoxon rank test results, presented in Table 6, show that CAB performed better than RGA, PSO, GSA, and DE considering the four problems f_9–f_{12}, whereas, from a statistical viewpoint, there is not difference in results between CAB and RGA for f_8 and between CAB and GSA for f_{13}. Evolutions of the "average best-so-far" solutions over 30 runs for functions f_{10} and f_{12} are shown in Figure 3.

Table 5: Minimization of benchmark functions in Table 13 with $n = 30$. Maximum number of iterations = 1000.

		RGA	PSO	GSA	DE	CAB
	Average best sofar	$\mathbf{-1.26 \times 10^4}$	-9.8×10^3	-2.8×10^3	-4.1×10^3	$\mathbf{-1.2 \times 10^4}$
f_8	Median best sofar	$\mathbf{-1.26 \times 10^4}$	-9.8×10^3	-2.6×10^3	-4.1×10^3	$\mathbf{-1.2 \times 10^4}$
	Average mean fitness	$\mathbf{-1.26 \times 10^4}$	-9.8×10^3	-1.1×10^3	-4.1×10^3	$\mathbf{-1.2 \times 10^4}$
	Average best sofar	5.90	55.1	15.32	30.12	$\mathbf{1.0 \times 10^{-3}}$
f_9	Median best sofar	5.71	56.6	14.42	31.43	$\mathbf{7.6 \times 10^{-4}}$
	Average mean fitness	5.92	72.8	15.32	30.12	$\mathbf{1.0 \times 10^{-3}}$
	Average best sofar	2.13	9.0×10^{-3}	6.9×10^{-6}	3.1×10^{-3}	$\mathbf{8.88 \times 10^{-16}}$
f_{10}	Median best sofar	2.16	6.0×10^{-3}	6.9×10^{-6}	2.3×10^{-3}	$\mathbf{2.97 \times 10^{-11}}$
	Average mean fitness	2.15	0.02	1.1×10^{-5}	3.1×10^{-3}	$\mathbf{9.0 \times 10^{-10}}$
	Average best sofar	1.16	0.01	0.29	1.0×10^{-3}	$\mathbf{1.14 \times 10^{-13}}$
f_{11}	Median best sofar	1.14	0.0081	0.04	1.0×10^{-3}	$\mathbf{1.14 \times 10^{-13}}$
	Average mean fitness	1.16	0.055	0.29	1.0×10^{-3}	$\mathbf{1.14 \times 10^{-13}}$
	Average best sofar	0.051	0.29	0.01	0.12	$\mathbf{2.32 \times 10^{-30}}$
f_{12}	Median best sofar	0.039	0.11	4.2×10^{-13}	0.01	$\mathbf{5.22 \times 10^{-22}}$
	Average mean fitness	0.053	9.3×10^3	0.01	0.12	$\mathbf{4.63 \times 10^{-17}}$
	Average best sofar	0.081	3.1×10^{-18}	$\mathbf{3.2 \times 10^{-32}}$	1.77×10^{-25}	$\mathbf{1.35 \times 10^{-32}}$
f_{13}	Median best sofar	0.032	2.2×10^{-23}	$\mathbf{2.3 \times 10^{-32}}$	1.77×10^{-25}	$\mathbf{2.20 \times 10^{-21}}$
	Average mean fitness	0.081	4.8×10^5	$\mathbf{3.2 \times 10^{-32}}$	1.77×10^{-25}	$\mathbf{3.53 \times 10^{-17}}$

Table 6: P values produced by Wilcoxon's test comparing CAB versus RGA, PSO, GSA, and DE over the "average best-so-far" values from Table 5.

CAB versus	RGA	PSO	GSA	DE
f_8	0.89	8.38×10^{-4}	1.21×10^{-4}	4.61×10^{-4}
f_9	7.23×10^{-7}	1.92×10^{-9}	5.29×10^{-8}	9.97×10^{-8}
f_{10}	6.21×10^{-9}	4.21×10^{-5}	1.02×10^{-4}	3.34×10^{-4}
f_{11}	7.74×10^{-9}	3.68×10^{-7}	4.10×10^{-7}	8.12×10^{-5}
f_{12}	1.12×10^{-8}	8.80×10^{-9}	2.93×10^{-7}	4.02×10^{-8}
f_{13}	4.72×10^{-9}	3.92×10^{-5}	0.93	2.20×10^{-4}

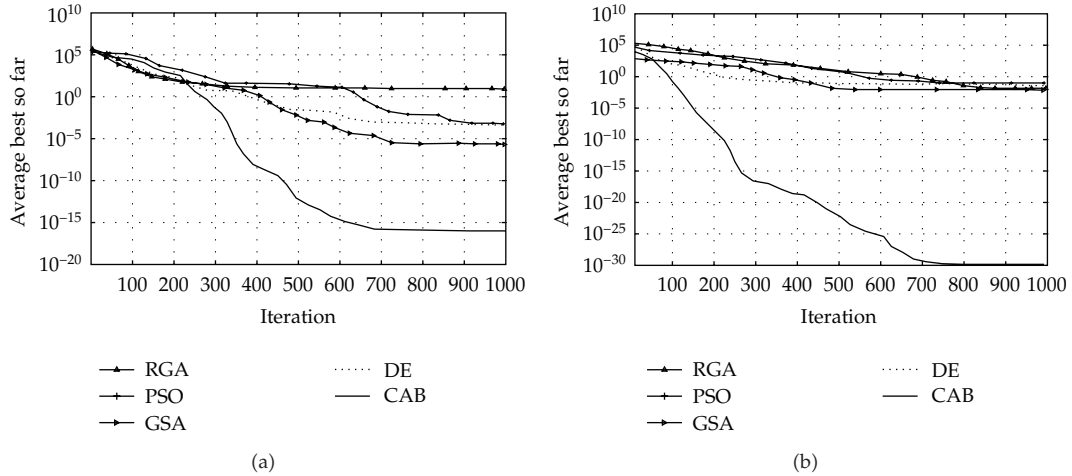

Figure 3: Performance comparison of RGA, PSO, GSA, DE, and CAB for minimization of (a) f_{10} and (b) f_{12} considering $n = 30$.

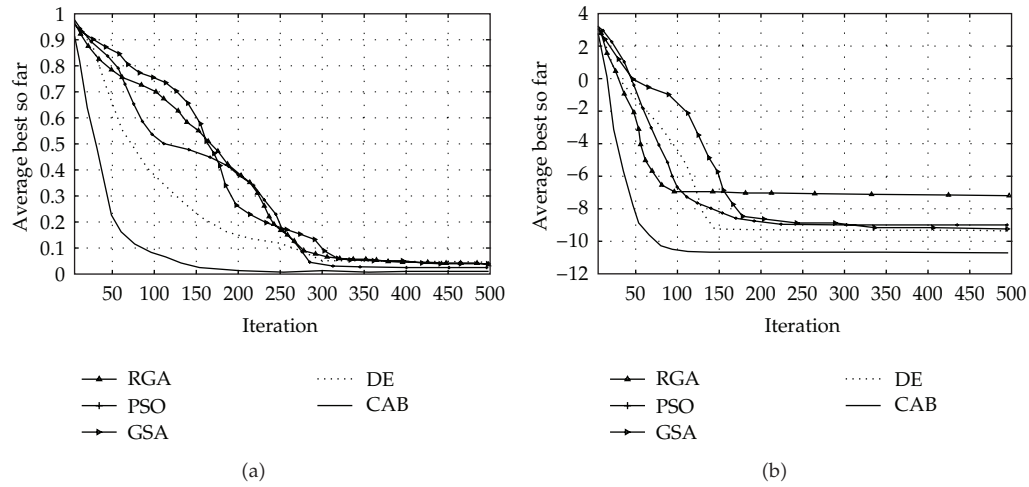

Figure 4: Performance comparison of RGA, PSO, GSA, DE, and CAB for minimization of (a) f_{15} and (b) f_{22}.

4.2.3. Multimodal Test Functions with Fixed Dimensions

In the following experiments the performance of the CAB algorithm is compared to RGA, PSO, GSA, and DE considering functions which are extensively reported in the metaheuristic-based optimization literature [49–54]. Such functions, represented by f_{14} to f_{23} in Tables 14 and 15, are all multimodal with fixed dimensions. Table 7 shows the outcome of such process. Results, presented in Table 7, show how metaheuristic algorithms maintain a similar average performance when they are applied to low-dimensional functions [58]. The results show that RGA, PSO, and GSA have similar solutions and performances that are nearly the same as it can be seen in Figure 4.

4.2.4. GKLS Test Functions

This section considers GKLS functions which are built using the GKLS-generator described in [54]. In the construction, the generator uses a set of user-defined parameters for building a multimodal function with known local and global minima. For conducting the numerical experiments, eight GKLS functions been employed which are defined by f_{24} to f_{31}. Details of their characteristics and parameters for their construction are listed in Tables 16 and 17. Results, over 30 runs, are reported in Table 8 (the best result for each function test is boldfaced). According to this table, CAB provides better results than RGA, PSO, GSA, and DE for all GKLS functions, in particular for functions holding bigger dimensions (f_{28}–f_{31}). Such performance is directly related to a better tradeoff between exploration and exploitation which is produced by CAB operators. Likewise, as it can be observed from Figure 5, the CAB algorithm possesses better convergence rates in comparison to other metaheuristic algorithms.

In order to statistically validate the results of Table 8, the Wilcoxon's test has been conducted. Table 9 shows the P values obtained after applying such analysis over 30

Table 7: Minimization result of benchmark functions in Table 14 with $n = 30$. Maximum number of iterations = 500.

		RGA	PSO	GSA	DE	CAB
f_{14}	Average best sofar	0.998	0.998	3.70	0.998	0.998
	Median best sofar	0.998	0.998	2.07	0.998	0.998
	Average mean fitness	0.998	0.998	9.17	0.998	0.998
f_{15}	Average best sofar	4.0×10^{-3}	2.8×10^{-3}	8.0×10^{-3}	2.2×10^{-3}	1.1×10^{-3}
	Median best sofar	1.7×10^{-3}	7.1×10^{-4}	7.4×10^{-4}	5.3×10^{-4}	2.2×10^{-4}
	Average mean fitness	4.0×10^{-3}	215.60	9.0×10^{-3}	2.2×10^{-3}	1.1×10^{-3}
f_{16}	Average best sofar	−1.0313	−1.0316	−1.0316	−1.0316	−1.0316
	Median best sofar	−1.0315	−1.0316	−1.0316	−1.0316	−1.0316
	Average mean fitness	−1.0313	−1.0316	−1.0316	−1.0316	−1.0316
f_{17}	Average best sofar	0.3996	0.3979	0.3979	0.3979	0.3979
	Median best sofar	0.3980	0.3979	0.3979	0.3979	0.3979
	Average mean fitness	0.3996	2.4112	0.3979	0.3979	0.3979
f_{18}	Average best sofar	−3.8627	−3.8628	−3.8628	−3.8628	−3.8628
	Median best sofar	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628
	Average mean fitness	−3.8627	−3.8628	−3.8628	−3.8628	−3.8628
f_{19}	Average best sofar	−3.3099	−3.3269	−3.7357	−3.3269	−3.8501
	Median best sofar	−3.3217	−3.2031	−3.8626	−3.3269	−3.8501
	Average mean fitness	−3.3098	−3.2369	−3.8020	−3.3269	−3.8501
f_{20}	Average best sofar	−3.3099	−3.2369	−2.0569	−3.2369	−3.2369
	Median best sofar	−3.3217	−3.2031	−1.9946	−3.2369	−3.2369
	Average mean fitness	−3.3098	−3.2369	−1.6014	−3.2369	−3.2369
f_{21}	Average best sofar	−5.6605	−6.6290	−6.0748	−6.6290	−10.1532
	Median best sofar	−2.6824	−5.1008	−5.0552	−6.0748	−10.1532
	Average mean fitness	−5.6605	−5.7496	−6.0748	−6.6290	−10.1532
f_{22}	Average best sofar	−7.3421	−9.1118	−9.3339	−9.3339	−10.4028
	Median best sofar	−10.3932	−10.402	−10.402	−9.3339	−10.4028
	Average mean fitness	−7.3421	−9.9305	−9.3399	−9.3339	−10.4028
f_{23}	Average best sofar	−6.2541	−9.7634	−9.4548	−9.7634	−10.5363
	Median best sofar	−4.5054	−10.536	−10.536	−9.7636	−10.5363
	Average mean fitness	−6.2541	−8.7626	−9.4548	−9.7634	−10.5363

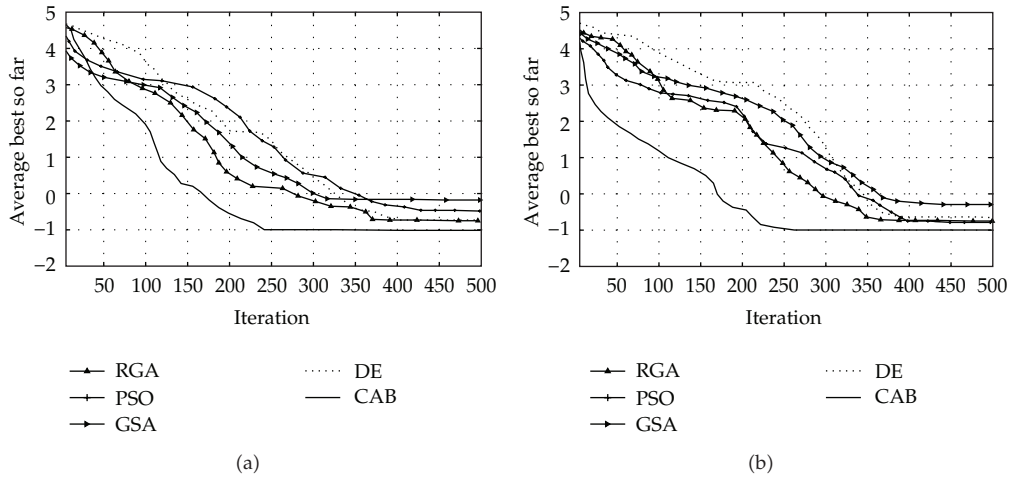

Figure 5: Performance comparison of RGA, PSO, GSA, DE and CAB for minimization of the GKLS-functions: (a) f_{28} and (b) f_{31}.

Table 8: Minimization result of GKLSfunctions in Table 16. Maximum number of iterations = 500.

		RGA	PSO	GSA	DE	CAB
	Average best sofar	−0.942004	−0.932011	−0.899812	−0.951937	**−1**
f_{24}	Median best sofar	−0.908521	−0.910058	−0.882597	−0.909844	**−0.999924**
	Average mean fitness	−0.907354	−0.909058	−0.882597	−0.903981	**−0.999865**
	Average best sofar	−0.931281	−0.941281	−0.813412	−0.968839	**−1**
f_{25}	Median best sofar	−0.900889	−0.899011	−0.803482	−0.909983	**−0.999961**
	Average mean fitness	−0.900115	−0.898545	−0.801143	−0.901101	**−0.999732**
	Average best sofar	−0.939845	−0.924521	−0.798799	−0.944561	**−1**
f_{26}	Median best sofar	−0.808034	−0.872132	−0.701174	−0.836621	**−0.999081**
	Average mean fitness	−0.801618	−0.864321	−0.698722	−0.816695	**−0.963632**
	Average best sofar	−0.948823	−0.939799	−0.778588	−0.948977	**−1**
f_{27}	Median best sofar	−0.818891	−0.798812	−0.668721	−0.812237	**−0.999552**
	Average mean fitness	−0.803487	−0.758892	−0.601179	−0.808721	**0.990978**
	Average best sofar	−0.888821	−0.858814	−0.618791	−0.871471	**−0.99907**
f_{28}	Median best sofar	−0.695712	−0.662715	−0.550711	−0.773419	**−0.889712**
	Average mean fitness	−0.599871	−0.500784	−0.443982	−0.612876	**−0.787712**
	Average best sofar	−0.872291	−0.880139	−0.642839	−0.885412	**−0.998681**
f_{29}	Median best sofar	−0.618732	−0.602568	−0.452974	−0.702591	**−0.857517**
	Average mean fitness	−0.552374	−0.459871	−0.400781	−0.610887	**−0.800181**
	Average best sofar	−0.798712	−0.779521	−0.607894	−0.807127	**−0.985712**
f_{30}	Median best sofar	−0.684521	−0.645828	−0.401896	−0.534519	**−0.882378**
	Average mean fitness	−0.551411	−0.497812	−0.400874	−0.458717	**−0.819784**
	Average best sofar	−0.788952	−0.792231	−0.613691	−0.798827	**−0.998712**
f_{31}	Median best sofar	−0.692354	−0.702387	−0.596711	−0.672895	**−0.842397**
	Average mean fitness	−0.601008	−0.652394	−0.482337	−0.604732	**−0.808897**

Table 9: P values produced by Wilcoxon's test comparing CAB versus RGA, PSO, GSA, and DE over the "average best-so-far" values from Table 8.

CAB versus	RGA	PSO	GSA	DE
f_{24}	0.0352	0.0312	0.0121	0.0389
f_{25}	0.0211	0.0237	0.0118	0.0311
f_{26}	0.0224	0.0238	0.0081	0.0301
f_{27}	0.0273	0.0231	0.0023	0.0308
f_{28}	0.0208	0.0198	0.0011	0.0210
f_{29}	0.0202	0.0219	0.0009	0.0258
f_{30}	0.0175	0.0165	0.0004	0.0221
f_{31}	0.0159	0.0166	0.0002	0.0208

independent executions. Since all P values, presented in Table 9, are less than 0.05, it indicates that the CAB results are statistically better.

4.3. Comparison to Continuous Optimization Methods

Finally, the CAB algorithm is also compared to continuous optimization methods by considering some functions of the appendix. Since the BFSG algorithm [61] is one of the most effective continuous methods for solving unconstrained optimization problems, it has been considered as a basis for the algorithms used in the comparison.

In order to compare the performance of CAB to continuous optimization approaches, two different tests have been conducted. The first one tests the ability of BFGS and CAB to face unimodal optimization tasks (see Section 4.3.1) is evaluated. The second experiment analyzes the performance of CAB and one BFGS-based approach, when they are both applied to multimodal functions (review Section 4.3.2).

4.3.1. Local Optimization

In the first experiment, the performance of algorithms BFGS and CAB over unimodal functions is compared. In unimodal functions, the global minimum matches the local minimum. Quasi-Newton methods, such as the BFGS, have a fast rate of local convergence although it depends on the problem's dimension [62, 63]. Considering that not all unimodal functions of Table 12 fulfill the requirements imposed by the gradient-based approaches (i.e., f_2 and f_4 are not differentiable meanwhile f_7 is nonsmooth), we have chosen the Rosenbrock function (f_5) as a benchmark.

In the test, both algorithms (BFGS and CAB) are employed to minimize f_5, considering different dimensions. For the BFGS implementation, $B_0 = I$ is considered as initial matrix. Likewise, parameters δ and σ are set to 0.1 and 0.9 respectively. Although several performance criteria may define a comparison index, most can be applied to only one method timely (such as the number of gradient evaluations). Therefore, this paper considers the elapsed time and the iteration number (once the minimum has been reached) as performance indexes in the comparison. In the case of BFGS, the termination condition is assumed as $\|g_5(\mathbf{x})\| \leq 1 \times 10^{-6}$, with $g_5(\mathbf{x})$ being the gradient of $f_5(\mathbf{x})$. On the other hand, the stopping criterion of CAB considers when no more changes to the best element in memory $\mathbf{M}_h$ are registered. Table 10 presents the results of both algorithms considering several dimensions

Table 10: Performance comparison between the BFGS and the CAB algorithm, considering different dimensions over the Rosenbrock function. The averaged elapsed time (AET) is referred in seconds.

f_5	AET		AIN	
n	BFGS	CAB	BFGS	CAB
2	0.15	4.21	6	89
10	0.55	5.28	22	98
30	1.35	5.44	41	108
50	2.77	5.88	68	112
70	4.23	6.11	93	115
100	5.55	6.22	105	121
120	6.64	6.71	125	129

Table 11: Performance comparison between the ADAPT and the CAB algorithm considering different multimodal functions. The averaged elapsed time (AET) is referred in the format M's (Minute'second).

Function	n	ADAPT				CAB		
		ALS	AET	AIN	ABS	AET	AIN	ABS
f_9	30	3705	45.4	23,327	1.2×10^{-2}	10.2	633	1.0×10^{-3}
f_{10}	30	4054	1'05.7	38,341	6.21×10^{-12}	12.1	723	8.88×10^{-16}
f_{11}	30	32,452	2'12.1	102,321	4.51×10^{-10}	15.8	884	1.14×10^{-13}
f_{17}	2	1532	33.2	20,202	0.3976	7.3	332	0.3979
f_{18}	2	1233	31.6	18,845	−3.8611	6.6	295	−3.8628

($n \in \{2, 10, 30, 50, 70, 100, 120\}$) of f_5. In order to assure consistency, such results represent the averaged elapsed time (AET) and the averaged iteration number (AIN) over 30 different executions. It is additionally considered that at each execution both methods are initialized in a random point (inside the search space).

From Table 10, we can observe that the BFGS algorithm produces shorter elapsed times and fewer iterations than the CAB method. However, from $n = 70$, the CAB algorithm contend with similar results. The fact that the BFGS algorithm outperforms the CAB approach cannot be deemed as a negative feature considering the restrictions imposed to the functions by the BFGS method.

4.3.2. Global Optimization

Since the BFGS algorithm exploits only local information, it may easily get trapped into local optima restricting its use for global optimization. Thus, several methods based on continuous optimization approaches have been proposed. One of the most widely used techniques is the so-called multistart [64] (MS). In MS a point is randomly chosen from a feasible region as initial solution and subsequently a continuous optimization algorithm (local search) starts from it. Then, the process is repeated until a near global optimum is reached. The weakness of MS is that the same local minima may be found over and over again, wasting computational resources [65].

In order to compare the performance of the CAB approach to continuous optimization methods in the context of global optimization, the MS algorithm ADAPT [66] has been chosen. ADAPT uses as local search method the BFGS algorithm, which is iteratively executed. Thus, ADAPT possess two different stop criteria, one for the local procedure BFGS

Table 12: Unimodal test functions.

Test function	S	f_{opt}
$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$	0
$f_2(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$	0
$f_3(\mathbf{x}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^n$	0
$f_4(\mathbf{x}) = \max_i \{\lvert x_i \rvert, 1 \leq i \leq n\}$	$[-100, 100]^n$	0
$f_5(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^n$	0
$f_6(\mathbf{x}) = \sum_{i=1}^{n} (x_i + 0.5)^2$	$[-100, 100]^n$	0
$f_7(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + \text{rand}(0, 1)$	$[-1.28, 1.28]^n$	0

Table 13: Multimodal test functions.

Test function	S	f_{opt}
$f_8(\mathbf{x}) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]^n$	$-418.98^* n$
$f_9(\mathbf{x}) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^n$	0
$f_{10}(\mathbf{x}) = -20\exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}) - \exp((1/n)\sum_{i=1}^{n} \cos(2\pi x_i)) + 20$	$[-32, 32]^n$	0
$f_{11}(\mathbf{x}) = (1/4000)\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$	$[-600, 600]^n$	0
$f_{12}(\mathbf{x}) = (\pi/n)\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$		
$y_i = 1 + (x_i + 1)/4$	$[-50, 50]^n$	0
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$		
$f_{13}(\mathbf{x}) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^n$	0

and other for the complete MS approach. For the comparison, the ADAPT algorithm has been implemented as suggested in [66].

In the second experiment, the performance of the ADAPT and the CAB algorithms is compared over several multimodal functions described in Tables 13 and 14. The study considers the following performance indexes: the elapsed time, the iteration number, and the average best so-far solution. In case of the ADAPT algorithm, the iteration number is computed as the total iteration number produced by all the local search procedures as the MS method operates. The termination condition of the ADAPT local search algorithm (BFGS) is assumed when $\|g_k(\mathbf{x})\| \leq 1 \times 10^{-5}$, $g_k(\mathbf{x})$ being the gradient of $f_k(\mathbf{x})$. On the other hand, the stopping criterion for the CAB and the ADAPT algorithms is considered when no more changes in the best solution are registered. Table 11 presents results from both algorithms considering several multimodal functions. In order to assure consistency, results ponder

Table 14: Multimodal test functions with fixed dimensions.

Test function	S	f_{opt}
$f_{14}(\mathbf{x}) = \left(1/500 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$, $a_{ij} = \begin{pmatrix} -32, -16, 0, 16, 32, -32, \ldots, 0, 16, 32 \\ -32, -32, -32, -32, 16, \ldots, 32, 32, 32 \end{pmatrix}$	$[-65.5, 65.5]^2$	1
$f_{15}(\mathbf{x}) = \sum_{i=1}^{11}\left[a_i - \frac{x_i(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$ $\mathbf{a} = [0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, 0.0342, 0.0342, 0.0235, 0.0246]$ $\mathbf{b} = [0.25, 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16]$	$[-5, 5]^4$	0.00030
$f_{16}(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]^2$	−1.0316
$f_{17}(x_1, x_2) = (x_2 - (5.1/4\pi^2)\,)x_1^2 + (5/\pi)x_1 - 6)^2 + 10(1 - (1/8\pi))\cos x_1 + 10$	$x_1 \in [-5, 10]$ $x_2 \in [0, 15]$	0.398
$f_{18}(x_1, x_2) = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2]$	$[-5, 5]^2$	3
$f_{19}(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$ $\mathbf{a} = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 30 \end{bmatrix}$, $\mathbf{c} = [1, 1.2, 3, 3.2]$, $\mathbf{P} = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.0381 & 0.5743 & 0.8828 \end{bmatrix}$	$[0, 1]^3$	−3.86
$f_{20}(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$ $\mathbf{a} = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$, $\mathbf{c} = [1, 1.2, 3, 3.2]$, $\mathbf{P} = \begin{bmatrix} 0.131 & 0.169 & 0.556 & 0.012 & 0.828 & 0.588 \\ 0.232 & 0.413 & 0.830 & 0.373 & 0.100 & 0.999 \\ 0.234 & 0.141 & 0.352 & 0.288 & 0.304 & 0.665 \\ 0.404 & 0.882 & 0.873 & 0.574 & 0.109 & 0.038 \end{bmatrix}$	$[0, 1]^6$	−3.32

Table 14: Continued.

Test function	S	f_{opt}
$f_{21}(\mathbf{x}) = -\sum_{i=1}^{5}[(\mathbf{x}-a_i)(\mathbf{x}-a_i)^T + c_i]^{-1}$, $\mathbf{a} = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix}$, $\mathbf{c} = [0.1, 0.2, 0.2, 0.4, 0.4, 0.6, 0.3, 0.7, 0.5, 0.5]$	$[0,10]^4$	−10.1532
$f_{22}(\mathbf{x}) = -\sum_{i=1}^{7}[(\mathbf{x}-a_i)(\mathbf{x}-a_i)^T + c_i]^{-1}$, $\mathbf{a}$ and $\mathbf{c}$, equal to f_{21}	$[0,10]^4$	−10.4028
$f_{23}(\mathbf{x}) = -\sum_{i=1}^{7}[(\mathbf{x}-a_i)(\mathbf{x}-a_i)^T + c_i]^{-1}$, $\mathbf{a}$ and $\mathbf{c}$, equal to f_{21}	$[0,10]^4$	−10.5363

Table 15: Optimum locations of Table 14.

Test function	$\mathbf{x}_{\text{opt}}$	Test function	$\mathbf{x}_{\text{opt}}$
f_{14}	$(-32, 32)$	f_{19}	$(0.114, 0.556, 0.852)$
f_{15}	$(0.1928, 0.1908, 0.1231, 0.1358)$	f_{20}	$(0.201, 0.15, 0.477, 0.275, 0.311, 0.657)$
f_{16}	$(0.089, -0.71)$, $(-0.0089, 0.712)$	f_{21}	5 local minima in $a_{ij}, i = 1, \ldots, 5$
f_{17}	$(-3.14, 12.27)$, $(3.14, 2.275)$, $(9.42, 2.42)$	f_{22}	7 local minima in $a_{ij}, i = 1, \ldots, 7$
f_{18}	$(0, -1)$	f_{23}	10 local minima in $a_{ij}, i = 1, \ldots, 10$

Table 16: Used GKLSfunctions.

Test function	n (N)	r^*	ρ^*	Function class	Function number	M	S	f_{opt}
f_{24}	2	0.9	0.1	D	87	10	$[-1, 1]^2$	-1
f_{25}	2	0.7	0.3	ND	20	10	$[-1, 1]^2$	-1
f_{26}	3	0.9	0.1	D	87	10	$[-1, 1]^3$	-1
f_{27}	3	0.7	0.3	ND	20	10	$[-1, 1]^3$	-1
f_{28}	6	0.9	0.1	D	87	10	$[-1, 1]^6$	-1
f_{29}	6	0.7	0.3	ND	20	10	$[-1, 1]^6$	-1
f_{30}	10	0.9	0.1	D	87	10	$[-1, 1]^{10}$	-1
f_{31}	10	0.7	0.3	ND	20	10	$[-1, 1]^{10}$	-1

Table 17: Optimum locations of the used GKLS functions.

Test function	$\mathbf{x}_{\text{opt}}$
f_{24}	$(-0.767, -0.076)$
f_{25}	$(0.172, 0.174)$
f_{26}	$(-0.262, 0.253, -0.161)$
f_{27}	$(-0.168, -0.859, -0.727)$
f_{28}	$(-0.361, -0.562, -0.650, 0.857, -0.070, 0.906)$
f_{29}	$(-0.286, 0.227, -0.692, -0.388, -0.299, 0.732)$
f_{30}	$(0.392, -0.139, -0.667, 0.899, 0.654, -0.609, -0.087, -0.700, -0.287, 0.893)$
f_{31}	$(0.723, 0.413, 0.473, -0.746, 0.054, -0.412, -0.332, -0.677, 0.996, 0.239)$

the averaged elapsed time (AET), the averaged iteration number (AIN) and the average best-so-far solution (ABS) over 30 different executions. In Table 11, the averaged number of local searches (ALS) executed by ADAPT during the optimization is additionally considered.

Table 11 provides a summarized performance comparison between the ADAPT and the CAB algorithms. although both algorithms are able to acceptably locate the global minimum for both cases, there exist significant differences in the required time for reaching it. When comparing the averaged elapsed time (AET) and the averaged iteration number (AIN) in Table 11, CAB uses significantly less time and fewer iterations to reach the global minimum than the ADAPT algorithm.

5. Conclusions

This article proposes a novel metaheuristic optimization algorithm that is called the collective animal behavior algorithm (CAB). In CAB, the searcher agents emulates a group of animals that interact with each other considering simple behavioral rules which are modeled as mathematical operators. Such operations are applied to each agent considering that the complete group has a memory storing the best positions seen so far by using a competition principle.

The CAB algorithm presents two important characteristics: (1) CAB operators allow a better tradeoff between exploration and exploitation of the search space; (2) the use of its embedded memory incorporates information regarding previously found local minima (potential solutions) during the evolution process.

CAB has been experimentally tested considering a challenging test suite gathering 31 benchmark functions. In order to analyze the performance of the CAB algorithm, it has been compared to other well-known metaheuristic approaches. The experiments, statistically validated, have demonstrated that CAB generally outperforms other metaheuristic algorithms for most of the benchmark functions regarding the solution quality. In this study, the CAB algorithm has also been compared to algorithms based on continuous optimization methods. The results have shown that althogh continuous-based approaches outperform CAB for local optimization tasks, they face great difficulties in solving global optimization problems.

Appendix

List of Benchmark Functions

For more details see Tables 12, 13, 14, 15, 16, and 17.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[15] 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.

[16] 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.
[17] 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.
[18] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.
[19] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Tech. Rep. 91-016, Politecnico di Milano, 1991.
[20] R. Oftadeh, M. J. Mahjoob, and M. Shariatpanahi, "A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search," *Computers and Mathematics with Applications*, vol. 60, no. 7, pp. 2087–2098, 2010.
[21] D. J. T. Sumpter, "The principles of collective animal behaviour," *Philosophical Transactions of the Royal Society B*, vol. 361, no. 1465, pp. 5–22, 2006.
[22] O. Petit and R. Bon, "Decision-making processes: the case of collective movements," *Behavioural Processes*, vol. 84, no. 3, pp. 635–647, 2010.
[23] 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.
[24] I. D. Couzin, "Collective cognition in animal groups," *Trends in Cognitive Sciences*, vol. 13, no. 1, pp. 36–43, 2009.
[25] 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.
[26] 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.
[27] 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.
[28] I. Couzin, "Collective minds," *Nature*, vol. 445, no. 7129, p. 715, 2007.
[29] 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.
[30] 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.
[31] 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.
[32] M. Bourjade, B. Thierry, M. Maumy, and O. Petit, "Decision-making processes in the collective movements of Przewalski horses families Equus ferus Przewalskii: influences of the environment," *Ethology*, vol. 115, pp. 321–330, 2009.
[33] 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.
[34] Y. Hsu, R. L. Earley, and L. L. Wolf, "Modulation of aggressive behaviour by fighting experience: mechanisms and contest outcomes," *Biological Reviews*, vol. 81, no. 1, pp. 33–74, 2006.
[35] 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.
[36] 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.
[37] L. Conradt and T. J. Roper, "Consensus decision making in animals," *Trends in Ecology and Evolution*, vol. 20, no. 8, pp. 449–456, 2005.
[38] A. Okubo, "Dynamical aspects of animal grouping: swarms, schools, flocks, and herds," *Advances in Biophysics*, vol. 22, pp. 1–94, 1986.
[39] C. W. Reynolds, "Flocks, herds and schools: a distributed behavioural model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.
[40] 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.
[41] A. Czirók and T. Vicsek, "Collective behavior of interacting self-propelled particles," *Physica A*, vol. 281, no. 1, pp. 17–29, 2000.

[42] M. Ballerini, N. Cabibbo, R. Candelier et al., "Interaction ruling animal collective 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, no. 4, pp. 1232–1237, 2008.

[43] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems," *Journal of Global Optimization*, vol. 31, no. 4, pp. 635–672, 2005.

[44] R. Chelouah and P. Siarry, "Continuous genetic algorithm designed for the global optimization of multimodal functions," *Journal of Heuristics*, vol. 6, no. 2, pp. 191–213, 2000.

[45] F. Herrera, M. Lozano, and A. M. Sánchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study," *International Journal of Intelligent Systems*, vol. 18, no. 3, pp. 309–338, 2003.

[46] M. Laguna and R. Martí, "Experimental testing of advanced scatter search designs for global optimization of multimodal functions," *Journal of Global Optimization*, vol. 33, no. 2, pp. 235–255, 2005.

[47] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-coded memetic algorithms with crossover hill-climbing," *Evolutionary Computation*, vol. 12, no. 3, pp. 273–302, 2004.

[48] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "Testing unconstrained optimization software," *Association for Computing Machinery. Transactions on Mathematical Software*, vol. 7, no. 1, pp. 17–41, 1981.

[49] D. Ortiz-Boyer, C. Hervás-Martínez, and N. García-Pedrajas, "CIXL2: a crossover operator for evolutionary algorithms based on population features," *Journal of Artificial Intelligence Research*, vol. 24, pp. 1–48, 2005.

[50] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, New York, NY, USA, 2005.

[51] R. S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

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

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

[54] M. Gaviano, D. E. Kvasov, D. Lera, and Y. D. Sergeyev, "Software for generation of classes of test functions with known local and global minima for global optimization," *Association for Computing Machinery. Transactions on Mathematical Software*, vol. 29, no. 4, pp. 469–480, 2003.

[55] C. Hamzaçebi, "Improving genetic algorithms' performance by local search for continuous function optimization," *Applied Mathematics and Computation*, vol. 196, no. 1, pp. 309–317, 2008.

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

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

[58] D. Shilane, J. Martikainen, S. Dudoit, and S. J. Ovaska, "A general framework for statistical performance comparison of evolutionary computation algorithms," *Information Sciences*, vol. 178, no. 14, pp. 2870–2879, 2008.

[59] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, pp. 80–83, 1945.

[60] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

[61] M. Al-Baali, "On the behaviour of a combined extra-updating/self-scaling BFGS method," *Journal of Computational and Applied Mathematics*, vol. 134, no. 1-2, pp. 269–281, 2001.

[62] M. J. D. Powell, "How bad are the BFGS and DFP methods when the objective function is quadratic?" *Mathematical Programming*, vol. 34, no. 1, pp. 34–47, 1986.

[63] E. Hansen and G. W. Walster, *Global Optimization Using Interval Analysis*, CRC Press, 2004.

[64] L. Lasdon and J. C. Plummer, "Multistart algorithms for seeking feasibility," *Computers & Operations Research*, vol. 35, no. 5, pp. 1379–1393, 2008.

[65] F. V. Theos, I. E. Lagaris, and D. G. Papageorgiou, "PANMIN: sequential and parallel global optimization procedures with a variety of options for the local search strategy," *Computer Physics Communications*, vol. 159, no. 1, pp. 63–69, 2004.

[66] C. Voglis and I. E. Lagaris, "Towards "ideal multistart". A stochastic approach for locating the minima of a continuous function inside a bounded domain," *Applied Mathematics and Computation*, vol. 213, no. 1, pp. 216–229, 2009.