Chapter 4
The Metaheuristic Algorithm of the Locust-Search

Metaheuristic is a set of soft computing techniques which considers the design of intelligent search algorithms based on the analysis of several natural and social phenomena. Many metaheuristic methods have been suggested to solve a wide range of complex optimization applications. Even though these schemes have been designed to satisfy the requirements of general optimization problems, no single method can solve all problems adequately. Consequently, an enormous amount of research has been dedicated to producing new optimization methods that attain better performance indexes. In this chapter, metaheuristic algorithm called Locust Search (LS) is presented for solving optimization tasks. The LS method considers the simulation of the behavior presented in swarms of locusts as a metaphor. In the algorithm, individuals imitate a group of locusts which operate according to the biological laws of the swarm. The algorithm defines two distinct behaviors: solitary and social. Depending on the behavior, each element is undergone to a set of evolutionary operators that emulate the distinct collective behaviors typically present in the swarm.

4.1 Introduction

The collective intelligent behavior of insect or animal groups in nature such as flocks of birds, colonies of ants, schools of fish, swarms of bees and termites have attracted the attention of researchers. The aggregative conduct of insects or animals is known as swarm behavior. Even though the single members of swarms are non-sophisticated individuals, they are able to achieve complex tasks in cooperation. The collective swarm behavior emerges from relatively simple actions or interactions among the members. Entomologists have studied this collective phenomenon to model biological swarms while engineers have applied these models as a framework for solving complex real-world problems. The discipline of artificial intelligence which is concerned with the design of intelligent multi-agent

E. Cuevas et al., *Advances in Metaheuristics Algorithms: Methods and Applications*, Studies in Computational Intelligence 775,
https://doi.org/10.1007/978-3-319-89309-9_4

algorithms by taking inspiration from the collective behavior of social insects or animals is known as swarm intelligence [1]. Swarm algorithms have several advantages such as scalability, fault tolerance, adaptation, speed, modularity, autonomy and parallelism [2].

Several swarm algorithms have been developed by a combination of deterministic rules and randomness, mimicking the behavior of insect or animal groups in nature. Such methods include the social behavior of bird flocking and fish schooling such as the Particle Swarm Optimization (PSO) algorithm [3], the cooperative behavior of bee colonies such as the Artificial Bee Colony (ABC) technique [4], the social foraging behavior of bacteria such as the Bacterial Foraging Optimization Algorithm (BFOA) [5], the simulation of the herding behavior of krill individuals such as the Krill Herd (KH) method [6], the mating behavior of firefly insects such as the Firefly (FF) method [7] the emulation of the lifestyle of cuckoo birds such as the Cuckoo Search (CS) [8], the social-spider behavior such as the Social Spider Optimization (SSO) [9], the simulation of the animal behavior in a group such as the Collective Animal Behavior [10] and the emulation of the differential evolution in species such as the Differential Evolution (DE) [11].

In particular, insect swarms and animal groups provide a rich set of metaphors for designing swarm optimization algorithms. Such methods are complex systems composed by individuals that tend to reproduce specialized behaviors [12]. However, most of swarm algorithms and other evolutionary algorithms tend to exclusively concentrate the individuals in the current best positions. Under such circumstances, these algorithms seriously limit their search capacities.

Although PSO and DE are the most popular algorithms for solving complex optimization problems, they present serious flaws such as premature convergence and difficulty to overcome local minima [13, 14]. The cause for such problems is associated to the operators that modify individual positions. In such algorithms, during their evolution, the position of each agent for the next iteration is updated yielding an attraction towards the position of the best particle seen so-far (in case of PSO) or towards other promising individuals (in case of DE). As the algorithm evolves, these behaviors cause that the entire population rapidly concentrates around the best particles, favoring the premature convergence and damaging the appropriate exploration of the search space [15, 16].

The interesting and exotic collective behavior of insects have fascinated and attracted researchers for many years. The intelligent behavior observed in these groups provides survival advantages, where insect aggregations of relatively simple and "unintelligent" individuals can accomplish very complex tasks using only limited local information and simple rules of behavior [17]. Locusts (Schistocerca gregaria) are a representative example of such collaborative insects [18]. Locust is a kind of grasshopper that can change reversibly between a solitary and a social phase, which differ considerably in behavior [19]. The two phases show many differences including both overall levels of activity and the degree to which locusts are attracted or repulsed among them [20]. In the solitary phase, locusts avoid contact each other (locust concentrations). As consequence, they distribute

throughout the space, exploring sufficiently the plantation [20]. On other hand, in the social phase, locusts frantically concentrate around the elements that have already found good food sources [21]. Under such a behavior, locust attempt to efficiently find better nutrients by devastating promising areas within the plantation.

In this chapter, the Locust Search (LS) is analyzed for solving optimization tasks. The LS algorithm is based on the simulation of the behavior presented in swarms of locusts. In LS, individuals emulate a group of locusts which interact to each other based on the biological laws of the cooperative swarm. The algorithm considers two different behaviors: solitary and social. Depending on the behavior, each individual is conducted by a set of evolutionary operators which mimic the different cooperative behaviors that are typically found in the swarm. Different to most of existent swarm algorithms, in LS approach, the modeled behavior explicitly avoids the concentration of individuals in the current best positions. Such fact allows not only to emulate in a better realistic way the cooperative behavior of the locust colony, but also to incorporate a computational mechanism to avoid critical flaws commonly present in the popular PSO and DE algorithms, such as the premature convergence and the incorrect exploration–exploitation balance. In order to illustrate the proficiency and robustness of the LS approach, it is compared to other well-known evolutionary methods. The comparison examines several standard benchmark functions which are commonly considered in the literature. The results show a high performance of the LS method for searching a global optimum in several benchmark functions.

This chapter is organized as follows. In Sect. 4.2, we introduce basic biological aspects and models of the algorithm. In Sect. 4.3, the novel LS algorithm and its characteristics are both described. Section 4.4 presents the experimental results and the comparative study. Finally, in Sect. 4.5, conclusions are drawn.

4.2 Biological Fundamentals

Social insect societies are complex cooperative systems that self-organize within a set of constraints. Cooperative groups are better at manipulating and exploiting their environment, defending resources and brood, and allowing task specialization among group members [22, 23]. A social insect colony functions as an integrated unit that not only possesses the ability to operate at a distributed manner, but also to undertake enormous construction of global projects [24]. It is important to acknowledge that global order in insects can arise as a result of internal interactions among members.

Locusts are a kind of grasshoppers that exhibit two opposite behavioral phases, solitary and social (gregarious). Individuals in the solitary phase avoid contact each other (locust concentrations). As consequence, they distribute throughout the space, exploring sufficiently the plantation [20]. In contrast, locusts in the gregarious phase form several concentrations. These concentrations may contain up to 10^{10} members, cover cross-sectional areas of up to 10 km^2, and travel up to 10 km per day for a

period of days or weeks as they feed causing devastating crop loss [25]. The mechanism for the switch from the solitary phase to the gregarious phase is complex, and has been a subject of significant biological inquiry. A set of factors recently has been implicated, including geometry of the vegetation landscape and the olfactory stimulus [26].

Only few works [20, 21] that mathematically model the locust behavior have been published. In such approaches, it is developed two different minimal models with the goal of reproducing the macroscopic structure and motion of a group of locusts. Since the method proposed in [20] models the behavior of each locust in the group, it is used to explain the algorithm LS in this chapter.

4.2.1 Solitary Phase

In this section, it is described the way in which the position of each locust is modified as a consequence of its behavior under the solitary phase. Considering that $\mathbf{x}_i^k$ represents the current position of the ith locust in a group of N different elements, the new position $\mathbf{x}_i^{k+1}$ is calculated by using the following model:

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \Delta\mathbf{x}_i, \tag{4.1}$$

where $\Delta\mathbf{x}_i$ corresponds to the change of position experimented by $\mathbf{x}_i^k$ as a consequence of its social interaction with all the other elements in the group.

Two locusts in the solitary phase exert forces on each other according to basic biological principles of attraction and repulsion (see, e.g., [20]). Repulsion operates very strongly over a short length scale in order to avoid concentrations. Attraction is weaker, and operates over a longer length scale, providing the social force necessary for maintaining the cohesion in the group. Therefore, it is modeled the strength of these social forces using the function:

$$s(r) = F \cdot e^{-r/L} - e^{-r} \tag{4.2}$$

Here, r is a distance, F describes the strength of attraction, and L is the typical attractive length scale. We have scaled the time and space coordinates so that the repulsive strength and length scale are unity. We assume that $F < 1$ and $L > 1$ so that repulsion is stronger and shorter-scale, and attraction in weaker and longer-scale. This is typical for social organisms [21]. The social force exerted by locust j on locust i is:

$$\mathbf{s}_{ij} = s(r_{ij}) \cdot \mathbf{d}_{ij}, \tag{4.3}$$

where $r_{ij} = |\mathbf{x}_j - \mathbf{x}_i|$ is the distance between the two locusts and $\mathbf{d}_{ij} = (\mathbf{x}_j - \mathbf{x}_i)/r_{ij}$ is the unit vector pointing from $\mathbf{x}_i$ to $\mathbf{x}_j$. The total social force on each locust can be modeled as the superposition of all of the pairwise interactions:

$$\mathbf{S}_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \mathbf{s}_{ij}, \tag{4.4}$$

The change of position $\Delta\mathbf{x}_i$ is modeled as the total social force experimented by $\mathbf{x}_i^k$ as the superposition of all of the pairwise interactions. Therefore, $\Delta\mathbf{x}_i$ is defined as follows:

$$\Delta\mathbf{x}_i = \mathbf{S}_i, \tag{4.5}$$

In order to illustrate the behavioral model under the solitary phase, Fig. 4.1 presents an example. It is assumed a population of three different members ($N = 3$) which adopt a determined configuration in the current iteration k. As a consequence of the social forces, each element suffers an attraction or repulsion to other elements depending on the distance among them. Such forces are represented by $\mathbf{s}_{12}, \mathbf{s}_{13}, \mathbf{s}_{21}, \mathbf{s}_{23}, \mathbf{s}_{31}, \mathbf{s}_{32}$. Since $\mathbf{x}_1$ and $\mathbf{x}_2$ are too close, the social forces $\mathbf{s}_{12}$ and $\mathbf{s}_{13}$ present a repulsive nature. On the other hand, as the distances $|\mathbf{x}_1 - \mathbf{x}_3|$ and $|\mathbf{x}_2 - \mathbf{x}_3|$ are quite long, the social forces $\mathbf{s}_{13}, \mathbf{s}_{23}, \mathbf{s}_{31}$ and $\mathbf{s}_{32}$ between $\mathbf{x}_1 \leftrightarrow \mathbf{x}_3$ and $\mathbf{x}_2 \leftrightarrow \mathbf{x}_3$ are from the attractive nature. Therefore, the change of position $\Delta\mathbf{x}_1$ is computed as the resultant between $\mathbf{s}_{12}$ and $\mathbf{s}_{13}$ $(\Delta\mathbf{x}_1 = \mathbf{s}_{12} + \mathbf{s}_{13})$. The values $\Delta\mathbf{x}_2$ and $\Delta\mathbf{x}_3$ of the locusts $\mathbf{x}_1$ and $\mathbf{x}_2$ are also calculated accordingly.

In addition to the presented model [20], some studies [27–29] suggest that the social force $\mathbf{s}_{ij}$ is also affected by the dominance of the involved individuals $\mathbf{x}_i$ and

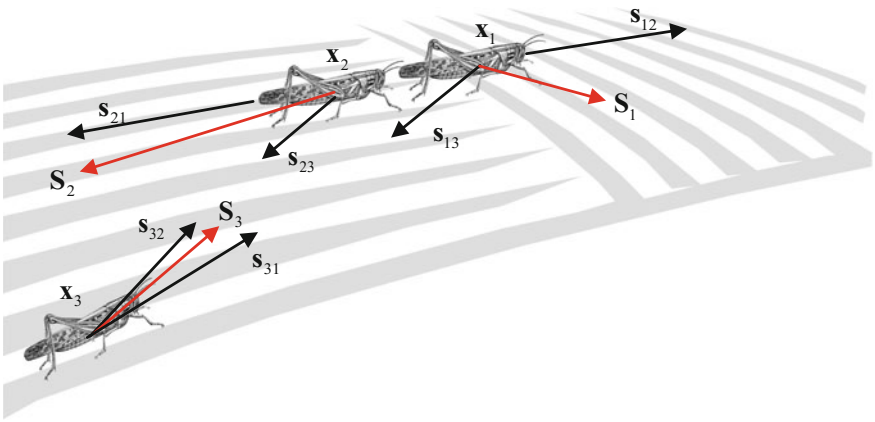

Fig. 4.1 Behavioral model under the solitary phase

$\mathbf{x}_j$ in the pairwise process. Dominance is a property that relatively qualifies the capacity of an individual to survive, in relation to other elements in a group. Dominance in locust is determined for several characteristics such as size, chemical emissions, location with regard to food sources, etc. Under such circumstances, the social force is magnified or weakened depending on the most dominant individual involved in the repulsion-attraction process.

4.2.2 Social Phase

In this phase, locusts frantically concentrate around the elements that have already found good food sources. Under such a behavior, locust attempt to efficiently find better nutrients by devastating promising areas within the plantation.

In order to simulate the social phase, to each locust $\mathbf{x}_i$ of the group, it is associated a food quality index Fq_i. This index reflex the quality of the food source where $\mathbf{x}_i$ is located.

Under this behavioral model, it is first ranked the N elements of the group according to their food quality indexes. Afterward, the b elements with the best food quality indexes are selected $(b \ll N)$. Considering a concentration radius R_c created around each selected element, a set of c new locusts is randomly generated inside R_c. As a result, most of the locusts will be concentrated around the best b elements. Figure 4.2 shows a simple example of behavioral model under the social phase. In the example, it is assumed a configuration of eight locust ($N = 8$), as it is illustrated in Fig. 4.2a. In the Figure, it is also presented the food quality index for each locust. A food quality index near to one indicates a better food source. Therefore, considering $b = 2$, the final configuration after the social phase, it is presented in Fig. 4.2b.

4.3 The Locust Search (LS) Algorithm

In this chapter, the behavioral principles from a swarm of locusts have been used as guidelines for developing a new swarm optimization algorithm. The LS assumes that entire search space is a plantation, where all the locusts interact to each other. In the LS approach, each solution within the search space represents a locust position in the plantation. Every locust receives a food quality index according to the fitness value of the solution that is symbolized by the locust. The algorithm implements two different behaviors: solitary and social. Depending on the behavior, each individual is conducted by a set of evolutionary operators which mimic the different cooperative behaviors that are typically found in the swarm.

From the implementation point of view, in the LS operation, a population $\mathbf{L}^k\left(\{\mathbf{l}_1^k, \mathbf{l}_2^k, \ldots, \mathbf{l}_N^k\}\right)$ of N locusts (individuals) is evolved from the initial point

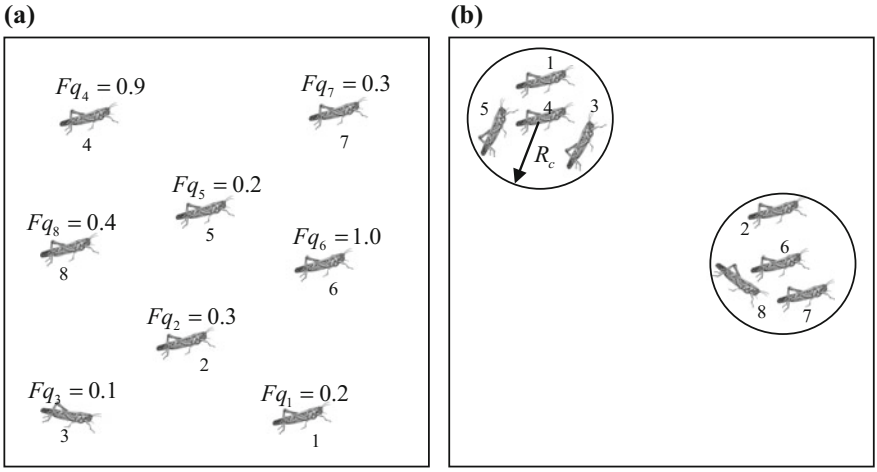

Fig. 4.2 Behavioral model under the social phase. **a** Initial configuration and food quality indexes, **b** final configuration after the operation of the social phase

($k = 0$) to a total *gen* number iterations ($k = gen$). Each locust $\mathbf{l}_i^k (i \in [1, \ldots, N])$ represents an n-dimensional vector $\left\{l_{i,1}^k, l_{i,2}^k, \ldots, l_{i,n}^k\right\}$ where each dimension corresponds to a decision variable of the optimization problem to be solved. The set of decision variables constitutes the feasible search space $\mathbf{S} = \left\{\mathbf{l}_i^k \in \mathbb{R}^n | lb_d \leq l_{i,d}^k \leq ub_d\right\}$, where lb_d and ub_d corresponds to the lower and upper bounds for the dimension d, respectively. The food quality index associated to each locust $\mathbf{l}_i^k$ (candidate solution) is evaluated by using an objective function $f\left(\mathbf{l}_i^k\right)$ whose final result represents the fitness value of $\mathbf{l}_i^k$. In LS, each iteration of the evolution process consists of two operators: (A) solitary and (B) social. Beginning by the solitary stage, the set of locusts is operated in order to sufficiently explore the search space. Then, during the social operation, existent solutions are refined within a determined neighborhood (exploitation).

4.3.1 Solitary Operation (A)

One of the most interesting features of the LS method is the use of the solitary operator to modify the current locust positions. Under this approach, locusts are displaced as a consequence of the social forces produced by the positional relations among the elements of the swarm. Therefore, near individuals tend to repel with each other, avoiding the concentration of elements in regions. On the other hand, distant individuals tend to attract with each other, maintaining the cohesion of the swarm. Different to the original model [20], in the proposed operator, social forces

are also magnified or weakened depending on the best fitness value (the most dominant) of the individuals involved in the repulsion-attraction process.

In the solitary operation, a new position $\mathbf{p}_i (i \in [1,\ldots,N])$ is produced by perturbing the current locust position $\mathbf{l}_i^k$ with a change of position $\Delta\mathbf{l}_i \left(\mathbf{p}_i = \mathbf{l}_i^k + \Delta\mathbf{l}_i\right)$. The change of position $\Delta\mathbf{l}_i$ is the result of the social interactions experimented by $\mathbf{l}_i^k$ as a consequence of its repulsion-attraction behavioral model. Such social interactions are pairwise computed among $\mathbf{l}_i^k$ and the other $N - 1$ individuals in the swarm. In the original model, social forces are calculated by using Eq. 4.3. However, in the LS method, it is modified to include the best fitness value (the most dominant) of the individuals involved in the repulsion-attraction process. Therefore, the social force exerted between $\mathbf{l}_j^k$ and $\mathbf{l}_i^k$ is calculated by using the following new model:

$$\mathbf{s}_{ij}^m = \rho(\mathbf{l}_i^k, \mathbf{l}_j^k) \cdot s(r_{ij}) \cdot \mathbf{d}_{ij} + rand(1, -1), \tag{4.6}$$

where $s(r_{ij})$ is the social force strength defined in Eq. 4.2 and $\mathbf{d}_{ij} = (\mathbf{l}_j^k - \mathbf{l}_i^k)/r_{ij}$ is the unit vector pointing from $\mathbf{l}_i^k$ to $\mathbf{l}_j^k$. Besides, $rand(1, -1)$ is a number randomly generated between 1 and -1.

$\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$ is the dominance function that calculates the dominance value of the most dominant individual from $\mathbf{l}_j^k$ and $\mathbf{l}_i^k$. In order to operate $\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$, all the individuals from $\mathbf{L}^k\left(\{\mathbf{l}_1^k, \mathbf{l}_2^k, \ldots, \mathbf{l}_N^k\}\right)$ are ranked according to their fitness values. The ranks are assigned so that the best individual receives the rank 0 (zero) whereas the worst individual obtains the rank $N - 1$. Therefore, the function $\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$ is defined as follows:

$$\rho(\mathbf{l}_i^k, \mathbf{l}_j^k) = \begin{cases} e^{-\left(5 \cdot \text{rank}(\mathbf{l}_i^k)/N\right)} & \text{if } \text{rank}(\mathbf{l}_i^k) < \text{rank}(\mathbf{l}_j^k) \\ e^{-\left(5 \cdot \text{rank}(\mathbf{l}_j^k)/N\right)} & \text{if } \text{rank}(\mathbf{l}_i^k) > \text{rank}(\mathbf{l}_j^k) \end{cases}, \tag{4.7}$$

where the function rank(α) delivers the rank of the α-individual. According to Eq. 4.7, $\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$ gives as a result a value within the interval (1,0).

The maximum value of one is reached by $\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$ when one of the individuals $\mathbf{l}_j^k$ and $\mathbf{l}_i^k$ is the best element of the population $\mathbf{L}^k$ in terms of its fitness value. On the other hand, a value close to zero, it is obtained when both individuals $\mathbf{l}_j^k$ and $\mathbf{l}_i^k$ possess quite bad fitness values. Figure 4.3 shows the behavior of $\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$ considering 100 individuals. In the Figure, it is assumed that $\mathbf{l}_i^k$ represents one of the 99 individuals with ranks between 0 and 98 whereas $\mathbf{l}_j^k$ is fixed to the element with the worst fitness value (rank 99).

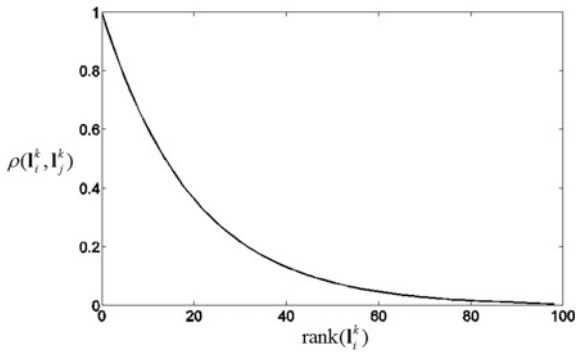

Fig. 4.3 Behavior of $\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$ considering 100 individuals

Under the incorporation of $\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$ in Eq. 4.6, social forces are magnified or weakened depending on the best fitness value (the most dominant) of the individuals involved in the repulsion-attraction process.

Finally, the total social force on each individual $\mathbf{l}_i^k$ is modeled as the superposition of all of the pairwise interactions exerted over it:

$$\mathbf{S}_i^m = \sum_{\substack{j=1 \\ j \neq i}}^{N} \mathbf{s}_{ij}^m, \tag{4.8}$$

Therefore, the change of position $\Delta\mathbf{l}_i$ is considered as the total social force experimented by $\mathbf{l}_i^k$ as the superposition of all of the pairwise interactions. Therefore, $\Delta\mathbf{l}_i$ is defined as follows:

$$\Delta\mathbf{l}_i = \mathbf{S}_i^m, \tag{4.9}$$

After calculating the new positions $\mathbf{P}(\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\})$ of the population $\mathbf{L}^k(\{\mathbf{l}_1^k, \mathbf{l}_2^k, \ldots, \mathbf{l}_N^k\})$, the final positions $\mathbf{F}(\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_N\})$ must be calculated. The idea is to admit only the changes that guarantee an improvement in the search strategy. If the fitness value of $\mathbf{p}_i(f(\mathbf{p}_i))$ is better than $\mathbf{l}_i^k\big(f(\mathbf{l}_i^k)\big)$, then $\mathbf{p}_i$ is accepted as the final solution. Otherwise, $\mathbf{l}_i^k$ is retained. This procedure can be resumed by the following statement (considering a minimization problem):

$$\mathbf{f}_i = \begin{cases} \mathbf{p}_i & \text{if } f(\mathbf{p}_i) < f(\mathbf{l}_i^k) \\ \mathbf{l}_i^k & \text{otherwise} \end{cases} \tag{4.10}$$

In order to illustrate the performance of the solitary operator, Fig. 4.4 presents a simple example where the solitary operator is iteratively applied. It is assumed a population of 50 different members ($N = 50$) which adopt a concentrated configuration as initial condition (Fig. 4.4a). As a consequence of the social forces, the set

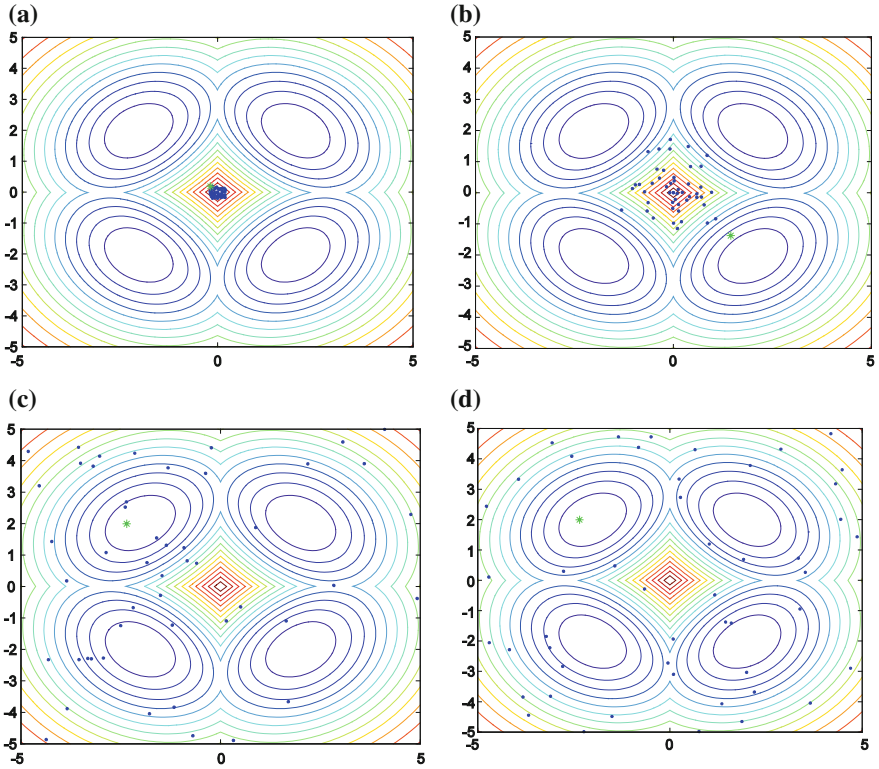

Fig. 4.4 Examples of different distributions. **a** Initial condition, **b** distribution after applying 25 operations, **c** 50 and **d** 100

of element tends to distribute through the search space. Examples of different distributions are shown in Fig. 4.4b–d after applying 25, 50 and 100 different solitary operations, respectively.

4.3.2 Social Operation (B)

The social procedure represents the exploitation phase of the LS algorithm. Exploitation is the process of refining existent individuals within a small neighborhood in order to improve their solution quality.

The social procedure is a selective operation which is applied only to a subset $\mathbf{E}$ of the final positions $\mathbf{F}$ (where $\mathbf{E} \subseteq \mathbf{F}$). In the operation first is necessary to sort $\mathbf{F}$ according to their fitness values and store the sorted elements in a temporal population $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N\}$. The elements in $\mathbf{B}$ are sorted so that the best individual receives the position $\mathbf{b}_1$ whereas the worst individual obtains the location

$\mathbf{b}_N$. Therefore, the subset $\mathbf{E}$ is integrated by only the first g locations of $\mathbf{B}$ (promising solutions). Under this operation, a subspace C_j is created around each selected particle $\mathbf{f}_j \in \mathbf{E}$. The size of C_j depends on the distance e_d which is defined as follows:

$$e_d = \frac{\sum_{q=1}^{n} \left(ub_q - lb_q\right)}{n} \cdot \beta \tag{4.11}$$

where ub_q and lb_q are the upper and lower bounds in the qth dimension, n is the number of dimensions of the optimization problem, whereas $\beta \in [0,1]$ is a tuning factor. Therefore, the limits of C_j are modeled as follows:

$$\begin{aligned} uss_j^q &= b_{j,q} + e_d \\ lss_j^q &= b_{j,q} - e_d \end{aligned} \tag{4.12}$$

where uss_j^q and lss_j^q are the upper and lower bounds of the qth dimension for the subspace C_j, respectively.

Considering the subspace C_j around each element $\mathbf{f}_j \in \mathbf{E}$, a set of h new particles $\left(\mathbf{M}_j^h = \left\{\mathbf{m}_j^1, \mathbf{m}_j^2, \ldots, \mathbf{m}_j^h\right\}\right)$ are randomly generated inside the bounds defined by Eq. 4.12. Once the h samples are generated, the individual $\mathbf{l}_j^{k+1}$ of the next population $\mathbf{L}^{k+1}$ must be created. In order to calculate $\mathbf{l}_j^{k+1}$, the best particle $\mathbf{m}_j^{best}$, in terms of fitness value from the h samples (where $\mathbf{m}_j^{best} \in \left[\mathbf{m}_j^1, \mathbf{m}_j^2, \ldots, \mathbf{m}_j^h\right]$), is compared to $\mathbf{f}_j$. If $\mathbf{m}_j^{best}$ is better than $\mathbf{f}_j$ according to their fitness values, $\mathbf{l}_j^{k+1}$ is updated with $\mathbf{m}_j^{best}$, otherwise $\mathbf{f}_j$ is selected. The elements of $\mathbf{F}$ that have not been processed by the procedure ($\mathbf{f}_w \notin \mathbf{E}$) transfer their corresponding values to $\mathbf{L}^{k+1}$ with no change.

The social operation is used to exploit only prominent solutions. According to the propose method, inside each subspace C_j, h random samples are selected. Since the number of selected samples in each subspace is very small (typically $h < 4$), the use of this operator reduces substantially the number of fitness function evaluations.

In order to demonstrate the social operation, a numerical example has been set by applying the proposed process to a simple function. Such function considers the interval of $-3 \leq d_1, d_2 \leq 3$ whereas the function possesses one global maxima of value 8.1 at (0,1.6). Notice that d_1 and d_2 correspond to the axis coordinates (commonly x and y). For this example, it is assumed a final position population $\mathbf{F}$ of six 2-dimensional members ($N = 6$). Figure 4.5 shows the initial configuration of the proposed example, the black points represents the half of the particles with the best fitness values (the first three element of $\mathbf{B}$, $g = 3$) whereas the grey points ($\mathbf{f}_2, \mathbf{f}_4, \mathbf{f}_6 \notin \mathbf{E}$) corresponds to the remaining individuals. From Fig. 4.5, it can be seen that the social procedure is applied to all black particles ($\mathbf{f}_5 = \mathbf{b}_1, \mathbf{f}_3 = \mathbf{b}_2$ and $\mathbf{f}_1 = \mathbf{b}_3, \mathbf{f}_5, \mathbf{f}_3, \mathbf{f}_1 \in \mathbf{E}$) yielding two new random particles ($h = 2$), characterized by

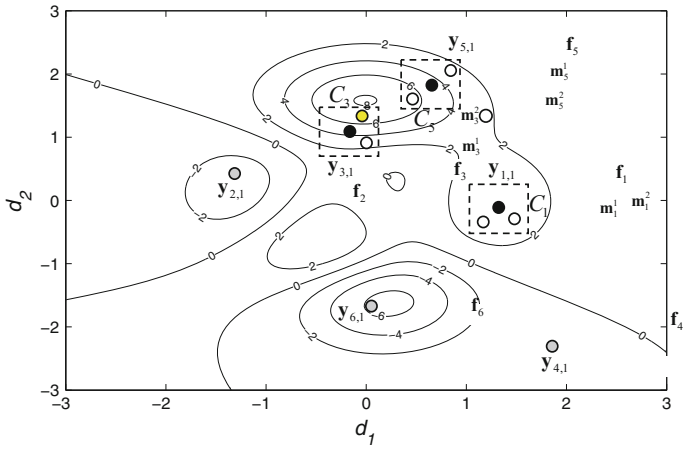

Fig. 4.5 Operation of the social procedure

the white points $\mathbf{m}_1^1, \mathbf{m}_1^2, \mathbf{m}_3^1, \mathbf{m}_3^2, \mathbf{m}_5^1$ and $\mathbf{m}_5^2$ for each black point inside of their corresponding subspaces C_1, C_3 and C_5. Considering the particle $\mathbf{f}_3$ in Fig. 4.5, the particle $\mathbf{m}_3^2$ corresponds to the best particle $\left(\mathbf{m}_3^{best}\right)$ from the two randomly generated particles (according to their fitness values) within C_3. Therefore, the particle $\mathbf{m}_3^{best}$ will substitute $\mathbf{f}_3$ in the individual $\mathbf{l}_3^{k+1}$ for the next generation, since it holds a better fitness value than $\mathbf{f}_3\left(f(\mathbf{f}_3) < f(\mathbf{m}_3^{best})\right)$.

4.3.3 Complete LS Algorithm

LS is a simple algorithm with only five adjustable parameters: the strength of attraction F, the attractive length L, number of promising solutions g, the population size N and the number of generations gen. The operation of LS is divided in three parts: Initialization, solitary operation and the social process. In the initialization ($k = 0$), the first population $\mathbf{L}^0\left(\{\mathbf{l}_1^0, \mathbf{l}_2^0, \ldots, \mathbf{l}_N^0\}\right)$ is produced. The values $\left\{l_{i,1}^0, l_{i,2}^0, \ldots, l_{i,n}^0\right\}$ of each individual $\mathbf{l}_i^k$ and each dimension d are randomly and uniformly distributed between the pre-specified lower initial parameter bound lb_d and the upper initial parameter bound ub_d.

$$l_{i,j}^0 = lb_d + \text{rand} \cdot (ub_d - lb_d); \quad i = 1, 2, \ldots, N;\ d = 1, 2, \ldots, n. \tag{4.13}$$

In the evolution process, the solitary (A) and social (B) operations are iteratively applied until the number of iterations $k = gen$ has been reached. The complete LS procedure is illustrated in the Algorithm 1.

Algorithm 1. Locust Search (LS) algorithm		
1:	**Input**: F, L, g, N and gen	
2:	Initialize $\mathbf{L}^0$ $(k = 0)$	
3:	**until** $(k = gen)$	
5:	$\mathbf{F} \leftarrow$ SolitaryOperation $\left(\mathbf{L}^k\right)$	Solitary operator (3.1)
6:	$\mathbf{L}^{k+1} \leftarrow$ SocialOperation $\left(\mathbf{L}^k, \mathbf{F}\right)$	Social operator (3.2)
8:	$k = k + 1$	
7:	**end until**	

4.3.4 Discussion About the LS Algorithm

Evolutionary algorithms (EA) 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 [30]. In an EA algorithm, search agents have to decide whether to explore unknown search positions or to exploit already tested positions in order to improve their solution quality. Pure exploration degrades the precision of the evolutionary process but increases its capacity to find new potentially solutions. On the other hand, pure exploitation allows refining existent solutions but adversely drives the process to local optimal solutions. Therefore, the ability of an EA to find a global optimal solution depends on its capacity to find a good balance between the exploitation of found-so-far elements and the exploration of the search space [31]. So far, the exploration–exploitation dilemma has been an unsolved issue within the framework of evolutionary algorithms.

Most of swarm algorithms and other evolutionary algorithms tend to exclusively concentrate the individuals in the current best positions. Under such circumstances, these algorithms seriously limit their exploration–exploitation capacities.

Different to most of existent evolutionary algorithms, in the LS approach, the modeled behavior explicitly avoids the concentration of individuals in the current best positions. Such fact allows not only to emulate in a better realistic way the cooperative behavior of the locust colony, but also to incorporate a computational mechanism to avoid critical flaws commonly present in the popular PSO and DE algorithms, such as the premature convergence and the incorrect exploration–exploitation balance.

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

4.4 Experimental Results

A comprehensive set of 13 functions, collected from Refs. [32–37], has been used to test the performance of the LS approach. Tables 4.5 and 4.6 in the Appendix present the benchmark functions used in our experimental study. Such functions are classified into two different categories: Unimodal test functions (Table 4.5) and multimodal test functions (Table 4.6). In these tables, n is the dimension of function, f_{opt} is the minimum value of the function, and $\mathbf{S}$ is a subset of R^n. The optimum location $\left(\mathbf{x}_{opt}\right)$ for functions in Tables 4.5 and 4.6, are in $[0]^n$, except for f_5, f_{12}, f_{13} with $\mathbf{x}_{opt}$ in $[1]^n$ and f_8 in $[420.96]^n$. A detailed description of optimum locations is given in Tables 4.5 and 4.6 of the Appendix.

We have applied the LS algorithm to 13 functions whose results have been compared to those produced by the Particle Swarm Optimization (PSO) method [3] and the Differential Evolution (DE) algorithm [11]. These are considered as the most popular algorithms for many optimization applications. In all comparisons, the population has been set to 40 ($N = 40$) individuals. The maximum iteration number for all functions has been set to 1000. Such stop criterion has been selected to maintain compatibility to similar works reported in the literature [34, 35].

The parameter settings for each of the algorithms in the comparison are described as follows:

1. PSO: In the algorithm, $c_1 = c_2 = 2$ while the inertia factor (ω) is decreasing linearly from 0.9 to 0.2.
2. DE: The DE/Rand/1 scheme is employed. The parameter settings follow the instructions in [11]. The crossover probability is $CR = 0.9$ and the weighting factor is $F = 0.8$.
3. In LS, F and L are set to 0.6 and L, respectively. Besides, g is fixed to 20 ($N/2$) whereas gen and N are configured to 1000 and 40, respectively. Once these parameters have been determined experimentally, they are kept for all experiments in this section.

Uni-modal test functions

Functions f_1 to f_7 are unimodal functions. The results for unimodal functions, over 30 runs, are reported in Table 4.1 considering the following performance indexes: the average best-so-far solution (ABS), the median of the best solution in the last iteration (MBS) and the standard deviation (SD). According to this table, LS provides better results than PSO and DE for all functions. In particular this test yields the largest difference in performance which is directly related to a better trade-off between exploration and exploitation produced by LS operators.

A non-parametric statistical significance proof known as the Wilcoxon's rank sum test for independent samples [38, 39] has been conducted with an 5% significance level, over the "average best-so-far" data of Table 4.1. Table 4.2 reports the p-values produced by Wilcoxon's test for the pair-wise comparison of the "average best so-far" of two groups. Such groups are formed by LS versus PSO and

Table 4.1 Minimization result of benchmark functions in Table 4.5 with $n = 30$

		PSO	DE	LS
f_1	ABS	1.66×10^{-1}	6.27×10^{-3}	4.55×10^{-4}
	MBS	0.23	5.85×10^{-3}	2.02×10^{-4}
	SD	3.79×10^{-1}	1.68×10^{-1}	6.98×10^{-4}
f_2	ABS	4.83×10^{-1}	2.02×10^{-1}	5.41×10^{-3}
	MBS	0.53	1.96×10^{-1}	5.15×10^{-3}
	SD	1.59×10^{-1}	0.66	1.45×10^{-2}
f_3	ABS	2.75	5.72×10^{-1}	1.61×10^{-3}
	MBS	3.16	6.38×10^{-1}	1.81×10^{-3}
	SD	1.01	0.15	1.32×10^{-3}
f_4	ABS	1.84	0.11	1.05×10^{-2}
	MBS	1.79	0.10	1.15×10^{-2}
	SD	0.87	0.05	6.63×10^{-3}
f_5	ABS	3.07	2.39	4.11×10^{-2}
	MBS	3.03	2.32	3.65×10^{-2}
	SD	0.42	0.36	2.74×10^{-3}
f_6	ABS	6.36	6.51	5.88×10^{-2}
	MBS	6.19	6.60	5.17×10^{-2}
	SD	0.74	0.87	1.67×10^{-2}
f_7	ABS	6.14	0.12	2.71×10^{-2}
	MBS	2.76	0.14	1.10×10^{-2}
	SD	0.73	0.02	1.18×10^{-2}

Maximum number of iterations = 1000

Table 4.2 p-values produced by Wilcoxon's test comparing LS versus PSO and DE over the "average best-so-far" values from Table 4.1

LS versus	PSO	DE
f_1	1.83×10^{-4}	1.73×10^{-2}
f_2	3.85×10^{-3}	1.83×10^{-4}
f_3	1.73×10^{-4}	6.23×10^{-3}
f_4	2.57×10^{-4}	5.21×10^{-3}
f_5	4.73×10^{-4}	1.83×10^{-3}
f_6	6.39×10^{-5}	2.15×10^{-3}
f_7	1.83×10^{-4}	2.21×10^{-3}

LS 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 LS results are statistically significant and that it has not occurred by coincidence (i.e. due to the normal noise contained in the process).

Multimodal test functions

Multimodal functions have many local minima, being the most difficult to optimize. For multimodal functions, the final results are more important since they reflect the algorithm's ability to escape from poor local optima and locate a near-global optimum. We have done experiments on f_8 to f_{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 4.3 as follows: the average best-so-far solution (ABS), the median of the best solution in the last iteration (MBS) and the standard deviation (SD). Likewise, p-values of the Wilcoxon signed-rank test of 30 independent runs are listed in Table 4.4.

Table 4.3 Minimization result of benchmark functions in Table 4.6 with $n = 30$

		PSO	DE	LS
f_8	ABS	-6.7×10^3	-1.26×10^4	-1.26×10^4
	MBS	-5.4×10^3	-1.24×10^4	-1.23×10^4
	SD	6.3×10^2	3.7×10^2	1.1×10^2
f_9	ABS	14.8	4.01×10^{-1}	2.49×10^{-3}
	MBS	13.7	2.33×10^{-1}	3.45×10^{-3}
	SD	1.39	5.1×10^{-2}	4.8×10^{-4}
f_{10}	ABS	14.7	4.66×10^{-2}	2.15×10^{-3}
	MBS	18.3	4.69×10^{-2}	1.33×10^{-3}
	SD	1.44	1.27×10^{-2}	3.18×10^{-4}
f_{11}	ABS	12.01	1.15	1.47×10^{-4}
	MBS	12.32	0.93	3.75×10^{-4}
	SD	3.12	0.06	1.48×10^{-5}
f_{12}	ABS	6.87×10^{-1}	3.74×10^{-1}	5.58×10^{-3}
	MBS	4.66×10^{-1}	3.45×10^{-1}	5.10×10^{-3}
	SD	7.07×10^{-1}	1.55×10^{-1}	4.18×10^{-4}
f_{13}	ABS	1.87×10^{-1}	1.81×10^{-2}	1.78×10^{-2}
	MBS	1.30×10^{-1}	1.91×10^{-2}	1.75×10^{-2}
	SD	5.74×10^{-1}	1.66×10^{-2}	1.64×10^{-3}

Maximum number of iterations = 1000

Table 4.4 p-values produced by Wilcoxon's test comparing LS versus PSO and DE over the "average best-so-far" values from Table 4.3

LS versus	PSO	DE
f_8	1.83×10^{-4}	0.061
f_9	1.17×10^{-4}	2.41×10^{-4}
f_{10}	1.43×10^{-4}	3.12×10^{-3}
f_{11}	6.25×10^{-4}	1.14×10^{-3}
f_{12}	2.34×10^{-5}	7.15×10^{-4}
f_{13}	4.73×10^{-4}	0.071

For f_9, f_{10}, f_{11} and f_{12}, LS yields a much better solution than the others. However, for functions f_8 and f_{13}, LS produces similar results to DE. The Wilcoxon rank test results, presented in Table 4.4, show that LS performed better than PSO and DE considering the four problems $f_9 - f_{12}$, whereas, from a statistical viewpoint, there is not difference in results between LS and DE for f_8 and f_{13}.

4.5 Conclusions

In this chapter, the Locust Search (LS) has been analyzed for solving optimization tasks. The LS algorithm is based on the simulation of the behavior presented in swarms of locusts. In the LS algorithm, individuals emulate a group of locusts which interact to each other based on the biological laws of the cooperative swarm. The algorithm considers two different behaviors: solitary and social. Depending on the behavior, each individual is conducted by a set of evolutionary operators which mimic the different cooperative behaviors that are typically found in the swarm.

Different to most of existent evolutionary algorithms, in the LS approach, the modeled behavior explicitly avoids the concentration of individuals in the current best positions. Such fact allows not only to emulate in a better realistic way the cooperative behavior of the locust colony, but also to incorporate a computational mechanism to avoid critical flaws commonly present in the popular PSO and DE algorithms, such as the premature convergence and the incorrect exploration–exploitation balance.

LS has been experimentally tested considering a suite of 13 benchmark functions. The performance of LS has been also compared to the following algorithms: the Particle Swarm Optimization method (PSO) [3], and the Differential Evolution (DE) algorithm [11]. Results have confirmed an acceptable performance of the LS method in terms of the solution quality for all tested benchmark functions.

The LS remarkable performance is associated with two different reasons: (i) the solitary operator allows a better particle distribution in the search space, increasing the algorithm's ability to find the global optima; and (ii) the use of the social operation, provides of a simple exploitation operator that intensifies the capacity of finding better solutions during the evolution process.

Appendix: List of Benchmark Functions

In Table 4.5, n is the dimension of function, f_{opt} is the minimum value of the function, and $\mathbf{S}$ is a subset of R^n. The optimum location $(\mathbf{x}_{opt})$ for functions in Table 4.5 is in $[0]^n$, except for f_5 with $\mathbf{x}_{opt}$ in $[1]^n$.

The optimum location $(\mathbf{x}_{opt})$ for functions in Table 4.6, are in $[0]^n$, except for f_8 in $[420.96]^n$ and $f_{12} - f_{13}$ in $[1]^n$.

Table 4.5 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
$f_3(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100,100]^n$	0
$f_4(\mathbf{x}) = \max_i \{ \lvert x_i \rvert, \ 1 \le i \le n \}$	$[-100,100]^n$	0
$f_5(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	$[-30,30]^n$	0
$f_6(\mathbf{x}) = \sum_{i=1}^{n} (x_i + 0.5)^2$	$[-100,100]^n$	0
$f_7(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + rand(0,1)$	$[-1.28,1.28]^n$	0

Table 4.6 Multimodal test functions

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

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press Inc, New York (1999)
2. Kassabalidis, I., El-Sharkawi, M.A., Marks II, R.J., Arabshahi, P., Gray, A.A.: Swarm intelligence for routing in communication networks. In: Global Telecommunications Conference, GLOBECOM '01, IEEE, vol. 6, pp. 3613–3617 (2001)
3. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948, December 1995
4. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report-TR06. Engineering Faculty, Computer Engineering Department, Erciyes University (2005)

5. Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst. Mag. **22**(3), 52–67 (2002)
6. Hossein, A., Hossein-Alavi, A.: Krill herd: a new bio-inspired optimization algorithm. Commun. Nonlinear Sci. Numer. Simul. **17**, 4831–4845 (2012)
7. Yang, X.S.: Engineering Optimization: An Introduction with Metaheuristic Applications. Wiley, USA (2010)
8. Yang, X.S., Deb, S.: Proceedings of World Congress on Nature & Biologically Inspired Computed, pp. 210–214. IEEE Publications, India (2009)
9. Cuevas, E., Cienfuegos, M., Zaldívar, D., Pérez-Cisneros, M.: A swarm optimization algorithm inspired in the behavior of the social-spider. Expert Syst. Appl. **40**(16), 6374–6384 (2013)
10. Cuevas, E., González, M., Zaldivar, D., Pérez-Cisneros, M., García, G.: An algorithm for global optimization inspired by collective animal behaviour. Discrete Dyn. Nat. Soc. art. no. 638275 (2012)
11. Storn, R., Price, K.: Differential evolution—a simple and efficient adaptive scheme for global optimisation over continuous spaces. Technical Report TR-95–012. ICSI, Berkeley, CA (1995)
12. Bonabeau, E.: Social insect colonies as complex adaptive systems. Ecosystems **1**, 437–443 (1998)
13. Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B., Tian, Q.: Self-adaptive learning based particle swarm optimization. Inf. Sci. **181**(20), 4515–4538 (2011)
14. Tvrdík, J.: Adaptation in differential evolution: a numerical comparison. Appl. Soft Comput. **9** (3), 1149–1155 (2009)
15. Wang, H., Sun, H., Li, C., Rahnamayan, S., Jeng-shyang, P.: Diversity enhanced particle swarm optimization with neighborhood. Inf. Sci. **223**, 119–135 (2013)
16. Gong, W., Fialho, Á., Cai, Z., Li, H.: Adaptive strategy selection in differential evolution for numerical optimization: an empirical study. Inf. Sci. **181**(24), 5364–5386 (2011)
17. Gordon, D.: The organization of work in social insect colonies. Complexity **8**(1), 43–46 (2003)
18. Kizaki, S., Katori, M.: A Stochastic lattice model for locust outbreak. Phys. A **266**, 339–342 (1999)
19. Rogers, S.M., Cullen, D.A., Anstey, M.L., Burrows, M., Dodgson, T., Matheson, T., Ott, S. R., Stettin, K., Sword, G.A., Despland, E., Simpson, S.J.: Rapid behavioural gregarization in the desert locust, Schistocerca gregaria entails synchronous changes in both activity and attraction to conspecifics. J. Insect Physiol. **65**, 9–26 (2014)
20. Topaz, C.M., Bernoff, A.J., Logan, S., Toolson, W.: A model for rolling swarms of locusts. Eur. Phys. J. Special Topics **157**, 93–109 (2008)
21. Topaz, C.M., D'Orsogna, M.R., Edelstein-Keshet, L., Bernoff, A.J.: Locust dynamics: behavioral phase change and swarming. PLoS Comput. Biol. **8**(8), 1–11
22. Oster, G., Wilson, E.: Caste and Ecology in the Social Insects. N.J. Princeton University Press, Princeton (1978)
23. Hölldobler, B., Wilson, E.O.: Journey to the Ants: A Story of Scientific Exploration (1994). ISBN 0-674-48525-4
24. Hölldobler, B., Wilson, E.O.: The Ants. Harvard University Press, USA (1990). ISBN 0-674-04075-9
25. Tanaka, S., Nishide, Y.: Behavioral phase shift in nymphs of the desert locust, Schistocerca gregaria: special attention to attraction/avoidance behaviors and the role of serotonin. J. Insect Physiol. **59**, 101–112 (2013)
26. Gaten, E., Huston, S.J., Dowse, H.B., Matheson, T.: Solitary and gregarious locusts differ in circadian rhythmicity of a visual output neuron. J. Biol. Rhythms **27**(3), 196–205 (2012)
27. Benaragama, I., Gray, J.R.: Responses of a pair of flying locusts to lateral looming visual stimuli. J. Comp. Physiol. A. **200**(8), 723–738 (2014)
28. Sergeev, M.G.: Distribution patterns of grasshoppers and their kin in the boreal zone. Psyche J. Entomol. **2011**, 9 pages, Article ID 324130 (2011)

29. Ely, S.O., Njagi, P.G.N., Bashir, M.O., El-Amin, S.E.-T., Hassanali1, A.: Diel behavioral activity patterns in adult solitarious desert locust, Schistocerca gregaria (Forskål). Psyche J. Entomol. **2011**, Article ID 459315, 9 (2011)
30. Yang, X.-S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press, Beckington (2008)
31. Cuevas, E., Echavarría, A., Ramírez-Ortegón, M.A.: An optimization algorithm inspired by the states of matter that improves the balance between exploration and exploitation. Appl. Intell. **40**(2), 256–272 (2014)
32. Ali, M.M., Khompatraporn, C., Zabinsky, Z.B.: A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. J. Global Optim. **31**(4), 635–672 (2005)
33. Chelouah, R., Siarry, P.: A continuous genetic algorithm designed for the global optimization of multimodal functions. J. Heuristics **6**(2), 191–213 (2000)
34. Herrera, F., Lozano, M., Sánchez, A.M.: A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study. Int. J. Intell. Syst. **18**(3), 309–338 (2003)
35. Laguna, M., Martí, R.: Experimental testing of advanced scatter search designs for global optimization of multimodal functions. J. Global Optim. **33**(2), 235–255 (2005)
36. Lozano, M., Herrera, F., Krasnogor, N., Molina, D.: Real-coded memetic algorithms with crossover hill-climbing. Evol. Comput. **12**(3), 273–302 (2004)
37. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**(1), 17–41 (1981)
38. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics **1**, 80–83 (1945)
39. Garcia, S., Molina, D., Lozano, M., Herrera, F.: 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. J. Heurist (2008). https://doi.org/10.1007/s10732-008-9080-4