An optimisation algorithm based on the behaviour of locust swarms

Erik Cuevas*, Adrián González,
Daniel Zaldívar and Marco Pérez-Cisneros

Departamento de Electrónica,
Universidad de Guadalajara, CUCEI,
Av. Revolución 1500, Guadalajara, Jal, México
Email: erik.cuevas@cucei.udg.mx
Email: daniel.zaldivar@cucei.udg.mx
Email: marco.perez@cucei.udg.mx
*Corresponding author

Abstract: In this paper, a novel swarm algorithm called locust search (LS) is proposed for solving optimisation tasks. The LS algorithm is based on the simulation of the behaviour presented in swarms of locusts. In the proposed algorithm, individuals emulate a group of locusts which interact to each other based on the biological laws of the cooperative swarm. Experimental results demonstrate a high performance of the proposed method for searching a global optimum in comparison with other well-known evolutionary methods.

Keywords: swarm algorithms; global optimisation; bio-inspired algorithms.

Reference to this paper should be made as follows: Cuevas, E., González, A., Zaldívar, D. and Pérez-Cisneros, M. (2015) 'An optimisation algorithm based on the behaviour of locust swarms', *Int. J. Bio-Inspired Computation*, Vol. 7, No. 6, pp.402–407.

Biographical notes: Erik Cuevas received his BS in Electronics Engineering from the University of Guadalajara, Mexico in 1995, MS in Industrial Electronics from ITESO, Mexico in 2000, and PhD from Freie Universität Berlin, Germany in 2006. From 2001, he was awarded a scholarship from the German Service for Academic Interchange (DAAD) as full-time researcher. He is currently a Full-time Professor in the Department of Electronics at the University of Guadalajara. He serves as an editorial board member for the *Journal of Mathematical Problems in Engineering* and *Journal of Imaging*. His current research interest includes computer vision and evolutionary computation.

Adrián González received his BS in Electronics from the University of Guadalajara, Mexico in 2011, MSc in Electronic Engineering and Computer Sciences from the University of Guadalajara, Mexico in 2014. Currently, he is a PhD student at University of Guadalajara. His current research interests include computer vision, image processing, artificial intelligence and metaheuristic optimisation algorithms.

Daniel Zaldívar received his BS with distinction in Electronics and Communications Engineering from the University of Guadalajara, Mexico in 1995, MSc in Industrial Electronics from ITESO, Mexico in 2000, and PhD from Freie Universität Berlin, Germany in 2005. From 2001, he was awarded a scholarship from the German Service for Academic Interchange (DAAD) as a fulltime researcher. Since 2007, he has been with University of Guadalajara, where he is currently a Fulltime Professor in the Department of Electronics. From 2008, he is a member of the Mexican National Research System (SNI). His current research interests include artificial intelligence and robotics.

Marco Pérez-Cisneros received his BS with distinction in Electronics and Communications Engineering from the University of Guadalajara, Guadalajara, Mexico in 1995, MSc in Industrial Electronics from Instituto Tecnologico de Estudios Superiores de Occidente, Guadalajara in 2000, and PhD from the Institute of Science and Technology, The University of Manchester, Manchester, UK in 2004. He was a research consultant on robotic manipulators for TQ Ltd., Derbyshire, UK in 2003. Since 2005, he has been with the University of Guadalajara, where he is currently Professor and Dean of the Science Division, Tonalá Campus. His current research interest includes robotics and computational vision, particularly visual servoing and intelligent robot control. He has been a member of the Mexican National Research System (SNI) since 2007.

Copyright © 2015 Inderscience Enterprises Ltd.

1 Introduction

The collective intelligent behaviour 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 behaviour. Several swarm algorithms have been developed by a combination of deterministic rules and randomness, mimicking the behaviour of insect or animal groups in nature. Such methods include the social behaviour of bird flocking and fish schooling such as the particle swarm optimisation (PSO) algorithm (Kennedy and Eberhart, 1995), the cooperative behaviour of bee colonies such as the artificial bee colony (ABC) technique (Karaboga, 2005), the emulation of the lifestyle of cuckoo birds such as the cuckoo search (CS) (Yang and Deb, 2009), the social-spider behaviour such as the social spider optimisation (SSO) (Cuevas et al., 2013), the simulation of the animal behaviour in a group such as the collective animal behaviour (Cuevas et al., 2012) and the emulation of the differential evolution (DE) in species such as the DE (Storn and Price, 1995).

The interesting and exotic collective behaviour of insects have fascinated and attracted researchers for many years. The intelligent behaviour 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 behaviour (Gordon, 2003). Locusts (*Schistocerca gregaria*) are a representative example of such collaborative insects (Kizaki and Katori, 1999). Locust is a kind of grasshopper that can change reversibly between a solitary and a social phase, which differ considerably in behaviour (Rogers et al., 2014). The two phases show many differences including both overall levels of activity and the degree to which locusts are attracted or repulsed among them (Topaz et al., 2008). In the solitary phase, locusts avoid contact each other (locust concentrations). As consequence, they distribute throughout the space, exploring sufficiently the plantation (Topaz et al., 2008). On other hand, in the social phase, locusts frantically concentrate around the elements that have already found good food sources (Topaz et al., 2012). Under such a behaviour, locust attempt to efficiently find better nutrients by devastating promising areas within the plantation.

To the best of our knowledge, only one locust-inspired algorithm aiming at solving optimisation problems has been proposed, i.e., the SSO devised by Chen (2009), which only modifies the popular PSO algorithm without considering the biological locust behaviour. However, the proposed algorithm is totally different from this algorithm in their biological backgrounds, motivations, implementations, and search behaviours.

In this paper, a novel swarm algorithm, called the locust search (LS) is proposed for solving optimisation tasks. The LS algorithm is based on the simulation of the behaviour presented in swarms of locusts. In the proposed 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 behaviours: solitary and social. Depending on the behaviour, each individual is conducted by a set of evolutionary operators which mimic the different cooperative behaviours that are typically found in the swarm. Experimental results demonstrate a high performance of the proposed method for searching a global optimum in comparison with other well-known evolutionary methods.

This paper is organised as follows. In Section 2, the novel LS algorithm and its characteristics are both described. Section 3 presents the experimental results and the comparative study. Finally, in Section 4, conclusions are drawn.

2 The LS algorithm

In this paper, the behavioural principles from a swarm of locusts have been used as guidelines for developing a new swarm optimisation algorithm. The LS assumes that the entire search space is a plantation, where all the locusts interact to each other. In the proposed 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 symbolised by the locust. The algorithm implements two different behaviours: solitary and social. Depending on the behaviour, each individual is conducted by a set of evolutionary operators which mimic the different cooperative behaviours that are typically found in the swarm.

From the implementation point of view, in the LS operation, a population $\mathbf{L}^k(\{\mathbf{l}_1^k, \mathbf{l}_2^k, \ldots, \mathbf{l}_N^k\})$ of N locusts (individuals) is evolved from the initial point ($k = 0$) to a total *gen* number iterations ($k = gen$). Each locust $\mathbf{l}_i^k (i \in [1, \ldots, N])$ represents an n-dimensional vector $\{l_{i,1}^k, l_{i,2}^k, \ldots, l_{i,n}^k\}$ where each dimension corresponds to a decision variable of the optimisation problem to be solved. The set of decision variables constitutes the feasible search space $\mathbf{S} = \{\mathbf{l}_i^k \in \mathbb{R}^n \mid lb_d \le l_{i,d}^k \le ub_d\}$, where lb_d and ub_d corresponds to the lower and upper bounds for the d dimension, respectively. The food quality index associated to each locust $\mathbf{l}_i^k$ (candidate solution) is evaluated by using an objective function $f(\mathbf{l}_i^k)$ whose final result represents the fitness value of $\mathbf{l}_i^k$. In LS, at each iteration of the evolution process consists of two operators:

a solitary

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 neighbourhood (exploitation).

2.1 Solitary operation

One of the most interesting features of the proposed 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 (Topaz et al., 2008), 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 $\Delta\mathbf{l}_i$ ($\mathbf{p}_i = \mathbf{l}_i^k + \Delta\mathbf{l}_i$). This change of position $\Delta\mathbf{I}_i$ is the result of the social interactions experimented by $\mathbf{l}_i^k$ as a consequence of its repulsion-attraction behavioural model. Such social interactions are pairwise computed among $\mathbf{l}_i^k$ and the other N-1 individuals in the swarm. Therefore, the social force exerted between $\mathbf{l}_j^k$ and $\mathbf{l}_i^k$ is calculated by using the following model:

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

where $\mathbf{d}_{ij} = (\mathbf{l}_j^k - \mathbf{l}_i^k) / r_{ij}$ is the unit vector, pointing from $\mathbf{l}_i^k$ to $\mathbf{l}_j^k$. Furthermore, *rand*(1, –1) is a number randomly generated between 1 and –1. The term $s(r_{ij})$ represents the social relation between $\mathbf{l}_j^k$ and $\mathbf{l}_i^k$, which is defined as follows:

$$s\left(r_{ij}\right) = F \cdot e^{-r_{ij}/L} - e^{-r_{ij}} \tag{2}$$

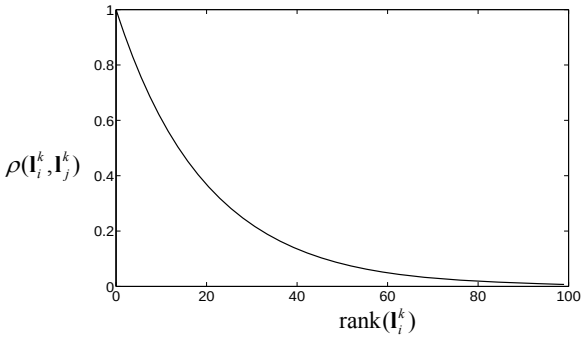
Here, r_{ij} is the distance between $\mathbf{l}_j^k$ and $\mathbf{l}_i^k$, F describes the magnitude of attraction, and L is the attractive length scale. We have scaled the space coordinates so that the repulsive strength and length scale are both represented by the unity. We assume that F<1 and L>1 so that repulsion is stronger in a shorter-scale, while attraction is applied in a weaker and longer-scale; both facts are typical for social organisms (Topaz et al., 2012). $\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$ is a function that calculates the dominance value of the most dominant individual between $\mathbf{l}_j^k$ and $\mathbf{l}_i^k$. In order to operate $\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$, all the individuals from $\mathbf{L}^k(\{\mathbf{l}_1^k, \mathbf{l}_2^k, \ldots, \mathbf{l}_N^k\})$ 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:

$$\rho\left(\mathbf{l}_i^k, \mathbf{l}_j^k\right) = \begin{cases} e^{-\left(5 \cdot \mathrm{rank}\left(\mathbf{l}_i^k\right)/N\right)} & \text{if } \mathrm{rank}\left(\mathbf{l}_i^k\right) < \mathrm{rank}\left(\mathbf{l}_j^k\right) \\ e^{-\left(5 \cdot \mathrm{rank}\left(\mathbf{l}_j^k\right)/N\right)} & \text{if } \mathrm{rank}\left(\mathbf{l}_i^k\right) > \mathrm{rank}\left(\mathbf{l}_j^k\right) \end{cases} \tag{3}$$

where the function rank(α) delivers the rank of the α-individual. The term $\rho(\mathbf{l}_i^k, \mathbf{l}_j^k)$ gives as a result a value within the interval [0, 1]. 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 1 shows the behaviour 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).

Figure 1 Behaviour 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 equation (1), 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 modelled as the superposition of all of the pairwise interactions exerted on it:

$$\mathbf{S}_i^m = \sum_{\substack{j=1 \\ j \neq i}}^{N} \mathbf{s}_{ij}^m \tag{4}$$

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

$$\Delta\mathbf{l}_i = \mathbf{S}_i^m \tag{5}$$

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 $\mathbf{l}_i^k(f(\mathbf{l}_i^k))$, 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 minimisation problem):

$$\mathbf{f}_i = \begin{cases} \mathbf{p}_i & \text{if } f(\mathbf{p}_i) < f(\mathbf{l}_i^k) \\ \mathbf{l}_i^k & \text{otherwise} \end{cases} \tag{6}$$

2.2 Social operation

The social procedure represents the exploitation phase of the LS algorithm. Exploitation is the process of refining existent individuals within a small neighbourhood in order to improve their solution quality. The social procedure is a selective operation which is applied only to a subset **E** of the final positions **F** (where $\mathbf{E} \subseteq \mathbf{F}$). In the operation first is necessary to sort **F** according to their fitness values and store the elements in a temporal population $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N\}$. The elements in **B** are sorted so that the best individual receives the position $\mathbf{b}_1$ $\{b_{1,1}, b_{1,2}, \ldots, b_{1,n}\}$ whereas the worst individual obtains the location $\mathbf{b}_N$. Therefore, the subset **E** is integrated by only the first g locations of **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:

$$e_d = \frac{\sum_{q=1}^{n} (ub_q - lb_q)}{n} \cdot \beta \tag{7}$$

where ub_q and lb_q are the upper and lower bounds in the q^{th} dimension, n is the number of dimensions of the optimisation problem, whereas $\beta \in [0, 1]$ is a tuning factor. Therefore, the limits of C_j are modelled as follows:

$$\begin{aligned} uss_j^q &= b_{j,q} + e_d \\ lss_j^q &= b_{j,q} - e_d \end{aligned} \tag{8}$$

where uss_j^q and lss_j^q are the upper and lower bounds of the q^{th}-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 $(\mathbf{M}_j^h = \{\mathbf{m}_j^1, \mathbf{m}_j^2, \ldots, \mathbf{m}_j^h\})$ are randomly generated inside the bounds defined by equation (8). 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 $\mathbf{m}_j^{best} \in [\mathbf{m}_j^1, \mathbf{m}_j^2, \ldots, \mathbf{m}_j^h]$), is compared with $\mathbf{f}_j$. 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 **F** that have not been processed by the procedure ($\mathbf{f}_w \notin \mathbf{E}$) transfer their corresponding values to $\mathbf{L}^{k+1}$ without 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 cannot be considered computational expensive.

2.3 Computational procedure

LS is a simple algorithm with only five adjustable parameters: the attraction magnitude 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: initialisation, solitary operation and the social process. In the initialisation ($k = 0$), the first population $\mathbf{L}^0(\{\mathbf{l}_1^0, \mathbf{l}_2^0, \ldots, \mathbf{l}_N^0\})$ is produced. The values $\{l_{i,1}^0, l_{i,2}^0, \ldots, l_{i,n}^0\}$ 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.

$$\begin{aligned} &l_{i,j}^0 = lb_d + \text{rand} \cdot (ub_d - lb_d); \\ &i = 1, 2, \ldots, N; \quad d = 1, 2, \ldots, n. \end{aligned} \tag{9}$$

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 Table 1.

Table 1 Computational LS algorithm

Algorithm 1: LS algorithm		
1:	**Input:** F, L, g, N and *gen*	
2:	Initialise $\mathbf{L}^0$ ($k = 0$)	
3:	**until** ($k = gen$)	
5:	$\mathbf{F} \leftarrow$ SolitaryOperation($\mathbf{L}^k$)	Solitary operator (2.1)
6:	$\mathbf{L}^{k+1} \leftarrow$ SocialOperation($\mathbf{L}^k$, $\mathbf{F}$)	Social operator (2.2)
8:	$k = k+1$	
7:	**end until**	

3 The experimental results

A comprehensive set of six functions, collected from Ali et al. (2005), has been used to test the performance of the proposed approach. Table 2 presents the benchmark functions used in the experimental study. In the table, 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 ($\mathbf{x}_{opt}$) is in $[0]^n$, except for f_1, f_5 and f_6 which are in $[420.96]^n$ and $[1]^n$, respectively. We have applied the LS algorithm to six functions whose results have been compared to those produced by the PSO method (Kennedy and Eberhart, 1995) and the DE algorithm (Storn and Price, 1995). These are considered as the most popular algorithms for many optimisation 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 1,000. Such stop criterion has been selected to maintain compatibility to similar works reported in the literature (Laguna and Martí, 2005). 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 Storn and Price (1995). 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 1,000 and 40, respectively. Once these parameters have been determined experimentally, they are kept for all experiments in this section.

Table 2 Optimisation result of benchmark functions

		PSO	*DE*	*LS*
f_1	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_2	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_3	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_4	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_5	ABS	6.87×10^{-1}	3.74×10^{-1}	5.58×10^{-3}
	MBS	4.66×10^{-1}	3.54×10^{-1}	5.10×10^{-3}
	SD	7.07×10^{-1}	1.55×10^{-1}	4.18×10^{-4}
F_6	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}

Table 3 p-values produced by Wilcoxon's test

LS vs.	*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

The results for the functions, over 30 runs, are reported in Table 3 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 (1945) rank sum test for independent samples has been conducted with an 5% significance level, over the 'average best-so-far' data of Table 1. Table 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 vs. PSO and LS vs. 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' (ABS) values of both approaches. For f_2, f_3, f_4 and f_5, LS yields a much better solution than the others (its values are less than 0.05).

However, for functions f_1 and f_6, LS produces similar results to DE. The Wilcoxon rank test results, presented in Table 2, show that LS performed better than PSO and DE considering the four problems f_2–f_5, whereas, from a statistical viewpoint, there is not difference in results between LS and DE for f_1 and f_6.

4 Conclusions

In this paper, a novel swarm algorithm, called the LS has been proposed for solving optimisation tasks. The LS algorithm is based on the behavioural simulation of swarms of locusts. In the proposed 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 behaviours: solitary and social. Depending on the selected behavioural scheme, each individual is conducted by a set of evolutionary operators which mimic the different cooperative conducts that are typically found in the swarm. Different to most of existent swarm algorithms, the behavioural model in the proposed approach explicitly avoids the concentration of individuals in the current best positions. Such fact allows not only to emulate in a good realistic way the cooperative behaviour of the locust colony, but also to incorporate a computational mechanism to avoid critical flaws that are 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 compared to the following algorithms: the PSO method (Kennedy and Eberhart, 1995), and the DE algorithm (Storn and Price, 1995). Results have produced an acceptable performance of the proposed method in terms of the solution quality for all tested benchmark functions.

Acknowledgements

The proposed algorithm is part of the optimisation system used by a biped robot under the project CONACYT CB 181053.

References

Ali, M.M., Khompatraporn, C. and Zabinsky, Z.B. (2005) '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.

Chen, S. (2009) 'Locust swarms – a new multi-optima search technique', *Evolutionary Computation*, *CEC '09*, *IEEE Congress on*, pp 1745–1752.

Cuevas, E., Cienfuegos, M., Zaldívar, D. and Pérez-Cisneros, M. (2013) 'A swarm optimization algorithm inspired in the behavior of the social-spider', *Expert Systems with Applications*, Vol. 40, No. 16, pp.6374–6384.

Cuevas, E., González, M., Zaldivar, D., Pérez-Cisneros, M. and García, G. (2012) 'An algorithm for global optimization inspired by collective animal behaviour', *Discrete Dynamics in Nature and Society*, Article ID 638275, 24pp, doi:10.1155/2012/638275.

Gordon, D. (2003) 'The organization of work in social insect colonies', *Complexity*, Vol. 8, No. 1, pp.43–46.

Karaboga, D. (2005) *An Idea Based on Honey Bee Swarm for Numerical Optimization*, TechnicalReport-TR06, Engineering Faculty, Computer Engineering Department, Erciyes University.

Kennedy, J. and Eberhart, R. (1995) 'Particle swarm optimization', in *Proceedings of the IEEE International Conference on Neural Networks*, December, Vol. 4, pp.1942–1948.

Kizaki, S. and Katori, M. (1999) 'A stochastic lattice model for locust outbreak', *Physica A*, Vol. 266, Nos. 1–4, pp.339–342.

Laguna, M. and Martí, R. (2005) 'Experimental testing of advanced scatter search designs for global optimization of multimodal functions', *Journal of Global Optimization*, Vol. 33, No. 2, pp.235–255.

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. and Simpson, S.J. (2014) 'Rapid behavioural gregarization in the desert locust, Schistocerca gregaria entails synchronous changes in both activity and attraction to conspecifics', *Journal of Insect Physiology*, Vol. 65, pp.9–26.

Storn, R. and Price, K. (1995) *Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimisation over Continuous Spaces,* TechnicalReportTR-95–012, ICSI, Berkeley, CA.

Topaz, C.M., Bernoff, A.J., Logan, S. and Toolson, W. (2008) 'A model for rolling swarms of locusts', *Eur. Phys. J. Special Topics*, Vol. 157, pp.93–109.

Topaz, C.M., D'Orsogna, M.R., Edelstein-Keshet, L. and Bernoff, A.J. (2012) 'Locust dynamics: behavioral phase change and swarming', *Plos Computational Biology*, Vol. 8, No. 8, pp.1–11.

Wilcoxon, F. (1945) 'Individual comparisons by ranking methods', *Biometrics*, Vol. 1, No. 6, pp.80–83.

Yang, X.S. and Deb, S. (2009) *Proceedings of World Congress on Nature & Biologically Inspired Computed*, IEEE Publications, India, pp.210–214.

View publication stats