Available online at www.sciencedirect.com

ScienceDirect

Applied Soft Computing 8 (2008) 687–697

Applied Soft Computing

www.elsevier.com/locate/asoc

On the performance of artificial bee colony (ABC) algorithm

D. Karaboga [*], B. Basturk

Erciyes University, Engineering Faculty, Computer Engineering Department, TR-38039 Kayseri, Turkey

Received 17 August 2006; received in revised form 9 January 2007; accepted 30 May 2007
Available online 2 June 2007

Abstract

Artificial bee colony (ABC) algorithm is an optimization algorithm based on a particular intelligent behaviour of honeybee swarms. This work compares the performance of ABC algorithm with that of differential evolution (DE), particle swarm optimization (PSO) and evolutionary algorithm (EA) for multi-dimensional numeric problems. The simulation results show that the performance of ABC algorithm is comparable to those of the mentioned algorithms and can be efficiently employed to solve engineering problems with high dimensionality.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Swarm intelligence; Bee colony algorithm; ABC; DE; PSO; EA; Numerical optimization

1. Introduction

Evolutionary algorithms (EA) are generally known as general-purpose optimization algorithms, which are capable of finding near-optimal solutions to the numerical, real-valued test problems for which exact and analytical methods do not produce optimal solutions within a reasonable computation time. One of the evolutionary algorithms which has been introduced recently is differential evolution (DE) algorithm [1]. The DE algorithm has been proposed to overcome the main disadvantage of poor local search ability of genetic algorithm (GA) [2]. The important difference between the GA and the DE algorithm is at the selection operations they employed.

At the selection operation of the GA, the chance of being selected of a solution as a parent depends on the fitness value of that solution. In DE algorithm, all solutions have an equal chance of being selected as parents, i.e. the chance does not depend on their fitness values. After a new solution is produced by using a self-adjusting mutation operation and a crossover operation, the new solution competes with its parent for the next generation and the better one wins the competition. In other words, a greedy scheme is applied to select one of them for the next generation. The use of a mutation operation, which has the self-adaptability feature, a crossover operation and a greedy process for the selection, makes DE a fast converging evolutionary algorithm. Besides its simplicity and flexibility, DE also does not face any Hamming Cliff problem like the binary GA [3,4]. Therefore, DE algorithm has received significant interest from researchers studying in different research areas and has been applied to several real-world problems [5–8].

Swarm intelligence has become a research interest to many research scientists of related fields in recent years. The swarm intelligence is defined as "...any attempt to design algorithms or distributed problem-solving devices inspired by the collective behaviour of social insect colonies and other animal societies..." by Bonabeau et al. [9]. Bonabeau et al. focused their viewpoint on social insects alone, such as termites, bees, wasps as well as different ant species. However, the term swarm is used in a general manner to refer to any restrained collection of interacting agents or individuals. The classical example of a swarm is bees swarming around their hive; nevertheless the metaphor can easily be extended to other systems with a similar architecture. For instance, an ant colony can be thought of as a swarm whose individual agents are ants; a flock of birds is a swarm of birds; an immune system [10] is a swarm of cells as well as a crowd is a swarm of people [11].

Particle swarm optimization (PSO) algorithm, which has become quite popular recently, models the social behaviour of bird flocking or fish schooling [12]. PSO is a population-based stochastic optimization technique and well adapted to the optimization of nonlinear functions in multi-dimensional

* Corresponding author. Tel.: +90 352 437 49 01x32577; fax: +90 352 437 57 84.
E-mail addresses: karaboga@erciyes.edu.tr (D. Karaboga), bahriye@erciyes.edu.tr (B. Basturk).

1568-4946/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2007.05.007

space. PSO consists of a swarm of particles moving in a search space of possible solutions for a problem. Every particle has a position vector representing a candidate solution to the problem and a velocity vector. Moreover, each particle contains a small memory that stores its own best position seen so far and a global best position obtained through communication with its neighbour particles.

A few models have been developed to model the intelligent behaviours of honeybee swarms and applied for solving combinatorial type problems [13–20]. There is only one numerical optimization algorithm in the literature based on intelligent behaviour of honeybee swarms [21]. Yang developed a virtual bee algorithm (VBA) [21] to solve the numerical optimization problems. VBA has been introduced to optimize only the functions with two parameters. In VBA, a swarm of virtual bees are generated and started to move randomly in the phase space. These bees interact when they find some target nectar corresponding to the encoded values of the function. The solution for the optimization problem can be obtained from the intensity of bee interactions. For optimizing multivariable numerical functions, Karaboga has described a bee swarm algorithm called artificial bee colony (ABC) algorithm [22], which is different from the virtual bee algorithm, and Basturk and Karaboga compared the performance of ABC algorithm with the performance of GA in [23].

This work compares the performance of ABC algorithm with that of DE and PSO algorithms, and EA for a set of well-known test functions. Also, the performance of ABC is analysed under the change of control parameter values. In Section 2, the behaviour of real honeybees is described and then the artificial bee colony algorithm is introduced in Section 3. In Section 4, the experimental study is described and finally, the simulation results obtained are presented and discussed in Section 5.

2. Behaviour of real bees

The minimal model of forage selection that lead to the emergence of collective intelligence of honey bee swarms consists of three essential components: food sources, employed foragers and unemployed foragers, and defines two leading modes of the behaviour: recruitment to a nectar source and abandonment of a source [24].

(i) Food sources: the value of a food source depends on many factors, such as its proximity to the nest, richness or concentration of energy and the ease of extracting this energy. For the simplicity, the ‘‘profitability’’ of a food source can be represented with a single quantity [25].
(ii) Employed foragers: they are associated with a particular food source, which they are currently exploiting or are ‘‘employed’’ at. They carry with them information about this particular source, its distance and direction from the nest and the profitability of the source and share this information with a certain probability.
(iii) Unemployed foragers: they are looking for a food source to exploit. There are two types of unemployed foragers—scouts searching the environment surrounding the nest for new food sources and onlookers waiting in the nest and finding a food source through the information shared by employed foragers. The mean number of scouts averaged over conditions is about 5–10% [25].

The exchange of information among bees is the most important occurrence in the formation of collective knowledge. While examining the entire hive, it is possible to distinguish some parts that commonly exist in all hives. The most important part of the hive with respect to exchanging information is the dancing area. Communication among bees related to the quality of food sources occurs in the dancing area. The related dance is called waggle dance. Since information about all the current rich sources is available to an onlooker on the dance floor, she probably could watch numerous dances and choose to employ herself at the most profitable source. There is a greater probability of onlookers choosing more profitable sources since more information is circulating about the more profitable sources. Employed foragers share their information with a probability, which is proportional to the profitability of the food source, and the sharing of this information through waggle dancing is longer in duration. Hence, the recruitment is proportional to profitability of a food source [15].

In order to understand the basic behaviour characteristics of foragers better, let us examine the Fig. 1. Assume that there are two discovered food sources: A and B. At the very beginning, a potential forager will start as unemployed forager. That bee will have no knowledge about the food sources around the nest.

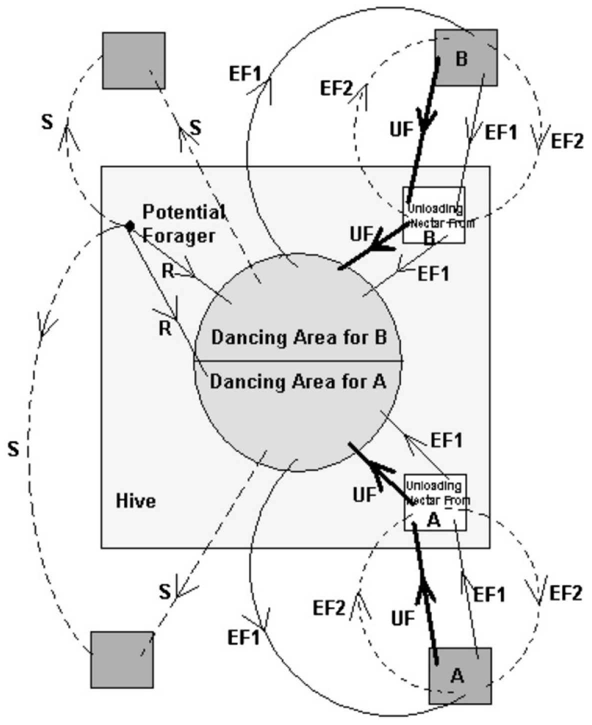

Fig. 1. Behaviour of honeybee foraging for nectar.

There are two possible options for such a bee:

(i) It can be a scout and starts searching around the nest spontaneously for a food due to some internal motivation or possible external clue ('S' in Fig. 1).
(ii) It can be a recruit after watching the waggle dances and starts searching for a food source ('R' in Fig. 1).

After finding the food source, the bee utilizes its own capability to memorize the location and then immediately starts exploiting it. Hence, the bee will become an "employed forager". The foraging bee takes a load of nectar from the source and returns to the hive, unloading the nectar to a food store. After unloading the food, the bee has the following options:

(i) It might become an uncommitted follower after abandoning the food source (UF).
(ii) It might dance and then recruit nest mates before returning to the same food source (EF1).
(iii) It might continue to forage at the food source without recruiting after bees (EF2).

It is important to note that not all bees start foraging simultaneously. The experiments confirmed that new bees begin foraging at a rate proportional to the difference between the eventual total number of bees and the number presently foraging.

3. Artificial bee colony (ABC) algorithm

In ABC algorithm, the colony of artificial bees contains three groups of bees: employed bees, onlookers and scouts. First half of the colony consists of the employed artificial bees and the second half includes the onlookers. For every food source, there is only one employed bee. In other words, the number of employed bees is equal to the number of food sources. The employed bee of an abandoned food source becomes a scout. The search carried out by the artificial bees can be summarized as follows:

- Employed bees determine a food source within the neighbourhood of the food source in their memory.
- Employed bees share their information with onlookers within the hive and then the onlookers select one of the food sources.
- Onlookers select a food source within the neighbourhood of the food sources chosen by themselves.
- An employed bee of which the source has been abandoned becomes a scout and starts to search a new food source randomly.

The main steps of the algorithm are given below:

```
Initialize
REPEAT
• Move the employed bees onto their food sources and determine their nectar amounts.
• Move the onlookers onto the food sources and determine their nectar amounts.
• Move the scouts for searching new food sources.
• Memorize the best food source found so far.
UNTIL (requirements are met)
```

Each cycle of the search consists of three steps: moving the employed and onlooker bees onto the food sources and calculating their nectar amounts and determining the scout bees and then moving them randomly onto the possible food sources. A food source represents a possible solution to the problem to be optimized. The nectar amount of a food source corresponds to the quality of the solution represented by that food source. Onlookers are placed on the foods by using "roulette wheel selection" method [26]. Every bee colony has scouts that are the colony's explorers. The explorers do not have any guidance while looking for food. They are primarily concerned with finding any kind of food source. As a result of such behaviour, the scouts are characterized by low search costs and a low average in food source quality. Occasionally, the scouts can accidentally discover rich, entirely unknown food sources. In the case of artificial bees, the artificial scouts could have the fast discovery of the group of feasible solutions as a task. In ABC algorithm, one of the employed bees is selected and classified as the scout bee. The classification is controlled by a control parameter called "limit". If a solution representing a food source is not improved by a predetermined number of trials, then that food source is abandoned by its employed bee and the employed bee associated with that food source becomes a scout. The number of trials for releasing a food source is equal to the value of "limit", which is an important control parameter of ABC algorithm.

In a robust search process, exploration and exploitation processes must be carried out together. In the ABC algorithm, while onlookers and employed bees carry out the exploitation process in the search space, the scouts control the exploration process. In the case of real honeybees, the recruitment rate represents a "measure" of how quickly the bee colony finds and exploits a newly discovered food source. Artificial recruiting could similarly represent the "measurement" of the speed with which the feasible solutions or the "good quality" solutions of the difficult optimization problems can be discovered. The survival and progress of the bee colony are dependent upon the rapid discovery and efficient utilization of the best food resources. Similarly the successful solution of difficult engineering problems is connected to the relatively fast discovery of "good solutions" especially for the problems that need to be solved in real time.

As other social foragers, bees search for food sources in a way that maximizes the ratio E/T (where E is the energy obtained and T is the time spent for foraging). In the case of bee swarms, E is proportional to the nectar amount of food sources discovered by bees and the bee swarm works to maximize the honey being stored inside the hive. In a maximization problem, the goal is to find the maximum of the objective function $F(\theta)$, $\theta \in R^p$. Assume that θ_i is the position of the ith food source; $F(\theta_i)$ represents the nectar amount of the food source located at

θ_i and is proportional to the energy $E(\theta_i)$. Let $P(c) = \{\theta_i(c)|i = 1, 2, \ldots, S\}$ (c: cycle, S: number of food sources around the hive) represent the population of food sources being visited by bees.

As mentioned before, the preference of a food source by an onlooker bee depends on the nectar amount $F(\theta)$ of that food source. As the nectar amount of the food source increases, the probability with the preferred source by an onlooker bee increases proportionally. Therefore, the probability with the food source located at θ_i will be chosen by a bee can be expressed as

$$P_i = \frac{F(\theta_i)}{\sum_{k=1}^{S} F(\theta_k)} \qquad (1)$$

After watching the dances of employed bees, an onlooker bee goes to the region of food source located at θ_i by this probability and determines a neighbour food source to take its nectar depending on some visual information, such as signs existing on the patches. In other words, the onlooker bee selects one of the food sources after making a comparison among the food sources around θ_i. The position of the selected neighbour food source is calculated as the following:

$$\theta_i(c+1) = \theta_i(c) \pm \phi_i(c) \qquad (2)$$

$\phi_i(c)$ is a randomly produced step to find a food source with more nectar around θ_i. $\phi(c)$ is calculated by taking the difference of the same parts of $\phi_i(c)$ and $\phi_k(c)$ (k is a randomly produced index) food positions. If the nectar amount $F(\theta_i(c + 1))$ at $\theta_i(c + 1)$ is higher than that at $\theta_i(c)$, then the bee goes to the hive and share her information with others and the position $\theta_i(c)$ of the food source is changed to be $\theta_i(c + 1)$, otherwise $\theta_i(c)$ is kept as it is.

Every food source has only one employed bee. Therefore, the number of employed bees is equal to the number of food sources. If the position θ_i of the food source i cannot be improved through the predetermined number of trials ‘‘limit’’, then that food source θ_i is abandoned by its employed bee and then the employed bee becomes a scout. The scout starts to search a new food source, and after finding the new source, the new position is accepted to be θ_i.

4. Experiments

In order to evaluate the performance of the ABC algorithm, some classical benchmark functions given by Krink et al. [27] are presented in Table 1. Results of ABC algorithm have been compared with the results presented by Krink et al. [27] of DE, PSO and EA. In the ABC algorithm, maximum number of cycles was taken as 1000 for $f_1(\vec{x})$ and $f_2(\vec{x})$; 5000 for $f_3(\vec{x})$, $f_4(\vec{x})$, $f_5(\vec{x})$ in order to equalize the total number of evaluation as 100,000 for the first two functions and 500,000 for the other three functions, respectively, as in ref. [27]. The percentage of onlooker bees was 50% of the colony, the employed bees were 50% of the colony and the number of scout bees was selected to be at most one for each cycle. In ABC, the number of onlooker bees is taken equal to the number of employed bees so that ABC has less control parameters. The increase in the number of scouts encourages the exploration as the increase of onlookers on a food source encourages the exploitation. The values of the control parameters of ABC algorithm used in the simulation studies and the values assigned for the control parameters of PSO, DE and EA in ref. [27] are given in Table 2. From the table, it is seen that the assigned values for DE and PSO in ref. [27] are the recommended values in the literature for the associated control parameters.

In experiments, $f_1(\vec{x})$ Schaffer function has 2 parameters, $f_2(\vec{x})$ Sphere function has 5 parameters, $f_3(\vec{x})$ Griewank, $f_4(\vec{x})$ Rastrigin and $f_5(\vec{x})$ Rosenbrock functions have 50 parameters. Parameter ranges, formulations and global optimum values of these functions are given in Table 1.

Function $f_1(\vec{x})$ is a two-dimensional Schaffer's F6 function. $\vec{x}$ is in the interval of [−100, 100]. Global minimum value for this function is 0 and optimum solution is $\vec{x}_{\text{opt}} = (x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$. Surface plot and contour lines of $f_1(\vec{x})$ are shown in Fig. 2. Function $f_2(\vec{x})$ is Sphere function that is continuous, convex and unimodal. $\vec{x}$ is in the interval of [−100, 100]. Global minimum value for this function is 0 and

Table 1
Numerical benchmark functions

Function	Ranges	Minimum value
$f_1(\vec{x}) = 0.5 + \frac{\sin^2(\sqrt{x_1^2+x_2^2}) - 0.5}{(1+0.001(x_1^2+x_2^2))^2}$	$-100 \le x_i \le 100$	$f_1(\vec{0}) = 0$
$f_2(\vec{x}) = \sum_{i=1}^{n} x_i^2$	$-100 \le x_i \le 100$	$f_2(\vec{0}) = 0$
$f_3(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{n}(x_i-100)^2\right) - \left(\prod_{i=1}^{n}\cos\left(\frac{x_i-100}{\sqrt{i}}\right)\right) + 1$	$-600 \le x_i \le 600$	$f_3(\vec{1}00) = 0$
$f_4(\vec{x}) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$-5.12 \le x_i \le 5.12$	$f_4(\vec{0}) = 0$
$f_5(\vec{x}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	$-50 \le x_i \le 50$	$f_5(\vec{1}) = 0$

Table 2
Parameter values used in the experiments

DE [26]		PSO [26]		EA [26]		ABC	
popSize	50	popSize	20	popSize	100	Colony size	100
CF	0.8	ω	$1.0 \rightarrow 0.7$	p_e	1.0	n_o	50% of the colony
f	0.5	φ_{min}	0	p_m	0.3	n_e	50% of the colony
		φ_{max}	2.0	σ_m	0.01	n_s	1
				n	10	Limit	$n_e \times D$

popSize, population size; CF, crossover factor for DE; f, scaling factor; ω, inertia weight; φ_{min}, φ_{max}, lower and upper bounds of the random velocity rule weight; p_c, crossover rate for EA; p_m, mutation rate; σ_m, mutation variance; n, elite size; n_o, onlooker number; n_e, employed bee number; n_s, scout number; D, dimension of the problem.

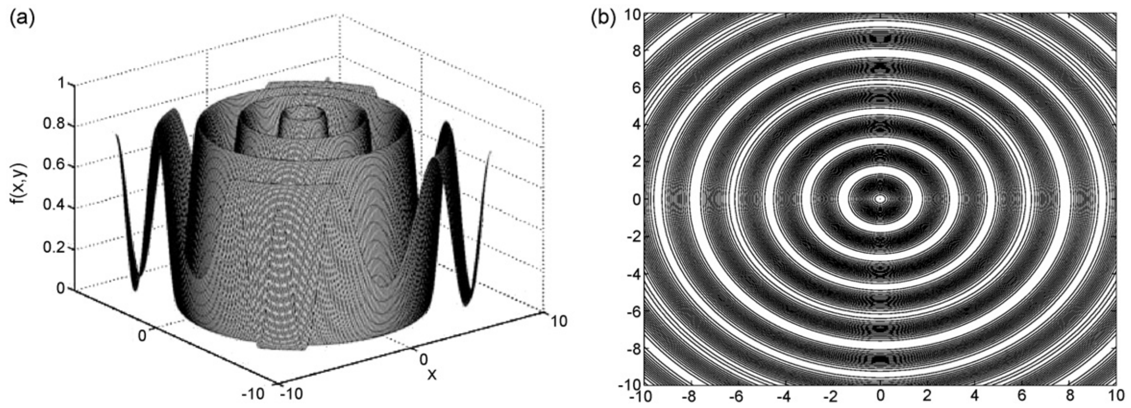

Fig. 2. Schaffer F6 function: (a) surface plot and (b) contour lines.

optimum solution is $\vec{x}_{opt} = (x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$. Surface plot and contour lines of $f_2(\vec{x})$ are shown in Fig. 3. Function $f_3(\vec{x})$ is Griewank function. $\vec{x}$ is in the interval of [−600, 600]. The global minimum value for this function is 0 and the corresponding global optimum solution is $\vec{x}_{opt} = (x_1, x_2, \ldots, x_n) = (100, 100, \ldots, 100)$. Since the number of local optima increases with the dimensionality, this function is strongly multimodal. The multimodality disappears for sufficiently high dimensionalities ($n > 30$) and the problem seems unimodal. Surface plot and contour lines of $f_3(\vec{x})$ are shown in Fig. 4. Function $f_4(\vec{x})$ is Rastrigin function. This function is based on Sphere function with the addition of cosine modulation to produce many local minima. Thus the function is multimodal. The locations of the minima are regularly distributed. The difficult part about finding optimal solutions to this function is that an optimization algorithm easily can be trapped in a local optimum on its way towards the global optimum. $\vec{x}$ is in the interval of [−5.12, 5.12]. The global minimum value for this function is 0 and the corresponding global optimum solution is $\vec{x}_{opt} = (x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$. Surface plot and contour lines of $f_4(\vec{x})$ are shown in Fig. 5. Function $f_5(\vec{x})$ is well-known classic optimization problem: Rosenbrock valley. The global optimum is inside a long, narrow, parabolic-shaped flat valley.

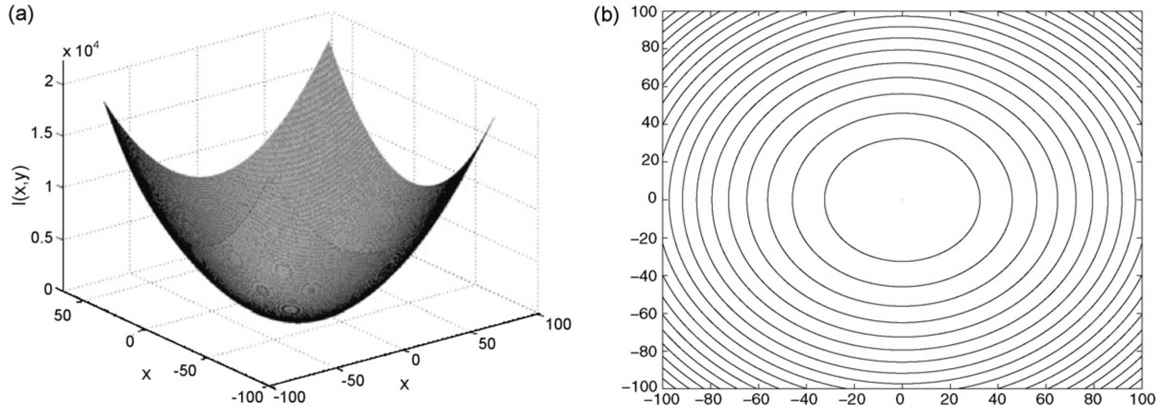

Fig. 3. Sphere function: (a) surface plot and (b) contour lines.

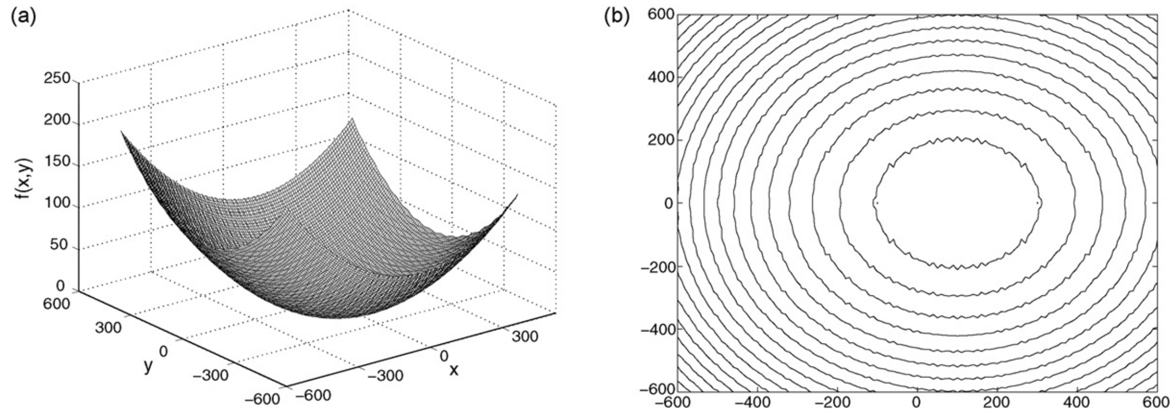

Fig. 4. Griewank function: (a) surface plot and (b) contour lines.

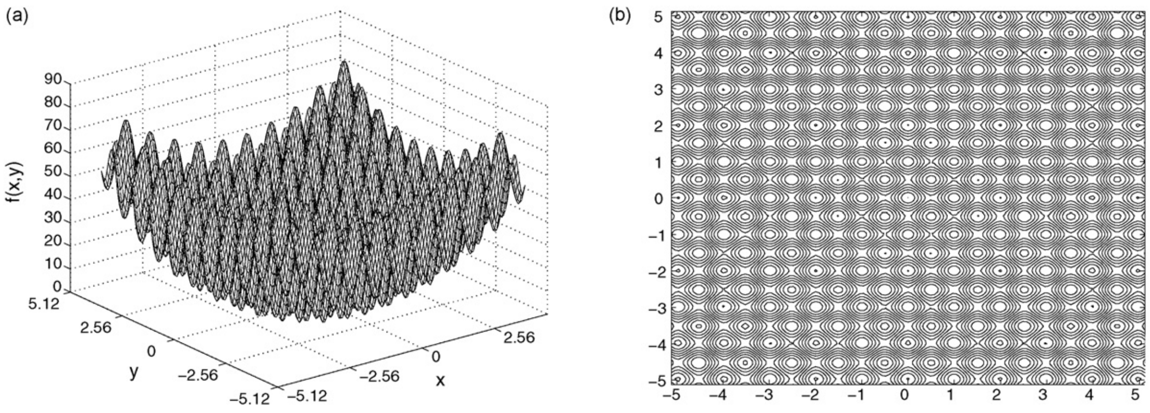

Fig. 5. Rastrigin function: (a) surface plot and (b) contour lines.

Since it is difficult to converge to the global optimum of this function, the variables are strongly dependent, and the gradients generally do not point towards the optimum, this problem is repeatedly used to test the performance of the optimization algorithms. $\vec{x}$ is in the interval of [−50, 50]. Global minimum value for this function is 0 and optimum solution is $\vec{x}_{\text{opt}} = (x_1, x_2, \ldots, x_n) = (1, 1, \ldots, 1)$. Global optimum is the only optimum, function is unimodal. Surface plot and contour lines of $f_5(\vec{x})$ are shown in Fig. 6.

5. Results and discussion

Each of the experiments was repeated 30 times with different random seeds, and the average function values of the best solutions found have been recorded. The mean and the standard deviations of the function values obtained by DE, PSO, EA [26] and ABC algorithms for under the same conditions are given in Table 3. Values less than E−12 are reported as 0. On $f_1(\vec{x})$ and $f_2(\vec{x})$ functions, DE, EA and ABC

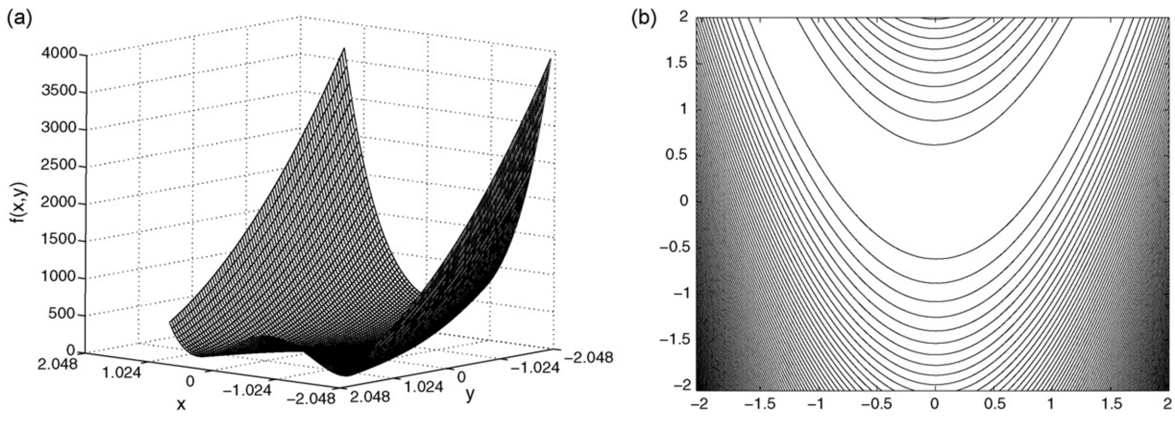

Fig. 6. Rosenbrock function: (a) surface plot and (b) contour lines.

Table 3
The results obtained by DE, PSO, EA and ABC algorithms

	DE [26]	PSO [26]	EA [26]	ABC
$f_1(\vec{x})$	0 ± 0	0.00453 ± 0.00090	0 ± 0	0 ± 0
$f_2(\vec{x})$	0 ± 0	$2.51130E{-}8 \pm 0$	0 ± 0	0 ± 0
$f_3(\vec{x})$	0 ± 0	1.54900 ± 0.06695	0.00624 ± 0.00138	0 ± 0
$f_4(\vec{x})$	0 ± 0	13.1162 ± 1.44815	32.6679 ± 1.94017	0 ± 0
$f_5(\vec{x})$	35.3176 ± 0.27444	5142.45 ± 2929.47	79.8180 ± 10.4477	$0.133109389824 \pm 0.262242170275$

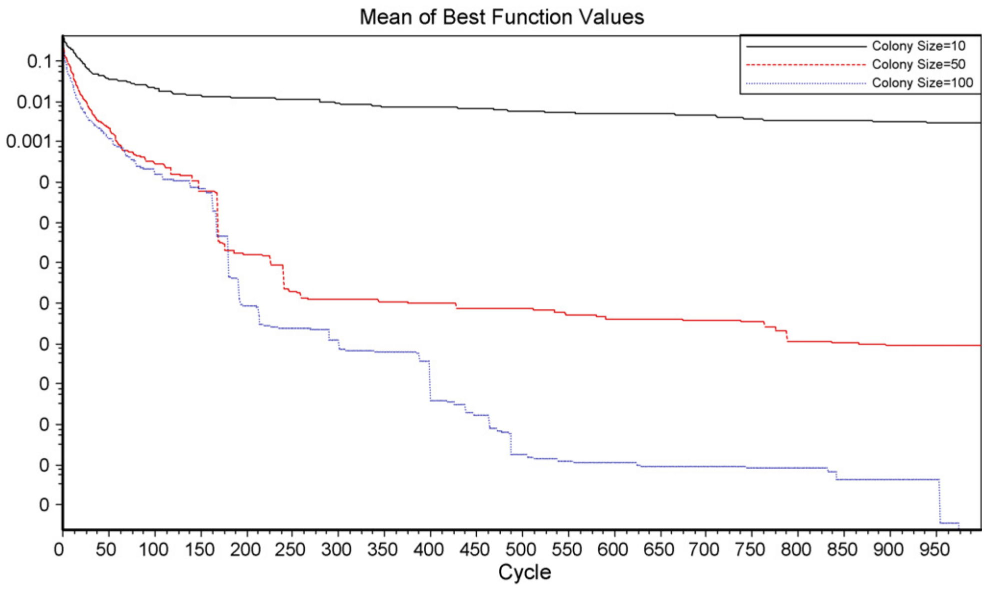

Fig. 7. Evolution of mean best values for Schaffer function ($f_1(\vec{x})$).

found the optimum value within the given cycle duration while PSO could not. On $f_3(\vec{x})$ and $f_4(\vec{x})$ functions, while DE and ABC showed equal performance and found the optimum, PSO and EA demonstrated worse performance than DE and ABC. On $f_5(\vec{x})$ function, ABC produced the best results. As seen from the results presented in Table 3, the ABC algorithm produces the best performance among the algorithms considered in the present investigation.

In order to analyse the behaviour of the ABC algorithm, it has been run with different population sizes (colony sizes) and

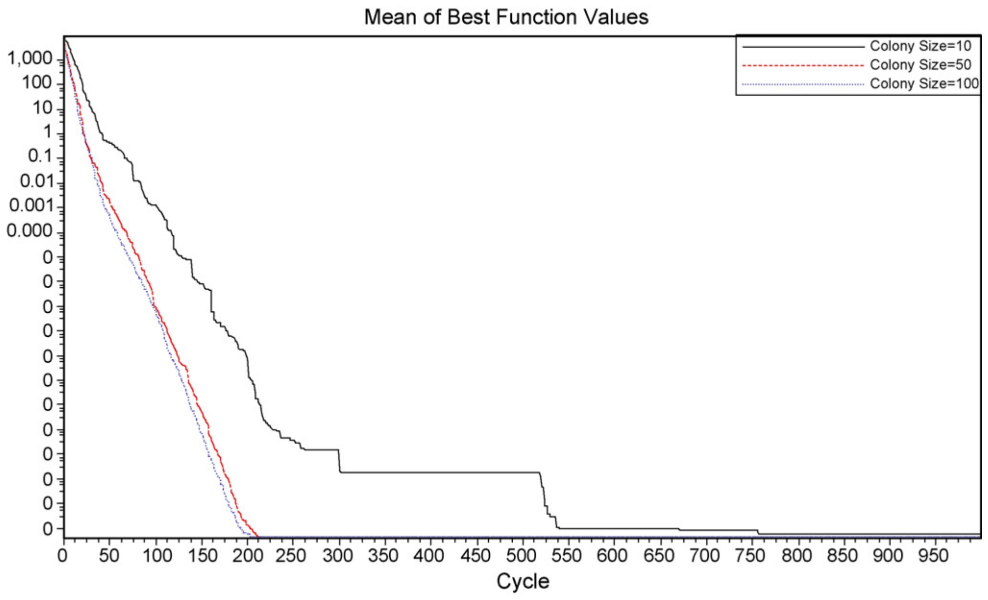

Fig. 8. Evolution of mean best values for Sphere function ($f_2(\vec{x})$).

Table 4
Mean of best function values obtained for 1000 cycle by ABC algorithm under different colony sizes

Functions	Colony size		
	10	50	100
$f_1(\vec{x})$	0.0029045	9.38E−09	2.48E−13
$f_2(\vec{x})$	5.85E−17	3.93E−17	4.30E−17
$f_3(\vec{x})$	0.0028331	1.28E−17	1.22E−17
$f_4(\vec{x})$	2.5229548	4.51E−16	4.37E−16
$f_5(\vec{x})$	9.2173464	0.159732	0.0852967

limit values. In Table 4, the mean of best function values with different colony sizes varying as 10, 50 and 100 have been presented. The progress of the mean best values presented in Table 4 has been shown in Figs. 7–11. From Table 4 and Figs. 7–12, it can be concluded that as the population size increases, the algorithm produces better results. However, after a sufficient value for colony size, any increment in the value does not improve the performance of the ABC algorithm significantly. For the test problems carried out in this work, the colony size of 50–100 can provide an acceptable convergence speed for search.

As mentioned before, the ‘‘scout bee’’ production is controlled by the control parameter ‘‘limit’’ in the ABC algorithm. There is an inverse proportionality between the value of ‘‘limit’’ and the scout production frequency. As the value of ‘‘limit’’ approaches to infinity, the total number of the scouts produced goes to zero. The results of the ABC algorithm presented in Table 3 were obtained for the colony size of 100. In order to show the effect of the scout production process on the performance of the algorithm, the average of the best function values found for the different ‘‘limit’’ values ($0.1 \times n_e \times D$, $0.5 \times n_e \times D$, $n_e \times D$ and ‘‘without scout’’) and colony sizes (20, 40 and 100) is given in Table 5. As seen from Table 5, for the multimodal functions $f_1(\vec{x})$, $f_3(\vec{x})$ and $f_4(\vec{x})$, when the scout production frequency is very high (limit value = $0.1 \times n_e \times D$) or zero (without scout), the results obtained by the ABC algorithm are worse than those produced by using the moderate values for limit, such as $0.5 \times n_e \times D$ and $n_e \times D$. For the unimodal functions $f_2(\vec{x})$ and $f_5(\vec{x})$, the production of scouts does not have any useful effect on the performance of the algorithm. However, as expected, it improves the search ability of the algorithm for the multimodal functions and its benefit becomes much clearer for the smaller colony sizes.

In ABC algorithm, while a stochastic selection scheme based on the fitness (nectar) values, which is similar to ‘‘roulette wheel selection’’ in GA, is carried out by onlooker bees, a greedy selection scheme as in DE is used by onlookers and employed bees to make a selection between the source position in their memory and the new source position. Moreover, a random selection process is carried out by scouts. Also, the neighbour source (solution) production mechanism used in ABC is similar to the mutation process, which is self-adapting, of DE. From this point of view, in DE and ABC algorithms, the solutions in the population directly affect the mutation operation since the operation is based on the difference of them. In this way, the information of a good member of the population is distributed among the other members due to the greedy selection mechanism employed. In ABC algorithm, there is no explicit crossover unlike DE and GA. However, the transfer of good information between the members is carried out by the mutation process in ABC, while this transfer is managed by the mutation and the crossover operations together in DE. Therefore, although the local converging speed of a standard DE is quite good, it might encounter the premature convergence in optimizing multimodal problems if a sufficient diversity is not provided within the initial population. In the ABC, while the intensification process is controlled by the stochastic and the greedy selection schemes, the diversification is controlled by the random

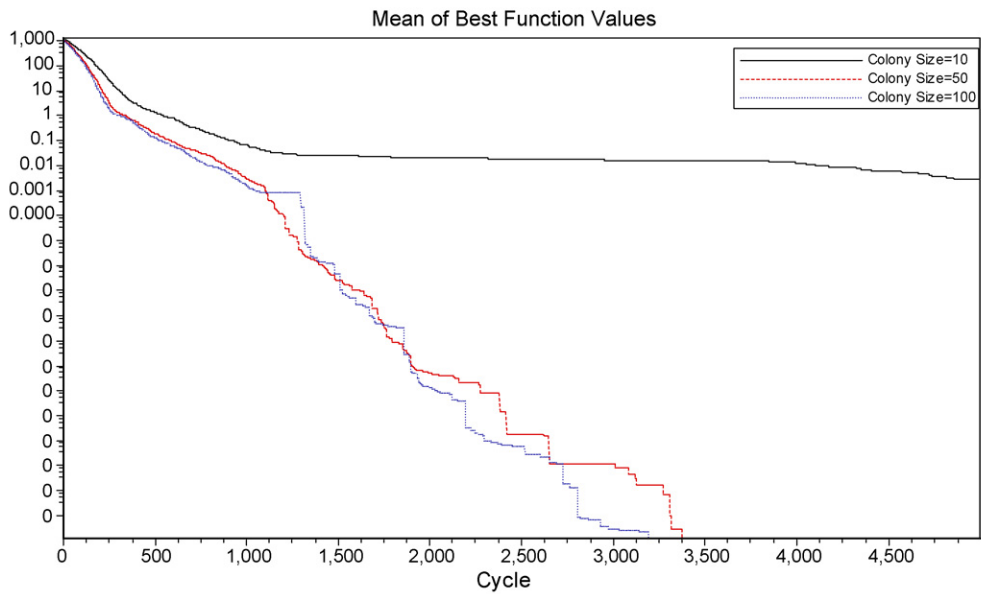

Fig. 9. Evolution of mean best values for Griewank function ($f_3(\vec{x})$).

Table 5
Effect of the ‘‘limit’’ value, which controls the scout production, on the performance of the ABC algorithm (The bold value indicates the best among the values obtained under different limit values for the same function)

Functions	Colony size	20 ($n_e = n_o = 10$)				40 ($n_e = n_o = 20$)				100 ($n_e = n_o = 50$)			
		Limit = 0.1 $\times n_e \times D$	Limit = 0.5 $\times n_e \times D$	Limit = n_e $\times D$	Without scout	Limit = 0.1 $\times n_e \times D$	Limit = 0.5 $\times n_e \times D$	Limit = n_e $\times D$	Without scout	Limit = 0.1 $\times n_e \times D$	Limit = 0.5 $\times n_e \times D$	Limit = n_e $\times D$	Without scout
$f_1(\vec{x})$ $(D = 2)$	Mean	0.0025368	0.0012855	**0.0002596**	0.0007829	0.0007739	0.0001606	**9.11E−07**	0.0001566	4.52E−05	5.25E−05	**5.80E−09**	2.97E−08
	S.D.	0.0015612	0.0016302	0.0004912	0.0007829	0.0009476	0.0003679	1.65E−06	0.0004697	0.00015	0.0002	1.78E−08	8.18E−08
$f_2(\vec{x})$ $(D = 5)$	Mean	0.30081	**3.52E−17**	**4.35E−17**	**4.07E−17**	0.0001875	**3.96E−17**	**4.17E−17**	**4.14E−17**	**9.38E−17**	**5.10E−17**	**7.28E−17**	**6.76E−17**
	S.D.	0.2334196	1.21E−17	1.01E−17	1.15E−17	0.0001494	1.18E−17	1.11E−17	1.01E−17	8.02E−17	2.99E−17	4.83E−17	4.92E−17
$f_3(\vec{x})$ $(D = 50)$	Mean	0.0004213	0.0004198	**0.0003442**	0.0041657	0.0002479	1.00E−11	**3.57E−12**	0.0009854	**0.0002758**	0.0003329	0.0006321	0.0013274
	S.D.	0.0022681	0.0022557	0.0018068	0.0012934	0.0013348	4.06E−11	1.36E−11	0.0037412	0.0005389	0.0006839	0.0022135	0.004603
$f_4(\vec{x})$ $(D = 50)$	Mean	0.1283467	0.0025718	**3.58E−14**	0.0994959	0.0331656	8.39E−05	**3.41E−07**	3.58E−5	5.1634613	5.4318354	**5.0961036**	5.4057093
	S.D.	0.3139938	0.0138492	1.84E−13	0.2984877	0.1786006	0.0004015	1.51E−06	1.09E−09	1.9603541	1.9868015	1.9478082	2.1876648
$f_5(\vec{x})$ $(D = 50)$	Mean	3.7243047	**2.4621987**	6.298043	44.237393	1.8110815	1.8449032	1.0749639	**0.0089114**	54.291323	53.33046	**45.981747**	46.032145
	S.D.	4.2432528	5.1234923	7.0465668	220.39974	2.3644827	2.2459113	1.1727714	0.043152	30.940497	26.549226	36.328997	27.775953

n_e, Number of employed bees; D, dimension of the problem; runs = 30; total evaluation number = 20,000 for $f_1(\vec{x})$ and $f_2(\vec{x})$; 100,000 for $f_3(\vec{x})$, $f_4(\vec{x})$ and $f_5(\vec{x})$.

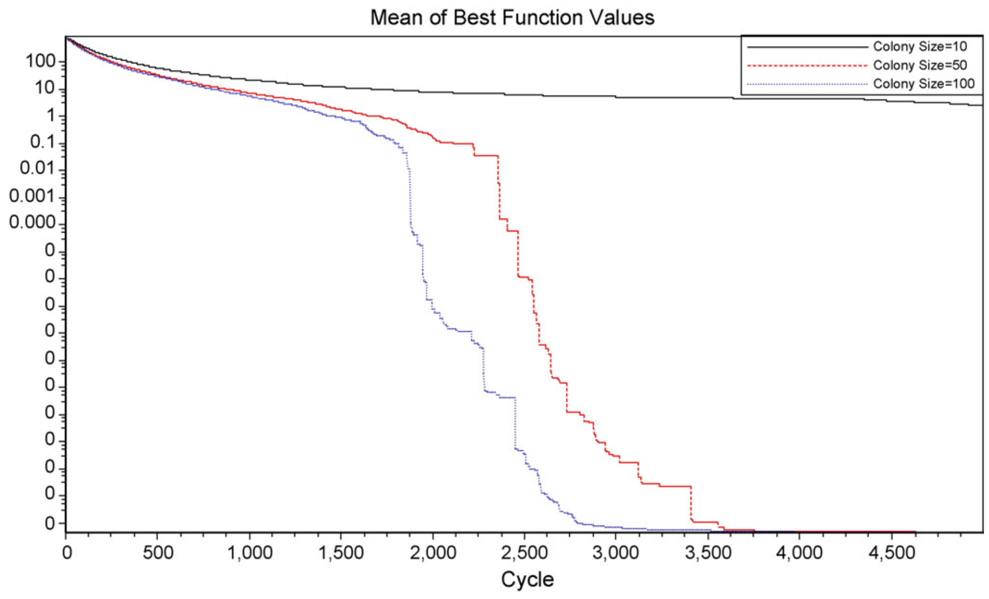

Fig. 10. Evolution of mean best values for Rastrigin function ($f_4(\vec{x})$).

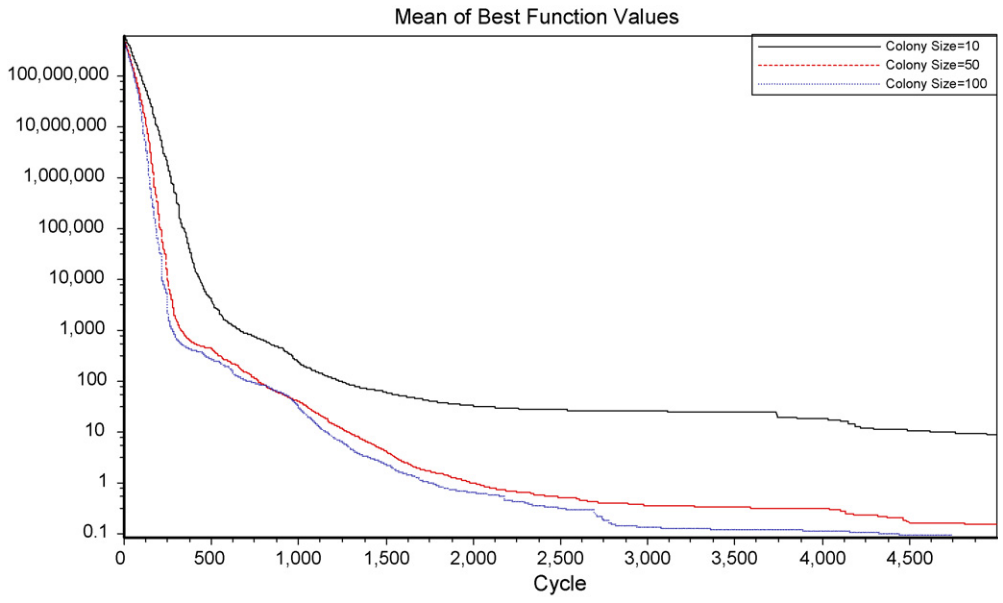

Fig. 11. Evolution of mean best values for Rosenbrock function ($f_5(\vec{x})$).

selection. The performance of ABC is very good in terms of the local and the global optimization due to the selection schemes employed and the neighbour production mechanism used. Consequently, the simulation results show that the ABC algorithm, which is flexible and simple to use and robust optimization algorithm, can be used efficiently in the optimization of multimodal and multi-variable problems.

6. Conclusion

In the present investigation, the performance of the ABC algorithm has been compared with that of differential evolution, particle swarm optimization and evolutionary algorithm for multi-dimensional and multimodal numeric problems. The behaviour of ABC algorithm under different control parameter values has also been analysed. Simulation results show that ABC algorithm performs better than the mentioned algorithms and can be efficiently employed to solve the multimodal engineering problems with high dimensionality.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.asoc.2007.05.007.

References

[1] K.V. Price, R.M. Storn, J.A. Lampinen (Eds.), Differential Evolution: A Practical Approach to Global Optimization, Springer Natural Computing Series, 2005.

[2] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[3] N. Chakraborti, A. Kumar, The optimal scheduling of a reversing strip mill: studies using multi-population genetic algorithms and differential evolution, Mater. Manuf. Processes 18 (2003) 433–445.

[4] N. Chakraborti, P. Mishra, S. Erkoç, A study of the Cu clusters using gray-coded genetic algorithms and differential evolution, 2004, J. Phase Equilib. Diffus. 25 (2004) 16–21.

[5] R. Storn, Differential evolution design of an IIR-Filter with requirements of magnitude and group delay, in: Proceedings of the IEEE Conference on Evolutionary Computation, 1996, pp. 268–273.

[6] R. Storn, System design by constraint adaptation and differential evolution, IEEE Trans. Evol. Comput. 3 (1999) 22–34.

[7] D. Karaboga, B. Basturk, Image segmentation using differential evolution algorithm, in: Conference of 13th IEEE SIU' 2005, Kayseri, (2005), pp. 33–36.

[8] N. Karaboga, Digital IIR filter design using differential evolution algorithm, Eurasip J. Appl. Signal Process. 8 (2005) 1–9.

[9] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Intelligence, NY: Oxford University Press, NewYork, 1999.

[10] L.N. De Castro, F.J. Von Zuben, Artificial Immune Systems, Part I. Basic Theory And Applications, Technical Report No. Rt Dca 01/99, Feec/Unicamp, Brazil, 1999.

[11] J. Vesterstrøm, J. Riget, Particle swarms extensions for improved local, multimodal and dynamic search in numerical optimization, M.Sc. Thesis, May 2002.

[12] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the 1995 IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.

[13] V. Tereshko, Reaction-diffusion model of a honeybee colony's foraging behaviour, in: M. Schoenauer, et al. (Eds.), Parallel Problem Solving from Nature VI, Lecture Notes in Computer Science, vol. 1917, Springer-Verlag, Berlin, 2000, pp. 807–816.

[14] V. Tereshko, T. Lee, How information mapping patterns determine foraging behaviour of a honey bee colony, Open Syst. Inf. Dyn. 9 (2002) 181–193.

[15] V. Tereshko, A. Loengarov, Collective decision-making in honey bee foraging dynamics, Comput. Inf. Syst. J. 1352-94049 (2005).

[16] D. Teodorovič, Transport modeling by multi-agent systems: a swarm intellgence approach, Transport. Plann. Technol. 26-4 (August) (2003) 289–312.

[17] P. Lucic, D. Teodorovič, Transportation Modeling: An Artificial Life Approach, ICTAI, 2002, pp. 216–223.

[18] D. Teodorovič, M. Dell'orco, Bee colony optimisation—a cooperative learning approach to complex transportation problems, in: 10th EWGT Meeting, Poznan, 13–16 September, 2005.

[19] K. Benatchba, L. Admane, M. Koudıl, Using bees to solve data-mining problem expressed as a max-sat one, artificial intelligence and knowledge engineering applications: a bioinspired approach, in: First International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2005, Las Palmas, Canary Islands, Spain, June 15–18, 2005.

[20] H.F. Wedde, M. Farooq, Y. Zhang, BeeHive: an efficient fault-tolerant routing algorithm inspired by honey bee behavior, ant colony, optimization and swarm intelligence, in: 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5–8, 2004.

[21] X.S. Yang, Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms, Lecture Notes in Computer Science, 3562, Springer-Verlag GmbH, 2005,, p. 317.

[22] D. Karaboga, An Idea Based On Honey Bee Swarm For Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[23] B. Basturk, D. Karaboga, An artificial bee colony (ABC) algorithm for numeric function optimization, in: IEEE Swarm Intelligence Symposium 2006, May 12–14, Indianapolis, IN, USA, 2006.

[24] R.L. Jeanne, The evolution of the organization of work in social insects, Monit. Zool. Ital. 20 (1986) 267–287.

[25] T.D. Seeley, The Wisdom of the Hive, Harvard University Press, Cambridge, MA, 1995.

[26] D.E. Goldberg, Genetic Algorithms in Search, in: Optimization and Machine Learning, Addison-Wesley Pub. Co., 1989, , ISBN: 0201157675.

[27] T. Krink, B. Filipic, G.B. Fogel, R. Thomsen, Noisy optimization problems—a particular challenge for differential evolution? in: Proceedings of 2004 Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, 2004, pp. 332–339.