An Efficient Multi Population Artificial Bee Colony

Vahid Zeighami, Reza Akbari, and Koorush Ziarati

***Abstract*— Artificial Bee Colony is an optimization algorithm that can be applied on a wide range of engineering problems. In this work, the standard ABC is extended by incorporating cooperative behaviors and an efficient algorithm called multi population ABC (or MPABC) is developed. MPABC aims at improving the performance of the standard ABC algorithm using benefits of cooperation as a social behavior. MPABC works by employing multiple populations that concurrently optimize the solution vector. Cooperation is obtained by sharing information between populations. The proposed algorithm was tested on a set of well known test functions. The results showed that the proposed algorithm is efficient, robust, produce good results, and outperforms other algorithms investigated in this paper.**

***Index Terms*—Artificial bee colony; multi population artificial bee colony; cooperative behaviors.**

I. Introduction

The unconstrained optimization problem is considered in this work. In recent years, many population based optimization techniques such as genetic algorithms [1], particle swarm optimization [2], and bee algorithms [3] have been developed to solve unconstrained optimization problems.

Genetic algorithms and particle swarm optimization are among most popular optimization techniques which have been applied on a wide range engineering problems. The algorithms which are inspired from intelligent behaviors (e.g. foraging or mating) of honey bees are among the newest optimization techniques and a few algorithms based on foraging behaviors of honey bees were presented in literatures for unconstrained optimization [3]-[7].

The idea of using cooperation in population based algorithms was first introduced in GA by Potter and De Jong [8] for optimizing numerical functions. In their model, the solution vector is partitioned to the two or more smaller vectors, and then each of these partial vectors is optimized using a separate population. Cooperation is obtained by assembling complete solution from the partial solutions. After that, this idea is used in PSOs, and a set of PSO algorithms, so called Cooperative PSO was emerged. Van den Berg applied the Potter technique on standard PSO and proposed three models of cooperative PSO, called CPSO-S, CPSO-S_k, and CPSO-H_k, in [10]-[11]. In CPSO-S_k method, the solution vector is divided to the k sub-vectors, and k swarms were used to optimize these sub-vectors. The CPSO-S method is a type of CPSO-S_k in which the complete D-dimensional solution vector is partitioned to the D one-dimensional vectors, and finally CPSO-H_k combines standard PSO and CPSO-S_k. They showed that cooperation between particle swarms provides a way to obtain better performance. Different approach for cooperative PSO such as MCPSO, CONPSO were presented in [12]-[14], where the solution vector is not partitioned, but multiple cooperative populations were employed. Usually, in these types of methods, cooperation is obtained through exchanging information about global best individuals [15]. Recently, the cooperative mechanism was used by Akbari and Ziarati [7] for designing cooperative variants of bee algorithms. They proposed three variants of cooperative bee algorithms called S-CBSO, MS-CBSO, and ML-CBSO.

In this work, a Cooperative artificial bee colony is proposed. The proposed algorithm uses the concepts presented in [3] for optimizing numerical function. This Cooperative approach employs PN populations in which each of the populations optimizes the complete solution vector. The cooperation is obtained by sharing information between populations.

The paper organized as follows. Description of the proposed algorithm is presented in section 2. Section 3 reports experimental analysis on the proposed algorithm. Finally, section 4 concludes this work.

II. Multi Population Artificial Bee Colony

This section presents the detailed description of MPABC algorithm. MPABC algorithm is designed based on standard ABC which has been introduced by Karaboga and Bastruk in [3]. Fig. 1 shows MPABC algorithm in pseudocode. MPABC employs PN populations which concurrently optimize the solution vector. Cooperation is obtained by sharing information between populations. Similar to ABC, each population in MPABC employs three types of bees. A bee waiting on the dance area for making decision to choose a food source is called onlooker; the bee going to the food source visited by herself just before is named as employed bee, and the bee who fly sponentanously is called scout.

MPABC employs PN populations of bees in four phases. At the first phase, the initialization phase, the MPABC generates PN randomly distributed initial populations. Each initial population contains SN/PN solutions, where SN/PN denotes the size of employed or onlooker bees. Each solution i in a population Pop_j represents a D-dimensional vector

Manuscript received May 12, 2012, revised May 30, 2012.

Vahid Zeighami is with Department of Mathematics, Shiraz University, Shiraz, Iran (e-mail: vahid.zeighami@gmail.com).

Reza Akbari is with Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran (e-mail: akbari@sutech.ac.ir).

Koorush Ziarati is with Department of Computer Science and Engineering and Information Technology, Shiraz University, Shiraz, Iran (e-mail: ziarati@shirazu.ac.ir).

$\vec{x}_{j,i}, i = 1,2,..., SN/PN$ and $j = 1,2,..., PN$. After initialization, the populations are evaluated and the fitnesses of their individuals are calculated.

The second phase starts the main body of algorithm. This phase is iterated for each population Pop_j . In this phase, each population evolves based on its own local information. This phase has three steps. As stated before, at the initialization phase of each population, a set of food sources is randomly selected by the bees and their nectar amounts are determined. At the first step of each cycle, the employers of population Pop_j come into the hive and share the nectar information of the sources with the onlookers waiting on the dance area. After sharing their information with onlookers, every employed bee goes to the food source area visited by herself at the previous cycle since that food source exists in her memory, and then chooses a new food source by means of visual information in the neighborhood of the one in her memory and evaluates its nectar amount.

At the second step, considering the information shared by employers in the dance area, an artificial onlooker bee in population Pop_j chooses a food source based on the probability value associated with that food source. The probability, $P_{j,i}$, is calculated by the following expression:

$$p_{j,i} = \frac{fit_{j,i}}{\sum_{n=1}^{SN/PN} fit_n} \quad (1)$$

where $fit_{j,i}$ is the fitness value of the *i*-th solution of population Pop_j which is proportional to the nectar amount of the food source in the position *i* and SN/PN is the number of food sources which is equal to the number of employed or onlooker bees in population Pop_j . An onlooker prefers a food source area depending on the nectar information distributed by the employed bees on the dance area. The probability of a food source selection increases as its nectar amount increases.

After an onlooker arrives at a promising area, she chooses a new food source in the neighborhood of the one in the memory depending on visual information as in the case of employed bees. The new food position is selected based on the following expression:

$$v_{j,i}^d = x_{j,i}^d + \phi_{j,i}^d \left(x_{j,i}^d - x_{j,k}^d \right) \quad (2)$$

where $k \in \{1,2,..., SN/PN\}$ and $d \in \{1,2,..., D\}$ are randomly chosen indexes. Although *k* is determined randomly, it has to be different from *i*. $\phi_{j,i}^d$ is a random number between [-1,1]. It controls the production of neighbor food sources around $x_{j,i}^d$ and represents the comparison of two food positions visually by a bee. If a parameter value produced by this operation exceeds its predetermined limit, the parameter can be set to an acceptable value.

At the third step, the nectar amount of the food sources are evaluated; the scout bees are determined and they are sent onto the possible new food sources. For this purpose, the positions are evaluated, and if a food source cannot be improved after a predetermined number of iterations (called *limit*), then the corresponding food source is abandoned. The limit parameter is determined manually. The abandoned food source is replaced with the new one found by the scouts. A scout produces a new position randomly and replaces the abandoned food source if the new food source has better nectar.

Function MPABC (*PN, Max_Iter, limit, SN***)**
- **For** *j*=1 to *PN*
  - **Initialize** the population Pop$_j$ of solutions $x_{j,i}, i = 1,2,..., SN/PN$
  - **Evaluate** population Pop$_j$
- **End For**
- **For** *iter*=1 to *Max_Iter*
  - **For** *j*=1 to *PN*
    - **Step 1) Produce** new solutions $v_{j,i}$ for the employed bees using (2) and evaluate them
      Apply the greedy selection process for the employed bees
    - **Step 2) Calculate** the probability values $P_{j,i}$ for the solutions $x_{j,i}$ by (1)
      Produce the new solutions $v_{j,i}$ for the onlookers from the solutions $x_{j,i}$ selected depending on $P_{j,i}$ and evaluate them
      Apply the greedy selection process for the onlookers
    - **Step 3) Determine** the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution $x_{j,i}$ by (3)
      Memorize the best solution achieved so far
  - **End For**
  - **Step1) Select** best solutions b_j from all the populations Pop$_j$, $1 \le j \le PN$
  - **Step2) For** *j*=1 to *PN*
    - **Produce** new solutions $v_{j,i}$ for the employed bees using (4) and evaluate them
    - **Apply** the greedy selection process for the employed bees
    - **End For**
  - **End For**
- **Return** best solution

Fig. 1. Pseudo code of standard MPABC algorithm

Assume that the abandoned source in population Pop_j is $x_{j,i}$ and $d \in \{1,2,..., D\}$, then the scout discovers a new food source to be replaced with x_i . This operation can be defined as follows:

$$x_{j,i}^d = x_{j,\min}^d + rand[0,1]\left(x_{j,\max}^d - x_{j,\min}^d \right) \quad (3)$$

After each candidate source position $v_{j,i}^d$ is produced and then evaluated by the artificial bee, its performance is compared with that of its old one. If the new food source has equal or better nectar than the old source, it will be replaced with the old one in the memory. Otherwise, the old one is retained in the memory. In other words, a greedy selection mechanism is employed as the selection operation between the old and the candidate one.

The cooperation among populations is performed in the

third phase. The cooperation is obtained by sharing best solutions found by the cooperative populations. This phase has two main steps. At the first step, each population Pop_j is evaluated and the best solution (i.e. $\vec{b}_j$) which is found by its individuals is determined. Next, these solutions are added to the elite list $B=\{b_1,b_2,...,b_{PN}\}$. After selecting the best solution, we start to update the food positions in each population Pop_j at the second step. The new food position is produced using the following expression:

$$\vec{v}_{j,i}=\vec{x}_{j,i}+\sum_{m=1}^{PN}\vec{\phi}_{j,i}\left(\vec{x}_{j,i}-\vec{b}_m\right) \quad (4)$$

where $\vec{b}_m$ is the best solution which is found by population Pop_m. The candidate food source is updated if the newly proposed position has better nectar.

The aforementioned processes are repeated through a predetermined number of iterations called *Max_Iter* or until a termination criterion is satisfied. The fourth phase terminates the algorithm and returns the best solution which is found by the cooperative populations.

III. Experimens

In this section, the experiments which have been conducted to evaluate the performance of the proposed algorithm for a number of analytical benchmark functions are described. The standard and cooperative optimization algorithms based on intelligent behaviors of bees were selected to show the performance of the proposed algorithm. The performances of MPABC algorithm are evaluated in comparison with standard ABC algorithm, and two cooperative variants of bee swarm optimization algorithm proposed in [7].

A. Experimental Settings

We use five well-known numerical functions to evaluate the performance of MPABC algorithm in terms of optimum solution after a predefined number of function evaluations and convergence speed. The analytical test functions and their parameters are presented in TABLE I and TABLE II. The first two functions are unimodal, and others are multimodal. A unimodal function has only one optimum while a multimodal function has two or more local optima. We select these test function as each of them is a candidate for a different class of real-world problems.

B. Settings of the Algorithms

The performance of the new method is compared with the performance of the standard artificial bee colony (ABC) [3], and two other cooperative bee algorithm called split bee swarm optimization (S_CBSO) [7], multi swarm bee swarm optimization (MS_CBSO) [7]. The S_CBSO is a cooperative bee approach that optimizes by decomposing the problem into the several sub-problems and solving each of the sub-problems by a distinct swarm. MS_CBSO is a cooperative approach optimizing a problem by exchanging information between populations without decomposing the problem into the sub-problems.

For the MPABC, five populations are used that cooperatively optimize the problem at hand. The cooperation is performed by sharing information among populations at each cycle. Each population executes a distinct copy of standard ABC algorithm. In S_CBSO, the solution vector is split into five parts. In MS_CBSO, five swarms are used to optimize the problem. Each swarm in both of the S_CBSO, and MS_BSO partitions its individuals as scouts (10%), onlookers (45%), and experienced foragers (45%). All experiments were run for 30 error function evaluations. The number of function evaluations is set at 2.0E+6.

TABLE I: The Mathematical Formulas of Test Functions

Func.	Formula
Sph.	$f_1(x)=\sum_{i=1}^{n}x_i^2$
Ros.	$f_2(x)=\sum_{i=1}^{n-1}\left(100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right)$
Ras.	$f_3(x)=\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)+10\right)$
Gri.	$f_5(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$
Ack.	$f_6(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{1}^{n}\cos(2\pi x_i)\right)+20+e$

TABLE II: The Parameters of Test Functions.

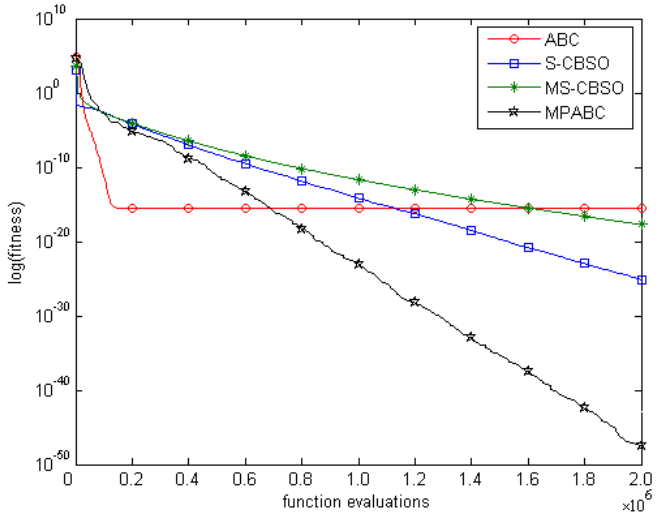
Function	Domain	Initial Range	Optimum	Threshold
Sph.	[-100,100]n	[50,100]n	(0,0,…,0)	0.01
Ros.	[-30,30]n	[15,30]n	(1,1,…,1)	0.1
Ras.	[-5.12,5.12]n	[2.56,5.12]n	(0,0,…,0)	100
Gri.	[-600,600]n	[300,600]n	(0,0,…,0)	0.01
Ack.	[-30,30]n	[15,30]n	(0,0,…,0)	0.01

C. Experimental Results

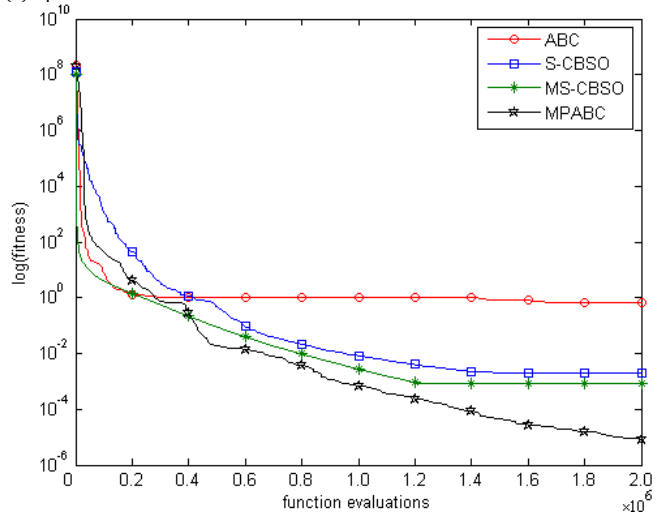
In the experiments the number of iterations to reach a predefined threshold was specified for each function. Different successful criteria for different functions are presented in the literatures. The success criterion for Rosenbrock and Rastrigrin is set at 0.1 and 100 respectively. The success criterion for the other test functions is set at 0.001. After the final iteration, if the minimum value was reached by the algorithm was not below the success criteria, the run was considered unsuccessful.

1) Performance Analysis

TABLE III presents mean, standard deviation (Stdv), and required function evaluations before convergence (C. Speed) of the algorithms on the unimodal test functions. The mean and standard deviation show quality of the results obtained by each algorithm, and the required function evaluations presents the convergence speed of the algorithms. To ease of observation, the best results obtained by the algorithms are shown in bold.



(a) Sphere



(b) Rosenbrock

Fig. 2. Evolution of average fitness on unimodal test functions.

The unimodal function Sphere shows an easy problem to solve. As can be seen from Fig. 2(a), all the algorithms successfully optimize the sphere function. The best result obtained by MPABC algorithm. The ABC has the fastest speed in converging to the success criterion. Rosenbrock function is a unimodal function that can be used to evaluate the ability of an algorithm in mitigating stagnation problem. The Rosenbrock function is a hard problem to solve. Fig. 2(b) shows that the Rosenbrock function is easily optimized by MPABC algorithms while the ABC has problem in converging to the success criteria. Also, MPABC obtains the fastest convergence speed over the Rosenbrock function.

TABLE IV presents means, standard deviations, and convergence speeds over the multimodal test functions. Also, the convergence behaviors of the algorithms are given in Fig. 3. The Rastrigrin function is a highly multimodal with frequent local optima. An algorithm with poor balance between exploration and exploitation simply trapped in local optima in early iterations. The best result is obtained by the MPABC and S_CBSO algorithm. The S_CBSO algorithm has the fastest convergence speed.

The ABC and MS-CBSO have competitive performance over the Griewank function. However, the best result is obtained by the S_CBSO and MPABC algorithms. The ABC algorithm has the fastest convergence speed over Griewank test function.

TABLE III: MEAN, STANDARD DEVIATION (STDV.), CONVERGENCE SPEED (C. SPEED) ON THE SPHERE FUNCTION

Function	Method	Mean(Stdv.)	C. Speed
Sphere	ABC	2.52E-16 ± (3.66E-17)	2.90E+04
	S-CBSO	7.32E-26 ± (9.54E-27)	4.85E+04
	MS-CBSO	1.94E-18 ± (5.67E-19)	6.45E+04
	MPABC	3.62E-48 ± (7.96E-49)	7.92E+04
Rosenbrock	ABC	6.80E-01 ± (7.23E-01)	1.70E+06
	S-CBSO	1.91E-03 ± (3.95E-04)	5.99E+05
	MS-CBSO	8.49E-04 ± (9.31E-04)	4.87E+05
	MPABC	8.59E-06 ± (1.05E-05)	4.29E+05

From Fig. 3(c), we can see that ABC algorithm rapidly converges to the success criteria when optimizing the Ackley function. The MPABC and ABC algorithms obtain better performance compared to the S_CBSO and MS_CBSO algorithms. However, the MPABC algorithm surpasses its standard version. In general, we can see that MPABC provide efficiency in optimizing numerical functions.

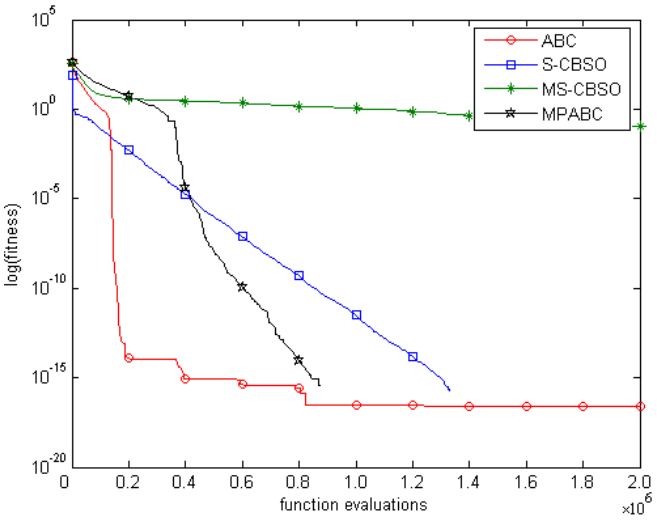
TABLE IV: MEAN, STANDARD DEVIATION (STDV.), CONVERGENCE SPEED (C. SPEED) ON THE MULTIMODAL TEST FUNCTIONS

Function	Method	Mean(Stdv.)	C. Speed
Rastrigrin	ABC	2.59E-17 ± (3.34E-18)	4.80E+04
	S-CBSO	0.00E+00 ± (0.00E+00)	1.50E+03
	MS-CBSO	1.12E-01 ± (2.35E-01)	2.10E+04
	MPABC	0.00E+00 ± (0.00E+00)	4.10E+04
Griewank	ABC	2.56E-16 ± (2.68E-17)	2.30E+04
	S-CBSO	0.00E+00 ± (0.00E+00)	3.44E+05
	MS-CBSO	7.94E-16 ± (8.59E-17)	3.46E+05
	MPABC	0.00E+00 ± (0.00E+00)	2.14E+05
Ackley	ABC	3.75E-14 ± (4.49E-15)	6.21E+04
	S-CBSO	2.83E-12 ± (5.36E-13)	3.34E+05
	MS-CBSO	4.13E-09 ± (7.19E-10)	2.78E+05
	MPABC	2.99E-14 ± (2.33E-15)	3.03E+05

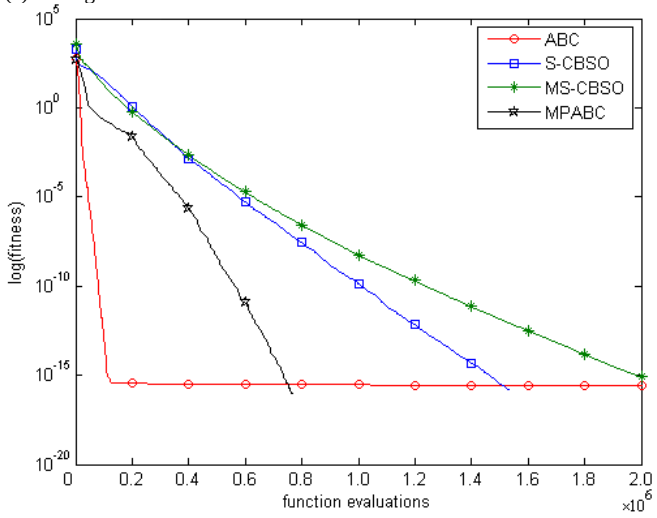
IV. CONCLUSION

The optimization algorithms which are inspired from intelligent behavior of honey bees are among the most recently introduced population based algorithms. In this paper, we have incorporated the cooperative behaviors into the standard ABC algorithm and a cooperative variant has been proposed. The MPABC algorithm works by employing a number of populations that concurrently optimize the solution vector. The cooperation is obtained by sharing information between cooperative populations.

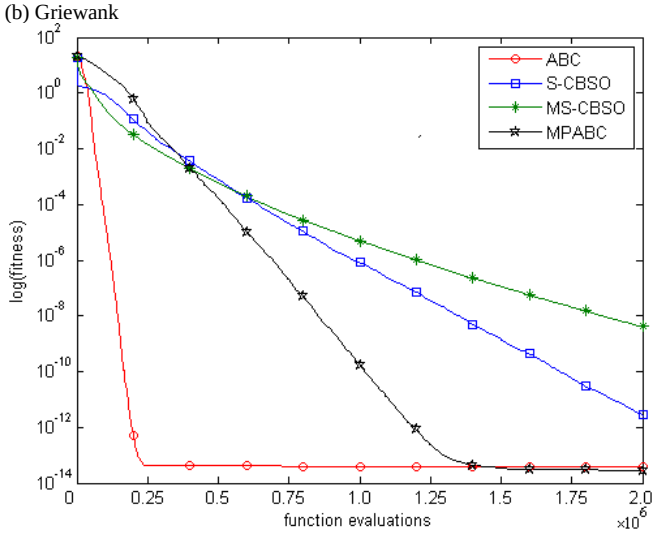
The proposed multi population ABC algorithm was compared with standard ABC and two other cooperative bee algorithms. The experimental results showed that the proposed algorithm is highly reliable and outperforms other algorithms investigated in this paper.



(a) Rastrigrin



(b) Griewank



(c) Ackley

Fig. 3. Evolution of average fitness on multimodal test functions.

REFERENCES

[1] Srinivasan D. and Seow T. H., 2003, "Evolutionary Computation", CEC '03, 8–12, Canberra, Australia, 2292–2297.

[2] Kennedy J., and Eberhart R., 1995, "Particle swarm optimization," in *Proceeding of IEEE Int. Conf. Neural Networks*, 4, 1942–1947.

[3] Karaboga, and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", in *Journal of Global Optimization*, 39, 2007, pp. 459–471.

[4] D. Karaboga, B. Basturk, "On the performance of artificial bee colony (ABC) algorithm", in *Journal of Applied Soft Computing*, Vol. 8, pp. 687–697, 2008.

[5] D. Karaboga, B. Akay, "A comparative study of Artificial Bee Colony algorithm", in *Journal of Applied Mathematics and Computation*, 2009.

[6] Akbari R., Mohammadi M., Ziarati K., "A novel bee swarm optimization algorithm for numerical function optimization", In: *Journal of Communications in Nonlinear Science and Numerical Simulation* 15(2010) 3142-3155.

[7] Akbari R., Ziarati K., "A novel bee swarm optimization algorithm and its cooperative variants for numerical function optimization", *applied soft computing Journal, doi: 10.1016/j.asoc.2010.08.009*.

[8] Potter M. A. and De Jong K. A., 1994, "A cooperative coevolutionary approach to function optimization," in *the Third Parallel Problem Solving from Nature*. Berlin, Germany: Springer-Verlag, 249–257.

[9] M. A. Potter, and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *the Third Parallel Problem Solving from Nature*. Berlin, Germany: Springer-Verlag, 1994, pp. 249–257.

[10] F. Van Den Bergh, and A. P. Engelbrecht, "Cooperative learning in neural networks using particle swarm optimizers," *South African Comput. J.*, 26, 2000, pp. 84–90.

[11] F. Van Den Bergh, and A. P. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization", in *IEEE Transactions on Evolutionary Computation*, 8(3), 2004, pp. 225-239.

[12] S. Baskar, and P. N. Suganthan, "A novel concurrent particle swarm optimization," in *Proc. of IEEE Congress on Evolutionary Computation*, 1, 2004, pp. 792–796.

[13] M. Belal, and T. El-Ghazawi, "Parallel models for particle swarm optimizers," in *International Journal on Intelligent Cooperative Information Systems*, 4(1), 2004, pp. 100–111.

[14] B. Niu, Y. Zhu, X. He, H. Wu, "MCPSO: A multi-swarm cooperative particle swarm optimizer", *In Journal of Applied Mathematics and Computation*, 185, 2007, pp. 1050–1062.

[15] M. El-Abd, and M. S. Kamel, "A Taxonomy of Cooperative Particle Swarm Optimizers", in *International Journal of Computational Intelligence Research*, 4(2), 2008, pp. 137–144.

Vahid Zeighami received his B.S. in Applied Mathematics from Shiraz University in 2008, and his M.Sc. in Operations Research from Shiraz University in 2011. His areas of research include Meta-heuristic Algorithms, optimization problems, operation research, integer programming, linear and non-linear optimization.

Reza Akbari received his M.Sc. in artificial intelligence from Isfahan University of Technology (2006). He has received his Ph.D. from Shiraz University (2011). He is an Assistant Professor at the department of computer engineering and information technology of the Shiraz University of Technology. His areas of research include evolutionary computation, engineering optimization, and search based software engineering.

Koorush Ziarati graduated in electronics Engineering (control), from University of Science & Technologies Houari Boumedienne in Algeria (1990). He obtained his M.Sc. degree in control-Robotics and his Ph.D. in Operations Research from Ecole Polytechnique de Montreal, Canada in 1993 and 1997, respectively. He has been a faculty member of Department of Computer Science & Engineering and Information Technology in Shiraz University since 1999. Currently his research concern transportation, assignment and Scheduling problem, Heuristic methods for global optimization problem, integer and convex programming.