Best Supported Emigrant Creation For Parallel Implementation Of Artificial Bee Colony Algorithm

Dervis KARABOGA[1] and Selcuk ASLAN[2]

[1]Department of Computer Engineering, Erciyes University, Kayeri, Turkey
[2]Department of Computer Engineering, Ondokuz Mayis University, Samsun, Turkey
karaboga@erciyes.edu.tr, selcuk.aslan@bil.omu.edu.tr

Abstract: *Artificial Bee Colony algorithm inspired by the foraging behavior of real honey bees is one of the most popular swarm intelligence based optimization techniques. Like other population based evolutionary computation approaches, Artificial Bee Colony algorithm is intrinsically suitable for distributed architectures. However, determining which food source should be chosen to distribute between subcolonies and communication topology applied still remain as an important problem for parallel implementations. In this study, a new schema for increasing the quality of the distributed source by changing best solution is presented. The proposed model is adapted to ring migration topology and its effectivenes is compared with conventional ring based topology in which best food sources in each subpopulation are distributed and the original sequential counterpart. Comparative results show that the proposed model increased the quality of the solutions and early convergence speed while protecting the speedup gain.*

Keywords: *Artificial Bee Colony algorithm, parallelization, emigrant creation.*

1. Introduction

In recent years, heuristic approaches have been developed as an alternative to the traditional methods for complex numerical and combinatorial optimization problems needed to be solved in a predetermined or reasonable amount of time with acceptable quality. Swarm intelligence based algorithms that is a branch of natural inpired heuristic mainly focus on the collective behaviors of insect colonies especially their problem solving abilities and they have been applied to the many real-world problems [1-6]. Artificial Bee Colony (ABC) algorithm proposed by Karaboga to solve numerical optimization problems in 2005 is one of these swarm intelligence based algorithms and tries to mimic natural behavior of real honey bees in food foraging [1-6].

Due to its robust structure and less control parameters which are important to being determined before starting search progress like crossover and mutation rate used by Genetic Algorithm (GA), ABC algorithm has been successfully applied a wide variety of numerical or combinatorial problems ranging from the neural network training [7, 8], routing packages within a wireless sensor networks [9, 10], aligning protein sequences [11, 12] and predicting secondary structures of them [13] to image quantization [14, 15] and so on [2, 6]. In spite of all those advantages, some modifications made in order to improve the performance of the ABC algorithm and raise the speed of convergence to globally optimal solution add extra burden to the standard implementation of the ABC algorithm and it still requires a long execution time to find optimal or neal optimal solutions of problems that have many parameters to be optimized and need large colony size [16-21].

It can be observed that many parts of the ABC algorithm can be run in parallel. However, some dependencies on the asynchronous workflow of the sequential ABC algorithm should be changed by considering the quality of the final solutions, convergence speed and efficiency of the used parallel architectures. Driven by these mentioned large computational demands and variations of parallelizable part of the ABC algorithm, many researchers have developed parallel ABC algorithms in order to increase the speedup on both shared and distributed memory concepts. Parallelization approaches of the ABC algorithm was roughly classified into two categories based on the number of colonies. In the first category, multiple colonies able to communicate each other are used on the same search space. Rather than utilizing multiple colonies, using a single colony divided into subcolonies and then distributed to the processors to work simultaneously is evaluated in the second category.

Narasimhan presented a parallel version of the ABC algorithm in which the entire colony of bees is divided equally then distributed among selected processors so that each processor tries to improve the local set of solutions and obtained satisfying results for both quality of final solutions and running performance [22]. In that study, each subcolony is placed in the local memories of the related

Received on: 15.02.2016
Accepted on: 25.03.2016

processors and the entire colony is stored in the global shared memory [22]. At the end of each cycle, improved solutions on each processor are copied into the corresponding locations in the global shared memory in order to maintain the relationship between bees in the sequential ABC algorithm [22]. Banharnsakun et al. designed a parallel ABC algorithm for distributed memory systems and improved the scalability problems on the hardware [23]. After completing a predetermined number of cycles, local best solution exchanging between two different subgroups which are randomly determined is carried out [23]. Luo et al. proposed a food source sharing approach between compute nodes called ripple-communication strategy and showed that ripple-communication strategy increases the accuracy of the ABC algorithm on finding near best solutions [24]. Subotic et al. used multiple bee colony in a communication manner that each bee colony shares their best-so-far solutions with all other colonies after predetermined number of cycle was completed [25, 26]. Parpinelli et al. investigated parallel performance of the ABC algorithm by adapting it for master-slave, multi-hive with migration and hybrid hierarchical models [27]. A more detailed examination of the parallel ABC algorithm has been conducted by Basturk and Akay. They first introduced a synchronous ABC algorithm and compared its performance with the asynchronous sequential counterpart on large-scale benchmark functions [28]. Secondly, a coarse-grained parallel model of the ABC algorithm has been presented [29]. While their parallel implementation of the ABC algorithm has been tested using high dimensional numeric compute expensive problems with different number of subpopulations, migration intervals and migration topologies in the first part of the experimental studies, the second part was devoted to the studies on training artificial neural network by utilizing the proposed parallel model [29].

Changing local best food source with a food source in the topological neighbor subpopulation is the common part of the parallel ABC algorithms. This type of changing process is as important as the established neighbrohood relationship between subcolonies to maintain the population diversity. However, changing process stops the parallel execution and increases the total running time by adding the communication overhead. Because of this reason deciding which food source should be chosen as an emigrant rather than the local best solution is substantial for both maintaining variety of the subpopulations and performance gain compared to the sequential counterpart. In the proposed parallel ABC algorithm, food sources that will be swapped based on the used neighborhood topology is determined by combining the local best food source with a randomly determined food source. The rest of the paper is organized as follows. Section 2 provides a detailed description of the original sequential ABC algorithm. The proposed approach for determining the distributed food sources in each subpopulation is explained in Section 3. Experimental studies are reported in Section 4. Finally, conclusing and future research lines are provided in Section 5.

2. Artificial Bee Colony Algorithm

Foraging behaviors, memorizing and information sharing characteristics of the real honey bees are the main motivations used by the ABC algorithm [1]. ABC algorithms classifies the bees in the colony by their role played in the minimal foraging model as employed, onlooker and scout [30-35]. Employed bees exploit food sources, carry information back to the hive and then share the information about the sources with onlooker bees. Onloker bees wait in the hive and try to choose a food source by means of the information shared by employed bees. The tendency of the choosing a food source by onlookers is directly proportional to the quality of the food sources [30-35]. If a food source is exploited or abandoned, and employed bee associated with this source becomes a scout bee and searches environment randomly to find a new food source.

When using ABC algorithm to solve an optimization problem, food sources in the search space correspond to the possible solution of the problem and the nectar amount of the food source represents the fitness value of the solution [30-35]. The main steps of the ABC algorithm which reflects the cyclical relationship between employed bees, onlooker bees and scout bees can be summirized in the Fig. (1) below.

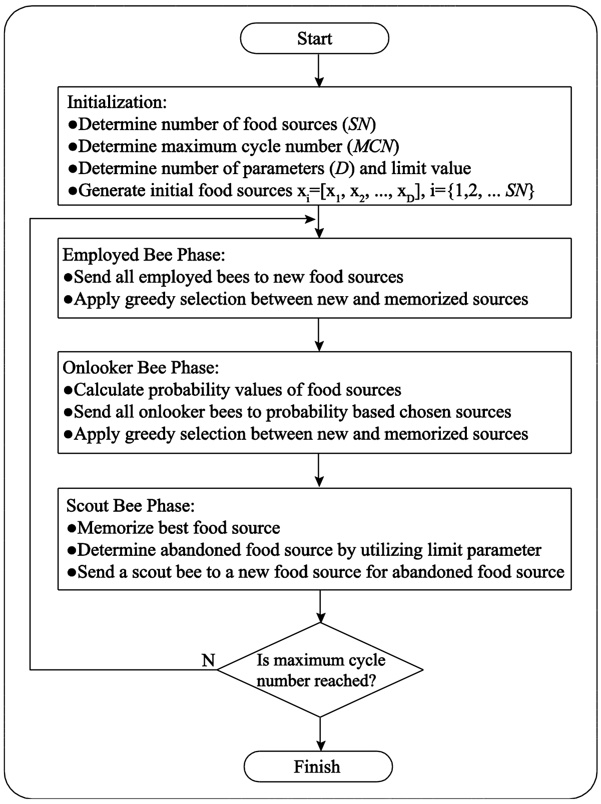
Start
Initialization:
●Determine number of food sources (SN)
●Determine maximum cycle number (MCN)
●Determine number of parameters (D) and limit value
●Generate initial food sources x_i=[x_1, x_2, ..., x_D], i={1,2, ... SN}
Employed Bee Phase:
●Send all employed bees to new food sources
●Apply greedy selection between new and memorized sources
Onlooker Bee Phase:
●Calculate probability values of food sources
●Send all onlooker bees to probability based chosen sources
●Apply greedy selection between new and memorized sources
Scout Bee Phase:
●Memorize best food source
●Determine abandoned food source by utilizing limit parameter
●Send a scout bee to a new food source for abandoned food source
N
Is maximum cycle number reached?
Finish

Figure 1. Fundamental steps of the ABC algorithm

2.1. Generating Initial Food Sources

ABC algorithm start its optimization progres by randomly generating an initial set of food sources which corresponds to the possible solutions. In the ABC algorithm, for a numerical problem that needs to optimize *D* different parameters identified by lower bound x_j^{min} and upper bound x_j^{max}, x_{ij} parameter of a solution vector x_i, where $j \in \{1, 2, ..., D\}$ and $i \in \{1, 2, ..., SN\}$, in the initial food source population consists of SN solution vectors is formulated as given in Eq. (1) [30-35].

$$x_{ij} = x_j^{\min} + rand(0,1)\left(x_j^{\max} - x_j^{\min}\right) \tag{1}$$

2.2. Sending Employed and Onlooker Bees to Food Sources

In ABC algorithm, each food source is associated with only one artificial bee and this bee attempts to produce a new food source depending on the location infomration in its memory [30-35]. If the nectar quality of the new food source is better than the known source, the bee will decide to forget the previous food source information in its mind, which is considered as a greedy selection mechanism, to utilize it for the next search cycle. The mathematial expression used both the employed and onlooker bees to produce a candidate food source in the neighborhood of the memorized food source is given in Eq. (2).

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) \tag{2}$$

ϕ_{ij} is a random number between *-1* and *1*, $k \in \{1, 2, ..., SN\}$ and $j \in \{1, 2, ..., D\}$ where SN and D denote number of food sources and dimensions of the solution vectors, respectively, are randomly chosen indexes [30, 31]. Although, the value of *k* is randomly determined, it should be noticed that identical values are not assigned to *k* and *i* indices. v_{ij} is the newly created *jth* parameter for the solution vector v_i whose parameters have the same with the solution vector x_i except the randomly selected *jth* parameter value [30-35].

ABC algorithm accommodates the preference of a food source by an onlooker bee with the nectar amount of that food source. After employed bees have shared the information kept in ther minds on the dance area, an onlooker bee chooses a food source depending on the probability value associated with that food source. The probability of a food source which increases with the nectar quality of the sources is calculated as below;

$$p_i = \frac{fitness_i}{\sum_j^{SN} fitness_j} \tag{3}$$

where $fitness_i$ is the fitness value of the solution presented by the food source in the position *i* and *SN* is the number of food sources [30-35].

2.3. Abandoning Food Sources

In a robust search, exploitation and exploration progress should be maintained in a balanced manner. If a food source cannot be improved through a predetermined number of iterations or cycles, the employed bee associated with this food source will become a scout bee and leave the food source to start a random search operation. The number of cycle used to abandon a source is an improtant control parameter of the ABC algorithm called as limit value. As in basic ABC algorithm, one food source for which the limit value is exceed at most when compared to the other sources is abandoned and one employed bee becomes scout bee for each cycle [30-35].

3. Determining Distributed Food Source

In distributed architectures, dividing the whole population into subpopulations and then assigning these subpopuations to different compute nodes are probably the most preferred parallel computing model due to its suitability to implement and less communication overhead. Each subpopulation in different compute nodes is evaluated independently and exchanges the information about the selected individual with other subpopulations based on the neighborhood topology [29, 36]. Neighborhood topologies commonly used when determining the direction of the information exchange are given in the Fig. (2) [28, 29, 36]. However, when a population based metaheuristic is parallelized using this type of computation model, the speedup performance of the algorithm change with the selected neighborhood topology, number of subcolonies, communication interval between subcolonies and types of information being distributed between compute nodes [29].

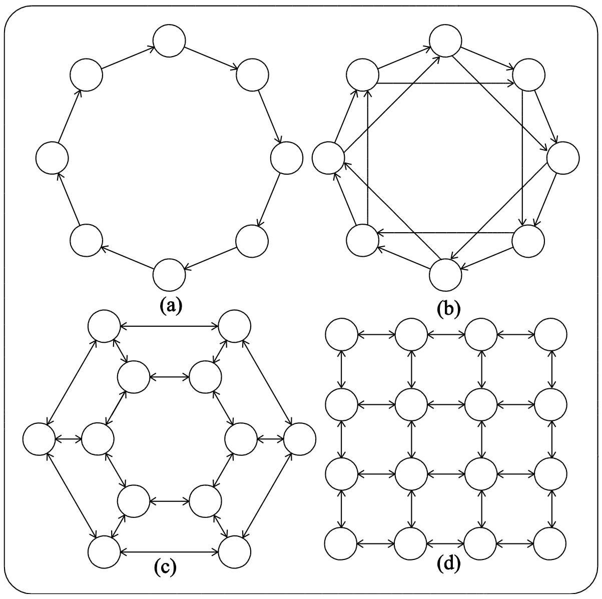

Figure 2. Ring (a), ring 1+2 (b), torus (c) and lattice (d)

Although all of these topologies have some advantages, deciding which solution to be exchanged shoud be main concern for increasing performance of the algorithm without deteriorating speedup and efficiency. In the vast majority of these topologies, the worst solution or solutions found in a subpopulation is replaced with the best solution or solutions of the topological neighbor subpopulation or neighbor subpopulations based on the used communication schema [23, 29]. The best solutions found in each subpopulation might be seen as a convenient migrants. But some situations, getting the best food source from the neighbor can not be enough to reflect the properties of other solutions found in the same population. Another limitation stemmed from the utilization of the best food source is that if the local best food source replaced with the worst food source of the neighbor subgroup in the previous migration time can not be improved until the next migration and then the same local best food source is sent more than once to the neighbor subgroup, population diversity is deteriorated. In other words, occurance of multiple copies of the same local best food source in the neighbor subgroup decreases the selection probability of the other food sources due to their relatively high fitness values [36].

Table 1. Benchmark functions used in the experiments

Function	Range	Formulation	Global Min.
Sphere	$[-100,100]$	$f_1(\vec{x})=\sum_{i=1}^{D}(x_i^2)$	$f_1(\vec{x})=0$
Griewank	$[-600,600]$	$f_2(\vec{x})=\frac{1}{4000}\left(\sum_{i=1}^{D}(x_i^2)\right)-\left(\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)\right)+1$	$f_2(\vec{x})=0$
Rosenbrock	$[-30,30]$	$f_3(\vec{x})=\sum_{i=1}^{D-1}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right)$	$f_3(\vec{x})=0$
Rastrigin	$[-5.12,5.12]$	$f_4(\vec{x})=\sum_{i=1}^{D}\left(x_i^2-10\cos(2\pi x_i)+10\right)$	$f_4(\vec{x})=0$
Dixon-Price	$[-10,10]$	$f_5(\vec{x})=(x_1-1)^2+\sum_{i=2}^{D}i\left(2x_i^2-x_{i-1}\right)^2$	$f_5(\vec{x})=0$
Penalized	$[-50,50]$	$f_6(\vec{x})=\frac{\pi}{D}\left(10\sin^2(\pi y_1)+(\sum_{i=1}^{D}(y_i-1)^2(1+10\sin^2(\pi y_{i+1})))+(y_D-1)^2\right)+\sum_{i=1}^{D}u(x_i,10,100,4)$ $u(xi,a,k,m)=\begin{cases}k(x_i-a)^m, x_i>a\\ 0, -a\le x_i\le a\\ k(-x_i-a)^m, x_i<-a\end{cases}$ and $y_i=1+\frac{1}{4}(x_i+1)$	$f_6(\vec{x})=0$

In the proposed model, the food source chosen as an emigrant between neighbor compute nodes is determined in a different manner that the best food source in the subpopulation is combined with randomly chosen food source by changing the parameters of the best food source with the more efficient parameters taken from the randomly chosen food source [36]. By utilizing this kind of cooperative schema, food sources to be exchanged between neighbor subcolonies carry more information about the situations of their colonies. Another important aspect of the proposed model is that population diversity in each compute node is protected more when compared with the local worst and local best changing approach [36]. If a population or colony consist of a set of solution in which some of them are the same or close to each other, probability of a major change that helps avoiding a local minima decreases. The working schema of the cooperative generation method and its integration in the parallel ABC algorithm is given in the Fig. (3) [36].

4. Experimental Studies

Benchmark functions that we used in order to test the performance of the standard, ring based parallel and ring based cooperative ABC algorithms are given in the Table 1. The Sphere function, *f*$_1$, is a convex, unimodal function which has no local minimum excep the global one. Griewank function, *f*$_2$, has a product term and number of local optimas increase with the dimensionality. Rosenbrock's Valley function, *f*$_3$, is one of the most difficult optimization problem. Its global optimum is inside a long, narrow, parabolic shaped flat valley. Rastrigin function, *f*$_4$, is constructed from the Sphere function by adding a cosine modulator term to produce many local minimas. Finally, *f*$_5$ and *f*$_6$ are Dixon-Price and Penalized functions, respectively.

```
1: Initialization:
2:       Assign values to limit andMCN parameters
3:       Generate initial food sources (SN) by using Eq. 1
4: Repeat
5:     Employed Bee Phase:
6:           Send employed bees to new food sources by using Eq. 2
7:     Onlooker Bee Phase:
8:           Find probability values of each food source by using Eq. 3
9:           Determine selected food source using $p_i$ values
10:          Send selected onlooker bee to food source by using Eq. 2
11:    Scout Bee Phase:
12:          Determine the abandoned food source using limit values
13:          Send a scout bee for this abandoned food source
14:    if Migration period is reached then
15:        $X_{Random}$ ← random food source in the compute node
16:        $X_{Best}$, $X_{Coop}$ ← best food source in the compute node
17:        for j ← 1...D do
18:             Change $X_{Coop,j}$ with $X_{Random,j}$
19:             Calculate fitness values of new $X_{Coop}$
20:           if fitness($X_{Coop}$) ≤ fitness ($X_{Best}$) then
21:                Change $X_{Coop,j}$ with $X_{Best,j}$
22:           end if
23:        end for
24:        Send $X_{Coop}$ to the neighbor subpopulation
25:    end if
26:    Memorize best food source found so far
27: UntilMaximum cycle number is reached
```

Figure 3. Cooperative model based parallel ABC algorithm

The size of the bee colony is chosen as 160 for the experiments. The dimension on each function is set to 400 and the value of *limit* is taken equal to the number of parameters. The proposed method is implemented in C programming language using Open Message Passing Interface (OpenMPI) library [37, 38]. All of our tests have been performed on the cluster that consist of compute nodes powered by Intel® i5 4670 processors with 2 gigabytes (GB) of random access memories (RAM). In all experiments, the maximum number of iterations is set 2000. Migration topology used in the parallel implementations is ring and migration interval that controls the frequency of

Table 2. Comparison results of ABC and its parallel implementations (Mig.Per.=20) on two processors

Functions	ABC		Ring-ABC		Coop-Ring ABC	
	Mean	Std.	Mean	Std.	Mean	Std.
$f_1(x)$	3.229954e+03	2.473820e+03	3.266018e+04	6.084122e+03	**1.034880e-01**	1.150681e-01
$f_2(x)$	3.535124e+01	2.409915e+01	2.897337e+02	7.884458e+01	**3.432151e-02**	3.217872e-02
$f_3(x)$	1.807207e+05	1.491821e+05	4.000289e+07	2.388123e+07	**1.313634e+03**	5.630211e+02
$f_4(x)$	8.502711e+02	3.983880e+01	1.216464e+03	5.327817e+01	**2.335463e+01**	1.294404e+01
$f_5(x)$	1.044298e+04	1.038023e+04	3.446708e+06	2.190490e+06	**5.391258e+02**	9.035605e+01
$f_6(x)$	2.381271e-01	1.640537e-01	4.363432e-01	3.346655e-01	**3.299542e-05**	4.592101e-05

Table 3. Comparison results of ABC and its parallel implementations (Mig.Per.=20) on four processors

Functions	ABC		Ring-ABC		Coop-Ring ABC	
	Mean	Std.	Mean	Std.	Mean	Std.
$f_1(x)$	3.229954e+03	2.473820e+03	4.598622e+04	9.963033e+03	**8.648452e-01**	8.373927e-01
$f_2(x)$	3.535124e+01	2.409915e+01	3.766516e+02	6.249915e+01	**2.378765e-01**	2.254262e-01
$f_3(x)$	1.807207e+05	1.491821e+05	9.187267e+07	4.842134e+07	**1.398775e+03**	2.151771e+02
$f_4(x)$	8.502711e+02	3.983880e+01	1.297987e+03	6.566077e+01	**4.375327e+01**	6.205766e+00
$f_5(x)$	1.044298e+04	1.038023e+04	1.250928e+07	7.023127e+06	**5.350509e+02**	7.779291e+01
$f_6(x)$	2.381271e-01	1.640537e-01	1.078099e+01	7.494439e+01	**1.020629e-03**	3.468314e-03

Table 4. Comparison results of ABC and its parallel implementations (Mig.Per.=40) on two processors

Functions	ABC		Ring-ABC		Coop-Ring ABC	
	Mean	Std.	Mean	Std.	Mean	Std.
$f_1(x)$	3.229954e+03	2.473820e+03	2.175066e+04	6.982652e+03	**1.484573e-02**	5.649759e-03
$f_2(x)$	3.535124e+01	2.409915e+01	2.203342e+02	5.419798e+01	**1.658796e-02**	8.670992e-03
$f_3(x)$	1.807207e+05	1.491821e+05	3.515014e+06	3.988365e+06	**1.706111e+03**	8.543510e+02
$f_4(x)$	8.502711e+02	3.983880e+01	1.184284e+03	4.751689e+01	**7.706627e+00**	1.945536e+00
$f_5(x)$	1.044298e+04	1.038023e+04	2.735696e+05	2.314094e+05	**5.939535e+02**	5.321407e+01
$f_6(x)$	2.381271e-01	1.640537e-01	1.991814e-01	1.137070e-01	**1.114830e-05**	3.730155e-06

Table 5. Comparison results of ABC and its parallel implementations (Mig.Per.=40) on four processors

Functions	ABC		Ring-ABC		Coop-RingABC	
	Mean	Std.	Mean	Std.	Mean	Std.
$f_1(x)$	3.229954e+03	2.473820e+03	2.939110e+04	5.203470e+03	**1.263425e-02**	1.730070e-02
$f_2(x)$	3.535124e+01	2.409915e+01	2.784435e+02	4.413102e+01	**1.195933e-02**	1.630222e-02
$f_3(x)$	1.807207e+05	1.491821e+05	2.733728e+07	2.000934e+07	**1.287916e+03**	5.716947e+02
$f_4(x)$	8.502711e+02	3.983880e+01	1.246378e+03	5.017284e+01	**1.101184e+01**	3.146331e+00
$f_5(x)$	1.044298e+04	1.038023e+04	3.063510e+06	1.357753e+06	**4.872473e+02**	7.290517e+01
$f_6(x)$	2.381271e-01	1.640537e-01	1.003665e-01	1.058251e-01	**5.533145e-06**	4.258351e-06

the food source exchanging between subpopulations is set two different values; 20 and 40. Each of the experiments is repeated 20 times with different random seeds and the mean best values and standard deviations have been recorded. From the simulation results given in Tables 2-5 for different number of compute nodes and migration periods, it is clear that the mean best objective function values obtained by the proposed cooperative model outperform the standard ABC algorithm and ring schema based parallel ABC algorithm. By distributing

Table 6. Speedup and efficiency values for Ring and Coop-Ring ABC algorithms (Mig.Per.=40) on two processors

Functions	ABC	Ring-ABC	Coop-Ring ABC	Speedup		Efficiency	
	Time(s)	Time(s)	Time(s)	ABC/Ring	ABC/Coop	Ring ABC	Coop-Ring
$f_1(\vec{x})$	0.335517	0.190797	0.207589	1.7585	1.6163	0.8793	0.8081
$f_2(\vec{x})$	9.541212	4.780286	4.865502	1.9960	1.9610	0.9980	0.9805
$f_3(\vec{x})$	0.742329	0.375454	0.412685	1.9772	1.7988	0.9886	0.8994
$f_4(\vec{x})$	4.167828	2.092572	2.111593	1.9917	1.9738	0.9959	0.9869
$f_5(\vec{x})$	2.708567	1.364566	1.415034	1.9849	0.9925	1.9141	0.9571
$f_6(\vec{x})$	8.117327	4.396277	4.510353	1.8464	0.9232	1.7997	0.8999

Table 7. Speedup and efficiency values for Ring and Coop-Ring ABC algorithms (Mig.Per.=40) on four processors

Functions	ABC	Ring-ABC	Coop-Ring ABC	Speedup		Efficiency	
	Time(s)	Time(s)	Time(s)	ABC/Ring	ABC/Coop	Ring ABC	Coop-Ring
$f_1(\vec{x})$	0.335517	0.103096	0.119990	3.2544	2.7962	0.8136	0.6991
$f_2(\vec{x})$	9.541212	2.399108	2.501098	3.9770	3.8148	0.9942	0.9537
$f_3(\vec{x})$	0.742329	0.203075	0.246921	3.6554	3.0063	0.9139	0.7516
$f_4(\vec{x})$	4.167828	1.056093	1.097356	3.9465	3.7981	0.9866	0.9495
$f_5(\vec{x})$	2.708567	0.691373	0.858954	3.9177	3.1533	0.9794	0.7883
$f_6(\vec{x})$	8.117327	2.107239	2.240383	3.8521	3.6232	0.9630	0.9058

Table 8. Speedup and efficiency values for Ring and Coop-Ring ABC algorithms (Mig.Per.=20) on two processors

Functions	ABC	Ring-ABC	Coop-Ring ABC	Speedup		Efficiency	
	Time(s)	Time(s)	Time(s)	ABC/Ring	ABC/Coop	Ring ABC	Coop-Ring
$f_1(\vec{x})$	0.335517	0.206929	0.230228	1.6214	1.4573	0,8107	0,7286
$f_2(\vec{x})$	9.541212	4.791809	4.951232	1.9912	1.9270	0,9956	0,9628
$f_3(\vec{x})$	0.742329	0.406193	0.456462	1.8275	1.6263	0,9137	0,8131
$f_4(\vec{x})$	4.167828	2.098474	2.137868	1.9861	1.9495	0,9930	0,9747
$f_5(\vec{x})$	2.708567	1.390291	1.566196	1.9482	1.7294	0,9741	0,8647
$f_6(\vec{x})$	8.117327	4.433858	4.550353	1.8308	1.7839	0.9154	0.8919

Table 9. Speedup and efficiency values for Ring and Coop-Ring ABC algorithms (Mig.Per.=20) on four processors

Functions	ABC	Ring-ABC	Coop-Ring ABC	Speedup		Efficiency	
	Time(s)	Time(s)	Time(s)	ABC/Ring	ABC/Coop	Ring ABC	Coop-Ring
$f_1(\vec{x})$	0.335517	0.106680	0.141885	3.1451	2.3647	0,7862	0,5911
$f_2(\vec{x})$	9.541212	2.399059	2.501293	3.9771	3.8145	0,9942	0,9536
$f_3(\vec{x})$	0.742329	0.204072	0.290406	3.6376	2.5562	0,9094	0,6390
$f_4(\vec{x})$	4.167828	1.073781	1.124635	3.8815	3.7059	0,9703	0,9264
$f_5(\vec{x})$	2.708567	0.693667	0.926330	3.9047	2.9240	0,9761	0,7310
$f_6(\vec{x})$	8.117327	2.190383	2.251761	3.7059	3.6049	0.9265	0.9012

cooperative best food source between ring based neighbor subcolonies, the chance of getting different best food source which is more qualified than the previously swapped has been increased. Another important contribution with this approach is that diversity in the subcolonies has been maintained with the emigrant food sources that reflects the important properties of the randomly chosen food source in its subcolony.

Another comparison has been made on the speedup and efficiency values for the parallel ABC algorithms. Speedup and efficiency are commonly used metrics to measure the performance of the parallel algorithms. Speedup value is the ratio of sequential execution time to the parallel execution time and efficiency value is the ratio of speedup and the number of processors used. Optimum value of the speedup metric is equal to the number of processors and the optimum value of the efficiency is equal to 1. In Tables 6-9, average total running times over 20 different runs, speedup and efficiency value are given for different number of nodes and periods. In the calculation of the average running time for parallel ABC algorithms, total elapsed time for the slowest processor has been used. Since the generation of the cooperative food source require a comparison between all parameters of the local best food source and a randomly determined food source for each subcolony, the speedup and efficiency values of the Coop-Ring ABC algorithm lag slighlty behind the Ring-ABC algorithm especially for the functions that are less compute expensive. The effect of the proposed schema on the convergence speed of the algorithm can bee seen in the Figure 3 and Figure 4. When these figures are examined, all of them present a remarkable difference between Coop-Ring ABC and other implementations. Qualities of the best solutions are significantly increased with the start of the distribution of the cooperative food sources, that also leads to a fast convergence to the global minima of the problem, and continues to be improved with respect to the early migrations. While the effect of the proposed model is causing a gradual and fast convergence to the global

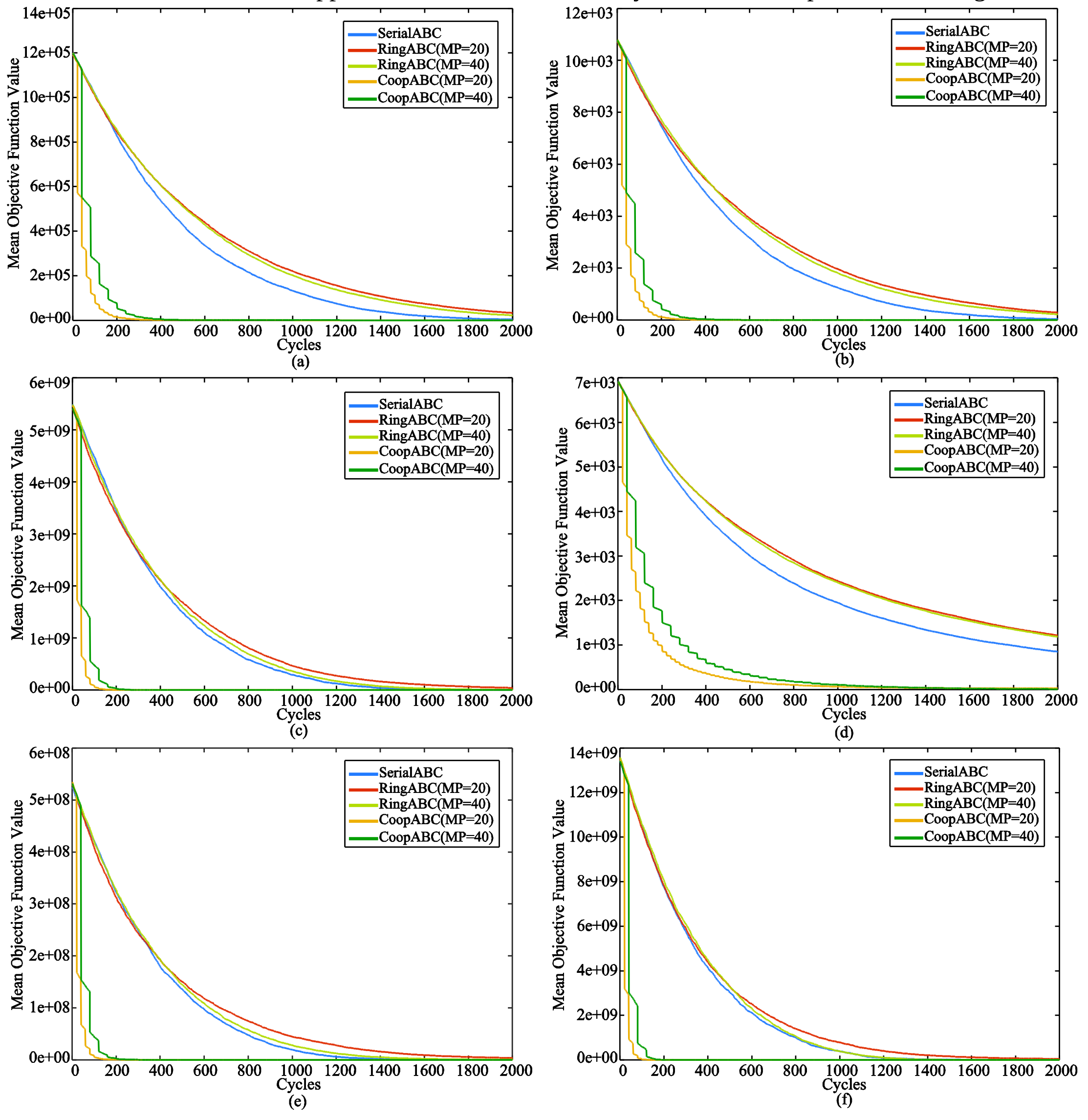

Figure 3. Convergence charactertics of the serial ABC algorithm and its ring schema based parallel implementations on two compute nodes for f_1 (a), f_2 (b), f_3 (c), f_4 (d), f_5 (e) and f_6 (f) functions

minima within the first quarter of the total cycles for the f_1, f_2 and f_4 functions, solution quality is more quickly improved within the early cycles and then convergence to the minima is stabilized as the number of cylces increases for the f_3, f_5 and f_6 functions. From the graphics given in Figure 2 and Figure 3, it is also clear that low values of the migration interval increases the convergence speed of Coop-Ring ABC algorithm by adding furher computation overhead. However, this type of quick convergence provided by the high frequency of migration does not contribute to the improvement of the best solutions in the other subpopulations.

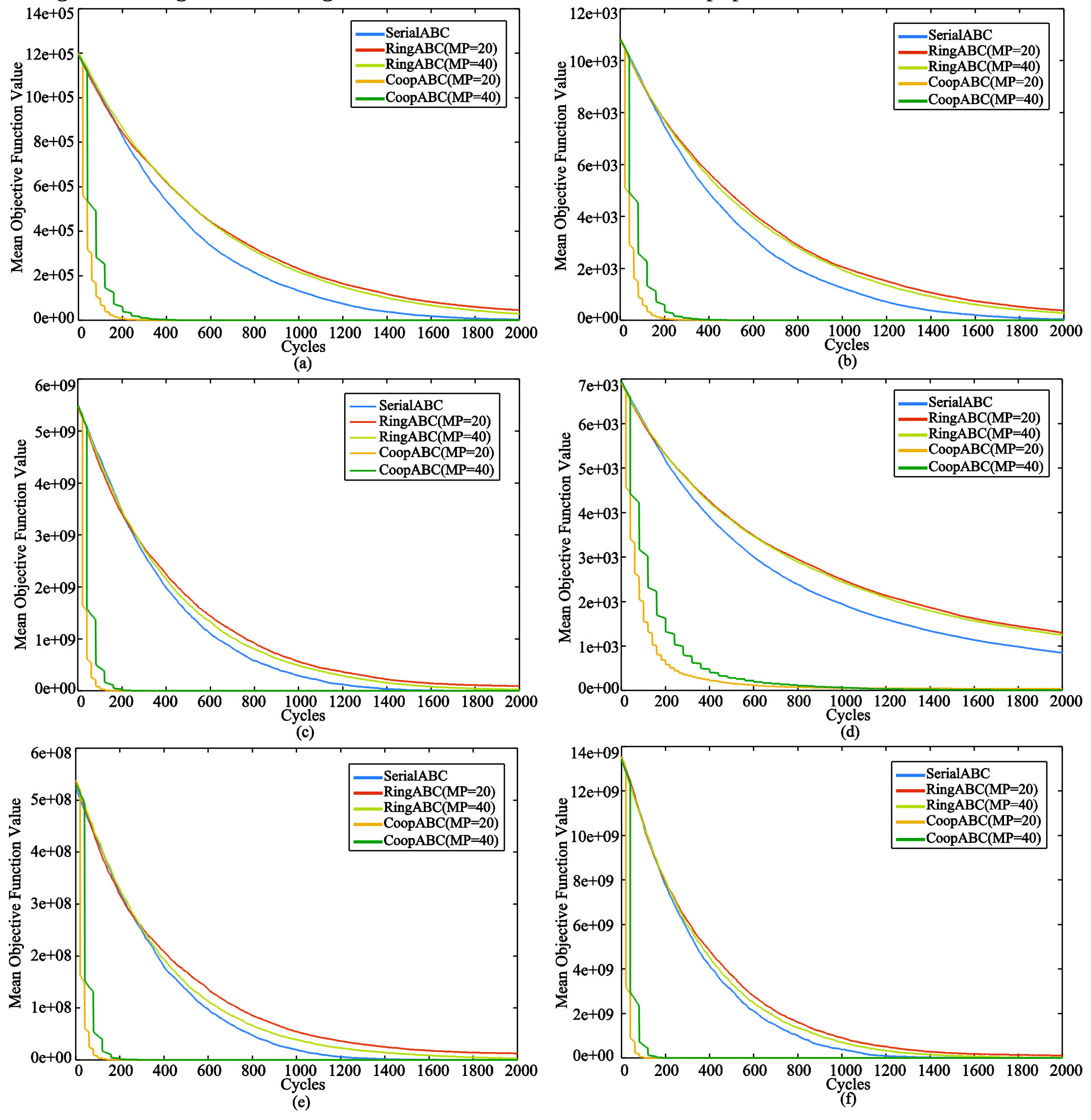

Figure 1. Convergence characteristics of the serial ABC algorithm and its ring schema based parallel implementations on four compute nodes for f_1 (a), f_2 (b), f_3 (c), f_4 (d), f_5 (e) and f_6 (f) functions

5. Conclusions

In this paper, a new creation schema for the emigrant food source between neighbor subcolonies is presented and performance effect of the proposed approach in terms of solution quality, convergence speed and running time has been investigated. Experimental studies showed that the new definition significantly improved the quality of the final solutions and convergence performance of parallel ABC algorithm with the ring migration topology when compared to the standard sequential ABC algorithm and ring based parallel ABC algorithm in which local best food sources for each subcolony are chosen to being exchanged with the local worst food sources. A future development of this work can focus on adapting the proposed schema to other migration topologies with different number of compute nodes and migration periods and its implementation on combinatorial optimization problems that require more computational time due to the necessity of the constraints.

6. References

[1] D. Karaboga, "An idea based on bee swarm for numerical optimization", *Tech. Rep.*, Turkiye, 2006.

[2] D. Karaboga, B. Akay, "A survey: algorithms simulating bee swarm intelligence", Artif Intell Rev, vol. *31*, no. *1*, pp. *68-85*, 2009.

[3] J. C. Bansal, H. Sharma, S.S. Jadon "Artificial bee colony algorithm: a survey", *Int J Advanced Intelligence,* vol. *5,* pp. *123-159,* 2013.

[4] A. L. Bolaji, A. T. Khader, M. A. Al-betar, M. A. Awadallah, "Artificial bee colony algorithm, its variants and applications: a survey", *Journal of Theorical and Applied Information Technology,* vol. *47,* no. *2,* pp. *434-459,* 2013.

[5] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, "A comprehensive survey: artificial bee colony algorithm and application", *Artif Intell Rev,* vol. *42,* no. *1,* pp. *21-57,* 2014.

[6] B. Akay, D. Karaboga, "A survey on the applications of the artificial bee colony in signal, image and video processing", *Signal, Image and Video P,* vol. *9,* pp. *967-990,* 2015.

[7] D. Karaboga, B. Akay, "Artificial bee colony algorithm for training feed forward neural networks" in *IEEE 15th Signal Processing and Communication Applications Conference*, Eskişehir, 2007, pp. 1-4.

[8] D. Karaboga, C. Ozturk, "Neural network training by artificial bee colony algorithm on pattern classification", *Neural Network World,* vol. 19, pp. 687-697, 2009.

[9] D. Karaboga, S. Okdem, C. Ozturk, "Cluster based wireless sensor network routing using artificial bee colony algorithm", *Wirel Netw,* vol. 18, pp. 847-860, 2011.

[10] D. Karaboga, C. Ozturk, B. Gorkemli, "Probabilistic dynamic deployment of wireless sensor networks by artificial bee colony algorithm", *Sensors,* vol. 11, no. 6, pp. 6056-6066, 2011.

[11] X. Lei, J. Sun, X. Xu, L. Guo, "Artificial bee colony algorithm for solving multiple sequence alignment", in *IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications,* Changsha, 2010, pp. 337-342.

[12] S. Aslan, C. Ozturk, "Alignment of biological sequences by discrete artificial bee colony algorithm" in *IEEE 23th Signal Processing and Communications Applications Conference*, Malatya, 2015, pp. 678-681.

[13] C. M. V. Benitez, H.S. Lopes, "Parallel artificial bee colony approaches for protein structure prediction using 3dhp-sc model", Intelligent Distributed Computing IV, Springer-Berlin-Heidelber, pp. 255-264, 2010.

[14] E. Hancer, C. Ozturk, D. Karaboga, "Extraction of brain tumors from mri images with artificial bee colony algorithm based segmentation methodology", in *IEEE 8th International Conferece on Electrical and Electronics Engineering,* Bursa, 2013, pp. 516-520.

[15] E. Hancer, C. Ozturk, D. Karaboga, "Color image quantization: a short review and an application with artificial bee colony algorithm", *Informatica,* vol. 25, no. 3, pp. 483-503, 2014.

[16] D. Karaboga, B. Akay, "A comparative study of artificial bee colony algorithm", *Appl Math Comput,* vol. 214, pp. *108-132*, 2009.

[17] C. Zhang, D. Ouyang, J. Ning, "An artificial bee colony approach for clustering", *Expert Syst Appl,* vol. 37, pp. *4761-4767,* 2010.

[18] G. Zhu, S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization", *Appl Math Comput,* vol. *217,* no. *7,* pp. *3166-3173,* 2010.

[19] D. Karaboga, B. Akay, "A modified artificial bee colony algorithm for constrained optimization problems", *Appl Soft Comput,* vol. *11,* pp. *431-441,* 2011.

[20] W. Gao, S. Liu, L. Huang, "A global best artificial bee colony algorithm for global optimization", *J Comput Appl Math,* vol. *236,* pp. *2741-2753*, 2012.

[21] D. Karaboga, B. Gorkemli, "A quick artificial bee colony algorithm and its performance on optimization problems", *Appl Soft Comput,* vol. *23,* pp. *227-238,* 2014.

[22] H. Narasimhan, "Parallel artificial bee colony algorithm", in *World Congress on Nature & Biologically Inspired Computing,* Coimbatore, 2009, pp. *306-311.*

[23] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, "Artificial bee colony algorithm on distributed environments", in *Second World Congress on Nature & Biologically Inspired Computing,* 2010, Fukuoka, pp. *13-18.*

[24] R. Luo, T. Pan, P. Tsai, J. Pan, "Parallelized artificial bee colony algorithm with ripple-communication strategy", in *4th International Conference on Genetic and Evolutionary Computing*, 2010, Shenzen, pp. *350-353.*

[25] M. Subotic, M. Tuba, N. Stanarevic, "Parallelization of the artificial bee colony algorithm", in *Proceedings of the 11th WSEAS Internation Conference on Neural Networks and 11th WSEAS International Conference on Evolutionary Computing,* 2010, Wisconsin, pp. *191-196.*

[26] M. Subotic, M. Tuba, N. Stanarevic, "Difference approaches in parallelization of the artificial bee colony algorithm", International Journal of Mathematical Models and Methods in Applied Sciences, vol. 5, pp. *755-762*, 2011.

[27] R. S. Parpinelli, C. M. V. Benitez, H.S. Lopes, "Parallel approaches for the artificial bee colony algorithm", in *Handbook of Swarm Intelligence,* Springer Berlin Heidelberg, 2011, pp. *329-345.*

[28] A. Basturk, R. Akay, "Parallel Implementation of synchronous type artificial bee colony algorithm for global optimization", J Optim Theory Appl, vol. *155,* pp. *1095-1104,* 2012.

[29] A. Basturk, R. Akay, "Performance analysis of the coarse-grained parallel model of the artificial bee colony algorithm", Inform Sciences, vol. *253,* pp. *34-55,* 2013.

[30] D. Karaboga, B. Akay, "A Powerful and efficient algorithm for numerical function optimization: artificial bee colony algorithm", Journal of Global Optimization, vol. *39,* pp. *459-471,* 2007.

[31] D. Karaboga, B. Akay, "On the performance of artificial bee colony algorithm", Applied Soft Computing Journal, vol. *8,* pp. *687-697,* 2008.

[32] B. Akay, D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization", Journal of Intelligent Manufacturing, vol. *23,* no. 4, pp. *1001-1014,* 2012.

[33] B. Akay, "Synchronous and asynchronous Pareto-based multi-objective artificial bee colony algorithms", Journal of Global Optimization, vol. *57,* no. 2, pp. *415-445,* 2013.

[34] F. Koylu, M. Celik, D. Karaboga, "Performance analysis of ABCMiner algorithm with different objective functions", in *IEEE 21th Signal Processing and Communication Applications Conference,* Haspolat, 2013, pp. 1-5.

[35] M. Celik, F. Koylu, D. Karaboga, "CoABCMiner: An algorithm for cooperative fule classification system based on artificial bee colony algorithm", International Journal on Artificial Intelligence Tools, vol. 24, pp. 1-40, 2015.

[36] D. Karaboga, S. Aslan, "A new emigrant creation strategy for parallel artificial bee colony algorithm", in *IEEE* 9th *International Conference on Electrical and Electronics,* Bursa, 2015, pp. 689-694.

[37] A Grama, G. Karypis, V. Kumar, A. Gupda, "Introduction to parallel computing", Addison Wesley, Harlow, England, 2003.

[38] P. Pacheco, "An introduction to parallel programming", Morgan Kaufmann, Burlington, USA, 2011.

Dervis Karaboga received M.Sc. degree in 1988 from the Department of Electronics and Communication Engineering, Istanbul Technical University, Turkey and the Ph.D. degree in 1994 from Systems Engineering Department, University of Wales, College of Cardiff, United Kingdom. He is currently professor at the Department of Computer Engineering, Erciyes University, Kayseri, Turkey. His research areas include optimization, fuzzy systems, neural networks and engineering applications of intelligent methods.

Selcuk Aslan received B.Sc. and M.Sc. degree in the Computer Engineering from Erciyes University, Department of Computer Engineering, Kayseri, Turkey in 2011 and 2013, respectively. He is currently Ph.D. candidate in Erciyes University, Department of Computer Engineering, Kayseri, Turkey. His research interests include combinatorial optimization in bioinformatics, parallel and distributed computation.