ORDER BASED EMIGRANT CREATION STRATEGY FOR PARALLEL ARTIFICIAL BEE COLONY ALGORITHM

Alperen AKSOY[1,+], Selçuk ASLAN[2], Derviş KARABOĞA[3]

[1]Alanya Alaaddin Keykubat University, Engineering Faculty, Computer Engineering Department, Antalya, Turkey

[2]Ondokuz Mayıs University, Engineering Faculty, Computer Engineering Department, Samsun, Turkey

[3]Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, Turkey

alperen.aksoy@alanya.edu.tr, selcuk.aslan@omu.edu.tr, karaboga@erciyes.edu.tr

Abstract

Artificial Bee Colony (ABC) algorithm inspired by the foraging behaviors of real honey bees is one of the most important swarm intelligence based optimization algorithms. When considering the robust and phase divided structure of the ABC algorithm, it is clearly seen that ABC algorithm is intrinsically suitable for parallelization. In this paper, we proposed a new emigrant creation strategy for parallel ABC algorithm. The proposed model named order based emigrant creation strategy depends on sending best food source in a subcolony after modifying it with another food source chosen sequentially from the same subcolony at each migration time. Experimental studies on a set of numerical benchmark functions showed that parallel ABC algorithm powered by the newly proposed model significantly improved quality of the final solutions and convergence performance when compared with standard serial ABC algorithm and parallel ABC algorithm for which the best food sources in the subcolonies directly used as emigrants.

Keywords: swarm intelligence, Artificial Bee Colony algorithm, parallelization, mobile platform.

[+] This paper has been presented at the ICENTE'17 (International Conference on Engineering Technologies) held in Konya (Turkey), December 07-08, 2017.

1. Introduction

In recent years, modeling complex biological behaviors of species living together has become a major technique for solving optimization problems and many nature-inspired algorithms have been proposed including Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Bacterial Foraging Optimization (BFO), Artificial Bee Colony (ABC) and so on [1] – [4]. Although all of these population based optimization algorithms successfully applied fundamental mechanisms that are leading to emerge swarm intelligence, ABC algorithm is in a special position due to its phase divided structure, easy implementation and also requiring less control parameters [1] – [4].

By considering the advantages of ABC algorithm, researchers used it for solving different types of problems ranging from aligning three or more biological sequences [5], [6] and guessing their three-dimensional structures [7] to training deep neural networks [8], image segmentation [9] to designing filters [10]. Although original ABC algorithm produced sufficiently good results in most cases, some studies have been made to further increase the quality of the final solutions and change the convergence characteristics of the algorithm. Zhu and Kwong proposed the gbest-guided ABC algorithm in which the parameters of the best food source are used instead of the parameters of the randomly determined food source [11]. Coelho and Alotto changed the candidate generation equation of the ABC algorithm by utilizing the coefficients extracted from the Gaussian distribution [12]. Karaboga and Gorkemli changed the search characteristics of the onlooker bees and proposed a new ABC algorithm variant called quick-ABC algorithm [13]. Gao and Liu used mutation operation of Differential Evolution (DE) algorithm and proposed a new ABC algorithm [14]. Xiang et al. combined ABC algorithm and DE and solved numerical optimization problems successfully [15]. Wu et al. propose HHSABC algorithm that is based on the combination of Harmony Search (HS) and ABC algorithms [16]. Banitalebi et al. proposed an improved ABC algorithm by combining it with the Estimation of Distribution (ED) algorithm [17].

Other important studies about the ABC algorithm were made on the parallelization of it for distributed and shared memory based architectures. Narasimhan

parallelized ABC algorithm by dividing the whole bee colony into subcolonies and then assigning them to the different compute units [18]. In order to maintain population diversity, copies of each subcolony is also stored in the global memory [18]. Banharnsakun et al. parallelized ABC algorithm for distributed memory based architectures [19]. After completion of predetermined number of cycles, local best food sources of the two randomly determined subcolonies are exchanged [19]. Luo et al. introduced a new emigrant exchange schema called ripple-communication and applied it to the parallel ABC algorithm [20]. Experimental studies showed that ripple-communication schema increased the accuracy of the ABC algorithm [20]. Subotic et al. utilized multiple bee colonies in their parallel implementation of ABC algorithm [21], [22]. Each colony is in communication by shared local best food sources among them [21], [22]. The effect of the migration period, migration topology and number of subcolonies on the parallel implementation of the ABC algorithm was investigated detailly by Basturk and Akay [23]. They first tested parallel ABC algorithm for solving high-dimensional numeric optimization problems and thenu training neural networks for classification problems [23]. Karaboga and Aslan focused on the emigrant selection and creation approaches for the parallel ABC algorithm [24]. They first proposed a new emigrant creation approach in which the local best food source is modified before it is sent to the neighbor subcolony. Secondly, they used all local best food sources to generate a global emigrant and the sent it to all subcolonies [25].

Utilizing local best food sources as an emigrant is common part for most of the parallel implementations of the ABC algorithm. However, if the local best food sources can not be improved between two consecutive migration periods, it is possible to see the same solution more than once in the same subcolony. If the subpopulation diversity that has already been decreased by dividing the whole colony is also deteriorated because of the same food sources coming from the neighbor subcolonies, parallelization of the ABC algorithm does not go beyond a study only aiming at acceleration with the power of the compute units. In order to overcome these mentioned issues, we proposed a new technique called order based emigrant creation strategy. In order based emigrant creation strategy, the local best food source found at the current migration time of a subcolony is powered with the more useful parameters of another food source. However, the selection of the auxiliary food source for the local best is managed in a

sequential manner for decreasing the possibility of sending the same emigrant. The rest of the paper is organized as follows: In the second section, fundamental steps of the ABC algorithm are introduced. Order based emigrant creation strategy and its working schema is summarized in the third section. Experimental studies and conclusion are given in the fourth and fifth sections, respectively.

2. Artificial Bee Colony Algorithm

ABC algorithm proposed by Karaboga at 2005 is inspired by the intelligent foraging behaviors as real honey bees. In ABC algorithm the whole colony is divided into three groups by considering special foraging characteristics of the bees [26]. The first group of bees is composed of the employed bees. Employed bees are responsible for finding food sources and carry nectar extracted from the sources. When employed bees come back to the hive, they share information about the nectar quality of the sources, distance to the hive and so on, with the second group of bees. The second group of bees consists of onlooker bees. Onlooker bees wait on the hive and select a food source to exploit it after checking the information supplied by the employed bees. The food source selection of the onlooker bees is not a randomized procedure directly. Selection probability of a food source has a relationship with its quality and if the nectar quality of a food source is high, its preference by onlookers is also high when compared other sources [26]. After an onlooker bee selects a food source, she continues the source exploration and exploitation processes as done by the employed bees. The final group of bees in the ABC algorithm consists of scout bees. Scout bees like onlooker bees wait on the hive, but food source selection behavior of them is managed by an internal motivation that leads to send scouts to unvisited food sources randomly [26]. The bee phases determined by the employed, onlooker and scout bees and their cyclical relationships continue until the end of a predetermined number of iterations-cycles is completed are given in the Algorithm 1.

2.1 Generating Initial Food Sources

In ABC algorithm, each food source corresponds possible solutions of the optimization problem being solved and nectar amounts of the sources are related with the fitness values of them. ABC algorithm generates these food sources randomly

before employed bees are assigned to them. The *jth* parameter of the *ith* food source or solution within *SN* different solutions is determined randomly between lower bound of the parameter showed by x_j^{min} and upper bound of the parameter showed by x_j^{max} using the Eq. 1 [26].

$$X_{ij} = x_j^{\min} + rand(0,1)(x_j^{\max} - x_j^{\min}) \quad j = 1,2, \ldots D \tag{1}$$

In Eq. 1, *D* corresponds to the number of parameters being optimized and *rand*(0,1) is used to define a random number of varies between 0 and 1.

Algorithm 1 Fundamental steps of the ABC algorithm

1: **Initialization:**
2: Assign values to the control parameters.
3: Generate SN initial food sources with D parameters.
4: **Repeat**
5: //Employed bee phase
6: **for** $i \leftarrow 1 \ldots SN$ **do**
7: Generate new solution around the memorized one.
8: Apply greedy selection between new and memorized sources.
9: **end for**
10: //Onlooker bee phase
11: $sentBees \leftarrow 0$, $currentSource \leftarrow 1$
12: Find probability values of each food source.
13: **while** $sentBees \neq SN$ **do**
14: **if** $pcurrentSource > rand(0, 1)$ **then**
15: $sentBees \leftarrow sentBees + 1$:
16: Generate new solution around the selected one.
17: Apply greedy selection between new and memorized sources.
18: **end if**
19: $currentSource = (currentSource + 1) \bmod SN$:
20: **end while**
21: //Scout bee phase
22: Determine the abandoned food source.
23: Generate a new source for this abandoned food source.
24: **Until** Termination criteria is met

2.2 Sending Employed and Onlooker Bees

After generating initial population of *SN* different food sources, each food source is associated with only one employed bee by the ABC algorithm. The mathematical model used by the employed bees to generate a candidate food source around the memorized one is given in the Eq. (2) [26].

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

In the Eq. (2), v_{ij} is the parameter value of the candidate v_i food source whose parameters are same with the parameters of the x_i food source except the *jth* parameter [26]. x_{ij} and x_{kj} are the parameter values of the *jth* parameter of the x_i and x_k food sources, respectively and Θ_{ij} corresponds to a random number between -1 and 1 [26]. If the $fit(v_i)$ fitness value of the v_i food source calculated as in the Eq. (3) for a minimization problem is bigger than the $fit(x_i)$ fitness value of the x_i food source, the employed bee applies a greedy selection procedure between x_i and v_i food sources and trial counter for x_i food source is set to zero. Otherwise, the trial counter for x_i food source is incremented by one to show that it is not improved in the current cycle.

$$fit(x_i) = \begin{cases} 1 + |obj(x_i)|; & if\ obj(x_i) > 0 \\ 1/(1 + obj(x_i)); & if\ obj(x_i) \le 0 \end{cases} \quad (3)$$

When the employed bee phase is completed, the food sources memorized by the employed bees are introduced to onlooker bees. As mentioned before selection of a food source is not a randomized operation exactly and it is directed by the qualities of the food sources [26]. In ABC algorithm, to direct the source selection procedure of the onlooker bees, each food source is identified by a selection probability. Selection probability of the x_i food source that increases with the appropriate fitness values of it is calculated as in the Eq. (4). When the selection procedure is completed, the onlooker bees associated with these sources become employed bees and generate candidate food sources by utilizing the Eq. (2) [26].

$$p(x_i) = \frac{fit(x_i)}{\sum_{j}^{SN} fit(x_j)} \quad (4)$$

2.3 Abandoning Consumed Food Sources

In ABC algorithm, it is easily seen that the exploitation characteristics of the employed and onlooker bee phases are more dominant than the exploration characteristics of them [26]. However, for a proper search process, exploration and exploitation operations should be balanced. This subtle balance between exploration and exploitation is maintained by the scout bee phase. In scout bee phase, a food source is abandoned if required and a scout bee for abandoned food source is sent from hive to find a food source that has been discovered yet. The decision about which food source will be abandoned is made by comparing trial counters of them with ABC algorithm

specific control parameters called *limit* [26]. The food source for which its trial counter exceeds the *limit* parameters value at most is abandoned and a new food source created by the Eq. (2) is replaced with the consumed one. The value of the *limit* parameter for a numerical optimization problem is determined by the Eq. (5) given below [26].

$$\lceil a \times SN \times D \rceil \; and \; a \in \square^{+} \tag{5}$$

3. Order Based Emigrant Creation Strategy

If the parallelization approach of the ABC algorithm depends on dividing the whole colony into subcolonies and then assigning them to the different compute nodes or processor cores, it is clearly seen that the best food source found at the current migration time are chosen for sending to the neighbor subcolony or subcolonies in the vast majority of the studies. Determining a food source as an emigrant is not a single decision that should be made when parallelization of a population algorithm to decrease the execution time without deteriorating the quality of the final solutions and convergence characteristics. After deciding which sources are chosen as emigrants, they should send to the neighbor subcolonies in order to maintain the diversity and increase the quality of the existing solutions. The neighborhood relationship between subcolonies is directly determined by applying one of the commonly used migration topologies given below in the Fig. (1).

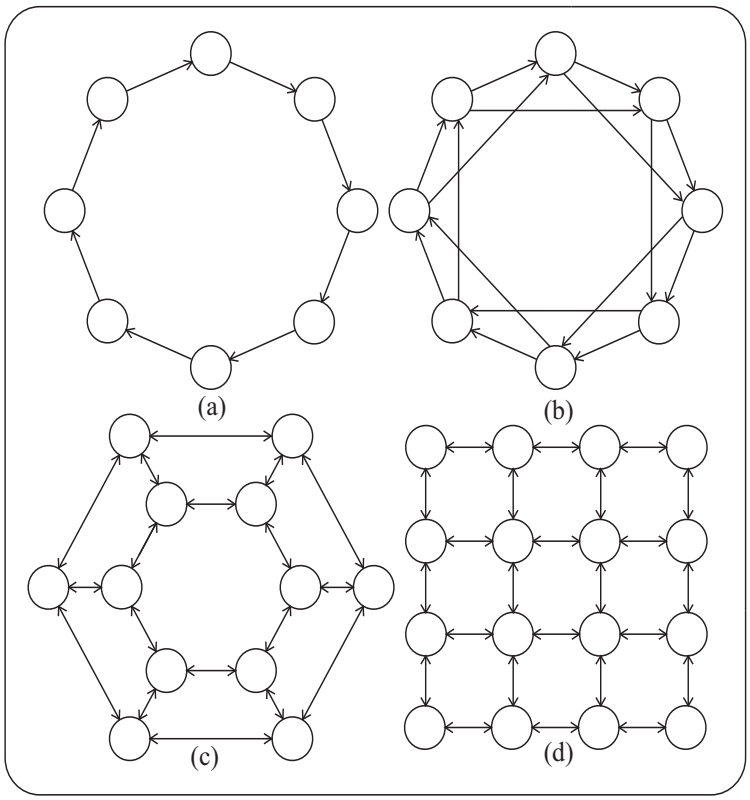

Figure 1: Ring (a), Ring 1+1 (b), Tourus (c) and Lattice (d) topologies

Although all of these topologies have some advantages, their implementations require certain number of compute nodes or processor cores and migration period or interval should be carefully chosen to decrease the total execution time on the acceptable rates. When the neighborhood topology is chosen by considering system resources and the appropriate subcolony size is determined, the required population diversity is maintained by most of the topologies at the limited rates. By considering these situations, it can be said that type of information being exchanged between subcolonies becomes more important in parallelized implementations.

Choosing the best food source in a subcolony and then sending them to the neighbor subcolony and then replacing it with the worst food source in the neighbor subcolony might be observed the most appropriate emigrant selection-creation strategy. However, mentioned emigrant strategy has some possible disadvantages. One of the main drawbacks stemmed from utilizing the best food sources as emigrants is related with the solution diversity. If the best food source used as an emigrant for the previous migration time cannot be improved until the next migration time, the same solution is sent more than once to the neighbor subcolony. Another drawback about the distribution

of the best food sources is that whether the food sources are enough to reflect the properties of the other solutions. The best food source determined as emigrant does not always guarantee that all of its parameters can be useful for generating candidate solutions.

In the proposed model, each food source is utilized in the emigrant creation procedure sequentially by considering their orders or indexes. Sending food sources sequentially can be thought appropriate because of the ability of distributing diverse information. However, if the emigrant selection-creation procedure is managed without considering fitness values of the emigrants or the subcolony where emigrants are sent contains food sources whose fitness values are better than the emigrants, the computational and communication burden are not worth sending and receiving emigrants. In other words, it can be said that the food source determined to be sent to the neighbor subcolony should be enough in terms of quality and increase the population diversity. By considering all of these limitations originated from the conventional emigrant creation-selection schemas, we proposed a new emigrant creation-selection strategy in which the best food source found at the current migration period is modified with the more suitable parameters of a food source that is changed sequentially in the same population for each migration period. The pseudo code given in the Algorithm 2 summarized the newly proposed strategy called order based emigrant creation.

When the pseudo code given in the Algorithm 2 is investigated, it is clearly seen that the *jth* parameter of the best food source x_b is updated by the corresponding parameter of the x_{ord} food source. x_{ord} food source is determined by applying modulo operation between *currentMigIndex* and *sizeOfSubcolony* that increment *ord* index one by one for each migration period. Although x_b food source is not improved between two subsequent migration time, p and $p+1$, order based emigrant strategy guarantees that it is sent to the neighbor subcolony after modification with x_{ord} and $x_{(ord+1)}$ food sources, respectively.

Algorithm 2 Order based emigrant creation strategy

1: *sizeOfSubcolony* ← number of food resources in the subcolony.
2: *migPeriod* ← the counter shows many times migration occurred.
3: *b* ← index of best food source in the subcolony.
4: *ord* ← *(migPeriod)mod(sizeOfSubColony)*
5: **if** *b == ord* **then**
6: *ord* ← *(ord+1)mod(sizeOfSubcolony).*
7: **end if**
8: x_b ← food source with the index *b*.
9: x_{ord} ← food source with the index *ord*.
10: **for** *j* ← *1...D* **do**
11: *temp* ← $x_{b,j}$
12: $x_{b,j} = x_{ord,j}$
13: $fit_{emigrant}$ ← *calcFit*(x_b).
14: **if** $fit_{emigrant} < fit(x_b)$ **then**
15: $x_{b,j}$ ← *temp.*
16: **else**
17: $fit(x_b) \leftarrow fit_{emigrant}$
18: **end if**
19: **end for**

4. Experimental Studies

Standard serial implementation of the ABC algorithm, s-ABC, its parallel implementation, p-ABC in which the best food source in a subcolony is chosen as an emigrant and it is changed with the worst food source in the neighbor subcolony and finally the parallel implementation with order based emigrant creation strategy, ord-p-ABC in which the best food source prepared by the mentioned strategy for migration is chosen as an emigrant and it is changed with the worst food source in the neighbor subcolony are tested using three different numerical benchmark functions given in the Table I below. The first function, f_1 is the Sphere function. This function has a single global minimum located at the [0, 0,…,0] point in the search space and lower and upper bounds of the parameters are set to -100 and 100, respectively. The second function, f_2 is Griewank function and has many local minimums that are regularly distributed on the search space. The global minimum of the Griewank function is located at the [0, 0,…,0] point and the values that can be assigned to the parameters are limited between -600 and 600. The last function, f_3 , is Rosenbrock function. Because of the difficulty for finding global optimum that is located at the end of a long and narrow valley, this function is

frequently used to check the convergence performance of the optimization algorithms. The global minimum of the Rosenbrock function is located at the [1, 1,…,1] point in the search space and lower and upper bounds of the parameters are set to -100 and 100, respectively.

TABLE I: Benchmark functions used in comparison

Function	Formulation
f_1	$f_1(\vec{x}) = \sum_{i=1}^{D}\left(x_i^2\right)$
f_2	$f_2(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{D} x_i^2,\right) - \left(\prod_{i=1}^{D} cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$
f_3	$f_3(\vec{x}) = \sum_{i=1}^{D-1}\left(100(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2)$

When solving the optimization problems with s-ABC, p-ABC and ord-p-ABC algorithms, number of food sources was taken equal to 100 and number of parameters was set to 500. The maximum cycle number was set to 2000 and *limit* parameter value was taken equal to the number of parameters. For p-ABC and ord-p-ABC algorithms, ring neighborhood topology was used and the whole colony was divided into four equal subcolonies and finally migration frequency was taken one percent of the maximum cycle number. All the experiments were performed on an Android® 5.1.1 operating system working on a mobile platform that is equipped with an ARM® based processor running 1.3 GHz with 1GB of RAM. s-ABC, p-ABC and ord-p-ABC algorithms were written in C programming language and parallelization and required synchronization between subcolonies were maintained by utilizing the built-in functions of the *pthreads* library. Each of the function was tested 20 different times with random seeds. Average best objective function values and standard deviations calculated over 20 different runs were recorded and given in the Tables II – IV.

When the results given in the Tables II, III and IV are investigated, it is seen that ord-p-ABC algorithm produced better average best objective function values than s-ABC and p-ABC algorithms for all of f_1, f_2, f_3 functions. While the results obtained by the p-ABC algorithm lagged slightly behind the results obtained by the s-ABC algorithm especially for the f_1 and f_3 functions, ord-p-ABC algorithm overcame the main issue to do with the decreased population diversity of the subcolonies that leads to

deteriorate the performance of the p-ABC algorithm and produced at least 10^5 times better results for the function f_1, 10^4 times better results for the function f_2 and 10^6 times better results for the function f_3 that the s-ABC and p-ABC algorithms.

TABLE II: Comparison of s-ABC and p-ABC algorithms

Func.	s-ABC		p-ABC	
	Mean	Std. Dev.	Mean	Std. Dev.
f_1	6.548575e+04	1.171718e+04	1.035821e+05	1.469255e+04
f_2	6.524802e+02	8.983277e+01	9.449440e+02	1.022350e+02
f_3	2.557243e+09	3.227430e+09	2.356143e+10	5.546746e+09

TABLE III: Comparison of s-ABC and ord-p-ABC algorithms

Func.	s-ABC		ord-p-ABC	
	Mean	Std. Dev.	Mean	Std. Dev.
f_1	6.548575e+04	1.171718e+04	4.640619e-01	5.420124e-01
f_2	6.524802e+02	8.983277e+01	9.077554e-02	6.457029e-02
f_3	2.557243e+09	3.227430e+09	3.291103e+03	3.390808e+03

TABLE IV: Comparison of p-ABC and ord-p-ABC algorithms

Func.	p-ABC		ord-p-ABC	
	Mean	Std. Dev.	Mean	Std. Dev.
f_1	1.035821e+05	1.469255e+04	4.640619e-01	5.420124e-01
f_2	9.449440e+02	1.022350e+02	9.077554e-02	6.457029e-02
f_3	2.356143e+10	5.546746e+09	3.291103e+03	3.390808e+03

Another comparison between serial and parallel implementations of the ABC algorithms was made by considering average execution times obtained from the 20 different runs in terms of second, speedup and efficiency values. The speedup is a ratio between serial and parallel execution times of the algorithms and it's maximum value can be equal to number of computing nodes or cores. Efficiency is another measure that can be found by divining speedup measure to the number of computing nodes or cores. The maximum value of the efficiency measure can be equal to 1. When the results given in the Table V and VI are analyzed, it is understood that order based emigrant creation strategy brings extra burden to the total execution time of the algorithm compared with the p-ABC algorithm. Because of the execution time needed to employ order based emigrant creation strategy, the average execution time of the ord-p-ABC algorithm is 9.194% higher than the average running time of the p-ABC algorithm for function f_1, 9.743% higher than the average running time of the p-ABC algorithm for function f_2

and finally 34.242% higher than the average running time of the p-ABC algorithm for function f_3. There is another point should be noted that when the computational complexities of the benchmark problems increase, the difference between execution times between p-ABC and ord-p-ABC algorithms decrease. The relationship between complexity of the functions and execution time of the parallel ABC algorithm showed that compute intensive problems are more suitable for order based emigrant creation strategy.

TABLE V: Speedup and efficiency values of p-ABC

Func.	s-ABC (exec. time)	p-ABC (exec. time)	Speedup	Efficiency
f_1	2.995314	1.774507	1.687969	0.421992
f_2	33.764449	10.905223	3.096172	0.774043
f_3	6.527048	2.033796	3.209293	0.802323

TABLE VI: Speedup and efficiency values of ord-p-ABC

Func.	s-ABC (exec. time)	ord-p-ABC (exec. time)	Speedup	Efficiency
f_1	2.995314	1.937657	1.545844	0.386461
f_2	33.764449	11.967766	2.821283	0.705321
f_3	6.527048	2.730220	2.390668	0.597667

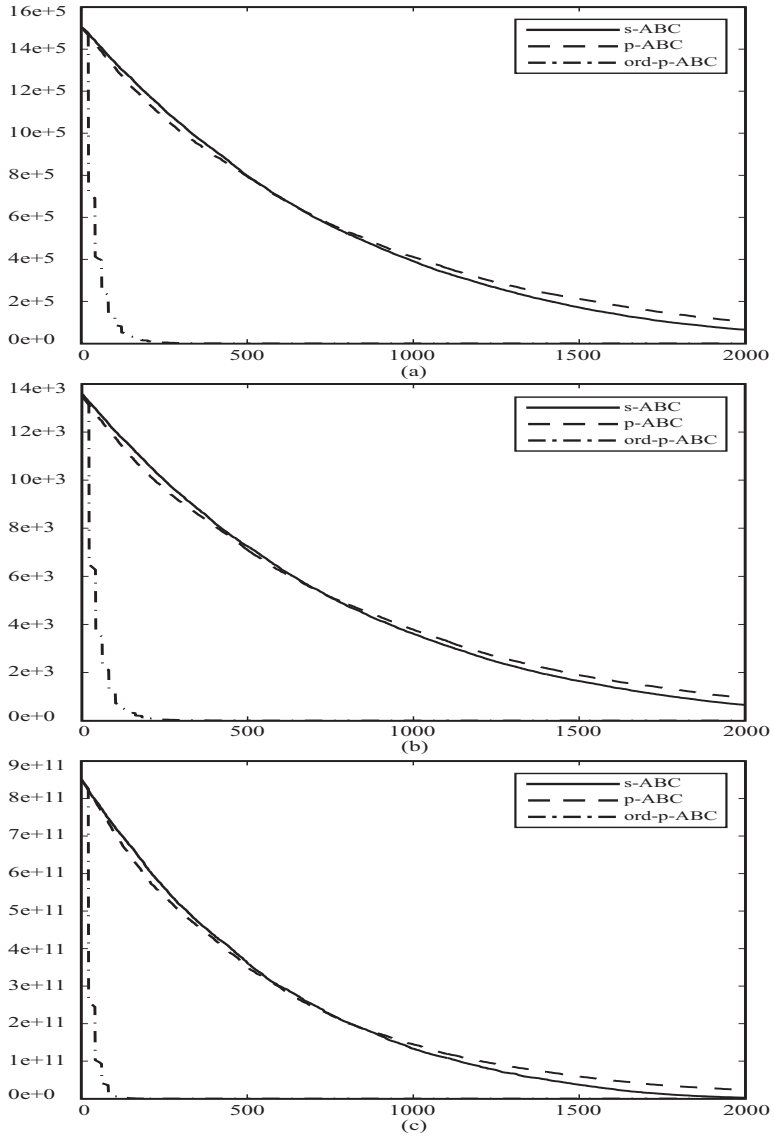

Figure 2: Convergence graphics of the ABC algorithms for f_1 (a), f_2 (b) and f_3 (c) functions

In order to analyze the effect of order based emigrant creation strategy on the convergence performance of the parallelized ABC algorithm, the graphics given in the Fig. (2) should be examined. As seen from the graphics, it is clear that the convergence performance of the ord-p-ABC algorithm is remarkably better than those of s-ABC and p-ABC algorithms. While p-ABC algorithm converges a bit faster that the s-ABC algorithm within the first quarter of the total number of cycles, s-ABC algorithm provides a sustainable convergence to the global optimum within the remaining cycles for the f_1, f_2 and f_3 functions. With the start of the distribution of the emigrant food sources, ord-p-ABC algorithm quickly diverges from the convergence curves of the s-

ABC and p-ABC algorithms and maintains its superior performance until the end of the cycles.

5. Conclusion

In this study, we proposed a new emigrant creation strategy in which the best food source found in the current migration period is modified by the more appropriate parameters of another food source determined sequentially. Newly proposed approach that is also called order based emigration creation strategy is used with the parallel implementation of the ABC algorithm. Experimental studies showed that order based emigrant creation strategy is significantly improved the quality of the finals solutions and convergence performance of the parallel ABC algorithm compared with the conventional serial and parallel implementation of the algorithm.

References

[1] B. Kumar, D. Kumar, "A Review on Artificial Bee Colony Algorithm", International Journal of Engineering & Technologies, vol. 2, no. 3, 2013.

[2] J.C. Bansal, H. Sharma, S.S. Jadon, "Artificial Bee Colony Algorithm: A Survey", International Journal of Advanced Intelligence Paradigm, vol. 5, pp: 123-159, 2013.

[3] 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 Theoretical & Applied Information Technology, vol. 47, no. 2, pp: 434-459, 2013.

[4] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, "A Comprehensive Survey: Artificial Bee Colony Algorithm and Applications", Artificial Intelligence Review, vol. 42, no. 1, pp: 21-57, 2014.

[5] P. Borovska, V. Gancheva, N. Landzhev, "Massively Parallel Algorithm for Multiple Biological Sequences Alignment", International Confe-rence on, 2013, pp: 638-642.

[6] S. Aslan, C. Öztürk, "A New Artificial Bee Colony Algorithm to Solve the Multiple Sequence Alignment Problem", International Journal of Data Mining and Bioinformatics, vol. 14, no. 4, pp: 332-353, 2016.

[7] T. Batbat, C. Ozturk, "Protein Structure Prediction with Discrete Ar-tificial Bee Colony Algorithm", International Journal of Informatics Technologies, vol. 9, no. 3, 2016.

[8] H. Badem, A. Baştürk, A. Çalışkan, M. Yüksel, "A New Efficient Tra-ining Strategy for Deep Neural Networks by Hybridization of Artificial Bee Colony and Limited-Memory BFGS Optimization Algorithms", Neurocompuing, 2017.

[9] B. Akay, D. Karaboga, "A Survey on the Applications of Artificial Bee Colony in Signal, Image and Video Processing", Signal, Image and Video Processing, vol. 9, pp: 967-990, 2015.

[10] N. Karaboga, "A new design method based on artificial bee colony algorithm for digital IIR filters", Journal of the Franklin Institute, vol. 346, no. 4, pp: 328-348, 2009.

[11] G. Zhu, S. Kwong, "Gbest-guided Artificial Bee Colony Algorithm for Numerical Function Optimization", Applied mathematics and computa-tion, vol. 217, no. 7, pp: 3166-3173, 2010.

[12] L. dos Santos Coelho, P. Alotto, "Gaussian artificial bee colony algo-rithm approach applied to Loneys´ solenoid benchmark problem", IEEE Transactions on Magnetics, vol. 47, no. 5, pp: 1326-1329, 2011.

[13] D. Karaboga, B. Gorkemli, "A quick artificial bee colony (qABC) algorithm and its performance on optimization problems", Applied Soft Computing, vol. 23, pp: 227-238, 2014.

[14] W. Gao, S. Liu, "Improved artificial bee colony algorithm for global optimization", Information Processing Letters, vol. 111, no. 17, pp: 871-882, 2011.

[15] W. Xiang, S. Ma, M. An, "Habcde: a hybrid evolutionary algorithm based on artificial bee colony algorithm and differential evolution", Applied Mathematics and Computation, vol. 238, pp: 370-386, 2014.

[16] B. Wu, C. Qian, W. Ni, S. Fan, "Hybrid harmony search and artificial bee colony algorithm for global optimization problems", Computers & Mathematics with Applications, vol. 64, no. 8, pp: 2621-2634, 2012.

[17] A. Banitalebi, M. I. A. Aziz, A. Bahar, Z. A. Aziz, "Enhanced compact artificial bee colony", Information Sciences, vol. 298, pp: 491-511, 2015.

[18] H. Narasimhan, "Parallel Artificial Bee Colony (PABC) Algorithm", World Congress on Nature and Biologically Inspired Computing (Na-BIC), Coimbatore, India, 2009, pp: 306-311.

[19] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, "Artificial Bee Colony Algorithm on Distributed Environments", Second World Congress on Nature and Biologically Inspired Computing (NaBIC), Fukuoka, 2010, pp: 13-18.

[20] R. Luo, T.S. Pan, P.W. Tsai, J.S. Pan, "Parallelized Artificial Bee Colony Algorithm with Ripple-Communication Strategy", Fourth International Conference on Genetic and Evolutionary Computing (ICGEC), Shen-zen, 2010, pp: 350-353.

[21] M. Subotic, M. Tuba, N. Stanarevic, "Parallelization of the Artificial Bee Colony Algorithm", Proceedings of the 11th WSEAS International Conference on Neural Networks, Evolutionary Computing and Fuzzy Systems, 2010, pp: 191-196.

[22] M. Subotic, M. Tuba, N. Stanarevic, "Different Approaches in Paralle-lization of the Artificial Bee Colony Algorithm", International Journal of Mathematical Models and Methods in Applied Sciences, vol. 5, pp: 755-762, 2011.

[23] A. Baştürk, R. Akay, "Performance Analysis of The Coarse-Grained Parallel Model of The Artificial Bee Colony Algorithm", Information Sciences, vol. 253, pp: 34-55, 2013.

[24] D. Karaboga, S. Aslan, "Best Supported Emigrant Creation for Parallel Implementation of Artificial Bee Colony Algorithm", IU-Journal of Electrical & Electronics Engineering, vol. 16, no. 2, pp: 2055-2064, 2016.

[25] S. Aslan, H. Badem, D. Karaboga, A. Baştürk, "A New Sychronous Parallel Artificial Bee Colony Algorithm", 1st International Conference on Engineering Technology and Applied Sciences, 2016, pp: 734-744.

[26] D. Karaboga, S. Aslan, "A Discrete Artificial Bee Colony Algorithm for Detecting Transcription Factor Binding Sites in DNA Sequences",Genetics and Molecular Research, vol. 15, no. 2, 2016.