A New Emigrant Creation Strategy for Parallel Artificial Bee Colony Algorithm

Dervis Karaboga[1], Selcuk Aslan[2]

[1,2] Department of Computer Engineering, Erciyes University, Kayseri, Turkey
karaboga@erciyes.edu.tr, selcukaslan@erciyes.edu.tr

Abstract

Artificial Bee Colony algorithm inspired by the foraging behaviour 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 sub-colonies 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 combining best solutions is proposed. The proposed model was adopted to ring migration topology and its effectiveness is compared with the ring based topology in which best food sources in each sub-populations are distributed and the original sequential counterpart. Comparative results show that the proposed model increased the quality of solutions and early convergence speed while protecting the speedup gain.

1. Introduction

In recent years, heuristic approaches have been developed as an alternative to the traditional methods for numerous complex numerical or combinatorial optimization problems needed to be solved in a predetermined or reasonable amount of time with the acceptable quality. Swarm intelligence based algorithms which is a branch of natural inspired heuristic mainly focus on the collective behaviours of insect colonies especially their problem solving abilities and they have been applied to the many real-world problems. 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 behaviour 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 execution time to the standard implementation of the ABC algorithm and it still requires a long execution time to find optimal or near 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 with each other are used on the same search space. Rather than utilizing multiple colonies, using a single colony divided into sub-colonies 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 sub-colony is placed in the local memories of the related 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 process between two different sub-groups 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 the near best solution [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 algorithms 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 sub-populations, 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 propose parallel model [29].

Changing local best food source with a food source in the topological neighbor sub-population is the common part of the parallel ABC algorithms. This type of changing process is as important as the established neighborhood relationship between sub-colonies 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 sub-populations 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 sub-population is explained in Section 3. Experimental studies are reported in Section 4. Finally, conclusion and future research lines are provided in Section 5.

2. Artificial Bee Colony Algorithm

Foraging behavior, memorizing and information sharing characteristics of the real honey bees are the main motivations used by the ABC algorithm [1]. ABC algorithm classifies the bees in the colony by their role played in the minimal foraging model as employed, onlooker and scout [1, 30, 31]. Employed bees exploit a food source, carry information back to the hive and then share this information with the onlooker bees. Onlooker 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, 31]. If a food source is exploited or abandoned, an 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, 31]. The main steps of the ABC algorithm which reflects the cyclical relationship between employed bees, onlooker bees and scout bees could be summarized below.

Algorithm 1 Fundamental Steps of the ABC Algorithm

1: **Initialization:**
2: Send all scout bees to the initial food sources.
3: **Repeat**
4: **Employed Bee Phase:**
5: Send all employed bees to new food sources.
6: **Onlooker Bee Phase:**
7: Send onlooker bees to food source using probability values.
8: **Scout Bee Phase:**
9: Send a scout bee for this abandoned food source.
10: **Memorize best food source found so far.**
11: **Until** Maximum cycle number is reached

2.1. Generating Initial Food Sources

ABC algorithm starts its optimization progress by randomly generating an initial set of food sources which corresponds to the possible solutions. In the ABC algorithm, for a numerical problem which needs to optimize D different parameters identified by lower bound x_j^{min} and upper bound x_j^{max}, x_{ij} where $j = 1, 2, \ldots, D$ parameter of a solution vector x_i where $i = 1, 2, \ldots, SN$ in the initial food source population consists of SN solution vectors is formulated as given in Eq. 1 [30, 31].

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

2.2. Sending Employed and Onlooker Bees to Food Sources

Each food source is associated with only one artificial bee and this bee attempts to produce a new food source depending on the location information in its memory in ABC algorithm [30, 31]. 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 and then keep the new food source information in its mind, which could be considered as a greedy selection mechanism, to utilize it for the next search cycle. The mathematical 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;

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

ϕ_{ij} is a random number between -1 and 1, $k \in 1, 2, \ldots, SN$ and $j \in 1, 2, \ldots, D$ where SN and D denote number of food sources and dimensions of the solutions 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 value with the solution vector x_i except the randomly selected jth parameter value [30, 31].

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 her 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 represented by the food source in the position i and SN is the number of food sources [30, 31].

2.3. Abandoning Food Sources

In a robust search, exploitation and exploration progress. If a food source cannot be improved through a predetermined number of iterations, the employed bee associated with this food source will become a scout bee and leave food source to start a random search operation. The number of cycle used to abandon a source is an important control parameter of the ABC algorithm called as *limit* value. As in basic ABC, one food source for which the *limit* value is exceed at most when compared the other sources is abandoned and one employed bee becomes scout bee for each cycle [30, 31].

3. Determining Distributed Food Source

In distributed architectures, dividing the whole population into sub-populations and then assigning these sub-populations to compute nodes are probably the most preferred parallel computing model due to its suitability to implement and less communication overhead. Each sub-population in different compute nodes is evaluated independently and exchanges the information about the selected individual with other sub-populations based on the neighborhood topology. Neighborhood topologies commonly used when determining the direction of the information exchange are given In the Fig. 2 [29, 32, 33]. However, when a population based meta-heuristic is parallelized using this type of computation model, the speedup and performance of the algorithm change with the selected neighborhood topology, number of sub-colonies, communication interval between sub-colonies and types of information being distributed between compute nodes [29].

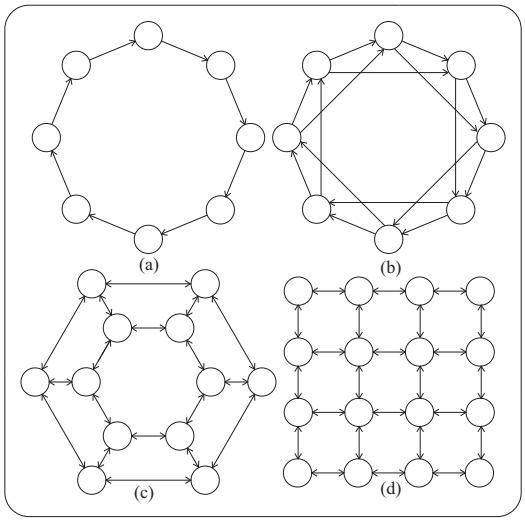

Fig. 1. Commonly used migration topologies: (a)Ring, (b)Ring 1+2, (c)Torus, (d)Lattice

Although all of these topologies have some advantages, deciding which solution to be exchanged should be main concern for increasing performance of the algorithm without deteriorating the speedup and efficiency. In the vast majority of these topologies, the worst solution or solutions found in a sub-population is replaced with the best solution or solutions of the topological neighbor sub-population or neighbor sub-populations based on the used communication schema [23, 29]. Best solutions found in each sub-population might be seen as a convenient migrants. But some situations, getting the best food source from the neighbor could not be enough to reflect the properties of its other solutions. 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, occurrence 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.

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 sub-population is combined with randomly chosen food source by changing the parameters of the best food source with the more efficient parameters taken from randomly chosen food source. By utilizing this kind of cooperative schema, food sources to be exchanged between neighbor sub-colonies carry more information about 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. If a population or colony consists 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 below.

Algorithm 2 Cooperative model based ABC algorithm

Initialization:
  Assign values to $limit$ and MCN parameters.
  Generate initial food sources (SN) by using Eq. 1.
Repeat
  Employed Bee Phase:
    Send employed bees to new food sources by using Eq. 2.
  Onlooker Bee Phase:
    Find probability values of each food source by using Eq. 3.
    Determine selected food source using p_i values.
    Send selected onlooker bee to food source by using Eq. 2.
  Scout Bee Phase:
    Determine the abandoned food source using $limit$ values.
    Send a scout bee for this abandoned food source.
  if Migration period is reached **then**
    $X_{Random} \leftarrow$ random food source in the compute node.
    $X_{Best}, X_{Coop} \leftarrow$ best food source in the compute node.
    for $j \leftarrow 1...D$ **do**
      Change $X_{Coop,j}$ with $X_{Random,j}$.
      Calculate fitness values of new X_{Coop}.
      if $fitness(X_{Coop}) \leq fitness(X_{Best})$ **then**
        Change $X_{Coop,j}$ with $X_{Best,j}$
      end if
    end for
    Send X_{Coop} to the neighbor sub-population.
  end if
  Memorize best food source found so far.
Until Maximum cycle number is reached

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 Table 1. Sphere function is a convex, unimodal function which has no local minimum except the global one. Rastrigin function was constructed from Sphere by adding a cosine modulator term to produce many local minima. Griewank function has a product term and number of local optima increases with the dimensionality. Rosenbrock valley is one of the most difficult optimization problem. Its global optimum is inside a long, narrow, parabolic shaped flat valley. For the experiments, the size of the bee colony was chosen as 160. The dimension on each function was set 400 and the value of $limit$ was taken as equal to the number of parameters. The proposed method was implemented in C using Open-MPI library. All of our tests have been performed on the cluster which consist of compute nodes powered by Intel(R) i5 4670 with 2 GB of RAM. In all experiments, the maximum number of iterations was set 2000. Migration topology used in the parallel implementations was ring and migration interval which

Table 1. Benchmark functions used in experiments

Function	Range	Formulation	Global Min.
Sphere	[-100, 100]	$f_1(\vec{x}) = \sum_{i=1}^{D} \left(x_i^2\right)$	$f_1(\vec{x}) = 0$
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]	$f_4(\vec{x}) = \sum_{i=1}^{D-1}\left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$	$f_3(\vec{x}) = 0$
Rastrigin	[-5.12, 5.12]	$f_3(\vec{x}) = \sum_{i=1}^{D}\left(x_i^2 - 10cos\left(2\pi x_i\right) + 10\right)$	$f_4(\vec{x}) = 0$
Dixon-Price	[-10, 10]	$f_5(\vec{x}) = (x_1 - 1)^2 + \sum_{i=2}^{D}\left(i\left(2x_i^2 - x_{i-1}\right)^2\right)$	$f_5(\vec{x}) = 0$

Table 2. Comparison results of ABC and parallel implementations of ABC algorithm on two processors

Functions	ABC		Ring-ABC		Coop-Ring ABC	
	Mean	Std	Mean	Std	Mean	Std
f_1	3.229954e+03	2.473820e+03	3.266018e+04	6.084122e+03	1.034880e-01	1.150681e-01
f_2	3.535124e+01	2.409915e+01	2.897337e+02	7.884458e+01	3.432151e-02	3.217872e-02
f_3	1.807207e+05	1.491821e+05	4.000289e+07	2.388123e+07	1.313634e+03	5.630211e+02
f_4	8.502711e+02	3.983880e+01	1.216464e+03	5.327817e+01	2.335463e+01	1.294404e+01
f_5	1.044298e+04	1.038023e+04	3.446708e+06	2.190490e+06	5.391258e+02	9.035605e+01

Table 3. Comparison results of ABC and parallel implementations of ABC algorithm on four processors

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

Table 4. Speedup and efficiency for Ring and Cooperative Ring ABC algorithms on two processors

Functions	ABC	Ring-ABC	Coop-Ring ABC	Speedups		Efficiency	
	Time(s)	Time(s)	Time(s)	ABC/Ring	ABC/Coop-Ring	Ring	Coop-Ring
f_1	0.335517	0.206929	0.230228	1.6214	1.4573	0,8107	0,7286
f_2	9.541212	4.791809	4.951232	1.9912	1.9270	0,9956	0,9628
f_3	0.742329	0.406193	0.456462	1.8275	1.6263	0,9137	0,8131
f_4	4.167828	2.098474	2.137868	1.9861	1.9495	0,9930	0,9747
f_5	2.708567	1.390291	1.566196	1.9482	1.7294	0,9741	0,8647

Table 5. Speedup and efficiency for Ring and Cooperative Ring ABC algorithms on four processors

Functions	ABC	Ring-ABC	Coop-Ring ABC	Speedups		Efficiency	
	Time(s)	Time(s)	Time(s)	ABC/Ring	ABC/Coop-Ring	Ring	Coop-Ring
f_1	0.335517	0.106680	0.141885	3.1451	2.3647	0,7862	0,5911
f_2	9.541212	2.399059	2.501293	3.9771	3.8145	0,9942	0,9536
f_3	0.742329	0.204072	0.290406	3.6376	2.5562	0,9094	0,6390
f_4	4.167828	1.073781	1.124635	3.8815	3.7059	0,9703	0,9264
f_5	2.708567	0.693667	0.926330	3.9047	2.9240	0,9761	0,7310

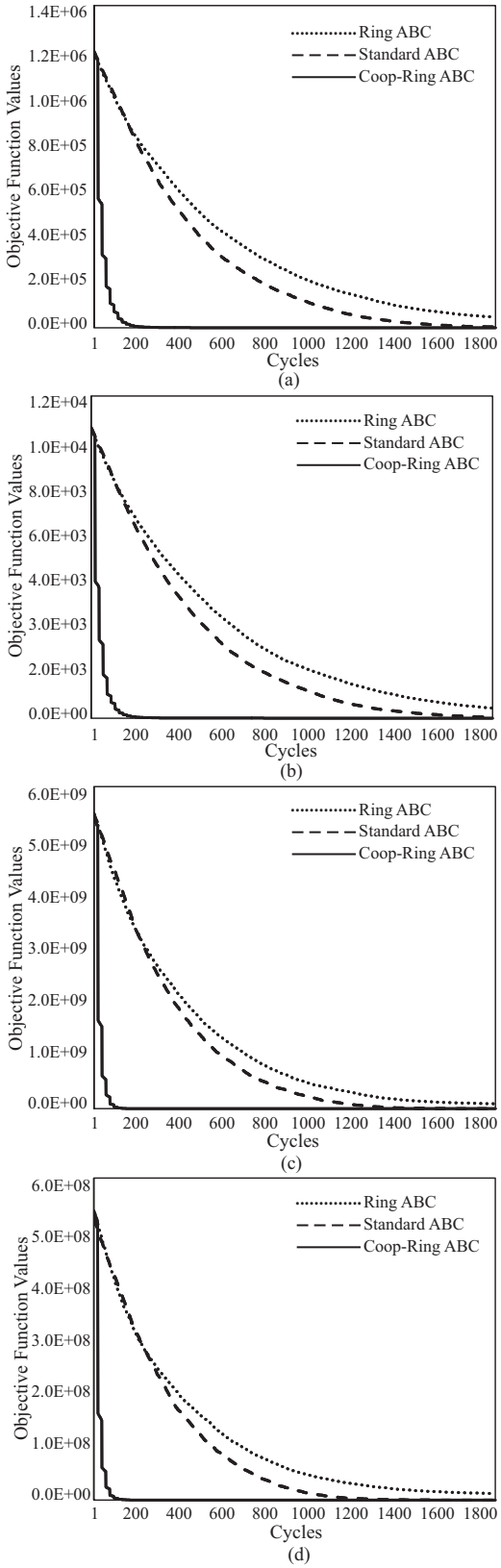

Fig. 2. Convergence characteristics of the standard ABC algorithm, Ring and Cooperative Ring ABC algorithms on four processors for (a) Sphere, (b) Griewank, (c) Rosenbrock, (d) Dixon-Price functions with 400 parameters

controls the frequency of the food source exchanging between sub-populations was set 20. Each of the experiments was repeated 20 times with different random seeds and the mean best values and standard deviation have been recorded.

From the simulation results given in the Table 2 and Table 3, it is clear that the mean best objective function values obtained by proposed cooperative model outperform the standard sequential ABC algorithm and ring schema based ABC algorithm. By distributing cooperative best food source between ring based neighbor sub-colonies, 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 sub-colonies has been maintained with the emigrant food sources that reflects the overall properties of the its original population. The effect of the proposed schema on the convergence speed of the algorithm could be seen in the Fig. 2. When these figures are examined, all of them present a remarkable difference between Coop-Ring ABC and the other ones.

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 parallel execution time and efficiency value is ration of speedup to the number of processors used. Optimum value of the speedup metric is equal to the number of processors and optimum value of the efficiency is equal to 1. In Table 4 and Table 5, average total running time for 20 different runs, speedup and efficiency values are given respectively. In the calculation of the average running time for the parallel ABC algorithms, total elapsed time for the slowest processor has been used. Since the generation of cooperative food source require a comparison between all parameters of the local best food source and a randomly determined food source for each sub-colony, the speedup and efficiency values of the Coop-Ring ABC algorithms lag slightly behind the Ring ABC algorithm especially for the functions that are less compute expensive.

5. Conclusion

In this paper, a new creation schema for the emigrant food source between neighbor sub-colonies was 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 the 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 sub-colony 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.*, Kayseri, 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 Paradigms,* 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 Theoretical 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 (ABC) 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", *Signal Processing and Communications Applications Conference (SIU)*, Eskisehir, 2007 pp: 1-4.

[8] D. Karaoga, C. Ozturk, "Neural network training by artificial bee colony algorithm on pattern classificiation", *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: 6056-6065, 2011.

[11] X. Lei, J. Sun, X. Xu, L. Guo, "Artificial bee colony algorithm for solving multiple sequence alignment", *Bio-Inspired Computing: Theories and Applications (BIC-TA)*, Changsha, 2010, pp: 337-342.

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

[13] C.M.V. Benitez, H.S. Lopes, "Parallel artificial bee colony approaches for protein structure prediction using the 3dhp-sc model", *Intelligence Distributed Computing IV Studies in Computational Intelligence*, Springer Berlin Heidelberg, pp: 255-264, 2010.

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

[15] C. Ozturk, E. Hancer, 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 (qABC) algorithm and its performance on optimization problems", *Appl Soft Comput*, vol. 23, pp: 227-238, 2014.

[22] H. Narasimhan, "Parallel artificial bee colony (PABC) algorithm", *World Congress on Nature and Biologically Inspired Computing (NaBIC), 2009 World Congress*, Coimbatore, India, 2009, pp: 306-311.

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

[24] R. Luo, T. Pan, P. Tsai, J. Pan, "Parallelized artificial bee colony algorithm with ripple-communication strategy", *Genetic and Evolutionary Computing (ICGEC) 2010 Fourth International Conference*, Shenzen, 2010, pp: 350-353.

[25] M. Subotic, M. Tuba, N. Stanarevic, "Parallelization of the artificial bee colony (ABC) algorithm", *Proceedings of the 11th WSEAS international conference on neural networks and 11th WSEAS international conference on evolutionary computing and 11th WSEAS international conference on Fuzzy systems*, 2010, pp: 191-196.

[26] M. Subotic, M. Tuba, N. Stanarevic, "Different 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, "Chapter Handbook of Swarm Intelligence-Parallel approaches for the artificial bee colony algorithm", *Springer-Verlag*, Berlin/Heidelberg, 2011.

[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 (Abc) algorithm", *Journal of Global Optimization*, vol. 39, pp: 459-471, 2007.

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

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

[33] P. Pacheco, "An Introduction to parallel Programming", *Morgan Kaufmann*, Burlington, USA, 2011.