An International Journal of Optimization and Control: Theories & Applications
ISSN:2146-0957 eISSN:2146-5703
Vol.9, No.1, pp.1-10 (2019)
http://doi.org/10.11121/ijocta.01.2019.00576

RESEARCH ARTICLE

Deployment in wireless sensor networks by parallel and cooperative parallel artificial bee colony algorithms

Selcuk Aslan

Department of Computer Engineering, Ondokuz Mayıs University, Samsun, Turkey
selcuk.aslan@omu.edu.tr

ARTICLE INFO

Article History:
Received 09 January 2018
Accepted 22 September 2018
Available 15 October 2018

Keywords:
Parallelization
ABC algorithm
Wireless sensor deployment

AMS Classification 2010:
65K10

ABSTRACT

Increasing number of cores in a processor chip and decreasing cost of distributed memory based system setup have led to emerge of a new work theme in which the main concern focused on the parallelization of the well-known algorithmic approaches for utilizing the computational power of the current architectures. In this study, the performances of the conventional parallel and cooperative model based parallel Artificial Bee Colony (ABC) algorithms on the deployment problem related to the wireless sensor networks were investigated. The results obtained from the experimental studies showed that parallelized ABC algorithm with the cooperative model is capable of finding similar or better coverage ratios with the increased convergence speeds than its serial counterpart and parallelized implementation in which the emigrant is chosen as the best food source in the current subcolony.

(cc) BY

1. Introduction

Wireless sensor networks including hundreds or sometimes thousands of stationary or mobile nodes have been used various times for industrial or military projects [1, 2]. Each of the sensor nodes is capable of sending or receiving data packages and gathering information from the environment or objects being tracked. [1, 2]. However, sensor nodes have limited computing abilities and storage spaces, their detection ranges are restricted with properties of the sensing units and finally required power for sensing and communication is maintained by a small battery which can not be recharged or changed easily.

By considering all of these limitations and budget constraints, the configuration and settlement of a wireless sensor network should be made in order to maximize the life or utilization time of the network and the area of interest [1, 2]. The life time and successfully covered area of a wireless sensor network are directly related to the positions of the sensor nodes. If all the sensor nodes are deployed to the monitoring area in a straightforward manner that concerns the highest coverage ratio, the requirements for changing the positions of the mobile nodes by consuming extra energy from the internal battery decrease and the overall network life-time is substantially extended [1, 2]. With the increased understanding about the relationship between the positions of the sensors and efficiency of the network, studies on the deployment of sensor nodes have attracted the researchers and different approaches for solving the sensor deployment problem have been proposed.

When the studies about the sensor deployment problem are investigated, it is clearly seen that evolutionary computing techniques are commonly used. Bhondekar et al. used Genetic algorithm (GA) as a placement methodology of sensor nodes with different operating modes [3]. They tried to optimize a fitness function in which operational energy, number of unconnected sensors, number of overlapping cluster-in-charge, field coverage and number of sensors per cluster-in-charge

are used as constraints. While the operational energy, number of unconnected sensors, number of overlapping cluster-in-charge constraints should be minimized, field coverage and number of sensors per cluster-in-charge constraints should be maximized [3]. Okay and Ozdemir analyzed the performances of the Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D) and Fast and Elitist Genetic Algorithm (NSGA-II) on optimization of sensing coverage area and total travel distances of the mobile nodes [4]. Obtained results from the experimental studies with 25 mobile sensors tracking 50 targets distributed to a $100m \times 100m$ area showed that NSGA-II is produced more robust deployments compared to MOEA/D in terms of tracked objects [4].

Li and Lei proposed a sensor deployment technique based on the Particle Swarm Optimization (PSO) algorithm called IPSO [5]. Distribution of 40 mobile sensors to a $80m \times 80m$ grid area with IPSO algorithm significantly improved the coverage ratio calculated with the probabilistic detection model compared to the Virtual Force (VF) algorithm [5]. One of the first studies about the using ABC algorithm as a sensor deployment technique has been carried out by Ugdata et al [6]. Ugdata et al. modeled sensor deployment problem as a data clustering problem and number of sensor nodes was used on behalf of clusters and locations of the sensor nodes were matched with the centroids of clusters [6]. Ozturk et al. investigated solving capabilities of the ABC algorithm for dynamic deployment problem of wireless networks with the two different studies [7, 8]. In the first study of them, ABC algorithm was used in order to maximize the coverage ratio of the network containing 100 mobile sensors [7, 8]. In the second study, they compared ABC algorithm with the PSO algorithm on solving a dynamic deployment scenario in which 20 mobile sensors are tried to be positioned at the suitable locations within a $10,000m^2$ square region [7, 8]. Results from the experimental studies showed that ABC algorithm is capable of producing more qualified solutions than the PSO algorithm. Yu et al. solved deployment problem by utilizing a modified ABC algorithm named as FNF-BL-ABC [9]. In the FNF-BL-ABC algorithm, the original equation of the ABC algorithm used to generate candidate solutions for onlooker bee phase was changed with the forgetting (F) and neighbor (N) factors [9]. In addition to these, they introduced a probabilistic model called back propagation learning (BL) for determining whether a solution is abandoned or not in the scout bee phase. Simulation results in an ideal area and an area with obstacles showed that TNF-BL-ABC algorithm produces better coverage ratios than standard ABC algorithm and increases the convergence speed [9]. Yadav et al. changed the search equation used by the employed and onlooker bee phases of the standard ABC algorithm and tested the proposed ABC algorithm variant for dynamic positioning of sensor networks [10].

In this study, the performances of the parallelized ABC algorithms powered with the conventional and cooperative emigrant creation strategies for solving the deployment problem of sensor networks were analyzed. The improving effects of the cooperative emigrant creation strategy already seen in numerical optimization problems were also investigated through sensor deployment problem. The rest of the paper is organized as follows: In the second section, definition of the sensor deployment problem, coverage calculation and sensor detection approach called binary detection are given. Fundamental steps of the ABC algorithm and its parallelization according to the mentioned emigrant creation strategies are summarized in third and fourth sections, respectively. Experimental studies with different control parameters are presented in fifth section. Finally, conclusions and future works are given in the sixth section.

2. Deployment problem in wireless sensor networks

When a wireless sensor network is established, the main purposes of the settlement are to maximize the utilization period of the network and the area where the sensors successfully in communication with each other by sending information obtained from the tracked objects or environmental variables [5–9]. To maximize these two conflicting objectives, exact positions of the mobile and stationary sensor nodes should be determined carefully. However, there is usually no priori information about the area of interest or the targets being tracked [5–9].

By considering all of these limitations, sensor deployment can be defined as a problem for which the coverage of the network is maximized by correctly positioning sensor nodes. When the sensor nodes are deployed, the coverage ratio of the network that shows the percentage of the successfully covered area is calculated as in the Eq. (1). In the Eq. (1), c_i is the coverage of the *ith* sensor in the set of sensors S and A is the size of the area [5–9].

$$CR = \frac{\bigcup c_i}{A}, i \epsilon S \tag{1}$$

If the area of interest is divided into equally sized subareas or grids, and P is a point corresponds to the corner of a grid at position (x, y), the Euclidean distance between the point P and the sensor s_i positioned at (x_i, y_i) is used to decide whether point P is in detection range of sensor s_i or not [5–9]. By taking the detection range of the sensor s_i as r and the Euclidean distance between the P and s_i as $d(P, s_i)$, the coverage of point P by s_i, $c_p(s_i)$, is equal to 1 if $d(P, s_i)$ is less than r, otherwise $c_p(s_i)$ is equal to 0. The binary sensor detection model used in the coverage calculation is given in the Eq. (2) for P and s_i [5–9].

$$c_i = \begin{Bmatrix} 1, & d(P, c_i) < r \\ 0, & d(P, c_i) \geq r \end{Bmatrix} \tag{2}$$

3. ABC algorithm and its adaptation to sensor deployment problem

In a real honey bee colony, an intelligent foraging behaviour is carried out by three groups of bees called employed, onlooker and scout bees, respectively [11–13]. Employed bees are responsible for finding new food sources around the previously visited ones and carry nectar to the hive. When an employed bee turns back to the hive, she shares the information about the nectar quality of the memorized food source, location and distance to the hive with the onlooker bees [11–13]. Onlooker bees wait on the hive and select food sources introduced by the employed bees. However, selection of a food source by an onlooker is actually not a random operation. If a food source introduced by an employed is rich in terms of nectar, it is highly possible that it attracts more onlooker bees compared with the poor sources [11–13]. After an onlooker bee selects a food source, she becomes an employed and continues the foraging operation as an employed. The final group of bees consists of scout bees and scout bees randomly search the environment to find an undiscovered food source.

By considering intelligent job division and foraging behaviours of bee colonies, Karaboga proposed a new population based optimization algorithm called ABC algorithm [11–13]. In ABC algorithm, positions of the food sources correspond to the possible solutions of the interested problem and the nectar quality of a food source is directly related to the appropriateness of the solution. ABC algorithm starts its optimization operations by randomly generating a set of food sources [14–16]. Assume that there are SN different food sources each of them contains D parameters, the jth parameter of the ith food source, shortly x_{ij}, can be generated between lower bound x_j^{min} and upper bound x_j^{max} as described in Eq. (3) [14–16].

$$x_{ij} = x_j^{min} + rand(0, 1)(x_j^{max} - x_j^{min}) \tag{3}$$

When solving sensor deployment problem, a food source is matched with the positions of the sensors belonging to the created network and a food source or solution containing S wireless sensors can be represented by a specialized D dimensional vector in which each element is filled with location information of the sensor. In Fig. 1, a food source is illustrated for deployment of D wireless sensors into a two dimensional area.

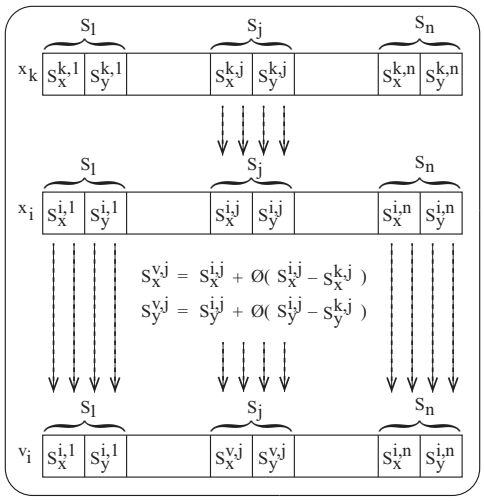


Figure 1. Representation of a solution for sensor deployment problem.

After generating initial food sources, each food source is associated only one employed bee. An employed bee is responsible with producing a candidate solution in the vicinity of the memorized food source by utilizing the Eq. (4) [17–19].

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{4}$$

In Eq. (4), v_{ij} is the jth parameter of the candidate food source v_i. It should be noted that v_i is same with the x_i food source except the jth parameter. x_{ij} and x_{kj} are the jth parameters of the x_i and x_k solutions, respectively [19–23]. Finally, θ is a random coefficient between -1 and 1. If the $fit(v_i)$ fitness value of the v_i solution calculated by using the $obj(v_i)$ objective function value for a maximization problem as in the Eq. (5) is higher than the $fit(x_i)$ fitness value of the x_i food source, x_i food source is replaced with the v_i food source and the trial counter $trial_i$ showing how many times the x_i food source is not improved is set to zero. Otherwise, the same counter is incremented by one and its value is used to make a

decision whether that food source is consumed or not [19–23].

$$fit(x_i) = \begin{cases} 1 + |obj(x_i)|; & obj(x_i) > 0 \\ 1/(1 + obj(x_i)); & obj(x_i) \leqslant 0 \end{cases} \quad (5)$$

When all the employed bees complete their operations and turn back to the hive, they share the information about the memorized food sources with the onlooker bees as mentioned before. Onlooker bees waiting on the hive select food sources and become employed foragers. However, each solution introduced by employed bees does not have equal chance for selection and qualified sources attract more onlookers. The relationship between choosability of a food source and its quality is modeled in ABC algorithm by assigning selection probability for each food source as calculated in Eq. (6) [19–23]. In Eq. (6), $p(x_i)$ shows the selection probability of the x_i solution with $fit(x_i)$ fitness value and it is clearly seen that $p(x_i)$ increases with the higher values of $fit(x_i)$. After a food source is chosen by an onlooker bee, this onlooker becomes an employed and produce a candidate solution using Eq. (4) [19–23].

$$p(x_i) = \frac{fit(x_i)}{\sum_j^{SN} fit(x_j)} \quad (6)$$

If a food source is not improved within employed and onlooker bee phases, a decision whether this food source is still consumed in the next cycle or not should be made to maintain the diversity of the solution set. In ABC algorithm, this decision is made by comparing the trial counters of the food sources with a control parameter called *limit*. A food source for which its trial counter exceeds the value of the *limit* parameter at most is abandoned and a scout bee is sent from the hive to discover a new food source as in the Eq. (3). In order to adjust exploration and exploitation characteristics of the algorithm, value of the *limit* parameter should be chosen carefully. For determining appropriate *limit* parameter of SN food sources when solving a D dimensional optimization problem, the formulation in Eq. (7) can be used [19–23].

$$\lceil a \times SN \times D \rceil \; and \; a \in \mathbb{Q}^+ \quad (7)$$

By considering the properties of the employed, onlooker and scout bee phases, the fundamental steps of the ABC algorithm and cyclical relationship between the mentioned bee phases are summarized in the Fig. 2.

```
1: Initialization:
2:   Assign values to limit and MFE parameters.
3:   Generate SN initial food source by using Eq. (3).
4:   Set evalCounter to zero.
5: Repeat
6: //Employed bee phase
7:    for i ← 1 ... SN do
8:     if evalCounter < MFE then
9:       Generate new solution x_new by using Eq. (4).
10:      Calculate fitness value of new solution.
11:      if fit(x_new) > fit(x_i) then
12:      Change x_i with x_new
13:      end if
14:      evalCounter ← evalCounter + 1
15:    end if
16:   end for
17: //Employed bee phase
18: //Onlooker bee phase
19:   sentBees ← 0, current ← 1
20:   Find probability values for each source by using Eq. (6).
21:   while sentBees ≠ SN and evalCounter < MFE do
22:    if p_current > rand(0,1) then
23:      sentBees ← sendBees + 1
24:      Generate new solution x_new by using Eq. (4).
25:      Calculate fitness value of new solution.
26:      if fit(x_new) > fit(x_i) then
27:       Change x_i with x_new
28:      end if
29:      evalCounter ← evalCounter + 1
30:    end if
31:    current ← ( current + 1 ) mod SN
32:   end while
33: //Onlooker bee phase
34: //Scout bee phase
35:   if evalCounter < MFE then
36:    Determine the abandoned food source using limit value.
37:    Generate a new source for the abandoned one by using Eq. (3).
38:    evalCounter ← evalCounter + 1
39:   end if
40: Until MFE is reached.
41: //Scout bee phase
```

Figure 2. Fundamental steps of the ABC algorithm.

4. Parallelization of ABC algorithm with conventional and cooperative model

Population based optimization algorithms including ABC algorithm are generally suitable for parallelization on distributed or shared memory based architectures. However, some steps of the algorithms require sequential operations and a limited set of modifications on the fundamental workflow of them should be made when they are tried to be parallelized. Dividing the whole colony into equally sized small colonies and evaluating them simultaneously on the different computing units are probably the most preferred parallelization approach [24–26]. However, number of bees in computing units is usually not enough compared to the serial implementations on single computing unit and parallelization does not go beyond a method that only focusing improvement on the execution times [24–26].

In order to address the problem to do with the number of bees in computing units, some solutions or individuals are migrated between subcolonies. The best solutions in each subcolony are the appropriate emigrant candidates and they usually change with the worst solutions in the neighbor subcolonies. This type of migration schema is the common part of the studies devoted to the parallelization and can be thought as the conventional approach [24–26]. However, if the best solutions can not be improved between subsequent migration periods, two or more copies of the same emigrant can be seen in the neighbor subcolony.

For increasing the efficiency of the emigrant solution and ensuring that the different emigrants are sent, an emigrant solution should be powered with other solution or solutions before it is sent to the neighbor subcolony. The mentioned idea about powering the best solution in a subcolony before migration is the main motivation of the cooperative model. In cooperative model, the best food source in a subcolony is strengthened by the more convenient parameters of the randomly chosen food source in the same subcolony. The Fig. 3 below illustrates the fundamental steps of the cooperative model in which neighborhood between subcolonies is determined by the ring topology.

Initialization:
  Assign values to *limit* and *MFE* parameters.
  Generate *SN* initial food source by using Eq. (1).
  Set *evalCounter* to zero and define a *migPeriod.*
  Determine *numOfSubCol*.
Repeat
After completion of a phase-triple
  if *migPeriod* is reached **then**
    if *evalCounter* < MFE **then**
      subColony ← index of current subcolony.
      x_{random} ← a random source in the *(subColony)*th subcolony.
      x_{best}, x_{coop} ← the best source in the *(subColony)*th subcolony.
      for i ← 1 ... D **do**
        Change $x_{coop,i}$ with $x_{random,i}$
        Calculate the fitness value of x_{coop}
        evalCounter ← *evalCounter* + 1
        if fit(x_{coop}) < fit(x_{best}) **then**
          Change $x_{coop,i}$ with $x_{best,i}$
        end if
      end for
  Send x_{coop} to the *((subColony + 1)* mod *numOfSubCol)*th subcolony.
Until *MFE* is reached

Figure 3. Fundamental steps of the parallel ABC algorithm with cooperative model.

As seen from the fundamental steps of the parallel ABC algorithm with cooperative model, the best food source chosen as an emigrant for the current migration period is modified with the parameters of the randomly selected food source. If the *ith* parameter of the randomly selected food source increases the fitness value of the best food source, the *ith* parameter of the best food source is replaced with the corresponding parameter of the randomly selected food source. By utilizing this type of emigrant creation schema, the probability of sending qualified food sources as emigrants and the chance for consumption more qualified solutions are significantly increased.

However, it should be noted that generation of cooperative emigrant requires D times more fitness evaluations compared to the conventional emigrant creation schema. If the migration period and neighborhood topology are chosen by considering the computational burden of the cooperative emigrant creation approach, the speedup and efficiency values of the parallelized ABC algorithm with cooperative model get closer to the speedup and efficiency values of the parallelized ABC algorithm in which the emigrant is determined as the local best food source in the subcolony and then it is sent to the neighbor subcolony without modification.

5. Experimental studies

In order to analyze the performance of the conventional and cooperative emigrant creation schema for solving the sensor deployment problem, a set of experimental studies has bee carried out with 100 mobile sensors that should be positioned at the suitable locations on a $100m \times 100m$ area by considering the maximization of the coverage. For serial ABC algorithm, sABC algorithm, parallel ABC algorithm with the conventional emigrant creation strategy, pABC algorithm, and parallel ABC algorithm with the cooperative emigrant creation strategy, coop-pABC algorithm, the colony size was set to 20 and the *limit* parameter was chosen as 100 for the experiments [7, 8].

Neighborhood topology of the pABC and coop-pABC algorithms was ring and for each subcolony only one emigrant was generated. When an emigrant was sent to its neighbor subcolony, it was replaced with the worst solution found in the neighbor subcolony. The migration period (migration rate) that determines the frequency of the communication between subcolonies was set to 20 which means that after completion of a 20 employed-onlooker-scout bee phase triple, subcolonies exchange their emigrants according to the used neighborhood topology. sABC algorithm, pABC and coop-pABC algorithms with four subcolonies were tested independently until the maximum evaluation number reached to 1, 000, 2, 000 and 10, 000 on a system equipped with Intel i5 750

processor and 4 GB of RAM. sABC, pABC and coop-pABC algorithms were coded in C programming language and the required synchronization between subcolonies or processor cores were maintained by using the built-in function in pthreads library. Each of the algorithm was run 20 different times with random seeds and the means best coverage ratios and standard deviations related to the 20 runs were recorded and given in the Tables 1-3.

When the results given in the Tables 1-3 are investigated it is clearly seen that the the coop-pABC algorithm is capable of producing better mean coverage ratios compared to the pABC algorithm for all of the three experimental cases and the sABC algorithm for the two of three different experimental cases. By starting distribution of the cooperative emigrants, parallelized ABC algorithm improves the qualities of the solutions in each subcolony. Even though the differences between mean best coverage ratios of the algorithms are relatively small, the complex structure of the deployment problem and the difficulty on improving coverage value after determining positions of the some sensors should be remembered.

Table 1. Coverage values obtained by the sABC and pABC.

Evaluations	sABC		pABC	
	Mean	Std.Dev.	Mean	Std.Dev.
1,000	**0.88257**	**0.00410**	0.87507	0.00594
2,000	**0.91207**	0.00638	0.90887	**0.00483**
10,000	0.96755	**0.00226**	**0.96904**	0.00372

Table 2. Coverage values obtained by the sABC and coop-pABC.

Evaluations	sABC		coop-pABC	
	Mean	Std.Dev.	Mean	Std.Dev
1,000	**0.88257**	**0.00410**	0.87970	0.00457
2,000	0.91207	0.00638	**0.91530**	**0.00362**
10,000	0.96755	**0.00226**	**0.97063**	0.00553

Table 3. Coverage values obtained by the pABC and coop-pABC.

Evaluations	pABC		coop-pABC	
	Mean	Std.Dev.	Mean	Std.Dev
1,000	0.87507	0.00594	**0.87970**	**0.00457**
2,000	0.90887	0.00483	**0.91530**	**0.00362**
10,000	0.96904	**0.00372**	**0.97063**	0.00553

One of the main purposed with the parallelization of an algorithm is actually decreasing the execution times compared to the its serial implementation while protecting the qualities of the final solutions or results. For measuring the gain in the execution times, two important metrics called speedup and efficiency are commonly used. Speedup measure can be explained as a ratio between average execution times between serial and parallel implementations of the same algorithm and its maximum value can be equal to the number of cores or computing nodes of the cluster. If the speedup value of the parallelization is equal to the number of core or computing nodes, it is said that parallelization is linear. Efficiency metric is defined as a ratio between speedup and number of computing units used in the parallelization schema.

If the parallelization overhead stemmed from the mechanism such as synchronization, mutual exclusion can not be neglected, the maximum value of the efficiency can be relatively close to one. In Tables 4-7, average execution times of the sABC, pABC and coop-pABC algorithms, speedup and efficiency values for parallel implementations are given. As seen from the results given in Tables 4-7, conventional emigrant creation strategy reaches more desired speedup and efficiency values when compared to the cooperative emigrant creation strategy based parallelization approach. If the reduction in execution time is the main concern of the parallelization, the migration period should be carefully chosen to balance the qualities of the final solutions and speedup-efficiency values.

Table 4. Average execution times for sABC and pABC.

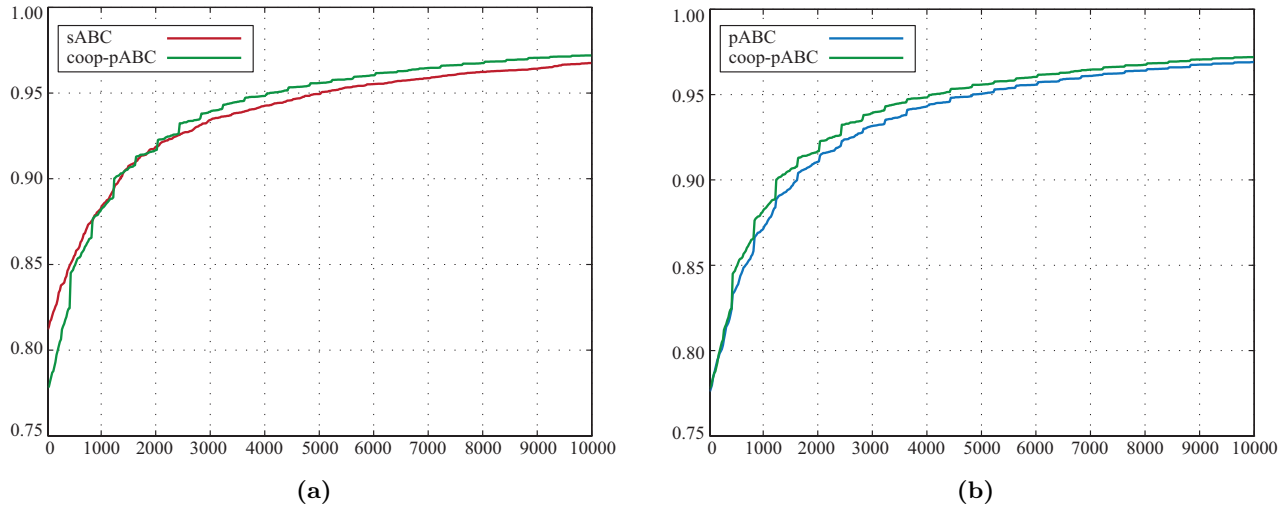
Evaluations	sABC		pABC	
	Mean	Std.Dev.	Mean	Std.Dev.
1,000	48.63031	2.01138	13.39241	0.60145
2,000	94.37129	3.73157	27.23307	1.18997
10,000	437.05957	11.35073	124.54752	3.29203

Table 5. Speedup and efficiency values of pABC.

Evaluations	ABC and pABC Algorithms	
	Speedup	Efficiency
1,000	3.63118	0.90779
2,000	3.46531	0.86633
10,000	3.50917	0.87729

Table 6. Average execution times for sABC and coop-pABC.

Evaluations	sABC		coop-pABC	
	Mean	Std.Dev.	Mean	Std.Dev.
1,000	48.63031	2.01138	18.78291	0.69383
2,000	94.37129	3.73157	37.47984	1.55957
10,000	437.05957	11.35073	187.71349	3.48997



(a) **(b)**

Figure 4. Convergence curves of sABC and coop-pABC (a) and pABC and coop-pABC (b).

Table 7. Speedup and efficiency values for coop-pABC.

Evaluations	ABC and coop-pABC Algorithms	
	Speedup	Efficiency
1,000	2.58907	0.64726
2,000	2.51792	0.62948
10,000	2.32833	0.58208

Another comparison between sABC and parallel ABC algorithms can be made about the convergence characteristics of them illustrated in Fig. 4 below. When the convergence curves given in Fig. 4 are investigated, it is clearly seen that convergence performance of the coop-pABC algorithm is better than the convergence performances of the sABC and pABC algorithms. Although the initial mean best coverage values of parallel ABC algorithms is less than the initial mean best coverage value of sABC algorithm, they reached sABC algorithm before completion of the first $1,000$ evaluations and then start to produce more eligible mean best coverage values than sABC algorithm.

In order to make a visual investigation how the sensors are positioned by the sABC, pABC and coop-pABC algorithms and how the areas being covered change for the different termination conditions, the Figs. 5-10 should be utilized. As easily seen from the Figs. 5-10, successfully coveraged areas by the algorithms are rational with the total number of evaluations. With the completion of the $1,000$ evaluations, both serial and parallel implementations of the ABC algorithm produce deployments in which some sensors are located relatively close positions and coverage areas of them are overlapped. However, when the number of evaluations is set to $10,000$, overlapped sensors are scattered more robustly and coop-pABC algorithm outperforms sABC and pABC algorithms in terms of mean best coverage ratios.

Deciding whether coop-pABC algorithm can be interchangeable with the sABC or pABC algorithms, an information extracted from a statistical test should be utilized. For this purpose, a nonparametric test called Wilcoxon signed rank test is used with the significance level (p) less than 0.05. From the test results given in the Table 8 for $10,000$ fitness evaluations, it is seen that there is no significant difference between serial and parallel implementations of the ABC algorithm even though coop-pABC algorithm produces better mean best coverage values and parallel implementations can be used on behalf of sABC algorithm if the running environments are designed for utilizing the multi-core or multi-node based architectures.

Table 8. Statistical comparison between ABC algorithms.

Test statistics	sABC/coop-pABC	pABC/coop-pABC
Z-Value	-1.784925	-1.274946
p-Value	0.074274	0.202328
Sign.	-	-

6. Conclusion

In this study, ABC algorithm was parallelized for running on a multi-core processor and it performance was tested on solving wireless sensor deployment problem. Parallelized ABC algorithm by dividing the whole bee colony into subcolonies running simultaneously was powered with the cooperative emigrant creation approach and the results obtained with the mentioned ABC algorithm were compared to the results obtained with

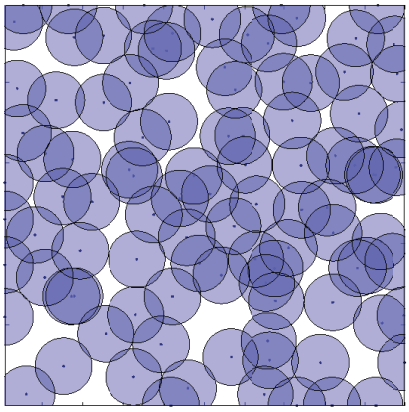


Figure 5. The best coverage of sABC for 1,000 evaluations

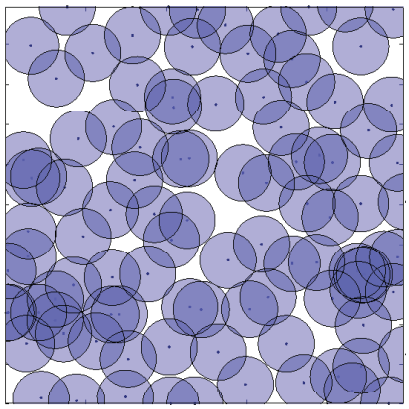


Figure 6. The best coverage of pABC for 1,000 evaluations

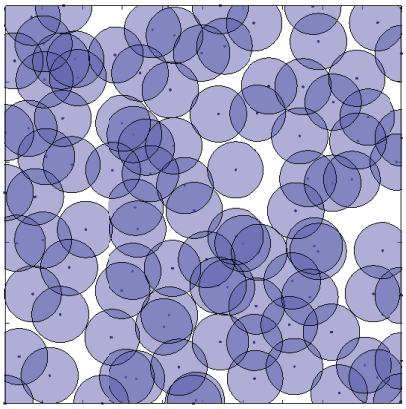


Figure 7. The best coverage of coop-pABC for 1,000 eveluations

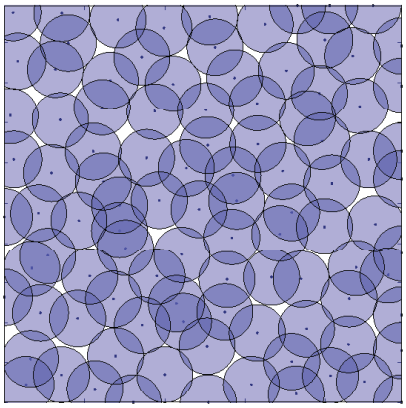


Figure 8. The best coverage of sABC for 10,000 evaluations

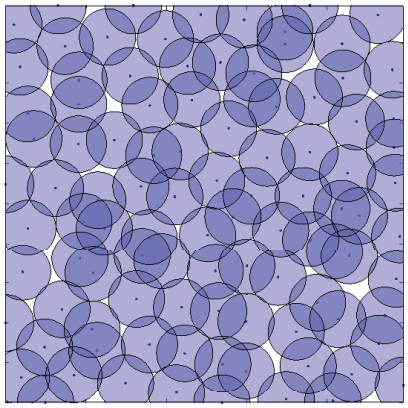


Figure 9. The best coverage of pABC for 10,000 evaluations

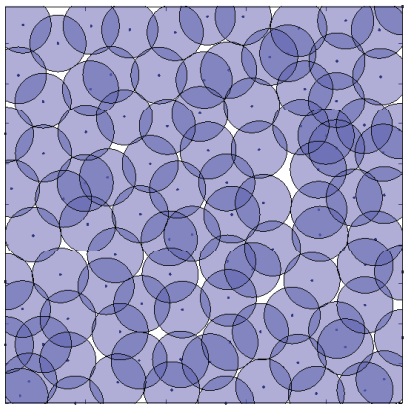


Figure 10. The best coverage of coop-pABC for 10,000 eveluations

standard serial and conventional parallel ABC algorithms. Comparative studies showed that cooperative model is still capable of increasing convergence speed and improving solution qualities of parallel ABC algorithm for sensor deployment problem as seen in the numerical benchmark problems by adding extra computational burden that changes directly with the migration period to the execution time of the algorithm.

References

[1] Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38, 393-422.

[2] Chakrabarty, K., Iyengar, S.S., Qi, H., Cho, E. (2002). Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers*, 51, 1448-1453.

[3] Bhondekar, A.P., Vig, R., Singla, M.L., C. Ghanshyam, Kapur, P. (2009). Genetic algorithm based node placement methodology for wireless sensor networks. *Proceedings of the International Multiconference on Engineers and Computer Scientists*, 1, 18-20.

[4] Okay, F.Y., Ozdemir, S. (2015). Kablosuz algılayıcı aglarda kapsama alanının cok amalı evrimsel algoritmalar ile artırılması. *Journal of the Faculty of Engineering & Architecture of Gazi University*, 30, 143-153.

[5] Li, Z., Lei, L. (2009). Sensor node deployment in wireless sensor networks based on improved particle swarm optimization. *Applied Superconductivity and Electromagnetic Devices*, 215-217.

[6] Udgata, S.K., Sabat, S.L., Mini, S. (2009). Sensor deployment in irregular terrain using artificial bee colony algorithm. *Nature & Biologically Inspired Computing*, 1309-1314 .

[7] Ozturk, C., Karaboga, D., Gorkemli, B. (2011). Probabilistic dynamic deployment of wireless sensor networks by artificial bee colony algorithm. *Sensors*, 11, 6056-6065 .

[8] Ozturk, C., Karaboga, D., Gorkemli, B. (2012). Artificial bee colony algorithm for dynamic deployment of wireless sensor networks. *Turkish Journal of Electrical Engineering & Computer Sciences*, 20, 255-262.

[9] Yu, X., Zhang, J., Fan, J., Zhang, T. (2013). A faster convergence artificial bee colony algorithm in sensor deployment for wireless sensor networks. *International Journal of Distributed Sensor Networks*, 9, 1-15.

[10] Yadav, R.K., Gupdaa, D., Lobiyal, D.K. (2017). Dynamic positionin of mobile sensors using modified artificial bee colony algorithm in wireless sensor networks. *International Journal of Control Theory and Applications*, 10, 167-176.

[11] Karaboga, D., Akay, B. (2009). A suvery: algorithms simulating bee swarm intelligence. *Artificial Intelligence Reviews*, 31, 233-253.

[12] Bansal, J.C., Sharma, H., Jadon, S.S. (2013). Artificial bee colony algorithm: a survey. *International Journal of Advanced Intelligence*, 5, 123-159.

[13] Bolaji, A.L., Khader, A.T., Al-betar, M.A., Awadallah, M.A. (2013). Artificial bee colony algorithm, its variants and applications: a survey. *Journal of Theorical and Applied Information Technology*, 47, 434-459.

[14] Karaboga, D., Akay, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony algorithm. *Journal of Global Optimization*, 39, 459-471.

[15] Karaboga, D., Akay, B. (2008). On the performance of artificial bee colony algorithm. *Applied Soft Computing*, 8, 687-697.

[16] Akay, B., Karaboga, D. (2012). Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, 23, 1001-1014.

[17] Celik, M., Koylu F., Karaboga, D. (2015). CoABCMiner: an algorithm for cooperative rule classification system based on artificial bee colony algorithm. *International Journal of Artificial Intelligence Tools*, 24, 1-50.

[18] Karaboga, D., Aslan, S. (2016). Best supported emigrant creation for parallel implementation of artificial bee colony algorithm. *IU-Journal of Electrical & Electronics Engineering*, 16, 2055-2064.

[19] Badem, H., Basturk, A., Caliskan, A., Yuksel, M.E. (2017). A new efficient training strategy for deep neural networks by hybridization of artificial bee colony and limited-memory BFGS optimization algorithms. *Neurocomputing*, 266, 506-526.

[20] Badem, H., Basturk, A., Caliskan, A., Yuksel, M.E. (2018). A new hybrid optimization method combining artificial bee colony and limited-memory BFGS algorithms for efficient numerical optimization. *Applied Soft Computing*, 266, 506-526 .

[21] Akay, B., Karaboga, D. (2017). Artificial bee colony algorithm variants on constrained optimization. *An Internation Journal of Optimization and Control: Theories & Applications*, 7, 98-111.

[22] Ozturk, C., Aslan, S. (2016). A new artificial bee colony algorithm to solve the multiple sequence alignment problem. *Internation Journal of Data Mining and Bioinformatics*, 14, 332-352.

[23] Karaboga, D., Aslan, S. (2016). A discrete artificial bee colony algorithm for detecting transcription factor binding sites in DNA sequences. *Genetics and Molecular Research*, 15, 1-11.

[24] Narasimhan, H. (2009). Parallel artificial bee colony algorithm. *Nature & Biologically Inspired Computing*, 306-311.

[25] Banharnsakun, A., Tiranee, A., Booncharoen, S. (2010). Artificial bee colony algorithm on distributed environment. *Nature & Biologically Inspired Computing*, 13-18 .

[26] Karaboga, D., Aslan, S. (2016). A new emigrant creation strategy based on local best sources for parallel artificial bee colony algorithm. In 24th Signal Processing and Communication Application Conference, 901-904.

Selcuk Aslan *received M.Sc. and Ph.D. degrees from the Department of Computer Engineering, Erciyes University, Kayseri, Turkey in 2013 and 2016, respectively. He is currently working as an assistant professor at Department of Computer Engineering, Ondokuz Mayıs University, Samsun, Turkey. His research interests include numeric and combinatorial optimization, parallel and distributed computations and their usage in engineering applications of intelligent methods.*

An International Journal of Optimization and Control: Theories & Applications (http://ijocta.balikesir.edu.tr)



This work is licensed under a Creative Commons Attribution 4.0 International License. The authors retain ownership of the copyright for their article, but they allow anyone to download, reuse, reprint, modify, distribute, and/or copy articles in IJOCTA, so long as the original authors and source are credited. To see the complete license contents, please visit http://creativecommons.org/licenses/by/4.0/.