



A Pheromonal Artificial Bee Colony (pABC) Algorithm for Discrete Optimization Problems

Dursun Ekmekci

To cite this article: Dursun Ekmekci (2019) A Pheromonal Artificial Bee Colony (pABC) Algorithm for Discrete Optimization Problems, Applied Artificial Intelligence, 33:11, 935-950, DOI: [10.1080/08839514.2019.1661120](https://doi.org/10.1080/08839514.2019.1661120)

To link to this article: <https://doi.org/10.1080/08839514.2019.1661120>



Published online: 06 Sep 2019.



[Submit your article to this journal](#)



Article views: 465



[View related articles](#)




[View Crossmark data](#)



Citing articles: 3 [View citing articles](#)



A Pheromonal Artificial Bee Colony (pABC) Algorithm for Discrete Optimization Problems

Dursun Ekmekci 

Engineering Faculty, Computer Engineering Department, Karabuk University, Karabuk, Turkey

ABSTRACT

The Artificial Bee Colony (ABC) algorithm, which simulates the intelligent foraging behavior of the honeybee colony, is one of the most preferred swarm intelligence-based metaheuristic methods for combinatorial optimization problems. In this study, the local search ability of the ABC algorithm, which can be spread to different regions of the solution space, is developed with the pheromone approach of ant colony optimization (ACO). The effects of the method, named pheromonal ABC (pABC), to the standard ABC and its competitiveness with other metaheuristic methods was presented with testing with popular benchmark problems in the NP-hard problem class. For 40 different benchmark problems, while 15 results with ABC have reached the most successful results were obtained in the literature, 25 results obtained with pABC have reached to literature. While ABC best results were behind literature with a percentage of up to 1.12%, pABC best results were behind the percentage of up to 0.63%

Introduction

Optimization in terms of artificial intelligence can be interpreted, the process of finding the optimum solution, that meets the desired constraints, of a problem. Optimization problems are generally complex problems that require long computation time. Metaheuristic algorithms have been developed to obtain valid solutions for such problems in the NP-Hard class in a reasonable time. These methods which are frequently preferred for combinatorial optimization problems, because they can make evolutionary calculations, often imitate the behavior of living beings in natural life (Karaboga and Gorkemli 2014). Ant colony optimization (ACO) (Dorigo, Maniezzo, and Colorni 1991) that imitates the foraging behavior of ant colonies, firefly algorithm (FA) (Yang 2013) that developed by addressing brightness sensitive social behavior of fireflies, particle swarm optimization (PSO) (Kennedy and Eberhart 1995) that imitates the social behavior of birds swarm and fish swarm, cuckoo search algorithm (CSA) (Yang and Deb 2014) based on the nature of cuckoo parasitism of cuckoo, and artificial bee colony (ABC) algorithm developed by Karaboga, (Karaboga 2005) that imitates

the foraging behavior of honey bees, are some of them. Many hybrid methods have been developed by using a combination of emphasizing features of these optimization algorithms (Kiran et al. 2012) (Tuba and Bacanin 2014) (Rizk-Allah, Zaki, and El-Sawy 2013).

The ABC algorithm is a swarm-intelligence-based algorithm that can generate acceptable solutions for many different types of optimization problems (Karaboga et al. 2014), and compete with other population-based metaheuristic methods in numerical comparison with few algorithm parameters (Karaboga and Basturk 2007) (Karaboga and Akay 2009). The researchers have successfully implemented the ABC algorithm in numerous applications, numerical function optimization, scheduling problems, clustering, program generation (Li and Yang 2016). The algorithm used for modeling and optimization of process parameters of wire electrical discharge machining (Rao and Pawar 2009). Researchers also applied ABC algorithm to neural network training (Karaboga and Akay Bilgisayar 2007) and ANFIS training (Karaboga and Kaya 2016).

Studies to improve the performance of the ABC algorithm are still in today. Especially, the studies focus on improving control parameters (Karaboga and Gorkemli 2012) and creating new hybrid metaheuristic methods. Rosenbrock ABC was developed by combining ABC with Rosenbrock's rotational direction method (Kang, Li, and Ma 2011). In another study, ABC was improved with the Gaussian distribution (Dos Santos Coelho and Alotto 2011). In some of the studies, to improve the ability of ABC, the algorithm was hybridized with particle swarm optimization (Qiu et al. 2013), with differential evolution (Rubio-Largo et al. 2012), with simulated annealing (Chen, Sarosh, and Dong 2012), with firefly algorithm (Tuba and Bacanin 2014), with monarch butterfly optimization (Ghanem and Jantan 2018), and local search algorithms (Fister et al. 2012).

In this study, a hybrid solution model developed for optimization problems, by combining the ABC algorithm with ACO, is introduced, and its usage for discrete problems is presented. The ability of the ABC algorithm, which can be spread to different regions of the solution space, in local search is strengthened by the pheromone approach, which analyzes the correlation between the solution elements. The method, called pheromonal ABC (shortly pABC), differs from standard ABC, in terms of the concept of 'food' and foraging behavior of onlookers. Food consists of pollen collected from different types of flowers. Employed bees diffuse pheromone between flowers while collecting pollen. Onlooker bees consider the amount of pheromone while they choose the flowers to go toward. Proposed method was tested on the capacitated vehicle routing problem one of the most popular combinatorial problems in the discrete structure NP-hard class. The results of ABC and pABC were compared with each other in detail and their best results compared with the best results of the literature obtained to date.

Standard ABC Algorithm

In the standard ABC, the bee colony consists of three groups of honeybees: scout bees, employed bees, and onlooker bees. In the concept of the algorithm, half of the colony consists of employed bees and the other half consists of onlooker bees. For each food source, only one employed bee is assigned. (Karaboga and Akay Bilgisayar 0000). Therefore, the number of food sources is equal to the number of employed bees and the number of onlooker bees. In other words, the colony size is twice the number of sources. The basic steps of the algorithm can be summarized as follows:

Initialization phase

Repeat

 Employed bees phase

 Onlooker bees phase

 Scout bees phase

 Memorized the best solution achieved so far

Until (requirements are met)

In the first step of the algorithm, the initial food sources around the hive are defined. Then in each cycle; employed bees and onlooker bees try to find better quality food sources about initial food sources. When the food source whose nectar was exhausted, has been abandoned, scout bees seek new sources. In the context of the algorithm, food sources correspond to the possible solutions of the problem whose optimal solution is investigated. The success of the respective solution is represented by the quality of the food source. Scout bees select the food source to be directed, according to the roulette wheel (Razali and Geraghty 2011). The scout bees seek a random food source, independent of the employed and onlooker bees. Within the scope of ABC, if any solution cannot be developed after “limit” number of attempts, a random solution is derived instead.

Initialization Phase

In the algorithm that starts with scout bees’ random foraging in nature, the food sources are randomly placed, as shown in (1).

$$s_{m,i} = l_i + rand(0, 1) * (l_i - u_i) \quad (1)$$

In Equation (1), S is the set of solutions and $s_{m,i}$ is the value of dimension i of solution m . m represents the lower bound and u_i represents the upper bound of the solution.

After the initial solutions are created, the $f(s)$ value of each solution is calculated according to the objective function of the problem.

Employed Bees Phase

Employed bees try to find the best quality food source in the area by examining other neighbor sources of the food source they were directed to. The new food source found is expressed by Equation (2).

$$t_{m,i} = s_{m,i} + \phi_{m,i}(s_{m,i} - s_{r,i}) \quad (2)$$

In Equation (2), s_r is a randomly selected solution, i is a randomly selected dimension and $\phi_{m,i}$ is a randomly selected coefficient in the range $[-1, 1]$.

The fitness value of the new solution is calculated and compared with the fitness value of the old solution; the more successful solution is preferred. $fit(s_m)$ (fitness value of solution s_m) is determined by the Equation (3) using $f(s_m)$ value calculated according to the objective function of the problem.

$$fit(s_m) = \begin{cases} 1/(1+f(s_m)) & \text{if } (f(s_m)) \geq 0 \\ 1/(1+abs(s_m)) & \text{if } (f(s_m)) < 0 \end{cases} \quad (3)$$

Onlooker Bees Phase

After each employed bee returns to the hive, she describes the position of the food source she has found, to the onlooker bees in the hive. Each onlooker bee following all employed bees prefers the food source she will head toward according to their nectar quality.

In the ABC algorithm, the selecting probability of each solution is calculated based on the fitness values of the solutions. Accordingly, r_m the probability of selecting the solution s_m , is calculated by Equation (4).

$$r_m = \frac{fit(s_m)}{\sum_{m=1}^{SN} fit(s_m)} \quad (4)$$

Once onlooker bees have selected the food source they will head toward, just like the employed bees, they will control the new food sources about the source they go to and they will head toward higher quality one. In the onlooker bees phase, deriving a new solution using the existing solution is expressed by Equation (2). The fitness value of the new solution is calculated by Equation (3) and a more successful solution is preferred by comparing it to the existing solution.

Scout Bees Phase

The nutrients in each food source will decrease over time and will become insufficient after a while. In such a process, the employed and the onlooker bee visiting the food source and the neighborhood of it turns into a scout bee when she cannot find the nutrients she needs and turns to the search for different food sources in nature.

In the ABC algorithm, in each cycle, existing solutions that fail to produce better solutions are considered unsuccessful. The solutions whose failure counter reaches the limit are abandoned and instead of them, new solutions are randomly generated with equation (1). Owing to this step, the algorithm can overcome the local optimal solution.

Proposed Method: pABC Algorithm

The pABC algorithm is a hybrid metaheuristic method developed by combining the ABC algorithm with the pheromone approach of ACO to strengthen ABC's success in local search. In the context of pABC, honeybees try to collect the best quality food. "Food" consists of pollen collected from different flowers. In other words, each part of the solution is defined as "flower". Employed bees obtain food by collecting pollen from the flowers and diffuse pheromones between the flowers in this process. Onlooker bees choose the flowers to head toward, considering the amount of pheromone. The general structure of pABC is designed as follows:

Initialization phase

Repeat

 Employed bees phase

Updating pheromone values

 Onlooker bees phase

 Scout bees phase

 Memorized the best solution achieved so far

Until (requirements are met)

In the initialization phase, first, the amount of pheromone between flowers is set. The initial pheromone level between any two flowers is set to be a small positive constant (Dorigo and Di Caro 1999). Other operations in the initialization phase, employed bees phase, and scout bees phase are similar to standard ABC. The flowchart of the pABC algorithm is shown in [Figure 1](#).

Updating Pheromone Values

After each employed bee returns to the hive, the pheromone trail between all flowers is updated. Pheromone values are updated in two ways. These are local pheromone update and global pheromone update. Equation (5) is used for local pheromone update.

$$\tau(i, j) = (1 - \rho)\tau(i, j) + \sum_{k=1}^S \Delta\tau(i, j) \quad (5)$$

In local pheromone update, first, the pheromones are evaporated in the specified proportions. In equation (5), $\rho(0 \leq \rho \leq 1)$ determined as the evaporation coefficient is used for reducing the amount of pheromone. Then, the pheromone trails diffused by employed bees between the flowers during pollen collection are

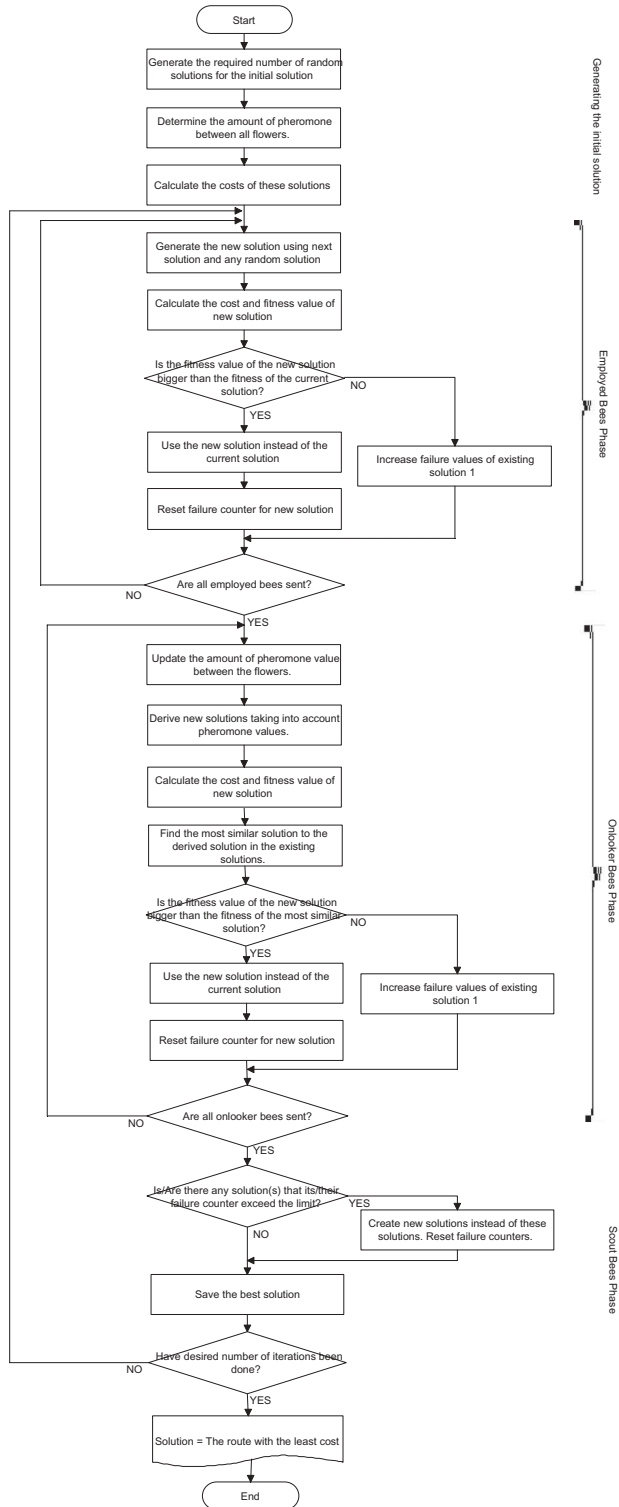


Figure 1. Flowchart of pABC algorithm.

increased. In equation (5), k represents the employed bee and S represents the number of solutions, and the increase of pheromone value between flowers is related to the quality of the food source found by employed bee k . The increase of pheromone value between flowers is shown in (6).

$$\Delta \tau(i, j) = \begin{cases} \frac{1}{f(s_k)} & \text{if } (i, j) \in \text{tour done by employed bee } (k) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

In equation (6), $f(s_k)$ is the solution cost of k .

In the global pheromone update, the pheromone value among the flowers visited by the employed bee, who finds the best quality food in the current step, is increased. In the global pheromone update, the operations in Equations (5) and (6) are repeated only for the employed bee who finds the most successful solution.

Onlooker Bees Phase in pABC

For an onlooker bee visited i number flowers, there are two alternative situations when choosing the next flower (j) to go to. In the first alternative, the flower with the maximum pheromone value and has not visited yet is selected. The likelihood of choosing this way is represented by $q0$ ($0 \leq q0 \leq 1$) in (7).

$$j = \max_{u \in V} \left\{ [\tau(i, u)]^\alpha \cdot [\eta(i, u)]^\beta \right\} \quad \text{if } q \leq q0 \quad (7)$$

In Equation (7): q is a randomly selected value between $[0,1]$. u represents the alternative flower that can be visited and $\delta(i, u)$ represents the distance (cost) between flower i and flower u . In this context, the ‘‘closeness value’’ is calculated with $\eta(i, u) = 1/\delta(i, u)$. α and β are heuristic parameters that determine the importance of the existing pheromone and the distance.

In the second alternative ($q > q0$) for the next flower selection, the selection is made based on the probability distribution calculated considering pheromone values. Selection is made randomly depending on the amount of pheromone. The probability of selecting the flower to be determined is calculated by Equation (8).

$$p_{i,j} = \begin{cases} \frac{[\tau(i,u)]^\alpha \cdot [\eta(i,u)]^\beta}{\sum_{u \in V} [\tau(i,u)]^\alpha \cdot [\eta(i,u)]^\beta} & \text{if } (j \in F) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

When each onlooker bee returns to the hive, the food she brings to the hive is controlled with other existing food and ‘‘the food most similar to the food brought’’ is determined. Comparing the quality of both foods, the higher quality is preferred. In other words, each derived new solution is controlled to existing solutions to determine ‘‘the current solution that most similar to the derived solution’’. Costs of the two solutions are compared with each other and the one has a lower cost is preferred. This process is repeated for all onlooker bees.

New solutions obtained in the initialize solution, employed bees, onlooker bees, and scout bees phases, are controlled by the validity check function


```

bool function Feasibility(Route[])
    feasible=true;
    startNode=0;
    coast=0;
    carryingRate[];
    for i=1 to Length(Route[]) do
        U+=di;
        if (Node==0) then
            if (U>Q) then feasible=false;
            else
                for j=startNode to i do
                    U = U - dj + pj;
                    if (U>Q) then feasible=false;
                    else
                        carryingRate[j]=U/Q;
                        coast += Cj-1,j;
                    end if
                end for
                U=0;
                startNode=i;
            end if
            if (feasible==false) then exit for
        end if
    end for
    return feasible;
end function

```

Figure 2. Algorithm for fitness control and cost calculation of the generated solution.

given in Figure 2. The solution routes are completed or redesigned according to the suitability response from this function. This function can also calculate the status of the carried load according to the vehicle capacity after the load update on each node, and the total cost of the route.

Computational Study

The developed algorithm pABC is coded in the C# programming language at .NET platform. The prepared application ran on a computer having i7-4710MQ 2.50 processor, 8 GB of RAM, and Windows 7 operating system. The application has been tested on 40 different benchmark problems that have complex, wide and fluctuating search space, designed for capacitated vehicle routing problems as discrete optimization problems.

Parameter Settings

When creating the parameter set, ACO parameters were designed as the values which ACO has the most successful results in capacitated vehicle routing problems (Sayyah, Larki, and Yousefikhoshbakht 2016). Accordingly, in all trials; $\alpha = 1$, $\beta = 1$, $\rho = 0.5$ and $q_0 = 0.9$ were used. For the control parameters of ABC, values were set as in standard ABC.

Capacitated Vehicle Routing Problems

The basic approach in capacitated vehicle routing problems is that all materials to be delivered to nodes are distributed from a depot and all materials to be collected from nodes are also collected in the depot (Baker and Ayechev 2003). Vehicle routing problem with simultaneous pickup and delivery (VRPSPD) is one of the most popular examples of capacitated vehicle routing problems, and it is a closed graph model $G = (V, E)$ as a general acceptance. In this graph, $V = \{0, 1, \dots, n\}$ is the set of nodes that depot is represented by node 0 and the customers to be visited are expressed in other numbers. $E = \{(i, j) : i, j \in V\}$ is a set of edges and the cost (distance) of each edge is about c_{ij} . All vehicles in the vehicle fleet have a homogeneous capacity (Q) and can be used in any number of vehicles for transportation. Each node ($i \in V$) can be only visited by one vehicle and only once. In the visited node, the load d_i is delivered, while the demand value p_i is collected from that node. Since the solution space of VRPSPD expands exponentially with the number of nodes, the problem is included in the NP-hard problems class (Zachariadis, Tarantilis, and Kiranoudis 2009).

Benchmarks

Many researchers who developed solutions for discrete optimization problems considered Dethloff benchmarks (Dethloff 2001) as important and tested the methods they developed on these samples (Baker and Ayechev 2003). Algorithms that can produce successful solutions for Dethloff samples have also been successful in larger volume benchmarks (Yousefikhoshbakht, Didehvar, and Rahmati 2014). Dethloff benchmark problems consist of 40 different samples, each with a warehouse center and 50 nodes to visit (Ai and Kachitvichyanukul 2009). The vehicles starting the route from the warehouse center, return to the same center and complete their route. These benchmark problems are divided into four groups as CON3, CON8, SCA3, and SCA8. They are due to the differences in pickup/delivery demands, distances between the nodes, and vehicle capacities, they have solution spaces in different characters. In the SCA samples, all the customer nodes are randomly distributed over an area of [100x100]. In CON samples, half of the customers are distributed randomly in the [100x100] area, while the other

half is randomly distributed in the $[(100/3) \times (200/3)]$ area. While the amount of load to be delivered to each customer is determined randomly between $[0-100]$, the amount of load to be picked up is calculated as $p_i = d_i * (0.5 + r_i)$. In the formula, r_i is a randomly derived value of $[0-1]$. Vehicle capacity in each sample was determined by Equation (9).

$$Q = \sum_{i=1}^M d_i / \mu \quad (9)$$

The value of μ , which is used to determine vehicle capacity, is set to 3 for SCA3 and CON3 samples, and 8 for SCA8 and CON8 samples.

Studies in Literature and Results

In the literature, many methods developed for combinatorial optimization problems have been tested on Dethloff benchmark problems. Some of these methods and the hardware specifications of the machine in which they ran are listed in Table 1. The best results obtained from the algorithms shown in Table 1 taken from (Goksal, Karaoglan, and Altiparmak 2013).

Computational Results

In all 40 benchmark problems, pABC was able to achieve more successful solutions with less iteration than ABC. For example, in the SCA3-1 problem which ABC and pABC reached the most successful result in the literature, ABC reached the result in 6981 iterations but pABC reached this result in 434 iterations. The results of these two algorithms in the first 1000 iterations are shown in Figure 3.

Table 1. Methods tested on Dethloff benchmarks and the machine equipment to which they are applied.

Algorithm	Computer
LNS (Large Neighborhood Search) (Ropke and Pisinger 2006)	Intel Pentium IV 1.5 GHz
ACS (Ant Colony System) (Gajpal and Abad 2009)	Intel Xeon 2.4 GHz
PILS (Parallel Iterative Local Search) (Subramanian et al. 2010)	Intel Pentium IV 2.4 GHz
AMM (Adaptive Memory Methodology) (Goksal, Karaoglan, and Altiparmak 2013)	Intel Xeon 2.66 GHz
h_PSO (hybrid Particle Swarm Optimization) (Goksal, Karaoglan, and Altiparmak 2013)	Intel Xeon 3.16 GHz and 1 GB RAM
MTSEAS (Modified Tabu Search and Elite Ant System) (Yousefikhoshbakht, Didehvar, and Rahmati 2014)	Intel Pentium IV 2.4 GHz and 2 GB RAM
ACSEVNS (Ant Colony System Empowered Variable Neighborhood Search) (Kalayci and Kaya 2016)	Intel Xeon E5-2650 2.0 GHz and 32 GB RAM
ALS (Adaptive Local Search) (Avci and Topaloglu 2015)	Intel Pentium IV 3.00 GHz
MILP (Mixed-Integer Linear Programming) (Rieck and Zimmermann 2013)	Intel 6-core processor, 3.46 GHz, 24 GB RAM
MN_GLS (Multiple Neighborhood Guided Local Search) (Zhu, Feng, and Li 2017)	Intel Core i3-4150 3.5 GHz and 4 GB RAM
EACO (Effective Ant Colony Optimization) (Sayyah, Larki, and Yousefikhoshbakht 2016)	Intel Pentium IV 3500 MHz; Core i3 and 8 GB RAM

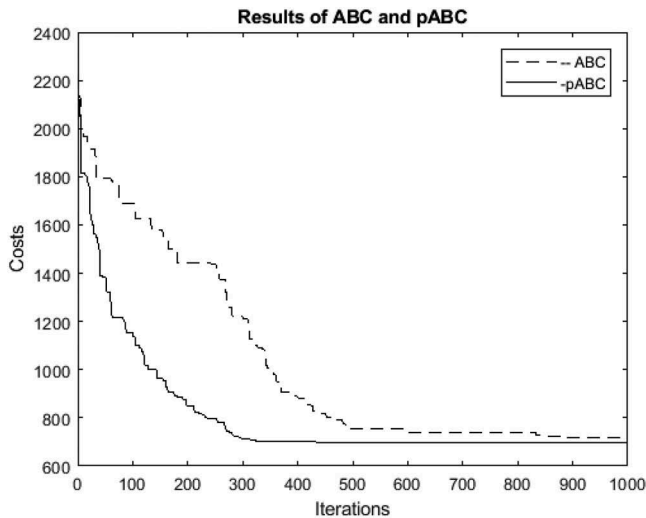


Figure 3. The results of ABC and pABC algorithms in the first 1000 iterations for SCA3-1 solution.

Table 2 presents the average values (AVG) and standard deviation (SD) obtained by ABC and pABC in each problem solution, the CPU time spent for the best result, and the average CPU time spent for each problem solution. When these data are analyzed, it is seen that the pABC resolution time is 8 minutes longer than standard ABC, but it has reached more successful results in all the problems. When the average results ($pABC \approx 766,21$ $ABC \approx 785,19$) and standard deviation values ($pABC \approx 6,99$ $ABC \approx 13,10$) of the solutions obtained by ABC and pABC are examined, it is seen that pABC can produce more successful solutions with a more stable approach. When the most successful results are compared; pABC results are up to 1.09% and 0.32% more successful than ABC results.

In Table 2, the best results of the algorithms developed for combinatorial optimization problems in CON and SCA problems are also compared with the best results obtained with standard ABC and pABC. Referring to Table 2, it can be said that the standard ABC and pABC algorithms are much more successful than Dethloff and can produce solutions close to the results obtained in the literature until recently. For 40 different benchmark problems, while 15 results with ABC have reached the most successful results were obtained in the literature, 25 results obtained with pABC have reached to literature. While ABC best results were behind literature with a percentage of up to 1.12%, pABC best results were behind the percentage of up to 0.63%.

Conclusion

In this study, a novel algorithm (pABC) that was developed by combining ABC with the ACO is introduced. The ability of the ABC algorithm in local search is consolidated by ACO's pheromone approach. The introduced method is tested



Table 2. Comparison of pABC and ABC results with each other and their best results in the literature.

Test Problem	Standard ABC										pABC				
	Literature Best	Best	AVG.	SD	% GAP (ABC best compared to Literature)	CPU Time (for Best)	Average CPU Time	Best	AVG.	SD	% GAP (pABC best compared to Literature)	% GAP (pABC best compared to ABC)	CPU Time (for Best)	Average CPU Time	
SCA3-0	689,00	640,55	664,05	17,88	-0,78	07:47,6	09:34,0	636,12	642,82	11,35	-0,08	0,69	12:45,2	13:11,6	
SCA3-1	765,60	697,84	715,60	12,16	0,00	08:16,6	07:45,8	697,84	712,86	8,75	0,00	0,00	12:12,7	11:45,8	
SCA3-2	742,80	659,30	685,43	18,88	0,00	08:45,9	10:41,1	659,34	680,73	11,62	-0,01	-0,01	15:50,2	13:26,1	
SCA3-3	737,20	680,04	683,11	17,40	-0,45	07:34,5	09:45,9	680,60	688,60	10,11	-0,08	0,37	11:40,2	11:45,7	
SCA3-4	747,10	690,50	692,57	13,09	-0,30	09:44,9	08:32,5	690,50	694,80	6,12	0,00	0,30	16:23,9	15:57,7	
SCA3-5	784,40	659,90	686,73	12,18	0,00	08:36,9	09:57,4	659,90	673,10	11,92	0,00	0,00	16:28,2	17:03,6	
SCA3-6	720,40	651,09	668,07	15,17	0,00	08:44,3	07:37,7	651,09	656,47	5,86	0,00	0,00	15:32,4	15:57,3	
SCA3-7	707,90	659,17	666,54	687,83	15,17	-1,12	03:29,3	03:17,1	659,27	672,41	4,86	-0,02	1,09	11:53,2	11:44,9
SCA3-8	807,20	719,47	723,44	742,13	19,10	-0,55	08:01,6	07:16,5	719,47	722,55	3,46	0,00	0,55	17:30,3	18:41,4
SCA3-9	764,10	681,00	685,16	704,50	11,51	-0,61	09:24,4	12:35,7	681,00	686,08	5,57	0,00	0,61	19:43,9	18:48,6
SCA8-0	1132,90	961,50	995,65	15,19	0,00	08:01,3	07:09,5	961,50	966,16	6,41	0,00	0,00	17:38,6	16:58,2	
SCA8-1	1150,90	1049,65	1060,63	1001,80	18,60	-1,05	07:41,3	08:12,0	1039,64	1045,69	11,58	-0,36	0,68	12:06,7	11:59,5
SCA8-2	1100,80	1039,64	1045,12	1092,20	16,37	-0,53	05:15,4	09:03,1	983,34	987,11	9,52	0,00	10:01,3	12:13,8	
SCA8-3	1115,60	983,34	1011,93	8,47	0,00	03:38,8	03:43,8	983,34	1067,21	1073,48	5,11	0,48	15:29,3	14:39,3	
SCA8-4	1235,40	1065,49	1072,39	1121,13	13,73	-0,65	08:35,3	08:03,5	1027,08	1030,93	4,31	0,00	11:23,8	10:59,2	
SCA8-5	1231,60	1027,08	1059,73	15,07	0,00	08:08,2	08:21,1	1027,08	980,37	7,23	-0,20	0,70	17:23,4	18:41,7	
SCA8-6	1062,50	971,82	980,71	1039,40	15,43	-0,91	08:04,7	10:31,6	973,81	1053,89	3,26	0,00	27:31,8	24:53,1	
SCA8-7	1217,40	1051,28	1059,28	1096,24	14,11	-0,76	19:13,5	19:41,8	1051,28	1075,98	5,18	0,82	21:10,4	22:21,6	
SCA8-8	1231,60	1071,18	1080,02	1131,53	12,53	-0,83	14:37,4	13:11,7	1071,18	1067,90	11,44	0,00	19:19,5	19:21,5	
SCA8-9	1185,60	1060,50	1060,50	1112,97	19,59	0,00	15:11,9	14:01,5	1060,50	1067,90	11,44	0,00	16:45,8	19:04,2	
CON3-0	672,40	616,50	637,93	19,19	0,00	08:52,9	09:17,2	616,50	621,07	6,71	0,00	0,00	12:32,4	11:58,0	
CON3-1	570,60	554,47	572,47	8,26	0,00	03:34,4	03:21,6	554,47	558,11	3,49	0,00	0,00	19:52,3	19:11,7	
CON3-2	534,80	518,00	523,47	538,97	7,35	-1,06	08:10,2	08:58,1	519,18	528,79	6,23	0,82	21:02,7	22:07,3	
CON3-3	656,90	591,19	595,46	611,33	14,75	-0,72	10:04,0	10:20,2	591,19	599,90	12,39	0,00	0,72	16:32,4	17:09,9
CON3-4	640,20	588,79	591,37	610,73	15,75	-0,44	09:52,4	10:12,5	588,79	603,71	7,21	0,44	19:40,7	10:35,4	
CON3-5	604,70	563,70	598,77	16,42	0,00	02:14,4	04:33,8	563,76	578,70	8,58	-0,01	-0,01	21:48,9	22:23,6	
CON3-6	521,30	499,05	502,63	517,43	14,34	-0,72	08:41,0	09:39,6	500,37	510,43	7,56	0,45	29:27,7	26:41,5	
CON3-7	602,80	576,48	580,87	611,87	16,70	-0,76	19:34,8	19:14,6	578,43	589,81	9,26	-0,34	29:27,7	26:41,5	

(Continued)

Table 2. (Continued).

Test Problem	Dethloff	Literature Best	Standard ABC						pABC					
			% GAP (ABC best compared to Literature)			CPU Time (for Best)			% GAP (pABC best compared to Literature)			CPU Time (for Best)		
			Best	AVG.	SD	Best	AVG.	SD	Best	AVG.	SD	Best	AVG.	SD
CON3-8	556,20	523,05	523,94	540,03	17,76	-0,17	17:35,5	525,38	2,83	0,00	0,17	22:30,2	24:11,9	
CON3-9	612,80	578,25	578,25	603,90	11,94	0,00	19:32,5	595,67	6,28	0,00	0,00	30:52,6	28:59,3	
CON8-0	967,30	857,17	864,52	879,77	10,12	-0,86	10:41,5	863,18	7,71	-0,30	0,55	18:04,2	18:41,1	
CON8-1	828,70	740,85	745,91	757,03	6,76	-0,68	07:40,2	743,03	4,47	0,00	0,68	12:11,7	13:04,7	
CON8-2	770,20	712,89	712,89	739,77	8,89	0,00	04:11,9	715,89	4,09	0,00	0,00	15:29,0	16:19,3	
CON8-3	906,70	811,07	816,38	840,58	16,29	-0,65	18:29,6	815,91	5,55	-0,35	0,31	24:01,8	24:11,8	
CON8-4	876,80	772,25	774,90	791,80	7,66	-0,34	05:45,8	777,41	6,16	0,00	0,34	13:52,1	13:16,3	
CON8-5	866,90	754,88	758,33	769,83	5,68	-0,46	06:05,4	758,88	5,38	0,00	0,45	17:06,2	19:41,4	
CON8-6	749,10	678,92	683,21	707,87	5,41	-0,63	02:52,9	686,39	4,82	-0,63	0,00	16:30,1	14:55,8	
CON8-7	929,80	811,96	811,96	828,77	8,78	0,00	02:48,6	815,03	4,19	0,00	0,00	09:17,8	09:09,6	
CON8-8	833,10	767,53	771,19	786,90	5,26	-0,48	04:55,0	771,19	4,99	-0,48	0,00	16:48,3	15:45,6	
CON8-9	877,30	809,00	809,00	824,70	4,88	0,00	08:20,6	810,72	2,71	0,00	0,00	19:01,3	18:38,7	

on discrete optimization problems and the results are compared with the best results obtained in the literature. When the results are examined, it can be said that the standard ABC and pABC can produce valid solutions for combinatorial optimization problems. Besides, it has been observed that pABC can be spread to solution space as much as ABC and with its ability in local search, it can produce more successful and more stable solutions.

In the next study, pABC will be designed for continuous optimization problems and its performance will examine in detail by testing on benchmark functions prepared for numerical optimization problems.

ORCID

Dursun Ekmekci  <http://orcid.org/0000-0002-9830-7793>

References

- Ai, T. J., and V. Kachitvichyanukul. 2009. A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research* 36 (5):1693–702. doi:10.1016/j.cor.2008.04.003.
- Avci, M., and S. Topaloglu. 2015. An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries. *Computers & Industrial Engineering* 83:15–29. doi:10.1016/j.cie.2015.02.002.
- Baker, B. M., and M. A. Ayechev. 2003. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research* 30 (5):787–800. doi:10.1016/S0305-0548(02)00051-5.
- Chen, S. M., A. Sarosh, and Y. F. Dong. 2012. Simulated annealing based artificial bee colony algorithm for global numerical optimization. *Applied Mathematics and Computation* 219 (8):3575–89.
- Dethloff, J. 2001. Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum* 23 (1):79–96. doi:10.1007/PL00013346.
- Dorigo, M., and G. Di Caro. 1999. Ant colony optimization: a new meta-heuristic. *Proceedings of the IEEE 1999 Congress on Evolutionary Computation (Cat. No. 99TH8406)* 2:1470–77.
- Dorigo, M., V. Maniezzo, and A. Colorni. 1991. Positive feedback as a search strategy.
- Dos Santos Coelho, L., and P. Alotto. 2011. Gaussian artificial bee colony algorithm approach applied to Loney's solenoid benchmark problem. *IEEE Transactions on Magnetics* 47 (5):1326–29. doi:10.1109/TMAG.20.
- Fister, I., I. Fister, J. Brest, and V. Žumer. 2012. Memetic artificial bee colony algorithm for large-scale global optimization. *IEEE Congress on Evolutionary Computation, CEC 2012* 10–15. doi:10.1109/CEC.2012.6252938.
- Gajpal, Y., and P. Abad. 2009. An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research* 36 (12):3215–23. doi:10.1016/j.cor.2009.02.017.
- Ghanem, W. A. H. M., and A. Jantan. 2018. Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems. *Neural Computing and Applications* 30 (1):163–81. doi:10.1007/s00521-016-2665-1.
- Goksal, F. P., I. Karaoglan, and F. Altiparmak. 2013. A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering* 65 (1):39–53. doi:10.1016/j.cie.2012.01.005.

- Kalayci, C. B., and C. Kaya. 2016. An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications* 66:163–75. doi:10.1016/j.eswa.2016.09.017.
- Kang, F., J. Li, and Z. Ma. 2011. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Information Sciences* 181 (16):3508–31. doi:10.1016/j.ins.2011.04.024.
- Karaboga, D. 2005. An idea based on honey bee swarm for numerical optimization.
- Karaboga, D., and B. Akay. 2009. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation* 214 (1):108–32.
- Karaboga, D., and B. Akay Bilgisayar; Mühendisliği Bölümü Erciyes Üniversitesi. 2007. Artificial bee colony (ABC) algorithm on training artificial neural networks.
- Karaboga, D., and B. Basturk. 2007. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization* 39 (3):459–71. doi:10.1007/s10898-007-9149-x.
- Karaboga, D., and B. Gorkemli. 2012. A quick artificial bee colony -qABC- algorithm for optimization problems. *INISTA 2012 : International Symposium on Innovations in Intelligent Systems and Applications* 1–5. doi:10.1109/INISTA.2012.6247010.
- Karaboga, D., and B. Gorkemli. 2014. A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. *Applied Soft Computing* 23:227–38. doi:10.1016/j.asoc.2014.06.035.
- Karaboga, D., B. Gorkemli, C. Ozturk, and N. Karaboga. 2014. A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review* 42 (1):21–57. doi:10.1007/s10462-012-9328-0.
- Karaboga, D., and E. Kaya. 2016. An adaptive and hybrid artificial bee colony algorithm (aABC) for ANFIS training. *Applied Soft Computing* 49:423–36. doi:10.1016/j.asoc.2016.07.039.
- Kennedy, J., and R. Eberhart. 1995. Particle swarm optimization. *Proceeding IEEE International Conference on Neural Networks* 4:1942–48. vol.4.
- Kiran, M. S., E. Özceylan, M. Gündüz, and T. Paksoy. 2012. A novel hybrid approach based on particle swarm optimization and ant colony algorithm to forecast energy demand of Turkey. *Energy Conversion and Management* 53 (1):75–83. doi:10.1016/j.enconman.2011.08.004.
- Li, X., and G. Yang. 2016. Artificial bee colony algorithm with memory. *Applied Soft Computing* 41:362–72. doi:10.1016/j.asoc.2015.12.046.
- Qiu, J., Y. Shen, J. Xie, and J. Wang. 2013. Pbest-guided artificial bee colony algorithm for global numerical function optimization. *International Journal of Applied Mathematics and Statistics* 43 (13):117–25.
- Rao, R. V., and P. J. Pawar. 2009. Modelling and optimization of process parameters of wire electrical discharge machining. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 223 (11):1431–40. doi:10.1243/09544054JEM1559.
- Razali, N. M., and J. Geraghty. 2011. Genetic algorithm performance with different selection strategies in solving TSP. *Proceedings of the World Congress on Engineering 2011 Vol II WCE 2011, July 6 - 8, 2011, London, UK.*
- Rieck, J., and J. Zimmermann. 2013. Exact solutions to the symmetric and asymmetric vehicle routing problem with simultaneous delivery and pick-up. *Business Research*. 6 (1):77–92. doi:10.1007/BF03342743.
- Rizk-Allah, R. M., E. M. Zaki, and A. A. El-Sawy. 2013. Hybridizing ant colony optimization with firefly algorithm for unconstrained optimization problems. *Applied Mathematics and Computation* 224:473–83.

- Ropke, S., and D. Pisinger. 2006. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research* 171 (3):750–75. doi:10.1016/j.ejor.2004.09.004.
- Rubio-Largo, A., D. L. Gonzalez-Alvarez, M. A. Vega-Rodriguez, J. A. Gomez-Pulido, and J. M. Sanchez-Perez. 2012. MO-ABC/DE-multiobjective artificial bee colony with differential evolution for unconstrained multiobjective optimization. *CINTI 2012-13th IEEE International Symposium on Computational Intelligence and Informatics Proceedings* 157–62. doi:10.1109/CINTI.2012.6496752.
- Sayyah, M., H. Larki, and M. Yousefikhoshbakht. 2016. Solving the vehicle routing problem with simultaneous pickup and delivery by an effective ant colony optimization. *Journal of Industrial Engineering and Management Studies* 3 (1):15–38.
- Subramanian, A., L. M. A. Drummond, C. Bentes, L. S. Ochi, and R. Farias. 2010. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research* 37 (11):1899–911. doi:10.1016/j.cor.2009.10.011.
- Tuba, M., and N. Bacanin. 2014. Artificial bee colony algorithm hybridized with firefly algorithm for cardinality constrained mean-variance portfolio selection problem. *Applied Mathematics & Information Sciences* 8 (6):2831–44. doi:10.12785/amis/080619.
- Yang, X. S. 2013. Multiobjective firefly algorithm for continuous optimization. *Engineering with Computers* 29 (2):175–84. doi:10.1007/s00366-012-0254-1.
- Yang, X. S., and S. Deb. 2014. Cuckoo search: Recent advances and applications. *Neural Computing & Applications* 24 (1):169–74. doi:10.1007/s00521-013-1367-1.
- Yousefikhoshbakht, M., F. Didehvar, and F. Rahmati. 2014. A combination of modified tabu search and elite ant system to solve the vehicle routing problem with simultaneous pickup and delivery. *Journal of Industrial and Production Engineering* 31 (2):65–75.
- Zachariadis, E. E., C. D. Tarantilis, and C. T. Kiranoudis. 2009. A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications* 36 (2 PART 1):1070–81. doi:10.1016/j.eswa.2007.11.005.
- Zhu, H., J. Feng, and H. Li. 2017. MN _ GLS for VRP with Simultaneous delivery and pickup. *Journal of Computers*, 28 (6):1–12. doi:10.3966/199115992017122806001.