SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ DERGİSİ

Sakarya University Journal of Science
SAUJS

e-ISSN 2147-835X | Period Bimonthly | Founded: 1997 | Publisher Sakarya University |
http://www.saujs.sakarya.edu.tr/en/

Title: An Adaptive Ant Colony System Memorizing Better Solutions (aACS-MBS) For Traveling Salesman Problem

Authors: Dursun EKMEKÇİ

Recieved: 2020-11-06 18:32:56

Accepted: 2021-04-07 22:33:18

Article Type: Research Article

Volume: 25
Issue: 3
Month: June
Year: 2021
Pages: 673-689

How to cite
Dursun EKMEKÇİ; (2021), An Adaptive Ant Colony System Memorizing Better Solutions (aACS-MBS) For Traveling Salesman Problem. Sakarya University Journal of Science, 25(3), 673-689, DOI: https://doi.org/10.16984/saufenbilder.822646
Access link
http://www.saujs.sakarya.edu.tr/en/pub/issue/62736/822646

Sakarya University Journal of Science 25(3), 673-689, 2021

An Adaptive Ant Colony System Memorizing Better Solutions (aACS-MBS) For Traveling Salesman Problem

Dursun EKMEKCİ*[1]

Abstract

Choosing the optimal one among the many alternatives that meet the criteria is one of the problems that occupy life. This kind of problem frequently encountered by commercial companies in daily life is one of the issues that operators focus on with care. Many techniques have been developed that can provide acceptable solutions in a reasonable time. However, one of the biggest problems with these techniques is that the appropriate values can be assigned to the algorithm parameters. Because one of the most important issues determining algorithm performance is the values to be assigned to its parameters. The Ant Colony System (ACS) is a metaheuristic method that produces successful solutions, especially in combinatorial optimization problems (COP). The Ant Colony System Memorizing Better Solutions (ACS-MBS) algorithm is an ACS version developed to associate the pheromone value more with the solution success. In this study, an Adaptive ACS-MBS (aACS-MBS) method is presented that updates the *q0* parameter dynamically, which balances the exploitation and exploration activities of the ACS-MBS. The method has been tested on the traveling salesman problem (TSP) of different sizes, and the obtained results are evaluated together with the change in the q0 parameter, and the solution search strategy of the algorithm is analyzed. With the pheromone maps formed as a result of the search, the effect of transfer functions was evaluated. Results obtained with aACS-MBS were compared with different ant colony optimization (ACO) algorithms. The aACS-MBS fell behind the most successful solution found in the literature, by up to 3.83%, in large-scale TSP benchmarks. As a result, it has been seen that the method can be successfully applied to the COP.

Keywords: ant colony optimization, ant colony system, ant colony system memorizing better solutions, adaptive ant colony system memorizing better solutions

*Corresponding author: dekmekci@karabuk.edu.tr
[1] Karabuk University, Faculty of Engineering, Computer Engineering Department, 78050, Karabük.
ORCID: https://orcid.org/0000-0002-9830-7793

1. INTRODUCTION

One of the main factors that determine the performance of swarm intelligence-based optimization algorithms, which are generally designed inspired by nature, is the model that the algorithm imitates. In this context, the most similar simulation of the problem characteristic should be preferred in the selection of the algorithm. While the model in nature is designed as a computer algorithm, the behaviors of the model are determined as procedures, and the components that affect these behaviors are determined as algorithm parameters. If the model whose algorithm is designed is a swarm that living as a colony, in general, the individual activities of the colony members determine the exploration ability of the algorithm, while collective activities affect the exploitation ability [1]. Another factor affecting the success of the algorithm is the appropriateness of the values assigned for the algorithm parameters. Researchers focusing on metaheuristic methods in the field of operational research have recently developed online and offline techniques that determine the appropriate values for the control parameters of these algorithms.

Ant Colony Optimization (ACO) is a swarm intelligence-based metaheuristic method that imitates the adventure of an ant colony in foraging and determining the shortest path between the food and the nest [2]. In real life, when ants, who leave the nest in search of food, come across a food source, they try to find the shortest path to move the food from that source to their nest. Ants emit an evaporating secretion called pheromone in the process of carrying food between the source and the nest, and the other ants following them continue their journey by following the more intense secretion. In this way, the shortest path is determined after a few tours. Even if the path used is no longer the shortest path due to different factors, this approach makes it easy for ants to find the shortest path. The shortest path determination behavior of the colony members participating in the food collection is illustrated in Figure 1.

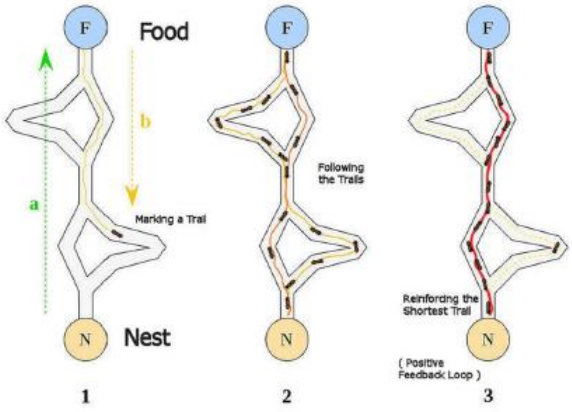

Figure 1 The shortest path finding prototype of artificial ants

The synergy, based on this collective behavior of the ant colony, inspired the development of ACO algorithms with strong exploitation ability, which is especially successful in solving the combinatorial optimization problem (COP) [3]. Although different ACO algorithms, versions, and extensions have been developed concerning this approach, the pheromone chemical is the basic component for all methods [4]. An effective metaheuristic algorithm should provide an appropriate balance between maximum utilization of the search experience so far, and the exploration of relatively unexplored search regions. The variation in ACO algorithms is based on differences in the approach in managing the pheromone trail to reach this targeted balance. The pheromone level induces a probability distribution in the search space and determines which regions of the field are sampled effectively, in other words, in which parts of the field the solutions constructed are concentrated. In this regard, the most successful performing ACO algorithms use strategies where the best solutions found during the search strongly contribute to pheromone updating [3]. In line with this understanding, the Ant Colony Optimization Memorizing Better Solutions (ACO-MBS) which uses "Edge Matrix" to directly correlate the correlation between solution components with solution success, was developed [5]. In the literature, suggestions for pheromone control [6] focus more on determining the evaporation rate [7], assigning pheromone value [8], and limiting the pheromone level [9].

After analyzing the correlations between solution components correctly, it can be said that the second important issue affecting the algorithm performance is the determination of the pheromone trail. Because ants will continue their journey by considering the pheromone density in the environment. Within the scope of the algorithm, turning to the components where the pheromone secretion is most intense increases the exploitation capability of the algorithm while decreasing the exploration capability. In Ant System (AS) [10], which is the first ACO algorithm, the roulette wheel approach based on pheromone concentrations is applied for the node selection, while in Ant Colony System (ACS) [11], the *q0* parameter is added to the algorithm. (These methods are explained in detail in the next section.) Although many metaheuristic methods use different operators for exploitation and exploration conformance, the ACS combines this balance in only one parameter. In the study, the Adaptive Ant Colony System Memorizing Better Solutions (aACS-MBS) method, which dynamically controlled the *q0* parameter of the ACS-MBS, which is improved to associate the pheromone value of the ACS with the solution success more, is proposed. The method has been tested on the traveling salesman problem (TSP), which is one of the popular examples of combinatorial optimization problems (COP), and the results obtained proved that the proposed method increases the exploratory ability of the ACS.

The remainder of the article is designed as follows: Section 2 describes ACO with its general features, the AS and the ACS. Next, the ACS-MBS, an improved version of ACS, is explained. In the section, finally, the proposed method is introduced. In Section 3, TSP, and selected TSP benchmarks are explained. In Section 4, the results obtained with the proposed method and different versions of the ACO are presented. In Section 5, the study has been concluded in all aspects.

2. BACKGROUND OF THE PROPOSED METHOD

In this section, the AS algorithm that applies the ACO approach in its basic form, and the ACS, developed with the updates in the algorithm, are explained in detail. Later, the ACS-MBS, which was developed to reflect the solution success directly to the correlation between the components, and the proposed aACS-MBS method is introduced to update the search orientation of the method in the study process.

2.1. Ant System (AS)

The first ACO algorithm developed inspired by the real ant colony is the AS algorithm. The main steps of the algorithm, the first versions of which are tested on TSP, are shown in Figure 2.

Algorithm 1 General Structure of ACO
Set parameters
Initialize pheromone trails
While termination conditions not met **do**
  for $i = 0$ to m **do**
    Construct an artificial ant solution (Si)
  end for
  Daemon actions (optional)
  Update pheromones
End while
Output : The best so far solution

Figure 2 General steps of ACO

As seen in Figure 2, in the first step, values for the AS parameters, and the initial pheromone levels are determined. For the initial solutions, to equalizing the selection probabilities of all components, initial pheromone values should be kept as low as 0.1 as possible [7].

While solutions are constructing with m determined artificial ants, the first item of the next solution is the node selected randomly. The solution array is then expanded by adding a node from the set of possible neighbors. This process continues until the solution is constructed. The node to be added to the solution array is determined by a probability calculation. In a problem that has V nodes, for the ant k which

coming to node *i*, the selection probability (p_{ij}^k) of node *j* is determined by (1).

$$p_{ij}^k = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{u \in V_i^k}(\tau_{iu})^{\alpha}(\eta_{iu})^{\beta}} \quad (1)$$

In (1), τ_{ij} represents the pheromone level between *i-j*, and V_i^k represents the possible nodes that ant k can visit after node *i*. δ_{ij} is the distance between *i-j* and η_{ij} is the inverse of this distance ($1/\delta_{ij}$). α and β are the heuristic parameters of the algorithm.

After solutions are constructed, some problem-specific calculations may be required before pheromone updates. These operations, called *"daemon actions"* are operations that facilitate local search. In the pheromone updating process, firstly the pheromone levels between the components are evaporated at the set ratio, and then the pheromone is added between the components in proportion to the solution success the ants obtain [12]. For these operations, (2) is used.

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k \quad (2)$$

In (2), ρ is the evaporation rate assigned in the range of (0, 1], and $\Delta\tau_{ij}^k$ is the amount of pheromone to be added between *i-j* in proportion to the solution cost (L_k) of the ant *k*. The amount of this pheromone to be added is calculated by (3).

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{L_k} & if\ ant\ k\ used\ edge\ (i-j) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

2.2. Ant Colony System (ACS)

The ACS includes the updates made for the AS algorithm to be greedier in generating new solutions. In this context, the updates are based on increasing the amount of pheromone between the components that make up the more successful solution and taking more consideration of the pheromone level in constructing solutions.

Two alternative methods are applied in deriving new solutions in ACS. If the value of *q* randomly selected in the range of [0,1] is bigger than *q0*, the node *j* after *i* is determined by (1). Otherwise, the node who has the highest pheromone concentration between node *i* is selected.

Pheromone updating process in ACS is divided into two categories: The operations in (2) and (3) are defined as *"local pheromone update"* and are applied for all ants in each iteration. *"Global pheromone update"* is applied with (4) for the most successful ant route in the iteration.

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \rho.\Delta\tau_{ij}^{best} \quad (4)$$

2.3. Ant Colony System Memorizing Better Solutions (ACS-MBS)

The ACS-MBS is one of the ACO versions that developed to reflect the solution success directly among the components that make up the relevant solution array and thus establish the correlation between the components more accurately. While the algorithm takes advantage of the ACS transition rules, it uses transfer functions in pheromone updating. Compared to other popular algorithms of ACO for TSP, the method produced more successful solutions than many of the algorithms [5].

In the ACS-MBS, apart from the pheromone matrix (PM), an edge matrix (EM) is used. In each cycle, after the solutions are constructed by artificial ants, the values in the cells corresponding to the solution components in the EM are compared with the cost of the new solution. If a lower cost solution (S_m) is constructed by using any edge, the cell ($EM_{S_{mi}S_{mj}}$) corresponding to this edge in EM is updated as the new solution cost. Then the relevant update is reflected in the PM. If the solution obtained is the best one so far *"global update"* is applied, if not *"local update"*. In local updating, firstly, the normalization value in the range (0.1 - 1) of the solution cost is calculated with (5).

$$normal_{S_m} = 1 - \frac{[cost(S_m) - \min(EM)] * 0.9}{[max(EM) - \min(EM)]} \quad (5)$$

The pheromone level of the relevant edge is calculated with the specified transfer function for the normalization value obtained by (5). The

pheromone update process in the ACS-MBS is illustrated in Figure 3.

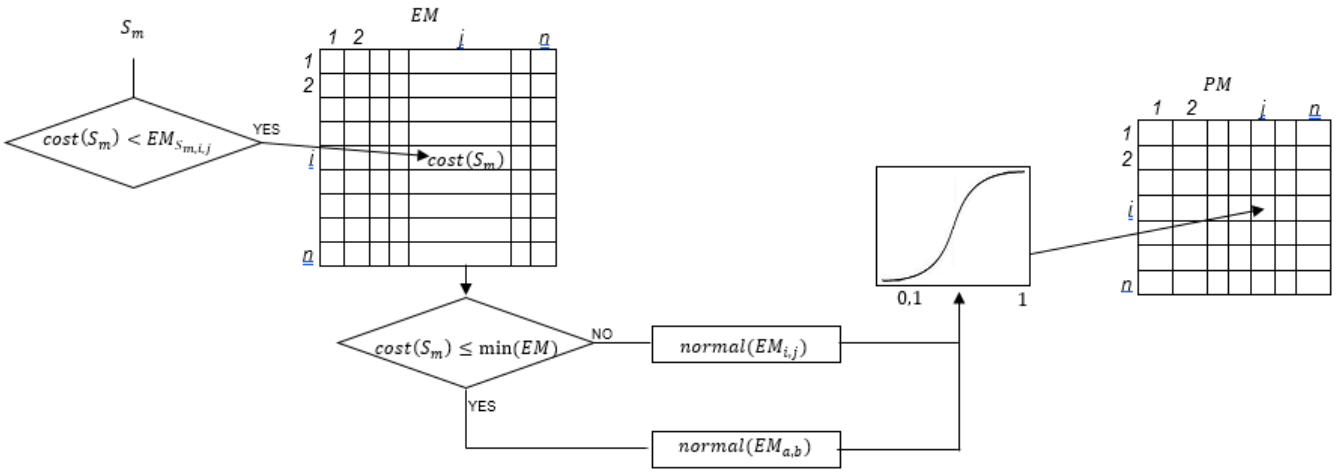

Figure 3 Pheromone updating process in ACS-MBS

Within the scope of the algorithm, different types of transfer functions can be used for the pheromone update, depending on the problem's estimated solution space, problem complexity, and parameter characteristics.

2.4. Proposed Method: Adaptive Ant Colony System Memorizing Better Solutions (aACS-MBS)

In AS and ACS, the amount of pheromone accumulated at the edges can reach values greater than 1, depending on the use of the relevant edge. In the ACS-MBS, the values in PM are limited to 0-1. Therefore, the change in PM in each cycle contains more objective information about the searching of the algorithm. With the proposed method, exploitation/exploration mobility of ACS-MBS is followed over the change in PM, and the value of *q0* is dynamically assigned for the balance between these two.

Firstly, the total amount of *change* in pheromone levels retained in PM in each cycle is calculated with (6).

$$change = \sum_{i=1}^{D}\sum_{j=1}^{D} |old(PM_{i,j}) - new(PM_{i,j})| \quad (6)$$

Then (7) is used to determine the value to be assigned to the *q0* parameter.

$$q0 = 0.8 + change * Dimension \quad (7)$$

Equation (7) is formed by considering the results of the experiments obtained with different parameter values. Experiments have shown that when values less than 0.8 are assigned for the *q0* parameter, the algorithm fails to produce good results, and when bigger values are assigned, the algorithm loses its exploration ability. For the algorithm not to lose its exploitation ability, the minimum value assigns to *q0* is set as 0.8. As seen in (7), the value to be assigned to *q0* is calculated by considering the amount of change in PM. The value calculated by (6) is multiplied by the number of dimensions (nodes) in the problem and added to 0.8.

In this context, the aABC-MBS algorithm is shown in Figure 4 with its main steps.

Algorithm: aACS-MBS

```
Set parameters {α, β, Transfer Function}
// Initialization phase
For i=1 To m
  Construct solution Si randomly
End For
Set EMi,j=max(S)
Set PMi,j=0
// Update EM
For i=1 To m
  For j=1 To n
    If (Si < EMSi,Sj)
      EMSi,Sj=Si
    End If
  End For
End For
// Update PM
For i=1 To n
  normalSn=1-(cost(Sn)-min(EM))*0.9/(max(EM)-min(EM))
  For j=1 To n
    // Calculate f(normalSn) according to the transfer function
    If (f(normalSn) > PMi,j)
      PMi,j=normalSn
    End If
  End For
End For
While termination conditions not met do
  For k=1 To m
    Select node i randomly   i={1, 2, 3, …, n}
    While solution Sk not completed
      Select q randomly
      If (q ≤ q0)
        Next node=max(τi,j)
      Else
        Select next node according to Eq. (1)
      End If
      // Update EM
      For i=1 To m
        For j=1 To n
          If (Si<EMSi,Sj)
            EMSi,Sj=Si
          End If
        End For
      End For
      // Update PM and calculate the change value
      change=0
      For i=1 To n
        normalSn=1-(cost(Sn)-min(EM))*0.9/(max(EM)-min(EM))
        For j=1 To n
          // Calculate f(normalSn) according to the transfer function
          If (f(normalSn)>PMi,j)
            change =abs(PMi,j - normalSn)
            PMi,j=normalSn
          End If
        End For
      End For
      q0=0.8+change * n
    End While
End While
Output: best(S)
```

Figure 4 The aACS-MBS algorithm

3. TSP

As one of the most popular examples of COP, TSP is one of the first benchmarks in which the methods developed by operations researchers were tested. The basic principle in the problem, the first examples of which were put forward in the 1800s, is to try to obtain a closed graph with the shortest length. In a network model with *n* nodes to visit, the salesman starts its travel from any node and returns to the starting node, provided they visit all nodes [13]. Solution cost is directly dependent on route distance. Since the shortest Hamilton tour is searched in *G(V, E)* closed graph, where *V* is the set of nodes, and *E* is the set of edges between these nodes, the problem is in the NP-hard class [14]. The distance matrix (*D*) is associated with *V*, and in many instances of TSP the distances between nodes are symmetrical (d_{ij}=d_{ji}). The Euclidean formula is usually used to calculate distances. In this context, the total route cost is calculated with equation (8).

$$f(x) = \left(\sum_{i=1}^{n-1} d_{ij}\right) + d_{n1} \quad (8)$$

4. EXPERIMENTAL STUDY

The proposed method was run on the .net platform, coded with C # programming language, and on a machine with i7-5600U CPU 2.60 processor, 8 GB RAM, and 64-bit Windows 8 Operating System. The performance of the aACS-MBS algorithm and its searching strategy has been tested on different sizes of TSP benchmarks. In this section, the results obtained with the algorithm are presented, and the analyzes are discussed.

4.1. TSP Benchmarks

The application has been tested on TSP samples taken from the TSPLIB library (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html). Selected benchmarks are eil51, kroA100, kroA150, kroA200, pr299 and pr439. The numbers of nodes in the problems and the best results known to date [15] [16] are presented in Table 1.

Table 1
Selected TSP benchmarks

Name	Number of Nodes	Best Result
eil51	51	417
kroA100	100	21282
kroA150	150	26524
kroA200	200	29368
pr299	299	48191
pr439	439	107217

4.2. Parameter Settings

The AS, ACS, ACS-MBS, and aACS-MBS algorithms have been run independently 30 times with a maximum cycle number (MCN) is 1000 for the selected test problems. For parameter settings, parameter analysis for the ACS-MBS was considered. In this context, the values in [5] have been assigned to the algorithm parameters. The parameter values assigned for the algorithms are shown in Table 2.

Table 2
Algorithm parameters

	AS	ACS	ACS-MBS	aACS-MBS
Colony Size	20	20	20	20
α	1	1	1	1
β	5	5	5	5
ρ	0.1	0.1	-	-
q0	0.95	0.95	0.95	-
Transfer Function	-	-	linear	linear

In order to analyze the searching strategy and convergence performance of the aACS-MBS in detail, the algorithm was also run by using the *"linear"*, *"sigmoid"* (9), and *"v-shaped"* (10) transfer functions. Transfer functions are drawn in Figure 5. Algorithm was run independently 30 times with these transfer functions, and MCN was 1000 in all experiments.

$$sigmoid = f(x) = \frac{1}{1+e^{-10*(normal_{S_i}-0.5)}} \quad (9)$$

$$v-shaped = f(x) = \begin{cases} 1 & if(x \le 0.1) \\ x^3 & otherwise \end{cases} \quad (10)$$

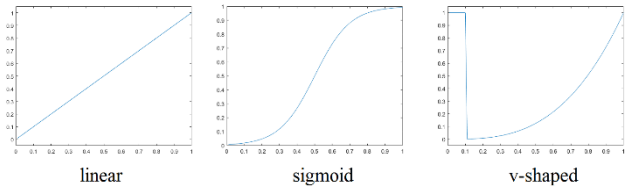

Figure 5 Transfer Functions

4.3. Results

The results obtained by the algorithm are presented in two categories. In the first category, the effect of transfer functions on the algorithm is analyzed. In the second category, aACS-MBS results are compared with solutions obtained with other ACO versions.

4.3.1. Searching Strategy Analysis of aACS-MBS

For the best solutions obtained by aACS-MBS with different transfer functions, the results in each iteration and the current *q0* values are shared in graphics (Figures 6-23). The figures also show the pheromone map that was obtained at the end of the searching process.

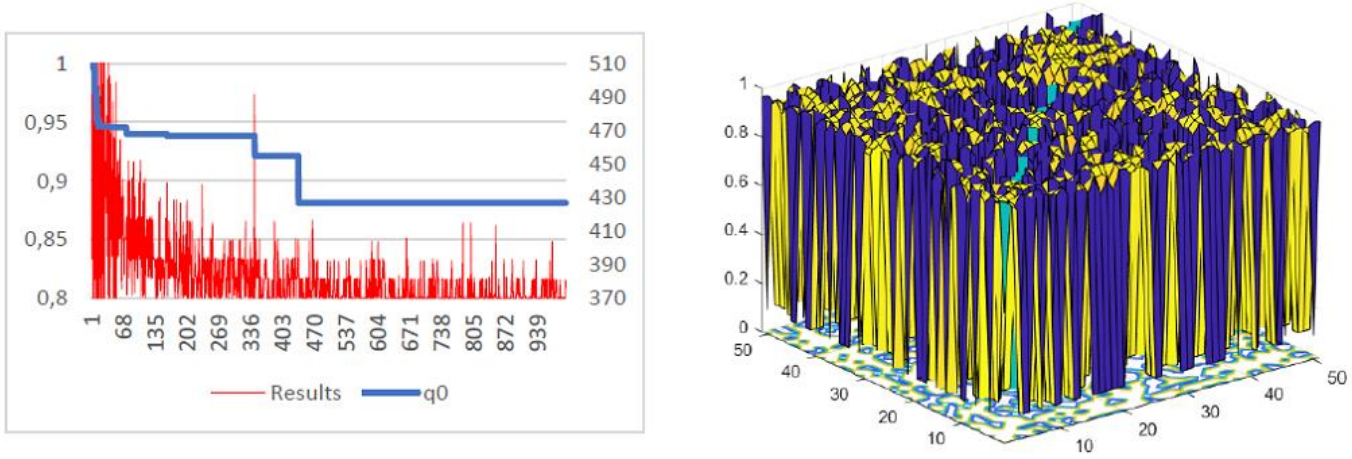

Figure 6 Results, change of q0, and pheromone map obtained in eil51 solution with linear transfer function

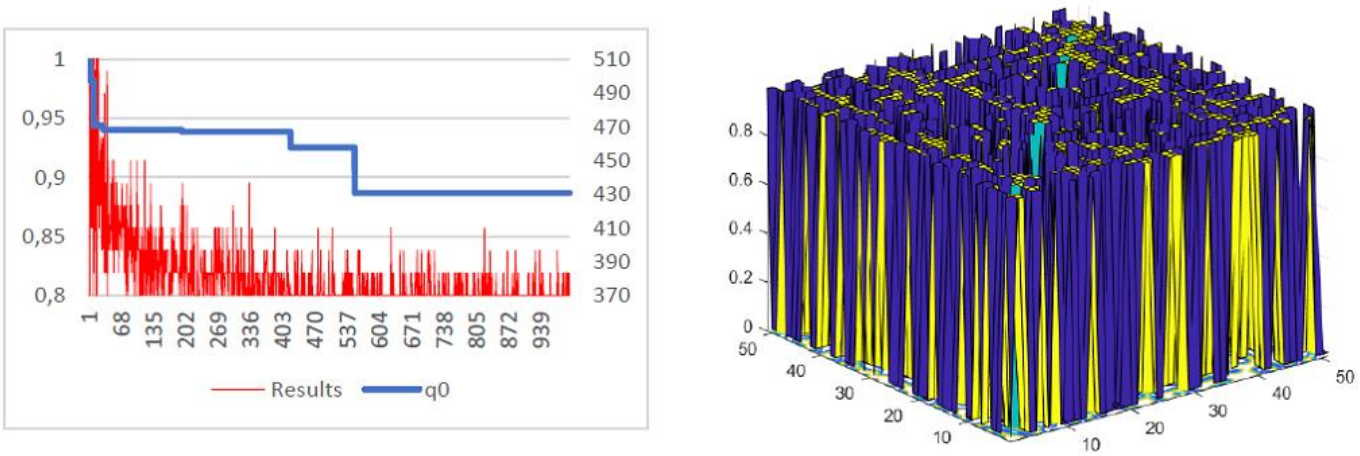

Figure 7 Results, change of q0, and pheromone map obtained in eil51 solution with sigmoid transfer function

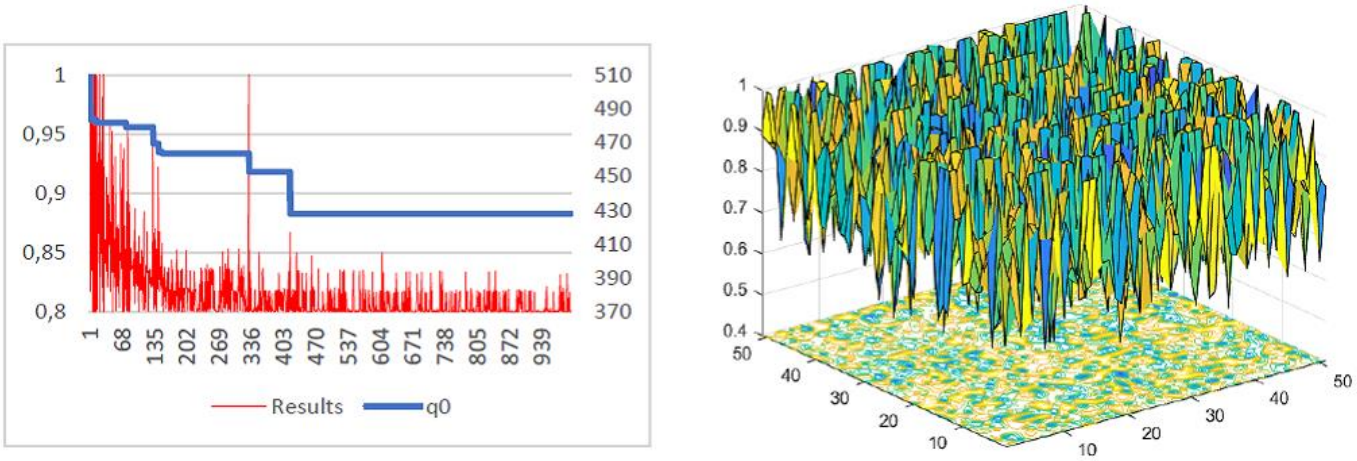

Figure 8 Results, change of q0, and pheromone map obtained in eil51 solution with v-shaped transfer function

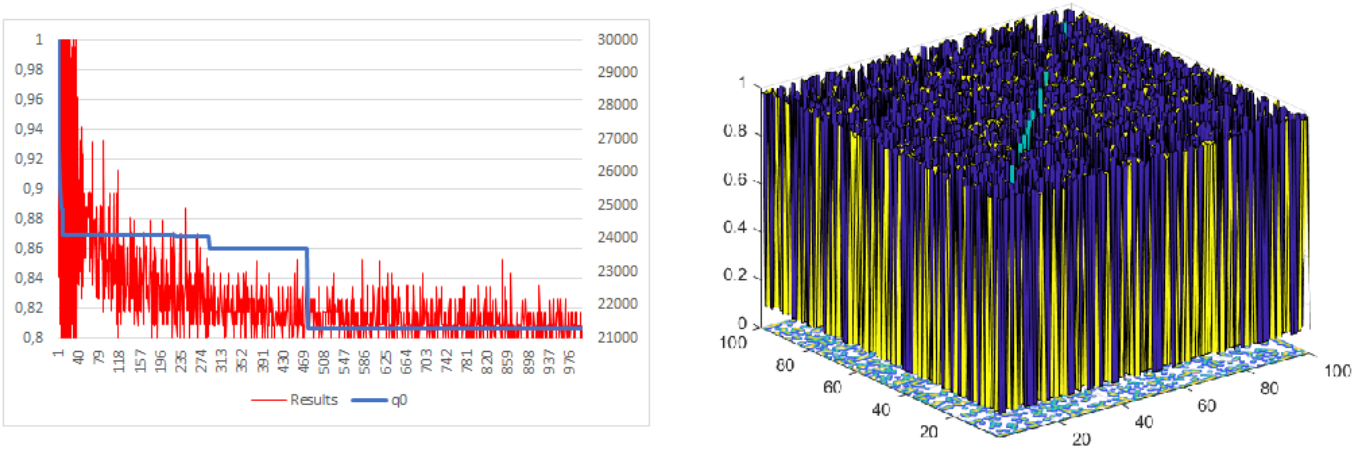

Figure 9 Results, change of q0, and pheromone map obtained in kroA100 solution with linear transfer function

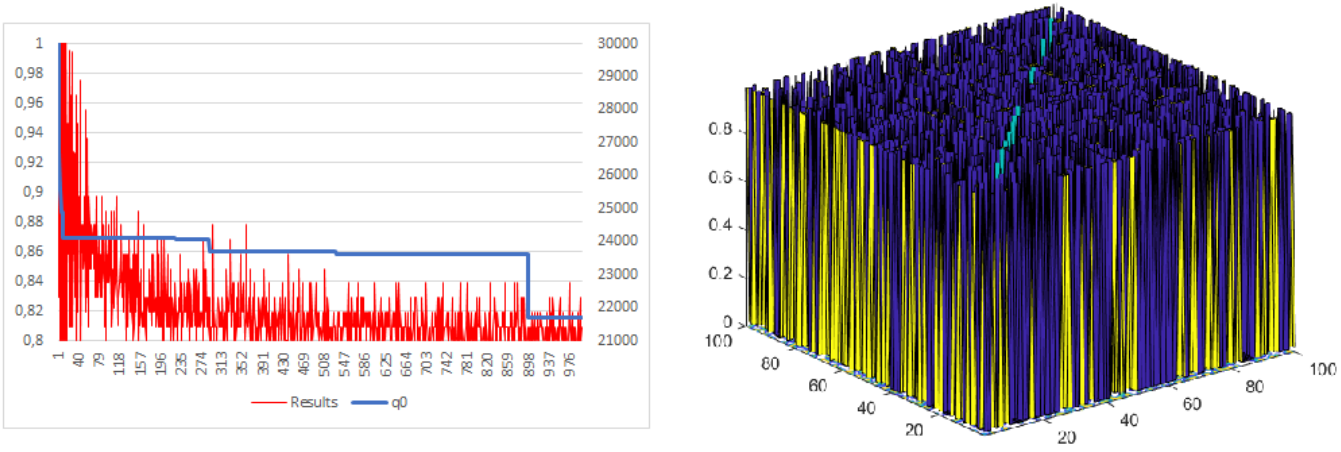

Figure 10 Results, change of q0, and pheromone map obtained in kroA100 solution with sigmoid transfer function

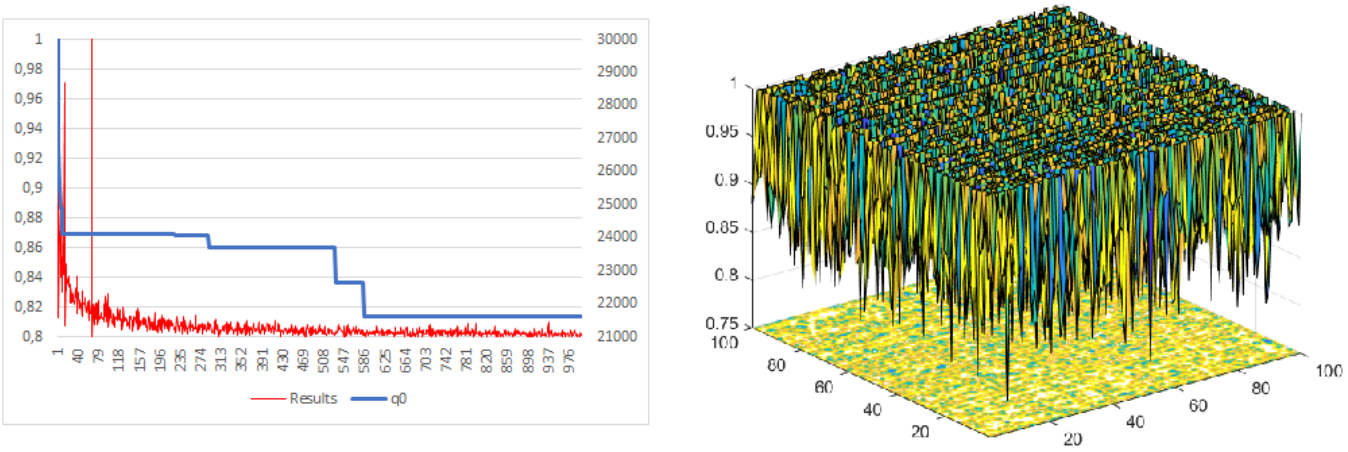

Figure 11 Results, change of q0, and pheromone map obtained in kroA100 solution with v-shaped transfer function

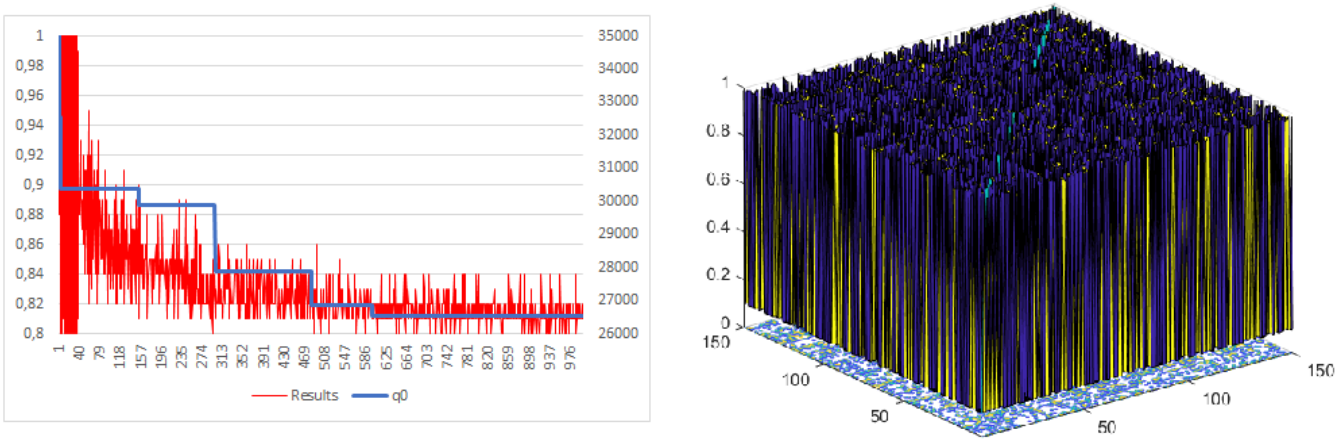

Figure 12 Results, change of q0, and pheromone map obtained in kroA150 solution with linear transfer function

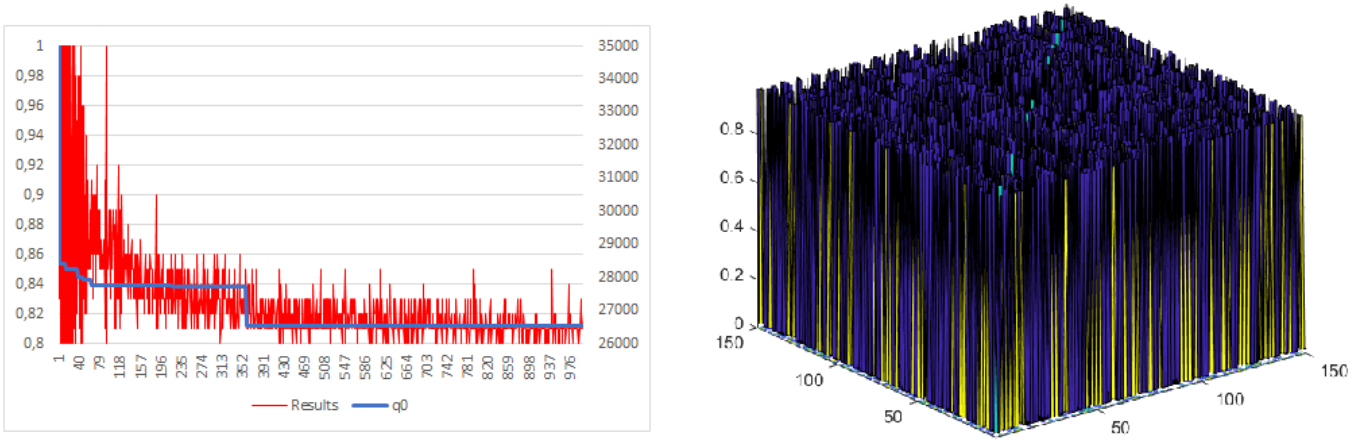

Figure 13 Results, change of q0, and pheromone map obtained in kroA150 solution with sigmoid transfer function

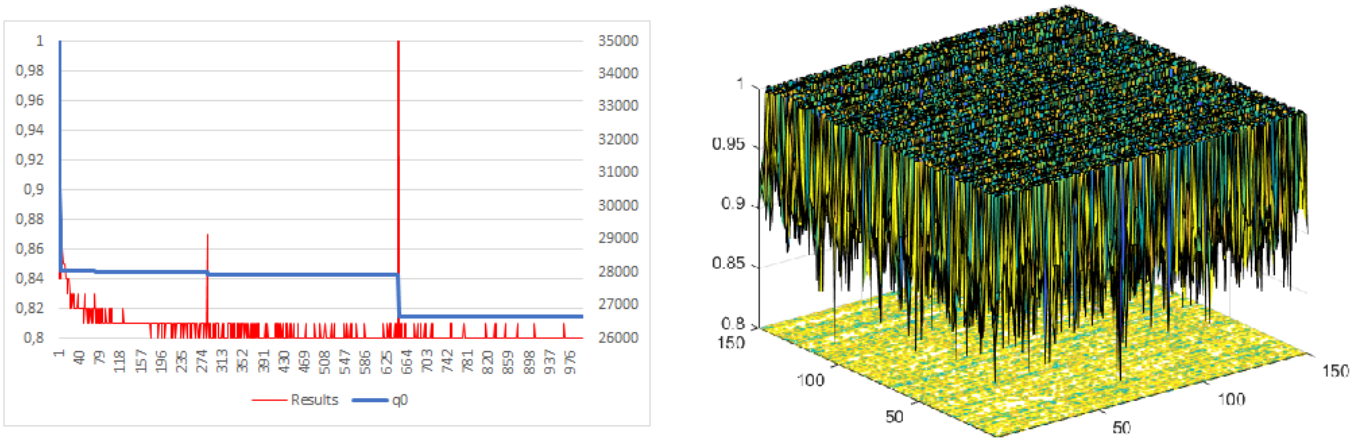

Figure 14 Results, change of q0, and pheromone map obtained in kroA150 solution with v-shaped transfer function

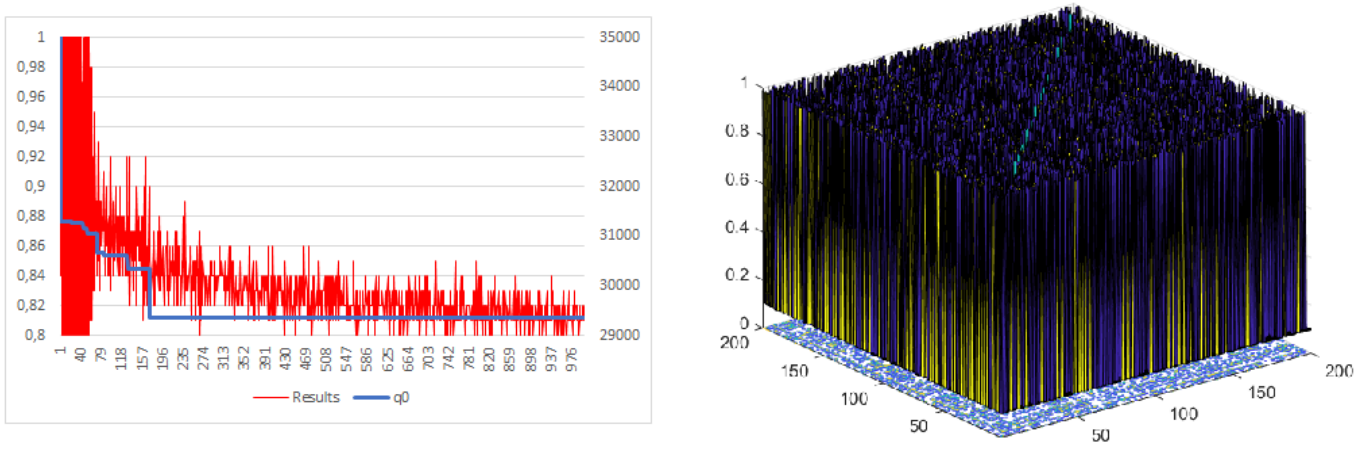

Figure 15 Results, change of q0, and pheromone map obtained in kroA200 solution with linear transfer function

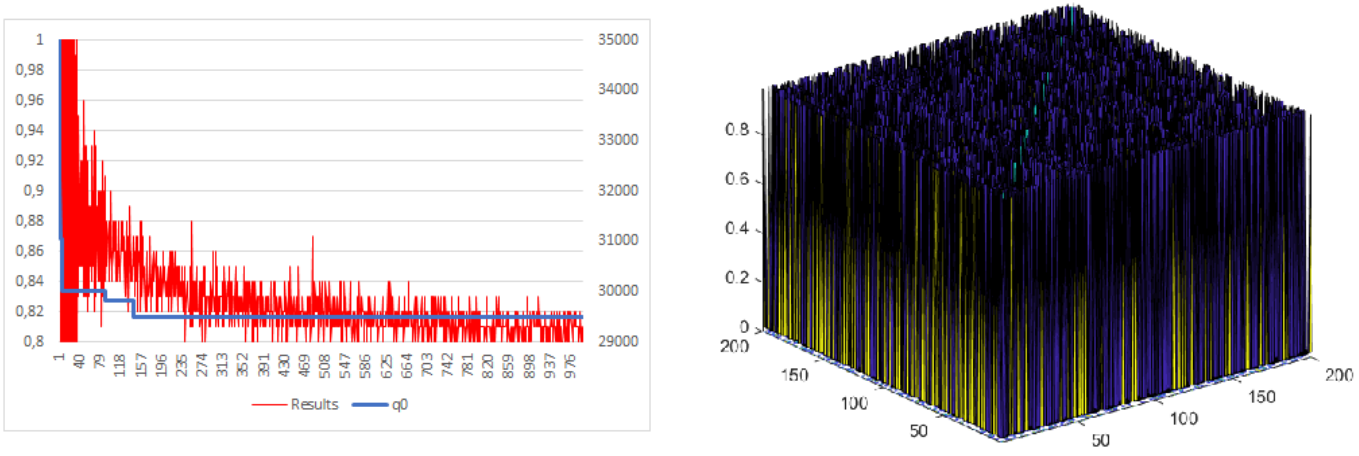

Figure 16 Results, change of q0, and pheromone map obtained in kroA200 solution with sigmoid transfer function

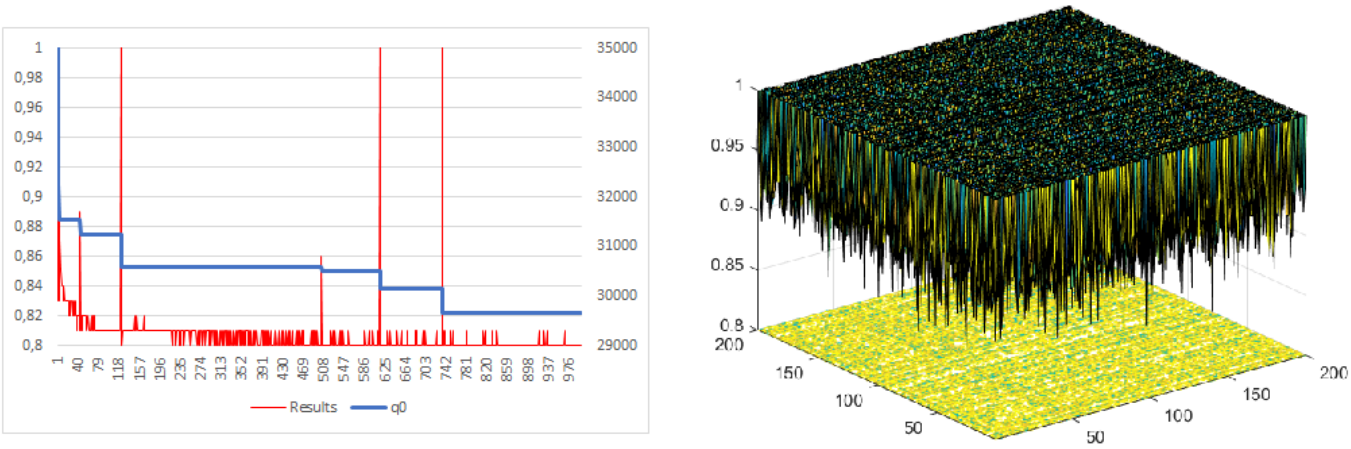

Figure 17 Results, change of q0, and pheromone map obtained in kroA200 solution with v-shaped transfer function

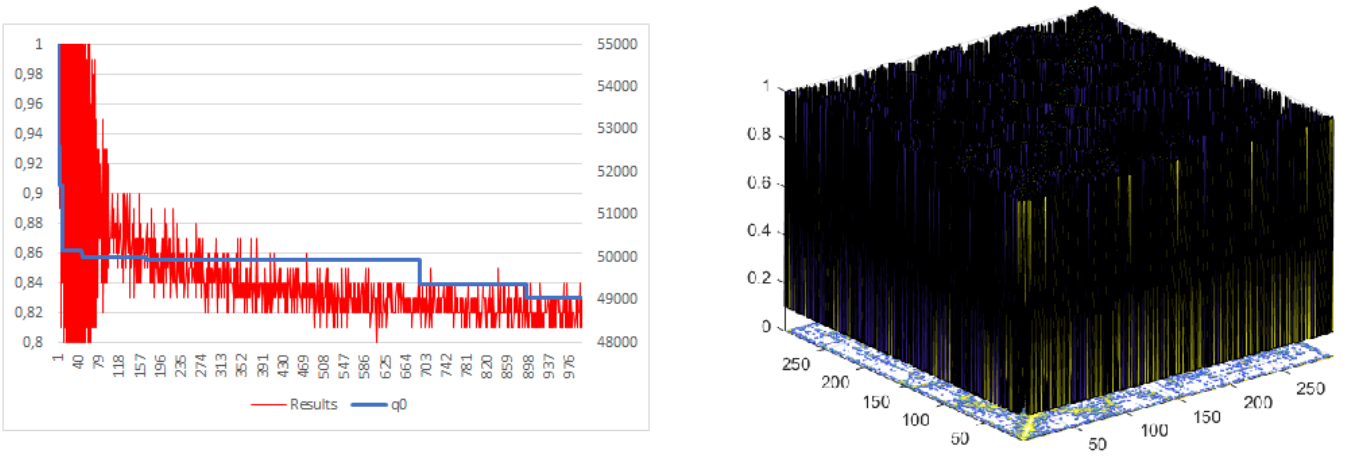

Figure 18 Results, change of q0, and pheromone map obtained in pr299 solution with linear transfer function

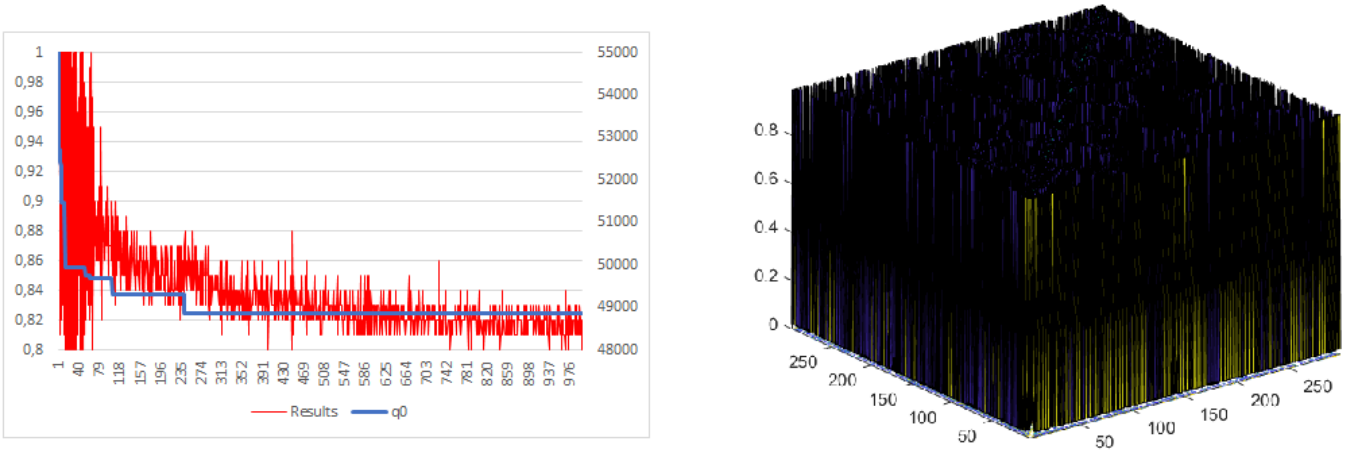

Figure 19 Results, change of q0, and pheromone map obtained in pr299 solution with sigmoid transfer function

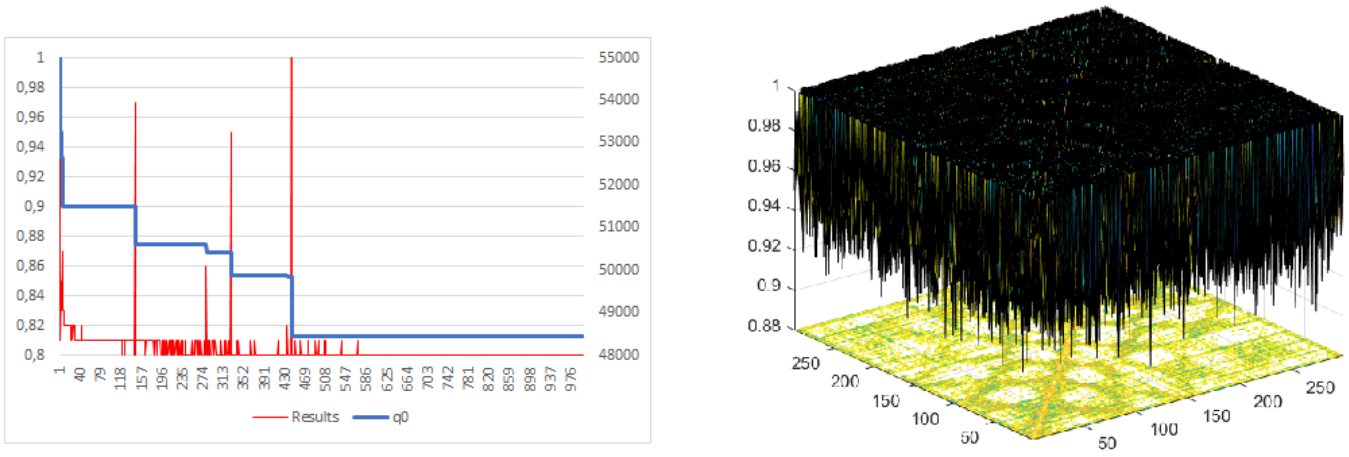

Figure 20 Results, change of q0, and pheromone map obtained in pr299 solution with v-shaped transfer function

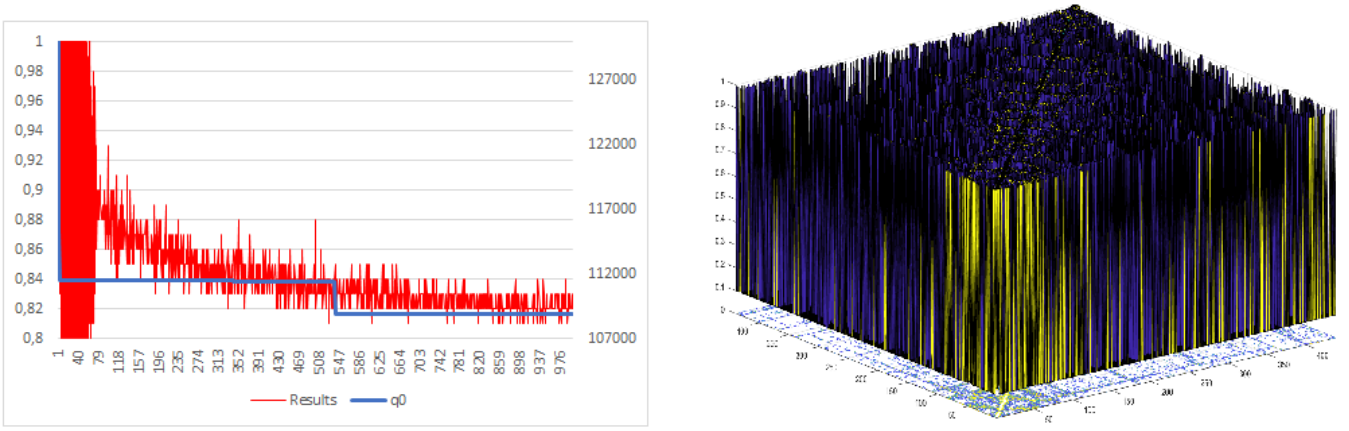

Figure 21 Results, change of q0, and pheromone map obtained in pr439 solution with v-shaped transfer function

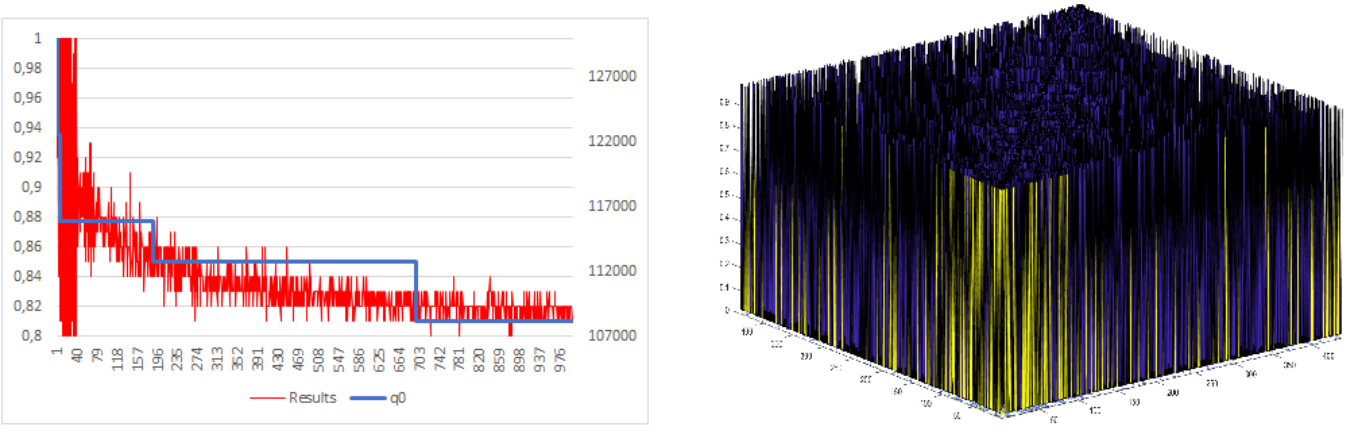

Figure 22 Results, change of q0, and pheromone map obtained in pr439 solution with v-shaped transfer function

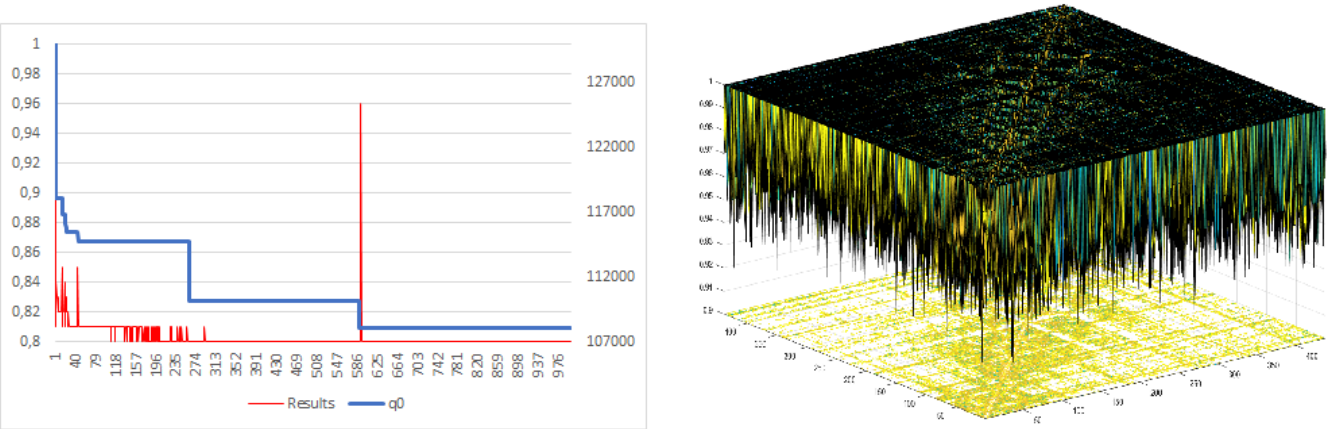

Figure 23 Results, change of q0, and pheromone map obtained in pr439 solution with v-shaped transfer function

4.3.2. Comparisons

The *best* and *worst* results obtained by the algorithms for test problems, the *means* and standard deviation (*SD*) values of the results obtained in 30 trials are presented in Table 3. Table 3 also includes the average CPU time as second, and the ratio of the best results to the best-known (*GAP*) calculated by (11).

$$GAP = \frac{100 * the\ best - known}{the\ best(S)} \quad (11)$$

The results produced by the algorithms were evaluated with statistical analysis. In this context, the Wilcoxon signed-rank method was selected to determine whether the difference is statistically significant. The value used is 0.05. The symbols "+", "=" and "-" indicate that the aACS-MBS performance is significantly “better”, “equal”, or “worse” than the compared algorithm, respectively. Notice that aACS-MBS is the control algorithm.

Table 3
Results of the algorithms for TSP benchmarks

		AS	ACS	ACS-MBS	aACS-MBS
	Best	457	444	431	426
	GAP (%)	91.25	93.92	96.75	97.89
	Worst	499	475	452	434
eil51	**Mean**	477.1	461.05	440.65	430.68
	SD	13.56	9.36	6.35	2.4
	p	0.0034(+)	0.0066(+)	0.0087(+)	
	Time	6.90	7.2	7.38	7.23
	Best	23055	21846	21282	21282
	GAP (%)	92.31	97.42	100	100
	Worst	25275	23251	21287	21695
kroA100	**Mean**	24313.18	22634.46	21773.73	21492.69
	SD	686.98	422.34	302.46	140.82
	p	0.16354(=)	0.05155(=)	0.03836(+)	
	Time	26.02	26.47	26.27	27.56
	Best	30222	28936	26530	27212
	GAP (%)	87.76	91.66	99.98	97.47
	Worst	33088	30842	27798	27746
kroA150	**Mean**	31610.54	29736.15	27044.14	27486.39
	SD	816.96	544.06	394.9	172.42
	p	0.00001(+)	0.00084(+)	0.22363(=)	
	Time	72.41	76.07	71.19	68.57
	Best	31844	30137	29377	29532
	GAP (%)	92.22	97.45	99.97	99.44
	Worst	34859	32060	30758	29967
kroA200	**Mean**	33543.95	31109.64	30108.25	29760.11
	SD	943.53	570.55	405.26	151.17
	p	0.02169(+)	0.11507(=)	0.30153(=)	
	Time	121.56	125.41	120.53	119.17
	Best	53761	54001	51242	50112
	GAP (%)	89.64	89.24	94.05	96.17
	Worst	58677	57359	53598	51073
pr299	**Mean**	55639.12	55497.73	52481.16	50614.59
	SD	1552.51	1038.08	683.69	284.02
	p	0.41294(=)	0.08851(=)	0.05592(=)	
	Time	361.86	365.79	357.11	353.91
	Best	119388	121460	113690	110961
	GAP (%)	89.81	88.27	94.31	96.63
	Worst	130285	129728	119029	114112
pr439	**Mean**	124467.36	125753.55	115943.09	113121.85
	SD	3163.9	2557	1732.17	669.84
	p	0.00001(+)	0.00001(+)	0.00006(+)	
	Time	1051.44	1100.56	1035.79	1047.81

4.4. Discussing

When the results obtained by the aACS-MBS algorithm (Figures 6-23) are examined, it is seen that the fluctuation in the *q0* value decreases as the problem size increases. For linear and sigmoid transfer functions, it can be said that the change in *q0* has similar fluctuation as the results. When a better solution is constructed, the variation in PM increases and this is reflected in the *q0* parameter. When the v-shaped transfer function is used, this fluctuation produces an impulse for *q0*. After

iterations obtaining a better solution, the *q0* value can be reduced to its previous level.

When pheromone maps are examined, it is seen that with the v-shaped transfer function, a more homogeneous pheromone distribution is formed compared to other transfer functions. When using the v-shaped transfer function, levels of pheromone trails at the edges are almost limited in the range of 0.8-1. Therefore, it can be considered that the algorithm constructs solutions using all edges in the searching process.

Examining Table 3, aACS-MBS produced better solutions compared to other ACO variants in 4 out of 6 benchmarks. The algorithm has the best mean value in 5 out of 6 problems. aABC-MBS achieved a statistically significant difference in 4 problems from AS, in 3 problems from ACS, and in 3 problems from ACS-MBS, however, it was not significantly behind in any problem. aACS-MBS also reached the best solution in 2 problems. aACS-MBS has produced solutions closer to the best solutions in the literature, falling behind these results by up to 3.83%.

Since TSP instances are NP-hard class problems, the larger the size of the problem, the more difficult it is to interpret the correlation between nodes. Given that the pheromone approach collects agents around good solutions, it can be more clearly understood that ACO algorithms are insufficient for exploration. Thus, the aACS-MBS is mostly failed to obtain the best solutions for large-scale problems. In this context, different transfer functions that make the exploitation-exploration balance more effective, or different dynamic approaches that determine *q0* can be developed. However, the fact that the algorithm results have lower mean and SD values prove that the algorithm can consistently generate successful solutions, not accidentally.

5. CONCLUSION

The pheromone approach, which can successfully analyze the correlations between solution components, provides an advantage to ACO algorithms, especially in COP solutions. Developing a more successful ACO version depends on a more accurate assignment of the pheromone trail and more accurate interpreting. Developed in parallel with this thought, ACS-MBS makes use of transfer functions to determine the pheromone value. Thus, the amount of pheromone is limited to 0-1. In this study, a solution proposal that dynamically updates the exploitation exploration balance of ACS-MBS using the *q0* parameter was introduced. The method has been tested on different sizes of TSP samples and has proven its applicability for COP solutions.

Acknowledgments

The author wants to thank the reviewers for all useful and instructive comments on the manuscript.

Funding

The author received no financial support for the research, authorship, and/or publication of this paper.

The Declaration of Conflict of Interest/ Common Interest

No conflict of interest or common interest has been declared by the author.

The Declaration of Ethics Committee Approval

The author declare that this document does not require an ethics committee approval or any special permission.

The Declaration of Research and Publication Ethics

The author of the paper declares that he complies with the scientific, ethical, and quotation rules of SAUJS in all processes of the paper and that he does not make any falsification on the data collected. In addition, he declares that Sakarya University Journal of Science and its editorial board have no responsibility for any ethical violations that may be encountered and that this study has not been evaluated in any academic

publication environment other than Sakarya University Journal of Science.

REFERENCES

[1] D. EKMEKCİ, "Optimizasyon Problemleri İçin Geliştirilmiş Feromonal Yapay Arı Koloni (gfYAK) Algoritması," *Eur. J. Sci. Technol.*, no. August, pp. 442–450, Aug. 2020.

[2] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Milano, Italy, 1991.

[3] M. Dorigo and T. Stützle, "Ant Colony Optimization: Overview and Recent Advances," in *International Series in Operations Research and Management Science*, vol. 272, 2019, pp. 311–351.

[4] B. Chandra Mohan and R. Baskaran, "A survey: Ant Colony Optimization based recent research and implementation on several engineering domain," *Expert Syst. Appl.*, vol. 39, no. 4, pp. 4618–4627, Mar. 2012.

[5] D. Ekmekci, "An Ant Colony Optimization Memorizing Better Solutions (ACO-MBS) for Traveling Salesman Problem," in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2019, pp. 1–5.

[6] Kwang Mong Sim and Weng Hong Sun, "Ant colony optimization for routing and load-balancing: survey and new directions," *IEEE Trans. Syst. Man, Cybern. - Part A Syst. Humans*, vol. 33, no. 5, pp. 560–572, Sep. 2003.

[7] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 1999, vol. 2, pp. 1470–1477.

[8] D. Zhao, L. Luo, and K. Zhang, "An improved ant colony optimization for the communication network routing problem," *Math. Comput. Model.*, vol. 52, no. 11–12, pp. 1976–1981, Dec. 2010.

[9] Z. Zhang and Z. Feng, "A novel Max-Min ant system algorithm for traveling salesman problem," in *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, 2009, vol. 1, no. 60875043, pp. 508–511.

[10] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. Syst. Man Cybern. Part B*, vol. 26, no. 1, pp. 29–41, 1996.

[11] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[12] M. Dorigo and T. Stützle, "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances," in *Journal of the Operational Research Society*, vol. 60, no. 7, 2003, pp. 250–285.

[13] E. Ateş, "Gezgin Satıcı Probleminin Çözümü Ve 3 Boyutlu Benzetimi," 2012.

[14] L. Gilbert, "The Traveling Salesman Problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, pp. 231–247, 1992.

[15] M. A. Tawhid and P. Savsani, "Discrete Sine-Cosine Algorithm (DSCA) with Local Search for Solving Traveling Salesman Problem," *Arab. J. Sci. Eng.*, vol. 44, no. 4, pp. 3669–3679, Apr. 2019.

[16] S. Rana and S. Ranjan Srivastava, "Solving Travelling Salesman Problem Using Improved Genetic Algorithm," *Indian J. Sci. Technol.*, vol. 10, no. 30, pp. 1–6, Sep. 2017.