



Contents list available at JMCS

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com



DRAFS: A Routing Algorithm based on Distributed Food Sources using Ant Colony Optimization

^aArash Ghorbannia Delavar, ^aEmetis Niazmand, ^aJavad Bayrampoor,

^bVahe Aghazarian

^a*Computer Science Department, Payame Noor University, PO BOX 19395-3697, Tehran, Iran*

^b*Islamic Azad University, Central Tehran Branch, Tehran, Iran*

*a_ghorbannia@pnu.ac.ir, emetis.niazmand@yahoo.com, javadbayrampoor@yahoo.com,
v_ghazarian@iauctb.ac.ir*

Article history:

Received July 2013

Accepted August 2013

Available online August 2013

Abstract

Distribution in routing algorithms based on food sources is a critical issue and the desired result could not be achieved through the old algorithms. For this purpose, participation of all sources through balanced distribution has been made in this proposed algorithm. In this paper, an improved routing algorithm based on distributed food sources is presented using the ant colony optimization. DRAFS algorithm helps us find the shortest path in order that we can generate a competence function, with the help of index parameters, to provide an optimal solution compared with other algorithms. Observing the distance and time parameters in finding the optimal solution, we introduce a target function which is accompanied by an increase in the algorithm efficiency. Comparing DRAFS algorithm with the previous routing algorithms, we have enjoyed the ants' collaboration mechanism that results in the ants with high efficiency guiding the ants with low efficiency. Consequently, an optimal quality is achieved in the algorithm compared with the existing solutions. Finally, these two techniques help us improve the efficiency and reliability of the algorithm and, in comparison with previous algorithms, provide a distributed food source to reduce time accessibility to the source in large datasets.

Keywords: Ant colony optimization, Distributed food sources, Ants' collaboration.

1. Introduction

The use of ant colony optimization (ACO) as an optimization technique in order to reach an optimal solution, or a set of approximate solutions to a range of problems in specific areas has evolved over the years. The first ACO algorithm was published by Marco Dorigo called Ant System (AS). ACO is metaheuristics used to find the best paths in a graph with respect to the predefined functional ones. This algorithm has been successfully applied to a variety of problems, and was first applied on the traveling salesman problem (TSP), where the goal was to find a closed tour of minimal length connecting n given cities with each city visited once and only once [13].

For many years, researchers and scientists, inspired by the environment, have managed to invent methods in order to address the existing issues, particularly issues that are NP-complete. Ant colony method is one of the technical approaches to solving NP-complete problems using the current state of the environment.

Using ACO algorithms with invariance property is definitely desirable for at least two main reasons: first, it reduces possible numerical problems in the implementations and therefore contributes to enhance the stability of the algorithm; second, it greatly improves the readability of the solution process [11].

Distribution is an important issue in generating efficient and optimal algorithms, such as ACO. In the distributed systems, each layer provides service for the upper layer, and of course the upper layer has to request this service. It should be noted that in the distributed system, cost is higher than centralized systems [7].

Basically, routings are divided into several categories: single-path and multi-path routings, source routing and next step, hierarchical and flat routings, centralized and distributed routings, data-centric and address-centric routings, QoS-based and best-effort routings, event-driven and queue-based routings and ant-net routing [20]. Ant-net routing using ant colony optimization technique provides a better result than others due to its real time computation and less control overhead. Comparing all routing algorithms with ACO, it is concluded that ants are relatively small, can be piggybacked in data packets and more frequent transmission of ants may be possible in order to provide updates of routing information for solving link failures. Routing in ACO is achieved by transmitting ants rather than routing tables.

ACO algorithm [18] minimizes complexity in the nodes at the expense of the optimality of the solution, and results to be particularly suitable in environments where fast communication establishment and minimum signaling overhead are requested. A fault tolerant routing protocol [19] using greedy ACO routing mechanism may tend to choose single path.

In the second part of this paper, the function of ant colony optimization and the related works done previously on it will be discussed. In the third part, the main task and the

formulas which include presenting the new algorithm and competence function will be analyzed in order to improve the effective function of the algorithm. Then, related simulations are conducted along with a comparison made between the proposed method with that of TSIACO and ACODA. In the last part, a conclusion will be presented.

2. Related Works

The basic idea of ACO-type algorithms applied to a combinatorial optimization problem to minimize a single objective function consists of four interwoven rules. First, each ant of the colony concurrently, independently, and asynchronously constructs a solution by selecting components and using a probabilistic rule (p) that considers both the experience acquired during the search (through τ , the trace of pheromone deposited) and heuristic information of the related components (η) [17].

$$P(i, j) = \frac{(\tau(i, j))^{\alpha} * (\eta(i, j, m_{ant}))^{\beta}}{\sum (\tau(i, k))^{\alpha} * (\eta(i, k, m_{ant}))^{\beta}} \quad (1)$$

The next stage optionally applies a local search method to improve the solutions. In the third stage, the pheromone trails are updated: the pheromone that stands for the weight of a graph edge evaporates after each iteration [2] (ρ is the evaporation rate), consequently the trace values are decreased and increased by deposited pheromone in the components used to construct solutions.

$$\tau^{new}(i, j) = (1 - \rho) * \tau^{old}(i, j) + \rho * \Delta\tau(i, j) \quad (2)$$

The net change in the pheromone value depends on the contributions of these two updating processes. $\Delta\tau$ equals Q/L_s , where Q is the quantity of trail laid by ants and L_s is the length of the shortest path.

Finally, in the last stage, the best solution found since the start of the algorithm (the best-so-far solution) is updated if a better solution is found.

Then, the algorithm iterates until a given stop condition is reached. ACO returns the best-so-far solution, when the stop condition is accomplished. Since the shorter paths have a higher traffic density, they can accumulate higher proportion of pheromone. Hence, the probability of ants following these shorter paths would be higher than that of those following the longer ones [1].

An ant colony optimization algorithm should have the following basic characteristics: an appropriate problem representation; a probabilistic transition rule which determines which node an artificial ant should visit next; a fitness function which determines the quality of the solution built by an artificial ant; a pheromone update rule which specifies how the modification of the pheromone trail laid along the edges of the graph will happen. For more

information and details about ACO-related methods the reader is advised to read the work of [9, 12].

2.1. Two-stage updating pheromone for invariant ant colony optimization (TSIACO) algorithm

In TSIACO algorithm, ordinal update rule is introduced. Compared with standard ACO algorithms, the quality function uses the solution order as its independent variable in one iteration. Using independent variable with problem dimension instead of solution as an independent variable for quality function can get an invariant ACO algorithm inherently [3].

Standard ACO algorithms with random proportional rule are independent of the scale of the problem under some conditions [11]. TSIACO algorithm also employs the random proportional rule. But the pheromone trail is updated stage by stage. In one stage, the first r iterative optimal solutions are employed to enhance search capability, and in another stage, only optimal solution is used to accelerate the speed of convergence.

In TSIACO algorithm, the pheromone trail is limited to the interval $[\tau_{min}, \tau_{max}]$, where in general case, τ_{max} is 0.999 and τ_{min} is 0.001. The limits in TSIACO are not to be updated every time when a new best-so-far solution is found.

There are two methods to determine the update stage. One is hard partition. That is, if $NC < mINmax$, use the first r solutions update pheromone trail; otherwise, use the iteration-best solution or the global-best solution update pheromone trail, where $Nmax$ is a maximum number of algorithm iterations, $0 < mI < 1$ is a real number, and NC is an iteration number. The similar partition method is also introduced in [8, 10]. The second method is soft partition. That is, if $cf < cI$, use the first r solutions update pheromone trail; otherwise, use the iteration-best solution or the global-best solution update pheromone trail, where cf is convergence factor, and cI is $0 < cI < 1$.

2.2. Ant colony optimization on a distributed architecture (ACODA)

In this paper we have made use of the distribution feature of ACODA to distribute food sources. ACODA allows the distributed, asynchronous and decentralized implementation of ACO. In this method the physical environment of the ants is conceptualized, represented and implemented as a distributed system [4]. The distribution of the ants is influenced more by the environment than by their interactions [14]. By distributing the food sources on several machines, the communication load is divided between them, thus resulting in the execution time being reduced. Whenever an ant completes a tour by reaching its destination node, it moves to the next district with more food source.

Various suggestions have been offered for ACODA and each of them carries its own particular design, competence function and update. Some of these suggestions have been presented in [5, 15, and 16].

Table1. Introducing the parameters used in DRAFS algorithm

Parameters	Definitions
m_{ant}	Number of ants
n	Number of nodes
d	Number of districts
v_i	Velocity of ants as a set of attributes
h_i	Size of ants as a set of attributes
a_i	Age of ants as a set of attributes
w_i	Weight of ants as a set of attributes ($i= 1, \dots, m_{ant}$)
m_h	Ants of high efficiency
m_l	Ants of low efficiency
F_j	Amount of food in nodes in each district ($j= 1, \dots, n$)
S	Shortest path from origin to destination
C_k	Proportion of F to S ($k= 1, \dots, d$)
HF	Total amount of food in nodes is high
HD	Total distance between nodes is high
LF	Total amount of food in nodes is low
LD	Total distance between nodes is low

3. Proposed algorithm

In this paper, a target function has been introduced that leads to a high efficiency of the proposed algorithm by optimizing the distance and time parameters existing in the previous algorithms. By making use of the invariance feature of ACO algorithm and, also, two-stage updating of pheromone and using the distributed food sources, we can access an optimal DRAFS algorithm. Taking into account the importance and complexity of the decentralized and distributed environments, a fast and optimal algorithm can play a key role in increasing the productivity of a system. In order to optimize the previous algorithms, DRAFS uses the heuristic information (η) parameter. η is considered one of the most important factors in ACO. Presumptions and formulas have been presented to calculate various heuristics and results have been achieved through simulations. Considerable results based on a new definition of η and calculating its value have also been produced.

We have been able to prevent undesired solutions (as far as the amount of distance and time is concerned) by exerting a set of attributes on ants by encouraging collaboration among weak ants (ants of low efficiency), and strong ants (ants of high efficiency). Consequently, this has reduced the existing time and calculation complexities in the previous cases.

Table 1. h (size of ants), v (velocity of ants), a (age of ants), and w (weight of ants)

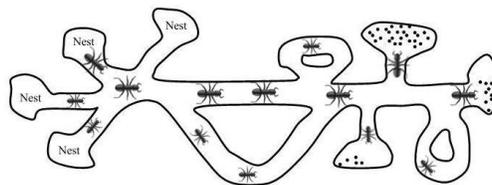
Type of ants (efficiency)	m	Set of attributes
	[0.5 3]	Low (h,v) , High (a,w)
	[3 4]	High (h,v) , Low (a,w)

Equations related to DRAFS algorithm will be mentioned later.

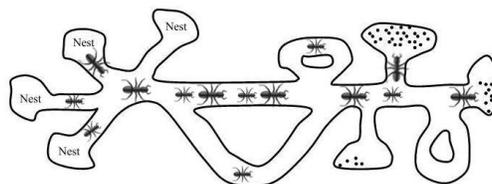
$$m_i = (v_i * h_i) / (a_i + w_i) \quad i=1, \dots, m_{ant} \quad (3)$$

Parameter m, here, is the set of attributes implemented on ants which includes speed (v), size (h), age (a), and weight (w). In Table 1, all parameters used in proposed algorithm have been shown. In Table 2, by using these parameters and based on the values defined for them, we have assumed intervals 0.5 to 4 for parameter m.

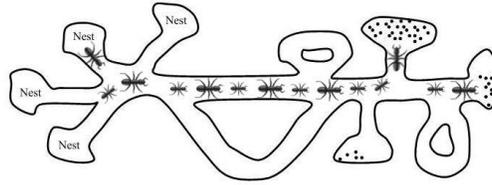
These intervals can vary in different experiments.



(i) Generating solutions by weak and strong ants individually



(ii) Encouraging collaboration among weak and strong ants to find the shortest path



(iii) Finding the best solution considering the food source with the largest mass

Fig1.  (Ants of high efficiency),  (Ants of low efficiency)

By encouraging collaboration among weak and strong ants, undesirable solutions are prevented which is defined by M . m_l represents ants whose value of m is at 0.5 to 2 intervals, that is, ants of low efficiency; and m_h represents ants whose value of m is at 2 to 4 intervals, that is, ants of high efficiency. Every time m_l with the least value is added to m_h with the highest value and this goes on.

$$M = m_h + m_l \quad (4)$$

As shown in Figure 1, first in (i), ants start finding solutions. Considering the power and strength of these ants, those with better attributes choose better solutions compared with ants of low efficiency. As this goes on, however, in (ii), by encouraging collaboration among ants of high efficiency and ants of low efficiency, longer solutions and lower access to food source are prevented. Finally, in (iii), ants manage to find the best path with shorter length and access to more food source, and consequently the efficiency increases.

By distribution we divide the structure of the related instances into districts. In this article we refer to Sh07, Lau15, Bays29, Dantzig42, and Wg59 instances selected from TSPLIB [6].

As shown in Figures 2, 3 and 4, the graph respecting Sh07, Lau15, and Bays29 consists of 7, 15, and 29 nodes respectively where the amount of food source from node (i) to node (j) and length of these two has been shown as ($F(i), F(j), D_{i,j}$) on each edge. Values of F have been considered for the processors of which the intervals are at 0.5 to 5 that we have assumed in this experiment. By calculating the ratio of the weight of processors ($\sum F$) to the shortest path in that district, priorities are given to these districts called C parameter. The more this ratio (C), the higher the priority of that district is. Because this higher ratio shows that the amount of food is more in that district (HF) and the distance to be traveled is less (LD). The less this ratio (C), the less the priority of that district is. Because this lower ratio shows that the amount of food is less in that district (LF) and the distance to be traveled is more (HD). HF is attributed to a district when the sum of foods is at 12 to 20 intervals, and LF is

attributed to a district when the sum of foods is at 1.5 to 11.5 intervals, supposing all the nodes contain food. All the above intervals were assumed by the authors, which can vary in different experiments. In Table 3, 4, 5 and 6 the values for the instances have been mentioned. Priorities are given to the districts according to these values. k is the number of districts, Sum of Foods is the total amount of food sources in each district, and Shortest Path is the shortest distance between the nodes in each district a, b, c, \dots

Therefore, efficiency will be increased in the algorithm by moving from the district of higher priority, that is, the district with larger C , to the district of lower priority, that is, the district with smaller C .

$$C_k = \sum F_j / S \quad k=1, \dots, d \tag{5}$$

j doesn't include all the nodes but covers only the nodes in each district.

Table3. Data set of Sh07 with $k=2$

Districts	Sum of Foods	Shortest Path	C
a	12	83	0.144
b	5	47.2	0.105

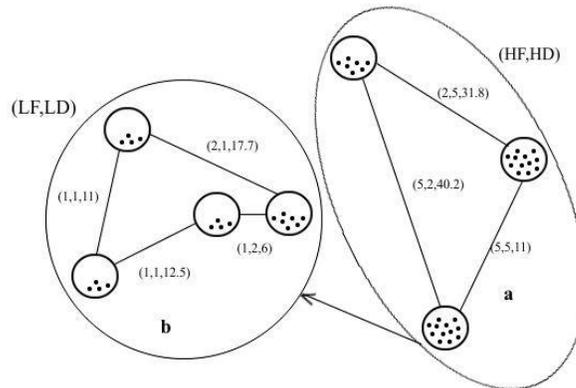


Fig2. Graph of Sh07

Regarding Sh07, C is calculated for each district taking into account the sum of foods, which we had defined earlier, and length of edges in this instance, as shown in Table 3. That is, we move from the district of high priority, a , to the district of low priority, b (see Figure 2).

Table4. Data set of Lau15 with k=4

Districts	Sum of Foods	Shortest Path	C
a	18	71	0.253
b	16	108	0.148
c	7	49	0.142
d	6	82	0.073

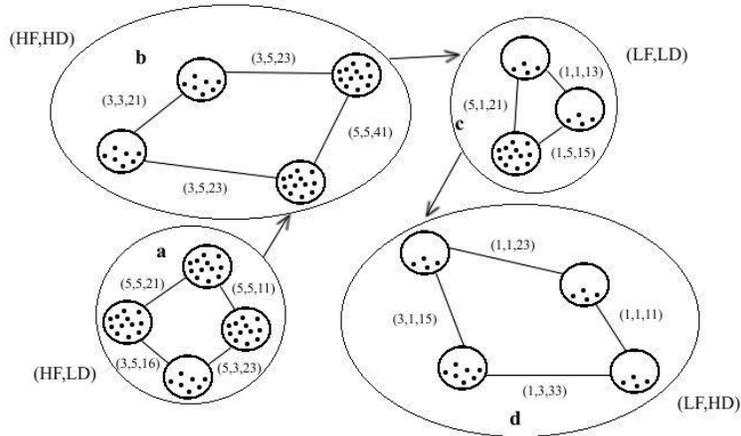


Fig3. Graph of Lau15

As far as Lau15 is concerned, it is divided into four districts and C is calculated for each one as shown before (see Table 4). Movement from one district to the other one has been shown in Figure 3.

Table5. Data set of Bays29 with k=6

Districts	Sum of Foods	Shortest Path	C
a	13	414	0.031
b	12.5	404	0.03
c	13	436	0.029
d	12.5	588	0.021
e	8.5	474	0.017
f	10.5	721	0.014

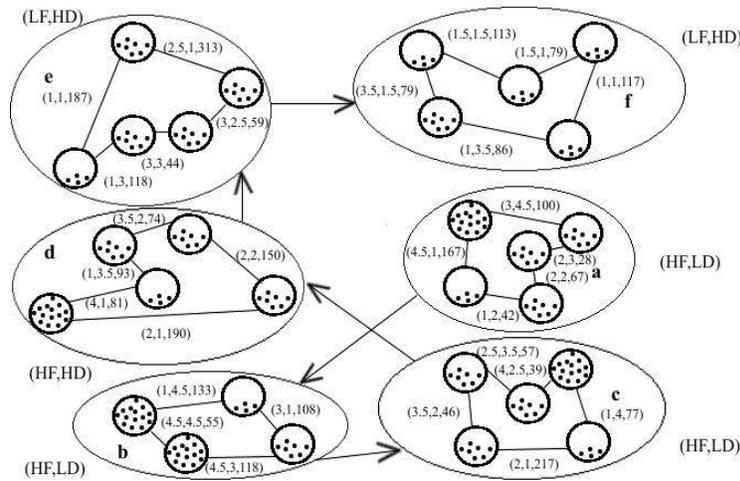


Fig4. Graph of Bays29

Six, ten, and twelve districts are divided into Bays29, Dantzig42, and Wg59 respectively as above. As observed in the instances discussed earlier, so far we have moved from districts with high foods to those with low foods considering, of course, the distance between the nodes. In some cases, in order to avoid very long distances, we may have to overlook districts with high foods and select the ones with lower foods. Furthermore, in most cases, according to the routing algorithms, we start from districts with high distance to those with lower distance. However, in instances like Sh07, in order to improve the algorithm, due to the lack of sufficient food sources in the district with low distance we may have to ignore it and select district with a little longer distance but with more food source. At any rate, the criteria for prioritization of districts will constantly be parameter C .

The pseudo-code respecting the execution of the proposed algorithm has been shown in Figure 5:

```

Set parameters, initialize pheromone trails
Consider attributes set for ants;
m= getM();
M= mh+ml;
For each districts:
i= 1,...,k;
i= i+ 1;
Ci= ΣFj /S;
Prioritization based on the value of C;
While Iteration <n_cycle
Construct ants solutions;
p= getP();
Update pheromone trail
End
End
    
```

Fig5. Pseudo-code of DRAFS Algorithm

Therefore, by implementing M and C parameters in the heuristic information, the formula will be as follows:

$$\eta = (g * M * C_i) / Task \quad (6)$$

Where $g > 0$ is an arbitrary constant.

The heuristic information for the distance traveled from the origin to the destination is calculated as follows:

$$\eta = (g * M * C_i) / Task_D \quad (7)$$

The heuristic information for the time of Task implementation and the time taken to go from the origin to the destination is calculated as follows:

$$\eta = (g * M * C_i) / Task_T \quad (8)$$

In the algorithm, if $q < q_0$, where q is selected randomly, the best Task is selected regarding the competence function, otherwise task selection will be carried out randomly.

After each ant selects the next node using Formula (1), of which η has been defined as above, pheromone value is updated by a local pheromone updating rule. Then, after all ants have completed their solutions at the end of each iteration, pheromone values corresponding to the best-so-far solution are updated by a global pheromone updating rule using Formula (2).

Once all the districts have been covered, the algorithm will end.

The flowchart respecting the execution of the proposed algorithm has been shown in Figure 6:

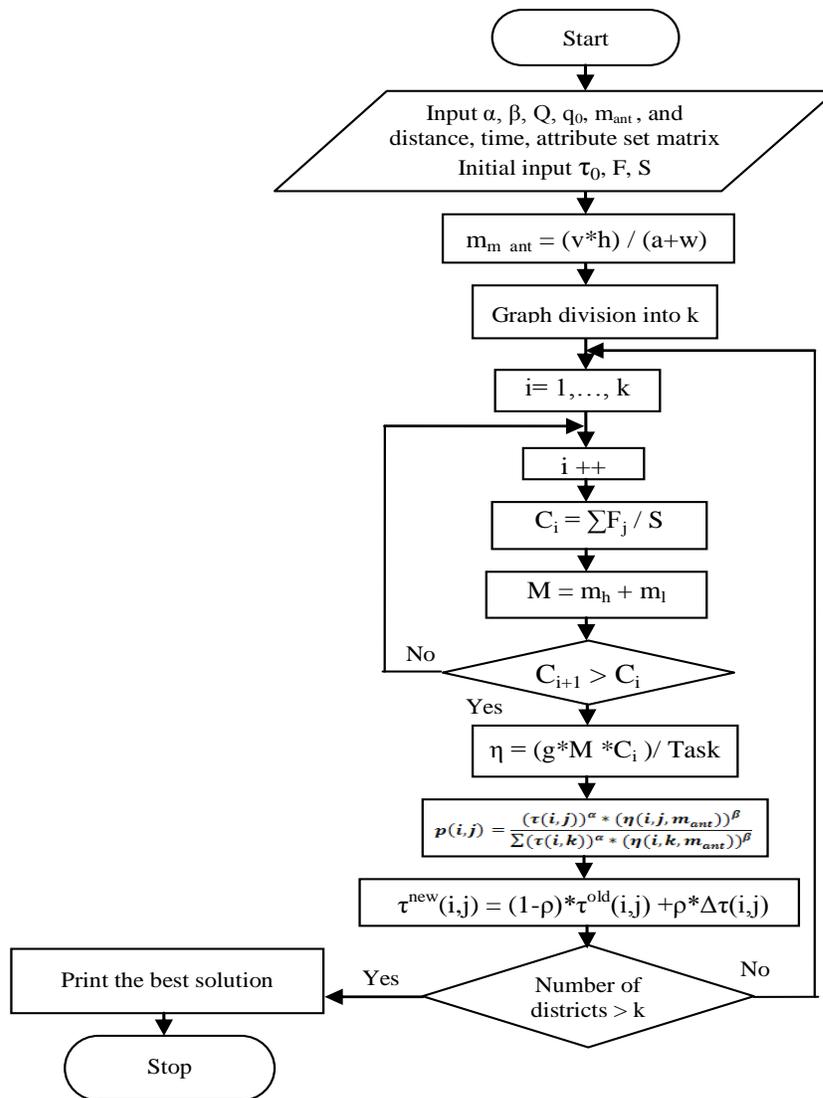


Fig6. DRAFS Flowchart

4. Simulation and Analysis

In this part the implementation of the previous algorithms and the proposed one has been taken into consideration in programming and simulation environments. In the end, a comparison and evaluation of the results achieved from the implementation of the algorithms are presented in the form of graphs based on the distance and time for the previous algorithms and the proposed one. Number of iterations was set as 100. These experiments have been carried out on five instances which are, of course, applicable to any other instance.

Since experiments on small datasets such as Sh07 and Lau15, compared with larger ones, like Bays29, Dantzig42, and Wg59, do not have significant effect and are not of considerable improvement; therefore we have just shown the experimental results excluding the simulations of small datasets. The reason for this is that distributed food sources are more usable for large datasets.

Results of the simulation along with the distance and time optimization have been shown in Table 6.

Compared with the previous algorithms, DRAFS indicates less time complexity and more efficiency.

4.1. Choosing the parameters of the algorithm

A range of parameters affect the proposed algorithm. Some of them play a more important role in the execution of the algorithm. Simulation and analysis in this paper based on giving preliminary value to the parameters is as follows: $n_{cycle}=100$, $\alpha=1$, $\beta=6$, $m_{ant}=100$ and $\tau_0=0.05$.

Experiments have been carried out based on the varied rates of evaporation and the results achieved will be discussed later. It should be noted that, taking into account the results, we have used the lower evaporation rate here, that is, $\rho = 0.001$.

Four controlling variables play an important role in the algorithm presented: α , pheromone effect; β , heuristic information effect; ρ , pheromone evaporation rate; and τ_0 , threshold for pheromone at the beginning of the algorithm. The influence of these four parameters has been studied thoroughly in the algorithm.

Simulations in Figures 7, 8, and 9 show the distance and time optimizations for Bays29, Dantzig42, and Wg59.

Table6. Experimental results

Instance	Algorithm	k	D_{min}	D_{avg}	T_{min}	T_{avg}
Sh07	TSIACO	2	66	44	10.86	7.63
	ACODA		66	55	13.2	11
	DRAFS		66	51	11	9.02
Lau15	TSIACO	4	345	288	70.4	59.8
	ACODA		385	290	72.8	65.6
	DRAFS		380	293	71.8	63.7
Bays29	TSIACO	6	2111	2182.7	669.5	706
	ACODA		2152	2199	661	678.5
	DRAFS		2047	2109	647	703
Dantzig42	TSIACO	10	845	867	547	569.2

	ACODA		848	884.5	562	576
	DRAFS		818	887.5	538	555.5
Wg59	TSIACO	12	1192.5	1304.7	721	751
	ACODA		1204	1268	718	753
	DRAFS		1127	1218.5	708	737.5

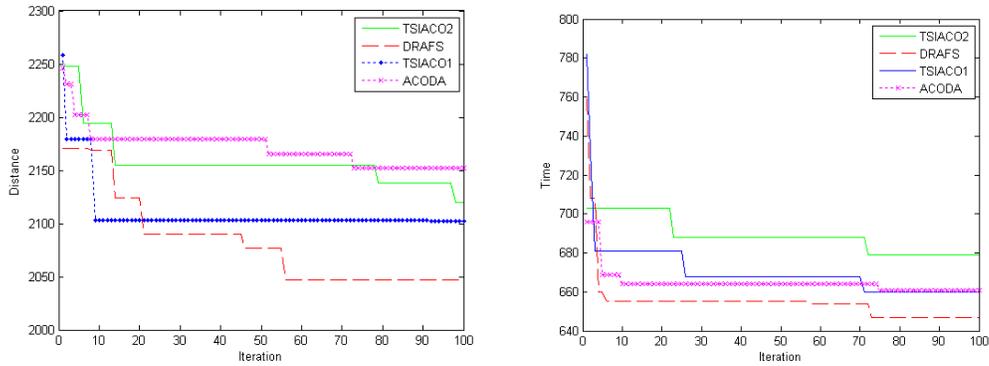


Fig7. Graphical representation of distance and time optimization for Bays29

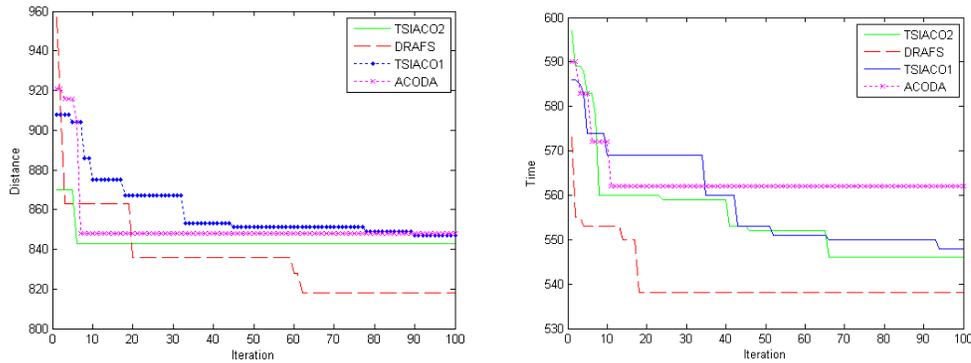


Figure8. Graphical representation of distance and time optimization for Dantzig42

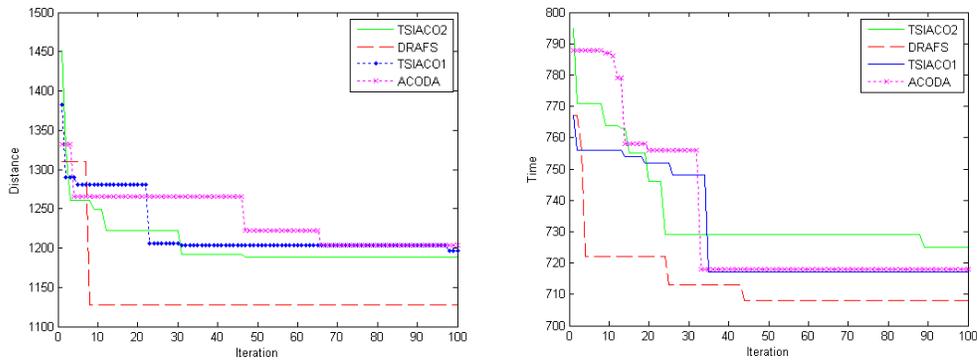


Figure9. Graphical representation of distance and time optimization for Wg59

5. Effect of ants collaboration in generating solutions on ρ

In the first step, by using Bays29, a mode is considered where a solution is found only by implementing a set of attributes on the ants. In this case, by carrying out various experiments based on varied evaporation rates, we prove that an optimal solution can be reached with larger evaporation rate traveling less distance in shorter time. Therefore, an evaporation rate of higher amount has to be used when ants do not collaborate with each other. As shown in Figure 10, using the evaporation rate of 0.005 has led to a better result.

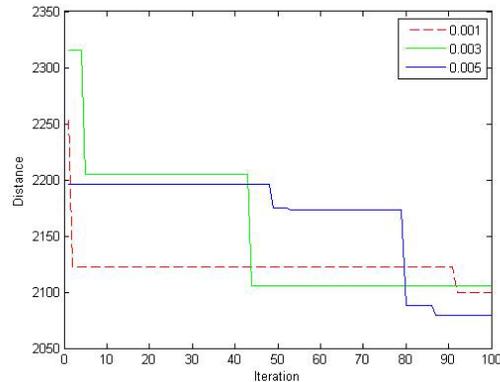


Fig10. Without ants' collaboration

Now we consider a mode where a solution is generated by both implementing a set of attributes on ants and encouraging collaboration among them. In this case, by carrying out various experiments based on varied evaporation rates, we prove that the less the evaporation rate is, we can reach an optimal solution by traveling less distance and using shorter time. Therefore, when ants collaborate with each other, an evaporation rate of lower amount should be used. As shown in Figure 11, using the evaporation rate of 0.001 has led to a better result.

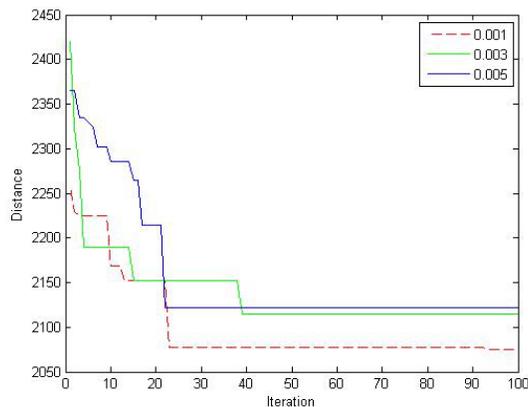


Fig11. With ants' collaboration

6. Conclusion

Results of the experiments indicate that by making use of the distributed food sources, DRAFS has managed to show more improvements than other routing algorithms. This high efficiency and less complexity in the proposed algorithm have led to a high improvement compared with TSIACO and ACODA. The average improvement in the proposed algorithm for distance optimization amounts to 3.77 and for time optimization is 2.29 compared with TSIACO. And as far as ACODA is concerned, the average improvement in the proposed algorithm for distance optimization amounts to 5.04 and for time optimization is 2.47.

References

- [1] C.J. Liao, Y.L. Tsai, C.W. Chao, An ant colony optimization algorithm for setup coordination in a two-stage production system, *Applied Soft Computing*, 11 (2011) 4521–4529.
- [2] V.A. Gromov, A.N. Shulga, Chaotic time series prediction with employment of ant colony optimization, *Expert Systems with Applications*, 39 (2012) 8474–8478.
- [3] Z. Zhang, Z. Feng, Two-stage updating pheromone for invariant ant colony optimization algorithm, *Expert Systems with Applications*, 39 (2012) 706–712.
- [4] S. Ilie, C. Bădică, Multi-agent approach to distributed ant colony optimization, *Science of Computer Programming*, 78 (2013) 762–774.
- [5] S. Ilie, A. Bădică, C. Bădică, Distributed agent-based ant colony optimization for solving traveling salesman problem on a partitioned map, in: *Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS '11*, ACM, 2011, 23:1–23:9.
- [6] G. Reinelt, Tsplib — a traveling salesman library. *ORSA Journal on Computing*, 3 (1991) 376–384.
- [7] A. Ghorbannia Delavar and V. Aghazarian and S. Sadighi, ERPSD: A New Model for Developing Distributed, Secure, and Dependable Organizational Software, *CSIT* (2009).
- [8] A. Puris, R. Bello, F. Herrera, Analysis of the efficacy of a Two-Stage methodology for ant colony optimization: Case of study with TSP and QAP *Expert Systems with Applications*, 37 (2010) 5443–5453.
- [9] R.J. Mullen, D. Monekosso, S. Barman, P. Remagnino, A review of ant algorithms, *Expert Systems with Applications*, 36 (2009) 9608–9617.
- [10] A. Puris, R. Bello, Y. Martinez, A. Nowe, Two-stage ant colony optimization for solving the traveling salesman problem. In *Nature inspired problem-solving methods in knowledge engineering, Second international work conference on the interplay between natural and artificial computation, La Manga del Mar Menor, Spain, IWINAC (2007)* 307–316.

- [11] M. Birattari, P. Pellegrini, M. Dorigo, On the invariance of ant colony optimization, *IEEE Transactions on Evolutionary Computation*, 11 (2007) 732–742.
- [12] M. Dorigo, C. Blum, Ant colony optimization theory: A survey, *Theoretical Computer Science*, 344 (2005) 243-278.
- [13] Y. Bai, W. Zhang, Z. Jin, An new self-organizing maps strategy for solving the traveling salesman problem, *Chaos, Solitons and Fractals*, 28 (2006) 1082–1089.
- [14] S. Depickere, D. Fresneau, J. Deneubourg, Effect of social and environmental factors on ant aggregation: A general response? *Journal of Insect Physiology*, 54 (2008) 1349–1355.
- [15] S. Ilie, C. Bădică, Distributed multi-agent system for solving traveling salesman problem using ant colony optimization. in: M. Essaïdi, M. Malgeri, C. Bădică (Eds.), *Intelligent Distributed Computing IV*, in: *Studies in Computational Intelligence*, Springer, Berlin/Heidelberg, 315 (2010) 119–129.
- [16] S. Ilie, C. Bădică, Effectiveness of solving traveling salesman problem using ant colony optimization on distributed multi-agent middleware, in: *Proceedings of the International Multi conference on Computer Science and Information Technology*, (2010) 197–203.
- [17] M. Pedemonte, S. Nesmachnow, H. Cancela, A survey on parallel ant colony optimization, *Applied Soft Computing*, 11 (2011) 5181–5197
- [18] R. Laura, B. Matteo, R. Gianluca, On ant routing algorithms in ad hoc networks with critical connectivity, *Ad Hoc Networks (Elsevier)*, 6 (2008) 827–859.
- [19] S. Misra, S.K. Dhurandher, M.S. Obaidat, K. Verma, P. Gupta, A low-overhead fault-tolerant routing algorithm for mobile ad hoc networks: A scheme and its simulation analysis, *Simulation Modelling Practice and Theory*, 18 (2010) 637–649.
- [20] A. Ghorbannia Delavar, S. Hoseyny, R. Maghsoudi, BCO-Based Optimized Heuristic Strategies for QoS Routing, *The Journal of Mathematics and Computer Science*, 5 (2012) 105-114.