Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 7, No. 3, August 2016), Pages: 77-91
www.jacr.iausari.ac.ir

An Effective Ant Colony Algorithm for the Traveling Salesman Problem

Majid Yousefikhoshbakht✉[1], Azam Dolatnejad[2]
1) Young Researchers & Elite Club, Hamedan Branch, Islamic Azad University, Hamedan, Iran
2) Young Researchers & Elite Club, Tehran North Branch, Islamic Azad University, Tehran, Iran
khoshbakht@iauh.ac.ir; a_dolatnejad@aut.ac.ir

Received: 2015/03/04; Accepted: 2016/04/23

Abstract

The traveling salesman problem (TSP) is a well-known combinatorial optimization problem and holds a central place in logistics management. The TSP has received much attention because of its practical applications in industrial problems. Many exact, heuristic and metaheuristic approaches have been proposed to solve TSP in recent years. In this paper, a modified ant colony optimization (MACO) is presented which possesses a new strategy to update the increased pheromone, a candidate list and a mutation operation to solve the TSP. In addition, some local search algorithms are utilized as an effective criterion and only a global updating is used in order to increase pheromone on the edges of the best route. The proposed metaheuristic algorithm is tested on the well-known TSP instances involving 14 benchmark problems from 48 to 200 nodes. The computational results show that our algorithm is better than other meta-heuristic algorithms in terms of solution quality. Furthermore, the gap of the proposed algorithm stays on average almost 1% of the execution time and also the most best known solutions of the benchmark problems are found.

Keywords: *Traveling Salesman Problem, Ant Colony Optimization, Global Updating, NP-hard Problems.*

1. Introduction

The traveling salesman problem (TSP) is one of the combinatorial optimization problems which has great importance in other sciences due to its widespread application in other problems [1]. In this problem a salesman starts to move from a certain node called depot and returns it after visiting *n* nodes so that each node is visited only once. The purpose is to find the shortest path in order to keep the cost at minimum. The existence of several TSP inspired problems like vehicle routing problem [2], the capability of changing other problems like sequencing job problem into this problem and solving them with the existing algorithms, and using this problem as a benchmark for testing new algorithms are the reasons which have attracted a lot of researchers' attention and encouraged them to analyze it in recent years [3]. In practice, the basic TSP is extended with constraints, for instance, on the allowed capacity of the salesman, length of the route, arrival, departure and service time, time of collection and delivery of goods. The extended classes of distance constrained TSP (DCTSP), TSP with time

windows (TSPTW), backhauls TSP (TSPB), pickup and delivery TSP (TSPPD), and simultaneous pickup and delivery TSP (TSPSPD) are some versions of the TSP. The main goal in all TSP problems is to obtain the minimal transportation cost.

The TSP can be shown by a complete weighted graph $G = (V, E)$ with V being the set of n nodes and E being the set of edges fully connecting the nodes in the graph G. In this graph, each edge E is given a weight d_{ij} which represents the distance between cities i and j. It may be important to emphasize that the distance between cities may be symmetric (where the distances between the cities are the same in either going or returning from the cities) or asymmetric (where due to some restrictions, possibly, due to one-way lanes or other reasons, the distances of going from city A to city B may not be the same). The basic minimization equation for TSP is given n cities and their coordinates, find an integer permutation$\pi = C_1, C_2, C_3,\ldots, C_n$ with C_n being the city "n", our task is to minimize the sum of the cities. Consider

$$f(\pi) = \sum_{i=1}^{n-1} d(c_i, c_{i+1}) + d(c_n, c_1)$$

Here, (C_i, C_{i+1}) represents the distance between city "i" and city $i + 1$ and (C_n, C_1) is the distance of city "n" and city "1." The aim is to find the shortest path between the adjacent cities in the form of a vector of cities.

Solving the TSP was an interesting problem during recent decades. Almost every new approach for solving engineering and optimization problems has been tested on the TSP as a general test bench. The methods for solving this problem like other operation research problems can be classified as exact and heuristic algorithms [4]. Exact approaches such as Lagrangian algorithm [5], Branch-and-Bound algorithm [6], and exact exponential algorithm [7] are successfully used only for relatively small problem sizes but they can guarantee optimality based on different techniques. These techniques use algorithms that generate both a lower and an upper bound on the true minimum value of the problem instance. If the upper and lower bound coincide, a proof of optimality is achieved.

Although there are other exact algorithms which investigate fewer solutions for the TSP, it is almost impossible even in such cases to find an optimum solution within a satisfactory time limit. In other words, when the size of the TSP problem increase, the complexity of the problem increases rapidly too and we have to solve these problems by heuristic algorithms. In contrast, despite the fact that heuristic methods cannot find an optimum solution, they are able to find a sub-optimum solution within a satisfactory time limit. In other words, in such algorithms finding an optimum solution is sacrificed in favor of finding the solution within a time limit. Furthermore, since such algorithms do not enjoy a good structure for escaping from local optimum points, they cannot converge to good solutions. Therefore, other versions of heuristic algorithms which employ random structures for finding solutions have been suggested in the last few decades. Such algorithms called meta-heuristic can escape from local optimum points as much as possible and converge to good solutions. The tabu search [8], genetic algorithm [9], african Buffalo Optimization [10], ant colony optimization (ACO) [11], bee colony algorithm [12], memetic algorithm [13], neural network [14] and particle swarm optimization [15] are examples of meta-heuristic algorithms.

Recently, many researchers have found that the employment of hybridization in optimization problems can improve the quality of obtained solutions in comparison with heuristic and meta-heuristic approaches. Since hybrid algorithms have greater ability to

find an optimal solution, they have been considered by researchers and scientists in recent years [16-17]. Besides, since the TSP is an NP-hard combinatorial optimization problem, metaheuristic methods have proved to be more suitable for dealing with such problem in terms of solution quality and computational cost. So, this article presents a modified version of ACO (MACO) for solving the TSP. The proposed algorithm uses only a global updating, which will increase pheromone on the edges of the best routes. Furthermore, a candidate list and some local search algorithms are combined by MACO for obtaining high-quality solutions. It can be concluded that making use of some proposed modifications leads the algorithm avoids risky edges and moves toward more appropriate edges and finds better solutions as a result. The proposed metaheuristic algorithm is tested on the well-known TSP instances from 48 to 200 customers. The computational results show that our MACO yields better than other meta-heuristic algorithms in terms of solution quality. Furthermore, the gap of the MACO stays on average almost 1% of the execution time and also the most best known solutions of the benchmark problems are found by the proposed algorithm.

The structure of the remainder of the paper is as follows. In the next section, the proposed idea based on ACO is explained in great detail. In addition, using a new candidate list, building the solution simultaneously using a new transition rule, applying three powerful local search techniques to improve the solution, and proposing a new method for updating global pheromone information which are four main steps of MACO described in more detail in the same section. In section 4, the proposed algorithm is compared with some of the other algorithms on standard problems belonging to TSP library. Finally, some concluding remarks are given in section 5.

2. The Proposed ACO

The ACO is inspired on an analogy with real life behavior of a colony of ants when looking for food. In their search they mark the trails they are using by laying a substance called pheromone. The amount of pheromone in a path lets other ants to know if it is a promising path or not. Initially proposed by Marco Dorigo in 1992, the first algorithm called ant system (AS) aimed at searching for an optimal path between two nodes in a graph [18]. In this algorithm, ants are procedures that build solutions to an optimization problem. According to how the solution space is explored some values are recorded in a similar way as pheromone acts, and objective values of solutions are associated with food sourced.

The TSP was the first problem to be solved by AS (Figure 1) because it was well adapted to this algorithm and it is a widely used problem on which a lot of algorithms have been implemented [19]. Since the AS could not produce satisfactory results compared with meta-heuristic algorithms of that time, scientists attempted to develop a newer version of the algorithm which was able to produce better results. In order to achieve this goal, algorithms like elite ant system (EAS), ant colony system (ACS), rank based ant system (RAS), and max-min ant system (MMAS) were developed. For more information on the modified versions of the mentioned algorithms read [20-25].

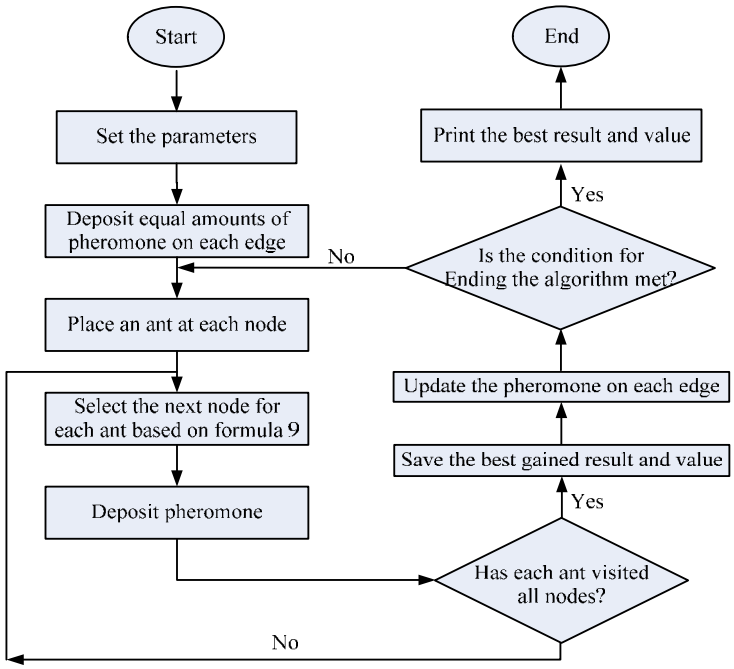

Figure 1. The AS algorithm for solving the TSP

Although the new versions of the ACO have better quality than the AS, these algorithms might lead to an early stagnation and premature convergence to suboptimal regions. In addition, before these algorithms finish a complete global search, tends to the local search and consequently relatively weak results are attained. Encouraging the best solution during the first steps of the ACO versions algorithm, where decisions are almost by chance, should be little but gradually increase. On the other hands, most successful meta-heuristic methods have paid attention to global search and search in the whole solution space as far as possible. As the algorithm proceeds, they move to better solutions and alter the global search for local. So, we have considered the point too and a modified ACO is proposed in order to escape from suboptimal points and obtains the best possibility solutions. There are several modifications in the MACO compared to ACO including candidate list, global pheromone updating rules and several local search techniques. Using the candidate lists leads to the concentration of the search on promising nodes in order to reduce the computational time. Moreover, these modifications help to avoid from regions with worse quality and then to search over the subspace in order to find the global optimum. To improve the mentioned shortcomings, the MACO has made some changes to the algorithm as follows:

1: Build n solution of the TSP parallel by using a new transition rule and a candidate list.

2: Use insert, swap and 2-opt moves for the best known solution until now in order to improve them more.

3: The algorithm uses only global pheromone release in order to avoid the edges of the worst solution in each iteration.

4: A minimum and maximum values of pheromone are considered in order to prevent fall in the local points.

In the classic ACO, every problem is divided into some sub-problems in which the simulated ants are expected to select the next node based on the amount of the

pheromone in a trail and the distance to the next node. Therefore, at the first stage of the NACO, *n* ants are initially positioned on the randomly node and each ant visits a node by using a candidate list and formula (1). This formula is a new transition rule which takes either exploitation or exploration into account was introduced. This step is iterated for *n* times until each solution is built by each ant. Therefore, *n* feasible solutions for the TSP are partially obtained in this step. In the proposed algorithm, the next node j from node i in the route is selected by ant k, among the unvisited candidate nodes O_i^k with *n** number, according to the following transition rule which shows the probability of each city being visited. In other words, the ant k at point *i* maintains a tabu list N_i^k in memory which defines the set of points still to be visited when it is at this point.

$$P_{ij}^k(t)=\begin{cases}1 & if\{q\le q_0 \;\&\; j=j^*\}\\ 0 & if\{q\le q_0 \;\&\; j\neq j^*\}\\ \dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)\kappa_{ij}^{\lambda}(t)}{\sum_{r\in J_i^k}\tau_{ir}^{\alpha}(t)\eta_{ir}^{\beta}(t)\kappa_{ij}^{\lambda}(t)} & Otherwise\end{cases} \tag{1}$$

Where:

τ_{ij} : The amount of pheromone on (i, j) edge

η_{ij} : The inverse distance between i and j. However,

κ_{ij} : The savings of combining two nodes on one tour as opposed to serving them on two different tours.

α , β and λ :Both powers are non-negative which can be changed by the user.

j^*: The node which has the highest value of $([\tau_{ir}(t)]^{\alpha}[\eta_{ir}(t)]^{\beta}[\kappa_{ij}(t)]^{\lambda})$ among all unvisited nodes in j_i^k

q : A random parameter which has been distributed evenly in [0, 1].

q_0: A constant threshold parameter in [0, 1] which determines the relative importance of extraction to discovery. It should be noted that selecting $q_0=0$ changes ACS into AS.

Therefore, when q is less than or equal to q_0, ants utilize discovery in order to select j^* as the next node in their tour. Whereas, if q is greater than q_0, the ant uses exploration based on probability in order to select the next node.

A candidate list is used to improve the quality of the solutions and computational time in the optimization problems. For example, it is used to determine the next customer which is not visited until now and is selected in a vehicle route based on the distance to all other customers in the customer set. In the proposed algorithm, the closest unvisited nodes are only considered in the candidate list for the current node *i* and are made available for selection as the next node to be visited in the route. It should be noted that the size of the candidate list has been determined by restricting its size to a fraction of the total number of nodes in the problem. It is noted that this restriction prevents the algorithm from wasting its efforts to consider moves to nodes which are a great distance from the current node and have very little chance of creating an improved solution to the problem.

In the proposed algorithm, the candidate list size is set equal to 25% of the total number of nodes. For example, in problems with 60 nodes the candidate list is limited to the rounded integer value of fifteen and in problems with 200 nodes which are common in different versions of the TSP; candidate lists might include as many as fifty nodes which is still a large number. As a result, in order to decrease the computational time and increase the probability of obtaining a higher-quality solution, upper limit 10 is fixed to the number of candidate list n*. If the size of this list is more than the maximum 10, then 10 is replaced by 0.25n* using formula (2).

$$n^* = \begin{cases} 10 & 0.25n > 10 \\ 0.25n & otherwise \end{cases} \tag{2}$$

The proposed MACO algorithm ranks the solutions constructed by ants. What distinguishes this algorithm from the other algorithms is the fact that in MACO the amount of released pheromone is based on the rank of the ants in finding solutions. In other words, after ants complete their tours, the edges of the best tours are updated by the formula (3):

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \rho(1/C_b) \tag{3}$$

Where $0 \le \rho \le 1$ is pheromone density reduced in each iteration (set by the user), C_b is the cost of the best T_b tour found from the start of the algorithm. It should be noted that global trail updating adjusts only the pheromone on the edges of the best tour, which encourages the ants to search the neighbors of the best tour in the following iterations.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{\mu=1}^{\sigma-1} \Delta\tau_{ij}^{\mu}(t) + \Delta\tau_{ij}^{gb}(t) \tag{4}$$

Where:

$$\Delta\tau_{ij}^{\mu}(t) = \begin{cases} (\sigma-\mu).\dfrac{Q}{L^{\mu}(t)} & (i,j) \in S^{\mu} \\ 0 & (i,j) \notin S^{k} \end{cases} \tag{5}$$

μ : The variable indicating ranking index.

S^{μ} : The edges traversed by an ant which has gained the μ th rank in finding the best solution.

σ : The number of ants which have been ranked and on their edges the pheromone has been deposited.

Q: A constant coefficient determined by the user.

$L^{\mu}(t)$: The length of the path traversed by the μ th ant.

The amount of the global pheromone release for the best ant is calculated by the formula (6):

$$\Delta\tau_{ij}^{gb}(t) = \begin{cases} \sigma.\dfrac{Q}{L^{gb}(t)} & (i,j) \in best\,route \\ 0 & (i,j) \notin best\,route \end{cases} \tag{6}$$

In order to prevent fall in the local points, a minimum and a maximum values which are equal to the half of the initial pheromone and 10 times more than the initial pheromone are considered for the edges respectively. When the amount of pheromone

on the edges falls below, this value is replaced with the minimum value. Furthermore, the maximum is considered when the amount of the pheromone violated upper constraint.

In AS all ants construct similar tours after some time. If the constructed tour is not good, the algorithm loses its ability in finding better solutions and the search and extraction process is halted as a result. Therefore, the algorithm loses its efficiency in solving large size problems. So, after constructing all solution in the algorithm, the ant which has been able to find the best known solution until now is encouraged in each iteration, which improves the effectiveness of the algorithm. The idea here is that better solution may have better chance to find a global optimum. In other words, a promising approach to obtaining high-quality solutions is used to couple a local search algorithm with a mechanism to generate initial solutions. A local search approach starts with an initial solution and searches within neighborhoods for better solutions. In the proposed algorithm, after the ants have constructed their solutions, each ant's solution is improved by applying a local search. However, local search is a time-consuming procedure. To save the computation time, we will only apply local search algorithms including insert, swap and 2-opt algorithms to the best known solution. We first apply a local search based on insert move to the ant, and then apply the swap move. In insert algorithm a node is moved. However, in swap algorithm a node is swapped with another node. The 2-opt local search is a powerful global search algorithm used after insert and swap algorithms. In this algorithm, two arcs are deleted and reconnect them by two new arcs. It also should be noted that the new solution will be only accepted in state that, novel tour will gain better value for problem than previous solution. These operators are shown in figure 2.

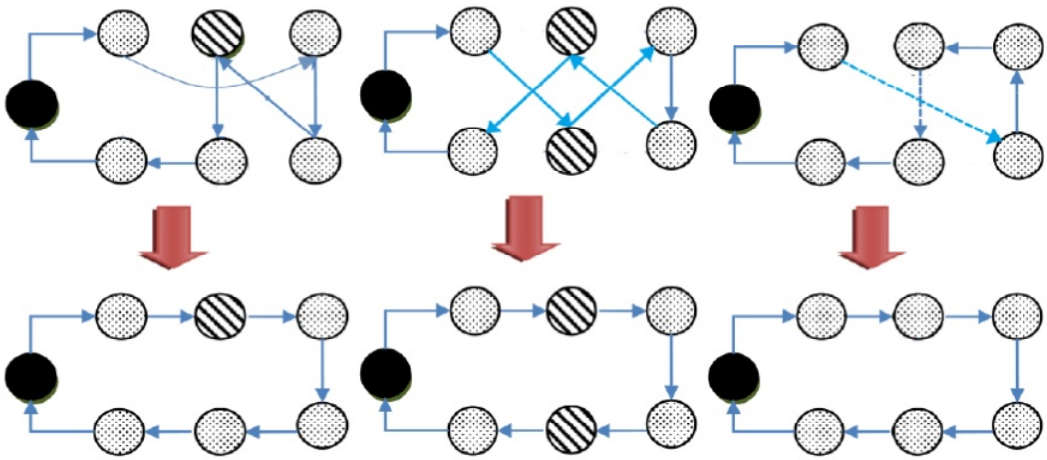

Figure 2. The local search algorithms for applying in the TSP

At this stage, the stop condition is checked. If this condition is met, the algorithm ends. Otherwise, if stop conditions do not satisfied, the algorithm is iterated by returning to transition rule step. To end the loop, if the best solution till now does not improve within a given ten generations in MACO, the algorithm will be stopped. This condition is checked at the end iteration of algorithm. If the condition is met, the algorithm ends and the obtained results and values up to now are considered as the best values and results of the algorithm. A pseudo-code of the MACO for the TSP is presented in the Figure 3.

```
procedure MACO for solving TSP
    s* = ϕ ;                          // s* is the best solution found yet //
    f* := ∞;                          // f* is value of the s* //
    n= the number of nodes;   // n is the number of ants //
    initialize pheromone trails as u;
     while do // main cycle //
    begin
        s** = ϕ ; f** := ∞;     // s** is the best solution found in current iteration
and f** is the     value of the s** //
        s*** = ϕ ; f*** := −∞;   // s*** is the worst solution found in current iteration
and f*** is the value of the s** //
            S:= none;                    // S is a matrix and population of solutions //
            for i := 1 to n do
            begin
                Construct a feasible solution si by using formula (1) based on
             candidate list and gain f( si ) as value of the si
                S=S∪ si ;
                If f( si )< f**
                begin
                        f** := f(si ) ; s** = si ;
                End
            end;
            Apply insert, swap and 2-opt local search algorithms on s** and gain f**
         again.
                If f( s** )< f*
                begin
                        f* := f(s** ) ; s* = s** ;
                End
                Update pheromone trails and apply constraints of pheromone on
             arcs
    Until the best known solutions are not changed for 10 times.
    show s* and f*
end // procedure //
```

Figure 3. The MACO for the TSP

3. Results

The presented algorithm was coded in Matlab 7 and executed on a Pentium IV 3.00 GHz machine with 1 Gb of RAM running Windows XP Home Basic Operating system. At first of this section, the best, worst and mean of the results of MACO are presented in table 1. This table contains 14 standard TSPs which possess an acceptable number of

nodes whose sizes are between 48 and 200. For more information regarding the provided examples visit. Then, in order to assess the effectiveness of the MACO, its results and some meta-heuristic algorithms have been compared in table 2.

http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

In table 1, N, Tbest, Worst, Mean, and Best are number of the nodes of each example, the number of independent algorithm implementation to find the best value, the worst solutions found, the mean solutions found and the best solutions by the proposed MACO algorithm respectively. Finally, the last column shows the best solutions ever found in literature.

As the results indicated in table 1, the proposed algorithm has not been able to find the best solutions in nine of the fourteen examples. Therefore, this algorithm has high quality solutions in order to escape from local optimum points and find the global optimum solution. Furthermore in three instances including ATT48, Eli51 and Berlin52, the algorithm can hold quality in each iteration and the worst, mean and best solutions are equal. In other instances, although these are not obtained equal together, the mean is so closer to the best solution than the worst solutions. It means that, most of the solutions are obtained close to the best solutions in ten iterations for the proposed MACO.

Table. 1. Instance properties and results of the PA

Number	*Instance*	*n*	*Tbest*	*Worst*	*Mean*	*Best*	*Best*
1	GR48	*48*	10	*5053*	*5053*	5053	***5046***
2	ATT48	*48*	10	***10628***	***10628***	**10628**	***10628***
3	Eil51	*51*	10	***426***	***426***	**426**	***426***
4	Berlin52	*52*	10	***7542***	***7542***	**7542**	***7542***
5	ST70	*70*	10	*689*	*681*	**675**	***675***
6	Eil76	*76*	10	*545*	*540*	**538**	***538***
7	KroA100	*100*	10	*21456*	*21312*	**21282**	***21282***
8	KroB100	*100*	10	*22304*	*22191*	**22141**	***22141***
9	Eil101	*101*	10	*651*	*638*	630	***629***
10	Lin105	*105*	10	*14703*	*14542*	**14379**	***14379***
11	KroA150	*150*	10	*26921*	*26721*	26535	***26524***
12	KroB150	*150*	10	*26439*	*26231*	**26130**	***26130***
13	KroA200	*200*	10	*30098*	*29743*	29375	***29368***
14	KroB200	*200*	10	*30672*	*29998*	*29761*	***29437***

In addition, in order to demonstrate the efficiency of the MACO, some of the solutions found for the examples in table 1 are presented in Figure 4. It should be noted that in four examples presented this figure, the MACO has been able to find the best solution ever found.

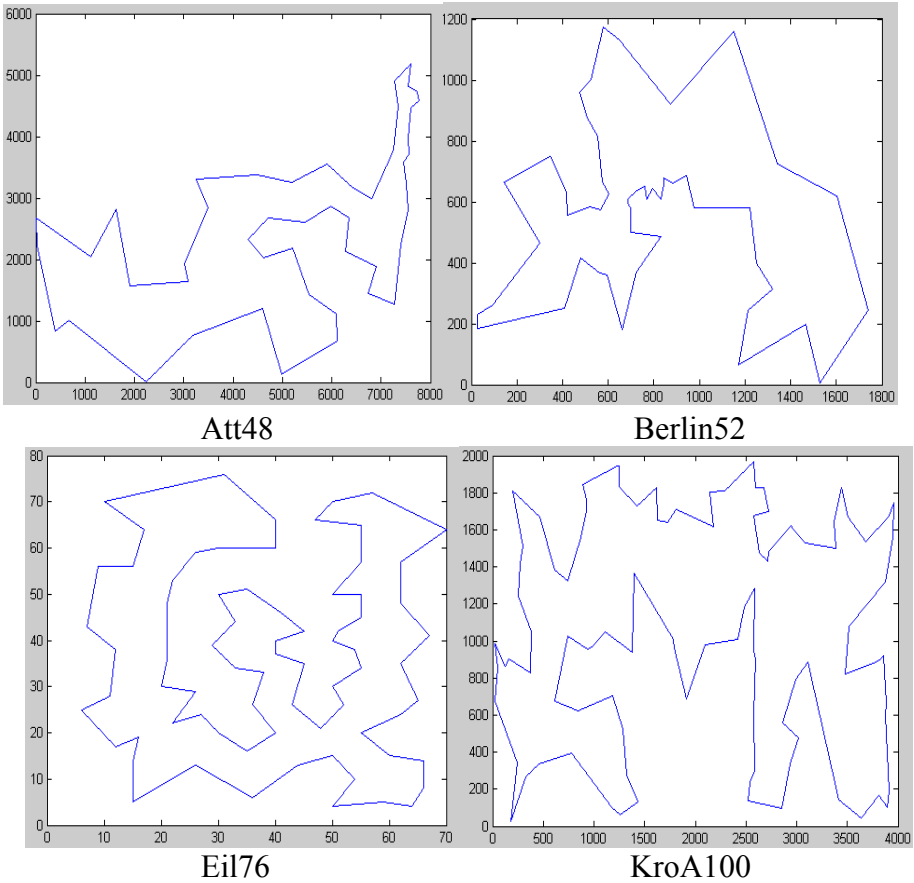

Figure 4. Some of the Solutions to the TSP Found by PA

In order to compare the MACO and classic ACO, the Eil51 has been used. Figure 5 presents the solutions found by both algorithms during their implementations. In this figure the horizontal axis represents the number of algorithm iterations as a condition for ending the algorithm. The vertical axis represents the best solution found during algorithm implementation. As this figure indicates, despite the fact that ACO has an acceptable efficiency at the start and can achieve 430 which is very close to the best value of 426 in 24 iterations, it fails to improve the value from that point on and cannot find better solutions. It can be concluded that the accumulation of pheromone on certain edges in this algorithm can prevent it from searching feasible space after some time. So, the search is conducted only in a part of this space and premature convergence occurs as a result.

Premature convergence encourages the ACO to consider the local optimum point with the value of 430 as the global optimum point and this algorithm fails to find other solutions due to pheromone increase on this route in each iteration. In contrast to ACO, pheromone is deposited on the best route in each iteration in MACO and therefore over accumulation of the pheromone on certain edges is avoided. This claim can be proved through Figure 5. As the figure shows, MACO has not only been able to improve the solutions rapidly in each iteration but it has also been able to escape the local optimum points and converge to the best value of 426 in almost 18 iterations. The reduction in pheromone release on less satisfactory routes and making use of exponential function for depositing pheromone have assisted the algorithm in converging to the best solutions, which helps to find the best solution in a satisfactory time.

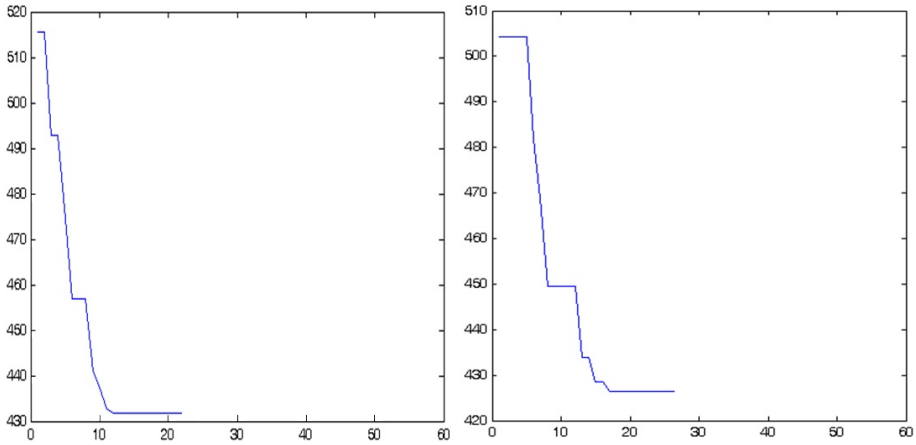

Figure 5. Comparison between MACO (right) and EAS (left) in Eil51 example

In table 2, the best solutions found by the MACO for the 14 examples are compared with seven meta-heuristic algorithms. In this table, columns 2 and 3 contain results of the algorithm of bee colony algorithm (BCO) [26] and particle swarm optimization (PSO) [27]. The algorithms belonging to versions of the ACO including AS and EAS are shown in columns 4 and 5. The imperialist competitive algorithm [28], BCO with local search [29] and genetic algorithm combined by simulated annealing, ACS and PSO (GSAP) [30] are shown in columns 6, 7 and 8. Finally column 9 contains the solutions of the MACO. In order to make better comparisons between algorithms, the best known values for each problem found by other algorithms (Best) have been presented in column 10.

From the comparison between PSO and the proposed algorithm, it can be seen that PSO in five examples has been able to find solutions with gap of less than 1 percent. Despite being able to find the best solution ever found for Berlin52, it has failed to achieve this result in the remaining four examples. However, MACO has found equal solution in Berlin52 and better solutions for the other four examples including Eli51, Eli76, KroA100 and KroA200. It can be concluded that MACO is able to produce better solutions in comparison with PSO and has been able to escape local optimum points. Furthermore, the obtained results by BCO for 11 instances are shown in table 2. The comparison between BCO and MACO shows that the BCO has produced better solution only in Eil101, while MACO has generated better solutions in the remaining 10 examples. It should be noted that BCO has had a weak performance in general and has not been able to produce the best solutions in most of the examples.

In order to test efficiency of the proposed MACO and to provide more convergence, results of the classic ACO and EAS were presented here. Compared with EAS, MACO has been able to find better solutions in 13 out of 14 examples. As it was mentioned, in nine examples the MACO has been able to find the best solutions. It can be seen that EAS and MACO have similar solutions in only one example and in the rest examples MACO has found better solutions than EAS. Furthermore, the results of the EAS are much better than the ACO in most of the instances and only for three instances ST70, Eli76 and KroA200 the ACO can obtain better results than the EAS. Therefore, in comparing the MACO with ACO and EAS, it can be concluded that the MACO has been capable of finding better solutions and making a satisfactory improvement in the performance of the algorithm. In more details, the MACO has found the best solutions

in 9 examples and has obtained solutions with gap of less than 1 percent in the remaining 6 examples among the 14 examples. The results for the ACO and EAS show that, these two algorithms are the weakest algorithms because they have not able to converge to the best solutions in any of the 14 examples.

Results of the two new algorithms for the TSP are ICA and ABS shown in the Table 2. Although the ICA can not obtain the best know solution for none of the instances, this algorithm has better solution than ABS in common instances except in Eli51. In more details in Eli51, quality of both algorithms are same and they can not obtain the best known solution. So, in compared to these algorithms, the MACO has produced better solutions. One of the most powerful algorithms for solving the TSP in table 2 is GSAP. This algorithm can gain high quality solutions and six best solutions are obtained by this algorithm. The results of comparison between this algorithm and MACO shows that the proposed algorithm gains equal solutions with the GSAP in Eli76, KroA100, KroB100, Eli101, Lin105 and KroB150, which are not very large scale problems, and it gains better solutions than the GSAP in following scale problems such as KroA200 and KroB200. Therefore, the Computational experiments also show that in general the proposed algorithm gives better results compared to GSAP algorithm in terms of the solution's quality for each instance.

Table. 2. Comparing algorithms for standard TSP problems

Instance	BCO	PSO	AS	EAS	ICA	ABC	GSAP	MACO	Best
GR48	-	-	5075	5053	-	-	-	5053	5046
ATT48	10661	-	10701	**10628**	-	-	-	**10628**	10628
Eil51	428	427	430	429	432	432	427	**426**	426
Berlin52	-	**7542**	7591	7576	7549	-	-	**7542**	7542
ST70	-	-	684	685	678	-	-	**675**	675
Eil76	539	540	545	548	543	561	**538**	**538**	538
KroA100	21763	21296	21456	21311	21303	21688	**21282**	**21282**	21282
KroB100	22637	-	22304	22520	22225	-	**22141**	**22141**	22141
Eil101	635	-	651	642	-	657	630	630	629
Lin105	15288	-	14703	14489	-	-	**14379**	**14379**	14379
KroA150	27858	-	26821	26711	-	-	**26524**	26535	26524
KroB150	26535	-	26439	26342	-	-	**26130**	**26130**	26130
KroA200	29961	29563	30098	29767	-	-	29383	29375	29368
KroB200	30350	-	30672	29938	-	-	29541	29499	29437

To measure the efficiency and the quality of an algorithm, a simple criterion is to compute the number of optimal solutions found in specific benchmark instances by algorithm. As it can be seen from Table 2, the proposed algorithm HACO finds the optimal solution for 9 out of 14 problem instances in a reasonable time and these solutions have been published in the literature. Moreover, AS, ABS, ICA, BCO, PSO, EAS and GSAP have been able to find 0, 0, 0, 0, 1, 1 and 6 optimal solutions from among these instances. These results indicate that HACO is a competitive approach compared to mentioned algorithms and the results are much better than the ones found by these algorithms. As a result, the algorithms in terms of their performance from the worst to the best can be listed as: AS, ABS, ICA, BCO, PSO, EAS, GSAP and MACO.

As it is shown in [31], direct comparisons of the required computational times cannot be conducted as they closely depend on various factors such as the processing power of

the computers, the programming languages, the coding abilities of the programmers, the compilers and the running processes on the computers.

4. Conclusion and Future Works

The aim of this study is to describe a modified meta-heuristic algorithm for solving the TSP, called MACO. The main advantage of this algorithm over the majority of other meta-heuristics is that it produces quite satisfactory solutions in reasonable amount of time by employing several effective modifications. The modifications improved the performance of the algorithm in escaping from local optimum points and finding better solutions in comparison with the classic version of the algorithm. It seems that combining the MACO with other meta-heuristic algorithms like particle swarm optimization or tabu search and making use of strong local algorithms like Lin-Kernigan algorithm can bring about better results. Furthermore, MACO can be used for other combinatorial optimization problems like vehicle routing problem and the capacitated clustering problem. Future projects will focus on working on such ideas and making them operational.

References

[1] M. Yousefikhoshbakht, F, Didehvar, F, and Rahmati, "Modification of the ant colony optimization for solving the multiple traveling salesman problem," Romanian academy section for information science and technology, 2013.

[2] M. Yousefikhoshbakht, F, Didehvar, F, and Rahmati, "Solving the heterogeneous fixed fleet open vehicle routing problem by a combined metaheuristic algorithm," International Journal of Production Research, 2014.

[3] M. Barketau, and E. Pesch, "An approximation algorithm for a special case of the asymmetric travelling salesman problem," International Journal of Production Research, 2015.

[4] J. Yang, R. Ding, Y. Zhang, M. Cong, F. Wang, and G. Tang, "An improved ant colony optimization (I-ACO) method for the quasi travelling salesman problem (Quasi-TSP), "International Journal of Geographical Information Science, 2015.

[5] B. Gavish, and K. Srikanth, "An optimal solution method for large-scale multiple traveling salesman problems," Operations Research,1986.

[6] M. Battarra, A, Pessoa, A. A, Subramanian, and E. Uchoa, "Exact algorithms for the traveling salesman problem with draft limits," European Journal of Operational Research, 2014.

[7] M. Xiao, and j. Nagamochi, "An Improved Exact Algorithm for TSP in Graphs of Maximum Degree 4," Theory of Computing Systems, 2016.

[8] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances," European Journal of Operational Research, 2011.

[9] M. Albayrak, and N. Allahverdi, "Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms," Expert Systems with Applications, 2011.

[10] J. B. Odili, and M. N. Mohmad Kahar, "Solving the Traveling Salesman's Problem Using the African Buffalo Optimization," Computational Intelligence and Neuroscience, 2016.

[11] M, Tuba, and R. Jovanovic, "Improved ACO Algorithm with Pheromone Correction Strategy for the Traveling Salesman Problem," International Journal of Computers, Communications & Control, 2013.

[12] Y. Marinakis, M. Marinaki, and G. Dounias, "Honey bees mating optimization algorithm for the Euclidean traveling salesman problem," Information Sciences, 2011.

[13] Y. T. Wang, J. Q. Li, K. Z. Gao, and Q. K. Pan, "Memetic Algorithm based on Improved Inver–over operator and Lin–Kernighan local search for the Euclidean traveling salesman problem," Computers & Mathematics with Applications, 2011.

[14] M. S. Tarkov, "Solving the traveling salesman problem using a recurrent neural network, Numerical Analysis and Applications, 2015.

[15] M. Anantathanavit, and M. unlin, "Using K-means Radius Particle Swarm Optimization for the Travelling Salesman Problem," IETE Technical Review, 2015.

[16] S. M. Chen, and C. Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," Expert Systems with Applications, 2011.

[17] S. M. Chen, and C. Y. Chien, "Parallelized genetic ant colony systems for solving the traveling salesman problem," Expert Systems with Applications, 2011..

[18] M. Dorigo, "Optimization, Learning and natural algorithms," Ph.D Thesis, Dip.Electtronica e Informazion, Politecnico di Milano Italy, 1992.

[19] M. Yousefikhoshbakht, F. Didehvar, and F. Rahmati, "Modification of the ant colony optimization for solving the multiple traveling salesman problem," Romanian academy section for information science and technology, 2013.

[20] M. Yousefikhoshbakht, F. Didehvar, and F. Rahmati, "An Efficient Solution for the Vehicle Routing Problem by Using a Hybrid Elite Ant Colony Optimization," International Journal of Computers, Communications and Control, 2014.

[21] Z. Zhang, and Z. Feng, "Two-stage updating pheromone for invariant ant colony optimization algorithm," Expert Systems with Applications, 2012.

[22] Z. Jing, and T. Wei-ming, "Solution to the problem of ant being stuck by ant colony routing algorithm," The Journal of China Universities of Posts and Telecommunications, 2009.

[23] B. Bullnheimer, R. F. Hartl, and C. Strauss, "A new rank based version of the ant system – A computational study," Central European Journal for Operations Research and Economics, 1997.

[24] S. Yadlapalli, W. A. Malik, S. Darbhaa, and M. Pachter, M. "A Lagrangian-based algorithm for a Multiple Depot, Multiple Traveling Salesmen Problem, Nonlinear Analysis: Real World Applications, 2009.

[25] V. Mak, and N. Boland, "Polyhedral results and exact algorithms for the asymmetric travelling salesman problem with replenishment arcs," Discrete Applied Mathematics, 2007.

[26] L. P. Wong, M. Y. H. Low, and C. S. Chong, "A bee colony optimization algorithm for traveling salesman problem. Modeling and Simulation," AICMS 08. Second Asia International Conference, 2008.

[27] W. L. Zhong, J. Zhang, and W. N. Chen, "A novel discrete particle swarm optimization to solve traveling salesman problem," InEvolutionary Computation, CEC 2007. IEEE Congress on (pp. 3283-3287).

[28] K. Nemati, S. M. Shamsuddin, and M. Saberi Kamarposhti, "Using imperial competitive algorithm for solving traveling salesman problem and comparing the efficiency of the proposed algorithm with methods in use," Australian Journal of Basic and Application, 2011..

[29] H. E. Kocer, and M. R. Akca, "An improved artificial bee colony algorithm with local search for traveling salesman problem," Cybernetics and Systems, 2014.

[30] S. M. Chen, C. Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," Expert Systems with Applications, 2011.

[31] C. D. Tarantilis, E. Zachariadis, and C. Kiranoudis, “A guided Tabu search for the heterogeneous vehicle routing problem,” Journal of the Operational Research Society, 2008.