International Journal of Industrial Engineering & Production Research March 2024 Vol. 35, No. 1: 1-13
DOI: 10.22068/ijiepr.35.1.1843

RESEARCH PAPER

A New Heuristic Algorithm Based on Minimum Spanning Tree for Solving Metric Traveling Salesman Problem

Malihe Masoumi[1] & Javad Behnamian[2*]

Received 26 June 2023; Revised 9 December 2023; Accepted 7 January 2024;

ABSTRACT

Due to the many applications of the travelling salesman problem, solving this problem has been considered by many researchers. One of the variants of the travelling salesman problem is the metric travelling salesman problem, where the triangle inequality holds. This is a crucial problem in combinatorial optimization as it is used as a benchmark for proving complexity or providing solutions to other problems in this class. The solution is used usually in logistics, manufacturing and other areas for cost minimization. Since this is an NP-hard problem, heuristic and meta-heuristic algorithms seek near-optimal solutions in polynomial time. For this purpose, in this paper, a heuristic algorithm based on the minimum spanning tree is presented to solve this problem. Then, by generating 20 instances, the efficiency of the proposed algorithm was compared with one of the most famous algorithms for solving the travelling salesman problem, namely the nearest neighbour algorithm and the ant colony optimization algorithm. The results show that the proposed algorithm has good convergence to the optimal solution. In general, the proposed algorithm has a balance between runtime and the solution found compared to the other two algorithms, so that the nearest neighbour algorithm has a very good runtime to reach the solution but does not have the necessary convergence to the optimal solution, and vice versa, the ant colony algorithm converges very well to the optimal solution, but, its runtime is very longer than the proposed algorithm.

KEYWORDS: *Travelling salesman problem; Metric TSP; Heuristic; Minimum spanning tree.*

1. Introduction

The travelling salesman problem (TSP) is one of the most popular combinatorial optimization problems. In this case, an attempt is made to obtain the shortest route for a salesman so that the path starts from one city and after passing through other cities, finally returns to the city of origin; provided that it passes through each of the cities only once. The distance traveled by this traveling salesman depends on the order of cities visited. Therefore, the objective function of this problem is to find the optimal route to visit the cities, so that the intended cost objective function is optimized. Although understanding this does not require complex mathematical skills, it is difficult to optimize. This problem was first raised in 1759 by Ulro under the name "Knight Path". The "Knight's path" was a path in a graph whose vertices were 64 square chessboards, each adjacent to two other vertices, and the Knight could only move to one adjacent square at a time. In the eighteenth century, this problem was studied as a subset of graph theory by the famous Irish mathematician William Hamilton; which was later named "Hamilton route" in his honour. The term "travelling salesman" was first used in a German book [1].

Menger [2] tried to solve this problem by presenting an algorithm similar to the nearest neighbour but could not reach an optimal solution. The main studies on TSP as a combinatorial optimization problem were started by Dantzig et al. [3]. Marinakis et al. [4] developed various neighbourhood search methods for solving TSP, using heuristic and probabilistic methods. By presenting a greedy algorithm, they solved the travelling salesman problem in a situation where only a subset of cities could be visited. Although

* Corresponding author: Javad Behnamian
Behnamian@basu.ac.ir

1. *Department of Industrial Engineering, Bu-Ali Sina University, Hamedan, Iran.*
2. *Department of Industrial Engineering, Bu-Ali Sina University, Hamedan, Iran.*

this algorithm could provide the final solution on a subset cities at a polynomial time, the quality of the final solutions was relatively low. This was especially the case when the number of cities increased. Goldberg [5], by conducting studies on evolutionary algorithms, introduced them as one of the most efficient optimization algorithms. In the case of TSP, Johnson et al. [6] conducted a significant study on this problem using different algorithms.

In general, TSP is one of the problems that has been studied to solve using various evolutionary algorithms [7]. In this regard, various genetic algorithms have been proposed to solve TSP. Nyder and Daskin [8] used a combination of genetic algorithm and probabilistic method to solve the travelling salesman problem. The genetic nature of this method in combination with probabilistic methods, also led to some issues. Due to the weakness of genetic operators, appropriate arrangements were not made to get rid of the local optimum. Although the algorithm sometimes provided optimal solutions, the number of premature convergences was high. On the other hand, it was observed in some research that the combination of the evolutionary algorithms with heuristic local search methods can improve the obtained results [8]. For example, Johnson et al. [6] concluded that to solve difficult combinatorial problems such as TSP using evolutionary algorithms, these algorithms must be integrated with local search methods. Such methods are commonly referred to as "local genetic search" algorithms. Brady [9], first, solved TSP using hybrid evolutionary methods. Then, the author used the 2-opt method as a local search. Kumar and Singh [10] solved the travelling salesman problem with a multi-objective evolutionary algorithm. They tried to converge the search process by presenting a new method in the selection stage in the proposed genetic algorithm. Their evaluations show that this method worked well in providing near-optimal solutions, but had two drawbacks. First, the runtime increased several times as the number of cities increased, and second, the optimality of the solutions decreased significantly as the number of cities increased.

The aim of the current study is to solve the travelling salesman problem in a way that can solve this combinatorial problem in a short runtime and with a good solution. Therefore, by presenting a heuristic algorithm to solve this routing problem, its ability has been compared with the nearest neighbour (NN) algorithm proposed by Hougardy and Wilde [11] and the ant colony optimization algorithm (ACO) proposed by Dorigo and Gambardella [12].

The remainder of this paper is organized as follows: after the literature review in the third section, the definition of the metric travelling salesman problem and its mathematical model are presented. Section 4 describes how to implement and solve the TSP problem using the proposed algorithm. In Section 5, the computational results are presented and finally, Section 6 is devoted to the conclusions and recommendations of the research.

2. Literature Review

The deterministic algorithms compute the same output for the same inputs, and these algorithms follow the same iterative step to a solution. But on the opposite side, the stochastic algorithms have a stochastic module that provides different solutions in different runs for the same inputs. Due to this feature, stochastic algorithms have become an essential tool in a wide range of optimization problems. Stochastic algorithms are also divided into two categories: heuristic algorithms and meta-heuristic algorithms. Heuristic algorithms refer to the process of trial and error in which there is no guarantee of finding the optimal solution. However, most of the solutions found by these algorithms are good and appropriate [13]. Heuristic algorithms are problem-dependent methods that use a systematic procedure to obtain relatively good solutions [14]. An example of a heuristic algorithm is the nearest neighbour (NN) algorithm. Meta-heuristic algorithms are a further development of heuristic algorithms that are mainly combined with artificial intelligence, local search, and randomization methods. Meta-heuristic algorithms inspired by natural processes such as chemistry, physics, and biology, include two main concepts of intensification and diversification in the search process.

Since the 1930s, many attempts have been made to solve the travelling salesman problem, using both deterministic and stochastic approaches. In this regard, Halim and Ismail [15] have classified the optimization algorithms for the TSP problem. Deterministic algorithms have drawbacks due to the lack of randomized concepts compared to stochastic algorithms [16], and their runtime increases exponentially by increasing the number of cities [17]. Therefore, heuristic methods are used more due to their strategies and less runtime. In general, for Np-hard problems, a heuristic search is more practical and appropriate [18].

As mentioned, the travelling salesman problem (TSP) is a well-known combinatorial optimization

problem. Although the mathematical model is straightforward to understand, it is complicated to solve this problem effectively, due to its NP-hardness [19]. Therefore, an algorithm that solves this problem in polynomial time is not yet provided. The main problem in TSP is to find a Hamiltonian route with the minimum cost in the graph. A variety of exact and heuristic methods have been proposed to solve the TSP problem [20]. From another perspective, there are two main groups of methods for solving optimization problems using permutation codes, which include methods based on path improvement and methods based on path construction [21]. The path improvement methods generate new solutions through crossover, mutation, switching, and so on. Path improvement methods usually do not use problem information and take less runtime to generate alternative solutions, but require a large number of iterations to achieve a quality solution. The path improvement algorithms include:

Local optimal method: The objective of these algorithms is to delete a set of edges and insert other edges. The simplest heuristic method for improving the tour that has been used in practice so far is the 2-OPT heuristic in which the two edges are switched. This algorithm has been introduced in the papers of Buck [22], Croes [23] and Flood [24]. Another algorithm has been introduced by Lane and Kernighan (LK), which is based on the method of finding the tour by *k*-OPT [25]. Currently, the most successful algorithm among the local optimal methods is Helsgaun's Modification Of LK (LKH) [26].

Simulated annealing algorithm: Simulation annealing algorithms mimic the physical annealing process of metals. These methods suggest periodic increases during the tour to avoid falling into the local optimum. Detailed explanations of this algorithm are provided by Laarhoven et al. [27].

*The evolutionary algorithm***:** These methods are inspired by the process of biological evolution, which includes the following: reproduction, mutation, crossover, and selection. More information can be found in reference [6].

*Collective intelligence method***:** These are nature-inspired meta-heuristic algorithms. These methods are based on decentralized and self-organized systems with simple factors such as ants, bees, birds, fish, and so on. The most popular algorithms the include artificial bee colony algorithm (ABC) [28], ant colony optimization (ACO) [12], particle swarm optimization (PSO) [29] and cuckoo search (CS) [30]. Path construction methods usually use the information in question, so they spend more time creating new solutions. In these methods, the tour is created by adding a new vertex at each step.

*Ordered sequence methods***:** The heuristic algorithm in these methods first ranks all the edges by appropriate criteria. Then the best edge is added to the solution in each iteration. The most common ordered sequence methods are greedy algorithms [34] and saving algorithms [35, 36].

*Increasing path methods***:** In these methods, a vertex or an edge is selected as the starting point of a path. In each iteration, an edge is selected according to appropriate criteria and added to the end of the path. The disadvantage of this method is that when the heuristic algorithm ends, the end of a path may be too far from the last edge, meaning that the last edge needed to turn the path into a tour is very expensive. There are two groups of increasing path methods: (a) neighbourhood selection [31], [32] and (b) space-filling curves [33].

Sub-tour insertion method: In these methods, an initial sub-tour is selected, for example, an edge. In each iteration, a vertex is selected based on certain criteria and added below the sub-tour. In order to insert a new vertex, some edges under the sub-tour are replaced with two other edges that are not under the sub-tour. The insertion heuristic algorithm is described in Rosenkrantz [31], and Johnson & McGeoch [6].

Hybrid methods: The most well-known method in this paper is the minimum spanning tree method proposed by Christofides in 1970 [34], [35]. In general, algorithms based on the minimum spanning tree to solve the travelling salesman problem include three main steps: (*i*) Find the minimum spanning tree (MST) for the graph,(*ii*) Create a graph containing an Eulerian path by removing and adding the edges of the MST, and (*iii*) Create a Hamilton tour from the Eulerian tour. To solve the metric TSP problem, there are two well-known algorithms, MST and Christofides (CH), with time complexity $O(n^2)$ and $\underline{O(n^3)}$, respectively, which are based on the above steps. The main idea of the Christofides algorithm is to prevent doubling the edges to go from MST to the Eulerian graph in the second step of the MST algorithm. In this algorithm, instead of all vertices, it only increases the set of edges, that is, each vertex in MST, which has an odd degree, is paired by adding an edge. However, the MST algorithm is more desirable despite the lower efficiency due to the linearity of the solution runtime. The CH algorithm is less desirable despite the better efficiency ratio because it finds the minimum weighted matching in $O(n^3)$ [43]. In 2014, Gang and Lee [36] compared four heuristic methods to

solve the travelling salesman problem: they compared the nearest neighbour algorithm, stochastic insertion, minimum spanning tree, and Christofides algorithm.

Meng and Wang [37] proposed a hybrid algorithm based on ant colony optimization to solve the TSP problem. They used the genetic algorithm to find the solution space to deal with the problem of the initial downturn in the travelling salesman problem. Also, they presented a new MST strategy with the nearest neighbour to generate the initial tour to improve TSP. Their simulation results show that the proposed new algorithm provides a significant improvement in finding global optimal solutions in large-size TSP. Xu and Rodrigues [38] proposed a deterministic algorithm based on the Christofides algorithm with an approximation of $(\sqrt{5/2}+1)$ to solve the TSP for an arbitrary metric. Given the symmetric metric cost at n vertices containing two predetermined vertices, They wanted to find the shortest Hamiltonian cycle between the two vertices, and the authors stated that the normal version of the Christofides algorithm, was, in fact, the best-known approximation method, but they modified the algorithm in some way. Instead of the minimum spanning tree, a primary spanning tree should be selected based on the optimal result obtained by Held-Karp relaxation and proved that this modification leads to a significant improvement in the performance of the Christofides algorithm. Xu and Rodriguez in 2017 [38] studied a travelling salesman problem in which several salesmen are located in several different warehouses. In this case, only a limited number (k) of these salesmen can be selected to serve customers. The authors state that there are only two approximate algorithms for solving the TSP problem with multiple salesmen and multiple warehouses. They improved the performance of these algorithms by developing the well-known Christofides algorithm, and in doing so, developed a framework that could be used to create new approximate algorithms to solve other vehicle routing problems. In 2018, Kumar et al. [39] presented a heuristic algorithm for finding a travelling salesman tour on a connected network. Their approach first identifies a vertex and two related locations that are desirable for inclusion in the tour. Displays this vertex with p and two selected edges output from this vertex with (p,q) and (p,k). It then finds a path connected to the two vertices q and k that passes through all the remaining vertices of the network. The sum of the distances of these edges in a possible tour creates an upper bound for solving the problem. This method was based on the minimum spanning tree, so the complexity of the problem is reduced. The network under consideration in this paper is a connected network with at least two edges output from each vertex.

3. Problem Definition

As has been said, the travelling salesman problem is a well-known problem, first addressed in the 18th century by William Hamilton and Thomas Kirkman. Then in the 1930s, its general form was studied by mathematicians such as Carl Menger. Despite the various methods that have been proposed to solve this problem, the general form of the problem has always been constant. Therefore, the problem can be represented by the same notation. Suppose $G=(V,E)$ is a graph (directional or non-directional) in which the set $V=\{1,2,3,\dots,n\}$ is a set of vertices and the set $E=\{e1,e2,\dots,en\}$ is a set of graph edges. If each edge is assigned a value as a cost (C_e), we can say that the matrix $C=(C_{ij})_{n,n}$ is the cost matrix in which C_{ij} is the value of the edge connecting the vertex i to j in graph G. The Hamiltonian cycle of G is a simple path that includes all vertices of V. TSP's objective function is to find the smallest Hamiltonian cycle in graph G so that the total cost of the edges is minimized. According to graph G, TSP is classified into two categories. If the matrix C is symmetric, it means that graph G is non-directional and the problem is known as symmetric TSP; otherwise, TSP is called asymmetric [10]. The model of the symmetric travelling salesman problem is formulated as follows.

$$min \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}X_{ij} \tag{1}$$

s.t.:

$$\sum_{i=0}^{n} X_{ij}=1 \qquad (j=1,\dots,n) \tag{2}$$

$$\sum_{j=0}^{n} X_{ij} = 1 \qquad (i = 1, \dots, n) \tag{3}$$

$$\sum_{i \in S} \sum_{j \in N-S} X_{ij} \geq 1 \quad (\emptyset \neq S, S \subset \{0, \dots . n\}, |S| \geq 2) \tag{4}$$

$$\sum_{i \in N-S} \sum_{j \in S} X_{ij} \geq 1 \quad (\emptyset \neq S, S \subset \{0, \dots . n\}, |S| \geq 2) \tag{5}$$

$$X_{ij} \in \{0,1\} \qquad (i = 0, \dots, n;\ j = 0, \dots, n) \tag{6}$$

In this model, the constraint mentioned in Equation (2) means that only one edge enters each vertex j. Constraint (3) indicates that only one edge comes out of each vertex i. Constraints (4) and (5) ensure that the found sub-tour that does not have initial vertices is not accepted as a solution. In Constraint (6), the binary variable expresses the problem, which indicates the path that the salesman has taken from vertex i to j.

The TSP problem discussed in this article is a special case of the TSP problem called the metric TSP problem (MTSP). The MTSP problem is a subset of the travelling salesman problem in which a triangular inequality is observed. This inequality states that if we go from vertex i to j in graph G and then from vertex j to vertex k, the sum of the distances between the two vertices must be less than or equal to the distance i to k. It is also assumed in the problem under consideration that TSP is a symmetric graph in which $c(i,j) = c(j,i)$.

4. Proposed Algorithm

The proposed algorithm in this research is a tree-based algorithm. The difference between the proposed algorithm and the algorithm presented by Christofide and other spanning tree-based algorithms is that in other MST-based algorithms, at least the initial spanning tree is plotted for the graph and by adding or subtracting the edges of vertices with an odd degree, they try to find a Hamiltonian tour, but in the proposed algorithm, by sequencing the edges, the best edges are first selected according to the minimum cost criterion, even if these edges do not form a forest or even a tree, then at the same time, the algorithm looks for the minimum spanning tree so that the degree of all vertices is 2 to form a Hamiltonian tour.

Algorithm 1. Pseudo-code of proposed algorithm

Step 1: Complete graph *G (V, E)* with set of vertices *V={a, b, c,…}* with n members and set of edges *E={ab, ac, ad,…}* with *n(n-1)/2* members, each edge having a cost (or distance) of $C=\{c_{ab}, c_{ac}, \dots\}$.
Step 2: Sort the set of costs (C) in ascending order based on one of the sorting methods.
Step 3: Select $n-1$ edges from the top of the list and draw the graph G' with vertices V and selected edges.
Step 4: Calculate the degree of each vertex in the new graph.
Step 5: If the maximum of 2 vertices' degree is exactly equal to one, and there are no vertices with degree 0, go to step 6. Otherwise, the number (total vertices with degree 0) + (1- (total vertices with degree 1)) edge is removed. The order of removing the edges is as follows:
- **Step Five-A**: Remove the costliest edge from the vertex with the highest degree (if the two vertices had the same condition, select the vertex that has the highest sum of the costs of the connected edges; if at the vertex selected to remove the edge, the two edges cost the same, choose the edge whose end is connected to the vertex with a higher degree; in both cases, if the conditions are still the same, choose randomly)
- **Step Five-B**: Calculate the degree of the vertices and go to Step Five-A until the desired number of edges is removed.
- **Step 5-C**: Add the next edges to the list in step two (if we get to the bottom of the list, update the list and start again) and go to step 4.

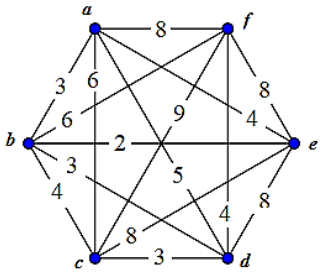
Step 6: Connect the two vertices of degree 1 and go to step 7.
Step 7: Check if the graph is connected, go to step 8 otherwise:
- **Step 7-A**: From the beginning of the list in step 2, add the first edge that connects the graph.
- **Step 7-B**: From all the vertices with degree 3, remove the edge with the highest cost (except the edge added in step 7-A) and go to step 8.

Step 8: Turn the graph G' into the symmetric directional graph.
Step 9: List the edges of the new graph and form the Hamiltonian tour as follows:
Step 9-A: For $k = 1: n-1$, follow the steps below
Step 9-A-A: Select the vertex with the lowest degree and place it inside H_k (if the degrees are equal, select a vertex of your choice)
Step 9-A-B: From the list of step 9, select the first edge that starts with H_k and place the end vertex of this edge in H_{k+1}.
Step 9-A-C: Remove edges containing H_k from the list.
Step 9-A-D: $k \leftarrow k + 1$ and go to step 9-A-B.
Step 9-A-E: Show the path $H_1, H_2, \dots, H_n$.
Step 10: Connect the vertex H_1 to H_n , and display the distance, then, calculate the cost.

3.1. A numerical example

This subsection provides an example that illustrates the worst-case scenario and shows all the steps of the algorithm.

Step 1: Consider the corresponding graph and matrix, which consists of 6 vertices and 15 edges, with distances written on each edge of the graph (see Figure 1).



$$\begin{pmatrix} 0 & 3 & 6 & 5 & 4 & 8 \\ 0 & 0 & 4 & 3 & 2 & 6 \\ 0 & 0 & 0 & 3 & 8 & 9 \\ 0 & 0 & 0 & 0 & 8 & 4 \\ 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 1. Graph with 6 vertexes and the corresponding matrix

Step 2: The ordered list of edges according to their distance is as shown in Table 1.

Tab. 1. Sorted list of edges based on the distances of each edge

Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Edge	*b-e*	*a-b*	*b-d*	*c-d*	*a-e*	*b-c*	*d-f*	*a-d*	*a-c*	*b-f*	*a-f*	*c-e*	*d-e*	*e-f*	*c-f*
Distance	2	3	3	3	4	4	4	5	6	6	8	8	8	8	9

Step 3: With the edges $b-e, a-b, b-d, c-d$ and $a-e$, we form the graph G' as follows.
Step 4: The degree of each vertex in the graph in Figure 2-A is written on each vertex.
Step 5: Since there is a vertex with degree 0 (vertex f) and a vertex with degree 1 (vertex c), so to the number *(Total vertices with degree 0) + (1-(total vertices with degree 1)) = 1+ (1-1) = 1* edge should be removed.
The order of removing the edges is as follows:
Step 5-A: From the vertex b with the highest degree, remove the edge with the highest cost (edge $b-a$ *or* $b-d$) (at the vertex selected to remove the edge, two edges have the same cost, we choose the edge that ends with the vertex which is connected with the vertex with the maximum degree; we see that these conditions are still the same, so we choose the edge b-d randomly (Figure 2-B)).
Step Five-B: Calculate the degree of the vertices and since only one edge must be removed, go to step Five-C.
Step Five-C: Add the next edges to the list in step two (edge $b-c$) (Figure 2-C) and go to step four (the degree of each vertex is written above it).
In the five step: Since there is a vertex (vertex f) with degree 0 and a vertex (vertex d) with degree 1, so to the number (Total vertices with degree 0) + (1- (total vertices with degree 1)) = 1+ (1-1) = 1 edge should be removed.
The order of removing the edges is as follows:
Step 5-A: From the vertex b with the highest degree, remove the edge with the highest cost (edge $b-c$) (Figure 2-D)

Step 5-B: Calculate the degree of the vertices and since only one edge should be removed, go to step 5-C.

Step 5-C: Add the next edges to the list in step two (edge d-f) (Figure 2-E) and go to step four (the degree of each vertex is written above it). Note that, in Step 4, the degree of each vertex is written above it.

Step 5: Since there are no vertices with degree 0, and there are only two vertices with degree 1 (Figure 2-F), go to Step 6.

Step 6: Connect the two vertices with degree 1 and go to Step 7 (Figure 2-G).

Step 7: The graph in Figure 2-F is not connected so:

Step 7-A: From the beginning of the list in Step 2, add the first edge that connects the graph (edge a-c) (Figure 2-G).

Step 7-B: Remove the edge with the highest cost (except the edge added in step 7-A) from all vertices with grade 3 (vertices $a\ and\ c$) and go to step 8. At this point, the edges $c-f$ and $e-a$ are removed.

Step 8: Convert the graph G' to the symmetric directional graph (Figure 2-H).

Step 9: List the edges of the new graph and form the Hamiltonian tour as follows:

Step 9-A: For $k = 1{:}\ 5$, follow the steps below.

Step 9-A-A: Select the vertex with the lowest degree (here the vertex $d\ or\ c$) and set it in H_k (if the degrees are equal, select a vertex randomly).

Step 9-A-B: From the list of the ninth step, select the first edge that starts with H_k and put the last vertex of this edge in H_{k+1}.

Step 9-A-C: Remove the edges containing H_k from the list.

Step 9-A-D: $k \leftarrow k+1$ and go to step **9-A-B**.

Step 9-A-E: Report $(H_1, H_2, \dots, H_6)$ as "e-b-a-c-d-f".

Step 10: Connect the vertex of H_6 to H_1 and the distance is achieved at the cost of 26.

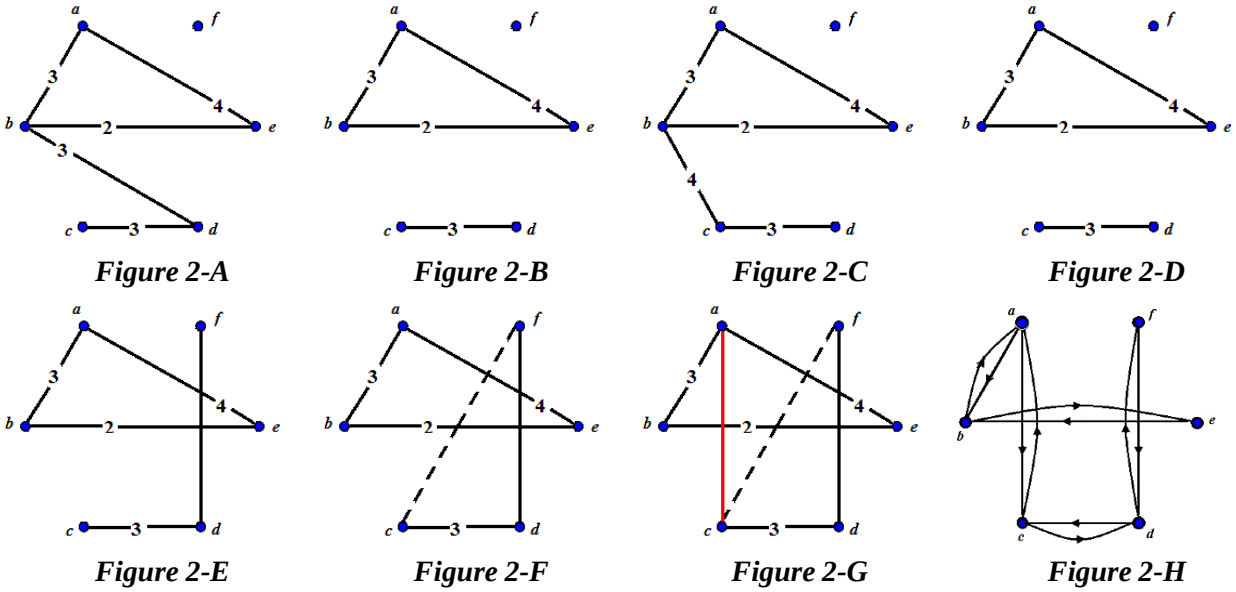


Figure 2-A *Figure 2-B* *Figure 2-C* *Figure 2-D*

Figure 2-E *Figure 2-F* *Figure 2-G* *Figure 2-H*

Fig. 2. Steps of the proposed algorithm

4. Computational Results

The aim of this section is to evaluate the performance of the proposed heuristic algorithm in solving the metric travelling salesman problem. In this regard, the proposed algorithm compared with one of the most well-known heuristics namely the nearest neighbour algorithm as the fastest algorithm [11] and the ant colony optimization algorithm (ACO) which has a very good convergence. The following is a brief description of these algorithms.

The nearest neighbour algorithm is one of the simplest algorithms to solve TSP. The basis of this algorithm is that when determining the path in each stage, the closest and cheapest path is always selected. Because this algorithm is a kind of greedy algorithm, it is easy to implement, and it works very fast. Although this algorithm sometimes provides good solutions, in general, it

can also generate many costly and inconvenient paths [10]. The steps of the NN algorithm, whose computational complexity is O (n^2) are as follows [11]:

Algorithm 2. Pseudo-code of nearest neighbour algorithm

1. Select a vertex (city) randomly.
2. Find the nearest vertex that has never been met and go to it.
3. If there is still a vertex that has not been met, go to step 2.
4. Return to the first city.

Ant colony optimization algorithm is one of the most advanced approximation optimization algorithms. The ant colony algorithm is a constructive meta-heuristic algorithm inspired by the natural behaviour of ants. This algorithm was first proposed by Dorigo in 1992 [19]. A basic algorithm was proposed to solve the problem of finding the best path in a graph based on the ant colony behaviour in search of a path between the nest and the food source. To use the ACO algorithm, we must first create a graph in which the probability of choosing the next path is determined based on Equation 7. Then the amount of pheromone or weight of each edge is updated according to Equation 8 [12].

$$P_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}(\frac{1}{d_{ij}})^{\beta}}{\sum_{k\in allallowedvertices}\tau_{ik}(t)^{\alpha}(\frac{1}{d_{ij}})^{\beta}} \quad (7)$$

$$\tau_{ij}(t+1) = (1-P)\tau_{ij}(t) + \sum_{k\in ant\ chose\ edge(i,j)} \frac{Q}{L_k} \quad (8)$$

where $P_{ij}(t)$ is the probability of the ant passing along the edges i and j, t_{ij} (t) is the number of pheromones corresponding to the edges i and j, d_{ij} is the distance between vertex i and vertex j, α and β are the parameters for controlling t_{ij} (t) and d_{ij} , P is the pheromone evaporation coefficient, L_k is the path cost of the ant k path, and Q is a constant.

The test problems of this problem were generated randomly under the triangle inequality rule. All heuristic and meta-heuristic algorithms are also coded by MATLAB R2017b and run on a computer with a 2.40 GHz processor and 4G memory. The proposed mathematical model is solved in GAMS 25.1.2.

4.1. Evaluation metric

The following metric is used to evaluate the performance of the algorithms**.**

$$GAP_A = \frac{\text{The best solution of the algorithms} - \text{The best solution of algorithm A}}{\text{The best solution of the algorithms}} \times 100 \quad (9)$$

4.2. Numerical results in small-size instances

Table 2 shows the results for the proposed algorithm of the small-size instances. In Table 2, the objective function of algorithms with solutions' deviation from the best-obtained solution is reported. It should be noted that the objective functions are the best solution found by running five times of each of the algorithms. Figure 3 shows a solution to a problem with 20 cities solved by the proposed heuristic algorithm. Figure 4 shows the deviation index value of the best solution found for each of the seven small numerical instances. From this figure, it can be obtained that the ACO algorithm has more deviation from the exact method obtained by GAMS software compared to other algorithms.

Tab. 2. Numerical results in small-size instances

Solving method	Problem size / Criterion	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6	Instance 7
		5	6	8	10	14	18	20
GAMS	Runtime (s)	0.015	0.125	0.143	0.116	0.213	1.22	2.1
	Objective function	19	25	26	31	38	33	34
	GAP%	0	0	0	0	0	0	0
Nearest neighbour algorithm (NN)	Runtime	0.06	0.09	0.091	0.085	0.112	0.21	0.371
	Objective function	24	26	36	50	65	79	100
	GAP%	26.3	4	38.4	61.3	71	140	194
Ant colony optimization (ACO)	Runtime	10.48	10.57	12.81	10.43	9.61	13.83	12.95
	Objective function	19	25	26	31	38	33	37
	GAP%	0	0	0	0	0	0	8.8
The proposed algorithm	Runtime	0.06	0.094	0.096	0.12	0.198	0.295	0.465
	Objective function	19	26	28	34	47	44	47
	GAP%	0	4	7.7	9.6	42	33	27

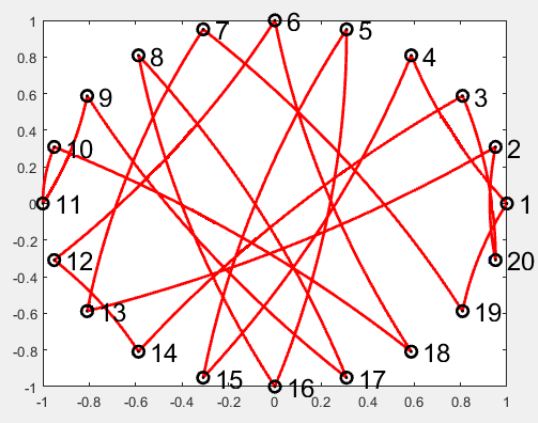


Fig. 3. The output of the proposed algorithm for instance 17 with 20 cities

Figure 5 shows the runtime for solving small-size problems in the studied algorithms. It is clear that the minimum runtime used to solve the problem is related to the nearest neighbor algorithm due to the limited steps of this algorithm. Then, with very little difference, our proposed algorithm had the minimum runtime, and finally, the ACO algorithm had the worst runtime among the compared algorithms

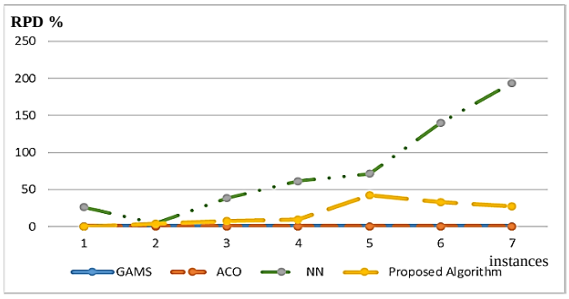


Fig. 4. The RPD of the solution for small-size instances

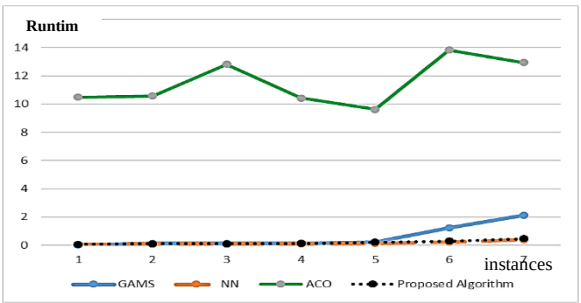


Fig. 5. Runtime of algorithms for small-size instances

4.3. Numerical results in large-size instances

Table 3 shows the computational results for the proposed algorithms in large-size instances. Figure 6 shows a solution to a problem with 100

cities solved by the proposed heuristic algorithm. Figure 7 shows the RPD% for each of the instances. In this way, it can be obtained that the ACO algorithm has the best solution compared to other algorithms and then with a slight difference, the proposed algorithm gives the better solution. Figure 8 shows the runtime used to solve large-size instances in the studied algorithms. As can be seen from the figure, the minimum runtime used to solve the problems is for the NN algorithm. Then the proposed algorithm with a small difference is in second place, and the maximum runtime used to solve the instances was for the ACO algorithm.

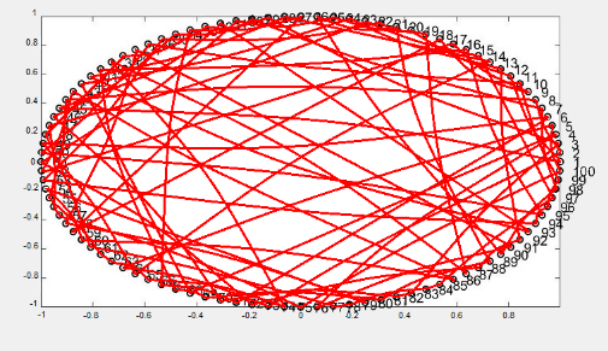


Fig. 6. The output of the proposed algorithm for the instance problem with 100 cities

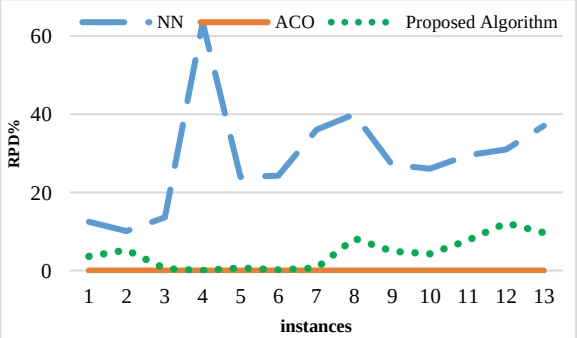


Fig. 7. The RPD of algorithms for large-size instances

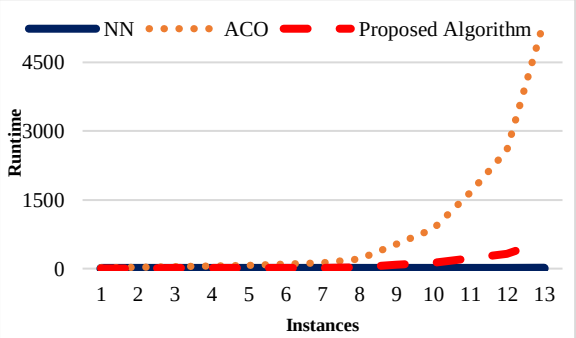


Fig. 8. Runtime of algorithms for large-size instances

5. Conclusion and Future Research

One of the important optimization problems in the field of combinatorial algorithms is travelling salesman problem. Due to its many applications, solving this problem is of interest to researchers and in particular as an important problem of network analysis is used in spatial information technologies. Despite its simple form, since this problem is Np-hard, several heuristic and meta-heuristic algorithms have been proposed. In this paper, a new heuristic algorithm based on a minimum spanning tree is presented. Then, by generating 20 instances, the performance of the proposed algorithm was compared with one of the most famous problem-solving algorithms of the travelling salesman, namely the nearest neighbour algorithm and the ant colony optimization algorithm (ACO) algorithm. The computational results showed that the proposed algorithm has good convergence to the optimal solution. In general, the proposed algorithm has a balance between runtime and the solution found compared to the other two algorithms, so that the nearest neighbour algorithm has a very good runtime to reach the solution, but did not converge well to the optimal solution, and vice versa, the colony optimization algorithm converges very well to the optimal solution, but the runtime is approximately seven times longer than the proposed algorithm. The researchers in this field can study the computational complexity of the proposed algorithm as future studies. Also, researchers can examine and adapt our proposed algorithm for non-metric travelling salesman problem.

Tab. 3. Numerical results in large-size instances

Solving method	Problem size / Criterion	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6	Instance 7	Instance 8	Instance 9	Instance 10	Instance 11	Instance 12	Instance 13
		30	40	45	55	60	70	80	100	150	225	300	350	400
Nearest neighbour algorithm (NN)	Runtime (s)	0.54	0.436	0.63	0.36	0.97	1.06	1.03	1.3	1.37	1.84	2.43	3.13	4.94
	Objective function	190	228	233	616	577	584	1075	2858	4775	7087	10364	13766	18853
	GAP%	12.42	10.1	13.65	63.5	24	24.2	36	40	27	26	29.5	31	37
Ant colony optimization (ACO)	Runtime (s)	19.44	29.15	37.58	60.31	70.13	92.61	120.31	207.7	536.4	871.1	1660.3	2583.4	5402
	Objective function	169	207	205	377	465	470	790	2042	3764	5633	8002	10507	13756
	GAP%	0	0	0	0	0	0	0	0	0	0	0	0	0
The proposed algorithm	Runtime (s)	4.35	5.82	8.43	10.21	15.31	16.24	16.95	25.32	81.23	125.7	223.61	328.9	598.6
	Objective function	175	218	206	377	468	471	759	2208	3952	5869	8626	11784	15081
	GAP%	3.55	5.3	0.5	0	0.64	0.21	0.63	8.13	4.99	4.9	7.79	12.15	9.63

References

[1] Lawler, L., *Kan and Shmoys, The Traveling Salesman Problem*. (1985), John Wiley & Sons.

[2] Menger, K. and K. Sigmund, *Ergebnisse eines mathematischen Kolloquiums*. (1998), Springer.

[3] Dantzig, G., R. Fulkerson, and S. Johnson, *Solution of a large-scale traveling-salesman problem.* Journal of the operations research society of America, Vol. 2, No. 4, (1954), pp. 393-410.

[4] Marinakis, Y., A. Migdalas, and P.M. Pardalos, *Expanding neighborhood search–GRASP for the probabilistic traveling salesman problem.* Optimization Letters, Vol. 2, No. 3, (2008), pp. 351-61.

[5] Goldberg, D.E. and JH Holland, *Genetic algorithms and machine learning.* (1988).

[6] Johnson, D.S. and L.A. McGeoch, *The traveling salesman problem: A case study in local optimization.* Local search in combinatorial optimization, Vol. 1, No. 1, (1997), pp. 215-310.

[7] Bakhouya, M. and J. Gaber, *An immune inspired-based optimization algorithm: Application to the traveling salesman problem.* Advanced Modeling and Optimization, Vol. 9, No. 1, (2007), pp. 105-116.

[8] Snyder, L.V. and M.S. Daskin, *A random-key genetic algorithm for the generalized traveling salesman problem.* European journal of operational research, Vol. 174, No. 1, (2006), pp. 38-53.

[9] Brady, R., *Optimization strategies gleaned from biological evolution.* Nature, Vol. 317, No. 6040, (1985), pp. 804-806.

[10] Kumar, R. and P. Singh, *Pareto evolutionary algorithm hybridized with local search for biobjective TSP*, in *Hybrid Evolutionary Algorithms*. (2007), pp. 361-398.

[11] Hougardy, S. and M. Wilde, *On the nearest neighbor rule for the metric traveling salesman problem.* Discrete Applied Mathematics, Vol. 195, (2015), pp. 101-103.

[12] Dorigo, M. and L.M. Gambardella, *Ant colony system: a cooperative learning approach to the traveling salesman problem.* IEEE Transactions on

evolutionary computation, Vol. 1, No. 1, (1997), pp. 53-66.

[13] Hosseini-Motlagh, S., Ghatreh Samani, M., Jokar, A. Presenting a Model and Heuristic Algorithm for Two-Echelon Location-Routing Problem under Uncertainty Considering the Simultaneous Pickup and Delivery. *Journal of Modeling in Engineering*, Vol. 16, No. 53, (2018), pp. 339-361.

[14] Yousefikhoshbakht, M., Dolatnejad, A., Khorram, E. A Hybrid Modified Ant Colony for Solving the Capacitated Open Vehicle Routing Problem. *Journal of Modeling in Engineering*, Vol. 15, No. 50, (2017), pp. 179-191.

[15] Halim, A.H. and I. Ismail, *Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem.* Archives of Computational Methods in Engineering, Vol. 26, No. 2, (2019), pp. 367-380.

[16] Yang, X.-S., *Introduction to computational mathematics*. World Scientific Publishing Company, (2014).

[17] Eppstein, D., *The traveling salesman problem for cubic graphs.* (2007).

[18] Hasanzadeh, M., Keynia, F. A New Hybrid Intelligent Search Method to Find Global Optimal Solution for Engineering Problems. J*ournal of Modeling in Engineering*, Vol. 17, No. 58, (2019), pp. 81-102.

[19] Maniezzo V., Gambardella L.M., de Luigi F. Ant Colony Optimization. In: New Optimization Techniques in Engineering. Studies in Fuzziness and Soft Computing, vol 141. Springer, Berlin, Heidelberg, (2004).

[20] Soak, S.-M. and B.-H. Ahn. *New subtour-based crossover operator for the TSP*. in *Genetic and Evolutionary Computation Conference*. (2003).

[21] Gündüz, M., MS Kiran, and E. Özceylan, *A hierarchic approach based on swarm intelligence to solve the traveling salesman problem.* Turkish Journal of Electrical Engineering & Computer Sciences, Vol. 23, No. 1, (2015), pp. 103-117.

[22] Bock, F. *An algorithm for solving travelling-salesman and related network optimization problems*. in *Operations Research*. (1958).

[23] Croes, G., *" A method for solving travelling salesman problems," Operation Resources,* Vol. 6, (1958), pp. 791-812.

[24] Flood, M., *" The traveling-salesman problem," Operation research,* Vol. 4, pp. 61-75. 1956.

[25] Lin, S. and BW. Kernighan, *An effective heuristic algorithm for the traveling-salesman problem.* Operations research, Vol. 21, No. 2, (1973), pp. 498-516.

[26] Helsgaun, K., *An effective implementation of the Lin–Kernighan traveling salesman heuristic.* European Journal of Operational Research,. Vol. 126, No. 1, (2000), pp. 106-130.

[27] Van Laarhoven, P.J. and EH Aarts, *Simulated annealing*, in *Simulated annealing: Theory and applications*. (1987), pp. 7-15.

[28] Görkemli, B. and D. Karaboga, *Quick combinatorial artificial bee colony-qCABC-optimization algorithm for TSP*. (2013).

[29] Liu, B., et al., *An effective PSO-based memetic algorithm for TSP*, in *Intelligent computing in signal processing and pattern recognition*. (2006), pp. 1151-1156.

[30] Yang, X.-S. and S. Deb, *Cuckoo search: recent advances and applications.* Neural Computing and Applications, Vol. 24, No. 1, (2014), pp. 169-174.

[31] Rosenkrantz, D.J., R.E. Stearns, and I. Lewis, Philip M, *An analysis of several heuristics for the traveling salesman*

problem. SIAM journal on computing, Vol. 6, No. 3, (1977), pp. 563-581.

[32] Glover, F. and A.P. Punnen, *The travelling salesman problem: new solvable cases and linkages with the development of approximation algorithms.* Journal of the Operational Research Society, Vol. 48, No. 5, (1997), pp. 502-510.

[33] Moore, E.H., *Errata: "On certain crinkly curves" [Trans. Amer. Math. Soc. 1 (1900), no. 1, 72–90; 1500526].* Transactions of the American Mathematical Society, Vol. 1, No. 4, (1900), p. 507.

[34] Christofides, N., *Worst-case analysis of a new heuristic for the travelling salesman problem*. Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, (1976).

[35] Christofides, N., *Graph theory: An algorithmic approach (Computer science and applied mathematics)*. Academic Press, Inc, (1975).

[36] Han, G., et al., *Probabilistic neighborhood location-point covering set-based data collection algorithm with obstacle avoidance for three-dimensional underwater acoustic sensor networks.* IEEE Access, Vol. 5, (2017), pp. 24785-24796.

[37] Meng, L. and L. Wang. *A multi-metaheuristic combined ACS-TSP system*. in *International Conference on Artificial Intelligence and Computational Intelligence*. (2011).

[38] Xu, Z. and B. Rodrigues, *An extension of the Christofides heuristic for the generalized multiple depot multiple traveling salesmen problem.* European Journal of Operational Research, Vol. 257, No. 3, (2017), pp. 735-745.

[39] Kumar, S., et al., *A minimum spanning tree based heuristic for the travelling salesman tour.* Opsearch, Vol. 55, No. 1, (2018), pp. 150-164.

Follow this article at the following site:

Malihe Masoumi & Javad Behnamian. A New Heuristic Algorithm Based on Minimum Spanning Tree for Solving Metric Traveling Salesman Problem. IJIEPR 2024; 35 (1) :1-13
URL: http://ijiepr.iust.ac.ir/article-1-1843-en.html