


A NEW HEURISTIC FOR SOLVING TRAVELLING SALESMAN PROBLEM: ADDING THE FURTHEST VERTEX TO THE CENTRAL TRIANGLE

Fidan Nuriyeva^{1,2*} , Şems Polat³

¹Dokuz Eylül University, Department of Computer Science, Izmir, Turkiye

²Institute of Control Systems of ANAS, Baku, Azerbaijan

³Robert College, Istanbul, Turkiye

Abstract. In this study, an algorithm based on the Insertion Heuristic is proposed to solve the Travelling Salesman Problem. In the proposed algorithm, firstly the center and center triangle are determined, then vertices are added to the initial tour by starting the furthest vertex from the center. Several examples of traveling salesman problem are solved using the proposed algorithm and results are compared with the Nearest Neighbor and Greedy Algorithms. Computational Experiments show that the proposed algorithm is efficient.

Keywords: travelling salesman problem, heuristics, insertion heuristic, nearest neighbor algorithm, greedy method.

AMS Subject Classification: 90C27, 90C59.

Corresponding author: Fidan, Nuriyeva, Dokuz Eylül University, Department of Computer Science, Izmir, Turkiye, e-mail: nuriyevafidan@gmail.com

Received: 25 January 2023; Revised: 3 March 2023; Accepted: 28 March 2023; Published: 30 April 2023.

1 Introduction

The Travelling Salesman Problem (TSP) is one of the most studied problem in combinatorial optimization (Johnson & Papadimitriou, 1985). TSP can be defined as given a set of cities and the distance between every pair of cities, the aim is to find the shortest tour that visits each city exactly once and returns to the starting city. The problem can be modelled as a minimization problem in graph theory as an undirected weighted complete graph, where cities are vertices, paths are the edges, and distance of path is the edge's weight. The TSP is to find a Hamiltonian cycle in a graph of least total cost (Gutin & Punnen, 2002).

The TSP is being applied in many fields such as vehicle routing, computer wiring, overhauling gas turbine engines, drilling of printed circuit boards, the order-picking problem in warehouses, X-Ray crystallography and etc (Gutin & Punnen, 2002; Matai et al., 2010).

The TSP can be divided into three types: symmetric TSP where the distance between each pair of cities is the same in each opposite direction, asymmetric TSP where the distance between each pair of cities in a given space can be different from the inverse distance or paths may not exist in both directions and multi TSP, assigning m salesmen to a given number of cities, and each city must be visited by a salesman exactly once and return to the initial position with the minimum travelling cost.

In this study, a new heuristic algorithm based on adding the farthest vertex to the center is proposed for a symmetric TSP.

The remainder of this paper is organized in the following manner. Section 2 describes the definition and mathematical formulation of the problem. In section 3, solution approaches of

TSP are described, in section 4 the well-known algorithms for solving TSP are introduced in detail. The steps of the proposed algorithm are described in sections 5 and 6. Section 7 covers experimental results and section 8 presents conclusion.

2 Problem Definition and Formulation

The TSP can be modelled as a minimization problem on a complete undirected weighted graph $G = (V, E)$. $V = \{1, 2, \dots, n\}$ is the set of vertices and $E = \{(i, j) : i, j \in V \text{ and } i \neq j\}$ is the set of edges. The problem is to find a Hamiltonian circuit of minimum total length that contains each vertex of V exactly once.

mathematical formulation of the TSP is as follows (Gutin & Punnen, 2002):

$$x_{ij} = \begin{cases} 1 & \text{if the traveller goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

d_{ij} : Travel distance from city i to city j

n : Number of cities

$$\sum_{i=1}^n \sum_{j=1}^n d_{ij} x_{ij} \rightarrow \min \quad (1)$$

$$\sum_{i=1}^n x_{ij} = 1, \quad j = 1, 2, \dots, n \quad (2)$$

$$\sum_{j=1}^n x_{ij} = 1, \quad i = 1, 2, \dots, n \quad (3)$$

$$u_i - u_j + n x_{ij} \leq n - 1; \quad \forall i, j = 1, 2, \dots, n; \quad i \neq j; \quad u_k \geq 0; \quad \forall k = 1, 2, \dots, n; \quad (4)$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, 2, \dots, n \quad (5)$$

Constraint (1) aims to minimize the total cost. Constraint (2) says that that we arrive at city j from exactly one other city. Constraint (3) says that we leave city i to go to exactly one other city. Constraint (4) is subtour breaking constraint. Finally, constraint (5) imposes binary integrality on the variables.

3 Solution Approaches for TSP

It is known that TSP is NP-Hard. The following solution approaches have been developed in order to solve TSP (Lin & Kernighan, 1973; Nuriyev & Nuriyev, 2018; Reinelt, 1994; Rosenkrantz et al., 1977).

3.1 Exact Solution Methods

The first method that comes to mind is the counting method, in which all possible possibilities are tried. Integer linear programming methods and dynamic programming methods can be shown as other methods. The Branch & Bound algorithm is other well-known solution algorithm for TSP. However, since TSP is in the NP-hard class, these methods are computationally expensive and cannot be solved in a reasonable time when the number of cities exceeds 20 (Johnson & Papadimitriou, 1985; Kızılateş & Nuriyeva, 2013).

3.2 Approximation Methods

Since TSP is an NP-hard class problem, due to its difficulty, there is a need for heuristic and approximation algorithms. Approximation algorithms are efficient algorithms that find approximate solutions to optimization problems with provable guarantees on the distance of the returned solution to the optimal one.

The best known approximate algorithms for TSP are Minimal Spanning Tree (MST) based algorithms (guarantee value $1/2$), Christofides Algorithm (guarantee value $3/2$) and others (Rosenkrantz et al., 1977).

3.3 Heuristic Methods

In computer science and optimization, a heuristic algorithm is a procedure that determines near-optimal solutions to an optimization problem in a reasonable computational effort. In practice, heuristic algorithms are preferred to exact algorithms for solving NP-hard problems.

The heuristic algorithms for the TSP fall into three categories: tour construction, tour improvement and composite approach including tour construction and tour improvement.

4 Some Heuristics for Solving TSP

This section presents the well-known algorithms such as Nearest Neighbor Algorithm, Greedy Algorithm and Insertion Heuristic.

4.1 Nearest Neighbor Heuristic

Nearest Neighbor (NN) is the simplest and the most straightforward algorithm for solving TSP (Kızılateş & Nuriyeva, 2013). The key to this approach is to always visit the closest city.

The steps of the NN algorithm are:

1. Select any vertex as a starting city.
2. At each step, go to any one of the closest remaining vertices.
3. Once every vertex has been visited, return to the starting city.

We can get the best result by running the algorithm starting from each vertex.

4.2 Greedy Heuristic

The Greedy heuristic for solving TSP begins by sorting all the edges. Then selects the edge with the minimum cost. It continuously selects the best next choices given a condition that no cycles are formed.

Steps of Greedy heuristic are:

1. Sort all of the edges.
2. Select the shortest edge and add it to tour if it does not violate any of the following conditions: there are no cycles in our tour with less than n edges or increase the degree of any vertex more than 2.
3. If we have n edges in our tour stop, else repeat step 2.
4. Stop.

4.3 Insertion Heuristic

Insertion heuristics are quite straightforward, and there are many variations of this algorithm. The basic of insertion heuristics are to start with a tour of a subset of all cities, and then to insert the rest by some heuristic. The initial sub tour is often a triangle. One can also start with a single edge as sub tour.

Steps of an insertion heuristic are:

1. Select the shortest edge and create a subtour.
2. Select a city not in the subtour, having the shortest distance to any one of the cities in the subtour.
3. Find an edge in the subtour such that the cost of inserting the selected city between the edge's vertices will be minimal.
4. Repeat step 2 until no more cities remain.
5. Stop.

5 Heuristic of Adding Furthest Vertex From the Center

In this study, Adding Furthest Vertex From the Center is proposed. The steps of the algorithm are:

1. Determine the center (O).
2. Build the initial sub tour.
3. Add the vertex (A_k) that is not in the sub tour and farthest from the center.
4. Repeat step 3 until there are no more vertices to add.
5. Stop.

5.1 Determination of the Center

To determine the center, the x and y coordinates are calculated as follows:

$$x = \frac{(\sum_{i=1}^n x_i)}{n}, \quad y = \frac{(\sum_{i=1}^n y_i)}{n}$$

The $O(x, y)$ is the center.

5.2 Generating initial central triangle subtours

In order to generate initial central triangle subtours, three vertices closest to the center (O) are selected. These vertices are $A_1(x_1, y_1)$, $A_2(x_2, y_2)$ and $A_3(x_3, y_3)$. Then, the initial tour of the graph that is consisting of these 3 vertices is constructed: $A_1 - A_2 - A_3 - A_1$. The vertices A_1, A_2, A_3 are deleted from the set V .

5.3 Adding the vertex that is not in the sub tour to the tour

Find the vertex (endpoints are (A_l, A_m)) where the difference in the length of the (A_l, A_m) edge is the least by the sum of the distances between the city to be added (A_k) and the endpoints of the edge (A_l) and (A_m) in a tour (A_l, A_m) . This edge (A_l, A_m) is deleted from the tour and the $A_l A_k$ and $A_k A_m$ edges are added to the tour. Here, the vertex A_k is selected under the condition $A_l A_k + A_k A_m - A_l A_m = \min$.

6 Proposed Algorithm

Proposed algorithm is based on the insertion heuristic. The proposed algorithm starts by determining the center initially and the initial subtour is constructed by selecting the three closest cities to the center. Then, starting from the farthest cities from the center, the cities are added to the tour respectively.

The steps of the algorithm are:

1. Determine the centre (O). Calculate centre with a formula $x = \frac{(\sum_{i=1}^n x_i)}{n}$, and $y = \frac{(\sum_{i=1}^n y_i)}{n}$. $O(x, y)$ is determined as the center.
2. Generate an initial tour passing through the three nearest cities to the center.
3. Find the city that is not in the sub tour and farthest from the centre. Let this city be A_k .
4. Find the edge (A_l, A_m) in the tour with endpoints (A_l) and (A_m) where the difference between the sum of the distances; between the end points and the length of the edge (A_l, A_m) is minimized for the city (A_k) to be added.
This edge (A_l, A_m) is deleted from the tour and the edges $A_l A_k$ and $A_k A_m$ are added. Here, the vertex A_k is selected from the condition $A_l A_k + A_k A_m - A_l A_m = \min$. Delete the vertex A_k from the set of V ($k = k + 1$).
5. Go to Step 3 until there are no more cities to add ($k > n$).
6. Stop.

The steps of proposed algorithm is shown in 11-vertex graph in Figure 1 given below.

7 Computational Experiments

This section presents the results of the computational experiments for the proposed heuristic algorithm. The proposed algorithm is coded in C programming language and tested on an Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz processor and 8,00 GB RAM, 64-bit Windows.

The proposed algorithm and other well-known algorithms are tested using the symmetric TSP instances obtained from TSPLIB (TSPLIB). Table I shows the length of the tour computed by NN, Greedy heuristics and by the proposed algorithm. The numbers at the end of the problem names show the size of the problem. In Table I, selected cells show the best results that algorithms get. Computational Experiments are done on 21 problems. The proposed algorithm produced better solution for 18 out of 21 problems.

8 Conclusions

In this paper, a new heuristic algorithm for solving the symmetric travelling salesman problem is proposed. The algorithm is built upon the concept of insertion heuristics, with a focus on adding the farthest vertex to the center. Computational experiments have been made using benchmarks from the TSPLIB and have been compared by many other methods already proposed in the literature. The experimental data reveals that the proposed heuristic exhibits great performance on the examined instances, often surpassing the effectiveness of numerous well-established approaches. This suggests that the novel heuristic has the potential to make a significant contribution to the field of combinatorial optimization and has applications in related fields as well as room for improvement.

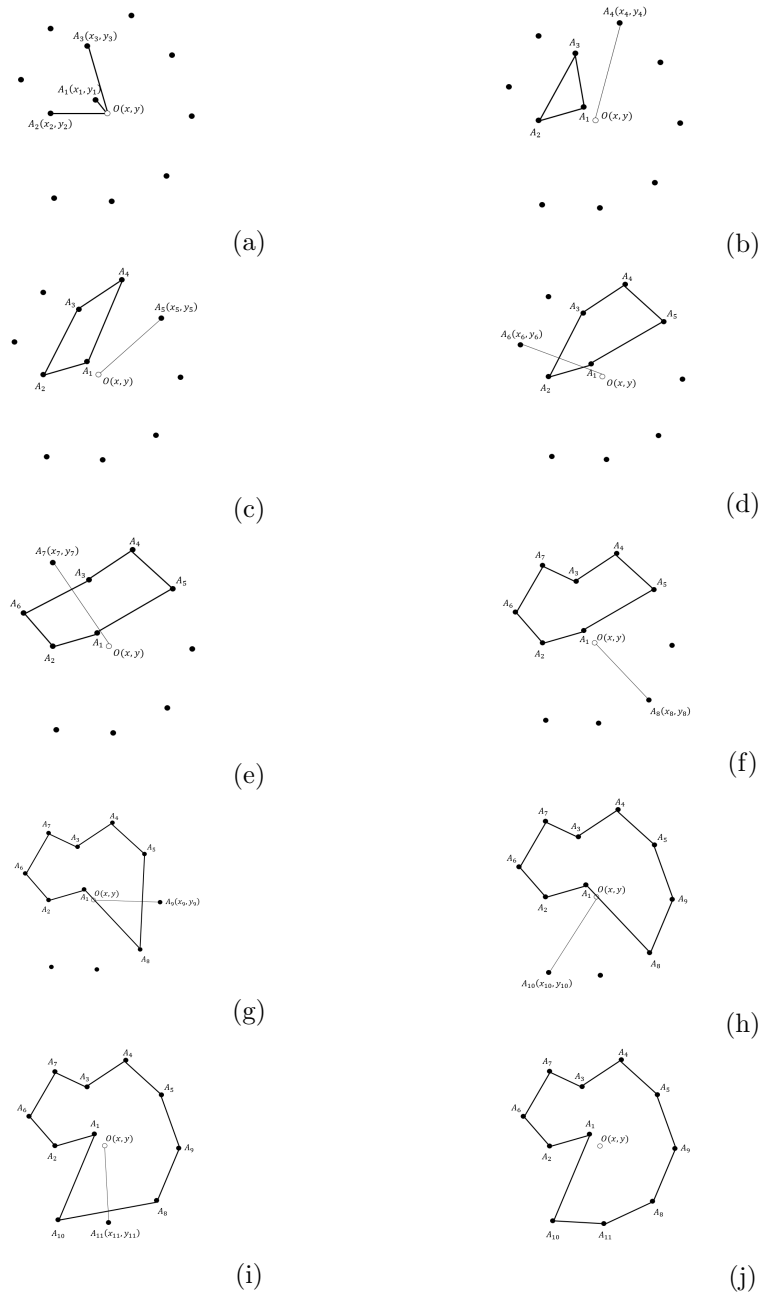


Figure 1: Demonstration of the steps of the proposed algorithm. (a) The center - the point $O(x, y)$ and the 3 closest cities to the center ($A_1(x_1, y_1)$, $A_2(x_2, y_2)$, $A_3(x_3, y_3)$) are determined. (b) The initial tour passing through these 3 cities is formed and the farthest city out of these 3 cities - $A_4(x_4, y_4)$ is determined. (c) Initial tour is updated by adding $A_4(x_4, y_4)$, and the farthest city from the center - $A_5(x_5, y_5)$ is determined among the unselected cities. In subsequent figures ((d), (e), (f), (g), (h), (i), (j)) tour is updated by adding new selected cities. Finally, the last city $A_{11}(x_{11}, y_{11})$ is added to the current tour and the tour passing through all the vertices is found.

Table 1: Computational Experiments

P	Optimal	NN Time(s)	Greedy Time(s)	Proposed Algorithm Time(s)
ulysses16	74.108	78.127 0.000	88.923 0.000	75.342621 0.000000
ulysses22	75.665	86.906 0.000	89.436 0.010	76.665993 0.000000
eil51	426	505.774 0.016	481.518 0.125	475.560638 0.000000
berlin52	7542	8182.192 0.000	9954.062 0.281	8167.796387 0.000000
st70	675	761.689 0.000	746.044 0.485	774.289124 0.000000
eil76	538	612.656 0.016	617.131 0.672	602.925293 0.015000
rat99	1211	1369.535 0.016	1528.308 1.875	1357.762695 0.000000
kroA100	21282	24698.497 0.016	24197.285 1.937	23980.250000 0.000000
kroB100	22141	25882.973 0.016	25815.214 2.469	23720.869141 0.000000
kroC100	20749	23566.403 0.015	25313.671 2.610	22542.970703 0.000000
kroD100	21294	24855.799 0.016	24631.533 2.359	22434.607422 0.000000
kroE100	22068	24907.022 0.016	24420.355 2.609	23542.224609 0.000000
rd100	7910	9427.333 0.015	8702.605 2.922	8621.426758 0.000000
eil101	629	736.368 0.015	789.112 2.609	691.597412 0.000000
lin105	14379	16939.441 0.015	16479.785 3.187	15517.557617 0.000000
pr107	44303	46678.154 0.016	48261.816 2.109	48709.437500 0.000000
ch130	6110	7198.741 0.016	7142.045 7.688	6717.113770 0.000000
kroA150	26524	31482.020 0.047	31442.994 11.094	30646.830078 0.000000
kroB150	26130	31320.340 0.047	31519.083 11.156	28478.873047 0.000000
rat195	2323	2628.561 0.109	2957.176 29.719	2629.051514 0.000000
kroA200	29368	34547.691 0.125	37650.812 45	32816.703125 0.000000

References

Gutin, G., Punnen, A. (eds.). (2002). The Traveling Salesman Problem and Its Variations. *Combinatorial Optimization*, 12. Kluwer, Dordrecht.

- Johnson, D., Papadimitriou, C. (1985). Performance guarantees for heuristics. In *Lawler et al. The Traveling Salesman Problem, A Guide for Tour of Combinatorial Optimization*. Chapter 5, 145-180.
- Johnson, D.S., McGeoch, L.A. (1997). *The Traveling Salesman Problem: A Case Study, Local Search in Combinatorial Optimization*. John Wiley & Sons, 215-310.
- Kizilates, G., Nuriyeva, F. (2013). On the nearest neighbor algorithms for the traveling salesman problem. In *Advances in Computational Science, Engineering and Information Technology*, KTO Karatay University, June 7-9, 2013, Konya, Turkey (pp. 111-118).
- Lawler, E.L., Lenstra, J.K., Rinnoy, Kan, A.H.G., Shmoys D.B. (1986). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons.
- Lin, S., Kernighan, B. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2), 498-516.
- Matai, R., Singh, S., and Mittal, M. L. Traveling Salesman Problem: An Overview of Applications, Formulations and Solution Approaches. InTech. 2010.
- Nuriyeva, F., Kizilates, G. (2017). A New Heuristic Algorithm for Multiple Traveling Salesman Problem. *TWMS Journal of Applied And Engineering Mathematics*, 7(1), 101-109.
- Nuriyev U., Nuriyeva F., (2018). Practical aspects of solving combinatorial optimization problems. *Advanced Mathematical Models and Applications*, 3(3), 179-191.
- Reinelt, G. (1994). *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer-Verlag, Germany.
- Rosenkrantz, D.J., Stearns, R.E. & Lewis, P.M. (1977). An Analysis of Several Heuristics for the Travelling Salesman Problem. *SIAM Journal on Computing*, 6(3), 563-581.
- TSPLIB, <http://comopt.ifl.uni-heidelberg.de/software/TSPLIB95/>