Journal of Modern Technology and Engineering
Vol.4, No.2, 2019, pp.53-61

Jomard Publishing

A METHOD BASED ON HIERARCHICAL CLUSTERING FOR TRAVELLING SALESMAN PROBLEM

Fidan Nuriyeva[1,2*]

[1]Department of Computer Science, Faculty of Science, Dokuz Eylul University,Izmir, Turkey
[2]Institute of Control Systems of ANAS, Baku, Azerbaijan

Abstract. The Travelling Salesman Problem is the most famous optimization problem in the NP-hard class. This paper presents a new heuristic algorithm called hierarchical clustering based on the clustering of the cities. Proposed algorithm works in a three stage. In the first stage, algorithm classifies cities into sets based on a distance between cities. At the second stage, connection between all clusters is made. Entry cities and exit cities is determined for every cluster. At the third stage, the shortest path is found from entry city to exit city covering all cities in every cluster. At the end 2-opt shifting algorithm is applied to the tour. Many problem instances from TSPLIB (travelling salesman problem library) were solved with Nearest Neighbor, Greedy and proposed method. The results obtained by the proposed algorithm have been compared with the Nearest Neighbor (NN) and Greedy algorithms for finding solutions to the symmetric Traveling Salesman Problem. The experimental results show that the proposed method is efficient.

Keywords: Travelling salesman problem, hierarchical clustering, heuristics, NP-hard problems.
AMS Subject Classification: 90C27, 90C59.
Corresponding author: Fidan Nuriyeva, Department of Computer Science, Faculty of Science, Dokuz Eylul University, Izmir, Turkey, e-mail: *nuriyevafidan@gmail.com*
Received: 24 June 2019; Accepted: 27 July 2019; Published: 07 August 2019.

1 Introduction

Travelling Salesman Problem is an NP-hard and intensively studied problem in the field of Combinatorial Optimization field Davendra (2010). The traveling salesman problem consists of a salesman and a set of cities. The main objective of TSP is to find the minimum distance by traversing each of the given set of cities at least once and then traversing back to the start city Johnson & Papadimitriou (1985b). In spite of being very simple to state, the problem is very hard to solve (except for a minimum number of cities) since the number of solutions to be tested grows very fast with n. No exact polynomial time algorithms exist for solving the TSP, however many approximate and heuristic approaches have been proposed Johnson & McGeoch (1997); Johnson & Papadimitriou (1985a); Reinelt (1994). TSP has several applications, such as computer wiring, planning logistics, job sequencing, X-ray crystallography, etc.

Symmetric TSP, Asymmetric TSP, The MAX TSP, The MIN TSP, and etc. are the several variations of TSP Gutin & Punnen (2002). In this paper, we focused on symmetric TSP. In the Symmetric TSP, the distance between two cities is the same in each direction. This paper presents a new algorithm for solving TSP based on hierarchical clustering. The paper is organized as follows. Firstly, in section 2, a mathematical formulation of TSP is presented. Secondly, in section 3, solving approaches for TSP is presented. In section 4, some heuristic algorithms for solving TSP is presented. Section 5, gives information about clustering. In section 6, the proposed algorithm is presented in details. Finally, the experimental results are shown in section

7 and it proposes the conclusion in the last section.

2 Problem Formulation

Travelling Salesman Problem can be modeled as a complete undirected weighed graph $G = (V, E)$, such that cities are the graphs vertices $V = \{1, 2, 3, \ldots, n\}$, and paths are the graphs edges E, and a path's distance is the edge's length. The objective of the TSP is to find shortest closed tour in such that each city is visited once and only once. In symmetric TSP each city pair have the same distance for both directions Lawler et al. (1986).

Lets define x_{ij} as

$$x_{ij} = \begin{cases} 1, & \text{if salesman travels from city } i \text{ to city } j \\ 0, & \text{otherwise.} \end{cases}$$

The integer programming model of the problem is given below Sigal et al. (2002):

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \longrightarrow min, \tag{1}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, 2, \ldots, n, \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, 2, \ldots, n, \tag{3}$$

$$u_i - u_j + nx_{ij} \leq n - 1, \forall i, j = 1, 2, \ldots, n : i \neq j; u_k \geq 0, \forall k = 1, 2, \ldots, n \tag{4}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j = 1, 2, \ldots, n. \tag{5}$$

The objective function (1) aims to minimize the total cost of the tour. Constraint (2) stipulate that only one city can be visited from city i and that only one city is visited at each stage of travel. Constraint (3) on the other hand ensure that a given city is visited at exactly one. Constraint (4) is sub tour elimination constraint. Finally, constraint (5) impose binary conditions on the variables.

3 Solving Approaches for TSP

The Travelling Salesman Problem is an NP-hard problem. Heuristic Algorithms, Metaheuristic Algorithms, Approximate Algorithms and Exact Algorithms are some approaches that's developed for solving TSP Held & Karp (1962); Nuriyev & Nuriyeva (2018).

3.1 Exact Approaches

The best known exact methods for solving TSP are: enumeration, branch and bound method, cutting plane method and dynamic programming. But these approaches are expensive to calculate and takes a long time for the number of cities greater than 20 Appligate et al. (2006).

3.2 Approximate Algorithms

Solving the TSP optimally takes too long, instead one normally uses approximation algorithms, or heuristics. The best known approximate algorithms for TSP are Christofides Algorithm (guaranteed value 3/2), Minimum-Spanning Tree (MST) based algorithms (guaranteed value 1/2), and others Rosenkrantz et al. (1977).

3.3 Heuristic Algorithms

One of the algorithm type which is used in computer science is heuristic algorithms Johnson & McGeoch (1997). Heuristic give no guarantee to find the optimal solution to the given problem, but in many cases gets solution of good quality and obtain it in acceptable time. Heuristic algorithms for TSP can be classified in three groups: heuristics composing the tour, heuristics improving the tour and hybrid heuristics using both.

Heuristics Composing the Tour: The characteristic of these algorithms is not try to improve the result when they find it. Algorithm stops at that point. Nearest neighbor, Greedy, Insertion heuristic and Christofides algorithms are known composing the tour heuristics Rosenkrantz et al. (1977).

Heuristics Improving the Tour: These algorithms try to improve the tour. 2-opt, 3-opt, Lin-Kernighan and similar local optimization algorithms are well known tour improving heuristics.

Hybrid Approaches: These algorithms use both composing and improving heuristics at the same time. Iterated Lin-Kernighan is an example for these algorithms. The best results are obtained by using hybrid approaches.

4 Some Heuristic Algorithms for Solving TSP

In this section, some heuristic approaches for solving TSP is presented.

4.1 Nearest Neighbor Algorithm

The Nearest Neighbor Algorithm (NN) is a heuristic algorithm for finding a sub-optimal solution to the TSP. In this algorithm, it starts with a randomly chosen city as a starting city always adds to the last city in the tour the nearest not yet visited city. The algorithm stops when all cities are on the tour. It finds a shortest path, but solution is not the optimal one Johnson & Papadimitriou (1985a); Kizilates & Nuriyeva (2013); Nuriyeva et al. (2012); Nuriyeva et al. (2013).
The algorithm of NN is:
Step 1. Start with any random city (vertex).
Step 2. Find an edge which gives minimum distance between the current city (vertex) and an unvisited city (vertex) and go there.
Step 3. Are there any unvisited cities (vertices) left? If yes, go to step 2; otherwise go to step 4.
Step 4. Return to the starting city.

4.2 Greedy Algorithm

The Greedy heuristic gradually constructs a tour by repeatedly selecting the shortest distance and adding it to the tour as long as it doesn't create a cycle with less than n distances, or increase the degree of any city by more than 2. We must not add the same distance twice of course Johnson & Papadimitriou (1985a); Kizilates & Nuriyeva (2013); Nuriyeva et al. (2013).
The steps of the algorithm are as following:
Step 1. Sort all edges in ascending order.
Step 2. Select the shortest edge and add it to the tour if it doesn't violate any of the above constraints.
Step 3. Do we have n edges in a tour? If no, go to step 2, otherwise go to step 4.
Step 4. Stop.

4.3 The Nearest Neighbor Algorithm (NND) from Both End-Points

The algorithm starts with a tour containing a randomly chosen city. Then, adds to the chosen city the nearest unvisited city. We will have two end cities. We add a city to the tour such that

this city has not visited before and it is the nearest city to these two end cities. We update the end cities. The algorithm stops when all cities are on the tour Kizilates & Nuriyeva (2013).
The steps of the algorithm are as following:
Step 1. Select randomly a city.
Step 2. Visit the nearest unvisited city to this city.
Step 3. Visit the nearest unvisited city to these two cities and update the end cities.
Step 4. Is there any unvisited city left? If yes, go to step 3.
Step 5. Go to the end city.

5 Clustering

Clustering, groups data instances into subsets in such a manner that similar instances are grouped together, while different instances belong to different groups. Clustering is one of the most studied topics in computer science Anderberg (1973). In clustering analysis, the aim is to present large-scale data with a smaller number of clusters and to present the summary information to the user. In this way, the clusters are hidden within the enormous data and the data is modeled with the clusters it has, so that the area where the user works is collapsed Tyron et al. (1970). Let's consider a set of finite number of elements in the n - dimensional space with m points: $A = \{a^1, a^2, \ldots, a^m\}$

The aim of the clustering problem is to separate the points in the set A, to k number of subsets according to the predefined rules given below:

$$A^j \neq \emptyset, \quad j = 1, 2, \ldots, k, \tag{6}$$

$$A^j \cap A^l = \emptyset, \quad j, l = 1, 2, \ldots, k, \quad j \neq l, \tag{7}$$

$$A = \cup_{j=1}^{k} A_j. \tag{8}$$

To explain the rules given above briefly; each cluster to be formed in (6) is not empty; (7) means that no cluster pair has a common element; in (8), it means that the combination of sets is equal to the data set.

Since clustering is the grouping of similar instances/objects, there is a need for measure that can determine whether two objects are similar or dissimilar. Two main type of measures known to estimate this relation: distance measures and similarity measures. The points from the same cluster are similar to each other and the points from different clusters are dissimilar to each other. Similarities of data points defined with a similarity measure value. As a similarity measure $d(x_1, x_2)$ can be used for $x_1, x_2 \in R^n$. Here, the selection of function $d(x_1, x_2)$ is application-dependent. The most commonly used distance measurement tool is the Euclidean distance. Euclidean distance for $x(x_1, x_2)$ and $y(y_1, y_2)$ on two-dimensional set of X is calculated as follows:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

5.1 Clustering Algorithms

Many clustering methods have been developed, each of which uses a different induction principle. There are two types of classical clustering methods. These are hierarchical clustering and non-hierarchical clustering. Hierarchical clustering algorithms treat each point as a single cluster, and then combine two nearest clusters (points). These operations continue until a single and full set remains. In non-hierarchical clustering, it is aimed to separate clustering data as much as the number of sets determined Xu et al. (2009).

5.2 Hierarchical Clustering Algorithms

Hierarchical clustering algorithms are the most commonly used clustering algorithms because of their simplicity and acceptable performance. These methods construct the clusters by recursively partitioning the instances in either a top-down or bottom-up fashion. These methods can be subdivided into two classes as agglomerative hierarchical clustering and divisive hierarchical clustering Hubert & Baker (1978); Jain & Dubes (1988).

In agglomerative hierarchical clustering, each object initially represents a cluster of its own and then clusters are merged successively until the desired cluster structure is obtained. In divisive hierarchical clustering, all objects initially belong to one cluster. Then the cluster is divided into sub-clusters, which are successively divided into their own sub-clusters. This process continues until the desired cluster structure is obtained.

Agglomerative methods called bottom to top method because they form the hierarchical tree structure, from the leaves to the root; divisive methods are called top to bottom method, because they follow a path from the root to the leaves.

6 New Algorithm for Travelling Salesman Problem

Let $A = \{a_j\}$ be a set of cities (vertices) and $L = \{l_j\}$ be a set of edges, where $l_j = (a_j^1, a_j^2)$, $a_j^1, a_j^2 \in A$, $|A| = n, |L| = \dfrac{n \cdot (n-1)}{2}$.
The proposed method consists of three stages.

Stage A (Clustering)
A.1. The shortest edge is found.
A set of K_1 consisting of the vertices $a_{j_1}^1$ and $a_{j_1}^2$, $l_{j_1} = min\{l_j \mid l_j \in L\}$. $l_{j_1} = (a_{j_1}^1, a_{j_1}^2) \Rightarrow K_1 = \{a_{j_1}^1, a_{j_1}^2\}$ is created.
A.2. The second shortest edge is found. $l_{j_2} = min\{l_j \mid l_j \in L - \{l_{j_1}\}$.
There are two cases: First case is that one of the endpoints of an edge l_j^2 can be in a set K_1. Then other endpoint added to the set K_1. Otherwise, a set K_2 consisting of $a_{j_2}^1$ and $a_{j_2}^2$ vertices is created.
A.3. The third edge with a shortest distance l_j^3 is found. $l_{j_3} = min\{l_j \mid l_j \in L - \{l_{j_1}, l_{j_2}\}$,
$l_{j_3} = (a_{j_3}^1, a_{j_3}^2)$
A.3.1. One of the endpoints of this edge may be in a set K_1 (or K_2), then other endpoint added to the set K_1 (or K_2).
In the next steps, both endpoints of this edge can be in the set K_2, then next edge with a smallest distance is found.
A.3.2. One of the endpoints may be in the set K_1, and another may be in the set K_2. Then, combining sets K_1 and K_2 we get a set $K_1 = K_1 \cup K_2$.
A.3.3. If neither of the endpoints of an edge is not in the set K_1 and K_2, then a set of K_3 is created consisting of $a_{j_3}^1$ and $a_{j_3}^2$.
A.4. This rule continues until all vertices are in clusters, thus m number of clusters is formed. Here, $1 \leq m \leq \lfloor \dfrac{n}{2} \rfloor$. Then, number of steps will be
$\lceil \dfrac{n}{2} \rceil \leq i < \dfrac{n \cdot (n-1)}{2}$.

Stage B (Connection between Sets)
B.1. Next edge with a shortest distance is found.
$l_{j_{i+1}} = min\{l_j \mid l_j \in L - \{l_{j_1}, \ldots, l_{j_i}\}$
B.1.1. If endpoints of this edge are in the same set then we move to the next shortest edge.
B.1.2 If endpoints of this edge are in different sets then connection between these sets is made and endpoints of this edge forms entry and exit vertices between these sets.

B.2 this rule continues until a tour is formed between all sets.

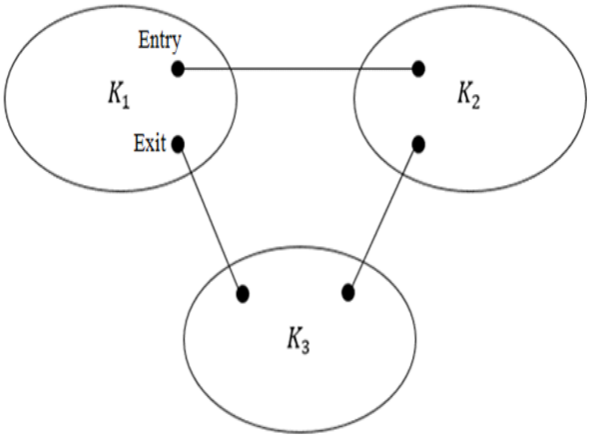

Figure 1: Connection between sets.

Stage C (Finding paths in a Set)
Within each cluster, the shortest path connecting entry and exit vertices and passing through all the vertices is found. Nearest Neighbor Algorithm from Both Endpoints run for every cluster which has entry and exit vertices.
C.1. After the stage B, two vertices are selected from each clusters as entry and exit vertices. An edge with a shortest distance from these vertices in each cluster selected as a member of a shortest path. Other endpoint of this edge considered as one of the starting vertex for Nearest Neighbor Algorithm from Both Endpoints.
C.2. This rule continues until all the vertices in the cluster are in tour and the shortest path connected entry and exit vertices is found.

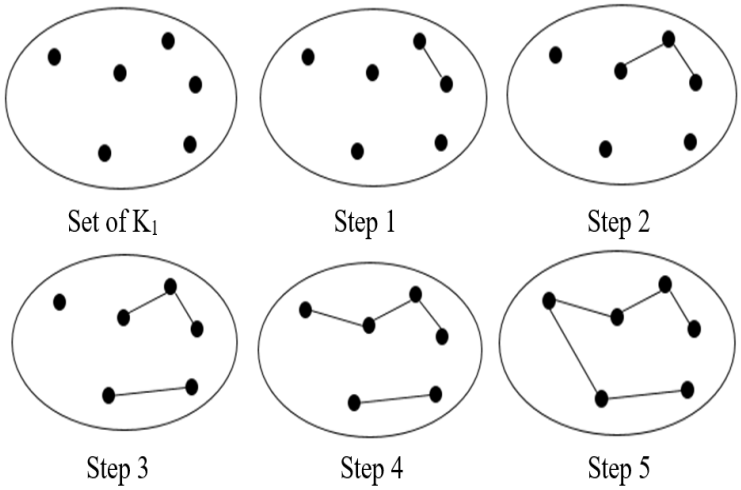

Figure 2: Finding paths in the clusters.

After finding the shortest path, one of the tour developing heuristics for TSP, 2-opt method is run. 2-opt method is the method of adding new two edges that can forms a tour by removing two edges. We do this only if the new tour will be shorter. According to 2-opt method, if endpoints of the edges are entry and exit vertices then 2-opt method cannot be applied.

Indexing the vertices affects the performance of the method. After running 2-opt method, tour will be shifted one step and 2-opt method run again. Two shifting steps is required for this method in this algorithm Nuriyeva et al. (2012); Nuriyeva et al. (2013).

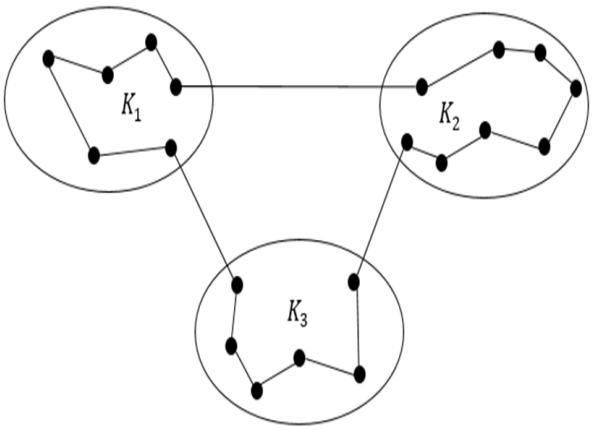

Figure 3: A tour after running algorithm.

7 Experimental Results

The proposed algorithm is implemented in C ++ and the algorithm is tested on the TSPLIB library problems (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/) (Web 1) web. The implementation of NN and Greedy algorithms are also done in C++. To compare the effectiveness of all three algorithms, we took fifteen instances, to see which algorithm gives better result. The comparison is done on the basis of the total distance travelled by the algorithms.

The first row in the table shows the names and dimensions (number of cities) of the library problems. In the other columns, respectively, the optimum results for the mentioned problems, the results obtained with the Nearest Neighbor algorithm, the results obtained with the Greedy Method and the results obtained with the proposed method is shown. The selected cells in the table represent the values closest to the optimum.

Table 1: Experimental Results

P	**Optimal**	**NN**	**Greedy**	**Proposed Algorithm**
eil51	429.983	505.774	481.518	**444.404**
berlin52	7544.365	8182.192	9954.062	**7904.467**
st70	678.597	761.689	**746.044**	756.397
eil76	545.387	612.656	617.131	**576.776**
kroA100	21236.951	24698.497	24197.285	**21893.042**
kroB100	22141	25882.973	25815.214	**22651.425**
kroD100	21294.290	24855.799	24631.533	**22593.949**
kroE100	22068	24907.022	24420.355	**23316.626**
rd100	7910.396	9427.333	**8702.605**	8818.848
eil101	642.309	736.368	789.112	**702.106**
lin105	14382.995	16939.441	16479.785	**14885.812**
ch130	6110.860	7198.741	7142.045	**6535.385**
kroA150	26524	31482.020	31442.994	**27635.015**
kroB150	26130	31320.340	31519.083	**27849.408**
kroA200	29368	34547.691	37650.812	**30572.195**

8 Conclusion

This paper presents a new algorithm based on Nearest Neighbor Algorithm from Double End Points (NND), Greedy algorithm, and hierarchical clustering method. The paper gives a comparison between Nearest Neighbor, Greedy and proposed method to solve the travelling salesman problem. The comparison is based on the criteria of number of cities travelled by all the algo-

rithms and then determines and tests the result on the basis of the distance travelled by them. Experimental results show that the algorithm is efficient.

References

Bekelman, J.E., Li, Y. & Gross, C.P. (2003). Scope and impact of financial conflicts of interest in biomedical research: a systematic review, *JAMA*, *289*(19), 454-465.

Anderberg, M.R. (1973). Cluster analysis for applications, *Academic Press Inc.*

Davendra, D. (2010). Tarvelling salesman problem, *Theory and Applicaitons*, *InTech.org*, 324 p.

Gutin, G. & Punnen, A. (2002). The traveling salesman problem and its variations. *Combinatorial Optimization*, *12*.

Held, M. & Karp, R.A. (1962). A dynamic programming approach to sequencing problems. *Journal of SIAM*, *10*, 196-210.

Hubert, L.J., & Baker, F.B. (1978). Applications of combinatorial programming to data analysis: The traveling salesman and related problems. *Psychometrika*, *43*(1), 81-91.

Jain, A.K. (1988). *Algorithms for Clustering Data*. Prentice Hall, New Jersey.

Johnson, D., Papadimitriou, C. (1985a). *Computational Complexity*. In Lawler et al, Chapter 3.

Johnson, D., Papadimitriou, C. (1985b). *Performance Guarantees for Heuristics*. In Lawler et al, Chapter 5, 145-180.

Johnson, D.S. & McGeoch, L.A. (1997). The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, *1*(1), 215-310.

Kizilateş, G. & Nuriyeva, F.(2013). On the Nearest neighbor algorithms for the traveling salesman problem. *Advances in Computational Science, Engineering and Information Technology. Advances in Intelligent Systems and Computing*, 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.

Nuriyeva F., Kızılateş G. & Berberler M. E.(2012). Experimental analysis of new heuristics for the TSP. *International Conference "Problems of Cybernetics and Informatics", PCI 2012, Section 7, "Control and Optimization", (4), 4-68, Baku, Azerbaijan, September 12-14, (IEEE Xplore Digital Library)*.

Nuriyeva F., Kızılateş G. & Berbeler M.E. (2013). Improvements on heuristic algorithms for solving traveling Salesman problem. *International Journal of Advanced Research in Computer Science*, *4*(11), 1-8.

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, Germany.

Rosenkrantz, D.J., Stearns, R.E. & Lewis, P.M. (1977). An analysis of several heuristics for the travelling Salesman problem. *SIAM Journal of Computing*, *6*, 563-581.

Sigal, I.Kh. & Ivanova, A.P. (2002). *Introduction to Applied Discrete Optimization: Models and Computational Algorithms.*Physics and Mathematics Publishers, International Academic Publishing Company Nauka, Russian Academy of Sciences, Moscow, 204 (in russian).

Tyron, R.C. & Bailey, D.E. (1970). *Cluster Analysis.* McGraw-Hill.

Xu, R. & Wunsch, D.C. (2009). *Clustering.* IEEE Press, New Jersey.

http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/.