Transaction of Azerbaijan National Academy of Sciences, Series of Physical-Technical and Mathematical Sciences: Informatics and Control Problems, Vol. XXXVI, No.3, 2016
www.icp.az/2016/3-10.pdf

F.R. NURIYEVA

HEURISTIC GRID ALGORITHM FOR TRAVELING SALESMAN PROBLEM

In this paper, a new heuristic grid algorithm is proposed for TSP. In this algorithm, a grid is constructed by selecting specific vertices firstly, then accepting these vertices as starting points, NND algorithm is applied in order to find a tour that includes all of the vertices. Most of the sample problems from TSPLIB Library are solved with NN, NND and the proposed algorithm respectively and their results are comparatively analyzed. Experimental results show that the algorithm proposed in this study could be accepted as an efficient alternative.

Keywords: traveling salesman problem, heuristic algorithms, nearest neighbour algorithm, nearest neighbour algorithm from both end points

1. Introduction. One of the most important member of the large set of combinatorial optimization problems is undoubtedly the traveling salesman problem (TSP), which involves the task of determining a route among a given set of nodes with the shortest possible length [1-3]. The study of this problem has attracted several researchers in various fields such as artificial intelligence, biology, mathematics, physics, and operations research, and there are many papers on the problem [1, 4, 5]. No known exact polynomial time algorithm exists to solve the TSP, however many approximate algorithms applying various heuristic approaches have been proposed in the past. It has several applications even in its purest formulation, such as planning logistics, threading of scan cells in a testable VLSI circuit, X-ray crystallography, etc. In standard TSP, the goal is to find the minimum length Hamiltonian cycle through a set of *n* cities, given the distances between all pairs of cities [1, 4, 6].

This paper presents a heuristic construction algorithm for solving TSP based on Nearest Neighbour (NN) algorithm and the nearest neighbor algorithm from both end points (NND) algorithm [7, 8]. This paper is organized as follows. In section 2, in section 2, a mathematical model of TSP is presented. In section 3, solution approaches for TSP are discussed. In section 4, the proposed algorithm is presented in detail. The experimental results are analyzed in Section 5. The conclusion is given in Section 6.

2. Mathematical model of TSP. The TSP is stated as, given a complete graph, *G*, with a set of vertices, *V*, a set of edges, *E*, and a cost, c_{ij}, associated with each edge in *E*. The value c_{ij} is the cost incurred when traversing from vertex $i \in V$ to vertex $j \in V$. Given this information, a solution to the TSP returns the cheapest Hamiltonian cycle of *G*. A Hamiltonian cycle is a cycle that visits each node in a graph exactly once. This is referred to as a tour in TSP terms.

A salesman is required to visit each of *n* given cities once and only once, starting from any city and returning to the original place of departure. The question discussed here is which tour should be chosen in order to minimize the total travel distance?

The distances between any pair of cities are assumed to be known to the salesman. Distance can be replaced by other notions, such as time or money. In the following, the term 'cost' is used to represent any such notions.

The TSP can be formulated as follows [3]:

$$D = \left(d_{ij}\right)_{n\times n} = \begin{cases} d_{ij} & (i \neq j) \\ M & (i = j) \end{cases} \tag{2.1}$$

where *D*, d_{ij}, *M* stands for cost matrix, cost of going from city *i* to city *j*, a sufficiently large weight respectively. A permutation of integers from 1 to *n* ($\pi[0]$, $\pi[1]$, and $\pi[n-1]$) gives a solution of TSP.

$$f(\pi) = \left(\sum_{i=0}^{n-2} d_{\pi[i],\pi[i+1]}\right) + d_{\pi[n-1],\pi[0]} \tag{2.2}$$

Apparently, a permutation π minimized $f(\pi)$ is the best solution of the TSP. Properties of the cost matrix *D* are used to classify problems.

If $d_{ij} = d_{ji}$ for all *i* and *j*, the problems is said to be symmetric; otherwise, it is asymmetric. In this paper, we focus on a symmetric TSP.

3. Approaches for solving TSP. The algorithms to solve TSP can be divided into three classes: exact algorithms, heuristic algorithms and approximate algorithms [1, 4, 10].

Exact algorithms are guaranteed to find the optimal solution with a bounded number of steps. However, these algorithms are quite complex and very demanding of computational capabilities [3].

The heuristic algorithms can obtain good solutions; however, it cannot be guaranteed that the optimal solution will be found. In general, heuristic algorithms are subdivided into the following three classes: tour construction algorithms, tour improvement algorithms and hybrid algorithms [2]. The tour construction algorithms gradually build a tour by adding a new city at each step, such as the nearest neighbor algorithm, the insertion algorithm, algorithm based on spanning tree, the saving algorithm and the random algorithm. Tour improvement algorithms improve a tour by performing various exchanges, such as 2-opt and 3-opt [6].

The Nearest Neighbour Algorithm (NN)

Among the tour construction heuristics, the nearest neighbor heuristic is the simplest one. The nearest neighbor (NN) algorithm for determining a traveling salesman tour is as follows. The salesman starts at a city then visits the city nearest to the starting city. Afterwards, he visits the nearest unvisited city, and repeats this process until he has visited all the cities, in the end, he returns to the starting city.

The steps of the algorithm are as follows:

Step 1. Select a random city.

Step 2. Find the nearest unvisited city and go there.

Step 3. Are there any unvisited cities left? If yes, go to Step 2.

Step 4. Return to the first city.

A better result can be obtained by running the algorithm over again for each vertex and repeat it for *n* times.

The Nearest Neighbour Algorithm from Both End Points (NND)

The algorithm starts with a vertex chosen randomly in the graph. Then, the algorithm continues with the nearest unvisited vertex to the chosen vertex. We have two end vertices. We add a vertex to the tour such that this vertex has not been visited before and it is the nearest vertex to these two end vertices. We update the end vertices. The algorithm ends after visiting all vertices [8, 9].

The steps of the algorithm are as follows:

Step 1. Choose an arbitrary vertex in the graph.

Step 2. Visit the nearest unvisited vertex.

Step 3. Visit the nearest unvisited vertex to these two vertices and update the end vertices.

Step 4. Is there any unvisited vertex left? If yes, then go to Step 3.

Step 5. Go to the end vertex from the other end vertex.

4. Proposed algorithm. We start the algorithm with finding the central vertex. Then algorithm continues with finding the farthest vertex to the central vertex among all vertices. Then, the found farthest vertex is added to the list of selected vertices. The list of selected vertices is updated by finding the farthest vertex at each step as long as the number of the selected vertices is less than 4. After these steps, we obtain four margin vertices. Middle point of the central vertex and margin vertices with intermediate vertex is found. Then the found vertices are assumed as a starting point and applying NND algorithm to these vertices we find the tour.

The steps of the proposed algorithm are as follows:

Step 1. Find the central vertex $O(x_0, y_0)$ in the graph.

Step 2. Find the farthest vertex $A1(x_1, y_1)$ to the central vertex among all vertices and add it to the list of selected vertices.

Step 3. Find the farthest vertex $A2(x_2, y_2)$ to the vertices in the selected list and add it to the list of selected vertices.

Step 4. If the number of the selected vertices is less than 4, go to Step 3. Consequently, by finding the vertices $A3(x_3, y_3)$ and $A4(x_4, y_4)$ we have 4 margin vertices.

Step 5. The midpoint $B1(z_1, t_1)$ of the vertices O, $A1$ and $A3$, is found as follows:

$z_1 = (x_0 + x_1 + x_3)/3,\ t_1 = (y_0 + y_1 + y_3)/3$

Similarly, the midpoint $B2$ of the vertices O, $A3$ and $A2$ the midpoint $B3$ of the vertices O, $A2$ and $A4$, the midpoint $B4$ of the vertices O, $A4$ and $A1$ are found.

Step 6. Find the intermediate vertices.

The midpoint C1(u1,v1) of the vertices O, B1, A1 and B4, is found as follows:

$u1 = (x_0 + z_1 + x_1 + z_4)/4,\ v1 = (y_0 + t_1 + y_1 + t_4)/4,$

Similarly, the midpoint $C2(u2, v2)$ of the vertices $O, B2, A3$ and $B1$, the midpoint $C3(u3, v3)$ of the vertices $O, B3, A2$ and $B2$ and the midpoint $C4(u4, v4)$ of the vertices $(O, B4, A4)$ and $B3$ are found.

Finally, we have central vertex O, end vertices $A1, A2, A3, A4$, middle vertices $B1, B2, B3, B4$ and intermediate vertices $C1, C2, C3, C4$.

Step 7. By assuming these vertices as starting point and applying NND algorithm to them, we obtain a tour for the TSP. By taking these 13 points as "Selected Vertices", we find the nearest neighbour to each one. Add the nearest neighbour to the list of the selected vertices and add the edge, which connects the nearest neighbour to the selected vertices list, to the list of selected edges. In this way, the algorithm proceeds until the tour is found that includes all vertices. Moreover, a tour is constructed by adding 9 vertices $(O, A1 - A4, B1 - B4)$ to the selected vertices list.

5. Example. The proposed algorithm is clarified with an example. In this example, we have 16 vertices.

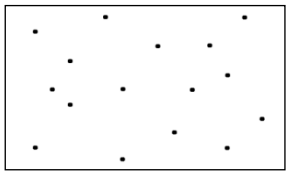

Fig. 1. Vertices of the problem

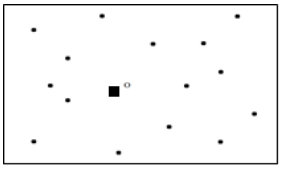

Fig. 2. Finding the central vertex O

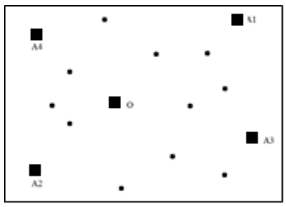

Fig. 3. Finding the margin vertices $A1, A2, A3, A4$

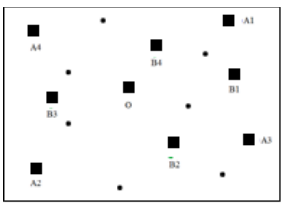

Fig. 4. Finding the middle points $B1, B2, B3, B4$ of the central vertex and $A1, A2, A3, A4$

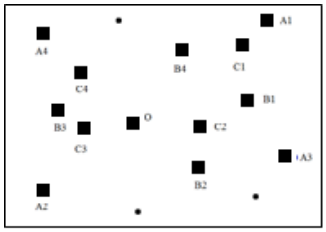

Fig. 5. Finding the intermediate vertices
$C1, C2, C3, C4$

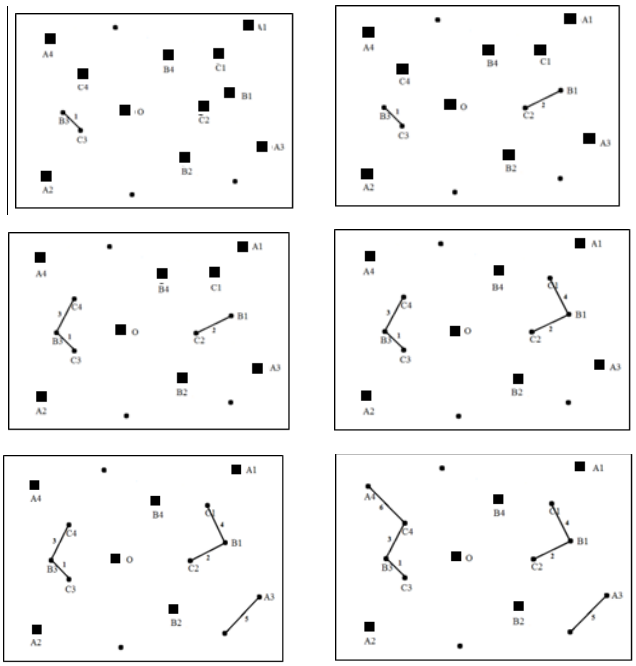

Fig. 6. Creating edges by applying NND algorithm to this vertices

By executing the algorithm, one can obtain a tour which is shown in Figure 7.

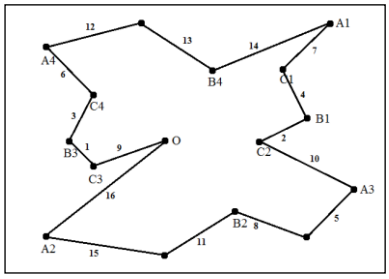

Fig. 7. The obtained tour

6. Experimental results. In this section, we report the experimental results obtained with NN, NND and proposed method on 14 TSP examples from TSPLIB [11, 12]. The results given in Table 1 show that the proposed method, in general, is able to find the best solution with higher quality of all solutions for almost all examples than NN and NND algorithm.

Table 1
Computational Results

Problem	Optimum	NN	NND	9 vertex	13 vertex
Eil51	426	514	511	463	461
Eil76	538	712	594	617	578
Rat99	1211	1565	1384	1313	1286
Rd100	7910	9941	9338	8879	8650
Eil101	629	825	744	725	719
Ch130	6110	7575	7018	6616	6705
Ch150	6528	8195	7028	7207	7426
Rat195	2323	2762	2623	2566	2561
D198	15780	18655	17849	17460	17459
Rd400	15281	19168	18305	17377	17627
D493	35002	43646	41194	40427	40241
Rat575	6773	8449	7969	7676	7513
D657	48912	61874	60542	56597	54227
Rat783	8806	11255	10830	9691	9774

The first column of Table 1 contains the names of the problem (the number in the name gives the problem dimension, that is, the number of cities), the second column contains optimal tour length, third column solutions with NN algorithm, fourth column solutions with NND algorithm.

The fifth column in Table 1 contains the solution obtained by using 9 vertices consisting of central vertex (O), margin vertices $(A1, A2, A3, A4)$ and middle points $(B1, B2, B3, B4)$ of the central vertex and margin vertices.

The sixth column in Table 1 contains the solution with 13 vertices consisting of central vertex (O), margin vertices $(A1, A2, A3, A4)$, middle points $(B1, B2, B3, B4)$ of central vertex (O) and margin vertex, and intermediate $(C1, C2, C3, C4)$ vertices.

In Table 1, cells marked with grey show the best results among the solutions.

To sum up, the proposed method performs better than NN and NND with respect to the quality of solution and the running time according to the experimental results given in Table 1.

7. Conclusion. In this paper, we present a heuristic algorithm to solve TSP based on finding a central point in a graph. The proposed algorithm is implemented in C++. Then we compare the proposed method with NN and NND. According to the experimental results, the proposed method is able to produce better solutions in reasonable time than NN and NND algorithms.

References

1. Gutin G., Punnen A. (eds.). The Traveling Salesman Problem and Its Variations. Vol. 12 of Combinatorial Optimization. Kluwer, Dordrecht, 2002.
2. Johnson D.S. and McGeoch L.A., The Traveling Salesman Problem: A Case Study, Local Search in Combinatorial Optimization, , pp. 215-310, John Wiley & Sons, 1997.
3. Lawler E.L., Lenstra J.K., Rinnoy Kan A.H.G. and Shmoys D. B. The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization, John Wiley & Sons, 1986.
4. Lenstra J.K. Clustering a Data Array and the Traveling-Salesman Problem, Operations Research, 22(2), pp. 413-414, 1974.
5. Appligate D.L., Bixby R.E., Chavatal V. and Cook W.J. The Travelling Salesman Problem, A Computational Study, Princeton Univesity Press, Princeton and Oxford, pp. 593, 2006.
6. Nuriyeva F., Kızılateş G. and Berberler M.E. Improvements on Heuristic Algorithms for Solving Travelling Salesman Problem, 4(11), pp.1-8, 2013.

7. Lin S. and Kernighan B. An effective heuristic algorithm for the traveling-salesman problem, Operations Research, 21(2), pp. 498-516, 1973.
8. Kızılateş G. and Nuriyeva F. On the Nearest Neighbour Algorithms for the Traveling Salesman Problem, The Third International Conference on Computational Engineering and Information Technology, pp. 111-118, Konya, Turkey, June 07-09, 2013.
9. Reinelt G. The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, Germany, 1994.
10. Kızılateş G. and Nuriyeva F. A Parametric Hybrid Method for the Traveling Salesman Problem, Mathematical and Computational Applications, Vol 18(3), pp. 459-466, 2013.
11. www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/
12. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ST

UOT 519.8

F.R. Nuriyeva

Tacir məsələsi üçün hiperevristik şəbəkə algoritmi

Tacir məsələsi üçün evristik şəbəkə metodu adlanan yeni bir alqoritm təklif edilmişdir. Bu alqoritm ilə əvvəlcə müəyyən təpələr seçilərək bir şəbəkə təşkil edilir, sonra isə bu təpələr başlanğıc nöqtələr olaraq qəbul edilib iki ucdan ən yaxın qonşu (NND) alqoritmi tətbiq edilərək bütün təpələrdən keçən dövrə tapılır. TSPLIB kitabxanasında yer alan bir çox problem təklif olunan alqoritm, NN (ən yaxın qonşu alqoritması) və NND alqoritmləri ilə həll edilərək müqayisə edilmişdir. Hesaplama eksperimentleri təklif edilən alqoritmin daha effektiv və səmərəli olduğunu göstərir.

Açar sözlər: tacir məsələsi, evristik alqoritmlər, ən yaxın qonşu alqoritmi, iki ucdan ən yaxın qonşu alqoritmi

УДК 519.8

Ф.Р. Нуриева

Гиперэвристический решетчатый алгоритм для задачи коммивояжёра

Предложен новый, так называемый, эвристический решетчатый алгоритм для задачи коммивояжёра. Этим алгоритмом сначала выбираются определенные точки, организуется решетка, а потом, принимая эти точки как начальные, применяется алгоритм ближайшего соседа с двух сторон (NND) и находится маршрут, проходящий через все точки. Проведенные вычислительные эксперименты со многими задачами из TSPLIB с предложенным алгоритмом NN (алгоритм ближайшего соседа) и алгоритмами NND показывают высокую эффективность предложенного алгоритма.

Ключевые слова: задача коммивояжёра, эвристические алгоритмы, алгоритм ближайшего соседа, алгоритм ближайшего соседа *с* двух сторон

Institute of Control Systems of ANAS
Dokuz Eylul University, Department of Computer Science
(Izmir, TURKEY)

Presented 25.02.2016