International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 5 | ISSN : 2456-3307

An Improved Nearest Neighbor Algorithm for Solving TSP

Emran Islam*[1], Mariam Sultana[2], Faruque Ahmed[3]

*[1,2]Research students, Department of Mathematics, Jahingirnagar University, Savar, Dhaka, Bangladesh

[3]Professor, Department of Mathematics, Jahingirnagar University, Savar, Dhaka, Bangladesh

ABSTRACT

Traveling salesman problem is one of the most important mathematical concepts having very high socio economic impact. But since TSP is NP-complete, finding an optimal solution becomes very hard when problem size increases. Conceptually TSP is very easy to understand but solving TSP is quite difficult than anyone can imagine. In this paper we have represented an improved form of nearest neighbor algorithm for solving TSP. We have also represented a comparative study between our proposed algorithm and original algorithm to test efficiency.

Keywords : Traveling Salesman Problem, Approximate algorithms, NNA

I. INTRODUCTION

The traveling salesman problem is one of the most widely studied optimization problems having very dense historical background [1]. It is one kind of NP hard problem. [2] This problem has become a testing ground for new algorithms for its simplicity of formulation. In TSP a bunch of cities, mutually connected, are considered as a connected graph. Distances between each city are given. Each city represents as node and their distance represents as arc. A salesman needs to visit each and every city exactly once without making any repetition then come back to the starting city, traveling minimum distance.[3] Therefore, it is clear that TSP is a minimization problem to find shortest path under the condition that all cities are visited without repetition and a coincidence happens to the starting and ending city. Literature [4,5,6] shows that TSP is widely connected with many fundamental problems of computer science and engineering. For that reason, finding an efficient algorithm to solve TSP is a major concern for mathematicians and computer scientists for decades [1]. There are four types of algorithms for solving TSP: exact algorithms, heuristic algorithms, approximate algorithms and metaheuristics algorithms [7]. Solving TSP using exact algorithm we always have an optimal solution. [8] However, these type of algorithms are quite complex and requires computer facility for large problems. The other three types of algorithms can obtain good solutions with less time complexity, but it cannot be guarantee that the optimal solution will be found [9,10]. The nearest-neighbor is one of the very efficient algorithms. In this algorithm computational steps increases in some reasonable proportion with the size of the problem. But computational process in NNA is quite difficult when problem size is large. Section II and section III represents original Nearest Neighbor Algorithm (NNA) and our improved Nearest Neighbor Algorithm (INNA).

II. Nearest Neighbor Algorithm

To solve TSP with a Nearest Neighbor Algorithm we need to look at all the arcs coming out of the node (city) that have not been visited and choose the next

closest city, finally return to the starting city when all the other cities are visited. To solve TSP using Nearest Neighbor Algorithm we can use the following steps: [11]

Step 1: Pick any starting node.

Step 2: Look at all the arcs coming out of the starting node that have not been visited and choose the next closest node.

Step 3: Repeat the process until all the nodes have been visited at least once.

Step 4: Check and see if all nodes are visited. If so return to the starting point which gives us a tour.

Step 5: Draw and write down the tour, and calculate the distance of the tour.

III. Proposed Algorithm

Here is a proposal of improved Nearest Neighbor Algorithm:

Step 1: Construct, if necessary, the given network G=(V,E) where V is the vertex set (cities) and E is the edges set (corresponding distances).

Step 2: Pick the designated root vertex $s \in V$. Set Vs={s}.

Step 3: Determine child vertex set of current root vertex Vc=V-Vs

Step 4: If the child vertex set Vc is not empty, goto step-5. If Vc is empty, goto step 7.

Step 5: Make a sorted list of all adjacent edges joining the root vertex and child vertices in ascending order. Choose the edge with minimum value (distance, cost etc). If there is more than one edge with same value, it's okay to select randomly.

Step 6: Mark the vertex adjacent with the selected edge at step-4 as next root vertex and update Vs. Go to step 3.

Step 7: Return to the designated root vertex and hence the optimal solution obtained.

IV. Implementation of Algorithms

For the implementation of NNA and our propose algorithm on a specific example, we construct a five city problem.

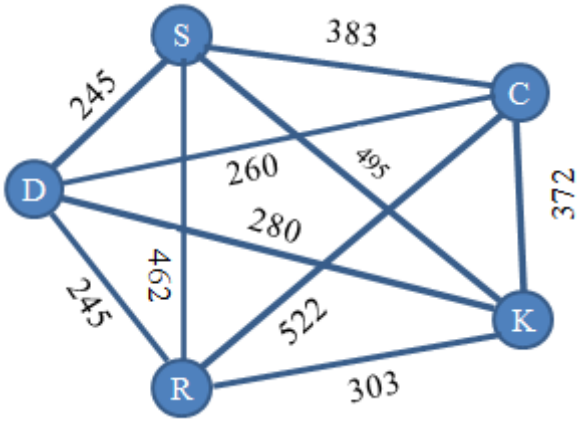
Consider the following distance matrix of five major cities of Bangladesh- Dhaka, Khulna, Chittagong, Sylhet and Rajshahi.[12] We use first latters of the cities to represent the distance matrices.

	D	K	C	S	R
D	∞	280	260	245	245
K	280	∞	372	495	303
C	260	372	∞	383	522
S	245	495	383	∞	462
R	245	303	522	462	∞

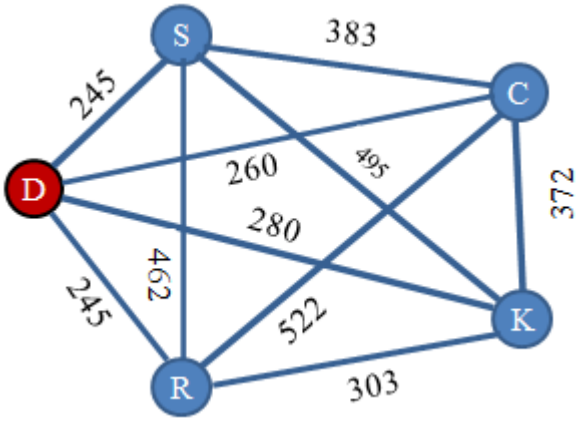
We consider Dhaka as our start vertex. Therefore, task of the problem is to find shortest path visiting all other cities only once and the tour must end at Dhaka.

A. Solution using NNA

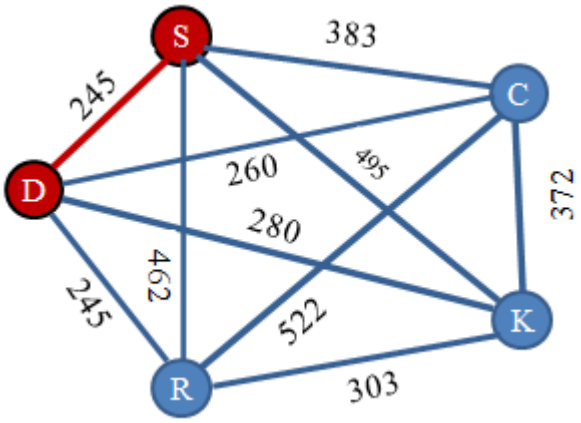
The corresponding graphs considering each city a node and denoting with their first latter is as follows:



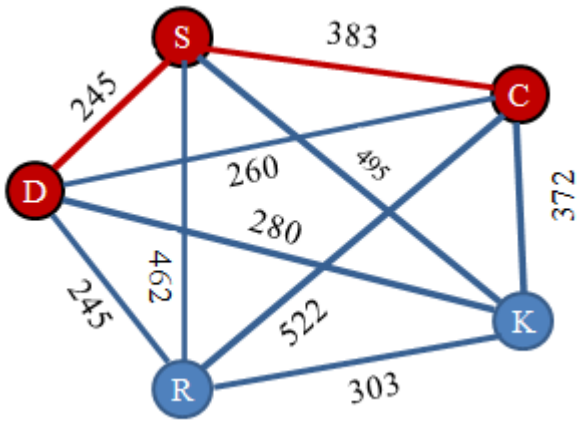
Start: We start from Dhaka.



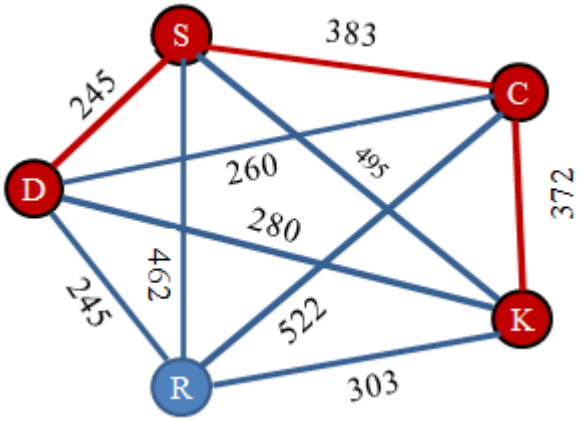
First step: From Dhaka its nearest vertex is Sylhet and Rajshahi with same distance 245 km. So, according to the algorithm we may choose randomly one of them. We choose Sylhet and mark it as visited.



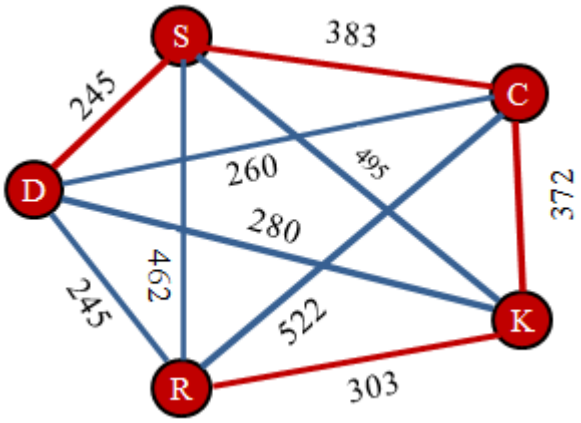
Second step: From Sylhet its nearest vertex is Chittagong with distance 383 km. we choose Chittagong and mark it as visited.



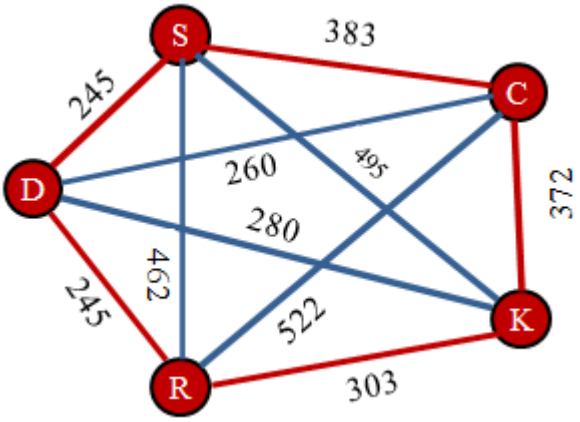
Third step: From Chittagong its nearest vertex is Khulna with distance 383 km. Because we can't choose Dhaka with distance 260km since it will create a sub tour. We choose Khulna and mark it as visited.



Fourth step: From Khulna the nearest neighbor is Rajshahi with distance 303 km.



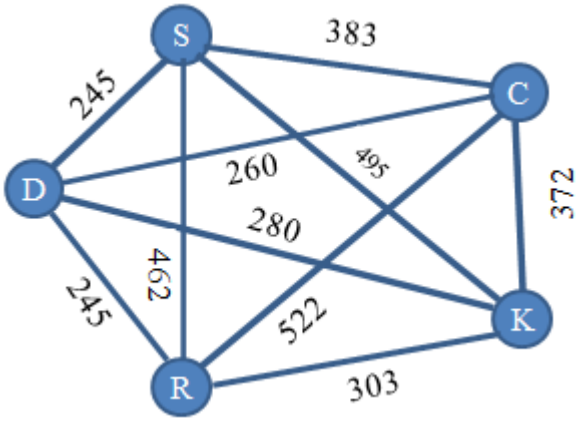
Final step: Since all vertices are visited, we come back to Dhaka.



Therefore our optimal route is D→S→C→K→R→D with total distance = 245 + 383 + 372 + 303 + 245 =1548

B. Solution Using Proposed Algorithm

Initialization: We construct following graph from the given distance matrix.



We want to start our tour from Dhaka. So, our designated start vertex is D. Therefore,

Root vertex set Vs={D}

Child vertex Set Vc=V-Vs={S,C,K,R}

Distance travelled = 0

Iteration 1: Sorted list of all adjacent edges joining the root vertex D and child vertices in ascending order,

DS	245
DR	245
DC	260
DK	280

We may choose DS or DR. We choose DS and mark S as root vertex.

Root vertex set Vs={D,S}

Child vertex Vc={C,K,R}

Travelled way till now: D→S

Distance travelled=0+245=245

Iteration 2: The adjacent edges with root vertex S and child vertices are SC, SK, SR. Sorting them in ascending order according to their distance we have,

SC	383
SR	462
SK	495

We choose SC and mark C as root vertex.

Root vertex set Vs={D,S,C}

Child vertex Vc={,K,R}

Travelled way till now: D→S→C

Distance travelled=245+383=628

Iteration 3: The adjacent edges with root vertex C and child vertices are CK,CR. Sorting them in ascending order according to their distance we have,

CK	372
CR	522

We choose CK and mark K as root vertex.

Root vertex set Vs ={D,S,C,K}

Child vertex Vc ={R}

Travelled way till now: D→S→C→K

Distance travelled=628+372=1000

Iteration 4: The adjacent edges with root vertex K and child vertex are KR with distance 303. We choose KR and mark R as root vertex.

Root vertex set Vs ={D,S,C,K,R}

Child vertex Vc ={}

Travelled way till now: D→S→C→K→R

Distance travelled=1000+303=1303

Our child vertex set is empty. So, we return to the start vertex and complete the tour. Therefore, total distance travelled = 1303+245=1548

And the shortest way to travel five cities is D→S→C→K→R→D

V. Result Analysis

Comparison of results of 5 city problem:

Methods	Result	Number of cities	Steps
Brute force	1548	5	12
NNA	1548	5	5
INNA	1548	5	4

To test optimality of the problem solved section IV, we use an exact algorithm named brute force. As we can see from the table, for this particular problem we get optimal solution with both NNA and INNA. The uncertainty of getting optimal solution using NNA and INNA increases with the increase of the size of the problem. Usually, for n≥20 both NNA and INNA are not effective to get optimal solution.

VI. Conclusion

This paper presents an improved form of Nearest Neighbor Algorithm. Since TSP is a P vs NP type problem, so the computational steps increase exponentially with the increase of the size of the problem. To avoid high time complexity, applying approximate algorithm for a near optimal solution is reasonable. NNA is best algorithm to obtain near optimal solution (sometimes optimal) for small size problems ($n \leq 20$). And INNA makes NNA more understandable and easily implementable for solving problems with equal efficiency and less computational steps.

VII. REFERENCES

1. Emran Islam, Mariam Sultana, Faruque Ahmed "A Tale of Revolution: Discovery and Development of TSP", International Journal of Mathematics Trends and Technology (IJMTT). V57(2):136-139 May 2018.
2. Davendra, Donald, et al. "Chaos driven evolutionary algorithm for the traveling salesman problem." Traveling Salesman Problem, Theory and Applications (2010): 55-70.
3. Gutin, Gregory, and Abraham P. Punnen, eds. The traveling salesman problem and its variations. Vol. 12. Springer Science & Business Media, 2006.
4. Lenstra, Jan Karel, and AHG Rinnooy Kan. "Some simple applications of the travelling salesman problem." Journal of the Operational Research Society 26.4 (1975): 717-733.
5. Punnen, Abraham P. "The traveling salesman problem: Applications, formulations and variations." The traveling salesman problem and its variations. Springer, Boston, MA, 2007. 1-28.
6. Lenstra, Jan Karel. "Clustering a data array and the traveling-salesman problem." Operations Research 22.2 (1974): 413-414.
7. Kizilates, Gzde, and Fidan Nuriyeva. "On the nearest neighbor algorithms for the traveling salesman problem." Advances in Computational Science, Engineering and Information Technology. Springer, Heidelberg, 2013. 111-118.
8. Laporte, Gilbert. "The traveling salesman problem: An overview of exact and approximate algorithms." European Journal of Operational Research 59.2 (1992): 231-247.
9. Johnson, D. S. "Performance guarantees for heuristics." The Traveling Salesman Problem: A. Guided Tour of Combinatorial Optimization (1985): 145-180
10. Golden, Bruce, et al. "Approximate traveling salesman algorithms." Operations research 28.3-part-ii (1980): 694-711.
11. Taiwo, Oloruntoyin Sefiu, et al. "IMPLEMENTATION OF HEURISTICS FOR SOLVING TRAVELLING SALESMAN PROBLEM USING NEAREST NEIGHBOUR ANDNEAREST INSERTIONAPPROACHES." (2013).
12. Source of distances of five cities: www.distanceto.com"