A Nobel Combinatorial Algorithm towards the TSP

Tanmoy Biswas[1], Debashis Roy[2], Arup Kr. Goswami[3], Susobhan Ghosh[4], Saptarshi Naskar[5]
[1]*Department of CS, Syamaprasad College (CU), India*
[2]*Department of CS, Jogesh Ch.Chaudhury College (CU), India*
[3]*Department of CS, Vidyasagar College (CU), India*
[4]*Department of CS, Sarsuna College (CU), India*
[5]*Department of CS, Dinabandhu Mahavidyalaya, Bongaon, India*

Abstract— **In our problem, if each of the vertices has an edge to every other vertex, we have a complete weighted graph. This graph has numerous Hamiltonian circuits, and we are to pick the one that has the smallest sum of distances (or weights). If there are only two vertices then, the problem is trivial, since only one tour is possible. For the three vertices TSP is also trivial. If all links are present then there are nearly (n!) different tours for n vertices TSP. The size of the search space becomes extremely large for large n, so that an exhaustive search is impracticable. The problem has some direct importance, since quite a lot of practical applications can be reduced to TSP. The TSP has become a standard test bed for combinatorial optimization methods which attempt to find near optimum solution to this NP-Complete problem. Several approximation techniques typically based on deterministic or probabilistic search heuristic have been proposed in the literature, including the classical local search algorithms, simulated annealing, threshold accepting, Tabu-search, elastic nets, neural networks, genetic algorithm, and ant colonies. It also has a theoretical importance in complexity theory, since the TSP is one of the classes of "NP Complete" combinatorial problem. NP-Complete problems have the property that no one has found any really efficient way of solving them for large n. They are also known to be equivalent to each other; if we knew how to solve one kind of NP Complete problem we could solve the lot. Holy Grail is to find a solution algorithm that gives a sub optimal solution in a time that has a polynomial variation with the size n of the problem. If we could find a method whose solution time grows polynomially with n then we can apply it to get sub optimal solution in reasonable time. The best the people have been able to do, however, is to solve it in a time that varies exponentially with n. Such algorithm runs out of puff at a certain level of n, more or less independently of computing power. If computation varies as 2n, say, then a thousand fold increases in computing power will only allow us to add another 10 vertices.**

Index Terms: **Combinatorial Algorithm, Optimal Solution, TSP, Sub-optimal.**

INTRODUCTION

Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. In our work we have taken the Travelling Salesperson Problem (TSP) and we have tried to find a solution for the Travelling Salesperson Problem. In our algorithm we have tried to reduce runtime for medium to large instances as we ensured to generate no duplicate Hamiltonian Path and Easy to visualize tours. In our solution we have also tried to minimize the memory usage as we are not storing all the possible Hamiltonian Paths of a graph but the one path with minimum cost.

PROPOSED ALGORITHM

Input: The graph is taken as input.
Output: A Hamiltonian Path from the graph taken as input is the output of this algorithm.

Step1: Arrange all the vertices in the graph in ascending order degree sequence and store in V_n.
Step2: Initially, $V_c \leftarrow \phi$, $V \leftarrow$ {All the vertices in the graph}.
Step3: Choose any vertex v_i having minimum degree such that $v_i \in v_n$ and $v_i \notin v_c$.
Step4: Set, $V_{lc} \leftarrow v_i$ and push v_i into V_c.
Step5: Arrange all the adjacent vertices of V_{lc} with ascending order degree sequence and store in V_{av}.
Step6: Select the lowest degree vertex a from V_{av} such that, $a \in V$ and $a \notin V_c$, $V_i \leftarrow a$.

Step7: If all the vertices of V are in V_c then, it is a Hamiltonian Path (The path covers all the vertices in V_c in-order) and go to Step 8. Else go to Step 4.
Step8: Stop

WORKED OUT EXAMPLE

Let us consider a graph as shown in the figure (1) with seven vertices named V_1, V_2, V_3, V_4, V_5, V_6 and V_7.

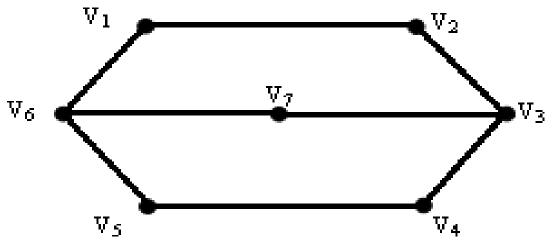


Figure 1

Vertex Set (V)	V_1	V_2	V_3	V_4	V_5	V_6	V_7
Degree	2	2	3	2	2	3	2

Table (1), Vertex set and degree of each vertex.

Sorted Vertex Set (V_n)	V_1	V_2	V_4	V_5	V_7	V_3	V_6
Degree	2	2	2	2	2	3	3

Table (2), Sorted vertex set and degree of each vertex.

The table (3), shows the Hamiltonian Path using our proposed algorithm.

V_{lc} (Current Vertex)	V_c (Hamiltonian Path)	V_{av} (Adjacent Vertices of Current Vertex)	V_i (Next Selected Vertex)
V_1	V_1	V_2, V_6	V_2
V_2	V_1, V_2	V_1, V_3	V_3
V_3	V_1, V_2, V_3	V_2, V_4, V_7	V_4
V_4	V_1, V_2, V_3, V_4	V_5, V_3	V_5
V_5	V_1, V_2, V_3, V_4, V_5	V_4, V_6	V_6
V_6	V_1, V_2, V_3, V_4, V_5, V_6	V_1, V_5, V_7	V_7
V_7	V_1, V_2, V_3, V_4, V_5, V_6, V_7	-	-

Table (3), Figured out Hamiltonian Path using our proposed algorithm

So, we get the Hamiltonian Path as $V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_5 \rightarrow V_6 \rightarrow V_7$.

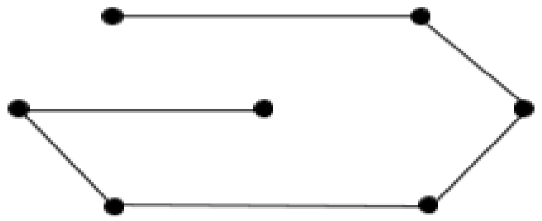


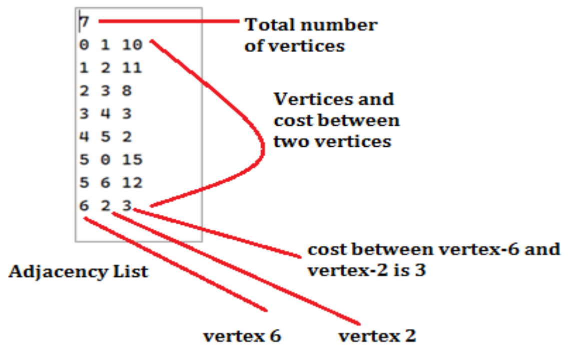
Figure 2 Hamiltonian Path.

This algorithm can be used to find a Hamiltonian Path out of a graph. But our goal is to find all possible Hamiltonian Paths of the graph that has been given and then find out the minimum cost Hamiltonian Path. In the above workout example, we do not have weights of the edges of the graph. But in our practical implementation (Given in the next i.e. result section) we do have weights of each edge. In that case we just need to sum up the costs or weights of all the edges included in the Hamiltonian Path to get the total cost of a path and then we can easily find out the minimum cost path.

We have used our previous defined algorithm in a recursive manner to find out all possible Hamiltonian Paths starting with every vertex of the graph.

RESULT

We have taken input of a graph through an input text file (Shown in Figure (3)) where the integer in the first line specifies the number of vertices (n) in the graph to be numbered from 0 through n – 1. After that each line having three space separated integers represent each edge in the graph where the first two integers specify the two end vertices of the graph and the third one specifies the weight or cost of the edge



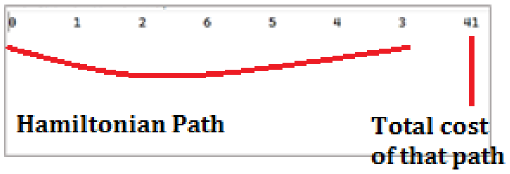
```
7
0 1 10
1 2 11
2 3 8
3 4 3
4 5 2
5 0 15
5 6 12
6 2 3
```

Total number of vertices
Vertices and cost between two vertices
cost between vertex-6 and vertex-2 is 3
Adjacency List
vertex 6
vertex 2

Figure 3 Input Graph

Figure (4) is the graph drawn using the information we have got from the input shown above. There are 7 vertices and 8 edges.

The output is stored in text files. One file contains all possible Hamiltonian Paths (Shown in Figure 6) from the input graph and another file contains the minimum cost Hamiltonian Path (Shown in Figure 5) among them. We have stored all possible Hamiltonian Paths just for demonstration purpose. Each of the output files consist of some space separated values where each line represents a Hamiltonian Path through the input graph by specifying the vertices to be traversed in order and the last integer in each line specifies the cost of to travel that Hamiltonian Path.



```
0    1    2    6    5    4    3    41
```

Figure 4 Graph drawn from the input information

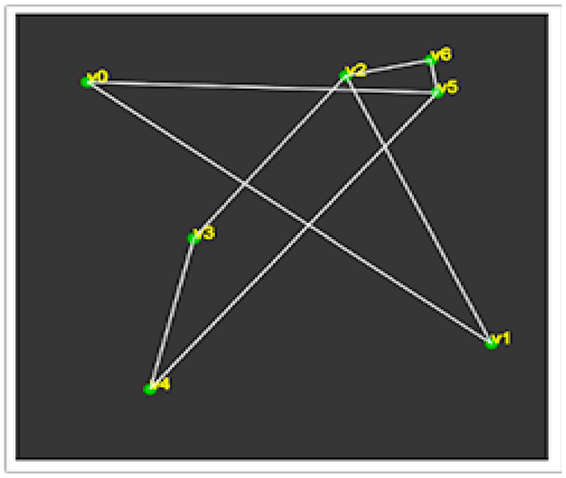
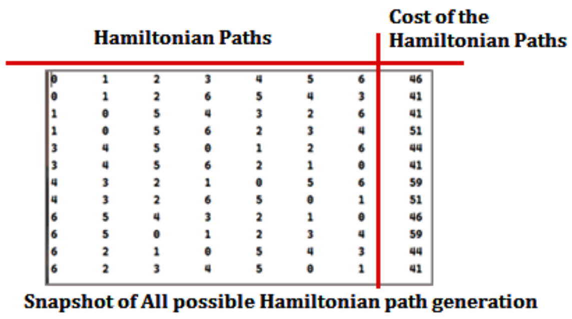


Figure 5 Minimum Cost Hamiltonian Path



```
0    1    2    3    4    5    6    46
0    1    2    6    5    4    3    41
1    0    5    4    3    2    6    41
1    0    5    6    2    3    4    51
3    4    5    0    1    2    6    44
3    4    5    6    2    1    0    41
4    3    2    1    0    5    6    59
4    3    2    6    5    0    1    51
6    5    4    3    2    1    0    46
6    5    0    1    2    3    4    59
6    2    1    0    5    4    3    44
6    2    3    4    5    0    1    41
```

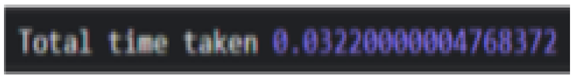
Hamiltonian Paths
Cost of the Hamiltonian Paths
Snapshot of All possible Hamiltonian path generation

Figure (6), All possible Hamiltonian Paths with cost.

The output is recorded in a machine with following specifications

Processor	AMD Ryzen 3 2200U
Ram	4GB
Storage	1TB HDD
Operating System	64-bit Windows OS (x64-based processor)

Table (4), Specifications of the computing machine used.



```
Total time taken 0.03220000004768372
```

Figure 7Time taken for implementation (Unit-seconds)

CONCLUSION

Our proposed algorithm is both time and space efficient as it is not storing all the Hamiltonian Paths of a graph but the minimum cost path and it does not generate any duplicate path. Our algorithm can find its application in several fields including BPO, Vehicle Routing, Bus Scheduling, Development of flight schedules, Crew Scheduling, Order-picking problem in warehouses, Printing press scheduling problem, Network cabling in a country, Computer wiring, Query workload ordering for optimization, Medical Science, Statistical Computation, Virology, Microbiology, Biotechnology, Genetic Engineering, VLSI chip design connectivity layout, Drilling of printed circuit boards, Hot rolling scheduling problem in Iron & Steel Industry, Overhauling gas turbine engines (Ordering nozzle guides), X-Ray crystallography (Ordering positions for measurement), Global navigation satellite system, Ordering test cases in regression suite to re-use components and many more.

REFERENCES

[1] Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J. (2006), The Traveling Salesman Problem, ISBN 978-0-691-12993-8

[2] Arora, Sanjeev (1998), "Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems" (PDF), Journal of the ACM, 45 (5): 753–782, doi:10.1145/290179.290180, MR 1668147, S2CID 3023351

[3] Bellman, R. (1960), "Combinatorial Processes and Dynamic Programming", in Bellman, R.; Hall, M. Jr. (eds.), Combinatorial Analysis,

Proceedings of Symposia in Applied Mathematics 10, American Mathematical Society, pp. 217–249.

[4] Bellman, R. (1962), "Dynamic Programming Treatment of the Travelling Salesman Problem", J. Assoc. Comput. Mach., 9: 61–63, doi:10.1145/321105.321111

[5] Kaplan, H.; Lewenstein, L.; Shafrir, N.; Sviridenko, M. (2004), "Approximation Algorithms for Asymmetric TSP by Decomposing Directed Regular Multigraphs", In Proc. 44th IEEE Symp. on Foundations of Comput. Sci, pp. 56–65

[6] Papadimitriou, Christos H. (1977), "The Euclidean traveling salesman problem is NP-complete", Theoretical Computer Science, 4 (3): 237–244, doi:10.1016/0304-3975(77)90012-3, MR 0455550

[7] Serdyukov, A. I. (1984), "An algorithm with an estimate for the traveling salesman problem of the maximum'", Upravlyaemye Sistemy, 25: 80–86.

[8] Woeginger, G.J. (2003), "Exact Algorithms for NP-Hard Problems: A Survey", Combinatorial Optimization – Eureka, You Shrink! Lecture notes in computer science, vol. 2570, Springer, pp. 185–2007