A HEURISTIC ALGORITHM FOR TSP OF COMPLETE GRAPH

Kanak Chandra Bora
Department of Computer Science & Engineering
Scholar of Gauhati University
Jalukbari, Guwahati-781014, Assam, India

Abstract*: In this paper, a heuristic algorithm has been developed for complete graph having vertices greater than equal to 4 using the concept of perfect matching.*

Keywords: Hamiltonian Cycle, Heuristic, Complete graph, Perfect Matching,

Section-1

Introduction

Travelling Sales man problem is a known NP- complete problem[1-4]. It is a popular problem of graph theory which has many applicationin different field of engineering like in networking. Extensive research have been done and numerous result have been reported.. There are many heuristic algorithms are there like greedy algorithm, Prim algorithm with triangle inequality [4].

Kalita B.[5] studied the complete graph K_{2m+2}, for $m \geq 2$ and reported some results regarding number of non-isomorphic Hamiltonian sub graph and the structure of metal atom cluster compound of chemistry. He also forwarded an algorithm for solving travelling sales man of the complete graph K_{2m+2}, for $m \geq 2$ under different cases. AnupamDutta et-al[6] studied the same graph and supplied some theories related to perfect matching and also developed an alogorithm for the same. AnupmDutta et-al[7], again introduced some theories related to planarity, bipartite and coloring and forwarded one more algorithm repeated weight. Jayanta Kr. Choudhury[8] studied the traveling sales man problem for K_{2m+3}, and developed an algorithm for different cases like 2m + 2 consecutive greatest weights are incident with a vertex. Jayanta Kr. Choudhury et-al[9] studied different types of graph factorization of the complete graphs K_{6m+2}, K_{6m-2} and $K_{6m}$and developed some theories related to 3-factors, 2-factors and 1-factores and also forwarded an algorithm for solving TSP using the concept of factorization. The decomposition of complete graph K_{2m+1}, for $m \geq 2$ into circulant graphs has been discussed by Choudhury J. K [10]. An algorithm has also been developed for the complete graph K_{2m+1}, for $m \geq 2$ under different situations when the weight of edges are non-repeated for the traveling salesman problem. Again Chodhury, J. K. and Kalita, B., discussed the generation the generation of maximal triangle free graph from the complete graph K_{2m+3}, for $m \geq 2$ and an algorithm was also developed under different cases to solve the traveling salesman problem of the said complete graph [11].

This paper is organized in three sections, in section-1, introduction including previous work, in section-2, a new heuristic algorithm for solving Traveling Sales Man Problem of complete graph has been developed using the concept of perfect matching. And in section-3, experimental results are cited.

Section-2

When the number of vertices of the complete graph is even, there is perfect matching and when the number of vertices of a complete graph is odd, there is no perfect matching.

Algorithm:

Input: Complete weighted graph with non-repeated weight and number of vertices ≥ 4.

Out Put: Least Cost Route.

Case 1: when number of vertices is even.

Step 1: Arrange the edges as per weight in ascending order and named the list as list-1..

Step 2: Find the perfect matching of the graph considering the ordered list-1. Selection of edges for perfect matching is done from lower to higher weight edges one after another from the list-1. First least weight edge is selected and the process is repeated until n/2 edges are selected, where n is the number of vertices. One edge is selected for the perfect matching set if selection of this edge does not make degree 2 of any vertex with previously selected edges. This perfect matching group is named as list-2.

Step 3: A new ordered edge list is obtained after deleting the list-2 from the list-1 and the new list is named as list-3. The list-3 is also in ascending order.

Step 4: From lower to higher one after another edge is considered from the list-3 and it is tested whether this edge connection to the list-2, will make degree of any vertex greater than 2& also it is tested for circuit formation of length less than n. If connection of this edge makes degree of any vertex greater than 2 or circuit formation of length less than n, then this edge is discarded, otherwise this edge is connected to the list-2. This process is repeated until n/2 edges are selected at this step.

Step 5: The route formed by the edges of the list-2, obtained at step 4 is the least cost route.

Step 6: Stop.

Case 2: when number of vertices is odd.

Step 7: Each vertex total weight will be calculated by adding weight of edges incident to a vertex.

Step 8: Arrange the edges in ascending order and named the list as list-4.

Step 9: Removing the highest weighted vertex from the graph we have a sub graph of n-1 vertices and arrange the edges of this sub graph in ascending order and named the list as list-5.

Step 10: Find the perfect matching of the ordered list-5 for the sub graph obtained at step-9. Selection of edges for perfect matching is done from lower to higher weight edges one after another. First least weight edge is selected and the process is repeated until half numbers of edges are selected. One edge is selected for the perfect matching set if selection of this edge does not make degree 2 of any vertex with previously selected edges. This perfect matching group is named as list-6.

Step 11: The lowest weighted edge incident with highest weighted vertex is added to the list-6. This edge group is called as list-7.

Step 12: Find a new ordered list of edges in ascending order as list-8 after deleting the list-7 from the list-4.

Step 13: From lower to higher one after another edge is considered from the ordered list-8, obtained at step 12 and it is tested whether this edge connection to the list-7 will make degree of any vertex greater than 2& also it is tested for circuit formation of length less than n where n is the total number of vertices in the graph. If connection of this edge to the list-7 makes degree of any vertex greater than 2 or circuit formation of length less than n, then this edge is discarded, otherwise this edge is connected to the list-7. This process is repeated until floor (n/2) edges are selected at this step.

Step 14. The route formed by the edges of the list-7, obtained at step 13 is the least cost route.

Step 15: Go to step 6.

Section-3

EXPERIMENTAL RESULT:

Example -1: The following cost matrix of a traveler for six cities is considered in table-1 below.

Table-1

	A	B	C	D	E	F
A	-	11	18	7	40	9
B	11	-	12	28	19	33
C	18	12	-	26	20	22
D	7	28	26	-	25	30
E	40	19	20	25	-	16
F	9	33	22	30	16	-

As per step-1 of the algorithm of case 1, ascending order of edges of the given graph are

(A, D), (A, F), (A, B), (B, C), (E, F), (A, C), (B, E), (C, E), (C, F), (D, E), (C, D), (B, D), (D, F), (B, F), (A, E). The name of this ordered list is list-1.

Now applying the step-2 of the algorithm we have the perfect matching edge group as follows.

(A, D), (B, C), (E, F). The name of this edge group is list-2.

As per step-3 of the algorithm we have the ordered list-3 as follows.

(A, F), (A, B), (A, C), (B, E), (C, E), (C, F), (D, E), (C, D), (B, D), (D, F), (B, F), (A, E).

Now applying the step-4 of the algorithm, the edges in the list-2 will be as follows.

(A, D), (B, C), (E, F), (A, F), (B, E), (C, D).

As per step 5 of the algorithm, the least cost route of graph will be A → D → C → B → E → F → A with cost 89.

Example 2: The following cost matrix of a traveler for seven cities is considered as shown in table-2.

Table-2

	A	B	C	D	E	F	G
A	-	25	17	19	18	20	21
B	25	-	8	7	9	15	16
C	17	8	-	2	12	11	5
D	19	7	2	-	10	6	14
E	18	9	12	10	-	13	3
F	20	15	11	6	13	-	1
G	21	16	5	14	3	1	-

Since the number of vertices of the considered graph is odd and hence the total weight of each vertex is calculated as per step 7 below in table-3.

Table-3

	A	B	C	D	E	F	G	Total
A	-	25	17	19	18	20	21	120
B	25	-	8	7	9	15	16	80
C	17	8	-	2	12	11	5	55
D	19	7	2	-	10	6	14	58
E	18	9	12	10	-	13	3	65
F	20	15	11	6	13	-	1	66
G	21	16	5	14	3	1	-	60

As per step-8, the edges are arranged in ascending order as follows.

(G, F), (C, D), (G, E), (C, G), (D, F), (B, D), (B, C), (B, E), (C, F), (C, E), (E, F), (D,G), (B, F), (B, G), (A, C), (A, E), (A, D), (A, F), (A, G), (A, B). The name of this order is list-4.

As per step-9, removing the highest weighted vertex A from the graph, the edges of the sub graph is arranged in ascending order as follows.

(G, F), (C, D), (G, E), (C, G), (D, F), (B, D), (B, C), (B, E), (C, F), (C, E), (E, F), (D, G),, (B, F), (B, G) . The name of this order is list-5.

Now according to step-10 of the algorithm, perfect matching of the sub graph considering the list-5 will be as follows.

(G, F), (C, D), (B, E).. The name of the list is list-6.

As per step-11 of the algorithm, the lowest cost edge (A, C) incident with the highest weighted vertex A, is added to the ordered list-6 as follows.

(G, F), (C, D), (B, E), (A, C). The name of the list is list-7.

As per step-12 of the algorithm, list-8 is obtained after deleting the edges of list-7 from list-4 as below.

(G, E), (C, G), (D, F), (B, D), (B, C), (C, F), (C, E), (E, F), (D, G), (B, F), (B, G), (A, E), (A, D), (A, F), (A, G), (A, B). The name of this order is list-8.

Applying the step-13 of the algorithm, the edges of the list-7 will be as follows.

(G, F), (C, D), (B, E), (A, C), (G, E), (D, F), (A, B) The name of the list is list-7.

As per step-14, the least cost route is formed with the edges of the list-7 is as follows.

A →B → E → G → F → D → C → A, with weight equal to 63.

Example 3: The following cost matrix of a traveler for eleven cities is considered in table-4 below.

Table-4

	A	B	C	D	E	F	G	H	I	J	K
A	-	10	95	11	91	18	19	28	33	42	44
B	10	-	20	85	21	101	24	29	34	43	46
C	95	20	-	30	75	31	12	22	32	36	47
D	11	85	30	-	40	65	41	13	23	37	39
E	91	21	75	40	-	50	55	51	14	26	38
F	18	101	31	65	50	-	60	45	61	16	27
G	19	24	12	41	55	60	-	70	35	71	17
H	28	29	22	13	51	45	70	-	80	25	81
I	33	34	32	23	14	61	35	80	-	90	15
J	42	43	36	37	26	16	71	25	90	-	100
K	44	46	47	39	38	27	17	81	15	100	-

Since the number of vertices of the considered graph is odd and hence the total weight of each vertex is calculated as per step 7 below in table-5.

Table-5

	A	B	C	D	E	F	G	H	I	J	K	Total
A	-	10	95	11	91	18	19	28	33	42	44	391
B	10	-	20	85	21	101	24	29	34	43	46	413
C	95	20	-	30	75	31	12	22	32	36	47	400
D	11	85	30	-	40	65	41	13	23	37	39	384
E	91	21	75	40	-	50	55	51	14	26	38	461
F	18	101	31	65	50	-	60	45	61	16	27	474
G	19	24	12	41	55	60	-	70	35	71	17	404
H	28	29	22	13	51	45	70	-	80	25	81	444
I	33	34	32	23	14	61	35	80	-	90	15	417
J	42	43	36	37	26	16	71	25	90	-	100	486
K	44	46	47	39	38	27	17	81	15	100	-	454

As per step-8, the edges are arranged in ascending order as follows .(A, B), (A, D), (C, G), (D, H), (E, I), (I, K), (F, J), (G, K), (A, F), (A, G), (B, C), (B, E), (C, H), (D, I), (B, G), (H, J), (E, J), (F, K), (A, H), (B, H), (C, D), (C, F), (C, I), (A, I), (B, I), (G, I), (C, J), (D, J), (E, K), (D, K), (D, E), (D, G), (A, J), (B, J), (A, K), (F, H), (B, K), (C, K), (E, F), (E, H), (E, G), (F, G), (F, I), (D, F), (G, H), (G, J), (C, E), (H, I), (H, K), (B, D), (I, J), (A, E), (A, C), (J, K), (B, F), . The name of this order is list-4.

As per step-9, removing the highest weighted vertex J from the graph, the edges of the sub graph is arranged in ascending order as follows.

(A, B), (A, D), (C, G), (D, H), (E, I), (I, K), (G, K), (A, F), (A, G), (B, C), (B, E), (C, H), (D, I), (B, G), (F, K), (A, H), (B, H), (C, D), (C, F), (C, I), (A, I), (B, I) , (G, I), (E, K), (D, K), (D, E), (D, G), (A, K), (F, H), (B, K), (C, K), (E, F), (E, H), (E, G), (F, G), (F, I), (D, F), (G, H), (C, E), (H, I), (H, K), (B, D), (A, E), (A, C), (B, F). The name of this order is list-5.

Now according to step-10 of the algorithm, perfect matching of the sub graph considering the list-5 will be as follows.

(A, B), (C, G), (D, H), (E, I), (F, K). The name of the list is list-6.

As per step-11 of the algorithm, the lowest cost edge (F, J) incident with the highest weighted vertex J, is added to the ordered list-6 as follows.

(A, B), (C, G), (D, H), (E, I), (F, K),. (J, F). The name of the list is list-7.

As per step-12 of the algorithm, list-8 is obtained after deleting the edges of list-7 from list-4 as below.

(A, D), (I, K), (G, K), (A, F), (A, G), (B, C), (B, E), (C, H), (D, I), (B, G), (H, J), (E, J), (A, H), (B, H), (C, D), (C, F), (C, I), (A, I), (B, I) , (G, I), (C, J), (D, J), (E, K), (D, K), (D, E), (D, G), (A, J), (B, J), (A, K), (F, H), (B, K), (C, K), (E, F), (E, H), (E, G), (F, G), (F, I), (D, F), (G, H), (G, J), (C, E), (H, I), (H, K), (B, D), (I, J), (A, E), (A, C), (J, K), (B, F). The name of the list is list-8.

Applying the step-13 of the algorithm, the edges of the list-7 will be as follows.

(A, B), (C, G), (D, H), (E, I), (F, K),. (J, F), (A, D), (I, K), (B, C), (J, H), (E, G). The name of the list is list-7.

As per step-14, the least cost route is formed with the edges of the list-7 is as follows.

A →B → C → G → E → I → K → F → J → H → D → A, with weight equal to 218.

Conclusion: The algorithm developed here is a heuristic algorithm. Existing algorithm discussed above are for solving TSP of complete graph of even number of vertices and some one for odd number of vertices. The new algorithm is for both even and odd number of vertices of complete graph.

REFERENCES

1. Bora, K. C. and Kalita, B. , (2013), "Exact Polynomial –time Algorithm for the Clique Problem and P = NP for Clique Problem", *International Journal of Computer Application, 73* (8), 19-23.

2. Bora, K. C., Kalita, B. (2014)," Particular Type of Hamiltonian Graphs and Their Properties", *International Journal of Computer Application, 96* (3), 31-36.

3. Bora, K. C., and Kalita, B. (2014), "A Heuristics Algorithm to Solve TSP for a Particular Type of Hamiltonian Graph", *GE-International Journal of Computer Application, 2* (4), 125-151.

4. Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009), "*Introduction to ALGORITHMS*", Second Edition, Eastern Economy Edition, PHI Learning Private Limited, pp. 1012-1032.

5. Kalita, B. (2006),"Sub-graphs of complete graph" *Proceeding, International conference on Foundation of Computer Science*, Lasvegas, U. S. A, pp-71-77.

6. Dutta, A., Kalita, B., Baurah, H. K., (2010), "Regular sub-graph of Complete Graph", *International Journal of Applied Engineering Research, 5* (8), 1315-1323.

7. Dutta, A., Kalita, B., Baruah, H., K., (2010), "Regular Planar sub-graphs of Complete Graph and their Application", *International Journal of Applied Engineering Research, 5* (3) 377-386.

8. Choudhury, J. K., Klita, B., (2011), "An Algorithm for Traveling Salesman Problem", *Bulletin of Pure and Applied Sciences, 30 E* (1) 111-118.

9. Choudhury, J. K., Kalita, B. (2012), "Graph Factorization and it's Application", *International Journal of Management, IT and Engineering, 2* (3) 208-220.

10. Choudhury, J. K.., Dutta, A,. Kalita, B. (2013), "Decomposition of Complete Graph into Circulant Graphs and Its Application", *International Journal of Computer Engineering and Technology, 4*(6) 25-47.

11. Choudhury J. K., and Kalita, B. (2014), "Maximal Triangle Free Graphs and Traveling Salesman Problem", *International Journal of Computer Application, 85* (17) 22-30.