International Journal of Computer Applications (0975 – 8887)
Volume 115 – No. 8, April 2015

A Note on Computational Approach to Travelling Sales Man Problem

Shaik. Mohiddin Shaw
Narasaraopeta Engineering College,
Narasaraopet

Dharmaiah Gurram
Narasarao PET Engineering College,
Narasaraopet

ABSTRACT
Many real life situations for which there are no optimization algorithms which can solve polynomial time problems in the worst case. So researchers are trying for new approximation algorithms for such kinds of situations. Approximation algorithms give the solution which is close to the optimal solution of a particular situation. Traveling Salesman Problem (TSP) is a typical NP complete problem which lacks polynomial time algorithm. In this paper it is proposed an edge removal algorithm, which will give the nearly optimal solution within a limited time.

General Terms
Optimization, Travelling Sales man Problem

Keywords
Edge Removal Algorithm, Compression Algorithm, Back Tracking.

1. INTRODUCTION
Traveling Salesman Problem (TSP) is a famous NP hard problem and also a typical combinatorial optimization problem in Operation Research. With the increasing of number of cities, its solving time complexity grows rapidly in exponential degree, so enumerating each possible route and searching for the one with the smallest cost to optimally solve this problem becomes impossible in polynomial time. In the classical traveling salesman problem, a set of cities has to be visited in a single tour with the objective of minimizing the total length of the tour. This is one of the most studied problems in combinatorial optimization, together with its dozens of variations. In the asymmetric version of the problem, the distance from one point to another in a given space can be different from the inverse distance. This variation, known as the Asymmetric Traveling Salesman Problem (ATSP) arises in many applications; for example, one can think of a delivery vehicle traveling through one-way streets in a city, or of gasoline costs when traveling through mountain roads.

2. THE TRAVELING SALESMAN PROBLEM
The idea of the traveling salesman problem (TSP) is to find a tour of a given number of cities, visiting each cityexactly once and returning to the starting city where the length of this tour is minimized. TSP is of great significance in practical applications, it can be used to resolve the problems in allocation, path problem and vehicle scheduling problem and so on. The standard symmetric traveling salesman problem can be stated mathematically as follows:

Given a weighted graph G = (V, E), where V is the set of nodes, E is the set of edges, and the weight c_{ij} on the edge between nodes *i* and *j* is a non-negative value, finding the tour of all nodes that has the minimum total cost.

3. EDGE REMOVAL ALGORITHM
Algorithm:
Step 1: Read the Adjacency Matrix, (distances between different towns).

Step 2: Apply compression algorithm on the above adjacency matrix, to remove edges and find cost tables.

Step 3: Sort the edges related to each node in ascending order.

Step 4: calculate the minimum cost array to improve the efficiency of this algorithm.

Step 5: Apply backtracking on the sorted cost tables obtained in step 3 using step 4 as an optimization step.

3.1 Example:
Step 1: Input:

0	2	8	15	1	10	5	19	19	3
5	0	6	6	2	8	2	12	16	3
8	17	0	12	5	3	14	13	3	2
17	19	16	0	8	7	12	19	10	13
8	20	16	15	0	4	12	3	14	14
5	2	12	14	9	0	8	5	3	18
18	20	4	2	10	19	0	17	16	11
3	9	7	1	3	5	9	0	7	6
11	10	11	11	7	2	14	9	0	10
4	5	15	17	1	7	17	12	9	0

Step 2: Compression Step:
In this step remove these edges that are no need to visit to find the optimal solution.

By using single source shortest path algorithms it can be find the shortest paths from a node to all remaining nodes. For example, consider node number 1, the paths from node 1 to all other nodes is like below.

Table1:

Source Node	Destination Node	Minimum Path	Node to be visited first	Cost
1	2	(1 2)	2	2
1	3	(1 3)	3	8
1	4	(1 5 8 4)	5	5

1	5	(1 5)	5	1
1	6	(1 5 6)	5	5
1	7	(1 2 7)	2	4
1	8	(1 5 8)	5	4
1	9	(1 5 6 9)	5	8
1	10	(1 10)	10	3

In the next case extract the fields' node to be visited first and cost where those values in the 2nd and 4th column of the above table are equal. That is, nodes extracted for the above table are 2, 3, 5, 10.

Now the cost tables are generated for above extracted nodes and their costs.

Table 2: Cost table for node 1.

Node Number	Cost
2	2
3	8
5	1
10	3

Cost tables for all the other nodes are given below:

Table 3: Cost Table for node 2.

Node Number	Cost
1	5
3	6
5	2
7	2
10	3

Table 4: Cost Table for Node 3.

Node Number	Cost
6	3
9	3
10	2

Table 5: Cost table for Node 4.

Node Number	Cost
5	8
6	7
9	10

Table 6: Cost Table for Node 5.

Node Number	Cost
6	4
8	3

Table 7: Cost Table for Node 6.

Node Number	Cost
1	5
2	2
8	5
9	3

Table 8: Cost Table for Node 7

Node Number	Cost
3	4
4	2

Table 9: Cost Table for Node 8.

Node Number	Cost
1	3
3	7
4	1
5	3
6	5
9	7
10	6

Table 10: Cost table for Node 9.

Node Number	Cost
6	2

Table 11: Cost table for Node 10.

Node Number	Cost
1	4
2	5
5	1

Step 3: Sort the above tables based on their ascending order of their costs.

Table 12: Sorted Cost table for Node 1.

Node Number	Cost
5	1
2	2
10	3
3	8

Cost tables for all the other nodes are given below:

Table 13: Sorted Cost Table for node 2.

Node Number	Cost
5	2
7	2
10	3
1	5
3	6

Table 14: Sorted Cost Table for Node 3.

Node Number	Cost
10	2
6	3
9	3

Table 15: Sorted Cost table for Node 4.

Node Number	Cost
6	7
5	8
9	10

Table 16: Sorted Cost Table for Node 5.

Node Number	Cost
8	3
6	4

Table 17: Sorted Cost Table for Node 6.

Node Number	Cost
2	2
9	3
1	5
8	5

Table 18: Sorted Cost Table for Node 7

Node Number	Cost
4	2
3	4

Table 19: Sorted Cost Table for Node 8.

Node Number	Cost
4	1
1	3
5	3
6	5
10	6
3	7
9	7

Table 20: Sorted Cost table for Node 9.

Node Number	Cost
6	2

Table 21: Sorted Cost table for Node 10.

Node Number	Cost
5	1
1	4
2	5

Step 4: Now calculate the minimum cost array
The minimum_cost_array size is equal to the number of nodes. So,

minimum_cost_array[1]

= Sorted_Cost_Table[1][2] = 1

Minimum_Cost_Array[2]

= minimum_Cost_Array[1] + Sorted_Cost_Table[2][2]

= 1 + 2 = 3.

Minimum_Cost_Array[3]

= minimum_Cost_Array[2] + Sorted_Cost_Table[3][2]

= 3 + 2 = 5.

Minimum_Cost_Array[4]

= minimum_Cost_Array[3] + Sorted_Cost_Table[4][2]

= 5 + 7 = 12.

Minimum_Cost_Array[5]

= minimum_Cost_Array[4] + Sorted_Cost_Table[5][2]

= 12 + 3 = 15.

Minimum_Cost_Array[6]

= minimum_Cost_Array[5] + Sorted_Cost_Table[6][2]

= 15 + 2 = 17.

Minimum_Cost_Array[7]

= minimum_Cost_Array[6] + Sorted_Cost_Table[7][2]

= 17 + 2 = 19.

Minimum_Cost_Array[8]

= minimum_Cost_Array[7] + Sorted_Cost_Table[8][2]

= 19 + 1 = 20.

Minimum_Cost_Array[9]

= minimum_Cost_Array[8] + Sorted_Cost_Table[9][2]

= 20 + 2 = 22.

Minimum_Cost_Array[10]

= minimum_Cost_Array[9] + Sorted_Cost_Table[10][2]

= 22 + 1 = 23.

So, Minimum_Cost_Array[ ] =

{ 1, 3, 5, 12, 15, 17, 19, 20, 22, 23 }.

Step 5: Apply Backtracking step to the above cost Tables generated in Step 4:

The optimal tour is 1 5 8 4 9 6 2 7 3 10 1

Use of Step 4:

If you got a tour already, (and tour length is (say **OTL**))

For example, if there are '***N***' cities , and the number of cities that to visited already is '***X***' , and the distance travelled already is '***D***', Then it should travel atleast a distance of **Minimum_Cost_Array[n-x+1]** to get the next tour. So least possible distance for this tour is **D + Minimum_Cost_Array[n-x+1] .**

And it should satisy the condition " **D + Minimum_Cost_Array[n-x+1] < OTL " ,** then only it can say the path (tour) that are travelling now can be a minimum tour than the earlier tour **OTL.** If above condition is failed to satisfy then skip the combinations to improve efficiency.

But, in some cases if the number of edges removed are very less, that mean the cost table size is high then this algorithm won't give the optimal result, in that situation it is better to go for nearly optimal solution.

3.2 Example:

Consider this 24 city problem:

The Adjacency Matrix for this problem is shown on separate sheet in the next page.

But to calculate the optimal path from normal backtracking method need 23! combinations, tht is, 25852016738884976640000 combinations have to be performed, so that it go for the nearly optimal solution. This approach is efficient to find the nearly optimal solution.

The Optimal solution for the problem is

16 11 3 7 6 24 8 21 5 10 17 22 18 19 15 2 20 14 13 9 23 4 12 1 Cost 1272

But to get the solution in a polynomial time is not possible, so by giving time bound of 15 seconds to Edge Removal Algorithm, will get the nearly optimal cost of 1316. This algorithm is checked against several samples of real world data and the results are promising to be nearly optimal. All the results shows this algorithm give the nearly optimal solution less than the 2 times of optimal solution in all possible cases.

The adjacency matrix [2.2] :

$$
\begin{bmatrix}
0 & 257 & 187 & 91 & 150 & 80 & 130 & 134 & 243 & 185 & 214 & 70 & 272 & 219 & 293 & 54 & 211 & 290 & 268 & 261 & 175 & 25 & 192 & 121 \\
257 & 0 & 196 & 228 & 112 & 196 & 167 & 154 & 209 & 86 & 223 & 191 & 180 & 83 & 50 & 219 & 74 & 139 & 53 & 43 & 128 & 99 & 228 & 142 \\
187 & 196 & 0 & 158 & 96 & 88 & 59 & 63 & 286 & 124 & 49 & 121 & 315 & 172 & 232 & 92 & 81 & 98 & 138 & 200 & 76 & 89 & 235 & 99 \\
91 & 228 & 158 & 0 & 120 & 77 & 101 & 105 & 159 & 156 & 185 & 27 & 188 & 149 & 264 & 82 & 182 & 261 & 239 & 232 & 146 & 221 & 108 & 84 \\
150 & 112 & 96 & 120 & 0 & 63 & 56 & 34 & 190 & 40 & 123 & 83 & 193 & 79 & 148 & 119 & 105 & 144 & 123 & 98 & 32 & 105 & 119 & 35 \\
80 & 196 & 88 & 77 & 63 & 0 & 25 & 29 & 216 & 124 & 115 & 47 & 245 & 139 & 232 & 31 & 150 & 176 & 207 & 200 & 76 & 189 & 165 & 29 \\
130 & 167 & 59 & 101 & 56 & 25 & 0 & 22 & 229 & 95 & 86 & 64 & 258 & 134 & 203 & 43 & 121 & 164 & 178 & 171 & 47 & 160 & 178 & 42 \\
134 & 154 & 63 & 105 & 34 & 29 & 22 & 0 & 225 & 82 & 90 & 68 & 228 & 112 & 190 & 58 & 108 & 136 & 165 & 131 & 30 & 147 & 154 & 36 \\
243 & 209 & 286 & 159 & 190 & 216 & 229 & 225 & 0 & 207 & 313 & 173 & 29 & 126 & 248 & 238 & 310 & 389 & 367 & 166 & 222 & 349 & 71 & 220 \\
185 & 86 & 124 & 156 & 40 & 124 & 95 & 82 & 207 & 0 & 151 & 119 & 159 & 62 & 122 & 147 & 37 & 116 & 86 & 90 & 56 & 76 & 136 & 70 \\
214 & 223 & 49 & 185 & 123 & 115 & 86 & 90 & 313 & 151 & 0 & 148 & 342 & 199 & 259 & 84 & 160 & 147 & 187 & 227 & 103 & 138 & 262 & 126 \\
70 & 191 & 121 & 27 & 83 & 47 & 64 & 68 & 173 & 119 & 148 & 0 & 209 & 153 & 227 & 53 & 145 & 224 & 202 & 195 & 109 & 184 & 110 & 55 \\
272 & 180 & 315 & 188 & 193 & 245 & 258 & 228 & 29 & 159 & 342 & 209 & 0 & 97 & 219 & 267 & 196 & 275 & 227 & 137 & 225 & 235 & 74 & 249 \\
219 & 83 & 172 & 149 & 79 & 139 & 134 & 112 & 126 & 62 & 199 & 153 & 97 & 0 & 134 & 170 & 99 & 178 & 130 & 69 & 104 & 138 & 96 & 104 \\
293 & 50 & 232 & 264 & 148 & 232 & 203 & 190 & 248 & 122 & 259 & 227 & 219 & 134 & 0 & 255 & 125 & 154 & 68 & 82 & 164 & 114 & 264 & 178 \\
54 & 219 & 92 & 82 & 119 & 31 & 42 & 58 & 238 & 147 & 84 & 53 & 267 & 170 & 255 & 0 & 173 & 190 & 230 & 223 & 99 & 212 & 187 & 60 \\
211 & 74 & 81 & 182 & 105 & 150 & 121 & 108 & 310 & 37 & 160 & 145 & 196 & 99 & 125 & 173 & 0 & 79 & 57 & 90 & 57 & 39 & 182 & 96 \\
290 & 139 & 98 & 261 & 144 & 176 & 164 & 136 & 389 & 116 & 147 & 224 & 275 & 178 & 154 & 190 & 79 & 0 & 86 & 176 & 112 & 40 & 261 & 175 \\
268 & 53 & 138 & 239 & 123 & 207 & 178 & 165 & 367 & 86 & 187 & 202 & 227 & 130 & 68 & 230 & 57 & 86 & 0 & 90 & 114 & 46 & 239 & 153 \\
261 & 43 & 200 & 232 & 98 & 200 & 171 & 131 & 166 & 90 & 227 & 195 & 137 & 69 & 82 & 223 & 90 & 176 & 90 & 0 & 134 & 136 & 165 & 146 \\
175 & 128 & 76 & 146 & 32 & 76 & 47 & 30 & 222 & 56 & 103 & 109 & 225 & 104 & 164 & 99 & 57 & 112 & 114 & 134 & 0 & 96 & 151 & 47 \\
250 & 99 & 89 & 221 & 105 & 189 & 160 & 147 & 349 & 76 & 138 & 184 & 235 & 138 & 114 & 212 & 39 & 40 & 46 & 136 & 96 & 0 & 221 & 135 \\
192 & 228 & 235 & 108 & 119 & 165 & 178 & 154 & 71 & 136 & 262 & 110 & 74 & 96 & 264 & 187 & 182 & 261 & 239 & 165 & 151 & 221 & 0 & 169 \\
121 & 142 & 99 & 84 & 35 & 29 & 42 & 36 & 220 & 70 & 126 & 55 & 249 & 104 & 178 & 60 & 96 & 175 & 153 & 146 & 47 & 135 & 169 & 0
\end{bmatrix}
$$

The nearly optimal solutions found based on this algorithm within 15 seconds is given below:

16 6 7 8 21 5 24 12 4 1 9 13 23 14 10 17 19 2 20 15 22 18 3 11	Cost 1509
16 6 7 8 21 5 24 12 4 1 9 13 23 14 10 17 2 20 15 19 22 18 3 11	Cost 1473
16 6 7 8 21 5 24 12 4 1 9 13 23 14 10 17 20 2 15 19 22 18 3 11	Cost 1457
16 6 7 8 21 5 24 12 4 1 9 13 23 14 20 2 15 19 10 17 22 18 3 11	Cost 1453
16 6 7 8 21 5 24 12 1 4 23 9 13 14 10 17 22 19 2 20 15 18 3 11	Cost 1443
16 6 7 8 21 5 24 12 1 4 23 9 13 14 10 17 19 2 20 15 22 18 3 11	Cost 1415
16 6 7 8 21 5 24 12 1 4 23 9 13 14 10 17 2 20 15 19 22 18 3 11	Cost 1379
16 6 7 8 21 5 24 12 1 4 23 9 13 14 10 17 20 2 15 19 22 18 3 11	Cost 1363
16 6 7 8 21 5 24 12 1 4 23 9 13 14 20 2 15 19 10 17 22 18 3 11	Cost 1359
16 6 7 8 21 5 24 10 17 3 11 18 22 19 2 15 20 14 13 9 23 4 12 1	Cost 1355
16 6 7 8 21 5 24 10 17 3 11 18 22 19 15 2 20 14 13 9 23 4 12 1	Cost 1331
16 6 7 8 21 24 5 10 17 3 11 18 22 19 15 2 20 14 13 9 23 4 12 1	Cost 1316

4. CONCLUSION

This paper proposed a novel algorithm to get nearly optimal solution to the Travelling Sales man problem by taking time bound as a limit. This algorithm uses single source shortest path algorithm to remove unnecessary edges and by using compression algorithm. This algorithm is tested against several real world data and results are promising.

5. ACKNOWLEDGEMENT

Authors thanks to the management of Narasaraopeta Engineering College, for their continuous encouragement. Finally also thanks to the referees for their comments to improve the quality of this research paper.

6. REFERENCES

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, “The Design and Analysis of Computer Algorithms”, Addison-Wesley, 1974.

[2] E. Horowitz and S. Sahni, “Fundamental of Computer Algorithms” , Computer Science Press, 1982.

[3] E. L. Lawler, J. K. Lenstra, A. RinnooyKan, and D. B. Shmoys. The TravelingSalesman Problem: A Guided Tour of Combinatorial Optimization. Wiley,Chichester, England, 1985.

[4] Goldberg David E, Lingle R Jr. “Alleles, Loci, and the Traveling Salesman Problem.” Proc. Of 1st Int. Conf. on Genetic Algorithms and Their Applications, Lawrence Erlbaum Associates, 1985,154-159

[5] E. L. Lawler, J. K. Lenstra, A. RinnooyKan, and D. B. Shmoys. The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, Chichester, England, 1985.

[6] A. Gibbons and W. Rytter, "Efficient Parallel Algorithms" Cambridge University Press, 1988.

[7] M. Weiss, "Data Structures and Algorithm Analysis", Benjamin-Cummings, 1992.

[8] U. Manber, "Introduction to Algorithms: A Creative approach", Addison-Wesley, 1989.

[9] G. Gonnet and R. Baeza-Yates, "Handbook of Algorithms and Data Structures" , Addison- Wesley, 2 ed., 1991.

[10] B. Salzberg, "File Structures: An Analytic Approach", Prentice-Hall, 1988.

[11] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. Mathematics of Operations Research, 18(1):1–11, 1993.

[12] S. O. Krumke, W. E. de Paepe, D. Poensgen, and L. Stougie. News from the online traveling repairman. In J. Sgall, A. Pultr, and P. Kolman, editors, Proc. 26th Symp. on Mathematical Foundations of Computer Science, volume 2136 of Lecture Notes in Computer Science, pages 487–499. Springer-Verlag, 2001.

[13] Prof. Lenore Cowen, Scribe: Stephanie Tauber, Lecture notes on *"The Travelling Salesman Problem (TSP)"*, Comp260: Advanced Algorithms, Tufts University, Spring 2002.

[14] G. Gutin and A. P. Punnen, editors.The Traveling Salesman Problem and itsVariations. Kluwer, Dordrecht, TheNederlands, 2002.

[15] M. Lipmann. On-Line Routing.PhD thesis, Eindhoven University of Technology, 2003.

[16] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees, and minimum latency tours. In Proceedings of the 44th Annual Symposium on Foundations of Computer Science, Cambridge, Massachusetts, 2003.

[17] C. Chekuri and A. Kumar. Maximum Coverage Problem with Group Budget Constraints and Applications. Proc. Of APPROX-RANDOM, LNCS, 72–83, 2004.

[18] C. Chekuri and M. Pal. A Recursive Greedy Algorithm for Walks in Directed Graphs. Proc. of IEEE FOCS, 245–253, 2005.

[19] C. Chekuri and M. Pal. An O(log n) Approximation for the Asymmetric Traveling Salesman Path Problem. Proc. of APPROX, 95–103, 2005.

[20] K. Chen and S. Har-Peled. The Orienteering Problem in the Plane Revisited. Proc. of ACM SoCG, 247–254, 2006.

[21] V. Nagarajan and R. Ravi. Poly- logarithmic approximation algorithms for Directed Vehicle Routing Problems. Proc.of APPROX, 257–270, 2007.