One improvement to "nearest neighbor" method for solving "Traveling salesman" problem

mr Jozef Kratica[1], **mr Slobodan Radojevi}**[1]

Source: http://www.geocities.com/jkratica/papers/lira95.pdf

Abstract

This paper describes one improvement to "nearest neighbor" method for solving "traveling salesman" problem. This improvement finds sub optimal solutions (heuristic), so as original method.

The only algorithms that are known are of exponential execution time, because "traveling salesman" problem is NP complete ([4]). These algorithms are correct theoretically, and they easily find solution for small number of graph nodes (N<20), but for large number of nodes (N>20) execution time is enormously large.

Therefore, this paper describes improvement of "nearest neighbor" method, which gives suboptimal solutions (heuristic), but has $O(n^2)$ execution time.

AMS Subject Classifications:

90C27 Combinatorial programming
90C35 Network programming
90-08 Computational methods (optimization)

[1] University of Belgrade, Faculty of Mechanical Engineering, Department of Mathematics, 27. marta 80.

1. Introduction

1.1 NP complete problems

There is a class of problems which have not been theoretically proved to have exponential complexities, but no polynomial algorithms have been found. Some of problems (which are known as **NP complete problems**) are: Traveling salesman problem, problem of Hamiltonian paths, knapsack problem, problem of optimal graph coloring. If a polynomial solution could be found for any of these problems , than all of the NP problems would have polynomial solution. NP complete problems are described in more detail in [2] [4] [9] [10].

1.2 Traveling salesman problem

Let G be a connected graph with N nodes. Let a **tour** is sequence of all nodes, passing each node once, except the first, which is also the last node in sequence, and is passed twice. Problem is to find tour with minimal path (sum of including edges). Respecting the fact that problem is NP, three alternatives exist:

Use of existing algorithms of exponential complexity, which give optimal solutions, but are applicable only for small number of nodes (N<20).

Use of heuristic algorithms of polynomial complexity, which give solutions near optimal, but are applicable for large number of nodes (N<1000). Later in this text an improvement of such type will be presented.

Combining few different methods, which are used in several stages. First stage is one of heuristic algorithms, which give starting suboptimal solution. Next stages improve solutions (one example is k-interchange method).

1.3 Optimal solutions algorithms

All possible tours can be searched (number of all possible tours is (n-1)!) by backtracking. This method has small memory requirements, but largest O(n!) execution time. For example, for 20 nodes graph, execution time would be several millenniums.

We can use technique known as "dynamic programming" ([2]). It has execution time $O(n^2\ 2^n)$, which is smaller than , but has large $O(n\ 2^n)$ memory requirements. For example, in 20 nodes graph, execution time would be several hours, but memory requirement would be approximately 100MB.

Or, we can use technique known as "branch and bound" ([1] [2]). This is an improved solution of 1.3.1. (backtracking), but only feasible (and not all possible) tours are searched. This method has theoretical execution time of O(n!), same as "backtracking", but its execution time can be much shorter when applied on some particular graphs.

1.4 Heuristic methods

Problem can be partially reduced to problem of finding "minimal spanning tree" ([3]). Finding solution for this problem, by use of Prim or Kruscal algorithm ([2] [9] [10]), gives us solution to "traveling salesman" which is not more than twice of optimal solution length. It's applicability is bounded only to "Euclidean traveling salesman" problem, and it's execution time is O(n) ([2] [10]).

Nearest neighbor method ([7] [10]) finds minimal possible distance (in each iteration) connecting unused nodes. It has easy implementation and O(n) execution time.

Other ways of solving this problem are presented in [5] [7] [8].

"Best adjunct" algorithm, described in [6].

1.5 Combination of methods

The k-interchange method can be applied to solutions found by use of methods described in 1.4. This method checks for k edges, which give shorter path than actual "traveling salesman" path. For k=2, this method, named 2-interchange, give reasonable solution, with O(n) execution time. Increasing of k gives better solutions, but execution time and complexity of implementation increase rapidly ([7] [8] [10]).

To solution found by use of methods described in 1.4, and eventually improved as explained in 1.5.1., some of "branch and bound" methods, or their combination, can be applied. Previously found solution by heuristic, are used as a bound for feasible solutions. If previously found suboptimal solution is excellent, and we search all possible branches, optimal solution can be found in real time. Although this way can lead to near optimal or optimal solution, it's execution time is theoretically exponential ([7] [10]).

2. "Nearest neighbor" algorithm

2.1 Description of algorithm

"Nearest neighbor" algorithm is a heuristic algorithm (for more information on heuristic see [9] and [10]). Execution time is O(n), and memory requirements are O(n), because of distance matrix being stored.

Algorithm starts from arbitrary chosen node, in every iteration goes along the edge with minimal distance from current node, viewing only unused nodes and starting from current node. Terminating node of the edge chosen as described is add to the sequence and is pronounced to be the current node. Note that the last iteration leaves us with only one edge (leading to the starting node).

2.2 Improvement of algorithm

Algorithm can be modified to start from node that not chosen randomly (where favorable choice of starting node can give good results, and some other choice can give bad results), but from the node which is beginning node of the shortest edge in the entire graph. Next node is terminating node of that edge, and procedure continues the way described above.

This way of choosing starting node gives statistically better results than former method. Slightly better results can be achieved by recalculating sequence for each possible starting node, but that would rise execution time to $O(n^3)$.

The following described algorithm, implementation is in C programming language.

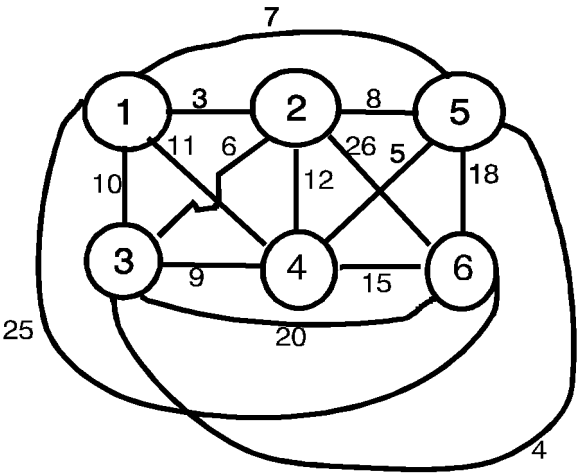

Figure 1. Graph from Example 1.

2.3 Examples

Example 1. Graph is given by Figure 1.

Classical "nearest neighbor" algorithm gives sequence starting from node **1**:

Node nearest to node **1** is node **2** (edge **1-2** has distance 3). Node nearest to node **2** is node **3** (with distance 6), and nearest to node **3** is **5** (with distance 4). Node **5**, nearest unused node is **4** (with distance 5), and to go from node **4,** there is only one unused node **6** (with distance 15).

Given sequence **1 2 3 5 4 6 1** has length 58.

Improved algorithm give sequence **1 2 3 5 4 6 1**, also of length 58, because minimal length edge in graph is **1-2** (with distance 3). That edge (1-2) starts the sequence, and later procedure has equal steps as first procedure.

Optimal solution in Example 1 is **1 2 3 6 4 5 1**, with length of 56.

6 1

0	3	10	11	7	25
3	0	6	12	8	26
10	6	0	9	4	20
11	12	9	0	5	15
7	8	4	5	0	18
25	26	20	15	18	0

Table 1. Distance matrix for graph given in Example 1

Note: In Tables 1-3 first number is number of graph nodes, second number is starting node, and other numbers represent distance matrix.

Example 2. Graph is given by Table 2.

Classical algorithm starting from node **1** gives solution **1 8 9 4 10 6 2 7 5 3 1**, length 403.

Improved algorithm gives solution **6 10 4 9 8 1 5 7 3 2 6**, length 323. For this example, that is optimal solution.

Example 3. Graph is given by Table 3, solutions are:

Classical algorithm starting from node **1** give a sequence **1 3 2 5 7 9 4 6 10 8 1**, length 805.

Improved algorithm gives sequence **8 10 2 3 1 5 7 9 4 6 8**, length 788.

Optimal tour is **1 3 2 8 10 5 7 9 4 6 1**, length 725.

10	1								
0	96	105	50	41	86	46	29	56	70
96	0	78	49	94	21	64	63	41	37
105	78	0	60	84	61	54	86	76	51
50	49	60	0	45	35	20	26	17	18
41	94	84	45	0	80	36	55	59	64
86	21	61	35	80	0	46	50	28	8
46	64	54	20	36	46	0	45	37	30
29	63	86	26	55	50	45	0	21	45
56	41	76	17	59	28	37	21	0	25
70	37	51	18	64	8	30	45	25	0

Table 2. Distance matrix for graph given in Example 2.

10	1								
0	72	43	217	92	183	116	143	161	141
72	0	51	222	52	182	99	83	186	72
43	51	0	229	97	199	121	115	203	101
217	222	229	0	219	40	156	285	63	299
92	52	97	219	0	179	57	123	159	100
183	182	199	40	179	0	130	265	65	251
116	99	121	156	57	130	0	158	91	154
143	83	115	285	123	265	158	0	269	14
161	186	203	63	159	65	91	269	0	255
141	72	101	299	100	251	154	14	255	0

Table 3. Distance matrix for graph given in Example 3

4. Conclusion

Application of "nearest neighbor" algorithm and its modification is (as of other suboptimal algorithms) mostly in practical work where graph with large number of nodes occur. In these conditions optimal solution algorithms are unusable because of their exponential execution time, i.e. their impossibility to yield solution in acceptable time.

Good features of improved "nearest neighbor" algorithm are:

Relatively short execution time $O(n^2)$.

Easy implementation.

Small memory requirements $O(n^2)$.

Solution can be improved using by the algorithms described in 1.5.

Deficiencies of algorithm are:

Being unable to find theoretical bounds of solutions suboptimality. For some algorithms theoretical bound of solutions suboptimality (see 1.4.1) can be calculated, but these methods statistically have worse results (see examples in [3] [10]), and are applicable for more specific graphs.

In case of graph with small number of edges which isn't totally connected may exist situations where this algorithm can't yield solution, though there exists.

5. References

[1] **Bellmore M., Nemhauser G. L.** "*The traveling salesman problem: A Survey*" Operations Research, 16(3), pp.538-558, 1968.

[2] **Brassard G., Bratley P .** "*Algorithmics - Theory and Practice*", Prentice-Hall International, 1988.

[3] **Christofides N.** "*Worst-case analysis of a new heuristic for the traveling salesman problem*", Management Sciences Research Report no. 388, Carnegie-Mellon University, Pittsburgh, 1976.

[4] **Garey M.R., Johnson D.S.** "*Computers and Intractability: A Guide to the Theory of NP Completness*", W.H. Freeman and Co., 1979.

[5] **Horowitz E., Sahni S.** "*Fundamentals of Computer Algorithms*" Computer Science Press, 1978.

[6] **Kratica J.** "*One method for solving Traveling salesman problem", Proceedings of the Fourth Symposium about Capacity utilization of Metal-refinement industry in reduced production limitations*, pp. 143-145, 1994.

[7] **Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B.** "*The Traveling Salesman Problem*", John Wiley & Sons, 1985.

[8] **Lin S., Kernighan B.W.** "*An effective heuristic for the traveling salesman problem*", Operations Research, 21, pp. 498-516., 1973.

[9] **Manber U.** *"Introduction to Algorithms - A Creative Approach"*, Addison-Wesley Publishing Company, 1989.

[10] **Nemhauser G.L., Wolsey L.A.** "*Integer and combinatorial optimization*", John Wiley & Sons, 1988.