A Simple Algorithm for Steiner Tree Problem in Networks

Sandeep Kumar
M.Tech Computer Science & Engineering
NIT Calicut, India

ABSTRACT

Many problems in combinatorial optimization are known to be NP-hard, and thus it is unlikely that there exists any polynomial-time algorithm to solve them. A number of these problems are of practical interest like Minimum Set Cover, Shortest Superstring, Steiner Tree and Traveling Salesman Problem etc. So people have turned to develop polynomial time approximation algorithms to solve them approximately because efficient optimal solution is not achievable.An α-approximation algorithm runs in polynomial time and it is sure to produce a solution of cost within α times the ptimal cost for any input.Steiner Tree problem is one of the open approximation algorithm problems. In this paper a study of various solutions have been done and we propose a new solution for Steiner Tree Problem in graphs (networks), which give results that are close to optimum on a variety of graphs.

Keywords : Approximation Ratio, Optimal Solution, Heuristic, Distance Network

1. INTRODUCTION

Most interesting real-world optimization problems are very challenging from a computational point of view. In fact, quite often, finding an optimal or even a near-optimal solution to a large-scale optimization problem may require computational resources far beyond what is practically available. An approximate algorithm is a way of dealing with NP-completeness for optimization problem. This technique does not guarantee the best solution. The goal of an approximation algorithm is to come as close as possible to the optimum value in a reasonable amount of time which is at most polynomial time. The design of good approximation algorithms is a very active area of research where one continues to find new methods and techniques. It is quite often that these techniques will become of increasing importance in tackling large real-world optimization problems. Since we do not know of efficient algorithms to find optimal solutions for NP-hard problems, a central question is whether we can efficiently compute good approximations that are close to optimal. It would be very interesting and practical if one could go from exponential to polynomial time complexity by relaxing the constraint on optimality, especially if we guarantee at most a relative small error [1].

A. Approximation Ratio

An algorithm approximately solves an optimization problem if it always returns a feasible solution whose measure is close to optimal. This intuition is made precise below. Let π be an optimization problem. We say that an algorithm A feasibly solves Π if given an instance $I \in \Pi$, $A(I) \in Sol(I);$ that is, A returns a feasible solution to I. Let A feasibly solves Π. Then we define the approximation ratio $\alpha(A)$ of A to be the minimum possible ratio between the measure of A(I) and the measure of an optimal solution. Formally[2],

$$\alpha(A) = \min \frac{m_I(A(I))}{m_I(OPT(I))} \quad (1)$$

For minimization problems, this ratio is always at least 1. Similarly, for maximization problems, it is always at most 1. Rest of the paper, Section 2 gives the deatils of the Steiner Tree Problem with special reference to Networks. In Section 3 various literature solutions have been studied. Section 4 gives details of our proposed approach with complexity. Then in Section 5, we have analyzed and compared our results with the traditional approach. Finally we have concluded this paper in Section 6.

2. STEINER TREE PROBLEM IN NETWORKS

In the network communications, data always starts from one or more sources and then sent to multiple destination nodes. This problem if often defined described as the multicast routing problem or in simple words if we want to lay communications network among n regions in order to achieve information sharing between all regions. How can we lay to make the total length of the communication lines shortest? The most common way is to seek to the minimum spanning tree connecting these n points, But if it is not limited to n points, while introduce other points apart from the n points, then make the total length of the communication lines connecting every region even shorter[3]. This is the source of Steiner Minimum Tree problem. Formally we can define Steiner Tree Problem in graphs as:-

Given an undirected network G = (V;E; c) where c : E → R is an edge length function, and a non-empty set N , $N \subseteq V$, of terminals. Find a subnetwork TG(N) of G such that:

- There is a path between every pair of terminals.
- Total length $|T_G(N)| = \sum_{e_i \in T_G(N)} c(e_i)$ is minimized.

The vertices in V \ N are called non-terminals. Nonterminals that end up in TG(N) are called Steiner vertices. The subnetwork TG(N) is called a Steiner minimal network for N in G [4]. A. Trivial Cases Let G = (V,E, c) be an undirected, connected network, and let N = v_1; v_2,…….., v_n be a set of terminals in G. Symbols z_1, z_2,…z_n are generally used to denote terminals. Under the assumption that all edges have positive length, there are two

special cases

- If n = 2, then the steiner tree problem reduces to the wellknown shortest path problem. There are many polynomial time algorithms for shortest path problem like Dijkastra's algorithm.
- If n = V , then the steiner tree problem reduces to the well-known minimum spanning tree problem. There are various solutions available for minimum spanning tree problem like Prim algorithm, Kuruskal's algorithm etc.

3. RELATED WORK

We have studied many available approaches for the Steiner Tree Probelm. A brief study of these heuristics have been given here :

A. A Steiner Tree Algorithm based on Sollin's Algorithm

This heuristic algorithm consists of nine steps. In first step, after assuming that T is equal with null, we obtained TMST by graph using Sollins algorithm but here we don't need to obtain complete MST. When our tree contains all of the terminals and is connected, we stop algorithm. As a result our tree that named TMST is produced. If we have in TMST non-terminal with degree one, in step two delete them. In the third step of algorithm, we assume two divisions of edges and paths between two terminals in TMST

- Existence direct edge in TMST .
- Not existence direct edge in TMST and existence direct edge in graph.

In the fourth step, for the first category, select the direct edges of two terminals which obtained by TMST and add into T in the next step, for the second category, make compare between paths and direct edges of two terminals if direct edge be shortest, add into T . If path be shortest, add into T. In the sixth step, we study T , to determine all disjoint terminals. In the next step obtain the shortest paths between disjoint terminal and other terminals in the graph with Dijkstra algorithm. Select shortest path between reached paths and add into T . In the eighth step, we study T, for connectivity. If we have forest, determine terminals with degree 1, then seek shortest path from this terminal to other terminals, and add it into T. At the end, study T for cycle. If we have cycle, delete cycle[5].

B. Minimum Spanning Tree and Pruning (MSTP)

The algorithm first arbitrarily choose a vertex from the terminals and Prim's algorithm for minimum spanning tree is run on the whole graph. Then nonterminal nodes are removed from the graph which are not necessary into the Steiner tree, because we need to span only terminal nodes by considering if necessary some Steiner nodes[3].

Algorithm 1 MST and Pruning Algorithm
Input: An undirected distance graph $G = (V, E, d)$ and set of steiner points $S \subseteq V$
Output: A Steiner tree, T_H , for G and S

1: Begin with a terminal as the root of MST
2: Run Prim's Algorithm on the graph
3: Remove unnecessary nonterminal nodes
4: Delete from this MST non terminals of degree one

C. Shortest Path Heuristic

The algorithm first arbitrarily choose a vertex D as a subtree, and regard the vertex as a sub-tree root, then add the shortest path from all other D vertexes to sub-root to generate feasible Steiner tree[4].

Algorithm 2 Shortest Path Heuristic for Steiner Tree

1: Begin with a subtree T_{sph} of G consisting of a single arbitraly chosen terminal
2: if $k = n$ then Stop
3: Determine a terminal $z_{k+1} \notin T_{sph}$ closest to T_{sph} . Add this terminal to T_{sph} a shortest path joining with z_{k+1} ,$k = K + 1$. Goto step 2
4: Determine a Minimum Spanning Tree for the subnetwork of G induced by the vertices in T_{sph}
5: Delete from this MST non terminals of degree one

D. Minimum Path Heuristic

Firstly, Choose a vertex v1 from set D randomly, the initial spanning tree $T1 = \{v_1\}, V_1 = \{v_1\}$: Secondly, for i=2,3,...,m the node $v_i \in D \setminus V_i$, make the cost from v_i to T_i-1 lowest, and connect vi to T_{i-1} through the lowest cost path *Path* $(v_i, T_{i-1}), T_i = T_{i-1} \cup$ *Path* (v_i; T_i-1); Vi is the node set of Ti, the final tree is TMPH[4].

4. PROPOSED SOLUTION

Given a connected undirected distance graph G = (V;E; d) and a set of terminal points $S \subseteq V$ consider the complete undirected distance graph G_1 = (V,E, d) constructed from G in such a way that for each $v_i, v_j \in E$, $d(v_i, v_j)$ is set to the distance of the shortest path from v_i to v_j in G. For each edge in G_1 these corresponds a shortest path in G. Given any spanning tree in G_1, we can construct a subgraph of G by replacing each edge in the tree by its corresponding shortest path in G. This heuristic Algorithm 3 for the steiner tree is simply explained by the Figure 1 given after detailed algorithm step by step.

Algorithm 3 Proposed Algorithm for Steiner Tree
Input: An Undirected distance graph $G = (V, E, d)$, a set of steiner points $S \subseteq V$
Output: A Steiner tree, T_H , for G and S

1: Construct the complete undirected distance graph $G_1 = (V, E, d)$ from G using Flyod Warshell's Algorithm [1]
2: Construct the subgraph, G_s of G_1 by replacing each edge by its corresponding shortest path in G .
3: Find the minimum spanning tree T_s of G_s.
4: Construct a Steiner Tree, T_H, from T_s by deleting edges in T_s, if necessary so that all leaves in T_H are steiner points.

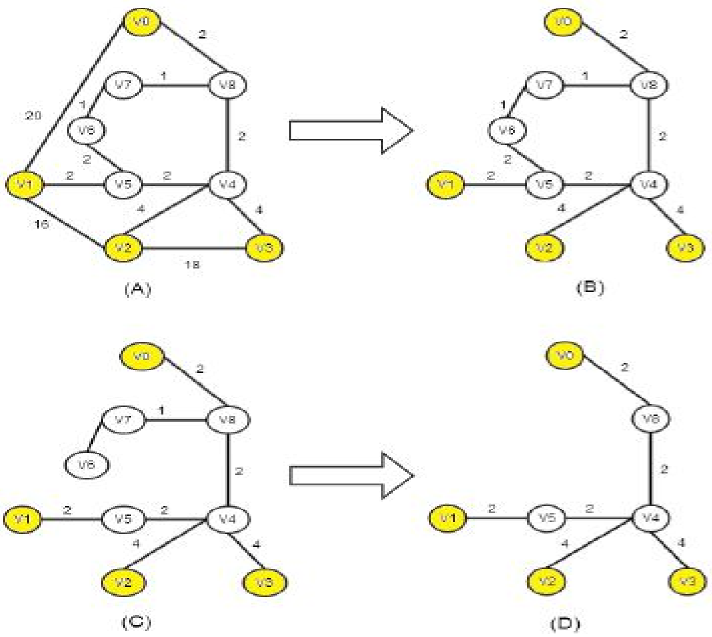

Fig. 1:. Example of Proposed Approach

When Flyod Warshall's algorithm is executed all those edges will be removed which are not in any of the shortest paths, as in case of our example Figure 1, edges with weights 20,16,18 have been removed in (B). Now MST of resulting graph will give a sub graph which contains edges which were in the shortest paths shown in (C). After removing non terminals of degree one, we get a Steiner Tree shown in (D).

A. Complexity of Proposed Algorithm

Our proposed algorithm 3 consists of 4 steps. In the first step, we are running Flyod Warshall's Algorithm whose running time is $\theta(V^3)$ [1]. Main time consuming portion is step 1 only. Step 2 of Algorithm can be done in constant time. Step 3 of the proposed approach uses Prim's Algorithm[1] whose running time is $\theta(\hat{V}^2)$. Step 4 can also be done in constant time. So total time of the proposed algorithm will be upper bound of step 1 and step 3, hence running time of our algorithm is $\theta(V^3)$.

5. ANALYSIS OF RESULTS

Both the heuristic algorithms Minimum Spanning Tree with Pruning(MSTP) and Proposed heuristic have been implemented in C language. For representing the graph data structure we used adjacency matrix representation[6]. For checking our results with OPTIMALITY we have used benchmark test cases referred from "11th DIMACS Implementation Challenge in Collaboration with ICERM: Steiner Tree Problems"[7]. A study of 15 different test cases from this library have been done for both MSTP and Proposed heuristic. Both heuristics have been tested on these data sets(test cases).

Test Case ID	OPT Cost	MSTP (Cost)	Proposed (Cost)
I080-001	1787	2986	2243
I080-041	1276	3256	1566
I080-002	1607	2434	1888
I080-025	1162	2533	1483
I080-142	1708	4002	2717
I080-221	3158	5389	4386
I080-232	4199	6616	6258
I080-312	4534	7397	6630
I080-333	5381	7152	7453
I080-334	5264	6448	6549
I080-341	4236	6782	6242
I080-344	4310	6952	5646
I080-345	4341	6700	6341

TABLE I. COMPARISON OF MSTP AND PROPOSED SOLUTION WITH STANDARD TEST CASES

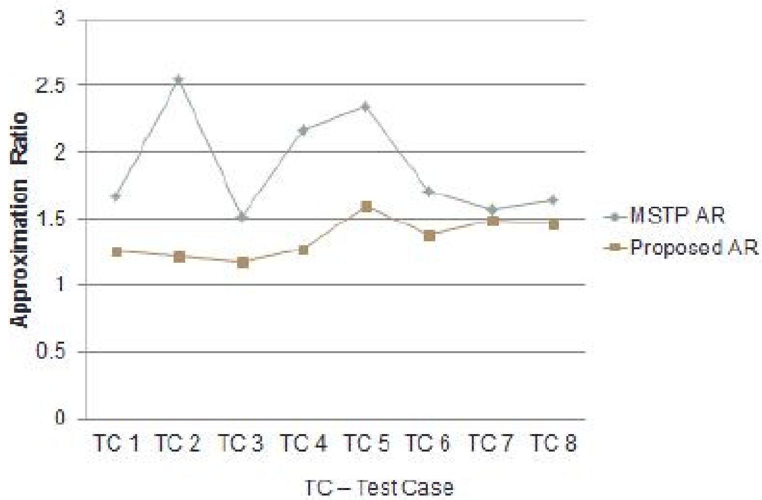

Fig. 2: Comparision of Approximation Ratio

Optimal cost have been noted from the library referred[7]. From the table, we can observe that the approximation ratio of our proposed solution is better almost in each test case. When graph size grows, then also we are getting better Optimal values as we can see in I080-221 and I080-232 with 3160 edges. The best proved approximation ratio for Steiner Tree Problem is 1:55. And from the table also we can see the proposed approach is almost beating the known ratio almost in each case. From the graph, we can observe that, MSTP curve have a lot of variation compared to our Proposed Approach curve, Hence we can say that our approach is more stable than the MSTP approach.

6. CONCLUSION

We have seen from the analysis of results, for smaller graphs there is not much difference in the Cost as well as Time for both MSTP and proposed approach. But as the graph size grows, cost of Steiner Tree obtained by the proposed approach is much closer to OPTIMAL than that obtained by MSTP. We also observed difference in Time is not much for very large graphs. So we can say that, for large graphs, it is better to use proposed approach for getting more OPTIMAL Cost by compromising slight increase in time, because our primary goal was to reduce the cost. Analytically this suggests that, we are getting an approximation ratio better than the known ratio for a variety of inputs. Steiner tree problem has many practical applications like circuit layout and network design. The problem is studied continuously and many researchers have given improvements in approximation ratio of Steiner Tree Problem using different heuristics. A study of these previously proposed algorithms and heuristics have been done in this paper, also a new solution have been proposed. By analyzing these heuristics and some more, one can extend the work on the Steiner Tree Problem.

REFERENCES

[1.] Ronald L.Rivest Thomas H.Coreman, Charles E.Leiserson. Introduction to Algorithms. The MIT Press,Cambridge, Massachusetts London, England, Third edition, 2009.
[2.] Vijay V. Vazirani. Approximation Algorithms. College of Computing Georgia Institute of Technology, First edition, 1997.
[3.] Pissinou N Kia Makki. A New Effective Heuristic for Solving Minimal Steiner Tree Problem on Graphs. In Computational Intelligence and Software Engineering (CiSE), 2010 International Conference, pages 1–4, December 2010.
[4.] P.Winter F.K. Hwang, D.S. Richards. The Steiner Tree Problem. North Holland, First edition, 1992.
[5.] Ali Nourollah Elnaz Pashaei and Mohammad Reza Meybodi. A New Steiner Tree Algorithm based on Sollin's Algorithm in Graphs. In Computer Science and Automation Engineering (CSAE),IEEE International Conference on (Volume:2), pages 489–493, December 2012.
[6.] Narasimha Karumanchi. Data Structures and Algorithms Made Easy. CareerMonk Publications, Second edition, 2013.
[7.] th DIMACS Implementation Challenge in Collaboration with ICERM: Steiner Tree Problems. DIMACS the DIMACS Special Focus on Information Sharing and Dynamic Data Analysis and by the Institute for Computational and Experimental Research in Mathematics (ICERM) http://steinlib.zib.de/showset.php?I080 Accessed on 5th May, 2014.
[8.] L.Kou G.Markowsk and L.Berman. A fast Algorithm for Steiner Tree. In Acta Informatica by Springer-Verlag, pages 141–145, May 1981.
[9.] L.Kou G.Markowsk and L.Berman. A fast Algorithm for Steiner Tree. IBM Thomas J Watson Research Center, First edition, 1981.
[10.] Kia Makki and N. Pissinou. The Steiner Tree Problem with Minimum Number of Vertices in Graphs. In Proceedings of the Second Great Lakes Symposium on VLSI, pages 204–206, May 1992.
[11.] Ali Nourollah Fatemeh Ghadimi. A Heuristic Algorithm for Solving Steiner Tree Problem on Urban Traffic Network. In 20th Iranian Conference on Electrical Engineering,(ICEE2012), Tehran, Iran, pages 651–655, May 2012.
[12.] Ashish Tamrakar Narender Maharjan, Abinash Shrestha and Sanjeeb Parsad Panday. Solving Minimum Steiner Tree Based on Behavior of Ants. In Internet (AH-ICI), 2012 Third Asian Himalayas International Conference,pages 1–4, November 2012.
[13.] Ali Nourollah Elham Pashaei and Alireza Bagheri. A Heuristic Algorithm for Euclidean Steiner Minimal Tree Inside Simple Polygon. In Computer Science and Automation Engineering (CSAE), IEEE International Conference on (Volume:1), pages 76–80, May 2012.
[14.] Wikipedia. Steiner tree problem wikipedia, the free encyclopedia, Acccessed on 12th May, 2014 http://en.wikipedia.org/wiki/Steiner_ tree_proble