The Steiner Tree Problem with Minimum Number of Vertices in Graphs

Kia Makki

Department of Computer Science
University of Nevada, Las Vegas
Las Vegas, Nevada 89154

Niki Pissinou

Department of Computer Science
University of Southern California
Los Angeles, California 90089

Abstract

The Steiner tree problem is to find a tree in a connected undirected distance graph $G = (V, E, d)$ which spans a given set $S \subseteq V$. The minimum Steiner tree for G and S is a tree which spans S with a minimum total distance on its edges. In this paper we consider a special case of the Steiner tree problem in graphs. For this problem we assume that the underlying graph G does not have any direct edge between the vertices in $S \subseteq V$. The problem is to find a tree in G which spans the vertices in S and uses minimum number of vertices in $V - S$.

1 Introduction

The Steiner tree problem is to find a tree in a connected undirected distance graph $G = (V, E, d)$ which spans a given set $S \subseteq V$. The minimum Steiner tree for G and S is a tree which spans S with a minimum total distance on its edges. The Steiner tree problem has a wide variety of practical applications such as communication networks, layout design of printed circuit board, the wire routing in physical VLSI design, and distribution and transportation networks.

It has been shown that finding a minimum Steiner tree for any given G and S is NP-Complete [4]. In fact even when distance functions are restricted to a particular class, the problem is still NP-Complete [3]. This means that it is unlikely that an efficient algorithm can be found to compute the minimum Steiner tree for any given G and S. Thus approximation algorithms were studied [7, 6, 8, 5]. [7] presents an $O(|S||V^2|)$ approximation algorithm for finding a Steiner tree in a given G and S. Moreover, it has been shown that

$$D(T_s)/D(T_{opt}) \leq 2(1 - 1/|S|)$$

where $D(T_s)$ denotes the total distance on the edges of the Steiner tree generated by [7] and $D(T_{opt})$ denotes the total distance on the edges of the optimal Steiner tree. [6] presents an approximation algorithm with the same time bound, but with $D(T_s)/D(T_{opt}) \leq 2(1 - 1/L)$ where L is the number of leaves in the optimal Steiner tree. The time bound of [6] has been further improved by [5] to $O(|E| + |V|log|V|)$.

The bound $2(1 - 1/L)$ has been known to researchers in the field for a number of years as the best worst-case ratio. It is not known whether there is an approximation algorithm with worst-case better than $2(1-1/L)$; in fact, it is widely speculated that no better worst-case bound for the Steiner tree problem can be achieved in polynomial time.

In this paper we consider a special case of the Steiner tree problem in graphs. For this problem we assume that the underlying graph G does not have any direct edge between the vertices in $S \subseteq V$. The problem is to find a tree in G which spans the vertices in S and uses minimum number of vertices in $V - S$. Here, we present and analyse an efficient approximation algorithm. The underlaying theme of this approximation algorithm is the use of the shortest paths between vertices in the graph. The approximation algorithm has a worst case time complexity of $O(|S|(|E| + |V|log|V|))$, where $|S|$, $|E|$ and $|V|$ represent the number of Steiner vertices, the number of edges, and the number of vertices in a given graph G respectively.

The remainder of this paper is organized as follows: In the second section we present some basic definitions. The approximation algorithm is given in section 3. The worst case time analysis of the algorithm is given in section 4. The final section provides some concluding remarks.

0-8186-2610-0/92 $3.00 © 1992 IEEE

2 Basic Definitions

In a connected undirected graph $G = (V, E, d)$ where V is a set of vertices, E is a set of pairs of vertices called edges and d is a distance function which maps E into the set of non negative numbers, let $S \subseteq V$ be a set of Steiner vertices. A path from a vertex v_x to a vertex v_y in G is a sequence of vertices $v_x, v_i, v_j, ..., v_t, v_y$ such that $\{ v_x, v_i \}, \{ v_i, v_j \}, ..., \{ v_t, v_y \}$ are edges in E; when $v_x = v_y$ then the path is called a loop. A simple path is a sequence of adjacent edges $\{ v_x, v_i \}, \{ v_i, v_j \}, ..., \{ v_t, v_y \}$ in which all the vertices $v_x, v_i, v_j, ..., v_t, v_y$ are distinct. The length of a path in G is the sum of the distances of its edges. Among all the different paths between a pair of vertices, say v_x and v_y, a path with minimum length is called a shortest path.

A connected subgraph of G is a tree if the removal of any edge in the subgraph causes it to be disconnected. We shall denote $D(T)$ to be the total distance on the edges of the tree T. Let $PATH(W, v)$ denote a path whose length is minimum among all shortest paths from vertices in W to vertex v where $W \subseteq V$ and $v \in V - W$.

3 The Approximation Algorithm

In the following we give an algorithm for finding a Steiner tree in a connected graph G.

Intialization:

Initially, we start with a subgraph $G_1 = (V_1, E_1)$ where $V_1 = \emptyset$ and $E_1 = \emptyset$. Also, initially, among all the vertices in $V - S$ we choose the one which is directly connected to the maximum number of vertices in S. We add these vertices to V_1 and the edge between them to E_1.

Main Loop:

Repeat the following steps until all the Steiner vertices are included in G_1.

Find all the non-Steiner vertices which are directly connected (through an edge) to the most recent non-Steiner vertex added to the subgraph G_1. Now we consider the following two cases. If none of these non-Steiner vertices are directly connected (through an edge) to one or more Steiner vertices then go to case 1, otherwise go to case 2.

Case 1: Find a vertex v_i in $S - V_1$ such that $PATH(V_1, v_i)$ has the minimum length (the length of a path here is the number of edges on that path). Add all the vertices on this path and their corresponding edges to the subgraph G_1. Also, add all the Steiner vertices which are directly connected (through an edge) to this path to the subgraph G_1.

Case 2: Among all these non-Steiner vertices find the one which is directly connected (through an edge) to the maximum number of Steiner vertices. Add such a non-Steiner vertex and all the Steiner vertices directly connected to it to the subgraph G_1.

4 Time Complexity Analysis

Theorem 1: The running time of the algorithm is proportional to $|S|(|E| + |V|log|V|)$.

Proof. Clearly the worst case time complexity of this algorithm is dominated by the computation of the shortest paths between all pairs of distinct Steiner vertices and the vertices in $V - S$. This can be done using Dijkstra's shortest algorithm [1, 2]. Hence the running time of the algorithm is in the worst case $|S|(|E| + |V|log|V|)$.

5 Summary and Conclusions

In this article we have presented an approximation algorithm for a special case of the Steiner tree problem in graphs. For this problem we assume that the underlying graph G does not have any direct edge between the vertices in $S \subseteq V$.

The problem we address here is to find a tree in G which spans the vertices in S and uses minimum number of vertices in $V - S$. It remains to be seen how well this algorithm performs in comparison with an optimal Steiner tree. An interesting open question is to decide whether or not there exists a constant C such that the worst-case error ratio of the algorithm is less than or equal to C.

References

[1] E. W. Dijkstra. A note on two problems in connection with graphs. *Numer. Math*, 1:269–271, 1959.

[2] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *IEEE*, pages 338–346, 1984.

[3] M. R. Garey, R. L. Graham, and D. S. Johnson. Some np-complete geometric problems. pages 10–22, 1976.

[4] R.M. Karp. *Reducibility among Combinatorial Problems In: Complexity of Computer Comptuations.* Plenum Press, 1972.

[5] L. Kou and K. Makki. An even faster approximation algorithm for the steiner problem in graphs. *Congressus Numerantium*, 59:147–154, 1987.

[6] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15:141–145, 1981.

[7] H. Takahashi and A. Matsuyama. An approximate solution for the steiner problem in graphs. *Math Japonica*, 24(6):573–577, 1980.

[8] Y. F. Wu, P. Widmayer, and C. K. Wong. A faster approximation algorithm for the steiner problem in graphs. *Acta Informatica*, 23:223–229, 1986.