Information Processing Letters 27 (1988) 125–128
North-Holland

25 March 1988

A FASTER APPROXIMATION ALGORITHM FOR THE STEINER PROBLEM IN GRAPHS *

Kurt MEHLHORN

Fachbereich 10 — Informatik, Universität des Saarlandes, D-6600 Saarbrücken, Fed. Rep. Germany

Communicated by T. Lengauer
Received 27 July 1987
Revised 14 September 1987

We present a new implementation of the Kou, Markowsky and Berman algorithm for finding a Steiner tree for a connected, undirected distance graph with a specified subset S of the set of vertices V. The total distance of all edges of this Steiner tree is at most $2(1-1/l)$ times that of a Steiner minimal tree, where l is the minimum number of leaves in any Steiner minimal tree for the given graph. The algorithm runs in $O(|E|+|V|\log|V|)$ time in the worst case, where E is the set of all edges and V the set of all vertices in the graph.

Keywords: Steiner tree, approximation algorithm, graph algorithm

1. Introduction

Consider a connected, undirected distance graph $G=(V, E, d)$ and a set $S \subseteq V$, where V is the set of vertices in G, E is the set of edges in G, and d is a distance function which maps E into the set of nonnegative numbers. A *path* in G is a sequence of vertices $v_1, v_2, \ldots, v_k$ of V such that, for all i, $1 \leqslant i < k$, $(v_i, v_{i+1}) \in E$ is an edge of the graph. The *length* of a path is the sum of the distances of its edges. A *tree subgraph* $G_s = (V_s, E_s, d_s)$ of G with $S \subseteq V_s \subseteq V$,

$$E_s \subseteq \{(v_1, v_2) \mid (v_1, v_2) \in E, \{v_1, v_2\} \subseteq V_s\},$$

and d_s equals d, restricted to E_s, is called a *Steiner tree* for G and S. Given a Steiner tree for G and S, $G_s=(V_s, E_s, d_s)$, $D(G_s)$ is defined as $\sum_{e \in E_s} d_s(e)$, and is called the *total distance* of G_s. A Steiner tree G_s for G and S is called a *Steiner minimal tree* if its total distance is minimal among all Steiner trees for G and S. This minimal distance is called $D_{\min}(G)$. Note that vertices in S are required to be in any Steiner tree for G and S. On the other hand, vertices in $V-S$, which are traditionally called *Steiner vertices*, are not required to be in a Steiner tree, but may be used to achieve a small total distance. Steiner trees were first considered by Gilbert and Pollak [3].

The problem of finding a Steiner minimal tree for given G and S has been shown to be NP-complete, even for a restricted class of distance functions (cf. [2]). Therefore, we are interested in finding a Steiner tree with total distance close to the total distance of a Steiner minimal tree. Takahashi and Matsuyama [6] presented an algorithm for finding a Steiner tree G' with

$$D(G')/D_{\min}(G) \leqslant 2(1-1/|S|),$$

whereas Kou, Markowsky and Berman [4] described a procedure for finding a Steiner tree G'' with

$$D(G'')/D_{\min}(G) \leqslant 2(1-1/l),$$

and l is the minimum number of leaves in any Steiner minimal tree for G and S. The runtime of both algorithms is proportional to $|S|\,|V|^2$. Note that $l \leqslant |S|$. More recently, Wu, Widmayer

* This research was supported by the DFG, Sonderforschungsbereich 124.

and Wong [8] improved the running time to $O(|E|\log|V|)$ and later Widmayer [7] to $O(|E|+(|V|+\min\{|E|, |S|^2\})\log|V|)$. In this paper, we describe a further improvement and achieve running time $O(|V|\log|V|+|E|)$. Our solution is not only faster than the last two solutions mentioned but also simpler, because we reduce the question at hand to a shortest path and a minimum spanning tree calculation.

2. An algorithm for approximating a Steiner minimal tree

Let $G=(V, E, d)$ be a given connected, undirected distance graph, and $S \subseteq V$ the set of vertices for which a Steiner tree is desired. Our algorithm is in line with Algorithm H in [4].

Algorithm H: Steiner Tree [4]

1. Construct the complete distance graph $G_1=(V_1, E_1, d_1)$, where $V_1=S$ and, for every $(v_i, v_j)\in E_1$, $d_1(v_i, v_j)$ is equal to the distance of a shortest path from v_i to v_j in G.
2. Find a minimum spanning tree G_2 of G_1.
3. Construct a subgraph G_3 of G by replacing each edge in G_2 by its corresponding shortest path in G. (If there are several shortest paths, pick an arbitrary one.)
4. Find a minimum spanning tree G_4 of G_3.
5. Construct a Steiner tree G_5 from G_4 by deleting edges in G_4, if necessary, so that no leaves in G_5 are Steiner vertices.

The most time-consuming step in Algorithm H is step 1. It requires the solution of $|S|$ single source shortest path problems and hence takes $O(|S|(|V|\log|V|+|E|))$ time using Fredman and Tarjan's [1] implementation of Dijkstra's algorithm. Wu, Widmayer and Wong [8] and later Widmayer [7] improved upon this by combining steps 1. and 2. into a single step. They achieved a time bound of $O(|E|\log|V|)$ and $O(|E|+(|V|+\min\{|E|, |S|^2\})\log|V|)$ respectively. We slightly refine their method, separate steps 1. and 2. again and achieve a running time of $O(|V|\log|V|+|E|)$. The details are as follows.

For every vertex $s\in S$ let $N(s)$ be the set of vertices in V which are closer to s than to any other vertex in S. More precisely, we consider a partition $\{N(s); s\in S\}$ of V, i.e., $V=\bigcup_{s\in S}N(s)$ and $N(s)\cap N(t)=\emptyset$ for $s, t\in S$, $s\neq t$, with

$$v\in N(s) \Rightarrow d_1(v, s)\leqslant d_1(v, t) \quad \text{for all } t\in S.$$

2.1. Remark. In the parlance of computational geometry we might call $N(s)$ the Voronoi region of vertex s. If a vertex v has equal distance to several vertices in S, then it belongs to the Voronoi region of one of them.

Next, we consider the subgraph $G_1'=(S, E_1', d_1')$ of G_1 defined by

$$E_1'=\{(s, t);\ s, t\in S \text{ and there is an edge } (u, v)\in E \text{ with } u\in N(s),\ v\in N(t)\}$$

and

$$d_1'(s, t)=\min\{d_1(s, u)+d(u, v)+d_1(v, t);\ (u, v)\in E,\ u\in N(s),\ v\in N(t)\}$$

Fig. 1 illustrates that d_1' is in general not the restriction of d_1 to the set E_1'. Nevertheless, a minimum spanning tree of G_1' is always a minimum spanning tree of G_1, as the following lemma shows.

2.2. Lemma. (a) *There is a minimum spanning tree G_2 of G_1 which is a subgraph of G_1'. Moreover, d_1 and d_1' agree on the edges of this tree.*

(b) *Every minimum spanning tree of G_1' is a minimum spanning tree of G_1.*

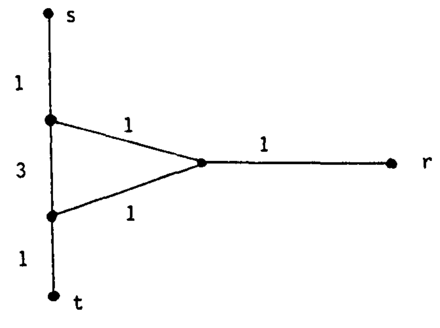

Fig. 1. $S=\{r, s, t\}$, $d_1(s, t)=4$, but $d_1'(s, t)=5$.

Proof. (a) Let $G_2 = (S, E_2)$ be that minimum spanning tree which minimizes the quantity $a := |E_2 - E_1'|$ and among those trees the one which has minimal total distance with respect to d_1'. If $a = 0$ and d_1 and d_1' agree on the edges of G_2, then we are done. Let us assume otherwise. Then either (case A) there is an edge $(s, t) \in E_2 - E_1'$ or (case B) $E_2 \subseteq E_1'$ and there is an edge $(s, t) \in E_2$ with $d_1'(s, t) > d_1(s, t)$. Let $v_0, \ldots, v_k$ with $v_0 = s$ and $v_k = t$ be a shortest path from s to t in G. For each vertex v_i let $s(v_i)$ be such that $v_i \in N(s(v_i))$. Then, for every i either $s(v_i) = s(v_{i+1})$ or $(s(v_i), s(v_{i+1})) \in E_1'$. Also, in the latter case, we have

$$\begin{aligned} &d_1(s(v_i), s(v_{i+1})) \\ &\quad \leqslant d_1'(s(v_i), s(v_{i+1})) \\ &\quad \leqslant d_1(s(v_i), v_i) + d(v_i, v_{i+1}) \\ &\qquad + d_1(v_{i+1}, s(v_{i+1})) \\ &\quad \leqslant d_1(s, v_i) + d(v_i, v_{i+1}) + d_1(v_{i+1}, t) \\ &\quad = d_1(s, t) \\ &(< d_1'(s, t) \text{ if case B applies}). \end{aligned}$$

Here, the first two inequalities are obvious and the third inequality follows from $v_i \in N(s(v_i))$ and $v_{i+1} \in N(s(v_{i+1}))$. The last inequality is only valid if case B applies.

Next, remove the edge (s, t) from the spanning tree. This splits the tree into two connected components. Thus, there must be an i such that $s(v_i)$ and $s(v_{i+1})$ are in different components.

Consider the tree

$$G_2' = (S, E_2 - \{(s, t)\} \cup (s(v_i), s(v_{i+1}))).$$

G_2' is clearly a spanning tree. Moreover, in case A, G_2' uses one more edge in E_1' than G_2 and has cost no larger than G_2 and, in case B, G_2' also uses only edges in E_1' and has with respect to d_1' a total distance strictly smaller than G_2. In either case, we derived a contradiction to the choice of G_2.

(b) Let G_2' be a minimum spanning tree of G_1' and let G_2 be a minimum spanning tree of G_1. By part (a) of the lemma we may assume that G_2 uses only edges in E_1' and that d_1 and d_1' agree on the edges of G_2. Thus,

$$d_1(G_2') \leqslant d_1'(G_2') \leqslant d_1'(G_2) = d_1(G_2) \leqslant d_1(G_2').$$

Here, the first inequality holds since $d_1(s, t) \leqslant d_1'(s, t)$ for all edges (s, t), the second inequality holds since G_2' is a minimum spanning tree of G_1', the equality holds by our assumption on G_2, and the third inequality holds since G_2 is a minimum spanning tree of G_1. We conclude that G_2' is a minimum spanning tree of G_2. □

We infer from this lemma that we may replace step **1.** of Algorithm H by

1'. Construct the auxiliary graph $G_1' = (S, E_1', d_1')$ where E_1' and d_1' are defined as above.

We discuss the implementation next. The graph G_1' has only $O(|E|)$ edges and hence step **2.** can be carried out in $O(|S| \log|S| + |E|)$ time (cf. [1]). The same amount of time certainly suffices for steps **3.**, **4.**, and **5.** It remains to describe an efficient implementation of step **1.**

We can compute the partition $\{N(s); s \in S\}$ by adjoining an auxiliary vertex s_0 and edges (s_0, s), $s \in S$, of length 0 to G and then performing a single source shortest path computation with source s_0. This takes $O(|V| \log|V| + |E|)$ time and yields for every vertex v the vertex $s(v) \in S$ with $v \in N(s(v))$ and the distance $d_1(v, s(v))$.

Next we go through all the edges (u, v) in E and generate the triples $(s(u), s(v), d_1(s(u), u) + d(u, v) + d_1(v, s(v)))$. We sort these triples by bucket sort according to the first two components (cf. [5, Section II.2.1]) and then select for each edge of G_1' the minimum cost. All of this takes $O(|E|)$ time.

We summarize our discussion in the following theorem.

2.3. Theorem. *For a connected, undirected distance graph $G = (V, E, d)$ and a set of vertices $S \subseteq V$, a Steiner tree G_s for G and S with total distance at most $2(1 - 1/l)$ times that of a Steiner minimal tree for G and S can be computed in time $O(|V| \log|V| + |E|)$.*

3. Conclusion

We presented a fast approximation algorithm for Steiner trees. Our algorithm is simpler than the algorithms by Wu, Widmayer and Wong [8] and Widmayer [7] since we reduce the Steiner tree approximation problem to a single source shortest path problem and a minimum spanning tree problem. Thus, any advances on those problems have direct implications for the Steiner tree problem.

References

[1] M.L. Fredman and R.E. Tarjan, *Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms* (IEEE, 1984) 338–346.

[2] M.R. Garey and D.S. Johnson, *Computers and Intractability* (Freeman, San Fransisco, CA, 1979).

[3] E.N. Gilbert and H.U. Pollak, Steiner minimal trees, *SIAM J. Appl. Math.* **16** (1) (1968) 1–29.

[4] L. Kou, G. Markowsky and L. Berman, A fast algorithm for Steiner trees, *Acta Informatica* **15** (1981) 141–145.

[5] K. Mehlhorn, *Data Structures and Efficient Algorithms* (Springer, Berlin, 1984).

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

[7] P. Widmayer, A fast approximation algorithm for Steiner's problems in graphs, *Graph-Theoretic Concepts in Computer Science, WG 86*, Lecture Notes in Computer Science, Vol. 246 (Springer, Berlin, 1986) 17–28.

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