Computer Science Journal of Moldova, vol.**1**, no.1(1), 1993

The last achievements in Steiner tree approximations

A.Zelikovsky

Abstract

The Steiner tree problem requires a shortest tree spanning a given point set S contained in a metric space (V, d). We describe a new approach to approximation solutions of this problem and analyze the time complexity of several algorithms.

1 Introduction

Let $M = (V, d)$ be a metric space and S be a subset of V. A tree T is a *Steiner tree* of S if S is contained in the vertex set of T.

Steiner Tree Problem (STP). Given M and S, find the shortest Steiner tree (also called the *Steiner minimal tree*) of S.

It is known that the Steiner tree problem is NP-hard when the metric is given as a graph [13] as well as when the metric is euclidean [8] or rectilinear [9]. Therefore algorithms which in polynomial time construct an approximate Steiner minimal tree are investigated. The quality of an approximation is measured by its performance ratio: an upper bound on the ratio between achieved length and the optimal length. Since STP is Max-SNP hard for graphs [3] the performance ratio should be larger than 1 for graphs.

A well-known heuristic (an *MST-heuristic*) for the Steiner tree problem approximates a Steiner minimal tree with a minimum length spanning tree (MST). It was proved that the lowest performance ratio of this heuristic equals 2 for graphs [17], $\frac{3}{2}$ for rectilinear metric [11] and $\frac{2}{\sqrt{3}}$ for euclidean metric [5].

The fastest known implementation of the MST-heuristic has a running time $O(E + V \log V)$ for a graph $G = (V, E)$ [15] (we use E, V, S and α to denote $\#E, \#V, \#S$ and the time complexity of the all-shortest-paths problem, respectively, in the order of a running time of an algorithm), $O(n \log n)$ ($\#S = n$) for rectilinear [16] and euclidean metrics. For many years, finding a better heuristic remained open.

Two better heuristics appeared recently [2,20] while consideration of a $k-$restricted Steiner tree.

First we introduce some denotations: $SMT(S)$ and $smt(S)$ are a Steiner minimal tree of S and its length, respectively. For a complete graph G_S, $M(G_S)$ denotes MST of G_S, and $m(G_S)$ denotes its length.

$SMT(S)$ may in general contain vertices of $V\backslash S$. So $SMT(S)$ contains the set S of *given vertices* and some additional vertices. $SMT(S)$ is called a *full Steiner tree* if S coincides with the set of leaves of $SMT(S)$. If $SMT(S)$ is not full, then we can split it into the union of edge-disjoint full Steiner subtrees. $SMT(S)$ is called *$k-$restricted* if every full component has at most k given vertices. Let the shortest $k-$restricted Steiner tree for the set S, denoted by $SMT_k(S)$, has the length $smt_k(S)$. Note, that $SMT_2(S) = M(F)$.

Let $r_k = sup\{smt_k(S)/smt(S)\}$. The bounds for the MST-heuristic imply that r_2 equals 2, $\frac{3}{2}$, $\frac{2}{\sqrt{3}}$ for graphs, rectilinear and euclidean metrics, respectively.

For graphs, it was proved that $r_3 = 5/3$ [19,20], $r_4 \leq \frac{3}{2}$ and $r_8 \leq \frac{4}{3}$ [1], moreover, $r_{2^k} \leq 1 + \frac{1}{k}$ [5].

For rectilinear metric, it was proved that $r_3 = \frac{5}{4}$ [21] and $r_k \leq 1 + \frac{1}{2k-2}$ [2]. The exact values of r_k are still unknown for $k > 3$. It was conjectured, that $r_k = 1 + \frac{1}{2k-3}$ [2].

For euclidean metric, the evaluation of r_k is more complicated. It was proved only that $\lim_{k\to\infty} r_k = 1$ and conjectured that $r_3 = \frac{1}{\sqrt{2}} + \frac{1}{1+\sqrt{3}}$ [5].

The above bounds arise a problem of finding an optimal k-restricted Steiner tree. This problem is NP-hard for $k > 3$ [13] and coincides with

MST for $k = 2$. For $k = 3$, the problem of exact polynomial solution is still opened.

The heuristics [2,20] approximate $k-$restricted Steiner minimal trees. We describe them in the rest of the paper.

A *greedy* heuristic has a performance ratio of $r_2 - (r_2 - r_3)/2$ [20]. Its implementation time is $O(S(E + VS + V \log V))$ [22] and $O(n^2)$ for graphs and rectilinear metric, respectively.

Another approach is suggested in [2]. A family of *evaluation* heuristics A_k is constructed. A_k achieves a performance ratio at most

$$r_2 - \sum_{i=3}^{k} \frac{r_{i-1} - r_i}{i - 1}$$

in time $O(\alpha + V^{k-2}S^{k-1} + S^{k+0.5})$ for graphs and $O(n^{\lfloor k/2 \rfloor + 3/2})$ [2]. For $k = 3$, the implementation time was reduced to $O(n^{3/2})$ for rectilinear metric [13].

Note that the problem of finding exact upper approximation bounds for the greedy and evaluation heuristics is still open.

The next section is devoted to the greedy heuristic and its implementation for graphs. In Section 3, we describe the evaluation heuristic and analyze its time complexity for rectilinear metric.

2 The Greedy Heuristic

Some preliminary definitions: given a triple $z = \{a, b, c\} \in S$, a Steiner minimal tree z^* for z (called a *star*) may include one additional vertex $v = v(z)$ (called a *center* of a star). The length of $z^* = (v(z); a, b, c)$ is denoted by $d(z) = d(v, a) + d(v, b) + d(v, c)$. For a set Z of triples, $d(Z)$ is the sum of lengths of its elements. $Triples$ denotes the set of all triples for S.

Let $T = M(G_S)$ and $d(T)$ denote the length of T. Given a pair of vertices a, b of T, we use $T[a, b]$ to denote an $MST(T \cup (a, b))$, where (a, b) is an edge of zero length. For any triple $z = \{a, b, c\}$ the graph $T[z]$ equals $T[(a, b)][(a, c)]$, i.e. it results from two reductions. For a set A consisting of pairs and triples we define $T[A]$ recursively:

$T[\emptyset] = T$, and $T[A \cup e] = T[A][e]$. For a set Z of triples, we define $win_T(Z) = d(T) - d(T[Z]) - d(Z)$. The equality $r_3 = \frac{5}{3}$ implies an existence of a set Z such that

$$d(T) - win_T(Z) \leq \frac{5}{3} smt(S) \tag{2.1}$$

The greedy heuristic chooses the best possible reduction of a previously achieved approximate solution. Below we present a rough version of the greedy heuristic.

Algorithm (greedy heuristic)

(0) $T \leftarrow M(G_S)$, $W \leftarrow \emptyset$;
(1) repeat forever
  (a) find $z = argmax\{win_T(z) | z \in Triples\}$;
  (b) if $win_T(z) \leq 0$ then exit repeat;
  (c) $T \leftarrow T[z]$; insert$(W, v(z))$;
(2) find a Steiner tree T_1 for $S \cup W$ in graph G using MST-heuristic.

A sequence of triples chosen by the greedy heuristic is called *greedy in* G_S. It was proved in [20] that the set of elements of a greedy sequence of triples H and an arbitrary set of triples Z

$$2win_T(H) \geq win_T(Z) \tag{2.2}$$

Inequalities (2.1) and (2.2) imply a performance ratio of 11/6 for the greedy heuristic.

An implementation of the greedy heuristic for graphs given in [20] generates stars for all triples of given vertices in time $O(\alpha + VS^2)$. The same procedure is necessary for the evaluation heuristic. This generation needs the shortest path distances between given vertices and additional vertices. Therefore, we can decrease its running time to $O(S(E + VS + V \log V))$ using an $O(E + V \log V)$-algorithm [17] for every given vertex $s \in S$ to find all shortest paths from s to other vertices of V.

Now we describe computing of the function win_T.

For a pair $e = (a, b)$ of given vertices define $save_T(e) = d(T) - d(T[e])$. Let $List(T) = \{t_1, ..., t_n\}$ be an nondecreasing order of edges of T.Then $save_T(e)$ is the length of the last edge of $List$, say t_i, in the unique cycle of $T \cup e$. The index i of t_i is denoted by $ind_T(e)$,i.e. $save_T(e) = d(t_{ind_T(e)})$. Note that $T[e] = e \cup T \backslash t_i$ and $List(T[e]) = \{e, t_1, ..., t_{i-1}, t_{i+1}, ..., t_n\}$.

Further, $\bar{z}$ denotes the set of three edges $\{(a, b), (b, c), (c, a)\}$, for a triple $z = \{a, b, c\}$. Note, that

$$win_T(z) = \max_{e \subset \bar{z}} save_T(e) + \min_{e \subset \bar{z}} save_T(e) - d(z)$$

This implies that it is sufficient to compute the function $save_T$. At first we find a binary tree T' which corresponds to T according to $List$. Inner vertices of T' correspond to edges of T, its leaves correspond to the vertices of T. A root of T' is t_n and sons of t_i are the last edges in two components which appear after deletion of t_i from $T' \backslash A(t_i)$, where $A(t_i)$ is the set of ancestors of t_i. If such component does not contain edges, then the son is the corresponding vertex of T. Moreover, using $preprocess(T')$ (preprocessing of time $O(S)$) we can find in time $O(1)$ the nearest common ancestor of any pair e of given vertices [10] which corresponds to the edge of T with the length $save_T(e)$.

Thus, to fulfill the step (1)(a) of Algorithm we need time $O(S^3)$, therefore, the step (1) demands time $O(S^4)$ and a running time of the whole algorithm is $O(S(E + VS + V \log V + S^3))$.

A faster modification of the greedy heuristic finds a best possible star with the center v for every vertex $v \in V \backslash S$ and then chooses the best one among all such stars [22]. This can be done in time $O(VS)$. As a result, the whole time complexity is $O(S(E+VS+V \log V))$. The time reduction is achieved by omitting of generating.

3 The Evaluation Heuristic

The evaluation heuristic is more complicated than the greedy one. It consists of four phases. The first phase generates all triples and then all k-tuples, for all $k = 4, ..., t$. As a result of this phase we have all exact Steiner trees for all k-subsets of S.

The second, *evaluation* phase consequently deals with k-tuples. For brevity, we consider only triples. If a triple has a positive win, then a certain pair of edges of $T = M_G(S)$ are replaced with two corresponding edges as greedy algorithm does. But the length of a new edge equal to the length of the removed edge minus the win of the triple. (The greedy algorithm reduce this length to zero.) In other words every time, when a triple has a positive win the metric is modified. Moreover, this phase forms a *Stack* which consists of the tuple, the pair of removed edges and and the pair of added edges. Note, that a k-tuple is discarded if it has a nonpositive win.

The third, *selection* phase repeatedly pops a tuple from the *Stack* and inserts it to $select-list$ if it decides to include the Steiner tree of this tuple in the output approximate Steiner tree. At first, the graph (S, E) contains all edges of G_S and all added edges. Every time this phase checks if all new edges of a tuple belong to the current minimum spanning tree. If this condition does not holds, then all these edges should be removed from the graph and a new MST should be constructed.

The last phase constructs the output Steiner tree from the $select-list$ tuples and remained MST-edges.

Now we will analyze the running time of the evaluation heuristic for rectilinear metric and $k = 3$ [2,14].

At first, we will show that it is sufficient to consider only linear number of stars while construction of a 3-restricted Steiner tree [14].

Further, for brevity, we assume that coordinates of all given points and subtractions between them are distinct.

First we need some definitions: A triple (s_0, s_1, s_2) is called a *star*, if $x_0 < x_1 < x_2$ and $(y_0 - y_1)(y_1 - y_2) < 0$ where $s_i = (x_i, y_i)$. There are four types of stars corresponding to the four possible orders of y_i. Below we consider the case of $y_1 < y_0 < y_2$ (the similar argument can be used for the other types of stars).

A *center* of a star is the point $c = (x_1, y_0)$ which is the additional point of SMT for the set $\{s_0, s_1, s_2)\}$. We use $z = (c; s_0, s_1, s_2)$ to denote such star. The three given points defining the star z also define a rectangle R where they lie on the boundary. R is empty, if there is no point (x, y) such that $x_0 < x < x_2, y_1 < y < y_2$.

It was shown that it is sufficient to consider a family of stars for which we know that there is a 3-restricted SMT with stars from this family [2]. Moreover, there is $SMT_3(S)$ with stars defining empty rectangles [2,21].

A star $z = (c; s_0, s_1, s_2)$ is called a *tree star* if $M(S \cup c)$ contains the edges of the set $\bar{z} = \{(c, s_0), (c, s_1), (c, s_2)\}$. It was proved that only tree empty stars may be considered.

A star is *positive* (*negative*) if $(y_2 - y_0) - (x_2 - x_1) > 0 (< 0)$. This means that the top vertex lies above (below) the diagonal through the center of the star. Yao [18] introduced a graph associated with the set S: every point is connected with the nearest points in all eight angles defined by aces and bisectors. The Yao graph contains $M(S \cup c)$. This fact implies directly that there are at most two distinct stars with the same center, namely the positive and the negative with the shortest length.

The last fact makes possible to prove that the number of empty tree stars is at most $36n$.[14]

The set of all empty tree stars can be generated in time $O(n \log^2 n)$ [14].

In the evaluation phase, the algorithm computes wins of $O(n)$ triples and recomputes the minimum spanning tree after adding new edges. This can be done using *dynamic trees* of Sleator and Tarjan [16] in time $O(\log n)$ per iteration. So the whole time for this phase is $O(n \log n)$.

In the selection phase, dynamic trees of Sleator and Tarjan are insufficient, since it is necessary to handle removal of edges. However, using Frederickson's data structure [6], we can update the minimum spanning tree in time $O(\sqrt{E})$ when an edge is deleted. Since $\#E = O(n)$, the whole time for the selection phase is $O(n^{1.5})$.

It is easy to see that the construction phase can be performed in $O(n)$ time.

Thus, the whole time complexity of the evaluation heuristic for rectilinear metric is $O(n^{1.5})$ [14].

References

[1] P. Berman and V. Ramaiyer. An approximation algorithm for the Steiner tree problem. *Tech. Rep.#CS-91-05, The Pennsylvania State University*, 1991.

[2] P. Berman and V. Ramaiyer. Improved approximations for the Steiner tree problem. *Proceedings, 3d SODA*, 325-334, 1992.

[3] M. Bern and P. Plassmann. The Steiner problems with edge lengths 1 and 2. *Inf. Proc. Lett.* 32, 171-176, 1989.

[4] D. Z. Du and F. K. Hwang. An approach for proving lower bounds: solution of Gilbert-Pollak's conjecture on Steiner ratio. *Proceedings, 31st FOCS*, 76-85, 1990.

[5] D. Z. Du, Y. Zhang and Q. Feng. On better heuristic for Euclidean Steiner minimum trees. *Proceedings, 32nd FOCS*, 431-439, 1991.

[6] G. N. Frederickson. Data structures for on-line updating of minimum spanning trees with applications. *SIAM J. Comput.* 14, 781-798, 1985.

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

[8] M. R. Garey, R. L. Graham, and D. S. Jonson. The complexity of computing Steiner minimal trees. *SIAM J. Appl. Math.* 32, 835-859, 1977.

[9] M. R. Garey and D. S. Jonson. The rectilinear Steiner problem is NP-complete. *SIAM J. Appl. Math.* 32, 826-834, 1977.

[10] D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestor. *SIAM J. Comp.* 13, 338-355, 1984.

[11] F. K. Hwang. On the Steiner minimal trees with rectilinear distance. *SIAM J. Appl. Math.* 30, 104-114, 1976.

[12] F. K. Hwang. An $o(n \log n)$ algorithm for rectilinear minimal spanning trees. *J. ACM* 26, 177-182, 1979.

[13] R. M. Karp. Reducibility among combinatorial problems. In Miller and Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, New York, 85-103, 1972.

[14] M. Kaufmann and A. Z. Zelikovsky. A fast 11/8-approximation algorithm for the rectilinear Steiner tree problem. (to appear)

[15] K. Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Inf. Process. Lett.* 27, 125-128, 1988.

[16] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. Comput. System. Sci.* 4, 362-391, 1985.

[17] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japonica* 24, 573-577, 1980.

[18] A. C. Yao. On constructing minimum spanning trees in $k-$dimensional space and related problems. *SIAM J. Comput.* 11, 721-736, 1982.

[19] A. Z. Zelikovsky. The Steiner problem in graphs without optimal vertices. *Matemat. Issled.* 121, 44-49, 1991 (Russian).

[20] A. Z. Zelikovsky. An 11/6-approximation algorithm for the Steiner problem on graphs. In J.Neŝetr̂il and M.Fiedler (eds.) Fourth Czechoslovakian Symposium on Combinatorics, Graphs and Complexity.*Ann. Disc. Math.* 51, 351-354, 1992.

[21] A. Z. Zelikovsky. An 11/8-approximation algorithm for the Steiner problem on networks with rectilinear distance. In Sets, Graphs and Numbers, Budapest 1991. *Coll. Math. Soc. J. Bolyai* 60, 733-745, 1992.

[22] A. Z. Zelikovsky. A faster 11/6-approximation algorithm for the Steiner tree problem in graphs. *Tech. Rep. #MPI-92-122, Max-Planck-Institut für Informatik*, 1992.

A.Z.Zelikovsky
Institute of Mathematics,
Academy of Sciences of Moldova,
5 Academiei str.,
Chişinău, 277028, Moldova

Received September 15, 1992