Online Steiner Trees on Outerplanar Graphs

Akira Matsubayashi[1,a)]

Abstract: This report addresses the classical online Steiner tree problem on edge-weighted graphs. It is known that a greedy (nearest neighbor) online algorithm is $O(\log n)$-competitive on arbitrary graphs with n nodes. It is also known that no deterministic algorithm is better than $\Omega(\log n)$-competitive even for series-parallel graphs. The greedy algorithm is trivially 1- and 2-competitive for trees and rings, respectively, but $\Omega(\log n)$-competitive even for outerplanar graphs. No other nontrivial class of graphs that admits constant competitive deterministic Steiner tree algorithms has been known. In this report, we propose an 8-competitive deterministic algorithm for outerplanar graphs. Our algorithm connects a requested node to a previous node using a path that is constant times longer than a shortest path between the nodes.

Keywords: Steiner Tree, outerplanar graph, online algorithm, competitive analysis

1. Introduction

This report addresses the classical online Steiner tree problem on edge-weighted graphs. We are given a graph $G = (V_G, E_G)$ with non-negative edge-weights $w : E_G \to \mathbb{R}^+$ and a subset R of vertices of G. The (offline) Steiner tree problem is to find a *Steiner tree*, i.e., a subtree $T = (V_T, E_T)$ of G that contains all the nodes in R and minimizes its cost $c(T) = \sum_{e \in E_T} w(e)$. In the online version of this problem, vertices $r_1, \ldots, r_{|R|} \in R$ are revealed one by one, and for each $i \geq 1$, we must construct a tree containing r_i by growing the previously constructed tree for $r_1, \ldots, r_{i-1}$ (null tree for $i = 1$) without information of $r_{i+1}, \ldots, r_{|R|}$.

It is known that a greedy (nearest neighbor) online algorithm is $O(\log n)$-competitive on arbitrary graphs with n nodes [6]. It is also known that no deterministic algorithm is better than $\Omega(\log n)$-competitive even for series-parallel graphs [6]. The greedy algorithm is trivially 1- and 2-competitive for trees and rings, respectively, but $\Omega(\log n)$-competitive even for outerplanar graphs. No other nontrivial class of graphs that admits constant competitive deterministic Steiner tree algorithms has been known. As for randomized algorithms, a probabilistic embedding of outerplanar graphs into tree metrics with distortion 8, presented by Gupta, Newman, Rabinovich, and Sinclair [5], implies an 8-competitive online Steiner tree algorithm against oblivious offline adversaries. Various generalizations of the online Steiner tree problem are also studied, such as generalized STP [2], node-weighted STP [7], and asymmetric STP [1].

In this report, we propose an 8-competitive deterministic algorithm for outerplanar graphs. Our algorithm connects a requested node to a previous node using a path that is constant times longer than a shortest path between the nodes. An interesting application of the online steiner tree problem is the file allocation problem, in which we maintain a dynamic allocations of multiple copies of data file on a network with servicing online read/write requests. Bartal, Fiat, and Rabani [3] propose a file allocation algorithm based on any online Steiner algorithm. With this result, our result implies an $8(2 + \sqrt{3}) (\approx 29.86)$-competitive randomized Steiner tree algorithm against adaptive online adversaries.

[1] Division of Electrical Engineering and Computer Science, Kanazawa University Kakuma-machi, Kanazawa, 920–1192 Japan
[a)] mbayashi@t.kanazawa-u.ac.jp

2. Preliminaries

Let $G = (V_G, E_G)$ be a planar graph with non-negative edge-weights $w : E_G \to \mathbb{R}^+$. The *weak dual* of G is a graph $H = (V_H, E_H)$, where V_H is the set of bounded faces of G, and E_H is the set of two bounded faces F and F' that have a common edge. G is *outerplanar* if it can be drawn on the plane so that all the vertices belong to the unbounded face, or equivalently, if H is a forest [4].

Throughout the report, we assume that G is biconnected because a Steiner tree of G is the union of Steiner trees of biconnected components of G. This assumption implies that H is a tree. Moreover, we assume that G has no edge uv with $w(uv) \geq d_G(u, v)$, where $d_G(u, v)$ is the distance (i.e., the length of shortest path) of vertices u and v on G. This is justified because there exists a Steiner tree not containing such an edge.

3. Algorithm and Analysis

3.1 Algorithm α-Detour

Suppose that we are given an outerplanar graph $G = (V_G, E_G)$ with edge-weights $w : E_G \to \mathbb{R}^+$, and a sequence $r_1, r_2, \ldots, r_{|R|} \in R \subseteq V_G$. Our algorithm, denoted by α-Detour ($\alpha > 1$), constructs trees $T_1, T_2, \ldots$ as follows:

For the first vertex r_1, we construct the tree T_1 consisting of the single vertex r_1. We suppose that the weak dual $H = (V_H, E_H)$ of G is a tree rooted by a face containing r_1. For the ith vertex r_i with $r \geq 2$, α-Detour performs the following steps:

α-Detour

(1) Find a shortest path $P = (p_1, p_2, \cdots, p_{|P|})$ from a vertex p_1 in T_{i-1} to $p_{|P|} = r_i$.
(2) Let $T_i := T_{i-1}$.
(3) For $j = 1$ to $|V_P| - 1$, if $p_{j+1} \notin V_{T_i}$, then call Detour-edge(α, p_j, p_{j+1}) defined below.
(4) Return T_i.

Detour-edge(x, u, v) is a procedure with arguments $x \geq 1$ and an edge uv with $u \in V_{T_1}$, $v \notin V_{T_1}$, and $w(uv) \leq d_G(T_i, v)$. The procedure is defined as follows:

Detour-edge(x, u, v)

(1) If uv is an *outer edge*, i.e., an edge contained in the unbounded face, then add uv to T_i, and returrn.
(2) If uv is an inner edge, then it corresponds a face F and its child F' that have uv in common. Let G' be the subgraph of G induced by *descendant edges of uv in H*, i.e., edges contained in $F \setminus uv$ or the descendant faces of F in H.
(3) Find a shortest path $Q = (q_1, \ldots, q_{|Q|})$ in G' from a vertex q_1 in T_i to $q_{|Q|} = v$.
(4) If $c(Q)/w(uv) > x$, then add uv to T_i, where $c(Q)$ is the sum of weights of edges in Q.
(5) Otherwise, call Detour-edge($x \cdot w(uv)/c(Q), q_j, q_{j+1}$) for $j = 1$ to $|Q| - 1$.
(6) Return.

3.2 Correctness

Since α-Detour and Detour-edge only add edges to T_{i-1}, T_i contains T_{i-1} as a subgraph. Therefore, it suffices to show that α-Detour connects r_i to T_i.

Lemma 1 *Detour-edge(x, u, v) adds a path of length at most $x \cdot w(uv)$ that connects a vertex of T_i and v.*

Proof We prove this lemma by induction of the *level of uv* in H, where the level of uv is the distance in H between the roof and the face containing uv. If uv is an outer edge, then the procedure choose uv as a path connecting u and v. Therefore, this path has length $w(uv) \leq x \cdot w(uv)$.

Assume that uv is an inner edge, and that the lemma holds for a higher level than that of uv. If $c(Q)/w(uv) > x$ in Step 4, then the lemma is proved in a similar way to the case that uv is an outer edge. Otherwise, by induction hypothesis, Detour-edge($x \cdot w(uv)/c(Q), q_1, q_2$) adds a path of length at most $x \cdot w(uv)w(q_1q_2)/c(Q)$ that connects a vertex in T_i and q_2 in the subgraph of G induced by the descendant edges of q_1q_2. For $1 < j < |Q|$, q_jq_{j+1} is not a descendant edge of other edges of Q, because uv is an ancestor edge of all the edges of Q, and because Q connects both q_j and q_{j+1} to v with the shortest distance. Therefore, q_j is a unique vertex of T_i in the subgraph of G induced by the descendant edges of q_jq_{j+1}, and hence, Detour-edge($x \cdot w(uv)/c(Q), q_j, q_{j+1}$) adds a path of length at most $x \cdot w(uv)w(q_jq_{j+1})/c(Q)$ that connects q_j and q_{j+1}. Concatenating the paths for all $1 \leq j < |Q|$, we conclude that Detour-edge(x, u, v) adds a path of length at most $\sum_j (x \cdot w(uv)w(q_jq_{j+1})/c(Q)) = x \cdot w(uv)$ that connects a vertex in T_i and v. □

Since α-Detour calls Detour-edge(α, p_j, p_{j+1}) unless p_{j+1} has already contained in T_i, by Lemma 1, we have the following lemma:

Lemma 2 *For $i \geq 2$, α-Detour connects r_i to T_i with a path of length at most $\alpha \cdot d_G(T_{i-1}, r_i)$.*

3.3 Competitiveness

To analyze competitiveness of α-Detour, we introduce a forest structure among edges of G obtained by modifying H as the Steiner tree grows. Then, we subdivide a planar drawing of G according to the structure.

Forest Structure

Let P_i be the path P constructed in Step 1 of α-Detour for r_i. For the first and second vertices $p_1 = r_1$ and p_2 of P_2, we define F as the subtree of H rooted by p_1p_2. Thereafter, every time Detour-edge(x, u, v) is called, we perform the following: If uv is an ancestor of one or more subtrees of F, then we cut the links between the roots of the subtrees and the parents of the roots. This means that the subtrees become connected components in the updated F, and that uv has no descendant in the subtrees in F, while it may have a descendant in the subtrees in H. Moreover, if uv has no parent in F, then the new connected component rooted by uv and consisting of descendants of uv in H is added to F. Note that edges of Q constructed in Step 3 of Detour-edge are descendants of uv in F, and that the links between uv and the edges of Q will never be cut. The latter is because if uv and an edge of Q would be cut, then Detour-edge($\cdot, u', v'$) must be called for an edge $u'v'$ that is both an ancestor of the edge of Q and a descendant of uv. Since such u' and v' are contained in T_i, however, Detour-edge($\cdot, u', v'$) should never be called later.

Subdivision

Suppose that we finished constructing $T := T_{|R|}$. For every edge uv such that Detour-edge($\cdot, u, v$) is called in the construction of T, let D_{uv} be the set of edges in Q constructed in Step 3. We want to define a similar edge set D_{uv} for any inner edge uv in some $D_{u'v'}$ such that Detour-edge($\cdot, u, v$) is not called. For this purpose, we perform the following procedure for such an edge uv and define D_{uv} as the set of edges in Q constructed in Step 3 as well.

Extend-edge(u, v)

(1) If uv is an outer edge, then let Q be the path consisting of uv, and return.
(2) Let G' be the subgraph of G induced by descendant edges of uv in H.
(3) Find a shortest path Q in G' connecting;
   (a) u and v if $\{u, v\} \cap V_T = \emptyset$,
   (b) a vertex in the subtree of T in G' containing u, and v if $\{u, v\} \cap V_T = \{u\}$,
   (c) u, and a vertex in the subtree of T in G' containing v if $\{u, v\} \cap V_T = \{v\}$,
   (d) a vertex in the subtree of T in G' containing u, and a vertex in the subtree of T in G' containing v if $\{u, v\} \cap V_T = \{u, v\}$.
(4) Call Extend-edge(u', v') for each edge $u'v'$ in Q.

We regard each edge uv in G as a line segment $L(uv)$ of length $w(uv)$. A path Q is regarded as the concatenation $L(E_Q) :=$

$\bigcup_{e \in E_Q} L(e)$. If uv has D_{uv}, then we define a mapping m_{uv} that linearly maps $L(D_{uv})$ to $L(e)$, such that for any line segment $s \subseteq L(D_{uv})$, $m_{uv}(s)$ is a line segment in $L(uv)$ of length $w(uv)w(s)/w(D_{uv})$, where $w(s)$ is the length of s and $w(D_{uv}) = \sum_{e \in D_{uv}} w(e)$.

There are two possibilities that an inner edge $u'v'$ has no $D_{u'v'}$. One is that $u'v'$ has no ancestor in F. For this case we define $m_{uv}(s) := \emptyset$ for any edge uv and $s \subseteq L(u'v')$. The other case is that there exists an ancestor edge uv of $u'v'$ such that some edges $D^*_{u'v'} \subseteq D_{uv}$ are descendant of $u'v'$ in F, and that $w(u'v') > w(D_{uv})$. For this case, we define that m_{uv} linearly maps $L(u'v')$ to $m_{uv}(L(D^*_{u'v'}))$. Moreover, we define that $m_{u'v'}$ linearly maps $L(D^*_{u'v'})$ to $L(u'v')$.

We recursively extend m_e in such a way that $m_e(s) := m_e(m_{e'}(s))$ for any e with D_e, a descendant edge e' with $D_{e'}$ of e in F, and any line segment s on a descendant edge of e' in F.

Lemma 3 *For an edge e with D_e, $e' \in D_e$, and e'' with $e' \in D^*_{e''} \subseteq D_e$, it follows that $w(m_e(L(e'))) \leq w(e') \leq w(m_{e''}(L(e')))$.*

Proof By the definition of the shortest path in Step 3 of Detour-edge or Extend-edge, if $w(uv) > w(D_e)$, then D_e instead of uv should have been chosen in the parent procedure. Therefore, $w(uv) \leq w(D_e)$, implying that $w(m_e(L(e'))) \leq w(e')$. Similarly, if $w(e'') < w(D^*_{e''})$, then e'' instead of D^*_{uv} should have been chosen in the parent procedure. Therefore, $w(e'') \geq w(D^*_{e''})$, implying that $w(m_{e''}(L(e'))) \geq w(e')$. □

Lemma 4 *Suppose that $uv \in E_{P_i}$ for some i, and that $\bar{P}_i$ is the path connecting a vertex of T_i and v that is constructed by Detour-edge(α, u, v) in Step 3 of α-Detour. If e is a descendant edge of $e' \in E_{\bar{P}_i}$, then it follows that $w(e) > \alpha \cdot w(m_{uv}(L(e)))$.*

Proof By Lemma 3, it suffices to show the lemma for the case of $e \in D_{e'}$. We prove the lemma by induction on the number of recursive depths for Detour-edge(α, u, v) to output e'. If there is no recursive calls, then $uv = e'$ and $w(D_{uv})/w(u, v) > \alpha$, implying $w(e) > \alpha \cdot w(m_{uv}L(e))$. Assume that α-Detour(α, u, v) invoke recursive calls and that the lemma holds for the smaller number of recursive calls. From this assumption, Detour-edge is recursively called with $x = \alpha \cdot w(u, v)/w(D_{uv})$ and some edge $u'v' \in D_{uv}$. By induction hypothesis, we have $w(e) > x \cdot w(m_{u'v'}(L(e))) = (\alpha \cdot w(u, v)/w(D_{uv})) \cdot w(m_{u'v'}(L(e))) = \alpha \cdot w(m_{uv}(L(e)))$. Thus, we have the lemma. □

For any outer edge o, we define $S(o) = \bigcup_e m_e(L(o))$ overall ancestor edges of o in F. $S(o)$ contains line segments of a sequence of edges from the root to the leaf o of a connected component of F. Let e_o be the last edge in the sequence that is contained in P_i for some i.

Lemma 5 *Suppose that O is the set of edges of the path connecting $r, r' \in R$ on the unbounded face, and that no other vertex of R is contained in the path. Then, any path Z connecting r and r' has length at least $\sum_{o \in O} w(m_{e_o} L(o))$.*

Proof S is a partition of a planar drawing of G as well as a subdivision of edges of G. Therefore, for each $o \in O$, there is an edge $z \in E_X$ such that $m_e(o) \cap S(o) \neq \emptyset$. Either z is a descendant of e_o, or z is an ancestor of e_o. If z is a descendant of e_o, then Lemma 3 implies that $w(m_{e_o}(L(o))) \leq w(m_z(L(o)))$.

Suppose that $O' \subseteq O$ be the set of edges o such that z is an ancestor of e_o. Because no vertex of $R \setminus \{r, r'\}$ are between r and r', e_o must go into and out of $\bigcup_{o \in O'} S(o)$ using a part of a path P_i for some i. Since G is planar, and by the assumption that z is an ancestor of e_o, a part of Z passes outside of the path P_i. Since P_i is a shortest path, the length of part of P_i is at most the length of the part of Z. In this way, we can see that for any edges e_o with $o \in O'$, there are parts of P_is and parts of Z such that the length of parts of P_is are at most the length of parts of Z. Therefore, we have the lemma. □

Lemma 6 *For any $o \in O$, it follows that*

$$\sum_{e \in \bigcup_i E_{P_i}, L(e) \cap S(o) \neq \emptyset} w(m_e(L(o))) < \frac{\alpha}{\alpha - 1} w(m_{e_o}(L(o))).$$

Proof By Lemma 4, for any edge $e \in \bigcup_i E_{P_i}$ and its descendant $e' \in \bigcup_i E_{P_i}$ in F, it follows that $w(e') > \alpha \cdot w(m_e(L(e')))$. This implies that if $e_1 = e_o, \ldots, e_k$ are the sequence of edges in $\bigcup_i E_{P_i}$ in the order from the leaf o to the root of a connected component of F, it follows that $w(m_{e_{j+1}}(L(o))) > \alpha \cdot w(m_{e_j}(L(o)))$. Solving the recurrence, we have the lemma. □

Lemma 7 *For any tree Z containing vertices R, $c(T)/c(Z) < 2\alpha^2/(\alpha - 1)$.*

Proof Suppose that $O_1, \ldots, O_{|R|}$ be the set of edges of paths on the unbounded face of G such that the ends of each O_j are nodes of R. Let Z_j be the subpath of Z connecting ends of O_j. Then, it follows from that

$$\begin{aligned}
c(T) &\leq \alpha \sum_{o \in O} \sum_{e \in \bigcup_i E_{P_i}, L(e) \cap S(o) \neq \emptyset} w(m_e(L(o))) \quad \text{(Lemma 1)} \\
&< \frac{\alpha^2}{\alpha - 1} \sum_{o \in O} w(m_{e_o}(L(o))) \quad \text{(Lemma 6)} \\
&\leq \frac{\alpha^2}{\alpha - 1} \sum_j c(Z_j) \quad \text{(Lemma 5)} \\
&\leq \frac{2\alpha^2}{\alpha - 1} c(Z).
\end{aligned}$$

□

Setting $\alpha = 2$, we have the following theorem:

Theorem 8 *Algorithm 2-Detour is 8-competitive.*

4. Concluding Remarks

Previously known lower bounds to be applied to outerplanar graphs is 2 for rings. We will present a lower bound of 4 in the future version of this report. We believe that the competitive ratio of 8 of our algorithm can be still improved, probably, to 4.

References

[1] Angelopoulos, S.: On the Competitiveness of the Online Asymmetric and Euclidean Steiner Tree Problems, *WAOA 2009*, pp. 1–12 (2010).
[2] Averbuch, B., Azar, Y. and Bartal, Y.: On-Line Generalized Steiner Problem, *Theoret. Comput. Sci.*, Vol. 324, pp. 313–324 (2004).
[3] Bartal, Y., Fiat, A. and Rabani, Y.: Competitive Algorithms for Distributed Data Management, *J. Comput. Sys. Sci.*, Vol. 51, No. 3, pp. 341–358 (1995).
[4] Fleischner, H. J., Geller, D. P. and Harary, F.: Outerplanar Graphs and Weak Duals, *J. Indian Math. Soc.*, Vol. 38, pp. 215–219 (1974).
[5] Gupta, A., Newman, I., Rabinovich, Y. and Sinclair, A.: Cuts, Trees, and ℓ_1-Embedding of Graphs, *Combinatorica*, Vol. 24, No. 2, pp. 233–269 (2004).
[6] Imase, M. and Waxman, B. M.: Dynamic Steiner Tree Problem, *SIAM J. Discrete Math.*, Vol. 4, No. 3, pp. 369–384 (1991).
[7] Naor, J. S., Panigrahi, D. and Singh, M.: Online Node-Weighted Steiner Tree and Related Problems, *Proc. 52nd Annual Symposium on Foundations of Computer Science*, pp. 210–219 (2011).