Non-greedy Online Steiner Trees on Outerplanar Graphs

Akira Matsubayashi(✉)

Division of Electrical Engineering and Computer Science, Kanazawa University, Kanazawa 920-1192, Japan
mbayashi@t.kanazawa-u.ac.jp

Abstract. This paper addresses the classical online Steiner tree problem on edge-weighted graphs. It is known that a greedy (nearest neighbor) online algorithm has a tight competitive ratio for wide classes of graphs, such as trees, rings, any class including series-parallel graphs, and unweighted graphs with bounded diameter. However, we did not know any greedy or non-greedy tight deterministic algorithm for other classes of graphs. In this paper, we observe that a greedy algorithm is $\Omega(\log n)$-competitive on outerplanar graphs, where n is the number of vertices, and propose a 5.828-competitive deterministic algorithm on outerplanar graphs. Our algorithm connects a requested vertex and the tree constructed thus far using a path that is constant times longer than the distance between them. The algorithm can be applied to a 21.752-competitive file allocation algorithm against adaptive online adversaries on outerplanar graphs. We also present a lower bound of 4 for arbitrary deterministic online Steiner tree algorithms on outerplanar graphs.

1 Introduction

This paper addresses the classical online Steiner tree problem (STP) 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 vertices 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 constructed tree for $r_1, \ldots, r_{i-1}$ (null tree for $i = 1$) without information of $r_{i+1}, \ldots, r_{|R|}$.

Imase and Waxman [12] proposed a greedy (nearest neighbor) online algorithm that is $O(\log n)$-competitive on arbitrary graphs with n vertices. They also proved that no deterministic algorithm is $o(\log n)$-competitive even on series-parallel graphs [12]. Westbrook and Yan [15] refined these upper and lower bounds to $\Theta(\log(\mathrm{diam}|R|/\textsc{opt}))$ with improving analysis, where diam is the diameter of the underlying graphs, and $\textsc{opt}$ is the cost of a minimum Steiner tree. The refined upper bound implies that the greedy algorithm is $O(\log \mathrm{diam})$-competitive for unweighted graphs. The greedy algorithm is trivially 1- and 2-competitive on trees and rings, respectively. With these results, the greedy algorithm has a tight competitive ratio for trees, rings, any class of graphs including

K. Jansen and M. Mastrolilli (Eds.): WAOA 2016, LNCS 10138, pp. 129–141, 2017.
DOI: 10.1007/978-3-319-51741-4_11

series-parallel graphs, and unweighted graphs with bounded diameter. However, we did not know any greedy or non-greedy tight deterministic algorithm for other classes of graphs. As for randomized algorithms, a probabilistic embedding of outerplanar graphs into tree metrics with distortion 8, presented by Gupta et al. [11], implies an 8-competitive online Steiner tree algorithm against oblivious adversaries on outerplanar graphs. This embedding was generalized to k-outerplanar graphs with distortion 200^k by Chekuri et al. [9], implying a 200^k-competitive online Steiner tree algorithm against oblivious adversaries on k-outerplanar graphs.

Online Steiner trees on the Euclidean space are studied in [1,4]. Various generalizations of the online Steiner tree problem are also studied, such as generalized STP [5,8,15], asymmetric STP [3,4], priority STP [2,3], and vertex-weighted STP [14].

In this paper, we observe that the greedy algorithm is $\Omega(\log n)$-competitive on outerplanar graphs, and propose a $3+2\sqrt{2} \approx 5.828$-competitive deterministic algorithm on outerplanar graphs. Our algorithm connects a requested vertex and the tree constructed thus far using a path of a maximal length within $\alpha > 1$ times longer than the distance between them. We prove that our algorithm is $(\frac{\alpha}{\alpha-1} + 2\alpha)$-competitive, implying a $(3 + 2\sqrt{2})$-competitive algorithm at $\alpha = 1 + 1/\sqrt{2}$ (Sect. 4). A technical overview will be discussed in the beginning of Sect. 4 after we observe the $\Omega(\log n)$ competitiveness of the greedy algorithm in Sect. 3. Though we do not know if our analysis is tight for $\alpha = 1 + 1/\sqrt{2}$, we observe a lower bound for our algorithm that matches our analysis with any $\alpha \geq 2$. We also present a lower bound of 4 for arbitrary deterministic online Steiner tree algorithms on outerplanar graphs (Sect. 5). Previous and our results are summarized in Table 1.

An application of the online Steiner tree problem is the *file allocation problem*, which is to find dynamic allocations of multiple copies of a data object, called *file*, of size $D \geq 1$ on a network G, such that the total cost of servicing online read/write requests and reallocating the copies is minimized [6,7,13]. Bartal, Fiat, and Rabani [7] proposed a $(2 + \sqrt{3})c$-competitive file allocation algorithm

Table 1. Summary of results of online STP on weighted graphs

Graphs	Competitive ratio	Adversary type	
General graphs	$O(\log n)$	Deterministic	[12]
Series-parallel graphs	$\Omega(\log n)$	Deterministic	[12]
General graphs	$\Theta(\log(\mathrm{diam}\lvert R\rvert/\textsc{opt}))$	Deterministic	[15]
Outerplanar graphs	5.828	Deterministic	This paper
Outerplanar graphs	≥ 4	Deterministic	This paper
Rings	2	Deterministic	[7]
Outerplanar graphs	8	Oblivious	[11]
k-outerplanar graphs	200^k	Oblivious	[9]

against adaptive online adversaries based on any c-competitive online Steiner tree algorithm against adaptive online adversaries. Combined with this result, our algorithm implies a $(2+\sqrt{3})(3+2\sqrt{2}) \approx 21.752$-competitive file allocation algorithm against adaptive online adversaries on outerplanar graphs.

2 Preliminaries

Graphs considered here are undirected and have non-negative edge-weights, $w(e) \geq 0$ for any edge e. For a graph G, we denote its vertex set and edge set by V_G and E_G, respectively. We use the notation of w also for graphs, i.e., $w(G) := \sum_{e \in E_G} w(e)$. For a subset R of vertices of G, a *Steiner tree of G for R* is a subtree T of G such that $R \subseteq V_T$. T is said to be *minimum* if T has the minimum cost $w(T)$ overall Steiner trees of G for R.

Suppose that G is a planar graph. The *weak dual* of G is a graph H such that V_H is the set of bounded faces of G, and E_H is the family of sets consisting of two bounded faces that have a common edge. The graph 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 [10]. We say an edge of G to be *outer* if the edge is contained in the unbounded face, *inner* otherwise.

In the rest of the paper, we assume that G is a biconnected outerplanar graph, because finding a minimum Steiner tree of G can easily be reduced to finding minimum Steiner trees of biconnected components of G. This assumption implies that H is a tree. Let $d_G(u,v)$ be the distance of vertices u and v in G. We use the notation of d_G also for the distance between a graph and a vertex, i.e., $d_G(G',v) := \min\{d_G(u,v) \mid u \in V_{G'}\}$ for a subgraph G' of G and $v \in V_G$.

3 Lower Bound for Greedy Algorithms

In this section, we prove that a greedy algorithm, which always takes a shortest path between a requested vertex and the current tree, is $\Omega(\log n)$-competitive on outerplanar graphs. The proof will be a hint of our algorithm and general lower bound in the following sections. We note that the lower bound is also admitted by another type of greedy algorithm that always takes a shortest path between the current request and one of the previously requested vertices.

Theorem 1. *For any integer $k \geq 0$, there exists a (2^k+1)-vertex outerplanar graph G_k such that if a greedy online Steiner tree algorithm on G_k is ρ-competitive, then $\rho \geq 1 + k/2$.*

Proof. G_k is recursively defined: G_0 consists of two vertices joined by an edge of weight 1. These vertices and edge are said to be of *level* 0. For $i \geq 1$, G_i is obtained from G_{i-1} by adding a new vertex u and edges su and ut of weight $2^{-i} + \epsilon$ for each edge st of level $i-1$, where $\epsilon > 0$. The added vertices and edges are said to be of *level i*. We illustrate G_4 in Fig. 1.

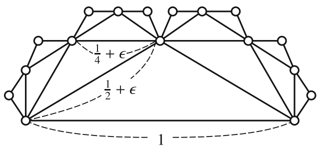

Fig. 1. Outerplanar graph G_4

If all the vertices of G_k are requested in an increasing order with regard to their levels, then a greedy algorithm chooses the unique edge of level 0 first and then 2^{i-1} edges in each level $1 \leq i \leq k$. On the other hand, a minimum Steiner tree has a cost at most that of the tree consisting only of edges of level k. Therefore, the competitive ratio is at least $\frac{1+\sum_{i=1}^{k} 2^{i-1}(2^{-i}+\epsilon)}{2^k(2^{-k}+\epsilon)}$, which tends to $1 + k/2$ as $\epsilon \to 0$. □

4 Algorithm α-Detour and Its Competitiveness

In this section, we define our algorithm, called α-Detour with $\alpha > 1$, and prove its competitiveness.

4.1 Overview

The basic idea of our algorithm is to suppress the cost of a greedy algorithm against the adversary in the proof of Theorem 1. Suppose that the greedy algorithm takes an edge of level i of G_k in the proof of Theorem 1. Then, the vertices incident to the edge can also be connected using edges with level higher than i. For ϵ close to 0, the edge of level i and the detour have nearly the same length. The greedy algorithm incurs an expensive cost of $O(\log n)$ since it takes edges of each level even in such a case. But obviously, we can avoid such an expensive cost by taking the detour with a single penalty of the detour. In our algorithm α-Detour, we take a certain detour of maximal length within the factor of α using edges of higher level. We formally define α-Detour and prove its correctness in Sects. 4.2 and 4.3, respectively. We will introduce in the definition a rooted forest structure of edges with regard to their levels. In G_k, for example, edges su and ut of level i added to an edge st of level $i-1$ are children of st.

In our analysis of the competitiveness, intuitively (not precisely), we charge the weight of each edge uv chosen by α-Detour to the path connecting u and v in a minimum Steiner tree. For an edge e of the minimum Steiner tree, edges charged to e are of three types: (i) ancestor edges of e, (ii) descendant edges of e or e itself, and (iii) otherwise. The amounts charged to e by edges of type (i) are essentially fragments of weights exponentially decreasing by the factor α^{-1}. Hence, the total charged amount of this type is at most $\sum_{i\geq 1} \alpha^{-(i-1)}w(e) < \frac{\alpha}{\alpha-1}w(e)$. For each of types (ii) and (iii), by the property of detours based on shortest paths, the total charged amount is at most $w(e)$ multiplied by α. Summing these amounts, we derive that α-Detour is $(\frac{\alpha}{\alpha-1}+2\alpha)$-competitive as desired. We formally prove this in Sect. 4.4. One non-intuitive part of the proof is that we charge an edge uv chosen by α-Detour to not the whole but only a part of the path joining uv in the minimum Steiner tree. Specifically, we do not charge uv to any part of the tree that was constructed by α-Detour before uv is chosen. The charging process is defined through dynamically modifying the graph and forest structure of edges, in such a way that the forest precisely represents the relation between shortest paths and their detours.

4.2 Definition

For the first requested vertex r_1, we suppose that the weak dual H of G is a tree rooted by a face containing r_1. We introduce a forest F with $V_F = E_G$ as follows. If a face C is the root of H, then all the edges of C are the roots of the connected components of F. If C is a face of G, and C' is a child of C in H, then all the edges in $E_{C'} \setminus E_C$ are the children of the unique edge $e \in E_C \cap E_{C'}$ in F. For any inner edge e of G, let G_e be the subgraph of G induced by the descendant edges of e in F. We note that G_e does not contain e. To clarify our discussion, we use the term *links* to denote elements of E_F.

For the ith requested vertex r_i, α-Detour constructs a tree T_i as follows:

α-Detour

```
1. If $i = 1$, then return the tree $T_1$ consisting only of $r_1$.
2. Suppose $i \geq 2$. If $r_i \in V_{T_{i-1}}$, then return $T_i := T_{i-1}$.
3. Otherwise, find a shortest path $P_i = (p_1, p_2, \ldots, p_{|P_i|})$ between a vertex $p_1$ in
   $T_{i-1}$ and $p_{|P_i|} = r_i$.
4. For $j = 1$ to $|P_i|-1$, if $p_{j+1} \notin V_{T_{i-1}}$, then call DetourEdge($\alpha, p_j, p_{j+1}$) defined
   below.
5. Return $T_i := T_{i-1}$.
```

DetourEdge(β, u, v) is a procedure to modify T_{i-1} by adding a path between T_{i-1} and v of maximal length at most $\beta \cdot w(uv)$. The inputs are $\beta \geq 1$ and an edge uv such that $u \in V_{T_{i-1}}$, $v \notin V_{T_{i-1}}$, and $w(uv) \leq d_G(T_{i-1}, v)$. The procedure is formally defined as follows:

DetourEdge(β, u, v)

```
1. If $uv$ is outer, then add $uv$ to $T_{i-1}$ and return.
2. If $uv$ is inner, then find a shortest path $Q_{uv} = (q_1, \ldots, q_{|Q_{uv}|})$ from a vertex
   $q_1$ in $T_{i-1}$ to $q_{|Q_{uv}|} = v$ in $G_{uv}$.
3. If $w(Q_{uv})/w(uv) > \beta$, then add $uv$ to $T_{i-1}$.
4. Otherwise, call DetourEdge($\beta \cdot w(uv)/w(Q_{uv}), q_j, q_{j+1}$) for $j = 1$ to $|Q_{uv}|-1$.
5. Return.
```

4.3 Correctness

Since α-Detour and DetourEdge 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-1} with a path of length at most $\alpha \cdot d_G(T_{i-1}, r_i)$.

Lemma 1. *DetourEdge(β, u, v) adds a path of length at most $\beta \cdot w(uv)$ between a vertex of T_{i-1} and v.*

Proof Sketch. Induction on the height of uv in F. □

Since α-Detour calls DetourEdge(α, p_j, p_{j+1}) for each j unless p_{j+1} has already been contained in T_{i-1}, by Lemma 1, we have the following lemma:

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

4.4 Competitiveness

We first introduce dynamic modification of G and F in such a way that any Q_{uv} found in DetourEdge(β, u, v) is a path connecting u and v. According to the modified graph and forest, we then charge the weights of edges of P_i in Step 3 of α-Detour to their descendant edges that are potentially used by a minimum Steiner Tree. We finally estimate the competitiveness by comparing the charged amounts multiplied by α, including the case that edges of P_i are charged not to their descendants, to the cost of the minimum Steiner Tree.

Modifying Graph. Every time DetourEdge(α, p_j, p_{j+1}) is called in Step 4 of α-Detour, we mark $p_j p_{j+1}$ "greedy". Before entering DetourEdge(α, p_j, p_{j+1}), we perform the following:

ModifyGraph

1. For each "greedy" edge pp' such that $p_j p_{j+1}$ is an ancestor of pp', and that there is no "greedy" edge that is a descendant of $p_j p_{j+1}$ and an ancestor of pp', we decompose the graph into the subgraph induced by pp' and its all descendants and the subgraph induced by other edges. We then contract the latter subgraph by identifying p and p'. If there is a self-loop of the identified vertex, then we remove it.
2. According to the modified graph, we modify the forest structure as well. I.e., we remove the link of the forest between pp' and its parent. This yields a subtree rooted by pp'. We then remove from the forest any self-loop of the identified vertex.

We note that if we performed ModifyGraph before calling DetourEdge (α, p_j, p_{j+1}) in Step 4 of α-Detour, then DetourEdge(α, p_j, p_{j+1}) would choose the same edges of the path between the current Steiner tree and p_{j+1}. Moreover, the path P_i chosen in Step 3 of α-Detour is not affected either. These are because we decompose or contract the graph only at "greedy" edges whose end-vertices are already contained in the Steiner tree. Therefore, we may discuss as if α-Detour and DetourEdge were performed with modifying the graph. We observe some properties related to the modification of the graph as stated in the following lemmas. To clarify our discussion, we use G and F to denote the initial graph and forest before processing r_1, respectively, and G^* and F^* to denote the final graph and forest after processing $r_1, \dots, r_{|R|}$, respectively. We also use G^*_e just as defined for G.

Lemma 3. *For any edge uv such that DetourEdge(β, u, v) is called, Q_{uv} is a path connecting u and v in the modified graph at the point that DetourEdge(β, u, v) is processed.*

Proof Sketch. Previous ModifyGraph for some "greedy" edge makes the current Steiner tree into a single vertex in the subgraph induced by descendant edges of the "greedy" edge. After that, any DetourEdge for an edge e constructs a path using only descendant edges of e. Since uv is not a descendant of such e, this means that when DetourEdge(β, u, v) is processed, u is the unique vertex of the current Steiner tree in the subgraph induced by the descendants of uv. □

Lemma 4. *For any edge uv such that DetourEdge(β, u, v) is called, uv and edges of Q_{uv} are contained in the same connected component of G^*.*

Proof Sketch. By Lemma 3, Q_{uv} connects u and v at the point that DetourEdge(β, u, v) is processed. After that, therefore, any edge that is a descendant of uv and an ancestor of an edge of Q_{uv} cannot be "greedy". □

Lemma 5. *For any edge uv such that DetourEdge(β, u, v) is called, Q_{uv} is a shortest path between u and v in G^*_{uv}.*

Proof Sketch. By Lemma 4, uv and Q_{uv} are contained in the same connected component of G^*. If there is a path Q' between u and v in G^*_{uv} shorter than Q_{uv}, then ModifyGraph for some "greedy" descendant pp' of uv must shorten Q' so that $w(Q') < w(Q_{uv})$. This means that before this ModifyGraph, Q' contained a subpath between p and p' with descendant edges of pp', some of which are removed by the ModifyGraph. However, since pp' is "greedy", the resulting subpath cannot be shorter than pp'. This means that ModifyGraph does not make Q' shorter than Q_{uv}. □

By Lemma 5, we immediately have the following:

Lemma 6. *For any edge e' in Q_e for some edge e, $w(e')$ is at most the distance between the end-vertices of e' in $G^*_{e'}$.*

Charging Weights. For a "greedy" edge e, we define the amount charged to any descendant e' of e in F^* as $w(e)$ multiplied by a factor $f_{e \to e'}$. Essentially, $f_{e \to e'}$ is defined as the ratio $w(e')/w(Q_e)$ for e' in Q_e. Moreover, we define the factor to be transitive, i.e., $f_{e \to e'} = f_{e \to e''} \cdot f_{e'' \to e'}$ if $f_{e \to e''}$ and $f_{e'' \to e'}$ are defined. To extend this definition to any e', we extend the notion of Q_e to any edge e.

We formally define the factor as follows: For any edge e in G^*, let S_e be a shortest path connecting the end-vertices of e in G^*_e if e is inner, e itself otherwise. We note that if e has Q_e, then $w(Q_e) = w(S_e)$ by Lemma 5. Suppose that e and e' are an edge and its descendant in F^*, respectively. If e' is in S_e, then let $f_{e\to e'} := w(e')/w(S_e)$. If e' is an ancestor of an edge of S_e, then let $f_{e\to e'} := w(S_{e'})/w(S_e)$. We note that $S_{e'}$ is a subpath of S_e in this case. If e' is a descendant of an edge of S_e, then there is a sequence of edges $e_1 = e, e_2, \dots, e_h = e'$ such that e_{i+1} is in S_{e_i} for $1 \le i \le h-2$, and either e_h is in $S_{e_{h-1}}$ or S_{e_h} is a subpath of $S_{e_{h-1}}$. For such e', we define $f_{e\to e'} := \prod_{i=1}^{h-1} f_{e_i \to e_{i+1}}$. In addition, we define $f_{e\to e} := 1$.

Lemma 7. *For any edge e and a path P connecting the end-vertices of e in G^*_e, it follows that $\sum_{e' \in E_P} f_{e\to e'} = 1$.*

Proof Sketch. Induction on the height of e in F^*. □

Lemma 8. *For any edge e and its descendant e' in F^*, $f_{e\to e'} \le w(e')/w(S_e)$.*

Proof Sketch. By the definition of $f_{e\to e'}$ and Lemma 6. □

Lemma 9. *Suppose that uv is a "greedy" edge, and that D_{uv} is the path between the Steiner tree and v constructed by DetourEdge(α, u, v) in Step 4 of α-Detour. If e is an edge in $G^*_{e'}$ for an edge e' in D_{uv}, then $w(e) > \alpha f_{uv\to e} w(uv)$.*

Proof Sketch. We prove the lemma by induction on the number of recursive levels for DetourEdge(α, u, v) to output e'. For the base case, i.e., $uv = e'$, by Lemmas 8 and 5 and $w(Q_{uv})/w(uv) > \alpha$, we can obtain $w(e) > \alpha f_{uv\to e} w(uv)$. For an induction step, assuming DetourEdge(β, u', v') called with $\beta = \alpha \cdot w(uv)/w(Q_{uv})$ for some edge $u'v'$ in Q_{uv}, we can obtain $w(e) > \alpha f_{uv\to e} w(uv)$ by Lemma 5. □

Comparison to Minimum Steiner Tree. Suppose that Z is any Steiner tree for R in G. Our aim is to decompose G into subgraphs according to Z, associate "greedy" edges with the decomposed subgraphs, and to estimate the amount charged to the edges of Z by "greedy" edges in each decomposed subgraph.

Specifically, for any edge e of Z, we decompose G into the subgraph induced by e and its descendant in F and the subgraph induced by edges that are not descendants of e in F. Decomposing G by all edges of Z, we obtain a set $\mathcal{B}$ of outerplanar subgraphs of G, each of which has edges of Z only in its unbounded face. For a subgraph $B \in \mathcal{B}$, B has either at most one edge or the all edges of the root face of G. If B has at most one edge of the root face, then B has a root edge e_B in B, i.e., an ancestor of all the other edges of B in F. We note that Z has e_B. If B has the entire root face, then for convenience, we suppose that Z has a null edge $e_B = r_1 r_1$ with the weight of 0, and that e_B is the parent of the other edges in the root face. I.e., we suppose that e_B is an ancestor of all the other edges of B also in this case. Let Z_B be the path induced by $E_B \cap E_Z$.

We associate a "greedy" edge uv with B if $E_B \setminus E_Z \cup \{e_B\}$ contains an edge of D_{uv}, where D_{uv} is the path between the Steiner tree and v constructed by DetourEdge(α, u, v). A "greedy" edge is said to be *open in* B if the edge is associated with B, and is an outer edge in $E_B \setminus E_Z$ or an ancestor of an outer edge in $E_B \setminus E_Z$ in F^*. A "greedy" edge associated with B and not open in B is said to be *closed in* B. In other words, all the outer edges of B that are descendants in F^* of a "greedy" edge closed in B are contained in Z.

Lemma 10. *For any edge z in $Z_B \setminus e_B$, the total amount charged to z by "greedy" edges associated with B is less than $w(z)/(\alpha - 1)$.*

Proof Sketch. Let $e_1, \ldots, e_h$ be the "greedy" edges associated with B such that in F^*, e_i is an ancestor of e_{i+1} for $1 \leq i < h$, and e_h is an ancestor of z. By Lemma 9, we can prove that the amount charged to z by e_i is less than $\frac{w(z)}{\alpha^{h-i+1}}$. Summing this overall i, we have the lemma. □

Lemma 11. *Let X_B be the set of "greedy" edges open in B. If there is $x \in X_B \setminus E_B$, then $w(X_B \setminus x) + f_{x \to e_B} w(x) \leq w(Z_B)$. Otherwise, $w(X_B) \leq w(Z_B)$.*

Proof. Let $X_B = \{e_1, \ldots, e_{|X_B|}\}$, and suppose that for each $i \geq 1$, e_i is marked "greedy" earlier than e_{i+1} is. We first assume that $X_B \setminus E_B = \emptyset$. When e_1 is marked "greedy", e_1 is contained in a shortest path P from the current Steiner tree to a request vertex. Since e_1 is the first open edge marked "greedy", e_1 is incident to at least one vertex s_1 of Z_B. For otherwise, the current Steiner tree has a vertex not incident to an edge of Z_B, which implies that there must be an edge open in B and marked "greedy" earlier than e_1. Because P must reach the request vertex, which is contained in Z_B, we can find the vertex $t_1 \in V_{Z_B} \cap V_P \setminus s_1$ nearest to s_1 on P. We note that the subpath P^1 of P between s_1 and t_1 consists only of open edges, say $e_1, \ldots, e_j$. This is justified through observing that e_h $(1 < h \leq j)$ cannot be closed if e_{h-1} is open and the vertex incident to both e_h and e_{h-1} is not in Z_B. We charge the weights of $e_1, \ldots, e_j$ to the subpath Z_B^1 of Z_B between s_1 and t_1. We note that since P is a shortest path, the total charged amount is at most $w(Z_B^1)$.

We continue the similar process for the remaining edges of X_B. I.e., we find the vertex t_2 in Z_B nearest to e_{j+1} on a shortest path P' from the current Steiner tree to a request vertex. One exception is that since e_{j+1} is not the first open edge in X_B, e_{j+1} may join two vertices in $V_B \setminus V_{Z_B}$. In this case, we set s_2 to the vertex in $\{s_1, t_1\}$ that is closer to t_2. If e_{j+1} is incident to a vertex in Z_B, then we set s_2 to the vertex as done for s_1. We charge the weights of the open edges in the subpath P^2 of P' from e_{j+1} to t_2 to the subpath Z_B^2 of Z_B between s_2 and t_2. We note again that P^2 consists only of open edges, and that the total charged amount is at most $w(Z_B^2)$. Moreover, Z_B^1 and Z_B^2 are edge-disjoint. For otherwise, s_2 or t_2 is in $V_{Z_B^1} \setminus \{s_1, t_1\}$. In either case, e_{j+1} joins two vertices of the cycle formed by P^1 and Z_B^1. By the definition of ModifyGraph, all the edges of P^1 that are descendants of e_{j+1} in F are identified to a single vertex in the subgraph of G^* containing e_{j+1}. These situations imply that e_{j+1} is neither an outer edge in $E_B \setminus E_Z$ nor an ancestor of such an outer edge in F^*, contradicting that e_{j+1} is open in B. Repeating this process, we can charge all the edges in X_B to Z_B in such a way that $w(X_B) \leq w(Z_B)$.

We then assume that there is $x \in X_B \setminus E_B$. Such x is an ancestor of e_B and unique. We process $e_1, e_2, \ldots$ as describe above, except that we charge $f_{x \to e_B} w(x)$ to e_B for x, i.e., e_B is the subpath Z_B^i for some i when processing x. We observe that the subpaths $Z_B^1, \ldots, Z_B^{i-1}$ do not contain e_B. For otherwise, there is a subpath $P^{i'}$ with $i' < i$ such that $Z_B^{i'}$ contains e_B. When x is marked "greedy", $P^{i'}$ is identified to a single vertex in the subgraph of G^* containing x. This implies that x is not open in B. We also observe that the subpaths $Z_B^{i+1}, \ldots$ do not contain e_B. This is because the assumption that x is associated with B implies that the end-vertices of e_B are added to α-Detour's Steiner tree when x is detoured. Thus, we can charge all the edges in X_B to Z_B in such a way that $w(X_B \setminus x) + f_{x \to e_B} w(x) \leq w(Z_B)$. □

Lemma 12. *It follows that* $w(T_{|R|}) < (\alpha/(\alpha-1) + 2\alpha) w(Z)$.

Proof Sketch. We can observe that each "greedy" edge e is associated with at least one subgraph B in $\mathcal{B}$. Moreover, if e is open in B, then $w(e)$ is directly charged to Z_B as described in Lemma 11. If e is closed in B, then $w(e)$ is fully charged to Z_B by Lemma 7. Thus, if we denote the set of "greedy" edges closed in B by Y_B, then it follows from Lemma 2 that

$$w(T_{|R|}) \leq \alpha \sum\nolimits_{B \in \mathcal{B}} \Bigg[\sum\nolimits_{e \in Y_B} w(e) + \begin{cases} \sum_{e \in X_B \setminus x} w(e) + f_{x \to e_B} w(x) & \text{if } \exists x \in X_B \setminus E_B \\ \sum_{e \in X_B} w(e) & \text{otherwise} \end{cases} \Bigg].$$

Applying Lemmas 10 and 11,

$$\begin{aligned} w(T_{|R|}) &\leq \alpha \sum\nolimits_{B \in \mathcal{B}} \left[\sum_{z \in E_{Z_B} \setminus e_B} \frac{w(z)}{\alpha - 1} + w(Z_B) \right] \\ &= \alpha \sum\nolimits_{B \in \mathcal{B}} \left[\frac{w(Z_B \setminus e_B)}{\alpha - 1} + w(Z_B \setminus e_B) + w(e_B) \right] \\ &= \alpha \sum\nolimits_{B \in \mathcal{B}} \left[\left(\frac{1}{\alpha - 1} + 1 \right) w(Z_B \setminus e_B) + w(e_B) \right] \\ &= \left(\frac{\alpha}{\alpha - 1} + 2\alpha \right) w(Z). \end{aligned}$$

□

Setting $\alpha = 1 + 1/\sqrt{2}$, we have the following theorem.

Theorem 2. *Algorithm* $(1 + 1/\sqrt{2})$*-Detour is* $3 + 2\sqrt{2} \approx 5.828$*-competitive.*

We do not know if the upper bound of 5.828 is tight. However, our analysis of Lemma 12 is tight for $\alpha \geq 2$.

Theorem 3. *For any* $\alpha > 1$*, there exists an outerplanar graph* G_α *such that if* α*-Detour is* ρ*-competitive on* G_α*, then* $\rho \geq \min\{3\alpha, \alpha/(\alpha - 1) + 2\alpha\}$.

5 Lower Bound for Arbitrarily Algorithms

In this section, we prove a lower bound of 4 for any deterministic Steiner tree algorithm on outerplanar graphs.

Overview. Our idea is based on Theorem 1. We recursively define a class of outerplanar graphs just as done in the proof of the theorem, except that arbitrarily many vertices and edges of level i are added to an edge of level $i-1$. Our adversary generates requests in phases; in the ith phase, vertices along the children of the online algorithm's tree up to the $(i-1)$st phase are requested. An online algorithm possibly chooses detours with (variable) factor α. The key of our proof is to define a sequence of upper bounds γ_i of α to be ρ-competitive at the ith phase. In fact, we can prove that for any $\rho < 4$, there is i such that $\gamma_i = 1$. This means that if there is a ρ-competitive algorithm with $\rho < 4$, then it tends to a greedy algorithm; however, this is impossible.

Definition of Graph. Let m be a positive integer and ϵ be a positive real number. Let G_0 be a path consisting of a single edge of weight 1. The unique edge of G_0 is said to be of level 0. For $i \geq 1$, let G_i be the graph obtained from G_{i-1} by adding m^i edges of weight $(1+\epsilon)^i / \prod_{j=1}^{i} m^j$ to each edge of level $i-1$ in such a way that the added m^i edges form a path connecting the end-vertices of the edge of level $i-1$. All the added edges are said to be of level i. We suppose $G := G_i$ with sufficiently large i. We define F as the rooted tree with $V_F = E_G$ such that for an edge e of level $i-1$, m^i edges added to e are children of e in F. We note that such children has the total weight of $(1+\epsilon)w(e)$.

Adversary. We use a sequence K_i for $i \geq 0$ defined as follows: Let $K_0 := 1$ and K_1 be less than but sufficiently close to 3. For $i \geq 1$, we define $K_{i+1} := (K_0 + K_1)(K_i - K_{i-1})$ if $K_i < (K_0 + K_1)(K_i - K_{i-1})$, and $K_{i+1} := K_i$ if $K_i \geq (K_0 + K_1)(K_i - K_{i-1})$.

Our adversary ADV generates a request sequence against a deterministic Steiner tree algorithm ALG on G. In the initial phase, called the 0th phase, ADV defines $Z_0 := G_0$ and requests vertices of Z_0. Let T_0 be the Steiner tree computed by ALG for these requests, and P_0 be the path in T_0 connecting the requests. For the ith phase with $i \geq 1$, ADV defines the path Z_i consisting of children of edges of P_{i-1}, and requests vertices of Z_i that have not been requested. Let T_i be the Steiner tree computed by ALG for all the requested vertices thus far. For an edge e in P_{i-1}, vertices incident to a child of e must be contained in the subgraph S of T_i induced by the descendants of e. If S is connected, then there is a path Q_e in S connecting the end-vertices of e. Otherwise, since T_i is connected, there is a unique child m_e such that $S \cup m_e$ has a path Q_e connecting the end-vertices of e. Let P_i be the path obtained by concatenating Q_e for all edges e in P_{i-1}. We can inductively observe that P_i and Z_i are Steiner trees for the requests up to the ith phase. If $w(P_i) > \gamma_i w(P_{i-1})$, then ADV quits generating requests, where $\gamma_i := K_i/K_{i-1} \geq 1$. Otherwise, ALG performs the next phase.

Analysis. The following lemma is used to guarantee that ADV quits in finite phases.

Lemma 13. *There exists $\ell \geq 1$ such that $K_{\ell+1} = K_\ell$.*

Proof Sketch. Observing that a sequence $(a_i)_{i\geq 0}$ with the recurrence $a_{i+1} = b(a_i - a_{i-1})$ oscillates for $0 < b < 4$, we have the lemma. □

Lemma 13 implies $\gamma_{\ell+1} = K_{\ell+1}/K_\ell = 1$. On the other hand,

$$w(P_i) \geq w(Z_i) = (1+\epsilon)w(P_{i-1}) \qquad (1)$$

by the definitions of P_i and Z_i. Therefore, ADV performs at most $\ell + 1$ phases.

The following lemma is used to estimate the ratio of the cost of ALG to the cost of ADV.

Lemma 14. $\sum_{i=0}^{j} K_i/K_{j-1} \geq K_0 + K_1$ *for any $j \geq 1$.*

Proof Sketch. Induction on j. □

Lemma 15. *If ADV quits at the qth phase, then $w(T_q)/w(Z_q)$ tends to 4 as $m \to \infty$, $\epsilon \to 0$, and $K_1 \to 3$.*

Proof. By definition, P_i consists of descendants of edges in P_{i-1}. This means that P_i and P_{i-1} are edge-disjoint. Therefore, it follows that $w(T_j) \geq \sum_{i=0}^{q} w(P_i) - \delta$, where δ is the sum of $w(m_e)$ overall edges e in $P_0, \ldots, P_{q-1}$ having m_e. We can upper bound δ by summing weight of one child of all edges; therefore,

$$\delta \leq \sum_{i\geq 1} \left(\prod_{j=1}^{i-1} m^j\right) \frac{(1+\epsilon)^i}{\prod_{j=1}^{i} m^j} = \sum_{i\geq 1} \left(\frac{1+\epsilon}{m}\right)^i < \frac{\frac{1+\epsilon}{m}}{1 - \frac{1+\epsilon}{m}} \to 0 \quad [m \to \infty].$$

Since ADV quits at the qth phase, it follows that $w(P_i) \leq \gamma_i w(P_{i-1})$ for $1 \leq i < q$ and $w(P_q) > \gamma_q w(P_{q-1})$. Therefore, it follows from Lemma 14 that

$$\begin{aligned}
\lim_{m\to\infty} \frac{w(T_q)}{w(Z_q)} &= \frac{\sum_{i=0}^{q} w(P_i)}{w(Z_q)} \geq \frac{\sum_{i=0}^{q-1} w(P_i) + w(P_q)}{(1+\epsilon)w(P_{q-1})} \qquad \text{[by (1)]} \\
&> \frac{\sum_{i=0}^{q-1} \prod_{j=i}^{q-2} \gamma_{j+1}^{-1} w(P_{q-1})}{(1+\epsilon)w(P_{q-1})} + \frac{\gamma_{q-1}}{1+\epsilon} = \frac{1}{1+\epsilon}\left(\frac{\sum_{i=0}^{q-1} K_i}{K_{q-1}} + \frac{K_q}{K_{q-1}}\right) \\
&\geq \frac{K_0 + K_1}{1+\epsilon} \to 4. \qquad [\epsilon \to 0, K_1 \to 3, K_0 = 1]
\end{aligned}$$

□

Thus, we have the following theorem.

Theorem 4. *If a deterministic online Steiner tree algorithm is ρ-competitive on outerplanar graphs, then $\rho \geq 4$.*

References

1. Alon, N., Azar, Y.: On-line steiner trees in the Euclidean plane. Discret. Comput. Geom. **10**, 113–121 (1993)
2. Angelopoulos, S.: Online priority steiner tree problems. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tóth, C.D. (eds.) WADS 2009. LNCS, vol. 5664, pp. 37–48. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03367-4_4
3. Angelopoulos, S.: Parameterized analysis of online steiner tree problems. In: Adaptive, Output Sensitive, Online and Parameterized Algorithms. Dagstuhl Seminar Proceedings (2009). http://drops.dagstuhl.de/opus/volltexte/2009/2121
4. Angelopoulos, S.: On the competitiveness of the online asymmetric and Euclidean steiner tree problems. In: Bampis, E., Jansen, K. (eds.) WAOA 2009. LNCS, vol. 5893, pp. 1–12. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12450-1_1
5. Averbuch, B., Azar, Y., Bartal, Y.: On-line generalized steiner problem. Theor. Comput. Sci. **324**, 313–324 (2004)
6. Awerbuch, B., Bartal, Y., Fiat, A.: Competitive distributed file allocation. Inf. Comput. **185**(1), 1–40 (2003)
7. Bartal, Y., Fiat, A., Rabani, Y.: Competitive algorithms for distributed data management. J. Comput. Syst. Sci. **51**(3), 341–358 (1995)
8. Berman, P., Coulston, C.: On-line algorithms for steiner tree problems. In: Proceedings of the 29th ACM Symposium on Theory of Computing, pp. 344–353 (1997)
9. Chekuri, C., Gupta, A., Newman, I., Rabinovich, Y., Sinclair, A.: Embedding k-outerplanar graphs into ℓ_1. SIAM J. Discret. Math. **20**(1), 119–136 (2006)
10. Fleischner, H.J., Geller, D.P., Harary, F.: Outerplanar graphs and weak duals. J. Indian Math. Soc. **38**, 215–219 (1974)
11. Gupta, A., Newman, I., Rabinovich, Y., Sinclair, A.: Cuts, trees, and ℓ_1-embedding of graphs. Combinatorica **24**(2), 233–269 (2004)
12. Imase, M., Waxman, B.M.: Dynamic steiner tree problem. SIAM J. Discret. Math. **4**(3), 369–384 (1991)
13. Lund, C., Reingold, N., Westbrook, J., Yan, D.: Competitive on-line algorithms for distributed data management. SIAM J. Comput. **28**(3), 1086–1111 (1999)
14. Naor, J.S., Panigrahi, D., Singh, M.: Online node-weighted steiner tree and related problems. In: Proceedings 52nd Annual IEEE Symposium on Foundations of Computer Science, pp. 210–219 (2011)
15. Westbrook, J., Yan, D.C.K.: The performance of greedy algorithms for the on-line steiner tree and related problems. Math. Syst. Theory **28**, 451–468 (1995)