Yale University

Department of Computer Science

The Performance of Greedy Algorithms for the On-Line Steiner Tree and Related Problems

Jeffery Westbrook, [1] Dicky C. K. Yan, [2]

YALEU/DCS/TR-911

November, 1992

[1]Department of Computer Science, Yale University, New Haven, CT 06520-2158. Research partially supported by NSF Grant CCR-9009753.

[2]Department of Operations Research, Yale University, New Haven, CT 06520-0162. Research partially supported by NSF Grant DDM-8909660 and a University Fellowship from the Graduate School, Yale University.

The Performance of Greedy Algorithms for the On-Line Steiner Tree and Related Problems

Jeffery Westbrook, [*] Dicky C. K. Yan, [†]

Abstract

The on-line Steiner Tree problem on a general metric space is studied. It is shown that the greedy on-line algorithm is $O(\log(\frac{d}{z}s))$-competitive, where s is the number of regular nodes, d the maximum metric distance between any two revealed nodes and z the optimal off-line cost. Our results tighten the previous known bound [8] and show that Algorithm **SB** of Bartal et. al. [6] for the on-line File Allocation problem is $O(\log\log N)$-competitive on an N-node hypercube or butterfly network. A lower bound of $\Omega(\log(\frac{d}{z}s))$ is shown to hold.

We further consider the on-line generalized Steiner problem on a general metric space. We show that a class of lazy and greedy deterministic on-line algorithms are $O(\sqrt{k}\cdot\log k)$-competitive and no on-line algorithms is better than $\Omega(\log k)$-competitive, where k is the number of distinct nodes that appear in the request sequence.

For the on-line Steiner problem on a directed graph, it is shown that no deterministic on-line algorithm is better than s-competitive and the greedy on-line algorithm is s-competitive.

1 Introduction

A well-known optimization problem is to find a Steiner tree of a given set of points. One is given a metric space $\mathcal{M}(M,\rho)$ with point set M and metric ρ, and a subset of points, $\sigma \subseteq M$, called the regular points (or terminals). It is required to find a tree of minimum

[*]Department of Computer Science, Yale University, New Haven, CT 06520-2158. Research partially supported by NSF Grant CCR-9009753.

[†]Department of Operations Research, Yale University, New Haven, CT 06520-0162. Research partially supported by NSF Grant DDM-8909660 and a University Fellowship from the Graduate School, Yale University.

weight that contains all the regular points and any number of other points, where the weight of an edge (x, y) is $\rho(x, y)$ and the weight of a tree is the sum of the weights of the edges in the tree. The above described problem, the Steiner Tree (ST) problem, is NP-hard. Winter [14] and Maculan [11] gave surveys of research results on the ST problem on different metric spaces, including the Euclidean space, the rectilinear metric space and the case when $\mathcal{M}$ is a weighted undirected graph.

1.1 The On-line Steiner Tree Problem

In [8] Imase and Waxman studied the on-line Steiner tree (ST) problem on graphs. Let $G(V, E)$ be a connected graph. An adversary reveals on-line a sequence σ of s distinct nodes to a server. The server must maintain a connected Steiner subgraph SS of G. Each time a node v is revealed, the server is required to expand SS, if necessary, so that it remains connected and contains v. It may do this by adding any nodes and edges to SS. Once an edge or node is added to SS, the server cannot subsequently remove it. The cost of a Steiner subgraph is the sum of the weights of the edges in the subgraph. The goal of the on-line server is to minimize the cost of the final SS obtained, after all nodes in σ are revealed.

Given G and a request sequence σ, let $OPT(G, \sigma)$ be the cost of the optimal Steiner tree on the set of nodes in σ. In general the on-line server cannot achieve $OPT(G, \sigma)$. Consider the simple example of a triangle abc with $ac = 5$, $ab = 3$, and $bc = 3$ and the sequences $\sigma_1 = (a, c)$ and $\sigma_2 = (a, c, b)$. After seeing only the common prefix (a, c) no on-line algorithm can determine whether it should connect a to b with the optimal solution for σ_1 or σ_2. Let $A(G, \sigma)$ be the cost incurred by the on-line server using (deterministic) on-line algorithm A for deciding how to include a newly revealed node in the current Steiner tree, and let $R_A(G, \sigma) = A(G, \sigma)/OPT(G, \sigma)$. Then A is said to be α-competitive if, for any given instance G and request sequence σ, $R_A(G, \sigma) \leq \alpha$ holds. Notice that α may be dependent on G and σ.

(Notation: Throughout this paper, σ will be used to represent both the requests for the off-line problem and a request sequence for the on-line problem under consideration and it has length s. To simplify notation, $OPT(G, \sigma)$ and $A(G, \sigma)$ will be written as $OPT(\sigma)$ and $A(\sigma)$ respectively. Given ρ, a metric, we shall use $\rho(H)$ to represent $\sum \rho(a, b)$ where H is a graph, a tree or a path and the summation is taken over all the edges (a, b) in H.)

In [8], the motivation for studying the on-line ST problem is for applications in the routing of multipoint connection in communication networks. In [6], Bartal et. al. studied the on-line problem of File Allocation on a network of processors, where file copies can be replicated and discarded in response to read and write requests at the processors. They devised a randomized on-line algorithm that is $(2+\sqrt{3})\cdot c$-competitive against an adaptive on-line adversary, where c is the competitive ratio of any existing on-line algorithm for on-line Steiner tree, against the same adversary. So good algorithms for the on-line ST problem lead immediately to good algorithms for the on-line File Allocation problem.

The greedy algorithm, GREEDY, connects a newly revealed node by the least weight path to a node in the current SS. Imase and Waxman [8] have shown that:

- There exists graphs and request sequences on which no deterministic on-line algorithm can be better than $(1+\frac{1}{2}\lfloor\log_2(s-1)\rfloor)$-competitive.
- GREEDY is $\lceil\log_2 s\rceil$-competitive.

In this paper we tighten the analysis of GREEDY and extend the problem to that on a general metric space. Let $OPT(\sigma)$ be the cost of the off-line optimal Steiner tree for a given sequence of vertices σ. Let $d(\sigma)$ be the diameter of the vertex set σ, *i.e.*, the maximum over all pairs of vertices $\{u,v\}\in\sigma\times\sigma$ of the metric distance $\rho(u,v)$. Note that $d(\sigma)\leq OPT(\sigma)$ and when $\mathcal{M}$ is an undirected graph G with the shortest path distance over G as the metric, $d(\sigma)$ is less than or equal to the diameter of G. We shall show that GREEDY is $O(\log(\frac{d(\sigma)}{OPT(\sigma)}s))$-competitive on any metric space.

In addition, we show that for any value of the ratio $d(\sigma)/OPT(\sigma)$, there exists a graph on which no on-line algorithm can be better than $\Omega(\log(\frac{d(\sigma)}{OPT(\sigma)}s))$-competitive.

Alon and Azar [1] studied the on-line ST problem on the Euclidean space and showed that no on-line algorithm can be better than $\Omega(\log s/\log\log s)$-competitive. The study of on-line ST in the Euclidean space is important, for example, for facilities planning (e.g. building of roads) on plane surfaces.

1.2 The Generalized Steiner Problem

The generalized Steiner (GS) problem was formulated by Krarup (see [14]). One is given an undirected graph $G(V,E)$, with nodes which represent possible communication sites, and m edges representing links that can be made between the sites. Each edge has a positive weight. A list σ of node pairs is given. They represent pairs of sites for which a communication link is required. For each such pair of nodes $\{a, b\} \in V \times V$, a positive integer r_{ab} is given. The problem is to find a minimum cost subgraph SS, so that for each $\{a, b\}$ pair there exists at least r_{ab} edge-disjoint paths in SS between nodes a and b. The subgraph SS need not be connected, as long as each revealed node pair lies in some common component. In general, the optimal solution consists of a forest of Steiner trees. If $r_{ab} = 1$ for all the node pairs $\{a, b\} \in \sigma$ and all the node pairs contain a particular node, the problem is equivalent to the Steiner Tree problem on a graph; so the off-line GS problem is NP-complete. Agrawal et. al. [2] found an approximation algorithm for the cases with general r_{ab}'s. Their algorithm finds, in $O(m \log m \log r_{max})$ time, a network of cost at most $(2 - 2/k) \lceil \log_2(r_{max} + 1) \rceil$ times that of the optimal network, where $k \geq 2$ is the total number of distinct nodes contained in σ, and r_{max} is the maximum of the r_{ab}'s.

We consider the on-line version of the above problem, with $r_{ab} = 1$, $\forall \{a, b\} \in \sigma$ on a metric space $\mathcal{M}(M, \rho)$, where point pairs arrive on-line. The on-line algorithm must connect each pair as it arrives by adding edges as necessary to SS. The pair can be connected directly or it can be connected via any number of intermediate connected components that have been previously constructed. Once a link is made, however, it cannot be changed. Again, the goal is to minimize the final cost to the server. As in on-line ST, the on-line server has no prior knowledge of σ. Since the on-line ST problem is a special case of the on-line GS problem, the lower bound of $\Omega(\log k)$ on competitiveness holds for on-line GS on any undirected graph. We shall show that a class of greedy algorithms are $O(\sqrt{k} \log k)$-competitive for the on-line GS problem.

1.3 The Steiner Tree Problem on a Directed Graph

In the directed Steiner tree problem on a directed graph (SPDG), a weighted directed graph $G(V, A)$ is presented with a set of regular nodes $\sigma \subseteq V$ and a particular node called the root node, r. It is required to find a directed ST of minimum weight such that there exists a path from r to each of regular nodes on the tree. The ST problem is a special case of the SPDG, so SPDG is NP-hard. Reference [11] contains a review of different methods and formulations for solving the SPDG. This problem was first introduced by Nastansky et. al. [12] for the case when G is acyclic. This occurs in practical applications. For example, r may represent a sewer that collects waste from the drainage system. Finding an optimal directed Steiner tree corresponds to designing an optimal drainage system, where the presence of gravity implies acyclicity in G. See [12] for a discussion of other applications which include snow removal in a city and the design of information flow in a hierarchial structure of a company. We look at the on-line version of SPDG, which is similar to on-line ST, with the added requirement that at any time, SS must contain a directed path from r to each of the revealed regular nodes. We shall show that no deterministic on-line algorithm is better than s-competitive, and the greedy on-line algorithm is s-competitive.

Uncapacitated Facility Location Problem

The uncapacitated facility (or plant) location problem (UFLP) is a much studied and important problem in operations research. It concerns the location of facilities at p possible sites $i_1, \cdots, i_p$ to serve q clients $j_1, \cdots, j_q$. Having a facility at location i will incur a fixed cost of f_i while using facility i to serve client j will incur a cost of c_{ij}. The problem is to decide how to choose the sites to place the plants and to assign the clients to the plants to minimize the overall costs. Each plant can serve as many clients as it is needed. This optimization problem is NP-hard.

Facility location problems have received much attention because of its wide applications in areas such as financial planning, network planning, and machine schedule etc. (See Krarup and Pruzan [9] for a detailed survey and Wong [16] for an annotated bibliography.)

In [15], Wong showed how to formulate UFLP as an SPDG. Consider the on-line version of UFLP where $\mathcal{J} = \{j_1, \cdots, j_q\}$ are potential clients and a sequence σ of s clients in $\mathcal{J}$ appear. Each time a client $v \in \mathcal{J}$ appears, it needs to be served by a facility. The on-line server has to decide which facility to assign to v, without any knowledge of the unrevealed part of σ. The building of plants and their assignment to clients are irreversible. Using Wong's formulation, we see that on-line UFLP is equivalent to on-line SPDG on a special type of directed graphs and it will be shown that no deterministic on-line algorithm is better than s-competitive for on-line UFLP.

This paper is organized as follows: in section 2, we shall restate the on-line ST problem and give our proof for the new results; in section 3, we shall discuss the on-line minimum spanning tree (MST) problem; section 4 gives the lower bound on the competitiveness of on-line algorithms for the on-line ST and MST problems; section 5 discusses the implication of our results on the on-line File Allocation problem; in section 6 we shall show that a class of greedy algorithms are $O(\sqrt{k}\log k)$-competitive for on-line GS; in section 7 we shall show that no algorithm is better than s-competitive for the on-line SPDG and UFLP problems.

2 On-Line ST Problem on a General Metric Space

Given a metric space $\mathcal{M}(M, \rho)$ with point set M, metric ρ and an initial point $v_1 \in M$, the adversary is going to reveal a sequence, $\sigma = (v_1, \cdots, v_s)$, called the regular points, in M. Each time a point v_i $(i \geq 2)$ is revealed, the on-line server is required to extend the on-line Steiner subgraph (SS), if necessary, so that that the new SS will contain v_i. The server is allowed to include Steiner points, points that are not in σ, in SS and will be charged a cost of $\rho(SS)$ after σ has been revealed.

We consider the performance of the following class, $\mathcal{C}$, of on-line algorithms to which GREEDY belongs:

> When node $2 \leq i \leq s$ is revealed, any on-line algorithm from $\mathcal{C}$ will not incur a cost greater than $\Delta_i = \min\{\rho(v_i, v_j) | 1 \leq j < i\}$.

Another on-line algorithm that belongs to $\mathcal{C}$ is one that connects v_i to the nearest previously revealed node. Let $C(\sigma) = \sum_{i=2}^{s} \Delta_i$; any algorithm from $\mathcal{C}$ will not incur a cost greater than $C(\sigma)$. We shall use T to denote the optimal Steiner tree and $z = \rho(T) = OPT(\sigma)$ the optimal cost. It follows from Theorem 2 that any algorithm in $\mathcal{C}$ is $O(\log(\frac{d(\sigma)}{z} s))$-competitive.

We restate the proof of Imase and Waxman for the convenience in proving our other results.

Theorem 1 *For all instances $\mathcal{M}(M, \rho)$ and σ, $C(\sigma) \leq z \cdot \log_2 s$, $\forall\ s \geq 2$.*

Proof: We define trees T_i $(i = 1, \cdots, s)$, as follows:

- Initially, T_1 is defined to be T, with root v_1, and other trees are not defined.
- When node v_i $(i \geq 2)$ is revealed, it is contained in exactly one of $T_1, \cdots, T_{i-1}$. Suppose it is contained in T_j $(1 \leq j < i)$. We find the edge e that contains the mid-point of the unique path $\mathcal{P}$ on tree T_j that runs from its root v_j to v_i. If the mid-point happens to fall on a node, we choose e to be one of the two edges incident to the node arbitrarily. We remove e from T_j and two trees are formed. The one containing v_i (v_j) will be called T_i (T_j) with v_i (v_j) as its root.

At any time, after i nodes have been revealed, we have i trees $T_1, \cdots, T_i$, with roots $v_1, \cdots, v_i$, respectively. The root is the only revealed node in a tree.

Given any tree T, with s regular nodes, $\rho(T) = z$ and only its root revealed, let $J(s,z)$ be the maximum possible value of $C(\sigma)$ over all sequences, σ, of revealing the nodes. Define $J(1,z) = 0$ for all z. We shall show by induction on s that $J(s,z) \leq z \cdot \log_2 s$. $J(2,z) = z$ for any σ. It can be easily shown that $J(3,z) \leq 3 \cdot z/2 < z \cdot \log_2 3$. Assume the claim holds for $s = 2, \cdots, (k-1)$, where $(k-1) \geq 3$. We show that it holds for $s = k$.

When the next node is revealed, T is split into two trees, say T_x and T_y with weights $\rho(T_x) = x \leq y = \rho(T_y)$, and t and $(k-t)$ regular nodes respectively. Let w be the weight of the edge (e) removed. So $z = w + x + y$ and $x \leq (z-w)/2$.

When node v_i $(i \geq 2)$ is revealed, we have $\Delta_i \leq \rho(v_i, v_j) \leq \rho(\mathcal{P})$. The last inequality follows from the triangular inequality. It can be shown that $\rho(\mathcal{P}) \leq 2 \cdot (w + x)$ and hence,

$$J(k,z) \leq J(t,x) + J(k-t,y) + 2 \cdot (w+x)$$

Using the induction hypothesis, it can be shown that $J(k,z) \leq z \cdot \log_2 k$ holds and hence the theorem. □

We now use the above analysis to prove our main theorem. We abbreviate

$$d(\sigma) = \max\{\rho(v_i, v_j) | v_i, v_j \text{ are distinct nodes in } \sigma\}$$

by d.

Given T and σ, the sequence of nodes to be revealed, we can construct the binary tree $\mathcal{T}(T,\sigma) = \mathcal{T}$ to represent the recursive decomposition of T into $\{T_1, \cdots, T_s\}$. The recursion tree $\mathcal{T}$ has the following characteristics:

1. It is a full binary tree where each node has either 2 children or none.

2. Each node in $\mathcal{T}$ corresponds to a subtree T_i, with root v_i.

A node i in $\mathcal{T}$ is called *heavy* if the tree it represents, T_i, has $\rho(T_i) \geq d$, otherwise it is called *light*. Let $\mathcal{T}_h$ be the subtree of $\mathcal{T}$ consisting only of heavy nodes. A leaf of $\mathcal{T}_h$ is a heavy node of $\mathcal{T}$ with two light children or no children. An internal node of $\mathcal{T}_h$ may have one heavy child and one light child.

Lemma 1 *There are no more than $(\lfloor z/d \rfloor - 1)$ heavy nodes with two heavy children.*

Proof: Let ℓ be the number of heavy leaves and let p be the depth of $\mathcal{T}_h$. Each of the trees represented by the heavy leafs in $\mathcal{T}$ are of weight at least d and they are all disjoint. Hence

$\ell \leq \lfloor z/d \rfloor$. Let n_i denote the number of heavy nodes at depth i and let l_i the number of heavy leaves at depth less than i. (The top of the tree $\mathcal{T}_h$ is at level 0 and the bottom of it is at level p). Consider the sum $m_i = n_i + l_i$. We have $m_0 = 1$, $m_d = \ell$ and $m_i \leq m_{i+1}$ for all $0 \leq i \leq p-1$. Let h_i be the number of nodes at level i with two heavy children. Then $m_i = m_{i+1} - h_i$. We have

$$\begin{aligned}\sum_{i=0}^{p-1} h_i &= \sum_{i=0}^{p-1} m_{i+1} - m_i \\ &= m_d - m_0\end{aligned}$$

Thus the number of heavy nodes with two children is equal to $\ell - 1$. $\square$

Theorem 2 $C(\sigma) \leq z \cdot (3 + \max\{\log_2 e, \log_2 \frac{s}{\lfloor z/d \rfloor}\})$.

Proof: The weight of the on-line tree is given by summing up the costs associated with each node in $\mathcal{T}$. By Lemma 1 there are not more than $[(z/d) - 1]$ nodes with two heavy children. Each of them has an associated cost at most d since in the tree represented by a heavy node, no two nodes are more than d apart. Hence the total cost from these nodes is not more than z.

Consider the set $\{x_1, x_2, \ldots, x_k\}$ of light nodes with heavy parents. Each x_i represents a subtree T_i of T of weight $\rho(T_i) < d$. The subtrees are mutually disjoint. Each tree is charged at most $[2 \cdot \rho(T_i) + 2 \cdot \rho(e_i)]$ for the split that created it, where e_i is the edge associated with the splitting up of its parent tree. (It is the cost associated with the light node's heavy parent, and corresponds to the sum $2 \cdot (x + w)$ in Theorem 1.) Hence, the total cost associated with these splits are not more than $2 \cdot \sum_{i=1}^{k} [\rho(T_i) + \rho(e_i)] \leq 2 \cdot z$.

Let $z_i = \rho(T_i)$. By Theorem 1, the cost associated with revealing the s_i nodes in the trees T_i is bounded above by:

$$Z^* = \max \sum_{i=1}^{k} z_i \cdot \log_2 s_i$$

$$\begin{aligned} s.t. \quad \sum_{i=1}^{k} s_i &\leq s \\ \sum_{i=1}^{k} z_i &\leq z \\ s_i &\geq 1 \qquad 1 \leq i \leq k \\ 0 \leq z_i &\leq d \qquad 1 \leq i \leq k \end{aligned}$$

where $z > 0$, $1 \leq k \leq s$ and the z_i's and s_i's are variables.

Case 1: $k \leq \lfloor z/d \rfloor$

We have $Z^* \leq d \cdot \sum_{i=1}^{k} \log_2 s_i \leq d \cdot k \cdot \log_2(s/k)$. After differentiation, it was found that if $s/e \leq \lfloor z/d \rfloor$, the last expression is maximized at $k = s/e$, otherwise, it is maximized at $k = \lfloor z/d \rfloor$. In the first case, since $k \cdot d \leq z$, $Z^* \leq z \cdot \log_2 e$; in the latter case, $Z^* \leq z \cdot \log_2 \frac{s}{\lfloor z/d \rfloor}$.

Hence, we have, $Z^* \leq z \cdot \max\{\log_2 e, \log_2 \frac{s}{\lfloor z/d \rfloor}\}$.

Case 2: $k > \lfloor z/d \rfloor$

Let $s_1, \cdots, s_k$ be the s_i values of any optimal solution, indexed such that $s_i \geq s_{i+1}$. Since $\log_2 s_i \geq \log_2 s_{i+1}$, a solution with

$$d_i = \begin{cases} d & 1 \leq i \leq \lfloor z/d \rfloor \\ \delta & i = \lfloor z/d \rfloor + 1 \\ 0 & \text{otherwise} \end{cases}$$

where $\delta = z - d \cdot \lfloor z/d \rfloor$, is optimal. Hence,

$$\begin{aligned} Z^* &\leq d \cdot \sum_{i=1}^{\lfloor z/d \rfloor} \log_2 s_i + \delta \cdot \log_2 s_{\lfloor z/d \rfloor + 1} \\ &\leq d \cdot \lfloor z/d \rfloor \log_2 \frac{x}{\lfloor z/d \rfloor} + \delta \cdot \log_2(s - x) \end{aligned}$$

where $x = \sum_{i=1}^{\lfloor z/w \rfloor} s_i$. The last expression is maximized with $x = \frac{\lfloor z/d \rfloor}{\lfloor z/d \rfloor + \delta/d} \cdot s = \frac{\lfloor z/d \rfloor}{z/d} \cdot s$, giving

$$\begin{aligned} Z^* &\leq d \cdot \lfloor z/d \rfloor \log_2 \frac{s}{z/d} + \delta \cdot \log_2(\frac{\delta/d}{z/d} s) \\ &\leq z \cdot \log_2 \frac{s}{z/d} + \delta \cdot \log_2 \frac{\delta}{d} \\ &\leq z \cdot \log_2 \frac{s}{z/d} \end{aligned}$$

Hence, $Z^* \leq z \cdot \log_2 \frac{s}{z/d}$.

Combining the two cases, the theorem follows. □

3 On-Line Minimum Spanning Tree (MST) Problem

The MST problem is a special case of the Steiner tree problem in which Steiner points are not allowed. We consider the on-line version of the MST problem, which can be described in two ways:

- The adversary reveals a sequence of s points in a known metric space $\mathcal{M}(M, \rho)$, with the first point revealed in advance. Each time a point is revealed, the on-line server has to connect the point to one of the previously revealed points and he is charged the metric distance between the two points. One example of $\mathcal{M}$ is the Euclidean space.
- The adversary reveals a complete graph with s nodes, with the first node revealed in advance. Each node v is revealed in turn with the edges connecting it and the previously revealed nodes; the edge weights follow the triangular inequality. The server has to connect v to the current on-line spanning subgraph and he is charged the total weight of the edges used.

Notice that the problem is trivial in a space where the distance measure does not follow the triangular inequality, for the adversary can reveal a last node at zero distance from all the other nodes, thus forcing an infinite competitive ratio for any on-line algorithm.

Algorithm GREEDY(MST):"When a new node is revealed, connect it to the nearest previously revealed node".

Theorem 3
(1) GREEDY(MST) is an optimal on-line algorithm.
(2) GREEDY(MST) is $(3 + \max\{\log_2 e, \log_2 \frac{s}{\lceil z/d \rceil}\})$*-competitive, where z is the cost of the optimal MST. In the first version of on-line MST, d is the maximum metric distance between pairs of points in σ; d represents the maximum edge weight in the complete graph in the second version of the problem.*

Proof: Let **B** be any on-line algorithm, deterministic or randomized. By the greediness of GREEDY(MST), each time a node is revealed, GREEDY(MST) cannot incur a cost more than the (expected) cost incurred by **B**. Hence, GREEDY(MST) cannot incur a bigger total cost than **B**.

The second claim follows because the proofs we have used for Theorems 1 and 2 can be used for proving the competitiveness of GREEDY(MST) for on-line MST, by redefining the variables s and s_i's to be the number of nodes in the corresponding trees (rather than the number of regular nodes) and z to be the weight of the optimal MST. □

4 Lower Bounds on Competitiveness

In [8], Imase and Waxman construct a class of graphs and corresponding request sequences, so that any deterministic on-line algorithm will incur a cost of at least $1 + \frac{1}{2} \lfloor \log_2(s-1) \rfloor$ for the on-line ST problem. We assume the reader is familiar with their lower bound proof. These graphs, G_n's, have the following characteristics

- Each edge has weight $1/(s-1)$, where $s = 2^n + 1$, for some non-negative integer n.
- The optimal Steiner tree T consists of a chain of $(s-1)$ edges, running from a node v_0 to node v_1.
- All the nodes in T are revealed.
- $d(\sigma) = z = 1$.

Corollary 1 *For any given $z \geq d(\sigma) > 0$, no deterministic on-line algorithm can be better than $(\frac{1}{2} + \frac{1}{2} \left\lfloor \log_2 \frac{(s-1)}{\lceil z/d(\sigma) \rceil} \right\rfloor)$-competitive for the on-line ST or the on-line MST problem, where z is the weight of the corresponding optimal tree and $d(\sigma)$ is defined as before.*

Proof: We first consider the lower bound for the on-line ST problem. Given any $z \geq d > 0$, let $k = \lceil 2 \cdot z/d \rceil - 1$ and $\delta = z - k \cdot \frac{d}{2}$. Given any $s' = 2^n + 1$, for some non-negative integer n, we construct graph H_n as follows:
Construct k copies of G_n's and scale their edge weights so that the chain of $(s'-1)$ edges described above, running from v_0 to v_1, has weight $d/2$. Construct an extra copy of G_n and scale its edge weights so that the $(s'-1)$-chain from v_0 to v_1 has weight δ. Identify node v_0 of all the copies of G_n's to be one node.

For each copy of G_n in H_n, the request sequence chosen as in [8] is applied. The optimal Steiner tree consists of $(k+1)$ chains meeting at node v_0 and has weight z; we also have $d(\sigma) = d$. Our request sequence has length $s = (k+1) \cdot s' - k$.

It follows from the analysis in [8] that any on-line algorithm will incur a cost of at least $[1 + \frac{1}{2} \lfloor \log_2(s'-1) \rfloor]$ that of the optimal cost. So no on-line algorithm can be better than $[1 + \frac{1}{2} \left\lfloor \log_2 \frac{(s-1)}{\lceil 2 \cdot z/d \rceil} \right\rfloor]$-competitive and the lower bound follows.

For the on-line MST problem, we construct the distance graph of H_n and the request sequence described above. The same request sequence is used in revealing the complete graph. It can be shown similarly that the same lower bound holds. □

5 On-Line File Allocation Problem

The on-line File Allocation problem concerns the distribution of file copies in a network of processors. Requests arrive at the processors for reading or writing the files. The on-line server has to decide on-line, whether to discard a second copy of a file or to duplicate a file at a cost. In [6], Bartal et. al. described Algorithm **SB**, a $(2+\sqrt{3}) \cdot c$-competitive randomized on-line algorithm, against an on-line adaptive adversary, where c is the competitiveness of a on-line algorithm for on-line ST against the same adversary. It follows from the results of Imase and Waxman that their algorithm is $O(\log N)$-competitive on a graph with N nodes if GREEDY is used.

Corollary 2 *Algorithm* **SB**[6] *is $O(\log\log N)$-competitive when it is applied to any N-node graph with uniform edges and a diameter of $O(\log N)$, and if GREEDY is used as the on-line Steiner tree algorithm.*

Graphs that fall into this category include hypercubes, butterflies, the Beneš network, cube-connected-cycles, pyramids and shuffle-exchange graphs. (See [10, 13] for a discussion of these networks.) Notice that the above corollary applies to any on-line Steiner tree algorithm in $\mathcal{C}$.

6 The Generalized Steiner Problem

In the on-line GS problem, a metric space $\mathcal{M}(M, \rho)$ with point set M and metric ρ is given. The adversary reveals a sequence of point pairs. Underlined letters will be used to represent point pairs and we let $\sigma = (\underline{v}_1, \cdots, \underline{v}_s)$. Each time a point pair $\{a, b\}$ is revealed to the on-line server, SS has to be extended, if necessary, so that the resultant SS contains a path from a to b. As in on-line ST, the on-line server has no prior knowledge of σ, and points and edges added to SS cannot be removed. Since the on-line ST problem is a special case of the on-line GS problem, the lower bound of $(1 + \frac{1}{2}\lfloor \log_2(k-1) \rfloor)$ on competitiveness holds for on-line GS on a graph, where k is the number of distinct points that appear in σ.

Using results from the performance of greedy algorithms for the on-line ST problem, we shall show that a class of greedy algorithms are $O(\sqrt{k}\log k)$-competitive for the on-line GS problem.

Definition: An on-line algorithm A is called *lazy* if it satisfies the following property: if when the request point pair $\{x, y\}$ arrives, points x and y are already connected by some path in SS, then A will not change SS.

It is natural to select on-line algorithms that are lazy. The following lemma shows that being lazy cannot hurt.

Lemma 2 *Given any deterministic on-line algorithm A, there exists a deterministic on-line algorithm A' that is lazy and $A'(\sigma) \leq A(\sigma)$ for all σ.*

Proof: A' simulates A but at any time includes only those edges used by A that are necessary to keep the current Steiner sites connected. The Steiner subgraph of A' is always a subgraph of that of A and the lemma follows. □

The definition of laziness and the above lemma can be extended similarly to the on-line ST and on-line SPDG.

In general, the number of request point pairs is $s = O(k^2)$ and $s \geq \lceil k/2 \rceil$. However, for lazy on-line algorithms, some of the requests may be redundant. Consider the case when $s = k = 3$ and points a, b and c appear in σ. Let $\underline{v}_1 = \{a, b\}$, $\underline{v}_2 = \{b, c\}$, and $\underline{v}_3 = \{a, c\}$. When $\underline{v}_3$ is revealed, points a and c are already connected by some path in SS and the presence of the request does not make any difference to any lazy on-line algorithm. Request pairs can be classed as redundant using the following algorithm.

```
(1) Let $U = \{u_1, \cdots, u_k\}$ be the set of $k$ points that appear in $\sigma$ and let graph
    $G = (U, \phi)$.
(2) for $i = 1, \cdots, s$ do

    (2.1) Reveal the point pair $\underline{v}_i = \{u_j, u_l\}$ and let edge $e = (u_j, u_l)$.
    (2.2) if adding edge $e$ to $G$ creates a cycle then
                  $\underline{v}_i$ is a redundant request
              else $\underline{v}_i$ is non-redundant and let $G = G + \{e\}$.
```

It can be easily verified by induction that at any time, points that belong to the same component in G are connected by some path in SS, independent of the on-line algorithm being used.

We observe that the number of non-redundant requests is not more than $(k-1)$, since each time we add in an edge, which corresponds to a non-redundant request, the number of components in G decreases by one.

Lemma 3 *Given any request sequence σ, let s be its length and k the number of distinct points that appear in σ. Let $\alpha : \aleph \times \aleph \longrightarrow \mathcal{R}^+$ be a non-decreasing function*[1]*. If a lazy on-line algorithm, A, is $\alpha(s,k)$-competitive for any request sequence that consists of only non-redundant requests, then A is $\alpha(k-1,k)$-competitive for any general request sequence.*

Proof: Let σ be a request sequence that may contain redundant requests. We use the algorithm above to obtain a subsequence of σ, called σ', that consists of all the non-redundant requests in σ. Let s' be its length and k' the number of distinct points in σ'. Hence, $k' = k$ and $s' \leq k' - 1$. We have,

$$\begin{aligned} A(\sigma) &= A(\sigma') \leq \alpha(s',k') \cdot OPT(\sigma') \leq \alpha(s',k') \cdot OPT(\sigma) \\ &\leq \alpha(k'-1,k') \cdot OPT(\sigma) = \alpha(k-1,k) \cdot OPT(\sigma) \end{aligned}$$

where the last inequality follows from the assumption that $\alpha(\cdot,\cdot)$ is a non-decreasing function. □

The above lemma allows us, in considering the competitiveness of any lazy on-line algorithm, assume that σ consists of non-redundant requests and has length $s \leq k-1$. We shall show that a class of lazy on-line algorithms are $O(\sqrt{s}\log s)$-competitive for any request sequence and it follows from the above lemma that they are $O(\sqrt{k}\log k)$-competitive.

We define the metric $\beta : \{V \times V\} \times \{V \times V\} \longrightarrow \mathcal{R}^+$ as follows:
Given point pairs $\underline{v} = \{a,b\}$ and $\underline{u} = \{x,y\}$,

$$\beta(\underline{u},\underline{v}) = \min\{[\rho(a,x)+\rho(b,y)],[\rho(a,y)+\rho(b,x)]\}$$

One can easily verify that β satisfies the triangular inequality and it forms a metric.

Let C_{GS} be the class of deterministic on-line algorithms that satisfy the following properties:

1. All algorithms in C_{GS} are lazy.

[1]$\mathcal{R}^+$ and $\aleph$ are the sets of non-negative real numbers and positive integers respectively.

2. Let $\underline{v}_i = \{x, y\}$,

$$\Delta_i = \begin{cases} \rho(x,y) & i = 1 \\ \min\{\rho(x,y), \min_{1 \le j < i} \beta(\underline{v}_j, \underline{v}_i)\} & 1 < i \le s \end{cases}$$

Algorithms in $\mathcal{C}_{GS}$ incur a cost of not more than Δ_i when point pair $\underline{v}_i$ is revealed $(1 \le i \le s)$.

The metric $\beta(\{x,y\},\{a,b\})$ measures the shortest distance in connecting point pair $\{x,y\}$ to point pair $\{a,b\}$ in the original metric ρ, with x and y connected to different points. Each time a request $\{x,y\}$ is revealed, an algorithm in $\mathcal{C}_{GS}$ will not incur a cost more than the smaller of the cost of directly connecting x and y and that of connecting $\{x,y\}$ to the nearest previously revealed point pair, where nearness between point pairs is as measured by β. The class $\mathcal{C}_{GS}$ includes the greedy algorithm that connects a point pair with the smallest possible additional cost. It also includes the point pair greedy algorithm that connects a revealed point pair either directly or to the nearest previously revealed pair, whichever is less expensive.

Let $C(\sigma) = \sum_{i=1}^{s} \Delta_i$; the total cost incurred by any algorithm in $\mathcal{C}_{GS}$ is not more than $C(\sigma)$, for any σ.

Lemma 4 *Suppose $C(\sigma) \le f(k) \cdot OPT(\sigma)$ for all σ such that the optimal off-line solution is a single Steiner tree, where $f(k)$ is some non-decreasing function of k, then $C(\sigma) \le f(k) \cdot OPT(\sigma)$ for all general request sequences.*

Proof: Suppose the optimal off-line solution for σ consists of a forest of p Steiner trees, $T_1, \cdots, T_p$. Let $\sigma_1, \cdots, \sigma_p$ be a partition of σ into subsequences such that σ_i contains all the requested points in T_i. Then we have: $C(\sigma) \le \sum_{q=1}^{p} C(\sigma_q)$, since, for each point pair revealed as part of request subsequence σ_q, Δ_i will be greater than if the point pair is revealed as part of σ. Let k_q be the number of points in T_q that are contained in σ, then

$$C(\sigma) \le \sum_{q=1}^{p} f(k_q) \cdot OPT(\sigma_q) \le f(k) \cdot \sum_{q=1}^{p} OPT(\sigma_q) = f(k) \cdot OPT(\sigma)$$

□

6.1 Single Steiner Tree Optimal Off-line Solution

We first look at the case when the optimal off-line solution consists of a single Steiner tree T, with weight $\rho(T)$. Furthermore, we assume that it is a single chain of $(k-1)$ edges; the end points of the edges are the k revealed points in σ. We define

- $\rho_T(x,y)$= the distance between nodes x and y on T,
- $\beta_T(\{x,y\},\{a,b\}) = \min\{[\rho_T(a,x)+\rho_T(b,y)],[\rho_T(a,y)+\rho_T(b,x)]\}$
- For request i,

$$\Delta_{Ti} = \begin{cases} \rho_T(x,y) & i=1 \\ \min\{\rho_T(x,y), \min_{1\leq j<i}\beta_T(\underline{v}_j,\underline{v}_i)\} & i>1 \end{cases}$$

- $C_T(\sigma) = \sum_{i=1}^{s}\Delta_{Ti}$

Thus we have redefined ρ and β in terms of distances on T. Clearly $\Delta_{Ti} \geq \Delta_i$ $(1 \leq i \leq s)$ and $C_T(\sigma) \geq C(\sigma)$ for all σ.

We construct an undirected complete graph G with $\binom{k}{2}+k$ nodes, each representing a different pair of the k nodes on T. Each point pair in σ corresponds to a single node in G. We shall use the same symbol for a node pair in T and for the corresponding single node in G. Each edge $(\underline{u},\underline{v})$ has weight $\beta_T(\underline{u},\underline{v})$.

We perform the following on-line game on G: The point pairs in σ are revealed one by one, with $\underline{v_1}$ revealed in advance. Each time a $\underline{v}_i$ $(i \geq 2)$ is revealed, we charge the on-line server for this game a cost equal to the distance between $\underline{v}_i$ and its nearest revealed neighbor in G. Let $C_{on\text{-}line}(\sigma)$ be the total on-line cost charged, and $C_{off\text{-}line}(\sigma)$ be the cost of the optimal off-line solution, which is a spanning tree.

Lemma 5 $C_T(\sigma) \leq \rho(T) + C_{off\text{-}line}(\sigma)\cdot\log_2 s$

Proof: Applying the bound in Section 2 for the on-line ST problem we have $C_{on\text{-}line}(\sigma) \leq C_{off\text{-}line}(\sigma)\log_2 s$. The lemma follows from $C_T(\sigma) \leq \rho(T) + C_{on\text{-}line}(\sigma)$, where the $\rho(T)$ term accounts for the cost for revealing node pairs $\underline{v_1}$ on T. □

Next we place an upper bound on the value of the optimal solution on G.

Lemma 6 $C_{off\text{-}line}(\sigma) \leq \rho(T)\cdot(2+\sqrt{s})$

Proof: To simplify notation, let $U = \{1,\cdots,k\}$ be the set of k points on T. Given a $(k-1)$-chain T, we form a grid graph G_T as follows:
Put T horizontally and let the k nodes along it be numbered from left to right, $1,\cdots,k$. Construct an identical copy of T, T', by turning T by ninety degrees clockwise about node 1. Draw a vertical line at each node on T and a horizontal line at each node on T'. Call

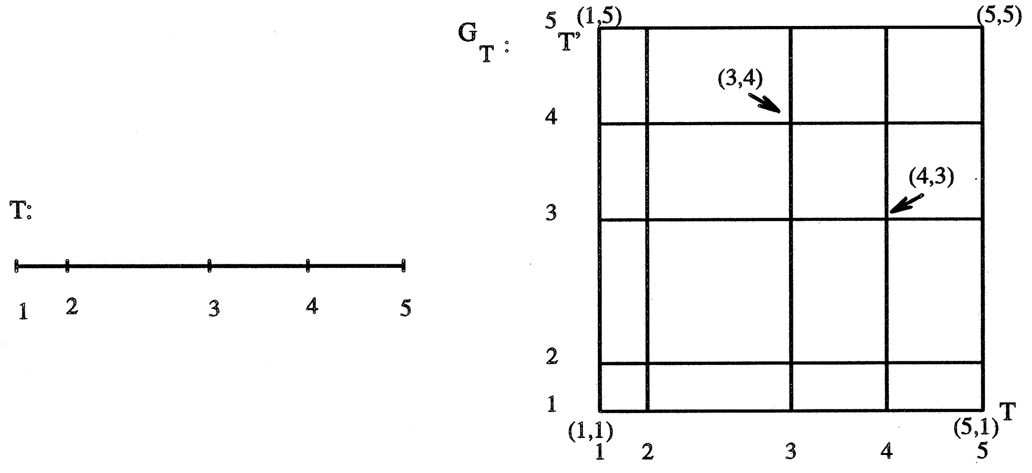

Figure 1: T and G_T

the node at the intersection of the vertical line drawn at node i on T and the horizontal line drawn at node j on T' (i, j). The grid graph G_T is formed by all the segments of the lines within the square with corners $(1,1), (1,k), (k,1)$ and (k,k) and it has vertex set $V_T = U \times U$. (See figure 1 .)

Thus each node (i, j) on G_T corresponds to a node $\{i, j\}$, called its image in G; each node $\underline{v} = \{i, j\}$ in G corresponds to two different nodes, (i, j) and (j, i), in G_T. If $i < j$, we call (i, j) and (j, i) the upper and lower images of $\underline{v}$ in G_T, since they lie above and below the 45^o degree line at $(1,1)$ respectively. It can be easily verified that $\beta_T(\underline{v}, \underline{u})$ is equal to the shortest distance on G_T, between the corresponding lower (or upper) images of $\underline{v}$ and $\underline{u}$ and it is not more than the shortest grid distance on G_T between an image of $\underline{v}$, upper or lower, and an image of $\underline{u}$, upper or lower. (In fact, the distance, over G_T, between an image of $\underline{v}$ in G_T and each of the two images of $\underline{u}$ corresponds to the two sums in the definition of β_T.) As a consequence, if we represent each node pair in σ by its lower image in G_T and then find their minimum Steiner tree, we can identify the images of the the nodes in the Steiner tree as nodes in G, and construct a corresponding Steiner tree in G, and this Steiner tree cannot have a greater cost than that of the Steiner tree in G_T. To prove Lemma 6, we only need to show the above described Steiner tree in the grid graph G_T does not have cost more than $\rho(T) \cdot (\sqrt{s} + 2)$, where $\rho(T)$ is the length of the side of the square grid.

In [7], Hannan gave an important theorem reducing the off-line minimum ST problem on the rectilinear metric space to that on a grid graph. Suppose we want to find the off-line minimum ST for a give set of regular points σ in the rectilinear metric space. We form a grid graph G_σ by drawing vertical and horizontal lines along each of the nodes in σ and let the points where these lines intersect be nodes of the graph. Hannan showed that G_σ contains a minimum ST of the original problem. Using Hannan's result, to prove Lemma 6, we need only bound the weight of the longest minimum Steiner tree in a unit square in the rectilinear metric space, where all the regular nodes lie on one side of the diagonal of the square.

In [3], Chung and Graham extended Few [5]'s results for finding the size of the largest possible minimum Steiner tree in a unit square with s regular points on the Euclidean space.

Theorem 4 (Chung and Graham[3]) *Let size(s) be the greatest length of a minimum Steiner tree for a set of s regular points contained in a unit square in the rectilinear metric space. Then* $\sqrt{s}+1 \leq size(s) \leq \sqrt{s}+1+o(1)$.

(Using the above theorem, Chung and Hwang [4] later showed that $size(s) = \sqrt{s}+1$ when $\sqrt{s}$ is an integer.) In our case, all the regular points lie below one of the diagonals. Using similar ideas as in [5], the proportionality constant for the $\sqrt{s}$ term in the upper bound above can be reduced from 1 to about 0.9. We shall leave the proof to the interested reader. Lemma 6 follows from the above theorem. □

Lemma 7 *If the optimal off-line solution is a single Steiner tree,*

$$C(\sigma) \leq 2 \cdot [1 + (2 + \sqrt{k-1}) \cdot \log_2(k-1)] \cdot OPT(\sigma)$$

Proof: If the optimal off-line solution is a single chain T, then Lemmas 3, 5 and 6, and the fact that $C(\sigma) \leq C_T(\sigma)$, imply

$$C(\sigma) \leq [1 + (2 + \sqrt{k-1}) \cdot \log_2(k-1)] \cdot \rho(T).$$

Suppose the off-line solution is a single Steiner tree T_o that is not a chain. Starting from an arbitrary leaf node, we can perform a depth-first search on T_o, marking the order in which the regular nodes are encountered on the tree. Let the nodes be encountered in the order $u_1, \cdots, u_k$. Construct the chain $\{(u_1, u_2), \cdots, (u_{k-1}, u_k)\}$, where edge (u_i, u_{i+1}) has weight $\rho_{T_o}(u_i, u_{i+1})$. The chain constructed, T, will not have weight more than twice that of T_o and $\rho(T) \leq 2 \cdot OPT(\sigma)$. Using the same lemmas and $C(\sigma) \leq C_T(\sigma)$ as before, the lemma follows. □

Theorem 5 *The algorithms in C_{GS} are $2 \cdot [1 + (2 + \sqrt{k-1}) \cdot \log_2(k-1)]$-competitive for the on-line generalized Steiner problem.*

Proof: This follows from Lemmas 4 and 7. $\square$

7 The On-line Steiner Problem on a Directed Graph (SPDG) and The On-Line Uncapacitated Facility Location Problem (UFLP)

In the on-line SPDG, a directed graph is given and the on-line server has to maintain a Steiner subgraph SS so that there always exists a directed path running from the root node r to each of the revealed nodes. The aim is to minimize the total edge weights in SS. We shall give an example in the setting of UFLP that proves a linear lower bound on the competitiveness of any deterministic on-line algorithm for both the on-line SPDG and UFLP problems.

Let $\mathcal{I} = \{i_1, \cdots, i_p\}$ be the set of sites available for locating facilities and $\mathcal{J} = \{j_1, \cdots, j_q\}$ be potential customers. Let f_i be the cost of establishing a facility at $i \in \mathcal{I}$ and c_{ij} be the cost of assigning a facility at i to client $j \in \mathcal{J}$. Each assigned facility must be established and a facility can be assigned to any number of clients. Wong [15] showed that UFLP is equivalent to the SPDG problem on the directed graph G with regular node set, $\sigma = \mathcal{J}$. G has $p + q + 1$ nodes: a source node r, one node for each $i \in \mathcal{I}$ and one node for each $j \in \mathcal{J}$. For each $i \in \mathcal{I}$, there is an arc (r, i) with weight f_i and for each $j \in J$ that can be assigned to i, there is an arc (i, j) with weight c_{ij}. See figure 2(a).

Thus UFLP is a special case of SPDG with $\mathcal{J}$ as the set of regular nodes; assigning a facility $i \in \mathcal{I}$ to a node $j \in \mathcal{J}$ is the same as adding the path from r to j via i to SS.

We are interested in the on-line version of UFLP where $\mathcal{J}$ represents the set of *potential* clients and a subset of s clients in $\mathcal{J}$ arrive in a sequence σ. Each time a client arrives, it has to be assigned to a facility i, incurring a cost c_{ij}. If facility i is not already established, it has to be done at a cost of f_i. So on-line UFLP is equivalent to on-line SPDG on the type of directed graphs shown in figure 2(a), with $\sigma \subseteq \mathcal{J}$.

Theorem 6 *No deterministic on-line algorithm is better than s-competitive for on-line UFLP, where s is the number of revealed clients.*

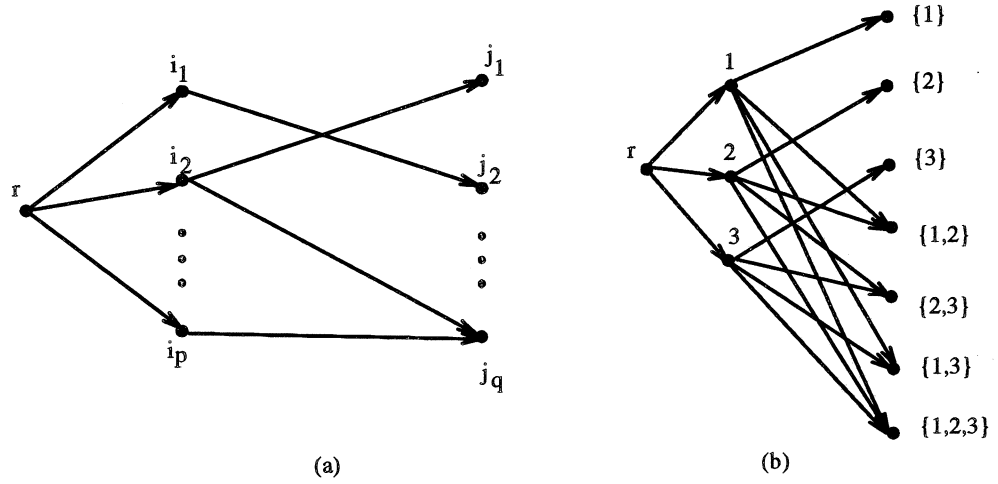

Figure 2: (a) UFLP as a SPDP, (b) Graph H for the case $s = 3$.

Proof: We shall construct a graph H with $\mathcal{I} = \{1, \cdots, s\}$ and $\mathcal{J}$ consisting of $2^s - 1$ nodes, each corresponding to a distinct non-empty subset of $\mathcal{I}$. We shall call the nodes in $\mathcal{J}$ by the subset of $\mathcal{I}$ they correspond to. There is an arc (i, j), where $i \in \mathcal{I}$ and $j \in \mathcal{J}$ if and only if $i \in j$. Arcs (r, i), $(i \in \mathcal{I})$, have unit weight and all other arcs have weight zero. An example for $s = 3$ is shown in figure 2(b).

Let A be any deterministic on-line algorithm. The adversary is going to reveal s nodes, each time forcing A to set up a different facility; A will incur a total cost of at least s. The optimal solution will only need to set up one single facility, giving $OPT(\sigma) = 1$. The adversary performs the following:

```
for $L = 1, \cdots, s$ do

    (1) Place a request at node $v \in \mathcal{J}$ that corresponds to the current set $\mathcal{I}$ in $H$.
    (2) Suppose $A$ assigns facility $i \in \mathcal{I}$ to $v$. Remove from $H$ the node $i$ and all the
        arcs incident to or from it.
    (3) Call the new graph $H$ and let $\mathcal{I} = \mathcal{I} - \{i\}$.
```

It can be shown by induction that at the beginning of each loop, a request is placed at a facility that can only be served by the $\mathcal{I}$ nodes present in H at that time. This forces A

to set up a new facility in each new loop. Also, in any loop L, all the revealed $\mathcal{J}$ nodes can be served by any one of the $\mathcal{I}$ nodes present in H at the beginning of the loop. At the beginning of loop $L = s$, only one $\mathcal{I}$ node will be left. It will be connected to all s revealed $\mathcal{J}$ nodes. The optimal off-line solution is to set up this facility and use it to serve all the s revealed customers. So $A(\sigma) \geq s \cdot OPT(\sigma)$. $\square$

Corollary 3 *No deterministic on-line algorithm is better than s-competitive for the on-line Steiner problem on directed graphs, even if the graphs are acyclic.*

It can be easily seen that the greedy algorithm that always connects the revealed node to SS via the least expensive feasible directed path is s-competitive for both problems.

8 Conclusion and Further Research

A number of interesting open problems remain.
We have shown that a class of algorithms are $O(\log(\frac{d(\sigma)}{z}s))$-competitive for the on-line Steiner tree problem and a similar bound holds for the greedy algorithm for the on-line minimum spanning tree problem. For both on-line problems it will be interesting to consider a situation in which requests can arrive in blocks of $B \in \aleph$. A lower bound on the competitiveness of $\Omega(\log(s/B))$ can be obtained for both problems. Let G be the graph used for the lower bound proof for the original problem (with $B = 1$). For each node v in G, we can duplicate $(B-1)$ copies, each connecting to v by an edge of zero length. Each time a node v is supposed to be revealed in the original request sequence for the lower bound proof, we reveal all B copies. The on-line server does not have any gain in seeing B copies of v at the same time and will incur the same cost as before. However, in the case, the adversary will have to use B times as many nodes as before and hence we replace s by s/B in the lower bound.

We have shown that a class of lazy and greedy on-line algorithms are $O(\sqrt{k} \cdot \log k)$-competitive for the on-line generalized Steiner problem. There is an obvious gap between the upper bound and the lower bound of $\Omega(\log k)$. Furthermore, we have not given a solution for arbitrary connectivity.

For all the on-line Steiner problems mentioned or studied in this paper, competitiveness is measure in terms of s, the size of the request sequence or the number of regular nodes. When the size, N, of the metric space is finite, measuring competitiveness in terms of

N shows how effective an on-line algorithm is when faced with N possible choices in the space, choices for both the server and the adversary. In terms of N, the results of Imase and Waxman imply that no deterministic on-line algorithm can be better than $\Omega(\log N)$-competitive and GREEDY is $O(\log N)$-competitive for the on-line ST problem. For the on-line Steiner problem on a directed graph, we have shown that no deterministic on-line algorithm can be better than s-competitive and the greedy algorithm is s-competitive. In terms of N, our example implies that no deterministic on-line algorithm is better than $\Omega(\log N)$-competitive. It can be shown that the greedy on-line algorithm for this problem is not better than N-competitive. So there exists a gap between the upper and lower bounds in terms of N for on-line SPDG.

Other interesting open problems include closing the gap between the lower and upper bounds in the on-line Steiner tree problem on the Euclidean space, and the same problem on the rectilinear metric space.

References

[1] N. Alon and Y. Azar, On-Line Steiner Trees in the Euclidean Plane, *Procedings of the 8th Symposium of Computational Geometry,* Berlin, 1992.

[2] A. Agrawal, P. Klein and R. Ravi, When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem in Networks, *Proceedings of the 23rd ACM Symposium on Theory of Computing*, (1991) 134-144.

[3] F. R. K. Chung and R. L. Graham, On Steiner Trees for Bounded Point Sets, *Geometrias Dedicata*, 11 (1981) 353-361.

[4] F. R. K. Chung and F. K. Hwang, The Largest Minimal Rectilinear Steiner Trees for a Set of n points Enclosed in a Rectangle with a Given Perimeter, *Networks*, 9 (1979) 19-36.

[5] L. Few, The Shortest Path and the Shortest Road Through n Points, *Mathematika*, 2 (1955) 141-144.

[6] Y. Bartal, A. Fiat, and Y. Rabani, Competitive Algorithms for Distributed Data Management, *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, (1992) 39-50.

[7] M. Hannan, On Steiner's Problem with Rectilinear Distance, *SIAM J. App. Maths.*, 14 (1966), 255-265.

[8] M. Imaze, and B. M. Waxman, Dynamic Steiner Tree Problem, *SIAM J. Disc. Math.*, 4(3) (1991) 369-384.

[9] J. Krarup and P. M. Pruzan, The Simple Plant Location Problem: Survey and Synthesis, *European Journal of Operations Research*, 12 (1983) 36-81.

[10] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[11] N. Maculan, The Steiner Problem in Graphs, *Ann. Dis. Maths.*, 31 (1987) 185-212.

[12] L. Nastansky, S. M. Selkow, N. F. Stewart, Cost-Minimal Trees in Directed Acyclic Graphs, *Zeitschrift für Operations Research*, 18 (1974), 59-67.

[13] M. J. Quinn, *Designing Efficient Algorithms for Parallel Computers*, McGraw-Hill Book Company, 1987.

[14] P. Winter, Steiner Problem in Networks: A Survey, *Networks*, 17 (1987) 129-167.

[15] R. T. Wong, A Dual Ascent Approach for Steiner Tree Problems on a Directed Tree, *Mathematical Programming*, 28 (1974), 271-287.

[16] R. T. Wong, Location and Network Design, in: M. O'hEigeartaigh, J.K. Lenstra and A. H. G. Rinnooy Kan, *Combinatorial Optimization: Annotated Bibliographies*, Wiley, New York, 1985, 127-147.