On-line Steiner Trees in the Euclidean Plane

Noga Alon *   Yossi Azar †

Abstract

Suppose we are given a sequence of n points in the Euclidean plane, and our objective is to construct, on-line, a connected graph that connects all of them, trying to minimize the total sum of lengths of its edges. The points appear one at a time, and at each step the on-line algorithm must construct a connected graph that contains all current points by connecting the new point to the previously constructed graph. This can be done by joining the new point (not necessarily by a straight line) to any point of the previous graph, (not necessarily one of the given points). The performance of our algorithm is measured by its competitive ratio: the supremum, over all sequences of points, of the ratio between the total length of the graph constructed by our algorithm and the total length of the best Steiner tree that connects all the points. There are known on-line algorithms whose competitive ratio is $O(\log n)$ even for all metric spaces, but the only lower bound known is of [IW] for some contrived discrete metric space. Moreover, for the plane, on-line algorithms could have been more powerful and achieve a better competitive ratio, and no nontrivial lower bounds for the best possible competitive ratio were known. Here we prove an almost tight lower bound of $\Omega(\log n / \log\log n)$ for the competitive ratio of any on-line algorithm. The lower bound holds for deterministic algorithms as well as for randomized ones, and obviously holds in any Euclidean space of dimension greater than 2 as well.

*Department of Mathematics, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel-Aviv, Israel. Supported in part by a U.S.A.- Israeli BSF Grant

†DEC Systems Research Center, 130 Lytton Ave. Palo-Alto, CA 94301.

1 Introduction

We consider the on-line Steiner tree problem in the Euclidean plane. The problem can be illustrated by the following example. Suppose a company is searching for oil in a certain (planar) region, and it is lucky enough to find a promising place for an oil well from time to time. In order to keep the communication between the various wells running smoothly, the company needs to maintain a connected system of roads that enables one to move from any well to any other one. This road system is in fact so crucial that it must be updated on-line; whenever a new well is found, the road system must be modified instantly. Suppose that the price of a road is proportional to its length, and thus the objective is to minimize the total length of roads. Obviously once an amount is spent building a road, it cannot be recovered by omitting it, and hence we may assume that the communication system is updated only by adding new routes to it. Moreover, the manager wishes to be able to come to the stock holders at the end of the searching process and show them that the total amount spent on the communication system is not much larger than what it could have been even if he had known all the places of the wells in advance.

This example and several similar ones that may arise naturally in the design of various networks suggest the following problem, that we call here the *on-line planar Steiner-tree problem.* Suppose we are given a sequence of n points $v_1, \ldots, v_n$ in the Euclidean plane, and our objective is to construct, on-line, a connected graph that connects all of them, trying to minimize the total sum of lengths of its edges. We assume that the points appear one at a time, v_i arriving at step i. At the end of step i, the on-line algorithm must construct a connected graph T_i that contains the points $v_1, \ldots, v_i$ by connecting the new point v_i to the previously constructed graph T_{i-1}. This can be done by joining v_i (not necessarily by a straight line) to any point of T_{i-1}, which need not necessarily be one of the previously given points v_j. An algorithm A for the above problem is a procedure that decides how to construct the graphs T_i for every given sequence of points v_i. Let $A(v_1, \ldots, v_n)$ denote the total length of the last graph T_n constructed by the algorithm on the input $v_1, \ldots, v_n$, and let $OPT(v_1, \ldots, v_n)$ denote the minimum possible length of a connected graph in the plane that contains all the points v_i, i.e., the length of the optimal Steiner tree for this set of points. The performance of the algorithm A is measured by its competitive ratio: the supremum, over all sequences $v_1, \ldots, v_n$ as above, of the ratio

$$\frac{A(v_1, \ldots, v_n)}{OPT(v_1, \ldots, v_n)}.$$

(Note that the exact computation of $OPT(v_1, \ldots, v_n)$ is in fact NP-hard (see [GJ]), but there are simple polynomial algorithms that approximate it up to a constant factor).

The Steiner tree problem is an extensively studied problem that has been considered not only in the plane but also in arbitrary metric spaces. Given a set $S = S_n$ of n vertices $\{v_i\}_{i=1}^n$ which are points from a connected metric space, the *minimum Steiner tree* for these points is the tree T of minimum weight that connects all the vertices in S. Here the weight of a tree is the sum of weights of its edges where a weight of an edge is the distance between its two end points. The Steiner tree problem is that of finding the minimum Steiner tree for a given set of points in a metric space, or good approximations to it. This problem plays an important role in the study of various communication networks. A survey can be found in [Wi]. The on-line version of this problem, described above for the planar case, is defined for a general metric space in the obvious way.

We evaluate the performance of on-line algorithms by the competitive ratio measure, introduced by [ST], which has received a considerable amount of attention recently in the study of various algorithmic problems. For the Steiner tree problem (in any metric space) the competitive ratio of an on-line algorithm is the supremum, over all possible sets S, of the ratio between the weight of the connected graph constructed by the algorithm and the weight of the optimal Steiner tree for the set S. Note, that we must assume that the points are drawn from a metric space, i.e. the triangle inequality holds. This assumption is essential, since otherwise the competitive ratio can be easily made unbounded as a function of n.

A natural simple on-line algorithm for constructing on-line a Steiner tree is the *greedy algorithm*. At each step i simply join v_i to its closest point in T_{i-1}. The *vertex greedy algorithm* is to join v_i to its closest neighbor in the set $\{v_1, \ldots, v_{i-1}\}$. Observe that the greedy algorithm always performs at least as well as the vertex greedy algorithm. As shown by Imase and Waxman in [IW] (using the ideas of [RSL]) the (vertex) greedy algorithm achieves an $O(\log n)$ competitive ratio in every metric space. Moreover, they also showed that there are metric spaces in which the competitive ratio of any on-line algorithm is at least $\Omega(\log n)$; however, the examples demonstrating this lower bound are not embeddable in the Euclidean plane, and as suggested by the example in the beginning of this section, the planar case is of special interest. Moreover, the lower bound of [IW] is proved for a contrived metric space for which, at each step, an algorithm must choose between symmetric paths and it always makes the wrong choice. However, in the Euclidean plane on-line algorithms are not restricted, have infinitely many possibilities to choose from and cannot be characterized in a simple way. They may also use unexpected curves which depend in a complicated way on the whole history and the current state. Thus, the task of obtaining a nontrivial lower bound is much harder for the Euclidean plane. In fact, we can even restrict ourselves to the case in which all the points are drawn from the $n \times n$ grid while the algorithm, which knows this information, can connect vertices by any curve in the plane. The greedy algorithm is, of course, $O(\log n)$ competitive here, too, and one may wonder if there is a much better algorithm for this case. Our main result is that the $O(\log n)$ estimate is nearly optimal, as stated in the following theorem.

Theorem 1.1 *No on-line algorithm can achieve a competitive ratio which is better than $\Omega(\log n / \log\log n)$ for the Steiner tree problem of n points in the plane, or even for n points in the n by n grid.*

Although this gives a rather tight estimate for the best possible competitive ratio there is still an $O(\log\log n)$ multiplicative gap between our lower bound and the $O(\log n)$ upper bound given by the greedy algorithm, and it would be interesting to decide which of the two bounds is closer to the truth for the optimal algorithm. Our technique for proving the lower bound is totally different than the one used in [IW] for obtaining the (much simpler) lower bound for the non-Euclidean case.

Our proof can be modified to deal with randomized algorithms too. The cost of a randomized on-line algorithm is defined to be the expected cost over all possible coin-flips performed by the algorithm, and the competitive ratio is defined as the supremum, over all n-points inputs, of the ratio between this cost and the cost of the best off-line algorithm on the same input. Our lower bound holds for the oblivious adversary, i.e., the one that does not get to know the coin-flips of the algorithm, and therefore, holds for all other types of adversaries (e.g. adaptive ones).

Theorem 1.2 *The $\Omega(\log n / \log\log n)$ lower bound for the competitive ratio of any on-line algorithm for the planar Steiner tree problem for n points in the plane holds for randomized on-line algorithms as well.*

Before concluding the introduction let us mention some related problems. It is interesting to note that for the on-line spanning tree problem, the situation is much simpler. There, the algorithm is not allowed to use Steiner points and thus can use only edges between current points. As observed in [CV], no on-line algorithm can perform better than the greedy algorithm for the on-line spanning tree problem. That follows from the fact that the cost of adding new vertex i by any on-line algorithm is at least $d(v_i, \{v_1, \ldots, v_{i-1}\})$, and the greedy algorithm (which is the same as vertex greedy) encounters precisely this cost. Therefore the competitive ratio for the best algorithm for this problem is $\Theta(\log n)$. Note also that for this problem algorithms cannot take advantage of the fact that the metric space is known in advance. Since the algorithms for the on-line spanning tree problem are so restricted the proofs of lower bounds are easy and not useful at all for Steiner trees.

It is well known that for any set S of points in an arbitrary metric space, the weight of the optimal traveling salesman tour for S must be at least that of the best Steiner tree for S and cannot exceed it by more than a factor of two. In fact these two problems are intimately related and there is a tight relationship between the nearest neighbor algorithm for the traveling salesman problem and the greedy algorithm for the Steiner tree problem (see, e.g., [RSL]). Various papers that explore properties of the traveling salesman problem and the nearest neighbor algorithm are [RSL, BS, Ne, Me, BM]. Our basic approach here resembles the one of Bentley and Saxe in [BS] but a few additional ideas are required.

The next section contains the main technical part of this short paper including the proof of Theorem 1.1 and a sketch of the modification needed to establish Theorem 1.2. In section 3 we present a very short proof of the $O(\log n)$ upper bound estimate for the greedy algorithm, first proved in a somewhat more complicated way in [IW].

2 The Lower Bound Proof

In this section we prove Theorem 1.1 and sketch the proof of Theorem 1.2. The metric space considered is the Euclidean plane and all the requested points that appear one at a time in the course of the algorithm will belong to the n by n grid. Define x by $x^{2x} = n$, so that x is $(\frac{1}{2} + o(1))(\log n / \log\log n)$. In order to prove the lower bound we next show that an adversary can construct a set of at most $2n$ points such that the weight of the optimal Steiner tree on these points is $O(n)$ whereas the

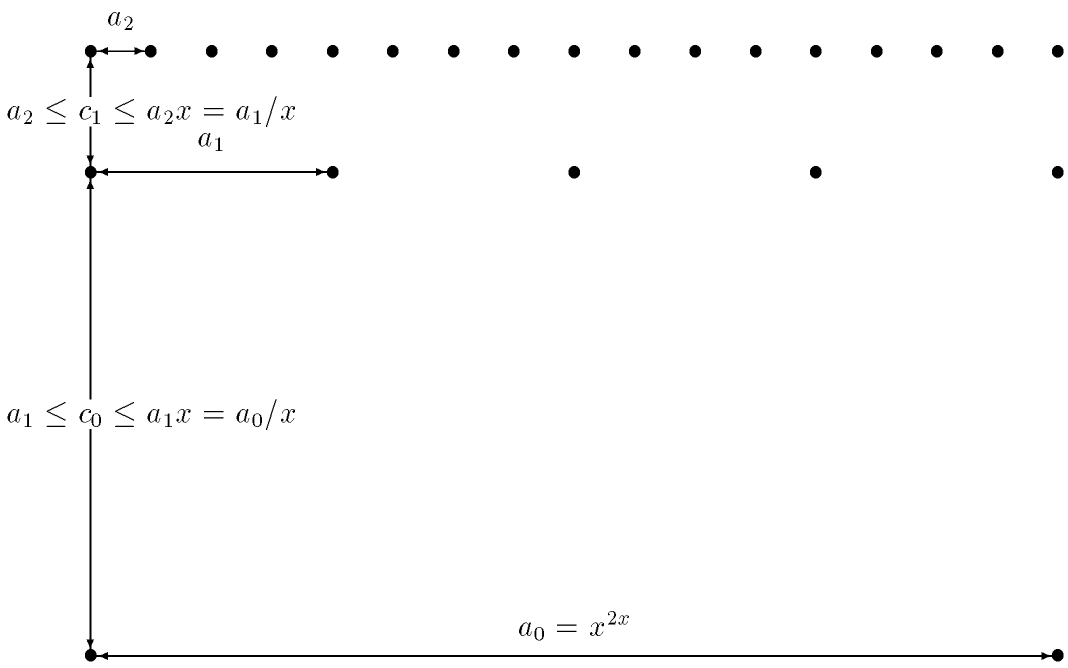

Figure 1: The construction for $x = 2$, $n = 16$

on-line cost of any algorithm will be at least $nx/8 = \Omega(n \log n / \log\log n)$. This yields a lower bound of $\Omega(\log n / \log\log n)$, as needed.

For simplicity of notation, we assume that x is an integer and omit all floor and ceiling signs; the proof can be repeated without this assumption with no real complications. Since we are interested in large values of n we assume that $x \geq 2$ and hence $n \geq 16$.

The points given by the adversary consist of x+1 layers, where each layer is a set of equally spaced points on a horizontal line of length $n = x^{2x}$. The coordinates of the points in layer i, $0 \leq i \leq x$, are (ja_i, b_i) where $a_i = x^{2x-2i}$ and $0 \leq j \leq n/a_i$. Thus $a_0 = x^{2x}(= n)$, $a_1 = x^{2x-2}$ and $a_x = 1$. Hence in layer 0 there are only two points, in layer 1 there are $x^2 + 1$ and so on up to layer number x which contains $n + 1$ points. (See Figure 1 for an example with $x = 2,\ n = 16$.) Let $b_0 = 0$. The vertical distance between layer number i and layer number $i + 1$ is $c_i = b_{i+1} - b_i$, where for all i,

$$a_{i+1} \leq c_i \leq a_{i+1}x = a_i/x.$$

In fact, $c_i = ka_{i+1}$ for some integer k between 1 and x, that is chosen by the adversary. The adversary presents the points to the algorithm layer by layer (bottom to top), where in each step it chooses c_i carefully forcing the algorithm to work hard in building its on-line Steiner tree. Note that the on-line algorithm knows all the information stated above (e.g. the x-coordinate of all points and the range of c_i for all i). The only piece of information which the on-line algorithm does not get to know is the exact value of c_i till it is done with all the points in layer i.

First, observe that the total number of points presented is

$$\sum_{i=0}^{x}(n/a_i + 1) = \sum_{i=0}^{x}(n/x^{2x-2i} + 1) = \sum_{i=0}^{x}(n/x^{2i} + 1) \leq 2n.$$

Note also that

$$b_x = \sum_{i=0}^{x-1} c_i \leq \sum_{i=0}^{x-1} a_i/x = n/x \sum_{i=0}^{x-1} \frac{1}{x^{2i}} \leq 2n/x \leq n$$

and therefore all the points lie, indeed, in the n by n grid.

Next observe that the length of the optimal (off line) Steiner tree is at most $O(n)$. Indeed, one can take the horizontal line in the last layer (layer number x) together with vertical lines from it to any other point. The total length of this tree is

$$n + \sum_{i=0}^{x-1} c_i(\frac{n}{a_i} + 1) \leq n(1 + \sum_{i=0}^{x-1} 2c_i/a_i) \leq n(1 + x\frac{2}{x}) = 3n.$$

The next lemma shows that the adversary can force the on-line algorithm to construct a connected graph of total weight $\Omega(nx)$. The on-line algorithm cannot imitate the adversary's tree since it does not know the exact value of c_i till it is done with all the points in layer i. Guessing the value, or being prepared for different values or any other strategy turn out to be either useless or too expensive as shown by the next lemma.

Lemma 2.1 *For each i, $1 \leq i \leq x$, the adversary can choose c_{i-1} in such a way that when it reveals the points in layer i to the on-line algorithm this algorithm will have to add total length of at least $n/8$ for connecting the new given points, unless the total length of the graph it has before these points appear is already at least $nx/8$.*

Once the lemma is proved, the assertion of the theorem follows easily. Indeed, if for some i it turns out that the on-line algorithm already has a graph of total length at least $nx/8$, there is nothing to prove. Otherwise, by the lemma, the adversary can force the algorithm to add total length of at least $n/8$ for each layer, giving again the required total $nx/8$ length, as needed.

It remains to prove the lemma. To this end, we consider, for a fixed i, the various possibilities to choose $c_{i-1} = ka_i$ where k is an integer in the range 1 to x. We must show that either the algorithm has to pay at least $n/8$ for at least one of these choices, or it has already paid at least $nx/8$.

For a fixed i, there are x possibilities to place the horizontal line of layer i, and for each such possibility, there are $(\frac{n}{a_i}+1)$ points. Consider the set of $(\frac{n}{a_i}+1)x$ points which is the union of points on all possible line placements of layer i. Let P be the set of all $(\frac{n}{a_i}+1)x \geq nx/a_i$ discs C_p whose centers are these points, where the radius of each disc is $a_i/2$. Note that since the distance between any two centers is at least a_i the interiors of the discs are pairwise disjoint. Define another set Q of discs C_q as follows. For each disc in P we have a disc in Q with the same center but with radius $a_i/4$. Clearly, $|P| = |Q|$ and the discs in Q are also pairwise disjoint. In fact the distance between any two of them is at least $a_i/2$.

Let T_{i-1} denote the graph of the on-line algorithm just before it gets the points in layer i. For each k, $1 \leq k \leq x$ let r_k be the number of discs in Q whose centers are on the k-th possible line of layer i, and no point in the disc contains any point of T_{i-1}. Put $r = \sum_{k=1}^{x} r_k$. If $r \leq \frac{nx}{2a_i}$ then there are more than $\frac{nx}{2a_i}$ discs of Q that contain points of T_{i-1}. But T_{i-1} is connected and contains points outside each disc in P (since it contains the points in layer $i-1$). Therefore for each disc C_q in Q that contains a point of T_{i-1} there must be a path that connects this point to a point on the boundary of the corresponding disc C_p in P that contains it. This path lies in the interior of C_p and its length is clearly at least $a_i/2 - a_i/4 = a_i/4$. Since there are at least $\frac{nx}{2a_i}$ such paths and they are pairwise disjoint we conclude that the total weight of T_{i-1} is at least $\frac{nx}{2a_i} \cdot \frac{a_i}{4} = nx/8$ as needed. (In fact this estimate can be improved by a factor of 2 by being a little bit more careful, but we make no attempt to optimize the constants here and in what follows).

It remains to check the case $r \geq \frac{nx}{2a_i}$. In this case there exists an l such that $r_l \geq \frac{n}{2a_i}$. The adversary can now choose $c_{i-1} = a_i l$. In order to connect the points of layer i corresponding to this choice of c_{i-1} to T_{i-1} the algorithm must add at least r_l paths connecting the centers of the discs in Q that lie in this horizontal line and contain no point of T_{i-1} to the circles bounding them, (since these centers have to be joined to the graph). These paths are pairwise disjoint (and are disjoint from T_{i-1})

as each of them lies completely inside the corresponding member of Q. Therefore, the algorithm must pay at least $r_l \cdot \frac{a_i}{4} \geq \frac{n}{2a_i} \cdot \frac{a_i}{4} = n/8$ total length, completing the proof of the lemma and hence that of Theorem 1.1. $\square$

The proof can be modified to apply to the randomized case (and hence establish Theorem 1.2). The argument, that is sketched below, combines the easy direction of a result of [Ya] with the above construction. More precisely, since any randomized algorithm is simply a probability distribution on deterministic ones it suffices to establish a lower bound for the expected time of deterministic algorithms over some probability distribution of the input. Thus, we define a distribution on the possible inputs such that the expected cost for any on-line algorithm is $\Omega(nx)$ while the cost of the off-line one on each possible instance is $O(n)$. The adversary can use essentially the same construction, but since it cannot compute the actual values of r_k for $1 \leq k \leq x$, it will choose the value of k uniformly at random for each layer independently. Consider any on-line algorithm: if the expected value of $r = \sum r_k$ at some step i is at most $\frac{nx}{2a_i}$ we are done since the expected weight of the graph is already at least $nx/8$ (by linearity of expectation). Otherwise, for each i the expected value of r at step i is at least $\frac{nx}{2a_i}$. Thus, for each i, the expected value of r_l at step i (when l is chosen at random) is at least $\frac{n}{2a_i}$. Thus, the expected cost for the algorithm at each step is at least $n/8$ and, therefore, the expected weight of the final graph is at least $nx/8$, completing the proof. $\square$

3 The Upper Bound

In this section we give a simple proof for the upper bound theorem of [IW] for the competitive ratio of the vertex greedy algorithm. We hope that such a simple proof may shed more light on the behavior of the algorithm.

Theorem 3.1 *The (vertex) greedy algorithm achieves a competitive ratio of $O(\log n)$ for the Steiner tree problem for n points in any metric space.*

Proof: We need the following.

Lemma 3.1 *Let l be the length of the optimal Steiner tree. The number of steps in which the greedy algorithm pays more than $2l/k$ is less than k*

Proof of Lemma: Let S be the set of points whose addition caused the greedy algorithm to pay more than $2l/k$. Clearly the distance between any two of them is more than $2l/k$. Thus the length of the shortest Hamilton tour on these points is more than $|S|2l/k$ and hence the weight of the optimum Steiner tree for them is more than $|S|l/k$. Since S is a subset of the original set of points the weight of the Steiner tree of S is at most l implying that $|S| < k$. □

To complete the proof of the theorem observe that the lemma implies that the weight of the k'th largest edge of the tree constructed by the on-line algorithm is at most $2l/k$ and thus its total weight is at most $\sum_{k=1}^{n} 2l/k = O(l \log n)$. □

4 Acknowledgement

We would like to thank Leo Guibas and John Hershberger for helpful discussions.

References

[BM] B. Bollobás and A. Meir, *A traveling salesman problem in the k-dimensional unit cube, Operations Research Letters*, to appear.

[BS] J.L. Bentley and J.B. Saxe, *An analysis of two heuristics for the Euclidean Traveling salesman, 18th Annual Allerton Conference on Communication, Control and Computing,* Monticello, 1980, pp. 41-49.

[CV] B. Chandra and S. Vishwanathan, *Constructing reliable communication networks of small weight on-line*, Manuscript.

[IW] M. Imase and B.M. Waxman, *Dynamic Steiner tree problem, SIAM J. Disc Math. 4,* (1991), pp. 369-384.

[GJ] M. R. Garey and D. S. Johnson, *Computers and Intractability: a guide to the theory of NP-completeness*, Freeman and Company, New York, 1979.

[Me] A. Meir, *A geometric problem involving the nearest neighbor algorithm, Operations Research Letters 6* (1987), pp. 289-291.

[Ne] D.J. Newman, *A problem seminar, Springer,* Berlin 1982, 9, Problem 57.

[RSL] D.J. Rosenkrantz, R.E. Strearns and P.M. Lewis II, *An analysis of several heuristics for the traveling salesman problem, SIAM J. Computing 6*, (1977), pp. 563-581.

[ST] D. Sleator and R. Tarjan, *Amortized efficiency of list update and paging rules, Communications of the ACM 28*, 2 (1985), 202–208.

[Wi] P. Winter, *Steiner problem in networks, a survey*, Networks 17 (1987), pp. 129-167.

[Ya] A. C. C. Yao, *Probabilistic computation: towards a unified measure of complexity*, Proc. 18th Annual IEEE FOCS, Providence, RI (1977), pp. 222-227.