Online Algorithms for Network Design

Adam Meyerson[1]

February, 2003

CMU-CS-03-115

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We give the first polylogarithmic-competitive online algorithms for two-metric network design problems. These problems are very general, including as special cases such problems as steiner tree, facility location, and concave-cost single commodity flow.

[1]Aladdin Project, Carnegie-Mellon University. Research supported by NSF grant CCR-0122581. Email: adam@cs.cmu.edu

Keywords: Online Algorithms, Network Design, Approximation Algorithms, Concave-Cost Flow

1 Introduction

Network design problems require purchasing various graph edges in order to connect a set of vertex demands. Problems within this domain include various integer flow problems, the steiner tree problem [38], facility location variants [29, 23, 3, 15, 25, 14, 13, 18, 37], access network design [2, 21], and buy-at-bulk network design[39, 5, 19]. Perhaps the most general formulation of network design is the problem of finding concave-cost flows. This problem has been well-studied in the operations research literature [40, 10, 9]. From a computer science theory standpoint, assuming that concave functions can be approximated in a piecewise linear manner, an $O(\log n)$ approximation for concave cost flow was given by [34] and later derandomized by [12].

Applications for network design are numerous. Some of the first applications in operations research involved transportation networks [17, 7] and placement of warehouses [30, 27]. In the computer science domain, similar techniques have been used to design telecommunications networks [2], select locations for web caches [26, 28], and classify large databases [20, 36].

We observe that many of the natural applications for network design are effectively online. New demands (customers, network users) will arrive after some portion of our network structure has been created. We would like to maintain a low-cost network (when compared to the best we could have done) at all times, avoiding "mistakes" due to a lack of foreknowledge as well as keeping our computation time reasonable.

This observation is not new; previous work has dealt with online versions of network design problems including flows [4, 6], steiner trees [8, 1, 22], facility location [32, 33], and even access network design [35].

We present the first online competitive algorithms for the problems of bounded diameter network design [31] and cost-distance network design [34], two of the most general problems which have been considered in the network design regime. The cost-distance network design problem includes as special cases such problems as steiner tree, facility location (with soft capacities), buy-at-bulk network design, the maybecast problem [24], and various natural combinations of those problems. For full details of these reductions, the reader is referred to [34]. For the problems we discuss, the best polynomial-time (offline) approximation known is $O(\log n)$, although there are no matching lower bounds and a constant approximation could potentially be feasible. No previous online approximations were known for these problems. In general, since steiner tree is a special case of the problems presented, we will not be able to produce better than an $O(\log n)$ *online* competitive result regardless of the computation time used or the relevent offline hardness results. We will present the first polylogarithmic-competitive algorithms for each problem.

Our techniques resemble those used for access network design [35]. The main idea is to divide the solution into a number of levels and assign each incoming demand to a level randomly, then connect through solely increasing levels. However, we are dealing with a much more general problem than access network design, and we will not be able to obtain the competitivity bounds of [35] (in fact, there are lower bounds indicating that no online algorithm can obtain such competitivity for the problems we study). Because of this, our analysis is quite different.

Several previous papers have made use of an assumption that the demands arrive in random (rather than adversarial) order [35, 33] to improve the competitivity results. In our case, the lower bound from online steiner tree will hold even if the order of arrival is random. With this in mind we will consider only adversarial order of arrival, as is traditional in the online algorithms community [11].

2 Online Bounded Diameter Network Design

We are given a graph $G = (V, E)$ along with a set of steiner nodes S. We additionally have two measures on the edges, $l : E \to R^+$ and $c : E \to R^+$, which we call cost and length. We would like to select a subtree T of the graph such that all the steiner nodes are members of the subtree $S \subset T$. We must guarantee that the diameter of T along the length metric is bounded by D ($diam_l(T) \leq D$), and would like to minimize the cost of the tree ($\sum_{e \in T} c(e)$) with these constraints.

This problem was introduced by Marathe et al [31]; they provided an algorithm with an $O(\log n)$ stretch on both the cost and diameter. We will consider the *online* version of the problem where the steiner nodes are not known in advance. Nodes announce themselves as members of S one at a time, and we must add them to the tree while maintaining bounded diameter and approximately optimum cost. If we set the diameter D to be infinitely large, this problem becomes equivalent to online steiner tree, giving us a $\log n$ lower bound for online competitivity [8, 1, 22].

As each node arrives online, we will assign it a *level* between 1 and $\log n$ (here $n = |S|$). The first steiner node is effectively assigned infinite level. Subsequent nodes are assigned level i with probability $\frac{1}{2^i}$. When a new steiner node arrives, we will find path P from the new node to some existing steiner node of strictly larger level. We select this path such that $c(P)$ is minimized, subject to the constraint that $l(P) \leq D$.

Lemma 2.1. *The algorithm described produces a tree with diameter at most* $diam_l(T) \leq O(\log n)D$.

Proof. Consider the path from any steiner node to the first node to arrive. This path passes through at most $1 + \log n$ levels. The path from each node to the node of the next level has length at most D because of the way the paths are selected. It follows that the total length of the path from any steiner node to the first-arriving node is at most $D \log n$, giving the tree a diameter bounded by $2D \log n = O(\log n)D$. ☐

Lemma 2.2. *The cost of the tree constructed is at most* $O(\log^2 n)C^*$ *where* C^* *is the cost of the optimum offline solution satisfying the required properties.*

Proof. We decompose the optimum tree into two disjoint components each containing at least $\frac{n}{3}$ nodes. We repeat this process recursively, yielding a decomposition of the optimum tree structure through $O(\log n)$ levels.

Instead of connecting each steiner node to the closest node of higher level, we consider connecting to a node of higher level in our current component of the optimum. If no such steiner node has arrived yet, we look at the next level of the decomposition, and so forth, until we find a node of higher level. Since this process may connect the current steiner node to a further place, its cost is only higher than the algorithm's cost.

Consider any given component of the optimum, decomposed into two regions. How many times will we connect a node of level i from one side of this decomposition to a node on the other side? We expect 1 node of level i to arrive on one side before a node of level $i+1$ arrives. Once a node of level $i+1$ arrives, future nodes of level i will connect to this node rather than crossing to the other side of the set. It follows that we expect a total of $\log n$ connections which cross from one region to the other. Each such connection has cost at most the entire cost of the optimum component. Any edge of the optimum solution appears in at most $O(\log n)$ components (one at each level of the decomposition), so the edge will be charged at most $O(\log^2 n)$ times (once for each connection which crosses the divide in a region which includes the edge). It follows that the cost of our tree is at most $O(\log^2 n)$ times the cost of the optimum. $\square$

Theorem 2.3. *The algorithm described gives an $O(\log^2 n), O(\log n)$ competitive algorithm (where the first number is the increase over optimum cost and the second is the increase in the diameter).*

If the value of n is not known in advance, we can start with only one level and increase the number of levels as more nodes arrive. Already-arrived nodes of the maximum level will join the next higher level with probability $\frac{1}{2}$, maintaining the desired structure of level probabilities.

3 Online Cost-Distance

We are given a graph $G = (V, E)$ along with a sink node $t \in V$ and a set of steiner nodes S. Each edge has a cost $(c : E \to R^+)$ and length $(l : E \to R^+)$. Our goal is to select a subset T of the edges such that $\sum_{e \in T} c(e) + \sum_{s \in S} d_T(s,t)$ is minimized, where $d_T(s,t)$ is the total length $l(e)$ of edges along the shortest path from s to t in T.

This problem was introduced by Meyerson et al [34] and an $O(\log n)$ approximation was given. We will consider the online version of the problem where the steiner nodes are not known in advance. Nodes announce themselves as members of S one at a time, and we must add them to T while maintaining a cost which is competitive against the optimum. We will give a polylogarathmic-competitive algorithm for this problem.

We will first define a new cost and length metric. For each edge and integer i between 1 and $\log n$, we will define a new edge with cost $C(e) = c(e) + 2^i l(e)$

and length $L(e) = \frac{1}{2^i}c(e) + l(e)$. Note that this new cost and length is strictly greater than the old. On the other hand, if we consider the optimum solution according to the old edges, if the solution sends flow $2^i \leq f \leq 2^{i+1}$ on some edge, we can use the new edge copy number i and the cost increases by at most a factor of 3. It follows that the optimum on the new costs is at most 3 times more expensive than the old optimum.

As each steiner node v arrives, we will assign it a level $i \leq \log n$ with probability $\frac{1}{2^i}$. Let S_i represent the subset of steiner nodes which are assigned to level i. We select a path which visits existing nodes in increasing order of assigned level. Nodes of level i will use outgoing edge copies of type i, such that their cost and length are in a 2^i ratio with one another.

Suppose the path P_v from node $v = w_i$ visits (in order) $w_{i+1}, w_{i+2}, ..., w_{\log n}, t$. We select these nodes and the paths between them such that we will minimize the value of $\sum_{j=i}^{j=\log n} l(w_j, w_{j+1})$. We show that the total length generated by this algorithm is not too large. On the other hand, the fixed cost could be expensive, since a given node v might "change its mind" about the best path to the sink any number of times (thus paying fixed cost over and over again). We will describe how to modify the algorithm to bound the cost of these changes without greatly increasing total lengths.

Lemma 3.1. *There is a solution which takes the form of a binary tree, such that the demand on every edge is a power of two, which has cost $C^* + L^* \leq OPT(\log n)$.*

Proof. Consider sending flow upwards along the optimum tree from the steiner nodes. Instead of accumulating all the flow at a given point, we will send separately all powers of two. The total flow along any edge in this model is unchanged. However, we have separately purchased each edge up to $\log n$ times (once for each power of two amount of demand). This produces a new tree with cost at most $\log n$ times the original cost, where every edge sends a power of two. If we have a node of high degree (in other words, the tree is not binary) we can place dummy nodes and edges of length zero to produce the required binary tree. □

Lemma 3.2. $E[\sum_{i=1}^{\log n} \sum_{v \in S_i} 2^i L(P_v)] \leq 2L^* \log^3 n$.

Proof. We consider the following routing scheme. We will send each node's demand upward along the optimum tree until we find that we are at the root of a subtree which contains a steiner node of level at least $i + 1$. We will then route downwards to this node, then progress upwards once again until we find a node of higher level, and so on. This routing scheme sends node v of level i to some w_{i+1}, w_{i+2} and so forth. Sine our original routing scheme finds the best such sequence of nodes, this modified routing scheme can only increase $L(P_v)$.

Now consider a single edge in the optimum solution. Suppose this edge emerges from a subtree which contains 2^δ nodes from S. How much will our routing scheme pay to send demand across this edge? We assume the optimum tree is binary and has depth at most $\log n$ by lemma 3.1. Consider each branch along the path from our edge to the root. Demand travels from this branch across our edge from vertices of type i only if this branch does not contain its own type $i+1$ vertex. We expect only 2^{i+1} nodes to arrive in this branch before it builds its own type $i+1$ vertex, and the expected demand carried by these nodes is at most $2^{i+1}\log n$. We conclude that the total demand crossing our edge from level i vertices is at most an expected $2^{i+1}\log^2 n$. On the other hand, if there is no vertex of level $i+1$ in our subtree, there cannot be any demand from level i or higher vertices. It follows that with probability at least $2^{\delta-i}$, there is no demand from level i vertices crossing the edge. Level i vertices pay $2^{-i}c(e)+l(e)$ on our edge e, so the total expected payment is bounded by:

$$\sum_{i\leq\delta} 2^{i+1}(\log^2 n)(2^{-i}c(e)+l(e)) + \sum_{i>\delta} 2^{\delta-i}2^{i+1}(\log^2 n)(2^{-i}c(e)+l(e))$$

We can bound this sum by $2^{\delta+1}(\log^3 n)l(e) + 2(\log^3 n)c(e)$. The optimum solution pays $2^{\delta}L(e) = 2^{\delta}l(e) + c(e)$ on this edge so we are within $O(\log^3 n)$ of optimum. □

Of course, this algorithm could pay very high fixed cost in terms of $\sum_{e\in T} C(e)$ because any given vertex could purchase many outgoing edges (in effect changing its mind many times about the shortest path) as new destinations arrive. We modify the algorithm as follows. When a node of type i arrives, we assume it comes with 2^i demand. When demand reaches some node v of type i, the demand simply accumulates at v (rather than being sent on to the sink via shortest path). Demand continues to accumulate at v until 2^i demand has accumulated, at which point we compute the shortest path through existing vertices (increasing values of i) to the sink. We send all 2^i demand along to the next node on this path (say w_{i+1}). The demand accumulated at v is reset to zero and now 2^i more demand is waiting at w_{i+1}. Using this delaying tactic guarantees that we will always send 2^i demand together from a node of type i, preventing us from sending tiny amounts of demand along many different outgoing edges. This enables us to bound the fixed cost as shown by the following lemma.

Lemma 3.3. *For the modified algorithm with delays, $E[\sum_{e\in T} C(e)] \leq O(\log^3 n)L^*$.*

Proof. The new algorithm sends 2^i demand along any outgoing edge from a vertex of type i. This is easy to see, since we only purchase edges after waiting for 2^i demand to accumulate. Additionally, vertices of type i use edges which have $C(e) = 2^i L(e)$. It follows that the total cost of edges is bounded by the total cost of in terms of $L(e)$ of moving demands from one vertex to another. Consider any demand. We send it along the shortest path, except that we might possibly delay at some intervening vertices. Delaying can only make paths shorter (as additional nodes might arrive while we wait), so we can conclude that $L(P_v)$ for any vertex v is at most the shortest path through the network of increasing vertex types when v arrived. It follows that $\sum_i \sum_{v\in S_i} 2^i L(P_v)$ will be only less in the delaying version of the algorithm than it would have been for the version where we immediately sent all demands along shortest path. Making use of lemma 3.2, we conclude that $E[\sum_{e\in T} C(e)] \le E[\sum_i \sum_{v\in S_i} 2^i L(P_v)] \le 2L^* \log^3 n$ yielding the desired bound. $\square$

We must now bound the total incremental cost of the solution, $\sum_{s\in S} d_T(s,t)$. At first glance, this appears to be just the sum of the path lengths $L(P_v)$ for every vertex $v \in S$. However, our algorithm delays some demands while waiting for 2^i to accumulate at a type i vertex.

Lemma 3.4. *At any time the algorithm with delays has* $E[\sum_{s\in S} d_T(s,t)] \le O(\log^3 n)L^*$.

Proof. We split the nodes $s \in S$ into two groups. First, we have the nodes whose demand actually reached t in the routing solution of our algorithm. For these nodes, we will have $d_T(s,t) \le L(P_s)$. However, there are other demands which were still being delayed at some intermediate node. The total demand waiting at a node of type i is bounded by $2^i - 1$ at any given time. On the other hand, we observe that our algorithm assumed a demand of 2^i arrived with each node of type i, whereas in reality only 1 unit of demand arrived at this time. It follows that each node of type i is sending out $2^i - 1$ units of "fake" demand in our algorithm. We can send the delayed demand along the path used by this "fake" demand, incurring the same cost. It follows that $\sum_{s\in S} d_T(s,t) \le \sum_i \sum_{v\in S_i} 2^i L(P_v)$, and using lemma 3.2 along with the observation that the paths are only shorter because of the delay mechanism gives the required result. $\square$

Theorem 3.5. *We have designed an* $O(\log^4 n)$ *competitive algorithm for online cost-distance.*

Proof. We combine lemmas 3.4 and 3.3 to show that the cost of our solution is at most an expected $O(\log^3 n)$ times the cost of the best binary tree solution $L^* + C^*$. Using lemma 3.1, this solution has cost at most $O(\log n)$ times the optimum cost, yielding the $O(\log^4 n)$ expected competitivity result. $\square$

There is an alternate formulation of the problem in which we must select a path P_v for each steiner node $v \in S$ as it arrives. Our current algorithm does not do this, since we may choose to delay the selection of the path beyond some particular node until more demand arrives. However, we can send each demand along the path of a preceding demand (from 2^i steps ago). Effectively, instead of amortizing the cost of the final delayed demands against the initial "fake" demand, we will amortize each set of 2^i demands against the preceding 2^i (with the initial demands being amortized against the initial "fake" demands). The algorithm is essentially the same and obtains the identical competitivity bounds.

4 Open Problems

This paper gives the first online algorithms for several network design problems. Of course, the competitive ratios could potentially be improved (to no better than $O(\log n)$).

The major remaining open problems in network design are in the offline domain. In particular, the problem of *multicommodity concave flow* remains open from an approximation standpoint. No non-trivial approximation algorithms are known for this problem. Assuming that the concave function is the same over all graph edges, an $O(\log n)$ approximation can be devised by combining the work of [5] with [16]. Another important open problem involves the approximability of cost-distance in the offline case. The best known approximation is $O(\log n)$ by [34] with a derandomization by [12] but the only known lower bound is from facility location, leaving the problem complexity open.

References

[1] N. Alon and Y. Azar. On-line steiner trees in the euclidean plane. *Discrete and Computational Geometry*, 10(2):113–121, 1993.

[2] Matthew Andrews and Lisa Zhang. The access network design problem. *IEEE Symposium on Foundations of Computer Science*, pages 40–49, 1998.

[3] V. Arya, N. Garg, R. Khandekar, V. Pandit, A. Meyerson, and K. Munagala. Local search heuristics for k-median and facility location problems. *Proceedings of the 33rd ACM Symposium on Theory of Computing*, 2001.

[4] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. *25th ACM Symposium on Theory of Computing*, pages 623–31, 1993.

[5] B. Awerbuch and Y. Azar. Buy-at-bulk network design. *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 542–47, 1997.

[6] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive online routing. *34th IEEE symposium on Foundations of Computer Science*, pages 32–40, 1993.

[7] M. Balinski. Fixed cost transportation problems. *Nav. Res. Log. Quarterly*, 8:41–54, 1961.

[8] P. Berman and C. Coulston. On-line algorithms for steiner tree problems. *Proceedings of the 29th ACM STOC*, 1997.

[9] D. Bienstock, S. Chopra, O. Gunluk, and C-Y. Tsai. Minimum cost capacity installation for multicommodity flow networks. *Mathematical Programming*, 81:177–199, 1998.

[10] D. Bienstock and O. Gunluk. Capacitated network design. *INFORMS Journal on Computing*, 8:243–259, 1996.

[11] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1996.

[12] C. Chekuri, S. Khanna, and S. Naor. A deterministic algorithm for the cost-distance problem. *Symposium on Discrete Algorithms*, 2001.

[13] F.A. Chudak. Improved algorithms for uncapacitated facility location problem. *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization*, 1998.

[14] F.A. Chudak and D. B. Shmoys. Improved approximation algorithms for capacitated facility location problem. *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, 1999.

[15] F.A. Chudak and D.P. Williamson. Improved approximation algorithms for capacitated facility location problems. *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization*, 1999.

[16] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *ACM Symposium on Theory of Computing*, 2003.

[17] P. Gray. Exact solution of the fixed charge transportation problem. *Operations Research*, 19:1529–1538, 1971.

[18] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[19] S. Guha, A. Meyerson, and K. Munagala. Improved combinatorial algorithms for single sink edge installation problems. *ACM Symposium on Theory of Computing*, 2001.

[20] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. *Proceedings of the 41st Symposium on Foundations of Computer Science*, 2000.

[21] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design problems. *IEEE Symposium on Foundations of Computer Science*, 2000.

[22] M. Imaze and B. Waxman. Dynamic steiner tree problem. *SIAM J. Discrete Math.*, 4(3):369–384, August 1991.

[23] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. *ACM Symposium on Theory of Computer Science*, 2002.

[24] D. Karger and M. Minkoff. Building steiner trees with incomplete global knowledge. *IEEE Symposium on Foundations of Computer Science*, 2000.

[25] M. Korupolu, C. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[26] M. Korupolu, G. Plaxton, and R. Rajaraman. Placement algorithms for hierarchical cooperative caching. *Proceedings of 10th ACM-SIAM SODA*, 1999.

[27] A. Kuehn and M. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9:643–666, 1963.

[28] B. Li, M. Golin, G. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the internet. *Proceedings of INFOCOM*, 1999.

[29] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. *APPROX*, 2002.

[30] A. Manne. Plant location under economies-of-scale-decentralization and computation. *Management Science*, 11:213–235, 1964.

[31] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III. Bicriteria network design problems. *Computing Research Repository: Computational Complexity*, 1998.

[32] R.R. Mettu and C.G. Plaxton. The online median problem. *IEEE Symposium on Foundations of Computer Science*, 2000.

[33] A. Meyerson. Online facility location. *IEEE Symposium on Foundations of Computer Science*, 2001.

[34] A. Meyerson, K. Munagala, and S. Plotkin. Cost-distance: Two metric network design. *IEEE Symposium on Foundations of Computer Science*, 2000.

[35] A. Meyerson, K. Munagala, and S. Plotkin. Designing networks incrementally. *IEEE Symposium on Foundations of Computer Science*, 2001.

[36] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming data for high-quality clustering. *IEEE Conference on Data Engineering*, 2002.

[37] M. Pál, É Tardos, and T. Wexler. Facility location with nonuniform hard capacities. *Proceedings of the 42nd IEEE Symposium on the Foundations of Computer Science*, 2001.

[38] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. *Proceedings of the 10th ACM-SIAM SODA*, 2000.

[39] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Buy-at-bulk network design: Approximating the single-sink edge installation problem. *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 619–628, 1997.

[40] W. Zangwill. Minimum concave cost flows in certain networks. *Management Science*, 14:429–450, 1968.