

Online Algorithms for Network Design

Adam Meyerson^{*}
University of California, Los Angeles
3731J Boelter Hall
Los Angeles, California 90095
awm@cs.ucla.edu

ABSTRACT

This paper presents the first polylogarithmic-competitive online algorithms for two-metric network design problems. These problems arise naturally in the design of computer networks (wired and wireless) as well as many other applications. In most cases the natural applications involve changing sets of communicating nodes, making the online model essential for addressing them.

Categories and Subject Descriptors

F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems Routing and Layout; C.2.1 [Computer-Communication Networks]: Network Architecture and Design Network Topology

General Terms

Algorithms, Design, Theory

1. INTRODUCTION

We consider two *online* problems in network design. In each case, we are given a multigraph on which each edge has a cost and a length. The cost function will be used to represent one-time costs for using the edge; this models properties such as physical installation cost or setup power consumption. The length function represents costs which will be paid by each node using the edge; this models properties like latency or the incremental cost (financial or in terms of power in a wireless network) of increasing the available bandwidth. Our goal is to select a subgraph to connect certain communicating nodes, while minimizing both the one-time costs and the total length. This can be viewed as simultaneously optimizing latency and power consumption (for example). The first problem, bounded diameter

^{*}Research partially supported by Aladdin Project, Carnegie-Mellon University through NSF grant CCR-0122581.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SPAA'04, June 27–30, 2004, Barcelona, Spain.
Copyright 2004 ACM 1-58113-840-7/04/0006 ...\$5.00.

network design, aims to minimize the total cost while maintaining the length (latency) to be at most a fixed bound. The second problem, cost-distance network design, aims to minimize a weighted sum of the total cost and average latency.

These problems have wide applications in the design of wireless (and wired) networks. In each case the natural applications are online, where nodes may join (and depart) the communicating group. Each problem is NP-Hard even in the offline case where the set of communicating nodes is static. We present the first *online-competitive* algorithms, thus dealing with adversarial arrival of communicating nodes. Our algorithms guarantee performance within poly-logarithmic factors of the best possible with full knowledge of future events and unlimited computing time. We note that our problems admit reductions from steiner tree, so their online variants will not have algorithms with competitive guarantees better than $O(\log n)$ against adversarial arrivals.

Our main technique involves randomly dividing the communicating nodes into levels. Each node communicates with a node on the next higher level. This level structure can be used to limit the tree depth and thereby control the latency, without greatly increasing the power consumption.

1.1 Previous Results and Applications

The two problems we consider were first considered in an offline setting (without arrivals or departures) by [32] and [35]. These results gave $O(\log n)$ approximations for the NP-Hard problems described. Our work extends this to the online setting.

These problems are very general. In particular, the problem of building trees under concave cost functions [35] includes many other network design problems as special cases. For a number of such special cases, better approximation algorithms are available. These include: the steiner tree problem [39], facility location variants [30, 25, 3, 15, 26, 14, 13, 18, 38], access network design [2, 19], buy-at-bulk network design [40, 5, 20], and the rent-or-buy problem [22, 23]. Concave-cost flows have a long history in the operations research literature as well [41, 10, 9].

Applications for this work are not restricted to design of computer networks. Earlier work discussed applying network design techniques to transportation networks [17, 7], placement of warehouses [31, 28], designing telecommunications networks [2], selecting locations for web caches [27, 29], and classifying large databases [21, 37].

We observe that many of these natural applications are effectively online. New demands (customers, network users,

database entries) will arrive after some portion of our structure has been created. In each case rebuilding our structure from the start will be costly or computationally inefficient.

This observation is not new; previous work has dealt with online versions of network design problems including flows [4, 6], steiner trees [8, 1, 24], facility location [33, 34], and even access network design [36]. This paper is the first to address the more general problems of two-metric network design in an online setting.

Several previous papers have made use of an assumption that the demands arrive in random (rather than adversarial) order [36, 34] to improve the competitiveness 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

We consider the problem of maintaining a communication tree of bounded latency and low installation cost. The latency and cost of an edge are distinct (perhaps unrelated) values. Nodes will join the communication tree in an online manner, and we must maintain connectivity, low cost, and low latency at all times. In general we will assume that the cost is paid upon connecting a pair of nodes, and that we cannot “reconnect” a node to a new parent without paying additional costs (if we completely remove this assumption, the problem reduces to the offline variant). For the time being, we will assume that there are only arrivals to the set of communicating nodes. Dealing with departures is straightforward and discussed in section 4. We pose this problem as an optimization question on graphs as follows.

We are given a graph $G = (V, E)$ along with a set of steiner nodes S , and a parameter D . We additionally have two measures on the edges, $l : E \rightarrow R^+$ and $c : E \rightarrow R^+$, which we call length and cost. 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 [32]; 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 competitiveness [8, 1, 24].

When the first node arrives, we assign it infinite level and set T to be that node alone. As each subsequent node arrives, we run the algorithm from figure 1. The algorithm takes polynomial time for each node; the search for a minimum cost path P_v may be performed using dynamic programming. We need to show that the tree T we construct at the end of the algorithm has bounded diameter and cost.

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 $1 +$

Input: Graph G , functions l, c , diameter target D , current tree T and a new node $v \in S$

Output: A new tree $T' \supseteq T$ such that $v \in T'$

1. Assign level i to v , where i is chosen from $1 \leq i \leq \log n$, level i with probability $\frac{1}{2^i}$
 2. Find a path P_v from v to a node of level at least $i + 1$ such that $l(P_v) \leq D$ and $c(P_v)$ is minimized.
 3. Let $T' = T \cup P_v$
-

Figure 1: Algorithm *Online Bounded Diameter*

$\log n$ 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 $2D \log n = O(\log n)D$. \square

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 2.1. *The algorithm described gives $O(\log^2 n)$ increase over optimum cost and $O(\log n)$ increase in diameter, when compared to the optimum (offline) solution.*

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 will now aim to minimize the sum of the total latency and the total cost. Once again the latency and cost on an edge will be viewed as unrelated. For this problem, it is possible to purchase many outgoing edges from a single node as new arrivals occur, but we will be forced to pay the cost for each of these edges. We again assume only arrivals to the communicating group; permitting departures is not difficult and will be described in section 4.

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 ($c : E \rightarrow R^+$) and length ($l : E \rightarrow R^+$). 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 [35] 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 polylogarithmic-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. When we refer to an edge of type i , we mean an edge which was thus scaled to have $C(e) = 2^i L(e)$, and we will let $L_i(u, v)$ represent the minimum length of a path from u to v using only type i edges.

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. \square

Our algorithm is described in figure 2. We will first bound the total length of the selected paths P_v . Of course, these are not the actual paths that will be used in the network since only the first step of such a path is necessarily followed. We define P_v to be the path which was initially selected for v when it arrived. The set S_i represent the steiner nodes of level i .

Input: Graph G , functions l, c , current tree T and a new node $v \in S$

Output: A new tree $T' \supseteq T$ such that $s \in T'$

1. Select a level $1 \leq i \leq \log n$, level i with probability $\frac{1}{2^i}$ and assign level i to v . Set $\tau(v) = 2^i$.
 2. While there exists any level i and node v of that level such that $\tau(v) \geq 2^i$, select a path P_v which visits nodes in increasing order of level, starting at $v = w_i$ and continuing through w_{i+1}, w_{i+2} and so forth ending at t . These nodes are selected to minimize $L(P_v) = \sum_{j=i}^{\log n} L_j(w_j, w_{j+1})$. Add an edge of type i from v to w_{i+1} . Set $\tau(v) = 0$ and increase $\tau(w_{i+1})$ by 2^i .
-

Figure 2: Algorithm *Online Cost-Distance*

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. Since 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:

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

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. \square

We now consider any edge in the network. We will continue using this edge until the lower-level adjacent node at-

tains $\tau(v) = 2^i$. We can use this fact to bound the total cost of the tree $c(T)$.

LEMMA 3.3. $E[\sum_{e \in T} C(e)] \leq O(\log^3 n)L^*$.

PROOF. We can view the algorithm as accumulating flow $\tau(v)$ at node v , then sending it all out together when 2^i accumulates. While this is not actually what occurs, the cost of the solution is equivalent since we can amortize the initial flow against the artificial 2^i units of flow starting at node v .

This “accumulating” algorithm sends 2^i demand along any outgoing edge from a vertex of type i . 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. Let R_v represent the route used by the “accumulating” algorithm to send from v to the sink. This is typically not the same as P_v (the original path selected for v) because the flow will typically accumulate at some intervening node. However these delays only make paths shorter (as additional nodes might arrive while we wait) so we can conclude that $L(R_v) \leq L(P_v)$. Any edge for which we pay the cost has 2^i demand so we conclude that $\sum_{e \in T} C(e) \leq \sum_{v \in S} L(R_v)$. Making use of lemma 3.2, we conclude that $E[\sum_{e \in T} C(e)] \leq E[\sum_i \sum_{v \in S_i} 2^i L(P_v)] \leq 2L^* \log^3 n$ yielding the desired bound. \square

We must now bound the total incremental cost of the solution, $\sum_{s \in S} d_T(s, t)$.

LEMMA 3.4. $E[\sum_{v \in S} d_T(v, t)] \leq O(\log^3 n)L^*$.

PROOF. We split the nodes $v \in S$ into two groups. First, we have the nodes whose demand would reach t in the “accumulating” version of our algorithm. For these nodes, we will have $d_T(v, t) \leq L(P_v)$. 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 $\sum_{v \in S} d_T(v, t) \leq \sum_i \sum_{v \in S_i} 2^i L(P_v)$, 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 3.1. *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 competitiveness result. \square

4. DEALING WITH DEPARTURES

The algorithms described considered only arrivals to the steiner set and not departures. Fortunately departures are easy to deal with. When a node departs, we simply reconnect its “children” into the tree by connecting to the nearest node of highest level (or constructing a path and connecting to the first node along that path, in the case of the second algorithm). Since we are now comparing against a *dynamic* optimum solution (otherwise our cost would be unbounded in comparing against a solution where many nodes arrive and then depart, since we had to serve those nodes somehow and the optimum final solution serves nothing) and the adversary is presumed unaware of our random choices, this will not change the competitive ratio. We observe that our construction still looks like a construction that could have been built without the removed nodes.

In addition, we consider the possibility of maintenance costs. Suppose that the “cost” portion of the result is paid at regular intervals (i.e. power consumption is per minute, not a one-time payment). We can simply recompute the tree at these intervals. If there is a setup cost which may exceed the maintenance cost, recompute with probability equal to the ratio of the maintenance over setup cost. This guarantees that we don’t switch too often (the expected maintenance cost equals the setup) but also that we don’t over-maintain a non-useful connection.

5. 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 [35] with a derandomization by [12] but the only known lower bound is from facility location, leaving the problem complexity open.

6. 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] M. Andrews and L. 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. Hierarchical placement and network design problems. *IEEE Symposium on Foundations of Computer Science*, 2000.
- [20] S. Guha, A. Meyerson, and K. Munagala. Improved combinatorial algorithms for single sink edge installation problems. *ACM Symposium on Theory of Computing*, 2001.
- [21] S. Guha, N. Mishra, R. Motwani, and L. O’Callaghan. Clustering data streams. *Proceedings of the 41st Symposium on Foundations of Computer Science*, 2000.
- [22] A. Gupta, A. Kumar, and T. Roughgarden. A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. *IEEE Symposium on Foundations of Computer Science*, 2002.
- [23] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. *ACM Symposium on Theory of Computing*, 2003.
- [24] M. Imaze and B. Waxman. Dynamic steiner tree problem. *SIAM J. Discrete Math.*, 4(3):369–384, August 1991.
- [25] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. *ACM Symposium on Theory of Computer Science*, 2002.
- [26] 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.
- [27] M. Korupolu, G. Plaxton, and R. Rajaraman. Placement algorithms for hierarchical cooperative caching. *Proceedings of 10th ACM-SIAM SODA*, 1999.
- [28] A. Kuehn and M. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9:643–666, 1963.
- [29] B. Li, M. Golin, G. Italiano, X. Deng, and K. Sohrawy. On the optimal placement of web proxies in the internet. *Proceedings of INFOCOM*, 1999.
- [30] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. *APPROX*, 2002.
- [31] A. Manne. Plant location under economies-of-scale-decentralization and computation. *Management Science*, 11:213–235, 1964.
- [32] 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.
- [33] R.R. Mettu and C.G. Plaxton. The online median problem. *IEEE Symposium on Foundations of Computer Science*, 2000.
- [34] A. Meyerson. Online facility location. *IEEE Symposium on Foundations of Computer Science*, 2001.
- [35] A. Meyerson, K. Munagala, and S. Plotkin. Cost-distance: Two metric network design. *IEEE Symposium on Foundations of Computer Science*, 2000.
- [36] A. Meyerson, K. Munagala, and S. Plotkin. Designing networks incrementally. *IEEE Symposium on Foundations of Computer Science*, 2001.
- [37] L. O’Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming data for high-quality clustering. *IEEE Conference on Data Engineering*, 2002.
- [38] 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.
- [39] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. *Proceedings of the 10th ACM-SIAM SODA*, 2000.

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