COST-DISTANCE: Two Metric Network Design

Adam Meyerson[*] Kamesh Munagala[†] Serge Plotkin[‡]

Abstract

We present the COST-DISTANCE *problem: finding a Steiner tree which optimizes the sum of edge costs along one metric and the sum of source-sink distances along an unrelated second metric. We give the first known $O(\log k)$ randomized approximation scheme for* COST-DISTANCE*, where k is the number of sources. We reduce several common network design problems to* COST-DISTANCE*, obtaining (in some cases) the first known logarithmic approximation for them. These problems include single-sink buy-at-bulk with variable pipe types between different sets of nodes, facility location with buy-at-bulk type costs on edges, constructing single source multicast trees with good cost and delay properties, and multi-level facility location. Our algorithm is also easier to implement and significantly faster than previously known algorithms for buy-at-bulk design problems.*

1 Introduction

Consider designing a network from the ground up. We are given a set of customers, and need to place various servers and network links in order to cheaply provide sufficient service. If we only need to place the servers, this becomes the facility location problem and constant-factor approximations are known [28, 12, 19]. If a single server handles all customers, and we impose the additional constraint that the set of available network link types is the same for every pair of nodes (subject to constant scaling factors on cost) then this is the single sink buy-at-bulk problem [27, 5]. We give the first known approximation for the general version of this problem to optimize both placement of servers and network topology.

*Supported by ARO DAAG-55-97-1-0221. Department of Computer Science, Stanford University CA 94305. Email: awm@cs.stanford.edu.

†Supported by ONR N00014-98-1-0589. Department of Computer Science, Stanford University CA 94305. Email: kamesh@cs.stanford.edu.

‡Supported by ARO DAAG55-98-1-0170 and ONR N00014-98-1-0589. Department of Computer Science, Stanford University CA 94305. Email: plotkin@cs.stanford.edu.

We reduce the network design problem to the following theoretical framework, which we call the COST-DISTANCE problem. We are given a graph with a single distinguished sink node (server). Every edge in this graph can be measured along two metrics; the first will be called *cost* and the second will be *length*. Note that the two metrics are entirely independent, and that there may be any number of parallel edges in the graph. We are given a set of sources (customers). Our objective is to construct a Steiner tree connecting the sources to the sink while minimizing the combined sum of the *cost* of the edges in the tree and sum over sources of the weighted *length* from source to sink. Note that our definition is a direct generalization of both the shortest path tree and the minimum cost Steiner tree. If *costs* and *lengths* are proportional, then constant-factor approximations [21, 6] are known.

We obtain the first general approximation algorithm for this problem with unrelated metrics. We prove an expected competitive ratio of $O(\log |S|)$ (where S is the set of sources) for our randomized algorithm. The algorithm is fairly simple to implement and runs in a relatively fast $O(|S|^2(m + n \log n))$ time bound.

Many standard problems in network design can be reduced to COST-DISTANCE. In particular, we describe simple reductions from single source buy-at-bulk and the metric facility location problem. Besides improving best-known performance bounds for single-source buy-at-bulk, we demonstrate that a natural combination of facility location and buy-at-bulk can be solved by reduction to COST-DISTANCE. In fact, we can generalize single-source buy-at-bulk to account for a scenario where not all network link types are available between every pair of nodes, or where costs do not scale linearly. This better models real-life situations where certain types of hardware may not be available (or may not be practical to install) in certain locations. Our algorithm provides the first known approximation for this more general problem. We also obtain the first known combinatorial approximations for the metric multi-level uncapacitated facility location problem [20, 1], and for constructing single source multicast trees with good cost and average per receiver delay [26].

From a more theoretical standpoint, consider routing single-source traffic through a graph where each edge has some function relating the total traffic along the edge to the cost of routing that traffic. If all functions are convex increasing (nondecreasing derivative) then exact solutions are known using min-cost flow techniques. We present the first approximations for the case where all functions are *concave increasing* (nonincreasing derivative). Previous work on buy-at-bulk [5, 4, 27] required that the concave functions between each pair of nodes be identical up to a constant scaling factor; we eliminate this requirement.

In addition to generalizing previous results, our algorithm is easy to implement and has a small running time. This makes it the algorithm of choice for many of the problems we have previously described. For example, previous algorithms for single-source buy-at-bulk depended on complicated methods of randomly selecting trees which approximate stretch [7, 8, 10, 11]. The algorithm for Access Network Design [4] depended on linear programming relaxation. Our algorithm's most time-consuming subroutine is single-pair shortest path.

Summary of Previous Results

If the cost and delay metrics are proportional, the offline version of COST-DISTANCE has constant factor approximation [6, 21], and there is an online algorithm performing a small number of rerouting of existing nodes [14]. If the cost and delay metrics are unrelated, this problem has no previously known approximation algorithm.

A related problem is to find a tree with low cost in the c metric such that the diameter is no more than L in the l metric. This problem has an $O(\log|S|, \log|S|)$ approximation on the cost and diameter [24, 23]. Our algorithm for COST-DISTANCE has the same basic structure and proof idea as the algorithm in [24].

Previous results for related network design problems are discussed in detail in Section 5.

2 The COST-DISTANCE Problem

We are given a graph $G = (E, V)$ along with a set of source vertices $S \subset V$ which need to be connected to a single sink vertex $t \in V$. We have two metrics along this graph. We will call the first metric cost, $c : E \mapsto \Re^+$ and the second metric length $l : E \mapsto \Re^+$. We are also given a weighting function $w : S \mapsto \Re^+$ on the sources. We denote the two metrics on an edge e as $(c(e), l(e))$.

We are asked to find a connected subgraph $G' = (E', V') \subset G$ which contains all sources ($S \subset V'$) and the sink ($t \in V'$) such that the following sum is minimized:

$$\sum_{e \in E'} c(e) + \sum_{s \in S} w(s) L'(s,t)$$

Here $L'(s,t)$ is the total length of the min-length path from s to t along the edges of G'.

Our algorithm will give an $O(\log|S|)$ approximation to this sum. It is important to notice that our approximation ratio does not depend on the number of edges, since there may potentially be a large number of edges connecting the same pair of nodes ($m \gg n^2$).

3 The Algorithm

The algorithm works by pairing up sources (or pairing sources with sink) until only the sink remains. At each stage we find a matching on the nodes, then choose one node from each matched pair to be "center." We transport the weight from the non-center node to the center, paying the appropriate edge costs and weight times distance costs. We then repeat this process on the centers until the sink is the only remaining node. The details of the algorithm follow.

1. Define $S_0 = S \cup \{t\}$ and $w_0 = w$. Create empty set E'.
2. Set $i = 0$.
3. For every pair of non-sink nodes $(u, v) \in S_i$:
   - Find the shortest $u - v$ path in G according to the metric $M(e) = c(e) + \frac{2w_i(u)w_i(v)}{w_i(u)+w_i(v)} l(e)$
   - Define $K_i(u, v)$ to be the length under metric $M(e)$ of this path.
4. For every non-sink node $u \in S_i$:
   - Find the shortest $u - t$ path in G according to the metric $M(e) = c(e) + w_i(u)l(e)$
   - Define $K_i(u, t)$ to be the length under metric $M(e)$ of this path.
5. Find a matching between nodes in S_i such that the number of unmatched nodes plus half the number of matched nodes is at most S_i/α and the value of $\sum_{(u,v) matched} K_i(u, v)$ is at most β times the value of the minimum K_i-cost perfect matching. We assume α and β are known constants.
6. For each matched pair (u, v) add the edges on the path defining $K_i(u, v)$ to the set E'.
7. Create an empty set S_{i+1}
8. For each pair of non-sink matched nodes (u, v):
   - Choose u to be the center with probability $w_i(u)/(w_i(u) + w_i(v))$. Otherwise v will be the center.

- Add the chosen center to S_{i+1} and assign the center a weight $w_{i+1}(center) = w_i(u) + w_i(v)$.

9. Add all unmatched nodes $u \in S_i$ to S_{i+1} and define $w_{i+1}(u) = w_i(u)$.

10. Add the sink to S_{i+1}.

11. If S_{i+1} contains only the sink, we are done. Otherwise increment i and return to step 3.

12. We return $G' = (E', V')$ where E' is the set of edges we constructed and V' is the set of adjacent nodes.

Each time through the steps, the size of our set S_i is reduced by α. Thus the process terminates after $\log_\alpha |S|$ iterations.

A little more detail is needed in step 5. We could find the minimum cost perfect matching on the set in polynomial time, obtaining $\alpha = 2$ and $\beta = 1$. Polynomial-time algorithms are known for minimum-cost perfect matching on non-bipartite graphs [25]. However, these algorithms tend to be impractical[1]. The following simpler procedure will work for us, causing only a small constant loss in our approximation ratio. We will find the cheapest pair of nodes to connect (minimum $K_i(u, v)$) and match them. We then remove these two nodes from consideration and repeat. We continue this process until half the nodes have been matched. The jth pair which we choose to match must have had matching cost at most equal to the $(2j - 1)$st cheapest edge in the perfect matching, according to the K_i metric. It follows that our total K_i-cost is at most half the K_i-cost of the perfect matching, guaranteeing $\alpha = 4/3$ and $\beta = 1/2$.

Each iteration of this algorithm finds shortest paths between all pairs in S_i. Since the metric is different for each pair, we cannot use all-pairs shortest path computations. Instead we perform $|S_i|^2$ single pair shortest paths. We first take $O(m)$ time to compute the metric on every edge. Using Dijkstra's Algorithm, we can compute the shortest path between a single pair of nodes in $O(m + n \log n)$ time. The matching step (stage 5) can be performed in $O(|S_i|^2 \log |S_i|)$ time. It follows that iteration i takes at most $O(|S_i|^2(m + n \log n))$ time. Since the size of S_i reduces by constant α each iteration, when we sum over iterations the total running time looks like $O(|S|^2(m+n \log n))$.

4 Analysis

The optimal solution will be a tree, which we will call T^*. To see this, notice that we can take any graph and produce the shortest-path (according to the length metric) tree connecting the sink to all sources. This shortest-path tree will have total cost at most the total cost of the graph and a distance-to-sink from every source node equal to the distance-to-sink from the graph. It follows that the optimal solution must be a tree, since a non-tree solution immediately gives rise to a tree solution with equal or superior total value.

We define the following quantities:

$C^* = \sum_{e \in T^*} c(e)$.

$L^*(v) =$ Total length of edges along the path from v to the sink in T^*.

$D^* = \sum_{v \in S} w(v) L^*(v)$.

The total "value" of the optimal solution which we will need to approximate is $C^* + D^*$. At each stage in our algorithm, we have some set of nodes S_i which we are trying to connect. We define the following potential function:

$$D_i^* = \sum_{v \in S_i} w_i(v) L^*(v)$$

Notice that $D_0^* = D^*$.

Since our algorithm is randomized, we need to analyze the expected performance. Each stage of the algorithm transports some weight from matched nodes to chosen centers. We can define the value of stage i to be the total cost of the edges used in stage i matching plus the cost to transport the weight across the appropriate edges to the center. The total value of our solution will then be the sum of the values of the stages.

We first prove a lemma bounding the expected potential function at each stage.

Lemma 4.1 *For every stage i,* $\mathbf{E}[D_i^*] \leq D^*$.

Proof: The proof will be by induction. For $i = 0$ we know $D_0^* = D^*$. Consider stage $i > 0$. Suppose we matched u and v in our previous matching. The contribution of u and v to D_{i-1}^* was $w_{i-1}(u)L^*(u) + w_{i-1}(v)L^*(v)$. We choose a random center. The expected distance from center to sink is now:

$$\frac{w_{i-1}(u)L^*(u) + w_{i-1}(v)L^*(v)}{w_{i-1}(u) + w_{i-1}(v)}$$

The weight of the new center is $w_{i-1}(u) + w_{i-1}(v)$. It follows that the new center's expected contribution to D_i^* is also $w_{i-1}(u)L^*(u) + w_{i-1}(v)L^*(v)$. Of course, unmatched nodes contribute equally much to both potentials, and nodes matched with the source contribute less to D_i^* since their weight will disappear. Thus the expected value of D_i^* is at most D_{i-1}^*. It follows that $\mathbf{E}[D_i^*] \leq \mathbf{E}[D_{i-1}^*]$ and the inductive hypothesis implies $\mathbf{E}[D_i^*] \leq D^*$. ∎

We will now relate the value of a stage to the metric K_i on which we approximated a min-cost perfect matching. This will allow us to bound the expected value of each stage.

[1] We could use the $O(n^2 \log n)$ approximation algorithm in [15] to get $\beta = 2$ and $\alpha = 2$. Here, n is the total number of nodes in the graph.

Lemma 4.2 *Given a tree $T = (E, V)$ and a set of nodes $S \subseteq V$, there exists a perfect matching of the nodes in S which uses each edge of the tree at most once.*

Proof: The proof will be by induction on the number of edges in the tree. If the tree includes zero edges, then $|V| = 1$ and the result is trivially true. Consider a larger tree. Suppose $v \in V$ is a leaf of this tree. If v is not included in S, then we can remove v and the edge connecting it to its parent from the tree to produce a smaller tree, T'. We inductively produce a perfect matching of the nodes in S on T' and use the same matching for T. If v is included in S, then we consider v's parent node. If the parent node is also in S, then we match v with its parent. We then remove v and it's edge from the tree to produce T' and inductively match the rest of S on T'. If the parent node is not in S, we produce S' by removing v from S and adding v's parent. We again produce T' and match the nodes. Some node u is matched to v's parent. We will use the identical matching to the one on T' except that we will match v with u by adding the edge from v to its parent to the relevent path. This produces the desired matching.

This result previously appeared in [24]. ■

Lemma 4.3 *The expected value of stage i at most $\beta(2D^* + C^*)$.*

Proof: Consider the tree T^*. By matching the nodes in S_i in the proper way, we can guarantee that we use only edges in T^* and no edge more than once as in lemma 4.2. This matching has edges with total cost at most C^*. The fraction $2w_i(u)w_i(v)/(w_i(u) + w_i(v))$ is at most twice the minimum of the two weights. Each edge in our matching would have to be along the path-to-source in the optimal tree for one of the two matched nodes. It follows that:

$$\sum_{(u,v) matched} K_i(u, v) \leq C^* + 2D_i^*$$

The minimum-cost perfect matching along metric K_i must do at least this well. Since the matching we actually use has cost at most β times the minimum-cost perfect matching, we guarantee a matching of K_i-cost at most $\beta(C^* + 2D_i^*)$. We need to relate this cost to the value of the stage.

The value of the stage is the total cost to transfer weight from matched nodes to their centers. Suppose we match u and v. If we choose v as the center, then we need to transport u's weight over to v. This induces a value of $w_i(u)l(u, v)$ in addition to the value induced by the cost of edges used. On the other hand, if we choose u as center then we pay $w_i(v)l(u, v)$ plus edge costs. The expected value is thus

$$\frac{w_i(v)w_i(u)l(u,v) + w_i(u)w_i(v)l(u,v)}{w_i(u) + w_i(v)} + c(u, v)$$

Notice that this expected value is exactly $K_i(u, v)$. It follows that the expected value of the stage is equal to the total K_i-cost of the matching found; at most $\beta(C^* + 2D_i^*)$. This of course depends on D_i^*, a random variable with expected value at most D^* (as per lemma 4.1). It follows that the expected value of stage i is at most $\beta(2D^* + C^*)$ as desired. ■

Theorem 4.1 *We obtain approximation ratio $2\beta \log_\alpha |S| = O(\log |S|)$ to the optimal.*

Proof: The expected value of our solution is equal to the sum of expected value of stages. This gives us total value $\mathbf{E}[V] \leq \sum_i \mathbf{E}[V_i]$. Using lemma 4.3 and our bound on the total number of stages, we can bound this by $\mathbf{E}[V] \leq \beta(\log_\alpha |S|)(2D^* + C^*)$. Since the optimal solution has value $D^* + C^*$, this proves the desired approximation ratio. ■

Using the described greedy algorithm to find a matching, we will attain expected approximation ratio $\log_{4/3} |S|$; exact perfect matchings would improve this to $2 \log_2 |S|$. There will be a small additional loss in the last stages where an uneven number of nodes could cause a few additional steps, however our total approximation will remain bounded by an expected $O(\log |S|)$.

We note that since the algorithm can optimize any linear combination of cost and distance, we can use the technique in [24] to obtain a $(O(\log |S|), O(\log |S|))$ approximation to the bicriteria problem of optimizing the cost given total distance and vice versa.

5 Relation to Network Design Problems

We will demonstrate approximation-preserving reductions from many commonly encountered network design probems to special cases of COST-DISTANCE. We emphasize that for all these problems, our algorithm produces a logarithmic approximation ratio, while being (in general) simpler to implement and faster to run than previously known algorithms.

5.1 Multicast Tree Design

In this problem, we are given a network $G(V, E)$ with costs and delays on the edges. We have a source node $s \in V$ and receivers $R \subseteq V$ for multicast data. The goal is to construct a tree T connecting the source to R so that the sum of the cost of the tree and the delays seen by the receivers is small. This problem is equivalent to COST-DISTANCE if the distance metric are the delays.

This problem has been extensively in the networks community [9, 18, 22, 26, 31], and many heuristics have been proposed. This problem also arises in the context of wire routing in programmable gate arrays and VLSI circuits [3]. We present the first approximation for this problem. Our approximation ratio is $O(\log |R|)$.

Note that our scheme works for the following problem as well. Each receiver i specifies a maximum delay D_i, and the goal is to construct the cheapest tree that satisfies all the delay requirements. The cost of our tree is $O(\log |R|)$ times the optimal cost, while the delay seen by receiver i is no more than $O(\log |R|)D_i$.

5.2 Metric Facility Location

We are given a weighted undirected graph $G(V, E)$ with a cost per unit demand $c : E \to \Re$ on the edges, which forms a metric. We have a set of demand points $D \subseteq V$ with demands d_i, and a set of facility locations $F \subseteq V$ with facility costs f_i. The goal is to open a subset of the facilities and assign demands to the open facilities so that the sum of cost of opening the facilities and the cost of routing the demand to the facilities is minimized.

For edge e in the graph, the bicriteria cost function is $(0, c(e))$. We add a dummy sink and connect it to all the facilities. For facility i, the cost of the edge is $(f_i, 0)$. The demand points will be our source vertices ($S = D$) and their weights will be equal to the demands ($w(v) = d_v$). The cost of a COST-DISTANCE solution on this modified graph is identical to the cost of its corresponding facility-location solution, so it follows that the reduction is approximation-preserving.

We can also consider the capacitated version of this problem, where facility i has capacity u_i. We can open multiple copies of a facility, but each copy opened at location i costs f_i. Again, we modify the graph exactly as before, but assign cost $(f_i, \frac{f_i}{u_i})$ on the edge connecting the sink to facility i. This causes the loss of an additional factor of two (at most) in the approximation ratio.

We have therefore obtained a $O(\log(|D|))$ approximation to these problems. Note that constant factor approximations are known for this problem [12, 19, 28].

5.3 Extended Single Sink Buy-at-bulk

In this problem [27], we are given a weighted graph $G(V, E)$ with length function $l : E \to \Re$. A subset $S \subseteq V$ of nodes have demands d_i. We have a special sink node t to which all this demand must be routed. The demand must be routed by choosing a tree and buying pipes along this tree. There are K types of pipes. The type i pipe has cost c_i per unit length and capacity u_i. We assume K is $O(\text{poly}(|V|))$. The goal is to minimize the total cost of pipe bought.

We modify the graph as follows. Replace every edge e in the graph with K parallel edges $e_1, e_2, \ldots, e_K$. The edge e_i has bicriteria cost $(l(e)c_i, l(e)\frac{c_i}{u_i})$. The weight of a node is its demand. This new graph is the instance of COST-DISTANCE that we solve. Intuitively, $l(e)c_i$ is the fixed cost of using pipe i, and $l(e)\frac{c_i}{u_i}$ is the incremental cost of routing demand.

It is implicit in the work of [5, 27] that the optimum tree with the modified cost function is no more than a factor 2 away from the optimum tree for the original problem.

The best previously known approximation for this problem is $O(\log |S| \log \log |S|)$ which follows by applying the techniques in [11] to the algorithm in [5]. These algorithms are based on the work in [7, 8, 10, 11] which show how to approximate any finite metric by a tree metric so that the distance between any two nodes in the graph is approximated well. For the special case of $K = 1$, Salman et al [27] showed a constant factor approximation by using previous results [6, 21] on balancing Steiner trees with shortest path trees.

All previous approximations assumed that all the K pipes are available between all pairs of nodes; it is straightforward to see that we can do away with this restriction. This problem arises naturally in network design. There may be a fixed cost of laying cables which depends on the location but is independent of the type of cable being laid (perhaps the cost of installing the cable outweighs the cost of the cable itself). Alternatively, certain types of services might not be available in certain locations. Our algorithm is the first to handle these sorts of situations.

5.4 Facility Location with Buy-at-Bulk

We can define a combination of the previous problems as follows. We are given the same graph as in the (capacitated) facility location problem, and also a set of K pipe types just as in the buy-at-bulk problem. We wish to open facilities and construct a forest routing the demands to the facilities. The demands must be routed by buying pipes along the edges of the forest. We wish to optimize the sum of the cost of laying out the pipes and the cost of opening the facilities.

This problem arises, for example, in placing caches over the web and connecting the demand points to the caches by laying out links of some fixed types (like T1, OC10, etc.) We wish to optimize the total cost of placing the caches and buying the links to route the demands.

It should be clear that the combination of the modifications we made in the previous problems gives an instance of COST-DISTANCE. The approximation ratio is therefore $O(\log |D|)$. This holds even if the set of available pipes differs for different pairs of nodes. As far as we are aware, this is the first approximation algorithm for this problem.

5.5 Multi-level Uncapacitated Facility Location

In the k-level facility location problem, we are given a graph $G(V, E)$ with a cost function c on the edges, and a set of demand nodes D. We have to route each demand through k levels of facilities. The cost of placing a facility of level i at location v is f_{iv}. The demand first goes to a facility of level 1, from there to a facility of level 2 and so on till it reaches a facility of level k. As in the classical facility location problem, the goal is to optimize the total cost of facility placement and the cost of routing the demands through the levels of facilities.

This problem has been extensively studied in operations research literature [20, 29, 30, 2, 28]. We note that Aardal, Chudak and Shmoys [1] present a 3 approximation for this problem. Their approach is however, not combinatorial.

We present the first known combinatorial approximation for general values of k. Our approximation ratio is $O(\log |D|)$. This result has been improved to a constant factor in [17].

We reduce the multi-level facility location problem to COST-DISTANCE. We first make k copies of G which we denote $G_1, G_2, \ldots, G_k$. We construct a new graph G' from these copies as follows. For an edge $e \in E$, its bicriteria cost in each of the copies is $(0, c(e))$. For $v \in G$, we add an edge between its corresponding vertices in G_i and G_{i+1} with cost $(f_{iv}, 0)$. We call these facility edges. We add a dummy sink node s and connect it to all vertices in G_k with edges of cost $(f_{kv}, 0)$ for vertex v. The demand nodes D map to the corresponding nodes in G_1.

Note that the only way to move from G_i to G_{i+1} is to take one of the level i facility edges. This means that the multi-level facility location problem is equivalent to COST-DISTANCE on the graph G' with sink s and the demands in G_1. We therefore have the following theorem.

Theorem 5.1 *There exists a $O(\log |D|)$ approximation for the k-level uncapacitated metric facility location problem running in $O(k|D|^2(|E| + |V| \log k|V|)$ time.*

5.6 Access Network Design

We note that the access network design problem [4] is a special case of the facility location with buy-at-bulk problem mentioned above. We therefore have a $O(\log |S|)$ approximation for this problem. This result has been improved to a constant factor in [17].

5.7 Concave Functions

Suppose we are given a graph and a set of sources and demands, and we wish to route all the demand to a single sink node. For every pair of nodes in the graph, we are given a *concave function* which determines the cost of routing between those nodes given the amount of demand to be transported. We can compute a tight approximation of such a concave function by viewing it as the minimum (at any demand value) of a series of lines of decreasing slope and increasing y-intercept. The bicriteria cost in the COST-DISTANCE problem can be seen to represent the y-intercept (*cost*) and the slope (*length*) of lines relating expenditure for an edge to amount of traffic routed. We can thus simulate the concave function by providing many parallel edges of different bicriteria cost.

6 Conclusions

We conclude by mentioning some open problems. First, we only have very weak lower bounds[2] on the approximability of COST-DISTANCE. We strongly suspect that the lower bound in the most general case is $\Omega(\log |S|)$. Second, it would be interesting to see if the algorithm provides better approximation guarantees for specific types of cost functions, specifically those arising from facility location[3]. Finally, we are curious as to whether any approximations can be given for the more general case where the cost-demand relationship for a pair of nodes in the graph follows an arbitrary nondecreasing function (neither convex nor concave).

There is a derandomization of the COST-DISTANCE algorithm due to Chekuri et al [13].

References

[1] K. Aardal, F. Chudak, and D.B. Shmoys. A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Information Processing Letters*, 72:161–167, 1999.

[2] K. Aardal, M. Labbé, J. Leung, and M. Queyranne. On the two-level uncapacitated facility location problem. *INFORMS J. Comput.*, 8:289–301, 1996.

[3] M. J. Alexander and G. Robins. New performance-driven FPGA routing algorithms. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 1996.

[4] M. Andrews and L. Zhang. The access network design problem. *39th IEEE Symposium on Foundations of Computer Science*, pages 40–49, 1998.

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

[2]The lower bound of 1.46 follows from [16].

[3]Some of the results in this paper have been improved in [17].

[6] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. *9th ACM Symposium on Principles of Distributed Computing*, pages 177–87, 1990.

[7] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *37th IEEE symposium on Foundations of Computer Science*, pages 184–193, 1996.

[8] Y. Bartal. On approximating arbitrary metrics by tree metrics. *30th ACM Symposium on Theory of Computing*, 1998.

[9] K. Bharath-Kumar and J.M. Jaffe. Routing to multiple destinations in computer networks. *IEEE Transactions on Communications*, 31(3):343–51, 1983.

[10] M. Charikar, C. Chekuri, A. Goel, and S. Guha. Rounding via trees: Deterministic approximation algorithms for group steiner trees and k-median. *30th ACM Symposium on Theory of Computing*, 1998.

[11] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin. Approximating a finite metric by a small number of tree metrics. *39th IEEE Symposium on Foundations of Computer Science*, 1998.

[12] M. Charikar and S. Guha. Improved combinatorial algorithms for facility location and k-median problems. *40th IEEE Symposium on Foundations of Computer Science*, 1999.

[13] C. Chekuri, S. Khanna, and J. Naor. A deterministic algorithm for the COST-DISTANCE problem. *manuscript*.

[14] A. Goel and K. Munagala. Balancing steiner trees and shortest path trees online. *11th ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[15] M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, April 1995.

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

[17] S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. *this proceedings*.

[18] S-P. Hong, H. Lee, and B.H. Park. An efficient multicast routing algorithm for delay-sensitive applications with dynamic membership. *IEEE INFOCOM*, pages 1433–40, 1998.

[19] K. Jain and V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. *40th IEEE Symposium on Foundations of Computer Science*, 1999.

[20] L. Kaufman, M. vanden Eede, and P. Hansen. A plant and warehouse location problem. *Operations Research Quarterly*, 28:547–557, 1977.

[21] S. Khuller, B. Raghavachari, and N. Young. Balancing minimum spanning and shortest path trees. *Algorithmica*, 14(4):305–321, 1994.

[22] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, 1(3):286–92, June 1993.

[23] G. Kortsarz and D. Peleg. Approximating shallow-light trees. *8th ACM-SIAM Symposium on Discrete Algorithms*, pages 103–110, 1997.

[24] 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.

[25] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., 1998.

[26] M. Parsa, Qing Zhu, and J.J. Garcia-Luna-Aceves. An iterative algorithm for delay-constrained minimum-cost multicasting. *IEEE/ACM Transactions on Networking*, 6(4):361–74, 1998.

[27] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Buy-at-bulk network design: Approximating the single-sink edge installation problem. *8th ACM-SIAM Symposium on Discrete Algorithms*, pages 619–628, 1997.

[28] D.B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. *29th ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[29] D. Tcha and B. Lee. A branch-and-bound algorithm for the multi-level uncapacitated location problem. *European J. Oper. Res.*, 18:35–43, 1984.

[30] T. Van Roy and D. Erlenkotter. A dual based procedure for dynamic facility location. *Management Sci.*, 28:1091–1105, 1982.

[31] L. Wei and D. Estrin. The trade-offs of multicast trees and algorithms. *International Conference on Computer Communications and Networks*, 1994.