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 (integrated logistics), constructing single source multicast trees with good cost and delay properties, priority Steiner trees, 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.

*A preliminary version of this work appeared in the 41^{st} IEEE Symposium on Foundations of Computer Science, 2000 [31]. This work was supported by ARO grant DAAG55-98-1-0170 and ONR grant N00014-98-1-0589.

†Department of Computer Science, University of California, Los Angeles CA 90095-1596. Email: awm@cs.ucla.edu

‡Department of Computer Science, Duke University, Durham NC 27708. Email: kamesh@cs.duke.edu. Research supported by NSF via a CAREER award and grant CNS-0540347.

§Department of Computer Science, Stanford University CA 94305. Email: plotkin@cs.stanford.edu

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 [36], and constant-factor approximations are known (please refer to [28] for a complete list of citations). 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 [35, 4]. We give the first known approximation for the general version of this problem to optimize both placement of servers and network topology.

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 unrelated, and that there may be any number of parallel edges in the graph. We are given a set of sources (customers). The 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 *distance* (*i.e.*, total *length*) from source to sink. Note that this 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 [24, 5] 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.

Theoretical Significance. In Section 5, we show that many standard problems in network design can be reduced to Cost-Distance, showing that this problem forms a general theoretical framework for network design. In particular, we describe simple reductions from single sink buy-at-bulk and the metric facility location problem. 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-sink 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. This problem was subsequently called "Integrated Logistics" by Ravi and Sinha [34]. Our algorithm provides the first known approximation for this more general problem. We also obtain among the first known combinatorial approximations for the metric multi-level uncapacitated facility location problem [23], the priority Steiner tree problem [11], and for constructing single source multicast trees with good cost and average per receiver delay [33].

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 [4, 3, 35] required that the concave functions between each pair of nodes be identical up to a constant scaling factor; we eliminate this requirement. We present all these connections in more detail in Section 5.

Technical Contributions: The chief technical contribution is to show that the Cost-Distance problem can be solved by layered aggregation of demands. We perform aggregation in pairs by

constructing matchings iteratively on a suitably defined complete graph. Our algorithm and analysis exploit the connection of Cost-Distance to the budgeted version of the problem [29, 26], where the goal is 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 $O(\log |S|, \log |S|)$ bicriteria approximation on the cost and diameter via layered aggregation by Marathe *et al* [29]. We need to make one non-trivial modification when the objective becomes aggregate distance instead of diameter – this is discussed in Section 3.1, and makes our algorithm randomized.

The technical framework of layered aggregation pervades all subsequent combinatorial algorithms for single sink aggregation problems, such as Access Network Design [19], single sink buy-at-bulk [20], and the rent-or-buy problem [21].

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-sink buy-at-bulk depended on methods of randomly selecting trees which approximate stretch [6, 7, 9, 10, 16]. The algorithm for Access Network Design [3] depended on a linear programming relaxation.

Previous Results: If the cost and distance metrics are proportional, the offline version of Cost-Distance has a constant factor approximation [5, 24], and there is an online algorithm performing a small number of rerouting of existing paths [17]. If the cost and distance metrics are unrelated, this problem has no previously known approximation algorithm. Previous results for many 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. Details of the algorithm are as follows:

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 distance function on the edges: $M_{uv}(e) = c(e) + \frac{2w_i(u)w_i(v)}{w_i(u)+w_i(v)} l(e)$
   - Let $K_i(u, v)$ be the length of this path under the distance function $M_{uv}(e)$.
4. For every non-sink node $u \in S_i$:
   - Find the shortest $u - t$ path in G according to the metric $M_{ut}(e) = c(e) + w_i(u)l(e)$
   - Let $K_i(u, t)$ be the length of this path under metric $M_{ut}(e)$.
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.

3.1 Discussion

The overall algorithm is very similar to that in [29], where the bicriteria (cost,diameter) version is solved. There is however one very important and non-trivial difference. Every time there is an aggregation, a choice has to be made about the "center" for the aggregate demand. In the case of [29], any center works since they are approximating the diameter of the tree. In our case, such a choice would be disastrous, and we instead resort to choosing the center at random with probability proportional to the demand aggregated there (Step 8). This method has been subsequently derandomized by Chekuri, Khanna, and Naor [13] using the dual of a natural linear programming formulation.

The metric used for constructing the matching is $M_{uv}(e) = c(e) + \frac{2w_i(u)w_i(v)}{w_i(u)+w_i(v)}l(e)$. This is the expected cost of transporting one demand of the pair (u, v) to the other location, when the choice of which location to transport to made at random in proportion to its demand value.

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 [32]. 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

[1]We could use the $O(n^2 \log n)$ approximation algorithm in [18] to get $\beta = 2$ and $\alpha = 2$. Here, n is the total number of nodes in the graph.

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: We will prove this 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. This lemma previously appeared in [29].

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 weight 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. ■

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

Proof: Consider the tree T^*. Using Lemma 4.2, there exists a matching of the nodes in S_i using only edges in T^*, such that no edge is used more than once. 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 $O(\log |S|)$.

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 minimized. This problem is equivalent to Cost-Distance if the distance metric are the delays. This problem has been extensively studied in the networks community [8, 15, 22, 25, 27, 33, 39], and many heuristics have been proposed. We present the first approximation for this problem. Our approximation ratio is $O(\log |R|)$.

5.2 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, which is seen as follows. Suppose $d > 0$ amount of demand is routed to a facility. Then, the cost paid by the COST-DISTANCE version is $f_i(1 + \frac{d}{u_i})$, whereas the cost paid by the actual facility location version is $f_i\lceil\frac{d}{u_i}\rceil$. Clearly $(1 + \frac{d}{u_i}) \leq 2\lceil\frac{d}{u_i}\rceil$

We have therefore obtained a $O(\log(|D|))$ approximation to these problems. Note that several constant factor approximations are known for this problem. The best known approximation ratio is 1.52 due to Mahdian, Ye, and Zhang [28], which builds on a long line of previous work. Please refer to this paper for previous work on this problem.

5.3 Extended Single Sink Buy-at-bulk

In this problem [35], 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 [4, 35] 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. This is seen as follows. Suppose that in the original problem, $d > 0$ amount of demand was routed on edge e using cables of type i, so that the cost paid is $l(e)c_i\lceil\frac{d}{u_i}\rceil$. In the COST-DISTANCE version, the cost paid for routing the same demand is $l(e)c_i + l(e)\frac{c_i}{u_i}d = l(e)c_i\left(1 + \frac{d}{u_i}\right)$. This is clearly at most a factor of 2 away from $l(e)c_i\lceil\frac{d}{u_i}\rceil$ provided $d > 0$.

The best previously known approximation for this problem was $O(\log|S|\log\log|S|)$ which follows by applying the techniques in [10] to the algorithm in [4]. These algorithms are based on the work in [6, 7, 9, 10] 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 [35] showed a constant factor approximation by using previous results [5, 24] on balancing Steiner trees with shortest path trees.

Subsequent to our work, several constant factor approximations have been obtained for this problem, the first one due to Guha, Meyerson, and Munagala [20], and the most recent (and best) due to Gupta, Kumar, and Roughgarden [21].

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 generalizations.

5.4 Combining 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. Ravi and Sinha [34] subsequently gave improved approximations for this problem and called it "Integrated Logistics".

5.5 Priority Steiner Trees

This problem [11] generalizes the Steiner tree problem in the following way. Each edge e has a priority p_e in addition to cost $c(e)$. Each terminal t also has a priority p_t. The goal is to connect these terminals to a sink using a Steiner tree such that all edges on the path from a terminal t to the sink have priority at most p_t. The goal is to minimize the total cost of the edges used. Charikar, Naor, and Schieber [11] present a $O(\log |R|)$ approximation, where R is the set of terminals.

This problem is a special case of Cost-Distance, as shown by a reduction due to Chuzhoy *et al* [14] that we outline below. The algorithm presented in Section 3 then yields a $O(\log |R|)$ approximation, matching the best known guarantee.

Specifically, a priority Steiner tree instance can be converted in an approximation preserving fashion to an instance of the extended single-sink buy-at-bulk problem defined in Section 5.3. Let $C = \max_e c(e)$, and assume $\min_e c_e \geq 1$ by scaling. Suppose there are k priority levels. For each node v with priority i, we set its demand $d_v = (nC)^{5(k-i)}$, and for each edge e with priority i, the capacity of the corresponding cable-type is $u_e = (nC)^{5(k-i)+2}$, and its cost per unit cable is $c(e)$. The length of the edge $l(e) = 1$. Given a solution to the priority Steiner tree, suppose edge e is used. Then, purchase one cable along this edge. Suppose this edge has priority i, then the total demand with priority at least i is at most $n \sum_{l \geq i} (nC)^{5(k-l)} \leq (nC)^{5(k-i)+2} = u_e$. This shows that the purchases yield a feasible solution. This yields a solution to the buy-at-bulk problem with the same cost as the priority Steiner tree. Conversely, given a solution to the buy-at-bulk problem, a demand with priority i will only be routed along edges of priority at most i; otherwise, at least $(nC)^3$ cables of higher priority need to be purchased along the violated edge, which costs at least $(nC)^3$. However, the total cost over all demands of purchasing cables along paths that satisfy the priorities is at most $n^2 C$, which is only cheaper. Therefore, in the optimal buy-at-bulk solution, all demands with priority i are routed on paths such that each edge in the path has priority at most

i; furthermore, only one copy of any edge is purchased. These set of edges are feasible for priority Steiner tree. A solution to the buy-at-bulk instance can be transformed to obtain a solution for the priority Steiner Tree instance, thus completing the reduction.

5.6 Multi-level 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 [23, 37, 38, 2, 36]. We note that Shmoys, Tardos and Aardal [36] present a 4 approximation for the case the number of levels k is 2, and Aardal, Chudak, and Shmoys [1] subsequently present a 3 approximation for general k. Both these algorithms are based on solving linear program relaxations and rounding the resulting solution.

We present the first known combinatorial approximation for general values of k. Our approximation ratio is $O(\log |D|)$. 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 *The* Cost-Distance *algorithm yields 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.*

5.7 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. Suppose further that this concave function for an edge e is of the form $f_e(d) = \min_{i=1}^{r_e}(a_{ie} + b_{ie}d)$. We can convert this problem to an instance of Cost-Distance as follows: For edge e, replace it by r_e parallel edges $e_1, e_2, \ldots, e_{r_e}$, where $c(e_i) = a_{ie}$, and $l(e_i) = b_{ie}$. Given d amount of demand, it is clear that in the new instance, the edge with index $\text{argmin}_i(a_{ie} + b_{ie}d)$ will be used, so that the cost of transportation is precisely $\min_i(a_{ie} + b_{ie}d)$. This is precisely the cost of transporting the demand along edge e in the original instance. Therefore, if the cost of each edge is a piece-wise linear monotonically non-decreasing concave function with polynomially many segments, this can be converted in polynomial time to an instance of Cost-Distance, hence leading to a $O(\log |S|)$ approximation.

6 Conclusion

We presented a general framework, Cost-Distance, for studying several network design problems which are unified by the themes of *economies of scale* and *single-sink*. We presented a simple to implement and intuitive randomized algorithm that achieves a $O(\log |S|)$ approximation, where S is the set of terminals. Since the preliminary version [31] appeared, this algorithm has been derandomized [13] using a linear programming formulation. Meyerson [30] presented a randomized online algorithm with polylogarithmic competitive ratio. It was shown recently by Chuzhoy *et al* [14] that this problem is $\Omega(\log \log |S|)$ hard to approximate, ruling out the possibility of a constant factor approximation. For the multi-terminal version of the problem, Chekuri *et al* [12] present a polylogarithmic approximation based on rounding a natural linear programming relaxation. We note that despite all this subsequent work, our algorithm still achieves the best known approximation ratio for the single-terminal Cost-Distance problem.

References

[1] K. Aardal, F. Chudak, and D. B. Shmoys. A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Inf. Process. Lett.*, 72(5-6):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. Andrews and L. Zhang. The access network design problem. *39th IEEE Symposium on Foundations of Computer Science*, pages 40–49, 1998.

[4] 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.

[5] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, pages 177–87, 1990.

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

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

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

[9] 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.

[10] 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.

[11] M. Charikar, J. Naor, and B. Schieber. Resource optimization in qos multicast routing of real-time multimedia. In *INFOCOM*, pages 1518–1527, 2000.

[12] C. Chekuri, M-T. Hajiaghayi, G. Kortsarz, and M. Salavatipour. Approximation algorithms for non-uniform buy-at-bulk network design problems. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, 2006.

[13] C. Chekuri, S. Khanna, and J. Naor. A deterministic algorithm for the cost-distance problem. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 232–233, 2001.

[14] J. Chuzhoy, A. Gupta, J. Naor, and A. Sinha. On the approximability of network design problems. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, 2005.

[15] M. Doar and I. Leslie. How bad is naive multicast routing. *IEEE INFOCOM*, pages 82–89, 1992.

[16] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455, 2003.

[17] A. Goel and K. Munagala. Extending greedy multicast routing to delay sensitive applications. *Algorithmica*, 33(3), 2002.

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

[19] S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. *Proceedings of 41st IEEE FOCS*, 2000.

[20] S. Guha, A. Meyerson, and K. Munagala. A constant factor approximation for the single sink edge installation problems. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 383–388, 2001.

[21] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 365–372, 2003.

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

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

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

[25] 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.

[26] G. Kortsarz and D. Peleg. Approximating shallow-light trees. *Proceedings of the eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 103–110, 1997.

[27] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15:141–45, 1981.

[28] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 229–242, 2002.

[29] 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.

[30] A. Meyerson. Online algorithms for network design. In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 275–280, 2004.

[31] A. Meyerson, K. Munagala, and S. Plotkin. Cost-distance: Two metric network design. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, 2000.

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

[33] 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.

[34] R. Ravi and A. Sinha. Integrated logistics: Approximation algorithms combining facility location and network design. In *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 212–229, 2002.

[35] 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.

[36] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[37] 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.

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

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