# Network Design for Information Networks

(Extended Abstract)

Ara Hayrapetyan [*†] Chaitanya Swamy[*‡] Éva Tardos[*§]

**Abstract**

We define a new class of network design problems motivated by designing information networks. In our model, the cost of transporting flow for a set of users (or servicing them by a facility) depends on the amount of *information* requested by the set of users. We assume that the aggregation cost follows economies of scale, that is, the incremental cost of a new user is less if the set of users already served is larger. Naturally, information requested by some sets of users might aggregate better than that of others, so our cost is now a function of the actual set of users, not just their total demand.

We provide constant-factor approximation algorithms to two important problems in this general model. In the *Group Facility Location* problem, each user needs information about a resource, and the cost is a linear function of the number of *resources* involved (instead of the number of clients served). The *Dependent Maybecast Problem* extends the Karger-Minkoff maybecast model to probabilities with limited correlation and also contains the 2-stage stochastic optimization problem as a special case. We also give an $O(\ln n)$-approximation algorithm for the *Single Sink Information Network Design* problem.

We show that the *Stochastic Steiner Tree* problem can be approximated by dependent maybecast, and using this we obtain an $O(1)$-approximation algorithm for the $k$-stage stochastic Steiner tree problem for any *fixed* $k$. Our algorithm allows scenarios to have different inflation factors, and works for *any* distribution provided that we can sample the distribution. This is the first approximation algorithm for the multi-stage problem in this general setting.

## 1 Introduction

We define a new class of network design problems, where the cost of transporting flow for a set of users, or servicing them by a facility, is a function of the *set* of users, not just their total demand. In traditional network design each user has a demand, and the cost of transporting flow for a set of users is a function of a single number: the net demand of those users. A set function allows us to express much more complex relations between the costs for different subsets of users. We consider two closely related problems in this framework: single source network design and facility location.

Our network design model captures settings that arise when a distributed set of users needs to send information to (or receive information from) central nodes and/or each other, and the information of different users can be aggregated. For example, in a sensor network application, distributed sensors need to send information to central nodes, and information sent from sensors can often be aggregated well along the paths. Another setting is content-based publish-subscribe distributed database systems [3], where users may "publish" or "subscribe" to information, and information flowing along the network can often be aggregated.

We define our information aggregation model as follows. We are given a graph $G = (V, E)$ with edge lengths $c_e \geq 0$, a set of clients (terminals, demand nodes) $\mathcal{D} \subseteq V$, and a cost function $h : 2^{\mathcal{D}} \mapsto \mathbb{R}^{\geq 0}, h(\emptyset) = 0$. By aggregating the information sent to a set of users $A$, we may be able to send much less than the sum of the information needs of the users in $A$, and thus incur savings. Some users may be more related than others, and hence some subsets may aggregate better than others, that is why the cost is specified by a set function. The function $h(A)$ models both the amount of total information needed by the set $A$, and the cost of sending this information. For the special case when cost depends only on the net demand, $h(.)$ is submodular iff the cost is a concave function of the demand. We will assume that the function $h(.)$ is monotone and submodular, i.e., $h(A) \leq h(B)$ and $h(A + i) - h(A) \geq h(B + i) - h(B)$ for all $A \subseteq B, i \notin B$. Submodularity of $h(.)$ models economies of scale in the cost of aggregation: the added cost of a new user is less if the set of users already served is larger. We assume that $h(.)$ is either given in an implicit way or as an "oracle", and are interested in algorithms whose running time and number of oracle queries is polynomial in other parameters of the problem, such as the size of the graph.

In the *single sink information network design* problem, we have a root $r \in V$ (representing a central authority), and we need to route information from the terminals to $r$, where routing information for a set of clients $A$ along edge $e$ incurs cost $h(A)c_e$. We also consider a variant of this problem with a facility cost component, where we have a set of candidate roots $\mathcal{F}$ (called facilities) instead of a single root $r$. We can route information to any facility, but incur a *facility cost* for routing information to a facility.

Traditional network design problems consider each client as having a demand (packets to send/receive), and the cost of an edge is a function of only on a single parameter — the total demand routed through the edge. For example, in

[*] Dept. of Computer Science, Upson Hall, Cornell University, Ithaca, NY 14853. Email: {ara,swamy,eva}@cs.cornell.edu

[†] Supported in part by NSF grant CCR-011337.

[‡] Supported in part by NSF grant CCR-9912422.

[§] Supported in part by NSF grant CCR-032553, ITR grant 0311333, and ONR grant N00014-98-1-0589.

the Steiner tree problem, the cost of an edge $e$ is a constant $c_e$ for any non-empty set of users, and in buy-at-bulk network design problems [2], it is well approximated by a concave function of the net demand. Using demand as a single measure for defining costs assumes that routing the flow to a set of clients we need to route the total demand of the set.

We examine these network design problems from the perspective of information aggregation. We are interested in the total *information flow* due to a set of users allowing for information aggregation. Information aggregation is an important issue in sensor networks, as sensors have limited battery power and wireless communication is power intensive. Sensor networks are often used to collect data that can be naturally aggregated: we may care about the average temperature in a region, and not about the individual readings of the temperature sensors. The problem of designing sensor networks that can efficiently aggregate information has received significant attention, see, e.g., [5] and its references.

Our model contains an extension of the Karger-Minkoff maybecast problem [12], where the motivation was to design networks under incomplete or uncertain information. In this problem, we are given a graph $G = (V, E)$ with edge costs $c_e \geq 0$, a set of terminals $\mathcal{D} \subseteq V$, and a source $r$. Each terminal $t$ is turned *on* with probability $p_t$ and when it is on, it needs to communicate with the source. To keep communication simple, each terminal $t$ selects a single path $P_t$ from $t$ to $r$ that will be used by it to communicate with the source when it is on. We only pay for edges that actually get used, so if an an edge $e$ lies on the paths of the terminals in set $A$, its cost is $c_e \cdot p(A)$, where $p(A)$ is the probability that some client in $A$ is turned on. Karger & Minkoff [12] define the maybecast problem by assuming that the probabilities $p_t$ are independent, and show that one can then view $p_t$ as the "demand" of $t$. Since the function $p(.)$ is submodular for *any* probability distribution, the general maybecast problem falls into our network design framework.

**Our Results** Our information aggregation model includes many interesting and useful classes of problems, and we believe that it will find applications outside the scope of this work. We provide good approximation algorithms to two important special cases of the problems mentioned above, and for various other combinatorial optimization problems in the general framework where costs depend on the identities of the clients. For the general single sink problem, by arguing as in [2], we obtain an $O\big(\ln |V|\big)$-approximation algorithm by embedding the graph into a tree metric with at most $O\big(\ln |V|\big)$ distortion [4].

Section 2 considers the *group facility location* problem, where we have a set $\mathcal{R}$ of resources and each user $j$ requests information about a specific resource $r(j)$. If $r(A)$ denotes the set of resources requested by users in $A$, then the amount of information corresponding to set $A$ is $|r(A)|$, capturing the fact that the information of clients requesting the same resource can be aggregated, and the cost of sending this information along an edge $e$ is $c_e \cdot |r(A)|$. The goal is to connect users to facilities, which have opening costs, and minimize the total facility opening costs and connection costs. This problem is a natural combination of the uncapacitated facility location (UFL) problem, the special case when users need different resources, and the Steiner tree problem, the special case with one facility and one resource.

We develop a primal-dual 4-approximation algorithm for this problem that is based on integrating the primal-dual algorithms for the UFL and Steiner tree problems. The challenging part is extending the cleanup phase since unlike UFL, in group facility location, it could be the case that two tentatively open facilities get contribution from the same resource, and yet are "far apart". Therefore, a cleanup based on ensuring that a component (of a resource) pays for at most one facility fares badly. To see this, consider the example in Figure 1 with facilities $i$ and $i'$ separated by a long path containing clients that all require the same resource; closing either facility would involve rerouting several clients requiring distinct resources and incurring a huge cost. The approach we develop in Section 2 involves a different cleanup phase, where a single component can contribute towards multiple facilities, and we are able to bound this overpayment and at the same time avoid a high rerouting cost. We extend our result to variants of the problem involving edge capacities, and facility costs which are concave functions of the number of resources served.

Ravi & Sinha [14] and Shmoys, Swamy & Levi [16] were motivated by similar objectives when they considered a setting where each client requests a specific commodity or service. However, they model a different aspect of commodities. They assume that the connection cost is directly proportional to the distance to a facility (and hence is not shared by clients requesting the same commodity), but instead assume that the facility cost depends on the set of commodities served. While this problem is also contained in our general framework, we consider here a different setting where flow is aggregated on edges, not at facilities.

In Section 3, we consider the *dependent maybecast* problem. Recall that in the Karger-Minkoff model [12], the probabilities $p_t$ of terminals being turned on, are independent. In this case, the the edge cost $c_e \cdot p(A)$ is well approximated by $c_e \cdot \min(1, \sum_{t \in A} p_t)$, which is a (concave) function of the total "demand" $\sum_{t \in A} p_t$ routed through the edge.

In our dependent maybecast problem, we consider the following more general model for generating the probabilities $p_t$. Let $\Gamma$ be a *distribution tree* (not related to the graph $G$), whose leaves are the terminals in $\mathcal{D}$. Each edge $e \in \Gamma$ has an associated probability $p_e$. To decide which terminals are on, we "turn on" each edge $e \in \Gamma$ independently with probability $p_e$. A terminal $t$ is on if all edges along the

path from the root to $t$ are on. The Karger-Minkoff model is the special case with a 1-level distribution tree. By allowing more general trees, we allow the terminal probabilities to be correlated. Our main result is a $2(k+1)$-approximation algorithm for dependent maybecast with a $k$-level distribution tree. We obtain this result using the concept of cost-shares as considered by Gupta, Kumar, Pál & Roughgarden [7].

Recently there has been considerable work on approximation algorithms in a different model of network design with uncertain inputs, the *Stochastic Steiner Tree Problem* [8, 10]. In the 2-stage stochastic (rooted) Steiner tree problem we are given a distribution over terminals, a graph $G = (V, E)$ with edge costs $c_e$, and a parameter $\gamma$. Initially (stage I) we may buy some edges based on the above information paying cost $c_e$ for edge $e$, and once a *scenario* $A$ determining the terminals to be connected is revealed, we can buy additional edges (stage II) to form a Steiner tree on the terminals in $A$ paying an increased cost of $\gamma c_e$. More generally, in the multi-stage problem, new information is revealed in each stage, and edges can be added to the solution at each stage, but the cost increases in each stage.

Despite the contrasting aspects of dependent maybecast and the stochastic Steiner tree problem — in one, we have to choose paths completely in advance and pay only for the edges actually used, while in the other, we pay for all the edges bought in stage I and can update the solution as more information is revealed — we will show in Section 4, that the $k$-stage stochastic Steiner tree problem with a polynomial number of scenarios is well approximated by dependent maybecast with a $k$-level distribution tree (up to a factor that depends only on $k$). Using our results on the dependent maybecast problem, we get an $O(1)$-approximation algorithm for the $k$-stage problem for any fixed $k$.In fact, our reduction models the $k$-stage problem even when the increase factor $\gamma$ is scenario-dependent. We further extend our algorithm to distributions with exponentially many scenarios, assuming that we have only black-box access to the scenario distribution. An algorithm for the 2-stage problem with scenario-dependent increase factors was also independently developed by [10]. Gupta et al. [9] also provide an $O(1)$-approximation algorithm for the $k$-stage problem for any fixed $k$ even when the increase factor $\gamma$ is scenario-dependent. Our technique gives the first approximation algorithm in this more general setting with only black-box access, even for the 2-stage problem.

Finally, in Section 5 we give algorithms for some other problems involving set-based cost functions.

## 2 The Group Facility Location Problem

We now consider the *group facility location* (GrpFL) problem and provide a 4-approximation algorithm for this problem. We are given a graph $G = (V, E)$ with costs $c_e \geq 0$ on the edges, a set of facilities $\mathcal{F} \subseteq V$, and a set of clients $\mathcal{D} \subseteq V$. Additionally, we are given a set of commodities or resources $\mathcal{R}$. Each facility $i$ has a *facility opening cost* of $f_i$, and each client $j$ in $\mathcal{D}$ requires a specific resource $r(j) \in \mathcal{R}$. We want to open a set of facilities $A$, and for each resource $r \in \mathcal{R}$, build a Steiner forest $\mathcal{T}_r$ that connects the clients requiring resource $r$ to open facilities, that is, $\mathcal{T}_r$ consists of a collection of Steiner trees, each of which is rooted at an open facility and connects (some of the) clients requiring resource $r$ to $i$. The cost of this solution is $\sum_{i \in A} f_i + \sum_{r \in \mathcal{R}} c(\mathcal{T}_r)$, where $c(\mathcal{T}_r)$ denotes the total cost of forest $\mathcal{T}_r$, i.e., $\sum_{e \in \mathcal{T}_r} c_e$. Our objective is to find a solution of minimum cost. GrpFL can be viewed as a network design problem where we want to connect each client $j$ in $\mathcal{D}$ to an open facility via a path $P_j$, and the cost of using edge $e$ to connect a set $D$ of clients is given by $c_e \cdot |r(D)|$, where $r(D)$ is the set of resources required by clients in $D$.

We can formulate GrpFL as a natural integer program and relax the integrality constraints to get a linear program (LP). We use $i$ to index the facilities in $\mathcal{F}$, $j$ to index the clients in $\mathcal{D}$, and $r$ to index the resources in $\mathcal{R}$. Let $\delta(S)$ denote the set of edges with exactly one end point in $S$. Let $\mathcal{D}_r \subseteq \mathcal{D}$ be the set of clients requiring resource $r$, and let $\mathcal{S}_r = \{S \subseteq V : S \cap \mathcal{D}_r \neq \emptyset\}$. We consider the following LP and its dual.

$$\text{(GrFL-P)} \quad \min \sum_i f_i y_i + \sum_{e,r} c_e x_{e,r}$$

$$\text{(2.1)} \quad \text{s.t.} \sum_{e \in \delta(S)} x_{e,r} + \sum_{i \in S} y_i \geq 1 \quad \text{for all } r, S \in \mathcal{S}_r$$

$$y_i, x_{e,r} \geq 0 \quad \text{for all } i, e, r.$$

$$\text{(GrFL-D)} \quad \max \sum_{r \in \mathcal{R}, S \in \mathcal{S}_r} \theta_{r,S}$$

$$\text{(2.2)} \quad \text{s.t.} \sum_{S \in \mathcal{S}_r : e \in \delta(S)} \theta_{r,S} \leq c_e \quad \text{for all } e, r$$

$$\text{(2.3)} \quad \sum_{r, S \in \mathcal{S}_r : i \in S} \theta_{r,S} \leq f_i \quad \text{for all } i$$

$$\theta_{r,S} \geq 0 \quad \text{for all } r, S \in \mathcal{S}_r.$$

Here $y_i$ indicates if facility $i$ is open, and $x_{e,r}$ indicates if edge $e$ is used to ship commodity $r$. Constraint (2.1) states that every set $S$ containing a client in $\mathcal{D}_r$ must either contain an open facility or have an edge on its boundary that is used to connect the clients in $\mathcal{D}_r \cap S$ to an open facility. Intuitively, the dual problem is the following *moat-packing* problem. We view the sets in $\mathcal{S}_r$ as *moats* around clients in $\mathcal{D}_r$ that need to be "crossed" so that these clients can be connected to open facilities, and $\theta_{r,S}$ as the width of moat $S$. Constraint (2.2) states that the total width of moats of a resource that an edge crosses should be at most the edge cost; constraint (2.3) states that the total width of all moats containing a facility $i$ should be at most $f_i$.

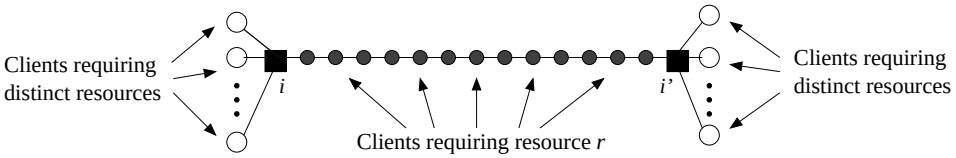


Figure 1: An example on which the natural primal-dual process and cleanup phase produces poor solutions.

**2.1 A Primal-Dual Algorithm** Our algorithm is based on the primal-dual schema. The basic idea is to construct a feasible dual solution, and then use this dual solution to extract a feasible (integer) primal solution. The algorithm is a dual ascent algorithm, so the dual variables are only increased throughout the execution of the algorithm. As mentioned earlier, group facility location generalizes both the uncapacitated facility location (UFL) and the Steiner tree problems. The primal-dual algorithms for these two problems employ quite different approaches for raising the dual variables, and we integrate these two qualitatively different approaches in our primal-dual algorithm. In fact, our algorithm reduces to the Jain-Vazirani (JV) algorithm for UFL, and the algorithm of Agrawal, Klein & Ravi [1] (see also [6]) for the Steiner tree problem in these two special cases.

While there are other problems that also generalize the UFL and Steiner tree problems, ({connected, capacitated cable} facility location), algorithms for these problems [17, 13] proceed by essentially "decoupling" the facility location and Steiner tree aspects. Group facility location does not seem to lend itself to such decoupling. We develop and analyze a natural primal-dual algorithm that combines features of both facility location and Steiner tree in a single process.

At a high level, our algorithm has a fairly simple and intuitive description. We run the Steiner forest algorithm of [1, 6], referred to as the GW algorithm from now, for each resource independently and build a separate forest for each resource. The algorithm maintains the connected components of the forest built so far for each resource, and a set of *tentatively open* facilities. Each resource $r$ component $S$ in the forest for commodity $r$ is associated with a dual variable (or cost contribution) $\theta_{r,S}$. We uniformly increase $\theta_{r,S}$ for each component $S$ that does not contain a tentatively open facility and gradually build a primal feasible solution. When a component of a resource "reaches" a facility, it starts "paying" towards the opening cost of that facility; when the total payment to a facility (from all resources) equals its opening cost, we tentatively open the facility, and freeze (i.e., stop growing) the dual variable $\theta_{r,S}$ for each component $S$ containing the open facility. This process ends when all components are frozen, so each resource $r$ client is in some resource $r$ component that contains an open facility. At this point, the primal solution generated may be quite expensive because of redundant edges and because a component could be contributing to multiple tentatively open facilities. We extract a low cost primal solution by having a cleanup phase where we remove redundant edges and select a subset of tentatively open facilities to open. While getting rid of redundant edges is not a problem, deciding which subset of facilities to open is more involved.

As mentioned earlier, a typical JV style cleanup approach that tries to ensure that a component pays for at most one facility fares badly in our problem. In the example in Figure 1, facilities $i$ and $i'$ are separated by a long path made up of clients that all require the same resource $r$, and numerous clients requiring *distinct* resources are hanging off the ends of the path. The dual ascent process will create a single component for resource $r$ containing all the clients on the path, which then starts contributing toward both facilities. So, although $i$ and $i'$ are far apart, they become *dependent* (in the above sense). But if we were to open only one of $i$ and $i'$, then the clients not in $\mathcal{D}_r$ adjacent to the unopened facility have to be rerouted incurring a huge cost relative to the optimal solution (that opens both facilities).

Thus we need a scheme where a component can pay for multiple facilities, if necessary, to keep the connection costs bounded, and where one can bound the overpayment (to multiple facilities). The crucial observation in the above example is that the contribution of the component to the multiple facilities is small compared to the *length of the path connecting the facilities*, that is, to the cost of the component itself. The cleanup step of our algorithm generalizes this observation. We show that one can afford to open multiple facilities to which a component contributes, provided that the facilities are sufficiently "far apart", by amortizing the net contribution of the component to facilities against the cost of the component. Conversely if the facilities are not far apart then we will be able to bound the rerouting cost.

**The dual ascent process.** This is similar to running the GW algorithm for each resource independently. We maintain a separate forest $\mathcal{T}_r$ for each resource $r$, and we have a set of tentatively open facilities. We say that a connected component $S$ of the forest $\mathcal{T}_r$ is is *active* if it does not contain a tentatively open facility; otherwise it is *frozen*. We have a notion of time, $t$. We start at time $t = 0$ with all variables $\theta_{r,S}$ set to 0, no facility being tentatively open, and each forest $\mathcal{T}_r$ consisting of the isolated clients in $\mathcal{D}_r$. As time increases, for every resource $r \in \mathcal{R}$ and active component $S$ of $\mathcal{T}_r$, we raise the dual variable $\theta_{r,S}$ uniformly at unit rate.

Initially, the active sets are the singletons $\{j\}$ where $j \in \mathcal{D}_r$. The dual variable $\theta_{r,S}$ contributes toward both adding (any number of) edges to $\mathcal{T}_r$, and also toward the facility opening cost of (multiple) facilities in the set $S$. As we increase the dual variables, two types of events may happen:

- For some active component $S$ of $\mathcal{T}_r$ and edge $e \in \delta(S)$, constraint (2.2) holds with equality: we say that $e$ has become *$r$-tight*, add $e$ to $\mathcal{T}_r$ and update the components, possibly merging components. If the new component $S'$ contains a tentatively open facility, then $S'$ is no longer active, and we do not increase $\theta_{r,S'}$ any further.
- Facility $i$ gets paid for, i.e., constraint (2.3) is tight for $i$: we declare $i$ to be *tentatively open*. For every resource $r$, if a component $S$ of the forest $\mathcal{T}_r$ contains $i$, then it becomes frozen and is no longer active.

We continue this process always raising the dual variables $\theta_{r,S}$ for active components of $\mathcal{T}_r$ until no set in $\mathcal{T}_r$ remains active. So at termination, the components of $\mathcal{T}_r$ connect all the clients in $\mathcal{D}_r$ to tentatively open facilities via $r$-tight edges. Let $(\theta)$ be the final dual solution obtained.

**Opening facilities.** Let $F$ be the set of tentatively open facilities. Let $t_i$ be the time at which facility $i \in F$ was declared tentatively open. We say that two facilities $i, i' \in F$ are *dependent* if for some resource $r$, there is an $r$-tight path (i.e., a path of $r$-tight edges) connecting $i$ and $i'$ of length less than $2\min(t_i, t_{i'})$. We pick an arbitrary maximal independent subset $F' \subseteq F$ and open the facilities in $F'$.

**Removing redundant edges.** For every resource $r$, we now remove redundant edges in the forest $\mathcal{T}_r$. Let $T$ be a component in $\mathcal{T}_r$ and let $T' \subseteq T$ be the minimal subgraph that spans all the nodes of $T \cap \mathcal{D}_r$. If $T'$ contains an open facility, we simply delete the edges in $T \setminus T'$. Otherwise, if $T$ contains an open facility, let $i$ be the open facility in $T$ with smallest $t_i$, else let $i$ be the *tentatively open* facility in $T$ with smallest $t_i$. We add the $r$-tight path connecting $i$ and $T'$ to $T'$, and delete the edges in $T \setminus T'$. Let $\mathcal{C}_r$ denote the collection of components for resource $r$ after this step.

**Rerouting components.** At this point, the primal solution may be infeasible due to components in $\mathcal{C}_r$ that do not contain any open facility. For each such component $C \in \mathcal{C}_r$, we simply add edges to $C$ along a shortest path connecting $C$ to the open facility nearest to it.

**2.2 Analysis** The analysis proceeds as follows. In Lemma 2.1 we show that for every resource $r$, the cost of a component $C \in \mathcal{C}_r$ obtained by removing redundant edges in step A3 from component $T \in \mathcal{T}_r$ is at most $2\sum_{S \subseteq T} \theta_{r,S}$; this can be proved by arguing as done in [6] for the Steiner tree problem. Lemma 2.2 shows that the *rerouting cost* in step A4 of a component $C \in \mathcal{C}_r$ with no open facilities is also bounded by $2\sum_{S \subseteq T} \theta_{r,S}$. We then argue that the cost of opening facilities can be charged to the components in $\bigcup_{r \in \mathcal{R}} \mathcal{C}_r$ that contain open facilities, and the net contribution of a component towards opening facilities is at most the cost of the component (Lemma 2.3). Combining the various bounds, we obtain that the total cost of the solution is at most $4\sum_{r, S \in \mathcal{S}_r} \theta_{r,S} \leq 4 \cdot OPT$, where $OPT$ is the cost of an optimal solution.

LEMMA 2.1. *Let $r$ be any resource and $C$ be a component in $\mathcal{C}_r$ obtained from component $T \in \mathcal{T}_r$ in step A3. Then $cost(C) \leq 2\sum_{S \subseteq T} \theta_{r,S}$.*

LEMMA 2.2. *Let $C \in \mathcal{C}_r$ be a component not containing any open facility, obtained from component $T$ in $\mathcal{T}_r$. The cost of adding edges to $C$ in step A4 is at most $2\sum_{S \subseteq T} \theta_{r,S}$.*

*Proof.* By construction, $C$ contains the tentatively open facility $i$ in $T$ with smallest $t_i$ value. Since $i$ is not open, there must be an open facility $i'$ such that $i$ and $i'$ are dependent implying that $c_{ii'} \leq 2\min(t_i, t_{i'}) \leq 2t_i$. We claim that $\sum_{S \subseteq T} \theta_{r,S} \geq t_i$ which will prove the lemma. This follows since at any time $t < t_i$, each client $j \in C \cap \mathcal{D}_r$ must be in some active component $S \subseteq T$ of the forest $\mathcal{T}_r$, otherwise $T$ would contain a tentatively open facility $i''$ such that $t_{i''} < t_i$ contradicting the choice of $i$. So the dual $\sum_{S \subseteq T} \theta_{r,S}$ increases at rate at least 1 at all times $t \in [0, t_i)$ which implies the claim.

For any facility $i$, let $\beta_{i,r} = \sum_{S: i \in S} \theta_{r,S}$ denote the contribution from resource $r$ to the facility cost $f_i$. So if $i$ is tentatively open then $\sum_r \beta_{i,r} = f_i$. Note that if $i \in F'$ lies in component $T \in \mathcal{T}_r$, then $\theta_{r,S} > 0$ for a set $S$ containing $i$ only if $S \subseteq T$, and so $\beta_{i,r} = \sum_{S \subseteq T: i \in S} \theta_{r,S} \leq t_i$ since at most one active component of $\mathcal{T}_r$ may contribute toward facility $i$ at any point of time. For an open facility $i \in T$, let $\sigma(i)$ be the client in $T \cap \mathcal{D}_r$ nearest to $i$.

CLAIM 2.1. *Let $i$ be an open facility in $T$ where $T \in \mathcal{T}_r$. Then $\beta_{i,r} = \tau - c_{i\sigma(i)}$ where $\tau$ is the time at which the active component containing $\sigma(i)$ freezes. Moreover, $\beta_{i,r} > 0 \implies \tau \leq t_i$.*

*Proof.* Let $j = \sigma(i) \in \mathcal{D}_r$ (see Fig. 2a). Let $S_t$ be the active component of $\mathcal{T}_r$ containing $j$ at time $t$. Note that at any time instant $t$, $S_t$ is the *only* component of $\mathcal{T}_r$ that can contain $i$, and hence contribute toward facility $i$. The earliest time $t$ at which $S_t$ may include $i$ is at time $t = t_1 = c_{ij}$, and since there is an $r$-tight path between $i$ and $j$, we have $\tau \geq t_1$. If $t_1 > t_i$ then $i$ is tentatively open at time $t_1$, so component $S_{t_1}$ containing $j$ freezes immediately and we have $\tau = t_1$; since no resource $r$ component contributes toward $i$, $\beta_{i,r} = 0 = \tau - t_1$. Otherwise, component $S_t$ will certainly freeze by time $t = t_i$, so $\tau \leq t_i$, and during the interval $[t_1, \tau]$ the dual variable $\theta_{r,S_t}$ contributes toward facility $i$ at rate 1, so $\beta_{i,r} = \tau - t_1 = \tau - c_{i\sigma(i)}$.

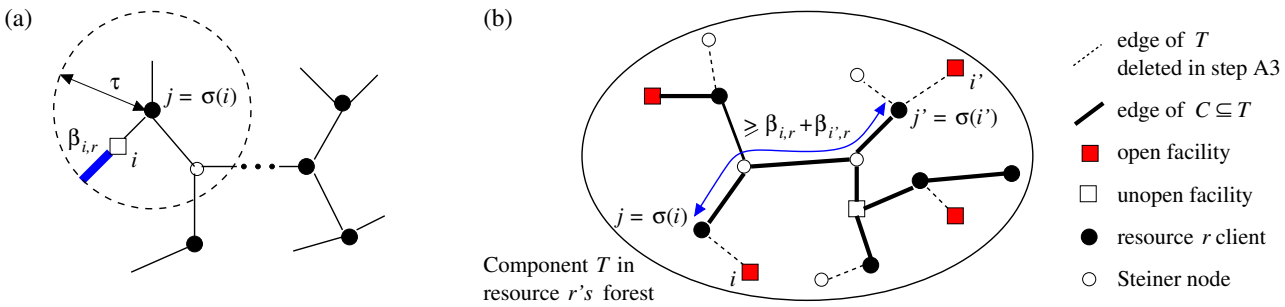


Figure 2: (a) Computing $\beta_{i,r}$ for facility $i$. (b) Bounding the net contribution from $T$ to open facilities.

LEMMA 2.3. *Let $C \in \mathcal{C}_r$ be a component that contains open facilities, obtained by removing edges from $T \in \mathcal{T}_r$. Then, $\sum_{i \in F' \cap T} \beta_{i,r} \leq cost(C)$.*

*Proof.* Let $A \subseteq F' \cap T$ be the open facilities for which $\beta_{i,r} > 0$. We show that the length of the $r$-tight path $\pi$ in $C$ between clients $j = \sigma(i)$ and $j' = \sigma(i')$ for any $i, i' \in A$, is at least $\beta_{i,r} + \beta_{i',r}$ (see Fig. 2b). In particular, this will show that if $i$ and $i'$ are distinct, then $\sigma(i)$ and $\sigma(i')$ are distinct. To see this, let $t_1, t_2$ be the times respectively at which the active components containing $j$ and $j'$ freeze. So $t_1 \leq t_i$ and $t_2 \leq t_{i'}$. If $t_1 = t_2$, the length of the $r$-tight path in $T$ between $i$ and $i'$ is at least $2\min(t_i, t_{i'}) \geq t_1 + t_2$ and at most $c_{ij} + c(\pi) + c_{i'j'}$, so $c(\pi) \geq (t_1 - c_{ij}) + (t_2 - c_{i'j'}) = \beta_{i,r} + \beta_{i',r}$ using Claim 2.1. Otherwise let $t_1 \leq t_2$ without loss of generality. At time $t_1$, $j$ and $j'$ are in different components. Suppose at time $t$, $j$ and $j'$ lie in the same component. Then $t_1 \leq t_2 \leq t$ (the active component containing $j'$ will freeze when it touches the component containing $j$), and $c(\pi) \geq t_1 + t \geq \beta_{i,r} + \beta_{i',r}$. Consider doubling the edges of $C$ and computing an Eulerian tour of $C$. The cost of the tour is at most $2\,cost(C)$. Also, the cost is at least $2\sum_{i \in F' \cap T} \beta_{i,r}$ since the tour can be partitioned into segments $(\sigma(i), \sigma(i'))$ where $i, i' \in A$ each of which has length at least $\beta_{i,r} + \beta_{i',r}$, and every client $\sigma(i), i \in A$ appears as an end point of at least 2 such segments.

THEOREM 2.1. *The above algorithm returns a solution of cost at most $4\sum_{r,S} \theta_{r,S} \leq 4 \cdot OPT$.*

*Proof.* This follows from the previous three lemmas. By Lemma 2.1, $\sum_r cost(\mathcal{C}_r) \leq 2\sum_{r,S} \theta_{r,S}$. Since $\sum_r \beta_{i,r} = f_i$ for every open facility, the facility costs can be charged to the components in $\bigcup_r \mathcal{T}_r$. Consider component $C \in \mathcal{C}_r$ obtained by removing redundant edges from $T \in \mathcal{T}_r$. If $T$ contains open facilities, then its net contribution is at most $cost(C) \leq 2\sum_{S \subseteq T} \theta_{r,S}$ by Lemma 2.3. Otherwise, its contribution is 0, but we need to reroute $C$ paying a cost of at most $2\sum_{S \subseteq T} \theta_{r,S}$ (Lemma 2.2). In either case, this incurs a cost of at most $2\sum_{S \subseteq T} \theta_{r,S}$ per component $T$, and so the total cost is at most $4\sum_{r,S} \theta_{r,S}$.

**Extensions** It is straightforward to extend our results to the case where resource $r$ has a weight $w_r$ and its connection cost is $w_r$ times the cost of its forest. We can also extend the algorithm to handle more general connection costs and facility costs such as (1) an edge capacitated version where the cost of an edge $e$ for resource $r$ is given by $\rho_r \lceil \frac{n(e,r)}{u_r} \rceil c_e$ where $n(e,r)$ is the number of resource $r$ clients using edge $e$ — so one copy of edge $e$ can transport $u_r$ clients of $\mathcal{D}_r$ at a cost of $\rho_r c_e$, and (2) the problem with concave facility costs, when the cost of facility $i$ is $f_i(n(i))$, where $n(i)$ is the number of resources that use $i$ and $f_i(.)$ is a concave function. We get a 5.52-approximation algorithm for (1) by decomposing the problem into a GrpFL instance where resource $r$ has weight $\rho_r$ and a UFL instance where clients of resource $r$ have *demand* $\frac{\rho_r}{u_r}$, solving these two problems separately, and then combining the two solutions without increasing the total cost. We solve (2) by first looking at linear cost functions $f_i + \mu_i n(i)$ and reducing this to GrpFL. A concave function is the lower envelop of a set of linear functions, so we can reduce the problem with concave costs to the linear case which in turn reduces to GrpFL. Thus we get a 4-approximation algorithm.

## 3 The Dependent Maybecast Problem

In this section we consider a class of network design problems involving monotone, submodular cost functions arising from a model that we call *dependent maybecast*. The maybecast problem was introduced by Karger & Minkoff [12] to model the Steiner tree problem with incomplete information. They consider a setting where each terminal requests service from a root node independently with certain probability.

Before formally defining the dependent maybecast problem we define our class of probability distributions that we will use to generate the requests. Let $\mathcal{D}$ be the set of terminals that can request service, and $\Gamma$ be a rooted tree with root $\sigma$ whose leaves are the terminals $\mathcal{D}$ (this tree is distinct from the graph $G$). Each edge $e$ of $\Gamma$ is marked with a probability $p_e \in [0, 1]$ (see Fig. 3a). The stochastic process associated with this model is as follows. Each edge $e$ is turned on

independently with probability $p_e$; the terminals that need to be serviced are those that are reachable via the on edges from the root $\sigma$ of $\Gamma$. We call these the *active* terminals, and refer to probability distributions of active terminals generated by this process as *tree-based distributions*. We use $\Gamma_\rho$ to denote the subtree of $\Gamma$ rooted at node $\rho$.

The *dependent maybecast* problem is defined as follows. We have a graph $G = (V, E)$ with edge costs $c_e$, a root $r$, and a set of terminals $\mathcal{D} \subseteq V$. We are also given a *distribution tree* $\Gamma$ on the terminal set $\mathcal{D}$. Without loss of generality we may assume that the graph is complete and the edge lengths satisfy the triangle inequality. We want to select a path $P_t$ for each terminal $t$ connecting $t$ to the root $r$. The cost of this solution (the set of paths $\bigcup_{t\in\mathcal{D}} P_t$) is the expected cost, evaluated using the distribution generated by $\Gamma$, of the edges used to connect the active terminals to the root (using path $P_t$ for terminal $t$), i.e., $\sum_{e\in E} c_e p(A_e)$ where $A_e$ is the set of terminals whose paths contain edge $e$, and $p(A_e)$ is the probability that *at least one* terminal in $A_e$ is active. The goal is to select a set of paths that minimizes this cost.

Let $k$ be the number of levels in $\Gamma$ (starting at level 0). We give a $2(k+1)$-approximation algorithm for this problem. We use sampling from the given tree distribution as our main design tool. The analysis uses cost-shares in a manner similar to that used by Gupta et al. [7] for the multicommodity rent-or-buy problem.

**3.1 The Algorithm** The algorithm proceeds in stages. In stage 0 we sample from the distribution generated by $\Gamma$. Let $\mathcal{D}_\sigma$ be the set of active terminals after this sampling. We build a minimum cost spanning tree (MST) $T_\sigma$ spanning the set $\mathcal{D}_\sigma \cup \{r\}$, and use the unique tree paths to define the paths $P_t$ for the terminals in $\mathcal{D}_\sigma$.

In general, at a stage $i$, we consider the set of nodes of $\Gamma$ at level $i$, denoted by $\mathsf{level}(i)$. For such a node $\rho$, let $\rho_0 = \sigma, \rho_1, \ldots, \rho_{i-1}, \rho_i = \rho$ be the nodes in $\Gamma$ on the path from $\sigma$ to $\rho$. Let $\Gamma_\rho$ denote the subtree of $\Gamma$ rooted at $\rho$. We sample from the distribution generated by the $\Gamma_\rho$ and obtain a set of active terminals $\mathcal{D}_\rho$. We build an MST $T_\rho$ connecting the terminals in $\mathcal{D}_\rho$ to the root in the graph $G'$ where the trees $T_{\rho_0}, \ldots, T_{\rho_{i-1}}$, built in previous stages corresponding to the ancestors of $\rho$, are contracted (note that $r \in T_{\rho_0}$). (The root of $G'$, also denoted by $r$, is the node that contains the root of $G$.) Note that $\mathcal{D}_\rho$ may contain terminals that were sampled in previous stages, that is, lie in $\mathcal{D}_{\rho_k}$ for some $k < i$, and are thus co-located with the root of $G'$. The trees $T_{\rho_0}, \ldots, T_{\rho_{i-1}}$ together with $T_\rho$ together form a Steiner tree (in graph $G$) on $\mathcal{D}_\rho \cup r$, and we use the unique paths in this tree to define the paths for the terminals in $\mathcal{D}_\rho$.

**3.2 Analysis** The analysis is in two parts. Let $OPT$ denote the cost of an optimal solution. First, we bound the expected cost of tree $T_\sigma$ built in stage 0 by $2 \cdot OPT$. Then we show that, for any stage $i+1$, the expected cost incurred in (building the trees in) stage $i+1$ is no more than that incurred in stage $i$. Since our algorithm has $k+1$ stages this gives a $2(k+1)$-approximation algorithm.

**Bounding the cost of the initial stage.** For any edge selected, the probability of it being used is at most 1, so the cost of stage 0 is at most $stage(0) = \sum_{e\in T_\sigma} c_e = cost(T_\sigma)$. Consider an optimal solution. Let $A \subseteq \mathcal{D}$ be a subset of terminals. Let $q_A$ be the probability that exactly this set of terminals is selected by the stochastic process, and let $c_A$ be the cost incurred by the optimal solution for set $A$. So, $OPT = \sum_{A\subseteq\mathcal{D}} q_A c_A$. The probability that our sampling results in set $A$ is exactly $q_A$. The paths used by the optimum solution to connect the terminals in $A$ include a Steiner tree on $A$. Since the cost of an MST is within a factor of 2 of the minimum cost Steiner tree, the cost we incur to build an MST for sample set $A$ is at most $2c_A$. Hence our expected cost is at most $\sum_A 2q_A c_{qA} = 2 \cdot OPT$.

**Cost-sharing.** We now introduce the notion of cost-sharing that we will use to bound the costs of later stages. A *cost-sharing method* in our framework is a function $\xi : G \times 2^{\mathcal{D}} \times \mathcal{D} \mapsto \mathbb{R}^{\geq 0}$. Intuitively $\xi(G, A, t)$, for $t \in A$, is node $t$'s share in the cost of building a tree on $A$ in graph $G$. Our cost-shares share the cost of the MSTs that the algorithm builds. Fix an MST on $A \cup \{r\}$. Define $\xi(G, A, t)$ to be the cost of the edge connecting $t$ to its parent. Clearly $\sum_{t\in A} \xi(G, A, t)$ is the cost of the MST. We set $\xi(G, A, t) = 0$ if $t \notin A$ for convenience. In later iterations of the algorithm, we select an MST in a graph where a subset of nodes $H$ is contracted. Let $G/H$ denote this contracted graph.

LEMMA 3.1. *For any sets $H \ni r$, $A \subseteq \mathcal{D}$, and $H' \subseteq H$, we have* $\sum_{t\in A} \xi(G/H, A, t) \leq \sum_{t\in A} \xi(G, A \cup H', t)$.

*Proof.* The left side is the cost of the MST on the set $A \cup \{r\}$ in graph $G/H$. To see the inequality, note that the right hand side sums up the cost of a set of edges that form a spanning tree on $A \cup \{r\}$ in $G/H$.

**Bounding the cost of subsequent stages.** Consider a node $\rho \in \mathsf{level}(i)$. Let $q_\rho$ be the product of the $p_e$s for edges along the path from $\rho$ to $\sigma$. In stage $i$, we sample a set $\mathcal{D}_\rho$ from the subtree $\Gamma_\rho$ and build an MST $T_\rho$ (in a contracted graph) connecting the terminals in $\mathcal{D}_\rho$ to the root. An edge $e \in T_\rho$ is used only by (some of) the terminals in $\Gamma_\rho$. So the probability that $e$ will be used is at most $q_\rho$, and the cost incurred for tree $T_\rho$ is at most $q_\rho \cdot cost(T_\rho)$. We define the cost of stage $i$ as $stage(i) = \sum_{\rho\in\mathsf{level}(i)} q_\rho \cdot cost(T_\rho)$.

The total cost of the solution is at most $\sum_{i=0}^{k} stage(i)$. We will prove that for any stage $i, 0 \leq i < k$, we have $\mathrm{E}\left[stage(i+1)\right] \leq \mathrm{E}\left[stage(i)\right]$. Combined with the fact that

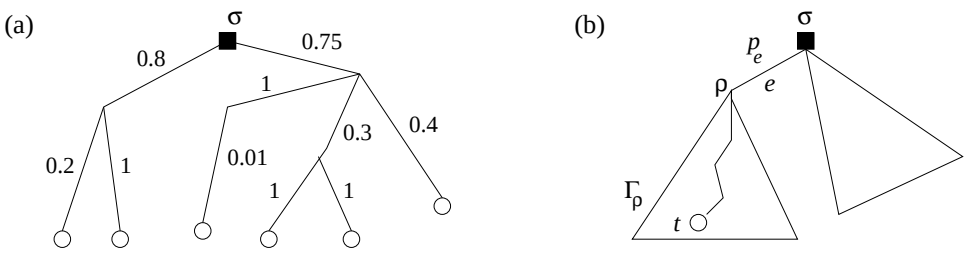


Figure 3: (a) An example of a distribution tree. (b) Bounding the cost of stage 1.

$\mathrm{E}\left[stage(0)\right] \leq 2 \cdot OPT$, this shows that the total expected cost is at most $2(k+1) \cdot OPT$.

LEMMA 3.2. *For any $i, 0 \leq i < k$, we have $\mathrm{E}\left[stage(i+1)\right] \leq \mathrm{E}\left[stage(i)\right]$.*

*Proof.* We show this for $i = 0$. The argument for subsequent stages is similar and is omitted from this extended abstract. Recall that $\mathcal{D}_\sigma$ is the set of terminals sampled in stage 0. Consider a node $\rho$ at level 1 connected to $\sigma$ with edge $e$, and a terminal $t$ in the subtree $\Gamma_\rho$ (see Fig. 3b).

Let us condition on the terminal set $H'$ that is selected in stage 0 from the other branches of the distribution tree $\Gamma$. We say that a terminal $t$ in $\Gamma_\rho$ is "attached" to $\rho$ if all edges on its path to $\rho$ are turned on. Note that in both stage 0 and stage 1, the *same random process* determines the set of terminals that are attached to $\rho$. If set $\mathcal{D}_\rho$ is attached to $\rho$ in stage 0, then its total cost-share is $\sum_{t \in \mathcal{D}_\rho} \xi(G, H' \cup \mathcal{D}_\rho, t)$ if $e$ is on, and 0 otherwise. If $\mathcal{D}_\rho$ is attached to $\rho$ in stage 1, that is, if $\mathcal{D}_\rho$ is the set of active terminals in $\Gamma_\rho$ in stage 1, then its total cost share is $\sum_{t \in \mathcal{D}_\rho} \xi(G/H, \mathcal{D}_\rho, t)$, where $H \supseteq H'$ is the set of terminals selected in stage 0. By Lemma 3.1 we have, $\sum_{t \in \mathcal{D}_\rho} \xi(G/H, \mathcal{D}_\rho, t) \leq \sum_{t \in \mathcal{D}_\rho} \xi(G, H' \cup \mathcal{D}_\rho, t)$. Note that the left term is the cost of the tree $T_\rho$. Multiplying the inequality by $p_e$ and taking the expectation over sets $\mathcal{D}_\rho$ and $H'$, we get that $\mathrm{E}_{H', \mathcal{D}_\rho}\left[p_e \cdot cost(T_\rho)\right] \leq p_e \cdot \mathrm{E}_{H', \mathcal{D}_\rho}\left[\sum_{t \in \mathcal{D}_\rho} \xi(G, H' \cup \mathcal{D}_\rho, t)\right]$. The LHS is simply $\mathrm{E}\left[p_e \cdot cost(T_\rho)\right]$, the stage 1 cost for subtree $\Gamma_\rho$. Note that if edge $e$ is turned on (with probability $p_e$), then $D_\sigma = H' \cup \mathcal{D}_\rho$, so we can rewrite the RHS as $\mathrm{E}\left[\sum_{t \in D_\sigma, t \in \text{ subtree } \Gamma_\rho} \xi(G, \mathcal{D}_\sigma, t)\right]$. So adding the inequality over all level 1 nodes $\rho$ shows that the expected stage 1 cost is at most the expected stage 0 cost.

THEOREM 3.1. *The above algorithm is a $2(k+1)$-approximation algorithm.*

## 4 The Stochastic Steiner Tree Problem

Recently, there has been a lot of interest in approximation algorithms for stochastic network design problems [8, 15, 10]. Both the stochastic Steiner tree problem, and the maybecast problem deal with network design in the face of uncertainty in the input. However, on the surface the two problems are quite different. Stochastic optimization allows for correction of the design after information is revealed (at an increased cost), while in the maybecast problem, the solution is fixed, and we only pay for the edges used. Despite these differences, in this section we will show that the dependent maybecast problem with a $k$-level probability tree can be used to model the $k$-stage stochastic (rooted) Steiner tree problem with a polynomial number of scenarios.

In the 2-stage stochastic Steiner tree problem we have a root $r$, and a distribution over the terminal set $\mathcal{D}$ that determines the terminals to connect to the root. We may buy an edge $e$ in stage I, or in stage II to connect the terminals activated in the scenario that materializes, paying either $c_e$ in stage I, or an increased cost of $\gamma_A c_e$ in scenario $A$. We want to pick edges to buy in stage I so as to minimize the total cost of stage I and the expected stage II cost. Gupta et al. [8] gave a 3.55-approximation algorithm when $\gamma_A = \gamma$ but for an arbitrary distribution. Independent of our work, [10] also gave an algorithm for arbitrary $\gamma_A$s.

In the $k$-stage Steiner tree problem, information about the scenarios is revealed in stages, edges can be purchased in each stage and become more expensive as more information is available. We show that the $k$-stage stochastic Steiner tree problem can be well approximated by dependent maybecast with a $k$-level probability tree, yielding an $O(1)$-approximation algorithm for this problem for any fixed $k$. We extend our result to settings with scenario-dependent inflation factors, and/or more than a polynomial number of scenarios assuming we can sample the scenario distribution.

We first show that the 2-stage stochastic (rooted) Steiner tree problem with a polynomial number of scenarios can be approximated using dependent maybecast with a 2-level probability tree. For each scenario $A$ and node $v \in A$, we create a node $v_A$ co-located with $v$ and make this a terminal in our maybecast instance. This duplication allows a node $v$ to select a separate path to the root in each scenario. Let $\mathcal{T}$ denote this set of terminals. In both problems, we choose paths to connect each terminal — node $v \in A$ in the 2-stage problem, or node $v_A$ in dependent maybecast — to the root, so a solution to one gives a solution to the other.

However, the objective functions of the two problems are different, and furthermore in the 2-stage problem we

distinguish edges bought in stage I and stage II. If edge $e$ is used in the 2-stage problem when a scenario in $\mathcal{A}$ occurs, we incur a cost of $\min\bigl(1, \sum_{A\in\mathcal{A}} p_A\gamma_A\bigr)c_e$: we can buy the edge either in stage I or in every scenario $A \in \mathcal{A}$. To model this via a maybecast problem, we use a distribution tree $\Gamma$ with root $\sigma$ and a level 1 node $\rho_A$ for each scenario $A$, and set the probability of edge $(\sigma, \rho_A)$ to $q_A = \min(1, p_A\gamma_A)$. The children of $\rho_A$ are the terminals $v_A$ for $v \in A$, and edge $e = (\rho_A, v_A)$ has $p_e = 1$. If $\mathcal{A}$ is the scenario set corresponding to the set of terminals using edge $e$, we pay a cost of $c_e \cdot \Pr[\text{edge } e \text{ will be used}]$, that is, $c_e\bigl(1-\prod_{A\in\mathcal{A}}(1-q_A)\bigr)$. As shown in [12], the terms $1 - \prod_{A\in\mathcal{A}}(1 - q_A)$ and $\min\bigl(1, \sum_{A\in\mathcal{A}} q_A\bigr) = \min\bigl(1, \sum_{A\in\mathcal{A}} p_A\gamma_A\bigr)$ are within a constant factor of each other, so we get a constant-factor approximation algorithm for the 2-stage problem.

We now give an algorithm for the 2-stage problem with an arbitrary scenario distribution, using only a black box to sample from the distribution, and an oracle that reveals $\gamma_A$ given a scenario $A$. Let $\gamma = \max_A \gamma_A$, which we assume is known. Whereas in the dependent maybecast instance each scenario $A$ is sampled independently, one can argue, by comparing directly our cost with the cost of the optimal solution for the 2-stage problem using the cost-sharing scheme in Section 3.2, that the following sampling procedure suffices: draw $\gamma$ independent samples and whenever scenario $A$ is sampled, keep it with probability $\gamma_A/\gamma$. As before, we build an MST on the terminals contained in the chosen scenarios and buy the edges of this tree in the first stage. This gives the first approximation algorithm for the 2-stage Steiner tree problem in the black-box model with scenario-dependent inflation factors.

THEOREM 4.1. *There is a 4-approximation algorithm for the 2-stage Steiner tree problem in the black-box model*[1] *and with scenario-dependent costs.*

The above arguments can be generalized to handle the multi-stage stochastic Steiner tree problem. In the $k$-stage problem, the uncertainty of terminals to be connected to the root evolves over $k$ stages and the scenario distribution is specified by a $(k-1)$-level tree, referred to as the *scenario tree*. Each node at level $i-1$ represents an outcome in stage $i$ and corresponds to a particular evolution of the uncertainty through stages $1, \ldots, i$; at each leaf node $\ell$, the uncertainty has completely resolved itself and we know the set of terminals $A_\ell$ to connect to the root. We call $A_\ell$ a *leaf-outcome*. At each stage, we have the option of purchasing edges, but the cost increases through the stages as we get more information. We consider the setting where the inflation factor is identical for all edges in any outcome $\mu$, denoting this by $\gamma_\mu$ ($\gamma_{\text{root}=1}$). Let $p_\mu$ be the probability that $\mu$ occurs and $\lambda_\mu = \frac{\gamma_\mu}{\gamma_\nu}$ where $\nu$ is the parent of $\mu$. So if we buy edge $e$ in outcome $\mu$, the expected cost incurred is $p_\mu \cdot \gamma_\mu$.

Analogous to the 2-stage case, we can model the $k$-stage problem by dependent maybecast by viewing each node $v$ in a leaf-outcome $A_\ell$, as a distinct terminal $v_{A_\ell}$ co-located with $v$ in the maybecast instance. The distribution tree now has $k$ levels, and is the scenario tree appended with leaves that are the $v_{A_\ell}$ nodes, each attached to its corresponding level $k-1$ nodes $\ell$ with an edge with label 1. An edge entering a non-leaf node $\mu \in \mathsf{level}(i-1)$ from node $\nu \in \mathsf{level}(i-2)$ is given a label that captures the expected increase in cost incurred by buying an edge in outcome $\mu$ in stage $i$, over buying the edge in outcome $\nu$ in stage $i-1$, or more precisely, $\min\bigl(1, \frac{p_\mu\lambda_\mu}{p_\nu}\bigr)$. One can show that for any edge $e$ used when a leaf-outcome in $\mathcal{A}$ occurs, the costs incurred in the $k$-stage problem, and in the dependent maybecast instance to route terminals $v_{A_\ell}$ where $A_\ell \in \mathcal{A}$, are within a constant $c_k$ of each other where $c_k$ depends only on the number of stages $k$. Thus we get a $O(k \cdot c_k)$-approximation algorithm for the $k$-stage problem. As in the 2-stage problem, one can specify the first-stage decisions given only the value $\gamma = \max_\mu \lambda_\mu$ and *black-box access* to the scenario distribution such that for any outcome $\mu$, we can sample leaf-outcomes conditioned on the event that outcome $\mu$ occurs. We first sample $\gamma$ times from the entire distribution, and for each sampled *level 1 outcome* $\mu_1$, keep it with probability $\frac{\lambda_{\mu_1}}{\gamma}$. Next for each kept outcome $\mu_1$ we sample $\gamma$ times from the conditional distribution on the leaf-outcomes in its subtree and keep each level 2 outcome $\mu_2$ (child of $\mu_1$) with probability $\frac{\lambda_{\mu_2}}{\gamma}$. Proceeding this way, we output a list of leaf-outcomes, and we buy an MST on the terminals of these leaf-outcomes in stage I.

THEOREM 4.2. *The above algorithm achieves an $O(k)$ approximation ratio for the $k$-level Steiner tree problem in the black-box model with outcome-dependent inflation factors.*

## 5 Other Problems

**Set cover and vertex cover.** In the set cover problem, we are given a universe $U$ of elements $e_1, \ldots, e_n$ and a collection of subsets $S_1, \ldots, S_m \subseteq U$. We want to choose a collection of these sets so that every element is included in some chosen set. Typically, set $S_i$ has an associated cost $c_i$, and the goal is to choose a minimum cost collection. We consider a setting where the cost of $S_i$ is given by a *monotone submodular function* $h_i : 2^{S_i} \mapsto \mathbb{R}^{\geq 0}, h_i(\emptyset) = 0$ with $h_i(T)$ specifying the cost of using set $S_i$ to cover set $T \subseteq S_i$ of elements. The goal is to assign each element to a set containing it so as to minimize $\sum_i h_i(T_i)$ where $T_i \subseteq S_i$ is the set of elements assigned to $S_i$. This problem can be used to model a probabilistic set cover problem, where each element is activated with certain probability, and the goal is to assign each element to a set containing it so as to minimize

[1] The factor is actually $2 + \rho_{ST}$, if we use a $\rho_{ST}$-approximation algorithm to construct a *Steiner tree* on the terminals of the sampled scenarios, and contract only these terminals to the root.

the expected cost of the sets assigned to the active elements. If $S_i$ is assigned a set $T_i \subseteq S_i$ of elements, then its cost is $h_i(T_i) = c_i \cdot \Pr[\exists \text{ active element } e_j \in T_i]$, which is a monotone submodular function.

We obtain a $\ln n$-approximation algorithm for this problem by creating a set $(S_i, T)$ of cost $h_i(T)$ for every subset $T \subseteq S_i$ and running the greedy set cover algorithmusing submodular function minimization to find the next best set. If the algorithm picks sets $(S_i, T_1), \ldots (S_i, T_k)$, then picking $(S_i, \bigcup_i T_i)$ instead is no worse, since $h_i(.)$ is submodular.

In vertex cover we want to cover edges of a graph by its vertices, and the cost for covering a set of edges is given by a monotone submodular function. One can extend the existing primal-dual algorithm for vertex cover to obtain a 2-approximation algorithm for this problem.

**The prize collecting Steiner tree problem.** In the (rooted) prize collecting Steiner tree (PCST) problem, given a graph $G = (V, E)$ with edge costs $c_e$, a penalty function $h : 2^V \mapsto \mathbb{R}^{\geq 0}$ and a root $r$, we want to connect a subset of nodes $S$ to the root by a tree $T$ so as to minimize $\sum_{e \in T} c_e + h(V \setminus S)$. We consider the case where the penalty function is a monotone submodular function. For example, the *reward* of connecting a set $S$ to the root could be proportional to its "sphere of influence" giving rise to supermodular reward functions, e.g., $\mathsf{reward}(S) = \sum_{u,v \in S} r_{uv}$. Equating penalty with the reward foregone, we get a submodular penalty function. We can show that the primal-dual algorithm of Goemans & Williamson [6] for PCST gives a 2-approximation algorithm for this problem.

**Facility location with penalties.** Here we consider the facility location problem with penalties, where we are given a set of facilities $\mathcal{F}$ and a set of clients $\mathcal{D}$ that need to be assigned to open facilities, and we incur a penalty, specified by a monotone submodular function $h : 2^{\mathcal{D}} \mapsto \mathbb{R}^{\geq 0}$, for not assigning clients. We can write the following LP for this problem: minimize $\sum_i f_i y_i + \sum_{j,i} c_{ij} x_{ij} + \sum_{S \subseteq \mathcal{D}} h(S) z_S$ subject to $\left\{\sum_i x_{ij} + \sum_{S \subseteq \mathcal{D}: j \in S} z_S \geq 1 \quad \forall j; \quad x_{ij} \leq y_i \quad \forall i, j; \quad x_{ij}, y_i, z_S \geq 0 \quad \forall i, j, S\right\}$. Variable $z_S$ indicates if we incur the penalty for set $S$. We can solve this LP in polynomial time since one can give a separation oracle for the dual program. Let $(x, y, z)$ be an optimal solution. We show that one can round this solution to an integer solution losing a factor of at most $(1 + \gamma)$ using an LP-based $\gamma$-approximation algorithm for UFL.

Let $N = \left\{j \in \mathcal{D} : \sum_i x_{ij} \geq \frac{\gamma}{\gamma+1}\right\}$. We incur the penalty for clients in $\mathcal{D} \setminus N$, and assign the clients in $N$ to open facilities by solving a UFL instance with facility set $\mathcal{F}$ and client set $N$ using the $\gamma$-approximation algorithm. Note that $\frac{\gamma+1}{\gamma} \cdot \left(\{x_{ij}\}_{j \in N}, y\right)$ is a feasible fractional solution to this instance. So using the $\gamma$-approximation algorithm, we obtain an *integer solution* to this instance of cost at most $(1+\gamma) \cdot \left(\sum_i f_i y_i + \sum_{j \in N, i} c_{ij} x_{ij}\right)$. For each client in $\mathcal{D} \setminus N$, we have $\sum_{S: j \in S} z_S \geq \frac{1}{\gamma+1}$. Since $h(.)$ is submodular, one can show that $\sum_S h(S) z_S \geq \frac{1}{\gamma+1} \cdot h(\mathcal{D} \setminus N)$. This shows that the overall cost is bounded by $(1 + \gamma) \cdot OPT$.

**Acknowledgment** We thank Martin Pál for suggesting the sampling approach of Section 3.

## References


[1] A. Agrawal, P. Klein, and R. Ravi. When trees collide: an approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Computing*, 24(3):440–456, 1995.

[2] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Proceedings of 38th FOCS*, pages 542–547, 1997.

[3] A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith. Efficient filtering in publish-subscribe systems using binary decision diagrams. In *Proc. 23rd ICSE*, pages 443–452, 2001.

[4] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of 35th STOC*, pages 448–455, 2003.

[5] A. Goel and D. Estrin. Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. In *Proceedings of 14th SODA*, pages 499–505, 2003.

[6] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Computing*, 24:296–317, 1995.

[7] A. Gupta, A. Kumar, M. Pál, and T. Roughgarden. Approximation via cost sharing: a simple approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings of 44th FOCS*, pages 605–615, 2003.

[8] A. Gupta, M. Pál, R. Ravi, & A. Sinha. Boosted sampling: approximation algorithms for stochastic optimization. In *Proceedings of 36th STOC*, pages 417–426, 2004.

[9] A. Gupta, M. Pál, R. Ravi, & A. Sinha. Personal communication. March, 2004.

[10] A. Gupta, R. Ravi, & A. Sinha. An edge in time saves nine: LP rounding approximation algorithms. *FOCS*, 2004.

[11] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and $k$-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

[12] D. R. Karger and M. Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings of 41st FOCS*, pages 613–623, 2000.

[13] R. Ravi and A. Sinha. Integrated logistics : approximation algorithms combining facility location and network design. In *Proceedings of 9th IPCO*, pages 212–229, 2002.

[14] R. Ravi and A. Sinha. Multicommodity facility location. In *Proceedings of 15th SODA*, pages 335–342, 2004.

[15] D. B. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. *FOCS*, 2004.

[16] D. B. Shmoys, C. Swamy, and R. Levi. Facility location with service installation costs. In *Proceedings of 15th SODA*, pages 1081–1090, 2004.

[17] C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.