MATHEMATICS OF OPERATIONS RESEARCH
Vol. 32, No. 2, May 2007, pp. 345–364
ISSN 0364-765X | EISSN 1526-5471 | 07 | 3202 | 0345

DOI 10.1287/moor.1060.0237

LP Rounding Approximation Algorithms for Stochastic Network Design

Anupam Gupta
Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213,
anupamg@cs.cmu.edu, http://www.cs.cmu.edu/~anupamg

R. Ravi
Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213,
ravi@cmu.edu, http://www.tepper.cmu.edu/andrew/ravi

Amitabh Sinha
Ross School of Business, University of Michigan, Ann Arbor, Michigan 48109,
amitabh@umich.edu, http://www.umich.edu/~amitabh

We study the *Steiner tree* problem and the *single-cable single-sink network design* problem under a two-stage stochastic model with recourse and finitely many scenarios. In these models, some edges are purchased in a first stage when only probabilistic information about the second stage is available. In the second stage, one of a finite number of specified scenarios is realized, which results in the set of terminals becoming known and the opportunity to purchase additional edges (under an inflated cost function) to augment the first-stage solution. We provide constant factor approximation algorithms for these problems by rounding the linear relaxation of IP formulations of the problems. Our algorithms involve solving the linear relaxation first, followed by a primal-dual routine that is guided by the LP solution. We also show that because our bounds are *local* (the cost of each component is bounded by its cost in the LP solution), we are able to obtain bounds that guard against a form of downside risk.

Key words: stochastic optimization; approximation algorithm; Steiner tree; network design; LP rounding; primal-dual method
MSC2000 subject classification: Primary: 90C15; secondary: 90C27
OR/MS subject classification: Primary: programming, stochastic; secondary: analysis of algorithms, networks/graphs
History: Received October 8, 2004; revised August 2, 2005 and May 23, 2006.

1. Introduction. Designing networks efficiently often involves facing an inherent dilemma: The actual usage and demand patterns become known only after the network is in place, yet a low-cost network that satisfies the usage requirements must be constructed before this information is known. Real-world instances of such situations abound: Cable companies have to plan their network to serve future population and business needs, oil companies have to build pipelines based on forecasts of supply and demand, local governments have to install electrical and transportation infrastructure to lure industry, etc.

Given the importance of the problem of constructing low-cost networks with uncertain future demands, there are surprisingly few approximation algorithms for network design that consider this problem: Most of the work in the area deals with designing networks where the requirements and costs are completely known up-front. In other words, this body of work maintains the optimistic view that the future will conform exactly to the predictions. At the other extreme is the pessimistic view of online algorithms, where one constructs solutions assuming that the future demands are given by an adversary: i.e., one attempts to minimize the cost in the *worst possible* future scenario. The field of stochastic optimization, in contrast, adopts the (arguably more realistic) view that partial, probabilistic information about the future is often available, and can be used wisely.

Stochastic optimization is a vast field, beginning with the works of Dantzig [5] and Beale [3] in the 1950s, and seeing much activity to this day; the reader is referred to the following texts and articles (Birge and Louveaux [4], Kall and Wallace [18], Klein Haneveld and van der Vlerk [21, 20]) that survey the current state of the field. There are several models in which stochastic optimization problems can be cast, depending on the timeline of events and availability of data. Among the more popular models is that of *two-stage stochastic optimization with recourse*. Informally, in this model, only partial data is available in the first stage, while the complete data is made available in the second stage; the user can make some decisions in the first stage to avoid potentially costly second-stage decisions. In this work, we will work in the *finite-scenario* model: The data in the second-stage is one of a finite set of distinct and specified scenarios. The goal in this model, as in most models of stochastic optimization, is to minimize the *expected* cost of the solution, where the expectation is taken over the probability distribution from which the second-stage scenario is drawn. Our model and problem will be specified in greater detail in §2.

We consider two specific network design problems, both on undirected edge-weighted graphs. Before we define these stochastic problems, let us describe the deterministic (i.e., nonstochastic) versions of these problems. The *Steiner tree problem* seeks to find a network spanning a given subset of the vertex set, the members of which are called the terminals. The cost of a solution network is the sum of the edge weights in this network, and the goal is to minimize this cost. The other problem is the *single-cable single-sink network design* problem (which we refer to as the network design problem for the rest of this paper); it also involves a graph and a set of terminals, but has an additional parameter called the *routing-cost multiplier*. The goal in this problem is to define a path from each terminal to a specified root vertex; the cost of a solution is the sum of the weights of edges lying in the union of these paths, *plus* an additional component that is the product of the routing-cost multiplier and the sum (over paths) of the weights of edges lying in each of the paths. Clearly, this model extends the Steiner tree problem; it also captures the economies of scale incurred in designing transshipment networks, because each edge in the constructed network now has an associated fixed cost (its weight), and a cost that depends on the number of paths using it.

The stochastic versions of these problems considered in this paper are those obtained by viewing these deterministic problems in the context of two-stage stochastic optimization. In stochastic Steiner tree, the set of terminals is now revealed only in the second stage (apart from one terminal, the *root*, which is known in the first stage), while edges purchased in the second stage are costlier by an inflation factor. A *scenario* consists of one possible realization of the second stage, and is specified by a set of terminals, the inflation factor, and the probability of occurrence of this scenario. Recall that we consider the finite-scenario model, where the complete specification of each scenario is known in the first stage. The stochastic network design problem is defined in a similar vein: The second stage consists of a set of terminals, an inflation factor for the routing cost, an inflation factor for the cost of purchasing edges, and the probability of occurrence. We reiterate that all edges have the same two inflation factors in one scenario, but these factors can change across scenarios.

1.1. Approximation algorithms. We use the well-developed framework of *approximation algorithms* (Vazirani [29] is an excellent recent text) to attack these problems in this paper. Formally, an approximation algorithm runs in time polynomial in the size of the input and guarantees that on every input instance, the cost of the solution computed by the algorithm is no worse than a prespecified factor times the cost of the optimum solution. This is in contrast to the work in traditional stochastic optimization literature (as in Birge and Louveaux [4], for example), where the focus is on obtaining exact solutions or close approximations without bounds on the running time.

Both of our problems have been the subject of intense study in the approximation algorithms literature. While it has been known for a while that the minimum spanning tree of the terminal set is a factor-2 approximation for minimum Steiner trees, the current best approximation algorithm is a 1.55-approximation algorithm due to Robins and Zelikovsky [25]. A primal-dual $(2 - 2/k)$-approximation algorithm (where k is the number of terminals) was obtained by Agrawal et al. [1], and the algorithm was generalized to constrained forest problems by Goemans and Williamson [8]. Our algorithm uses this primal-dual algorithm as a subroutine.

The network design problem has also received a lot of attention recently in the world of approximation algorithms, with several variants and generalizations of the problem having been considered (Awerbuch and Azar [2], Hassin et al. [15], Mansour and Peleg [22], Guha et al. [9], Gupta et al. [10]). A constant factor approximation for the version we study was first given in Salman et al. [26], while a similar problem known as the traveling purchaser problem (Ravi and Salman [23]) provides another algorithmic tool that we use in our work. The network design problem simultaneously tries to achieve the objectives of a minimum-cost Steiner tree and a shortest-path tree, and a precise characterization of such a trade-off was provided by Khuller et al. [19]. We use their *Light Approximate Shortest-Path Tree* construction in our algorithm for stochastic network design.

2. Overview.

2.1. Problem definitions. All the problems we consider are on undirected graphs $G = (V, E)$, which are equipped with nonnegative costs (or weights) on edges c: $E \to \mathbb{R}^+$. The term $c(e)$ refers to the cost of edge e, while for a subgraph H, the term $c(H)$ is defined as $c(H) = \sum_{e \in H} c(e)$. There is a designated *root* vertex, which is labeled r. Given a terminal set $R \subseteq V$ such that $r \in R$, let span(R) denote the set of subgraphs of G that span R.

In the stochastic Steiner tree problem, the terminal set R is specified only in the second stage. Moreover, the cost of edges in the second stage is multiplied by a factor of σ_R. Hence, the stochastic Steiner tree problem is to compute E_0 to minimize:

$$c(E_0) + \mathbf{E}[\sigma_R c(E_1)] \quad \text{such that} \quad E_0 \cup E_1 \in \text{span}(R). \tag{1}$$

Recall that this paper deals with the finite-scenario model. That is, there is a set of m scenarios, with the kth scenario specified by a terminal set S_k, a probability of occurrence $p_k \in (0, 1]$, and an inflation factor $\sigma_k > 0$. These scenarios and their probabilities are known in the first stage; the only uncertainty is the identity of which scenario will materialize in the second stage. Because precisely one scenario materializes in the second stage, we must have $\sum_{k=1}^{m} p_k = 1$. In this model, the stochastic Steiner tree problem (1) may be rewritten as:

$$\min c(E_0) + \sum_{k=1}^{m} p_k \sigma_k c(E_k) \quad \text{such that} \quad E_0 \cup E_k \in \text{span}(S_k) \ \ \forall k = 1, 2, \ldots, m. \tag{2}$$

We now define the stochastic network design problem. Along with G, r, and c, the first stage also includes a *building cost multiplier* σ_0 and a *routing cost multiplier* δ_0. The kth scenario is specified by its set of terminals S_k, probability of occurrence p_k, building cost multiplier σ_k, and routing cost multiplier δ_k. The first-stage solution is specified by an edge set E_0, with the understanding that a cost of $\sigma_0 c(e)$ is incurred for each edge $e \in E_0$, which permits us to incur a cost of $\delta_0 c(e)$ for each terminal using edge e in its path to the root. The second-stage solution for scenario k consists of an edge set E_k such that $E_0 \cup E_k \in \text{span}(S_k)$. For an edge $e \in E_k$, we incur a cost of $\sigma_k c(e)$ and pay an additional $\delta_k c(e)$ for each terminal that uses e in its path to the root. We model this by assuming that each terminal ships one unit of flow to the root along a specified path, with $f(e)$ used to denote the flow on edge e. Then, the stochastic network design problem is:

$$\min \sigma_0 c(E_0) + \sum_{k=1}^{m} p_k \left[\sigma_k c(E_k) + \sum_{e \in E_0} \delta_0 f(e) c(e) + \sum_{e \in E_k} \delta_k f(e) c(e) \right] \quad \text{s.t.} \quad E_0 \cup E_k \in \text{span}(S_k) \ \ \forall k. \tag{3}$$

In real-world applications, it is reasonable to assume that a network planner will neither be interested in a first-stage network E_0 that consists of scattered, disjoint fragments, nor in networks containing cycles. Therefore, in our formulation of this problem, we will model that E_0 is a single tree containing the root, via a flow-monotonicity requirement.

2.2. Overview of results and algorithms. Our main results are constant-factor approximation algorithms for stochastic Steiner tree and stochastic network design. Our algorithms rely on rounding linear programming relaxations for the respective problems.

Section 3 is devoted to our algorithm for stochastic Steiner tree. While the first-stage component of an optimal solution need not be a tree in general, we prove that there exists a near-optimal solution of the linear relaxation of stochastic Steiner tree where the flow paths are monotone, as in a tree. We then formulate an integer program (IP) for the problem where the first-stage solution is required to observe this flow monotonicity. We begin by solving the linear relaxation of this IP, and using the fractional solution to set up and run primal-dual subroutines similar to that in Agrawal et al. [1] and Goemans and Williamson [8]. For the first-stage tree, we select a set of terminals that are sufficiently far apart (as in Ravi and Salman [23]), and use the primal-dual algorithm to construct a Steiner tree. The primal-dual method is also used in the construction of the second-stage trees, but with further modifications that are necessary to obtain our performance guarantee.

We study the stochastic network design problem in §4, with the additional requirement that the first-stage solution is monotone. That is, the flow from each terminal to the root transitions at most once from second-stage edges to first-stage edges. Without loss of generality, this ensures that the first-stage solution is in fact a tree. Once again, we formulate an integer program for the problem, and begin by solving its linear relaxation. We construct Steiner trees as in the algorithm for the stochastic Steiner tree, with one modification to handle a special case. Our primary tool to accommodate the changed cost function is the Light Approximate Shortest-Path Tree construction (Khuller et al. [19]). We crucially use this construction to obtain our constant-factor approximation ratio for the routing costs.

An interesting feature of our algorithms is that all the bounds are *local*; that is, each component of the final solution (first-stage building cost, second-stage building cost, routing cost) is bounded using the cost of the corresponding component in the fractional solution of the linear program. This allows us to trade-off the costs of the two stages: Given upper bounds on the costs allowed in various scenarios, we either compute solutions close to the bounds or prove that the bounds make the problem infeasible. Thus, if a user is risk averse and is unwilling to incur a large second-stage cost, they can specify budgets for the second stage and ask for a solution that minimizes the overall expected cost subject to these budgets. Our results for these versions of stochastic Steiner tree and stochastic network design are provided in §5.

2.3. Related work. Approximation algorithms for stochastic optimization problems are a fairly new area of research; let us list a few lines of research that relate in different ways to our paper. The first such work that we are aware of is by Dye et al. [6], who provided an approximation algorithm for a problem where a service provider wished to maximize profits, faced with stochastic demand for its services. Immorlica et al. [17] considered the stochastic Steiner tree problem in a slightly different model: The second stage is obtained by each vertex v becoming a terminal with a prespecified probability p_v independent of all other terminals, and the cost inflation factor being constant in all scenarios ($\sigma_k = \sigma$ for all k). They provided a constant-factor approximation when the graph is an ultrametric, and an $O(\log |V|)$-approximation for general graphs.

An $O(1)$-approximation for stochastic Steiner tree in general metrics and arbitrary distributions that can be sampled efficiently was given by Gupta et al. [12]; while they also required that the cost inflation factor is constant across scenarios, their model and results can be extended to allow scenario-dependent values of the cost inflation factor. Although our approximation ratio for stochastic Steiner tree is weaker than theirs, it forms the basis of the algorithms for stochastic network design. Also, giving risk-bounded approximations currently does not seem possible with the techniques of Gupta et al. [12]. On the other hand, the techniques of that paper can be extended to give approximation algorithms for *multistage* stochastic optimization (as shown in Gupta et al. [13]); it remains open how to extend the results of our work to multiple stages.

The finite-scenario model we consider was also considered by Ravi and Sinha [24] for the stochastic shortest-path problem (which is a special case of the stochastic Steiner tree problem). They also considered a more general model in which the metric changes arbitrarily in each scenario, and provided polylogarithmic approximations and inapproximability results. Hayrapetyan et al. [16], in addition to other results, show how a multistage extension of the stochastic Steiner tree problem with scenario-dependent inflation factors can be approximated within a factor of the order of the number of stages; their algorithm also requires only samples from the second-stage distribution, as in Gupta et al. [12].

Recently, Shmoys and Swamy [27] presented a powerful result enabling the use of sampling for stochastic linear programs, thus providing approximation algorithms for such problems. In conjunction with rounding schemes, they also provide approximation algorithms for a stochastic version of some combinatorial optimization problems, including variants of the facility-location problem and multicommodity flow problems. It is worth noting that their algorithm can be used to obtain a $(1+\epsilon)$-factor approximation to the linear relaxation of a natural cut-covering formulation of the stochastic Steiner tree problem, although it is an open question whether the resultant fractional solution can be rounded to an integer solution with provable performance guarantees.

3. Stochastic Steiner tree. The main result of this section is a constant-factor approximation algorithm for the stochastic Steiner tree problem in the finite-scenario model of two-stage stochastic optimization with recourse. To this end, we first formulate the problem as an integer program, solve the linear programming relaxation, and then round the solution; the rounding step uses a new variant of the primal-dual method in which the process is guided by the solution to the linear programming relaxation.

Recall that the input is an undirected graph $G = (V, E)$ with edge-weights $c(e)$ and a distinguished root vertex r. Because we are working in the finite-scenario model, a set of m scenarios $\{S_1, \ldots, S_m\}$ is also explicitly given. The kth scenario is a set of terminals $S_k \subseteq V$, along with an associated probability of occurrence $p_k = p(S_k)$. (Note that $\sum_k p_k = 1$.) Furthermore, a *scale* or *inflation* factor σ_k is also given, where the cost of buying the edge e in the recourse network costs $\sigma_k c(e)$. A feasible solution is specified by a set of edges E_0 selected in the first stage, and for each scenario k a set of edges E_k to be selected in case scenario k materializes, such that $E_0 \cup E_k$ is a Steiner tree connecting $S_k \cup \{r\}$. The objective is to minimize the expected cost of the solution.

We can extend the well-known undirected cut formulation of the deterministic version of the Steiner tree problem studied in Agrawal et al. [1] and Goemans and Williamson [8] to formulate the stochastic Steiner tree problem as an integer linear program as in (4)–(7) below.

$$\min \sum_{e \in E} c(e) x_e^0 + \sum_{k=1}^{m} p_k \sigma_k \sum_{e \in E} c(e) x_e^k \tag{4}$$

$$\sum_{e \in \delta(S)} (x_e^0 + x_e^k) \geq 1 \quad \forall S\colon r \notin S, \quad S \cap S_k \neq \varnothing, \quad \forall k \tag{5}$$

$$x_e^k \geq 0 \quad \forall k, \quad e \in E \tag{6}$$

$$x_e^k \in \mathbb{Z} \quad \forall k, \quad e \in E. \tag{7}$$

In the integer program above, the variables x^0 and x^k are indicator variables for the sets E_0 and E_k, which in turn can be defined as $E_i = \{e: x_e^i = 1\}$. The set $\delta(S)$ denotes the cut formed by edges with exactly one endpoint in the set S; that is, $\delta(S) = \{e \in E: |e \cap S| = 1\}$. The edges purchased in the first stage incur a cost $\sum_{e\in E} c(e)x_e^0$; furthermore, scenario k occurs with probability p_k, in which case we incur an additional cost of $\sigma_k \sum_{e\in E} c(e)x_e^k$. The objective is to minimize the expected total cost.

Note that the critical part for the algorithm is to compute E_0, the edges to be bought in the first stage; given E_0, computing E_k is equivalent to contracting the edges in E_0 and finding a minimum Steiner tree on $S_k \cup \{r\}$, which can be well approximated. Of course, our choice of E_0 must allow us to prove a guarantee on the expected cost of a good completion in each scenario.

3.1. Tree solutions. As a first step towards obtaining a useful LP relaxation for the problem, let us prove a simple but key structural result. We show in Lemma 3.1 that there exists a *near-optimal* solution where the paths from any terminal to the root are *monotone*; i.e., they consist of an initial portion of recourse edges, followed by a final portion of first-stage edges. In other words, in this near-optimal solution, the first-stage solution must be a (connected) tree containing the root r. Note that the optimal solution itself may not exhibit this property, as we later show in Example 3.1.

The intuition behind the lemma is simply that if the expected cost (sum of probabilities times inflation factors) of a network in the second stage is more than the first-stage cost, then it is better to purchase it in the first stage. This key idea, while very simple, also forms the basis of the arguments in some of the previous work on stochastic optimization (e.g., Immorlica et al. [17], Ravi and Sinha [24]).

LEMMA 3.1. *Let* OPT *be the cost of an optimal integer solution to* (4)–(7)*, specified by* x^**. Then there exists a fractional solution* y *to the linear program* (4)–(6) *such that*:

(i) *The cost of* y *is no more than twice the cost of* x:

$$\sum_{e\in E} c(e)y_e^0 + \sum_{k=1}^{m} p_k\sigma_k \sum_{e\in E} c(e)y_e^k \le 2\left(\sum_{e\in E} c(e)x_e^{0*} + \sum_{k=1}^{m} p_k\sigma_k \sum_{e\in E} c(e)x_e^{k*}\right).$$

(ii) *For every scenario* k *and every terminal* $t \in S_k$*, there exists a set of paths from* t *to* r *with total capacity at least* 1 *(the capacity of an edge* e *being simply* $y_e^0 + y_e^k$*), such that each path consists of an initial second-stage segment followed by a first-stage segment.*

PROOF. Consider the integer solution to (4)–(7) defined by $y_e^0 = x_e^{0*} + \sum_{k=1}^m p_k\sigma_k x_e^{k*}$, and $y_e^k = x_e^{k*}$. On any given edge, the total cost of y is simply $c(e)x_e^{0*} + 2\sum_{k=1}^m p_k\sigma_k c(e)x_e^{k*}$. Summing over all edges proves (i).

Now consider a terminal t in some scenario k. In the solution given by x^*, there is a path from t to r consisting of a sequence of edges, where in each edge either $x_e^{0*} = 1$ or $x_e^{k*} = 1$. Let e' be the first edge (if it exists) with $x_e^{0*} = 1$ in this sequence. If no such edge exists, then the path from t to r in x^* consists of only second-stage edges and also exists in y, so t satisfies (ii). Therefore, for the rest of this proof, assume that e' exists.

By definition, e' belongs to some connected component E' of E^{0*}. If r belongs to the same component E', then the terminal t satisfies (ii), because the path from t to r consists solely of second-stage edges prior to e', and solely of first-stage edges after that. Therefore, for the rest of this proof, we assume that e' belongs to a component E' of E^{0*}, which does not contain r.

Consider some cut C such that $r \notin C$ and $E' \subseteq C$, and for which $\sum_{e\in\delta(C)} x_e^{0*} = 0$. Let $K_{e'}$ be the set of scenarios that use edge e' to connect the terminals in S_k to the root. Because E^{0*} is an optimal solution, we must have $\sum_{k\in K_{e'}} p_k\sigma_k \ge 1$. Each scenario in $K_{e'}$ has a flow path that passes through edge e' and terminates at r, and each such flow path has at least one edge in $\delta(C)$ where $x_e^{k*} = 1$ (because all edges in $\delta(C)$ have $x_e^{0*} = 0$). Summing over the scenarios $K_{e'}$ and the edges in these paths, we have $\sum_{e\in\delta(C)}\sum_k p_k\sigma_k x_e^{k*} \ge 1$. Therefore, by the definition of y we have that $\sum_{e\in\delta(C)} y_e^0 \ge 1$. In other words, there is a set of paths of total capacity of at least one consisting solely of first-stage edges from E' to r. Because all edges in E' have $y_e^0 = 1$, we can extend these paths through e' back to t, and this collection of paths ensures that t satisfies (ii). □

While the above lemma shows the existence of a monotone near-optimal solution, the optimal solution itself may not be monotone. That is, even though the optimal choice for E_0 is always a forest and has no cycles, it is not necessary that E_0 be a single tree, nor that it be connected to the root r. Example 3.1 below shows an instance where this is the case.

EXAMPLE 3.1. Consider a "fan" graph with $l > 1$ paths $\mathcal{P}_i$, each with l edges and originating at the same node v_0; path $\mathcal{P}_i$ is given by $\{v_0, v_{i1}, v_{i2}, \ldots, v_{il}\}$. These paths are referred to as the "spokes" of the fan. Node v_0 is connected to the root r by a single edge. The other endpoints v_{il} of all these spokes are also connected

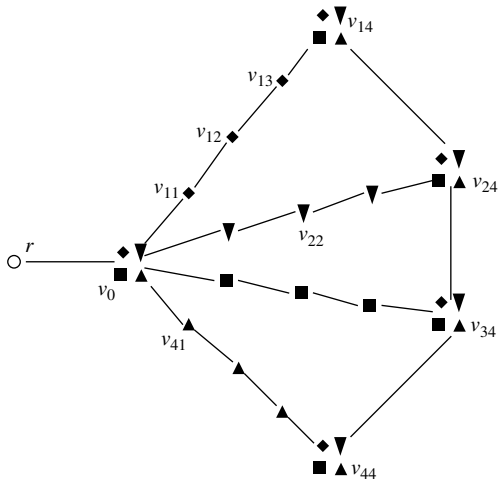

FIGURE 1. Graph for Example 3.1, with $l = 4$.

by a path $\mathcal{P}_r = \{v_{1l}, v_{2l}, \ldots, v_{ll}\}$ called the "rim" of the fan. All edges have unit cost. The stochastic Steiner tree instance consists of l scenarios with $p_i = 1/l$ and $\sigma_i = \sigma \in (1, l)$ for all i; the required set S_i for scenario i consists of all the vertices on the spoke $\mathcal{P}_i$ as well as those on the rim $\mathcal{P}_r$. An example with $l = 4$ is shown in Figure 1.

Observe that $p_i\sigma < 1$, but $\sum_{k=1}^{l} p_i\sigma = \sigma > 1$. Hence, the optimal solution is given by $E_0^* = \{(r, v_0), (v_{1l}, v_{2l}), (v_{2l}, v_{3l}), \ldots, (v_{l-1,l}, v_{ll})\}$ and $E_k^* = \{(v_0, v_{k1}), (v_{k1}, v_{k2}), \ldots, (v_{k,l-1}, v_{kl})\}$. That is, the first-stage component of the optimal solution consists of the edge adjacent to the root and the rim, while the second-stage solution for scenario k consists of the kth spoke. The cost of this solution is $l(1+\sigma)$. Observe that the first-stage solution is a forest with two trees that are not connected to each other.

Now suppose we ask for an optimal solution where *the first-stage component must be a tree*. There are two possible solutions, depending on the values of σ and l. Either $E_0^* = \{(r, v_0)\}$, leading to a net expected cost of $1 + \sigma(2l-1)$, or alternatively, $E_0^* = \{(r, v_0), (v_0, v_{11}), (v_{11}, v_{12}), \ldots, (v_{1,l-1}, v_{1l}), (v_{1l}, v_{2l}), \ldots, (v_{l-1,l}, v_{ll})\}$, with net expected cost $2l + (l-1)\sigma(l-1)/l$. As $l \to \infty$, the costs of these two solutions can be approximated by $2\sigma l$ and $2l + \sigma l$. At $\sigma = 2$, these solutions cost $4l$ while the optimal unrestricted solution costs $3l$, demonstrating that demanding the first-stage solution to be a tree may cause the solution to be asymptotically a factor $4/3$ worse than optimum.

In the above construction, if we were to consider the worst ratio for a tree solution in the first-stage that includes all edges of the first-stage optimal solution, then the second solution would fit the bill. Its cost is $2l + \sigma l$ as compared to the optimal solution's cost of $l(1+\sigma)$, so the ratio tends to $3/2$ as σ tends to 1.

3.2. LP rounding algorithm. In light of Lemma 3.1 and the ease of dealing with trees, we will henceforth solve the problem where E_0 is a tree, which we call T^0. In this case, the path from every terminal in scenario k consists of a portion of only recourse edges, followed by a portion consisting of only first-stage edges, as in the proof of Lemma 3.1. This enables us to write a *stronger* IP formulation for the problem, and we can then round the linear relaxation of this IP formulation to within a constant factor.

First we note some simplifying assumptions. These assumptions apply throughout the paper, including §4 and 5.

- The edge costs c obey the triangle inequality. This is without loss of generality, because if there is a violation of the triangle inequality in the input graph, we can replace an edge by the shortest path between its endpoints and use the shortest path in our solution whenever the corresponding edge is indicated for use by the algorithm.
- Each terminal occurs in at most one scenario S_k. This is also without loss of generality: Because we are considering finitely many scenarios and listing each scenario explicitly, we can just replicate vertices as required.

The revised IP is shown in (8)–(14). The x^0 variables are indicators for the installation of edges in the first stage, and $x^1, \ldots, x^k$ are the indicators for the recourse stage. For a terminal t in scenario k, the variable $r_e^k(t)$ indicates whether edge e is used in the recourse portion of t's path to the root, and $r_e^0(t)$ indicates whether it is used in the first-stage portion of the path. These flow variables are *directed*; that is, for $e = (uv)$, the variable $r_{uv}^k(t)$ denotes the flow of commodity t along a recourse edge in the direction u to v. Given these directed flow variables, the cut sets are defined as $\delta_+(S) = \{e = (u, v)\colon u \in S,\ v \notin S\}$ and $\delta_-(S) = \delta_+(V \backslash S)$. For a singleton

vertex v, we abuse notation slightly to denote $\delta_+(\{v\})$ by $\delta_+(v)$; $\delta_-(v)$ is defined similarly. Note, however, that the edge installation variables x_e^k refer to undirected edges, and the graph itself remains undirected.

$$\min \sum_{e\in E} c(e)x_e^0 + \sum_{k=1}^{m} p_k\sigma_k \sum_{e\in E} c(e)x_e^k \tag{8}$$

$$\sum_{e\in\delta_+(t)} (r_e^0(t) + r_e^k(t)) \geq 1 \quad \forall\, t \in S_k, \ \ \forall\, k \tag{9}$$

$$\sum_{e\in\delta_+(v)} (r_e^0(t) + r_e^k(t)) - \sum_{e\in\delta_-(v)} (r_e^0(t) + r_e^k(t)) = 0 \quad \forall\, v \notin \{t, r\}, \ \ \forall\, t \in S_k, \ \ \forall\, k \tag{10}$$

$$\sum_{e\in\delta_-(v)} r_e^0(t) \leq \sum_{e\in\delta_+(v)} r_e^0(t) \quad \forall\, v \notin \{t, r\}, \ \ \forall\, t \in S_k, \ \ \forall\, k \tag{11}$$

$$r_e^k(t) \leq x_e^k \quad \forall\, e, \ \ \forall\, t \in S_k, \ \ \forall\, k \tag{12}$$

$$r_e^k(t), x_e^k \geq 0 \quad \forall\, e, \ \ \forall\, t \in S_k, \ \ \forall\, k \tag{13}$$

$$r_e^k(t), x_e^k \in \{0, 1\} \quad \forall\, e, \ \ \forall\, t \in S_k, \ \ \forall\, k. \tag{14}$$

Constraint (9) ensures that there is one unit of flow leaving each terminal in each scenario. Constraint (10) enforces flow conservation for intermediate nodes. Recall that the path from a terminal to the root consists of a prefix comprising only recourse edges followed by a suffix of only first-stage edges; this means that at every intermediate node, for every terminal, the net first-stage inflow must be bounded from above by the net first-stage outflow; this is enforced by (11). In fact, this monotonicity constraint is slightly weaker than requiring the first-stage solution to be a tree. However, any optimal solution to the integer program, without loss of generality, results in a first-stage component that is a tree. Finally, constraint (12) ensures that edges with flow are bought and paid for in the objective function.

Rounding overview. The linear relaxation of our formulation is given by (8)–(13), obtained by dropping the integrality constraint (14). We begin by solving this linear relaxation; let (x, r) denote an optimal solution to (8)–(13). The basic approach is that if we have a graph with a set of terminals and fractional edge variables x, such that any cut separating some terminals from the root has x-value at least 1, we have a fractional Steiner tree that we can round within a factor of two, using, say, Agrawal et al. [1]. Our aim, therefore, is to extract a similar situation out of our fractional solution (x, r) where the cut values for the first-stage variables (x^0) are at least some constant, and round it to a first-stage Steiner tree. However, if the recourse costs dominate, we must use the recourse LP support to guide our choice of recourse trees. Our new idea here is to use the primal-dual algorithms for Steiner trees to grow such recourse trees, but to truncate this process when the growing moats (cuts) obey the first condition of having a constant support value for the first-stage variables crossing them. To implement this idea, we modify the graph a little in order to take care of various issues. The rounding algorithm has several stages, which we describe in detail below.

3.2.1. Path decomposition. We begin by decomposing the LP solution into a set of flow paths. Consider any terminal t in scenario k. Constraints (9)–(10) imply that the variables $r_e^k(t)$ and $r_e^0(t)$ constitute a fractional flow of one unit from t to r. Using the monotonicity constraint (11), we can decompose the fractional flow into a set of flow paths as defined below. The proof of this simple proposition is omitted. Here, P_t consists of a set of flow paths with total flow at least one unit; moreover, on each path p, the flow is monotone: using only recourse edges until a transition point v_p, and only first-stage edges from the transition point to the root. Any standard flow decomposition procedure can be used to find such a decomposition from a given feasible LP solution.

PROPOSITION 3.1. *Every terminal t (in scenario k) has a set of flow paths P_t, and each path $p \in P_t$ has a* flow value $f(p) \in (0, 1]$ *and a unique* transition point v_p *such that the following hold*:

(i) $|P_t| \leq |E|$;

(ii) $\sum_{p\in P_t} f(p) \geq 1$;

(iii) *if $p(xy)$ denotes the segment of path p that originates at node x and terminates at node y, then for every edge $e \in p(tv_p)$, we have $\sum_{p\in P_t: e\in p} f(p) = r_e^k(t)$ and for every edge $e \in p(v_p r)$, we have $\sum_{p\in P_t: e\in p} f(p) = r_e^0(t)$; and*

(iv) *P_t, f and $\{v_p\colon p \in P_t\}$ can be computed in polynomial time.*

Figure 2 illustrates the path decomposition of the flow from every terminal to the root.

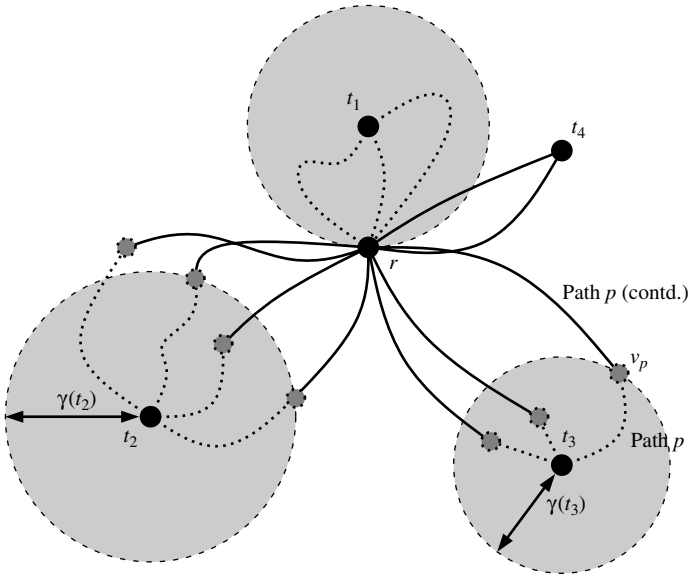

FIGURE 2. Illustration of transition points and critical radii. From each terminal, there are several paths carrying flow to the root r, with the segment of the path carrying recourse flow shown dotted and the segment carrying Stage 1 flow shown solid. For any path p, the transition point v_p is the point at which the flow switches from recourse to Stage 1 type; transition points are shown as dark grey nodes. The balls $B(t, \gamma(t))$ of radius equal to the critical radius for each terminal are shown as the light grey circles.

3.2.2. Balls and representatives. We now identify a distance for each terminal beyond which most of the flow is on Stage 1 cables, and within which most of the flow is of recourse type. Formally, given a distance $\gamma \geq 0$, define the γ-ball around t to be $\mathbf{B}(t, \gamma) = \{v\colon c(t, v) \leq \gamma\}$. (We can subdivide edges to consider interior points of edges to be at the boundary as appropriate.) For a terminal t, define the *critical radius* $\gamma(t) \in \mathbb{R}$ as $\gamma(t) = \min\{\gamma\colon \sum_{p\colon v_p \in B(t,\gamma)} f(p) \geq 1/2\}$; i.e., it defines the least distance around t that contains the transition points for at least half the flow. Because for every terminal t, every path $p \in P_t$ contains a unique transition vertex v_p, $\gamma(t)$ exists for every terminal and can be found by a shortest-path computation. We also define $\gamma(r) = 0$. The critical radii $\gamma(t)$ and balls $\mathbf{B}(t, \gamma(t))$ are also illustrated in Figure 2.

PROPOSITION 3.2. *For every terminal t and every subset $S \subseteq V\backslash\{r\}$ such that $\mathbf{B}(t, \gamma(t)) \subseteq S$, we have $\sum_{e\in\delta(S)} x^0_e \geq 1/2$.*

PROOF. Consider the set P_t of flow paths. Proposition 3.1 implies that there exists a subset of paths $P'_t \subseteq P_t$ such that for every $p \in P'_t$, we have $v_p \in S$. Moreover, using the same proposition, we have $\sum_{p\in P'_t} f(p) \geq 1/2$. Finally, Proposition 3.1 and Constraint (12) ensure that $\sum_{p\in P'_t} f(p) \leq \sum_{e\in\delta(S)} r^0_e(t) \leq \sum_{e\in\delta(S)} x^0_e$, and the proposition follows. □

If we select a set of such balls that are disjoint, and contract them, then we can round the x^0-values outside these balls to an integer Steiner tree in the contracted graph at a cost that is at most twice that of the linear relaxation. However, this does not give us Steiner trees in the original uncontracted graph because we need to pay for edges from the boundaries of these balls to the terminals at their centers. To handle this difficulty, we use an additional step introduced by Ravi and Salman in [23] and used subsequently in Garg et al. [7], Gupta et al. [14], and Talwar [28].

PROPOSITION 3.3. *There exists a set of terminals R^0 (called the* set of representative terminals*) computable in polynomial time such that*:

(i) *the root r lies in R^0*;

(ii) *For every pair of distinct terminals $t, t' \in R^0$, we have $\mathbf{B}(t, 2\gamma(t)) \cap \mathbf{B}(t', 2\gamma(t')) = \varnothing$; and*

(iii) *for any scenario k and any $v \in S_k\backslash R^0$, we have a* representative terminal $\mathrm{rep}(v) \in R^0$ *such that $\mathbf{B}(\mathrm{rep}(v), 2\gamma(\mathrm{rep}(v))) \cap \mathbf{B}(v, 2\gamma(v)) \neq \varnothing$; moreover, $\gamma(\mathrm{rep}(v)) \leq \gamma(v)$. Loosely, each terminal v is "close" to its representative; i.e., at distance at most $4\gamma(v)$ from* $\mathrm{rep}(v)$.

PROOF. Begin by including r in R^0, and proceed by examining all terminals in $\bigcup_k S_k$ in nondecreasing order of their critical radii, breaking ties arbitrarily. If terminal v is being examined and is such that $\mathbf{B}(t, 2\gamma(t)) \cap \mathbf{B}(v, 2\gamma(v)) = \varnothing$ for all $t \in R^0$, then include v in R^0. If not, then there exists $t \in R^0$ such that $\mathbf{B}(t, 2\gamma(t)) \cap \mathbf{B}(v, 2\gamma(v)) \neq \varnothing$; define $\mathrm{rep}(v) = t$ (t is marked as the representative of v), and proceed to the next vertex. □

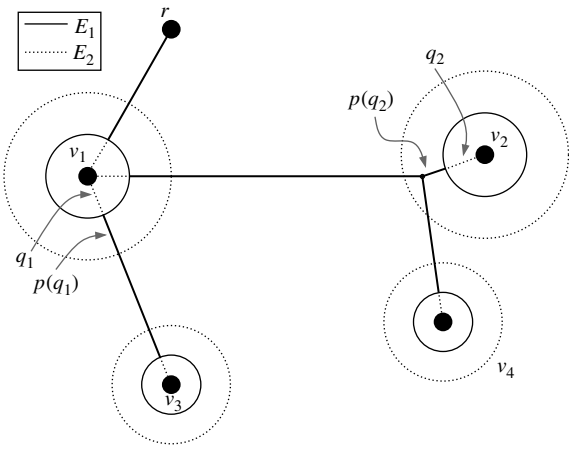

FIGURE 3. Illustration of first-stage tree computation described in Lemma 3.2. The balls with solid lines denote $\mathbf{B}(t, \gamma(t))$, while the balls with dotted lines denote $\mathbf{B}(t, 2\gamma(t))$.

The set of representatives R^0 is chosen as the set of terminals for the first-stage tree. The reason we construct our set of representatives as in Proposition 3.3 is this: For each terminal that has to be connected in the second stage, there is a first-stage terminal close by that it can try to connect to. We will now bound the cost of the first-stage tree, and then proceed to describe the construction of the second-stage tree. At this point, recall that the terminals of distinct scenarios are assumed to be disjoint.

3.2.3. The first-stage tree. Our first-stage tree is essentially a low-cost Steiner tree on the terminal set R^0, using the same distance function c on the original graph G. However, in order to obtain an upper bound on the cost of the tree, we use a slightly modified procedure. For an illustration, refer to Figure 3.

We begin by contracting $\mathbf{B}(t, \gamma(t))$ into a single vertex for each $t \in R^0$. We then compute a Steiner tree E_1 for this contracted graph, using the primal-dual Steiner tree approximation algorithm (Agrawal et al. [1], Goemans and Williamson [8]) with a performance ratio of two. In Figure 3, the edges in E_1 are shown as solid lines. We then connect each edge incident to $\mathbf{B}(t, \gamma(t))$ to the vertex t using a shortest path; these edges are shown by dotted lines in Figure 3 and are denoted as E_2. Our first-stage solution is the Steiner tree $T^0 = E_1 \cup E_2$.

The following lemma was first proved in a slightly different form in Ravi and Salman [23]; we provide a proof here for completeness.

LEMMA 3.2. *The cost of T^0 is at most $8 \cdot \sum_{e \in E} c(e) x^0_e$.*

PROOF. Let G' denote the graph obtained when the balls $\mathbf{B}(t, \gamma(t))$ are contracted to singleton vertices. Recall that the undirected cut IP formulation for a minimum-cost Steiner tree in G' is given by (15)–(17) below, where the set of terminals R refers to our representative set R^0.

$$\min \sum_{e \in E(G')} c(e) y_e \tag{15}$$

$$\sum_{e \in \delta(C)} y_e \geq 1 \quad \forall C\colon r \notin C, \quad C \cap R \neq \varnothing \tag{16}$$

$$y_e \in \{0, 1\} \quad \forall e. \tag{17}$$

Consider the fractional solution given by $y = 2x^0$. Proposition 3.2 and the construction of R^0 implies that y is feasible for Constraints (16). Hence, y is a feasible fractional solution for the linear relaxation of (15)–(17). The primal-dual algorithm for Steiner tree (Agrawal et al. [1], Goemans and Williamson [8]) implies that

$$\sum_{e \in E_1} c(e) \leq 2 \sum_{e \in E} c(e) y_e \leq 4 \sum_{e \in E} c(e) x^0_e. \tag{18}$$

We will now bound the total cost of edges in E_2 by the cost of the edges in E_1. Let ET^0 denote an Eulerian tour constructed by a depth-first search traversal of T^0. Let $p(t_1, t_2)$ denote the path in ET^0 between any two consecutive nodes t_1 and t_2. Divide this path into three disjoint segments: the segment $p(t_1, t_2) \cap \mathbf{B}(t_1, \gamma(t_1))$, the segment $p(t_1, t_2) \cap E_1$, and finally, the segment $p(t_1, t_2) \cap \mathbf{B}(t_2, \gamma(t_2))$. Proposition 3.3(ii) implies that the

cost of the middle segment $\sum_{e\in p(t_1,t_2)\cap E_1} c(e) \geq \gamma(t_1)+\gamma(t_2)$, while by construction we have the costs of the other two segments is $\sum_{e\in p(t_1,t_2)\cap \mathbf{B}(t_1,\gamma(t_1))\cup p(t_1,t_2)\cap \mathbf{B}(t_2,\gamma(t_2))} c(e) \leq \gamma(t_1)+\gamma(t_2)$. Hence, $\sum_{e\in p(t_1,t_2)\cap E_2} c(e) \leq \sum_{e\in p(t_1,t_2)\cap E_1} c(e)$. Summing over all segments of ET^0, we obtain:

$$2\sum_{e\in E_2} c(e) \leq 2\sum_{e\in E_1} c(e) \Rightarrow \sum_{e\in E_2} c(e) \leq \sum_{e\in E_1} c(e).$$

Combining this inequality with (18), we get that

$$\sum_{e\in T^0} c(e) = \sum_{e\in E_1} c(e) + \sum_{e\in E_2} c(e) \leq 2\sum_{e\in E_1} c(e) \leq 8\sum_{e\in E} x_e^0 c(e). \quad \square$$

This completes our first-stage solution. We now have to construct second-stage solutions for each scenario k to connect the vertices in $S^k\backslash T^0$ to the first-stage tree T^0, and bound their expected cost. While we do have a fractional solution to work with, it is not amenable to rounding. Hence, we have to run a primal-dual subroutine to construct the second-stage trees, but use the fractional solution to guide and "prematurely" halt the primal-dual subroutine.

3.2.4. Component growth. We examine each second-stage scenario separately, and construct a complete second-stage solution for scenario k using the corresponding second-stage fractional variables $(x_e^k, r_e^k(t))$ and the first-stage solution. Our algorithm adds one additional step to the classic primal-dual algorithm for the undirected Steiner tree problem described in Agrawal et al. [1] and Goemans and Williamson [8].

Fix a scenario k, and let $\hat{S}_k = S_k\backslash R^0$ be the set of terminals in scenario k that still need to be connected in the second-stage solution T^k.

Definition 3.1. A *valid moat collection* $\mathcal{M}$ is a set of subsets of V (called *moats*) such that the following holds for all moats $M \in \mathcal{M}$:

(i) the moats are disjoint; i.e., for any other moat $M' \in \mathcal{M}$, we have $M \cap M' = \varnothing$;
(ii) $M \cap \hat{S}_k \neq \varnothing$;
(iii) $M \cap R^0 = \varnothing$; and
(iv) for some terminal $t \in S_k \cap M$, we have $\sum_{p\in P_t: p(tv_p)\in M} f(p) \leq 1/2$.

Loosely, our goal will be to grow these moats as in the primal-dual algorithm, and build second-stage trees inside these moats. It suffices if a second-stage tree connects to the first-stage tree (via a terminal in R^0); hence, when Condition (iii) becomes false, we can terminate the growth of such a moat. We also need to bound the cost of the tree inside each moat using the fractional solution; this is done using Condition (iv) of Definition 3.1. Conditions (i) and (ii) are basic requirements for the primal-dual algorithm for the undirected Steiner tree problem. A more detailed description of our algorithm follows.

We initialize our moat collection $\mathcal{M}$ to be the singleton sets comprising terminals in $\hat{S}_k$, that is, $\mathcal{M} = \{\{t\}: t \in \hat{S}_k\}$. (Note that this moat collection is *valid*, as in Definition 3.1.) We also maintain a set $\mathcal{C}_k$ of *capped moats*, initialized to be the empty set. Each moat has an associated *dual value* z_M, initially zero. For the purposes of exposition, let us assume that each edge is subdivided into infinitesimally small edges such that all edges in the graph have length ϵ, and every such path consists of an even number of ϵ-length edges. Our algorithm is explained in terms of this expository artifact; the reader is referred to Agrawal et al. [1] and Goemans and Williamson [8] for details about a combinatorial, polynomial-time, artifact-free implementation of the algorithm.

We maintain a *time counter* τ, which is incremented by ϵ at each iteration. The iterations terminate when the moat collection $\mathcal{M}$ becomes empty. At each iteration, we update each moat $M \in \mathcal{M}$ by adding all vertices within distance ϵ from M. That is, $M := M \cup \{v \in V: c(u,v) \leq \epsilon \text{ for some } u \in M\}$. We also increase the dual value z_M by ϵ for each moat $M \in \mathcal{M}$. Before proceeding to the next iteration, we perform the following checks to maintain the invariant that $\mathcal{M}$ remains a valid moat collection according to Definition 3.1:

(i) If there exist moats $M, M' \in \mathcal{M}$ such that $M \cap M' \neq \varnothing$, then remove M and M' from $\mathcal{M}$ and add $M \cup M'$ to $\mathcal{M}$. Set its dual value $z_{M\cup M'} = 0$ at this point. Repeat this replacement one pair at a time until Property (i) of Definition 3.1 holds.

(ii) If for some M we have $M \cap R^0 \neq \varnothing$, remove M from $\mathcal{M}$ and add M to $\mathcal{C}_k$. Label M as "*R-capped*."

(iii) If for some M, we have $\sum_{p\in P_t: p(tv_p)\in M} f(p) \geq 1/2$ for all $t \in \hat{S}_k \cap M$, remove M from $\mathcal{M}$ and add it to $\mathcal{C}_k$. Label M as "*f-capped*."

(iv) If for some $M \in \mathcal{M}$ and some $M' \in \mathcal{C}_k$ we have $M \cap M' \neq \varnothing$, remove M from $\mathcal{M}$, and add M to $\mathcal{C}_k$. Label M as "*C-capped*."

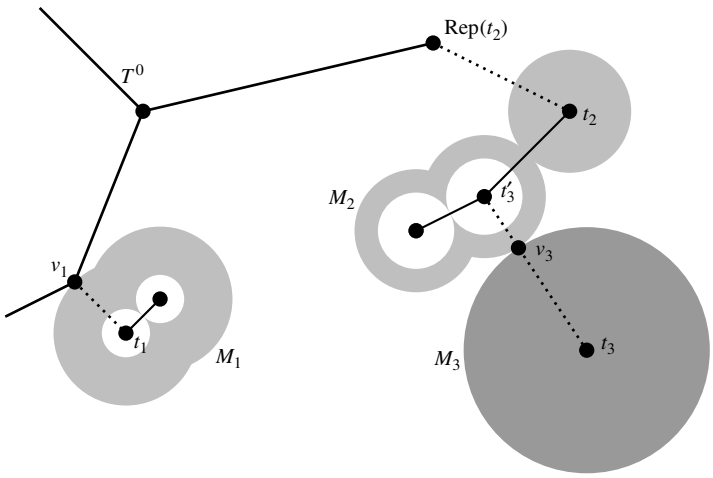

FIGURE 4. Illustration of the algorithm for the second-stage solution for a fixed scenario. There are three moats in $\mathscr{C}_k$, with M_1, M_2, and M_3, being, respectively, R-capped, f-capped, and C-capped. The thick lines denote T^0, the thin lines denote the internal trees, and the dotted lines denote the connecting paths.

Whenever a moat M gets added to $\mathscr{C}_k$, let $\tau_M = \tau$ denote the *termination time* of moat M. This procedure terminates when $\mathscr{M}$ becomes empty, which must happen in a finite number of iterations since $r \in R^0$. We now proceed to the construction of the second-stage tree for scenario k. For a brief illustration of this procedure, refer to Figure 4.

The following proposition is immediate based on the description of the algorithm, and is therefore stated without proof. It will be useful in the construction of the second-stage tree.

PROPOSITION 3.4. *For any moat $M \in \mathscr{M} \cup \mathscr{C}_k$ and any vertex $v \in M$, there exists a terminal $t \in \hat{S}_k \cap M$ such that $c(t, v) \le \tau_M$.*

3.2.5. Recourse Steiner trees for scenarios: Internal trees. Consider any capped moat $M \in \mathscr{C}_k$, and consider an instance of the undirected Steiner tree problem on G where the terminals are $\hat{S}_k \cap M$. The primal-dual algorithm of Agrawal et al. [1] grows moats in precisely the same way as we have described in our procedure. However, it simultaneously constructs a Steiner tree connecting the terminals in $\hat{S}_k \cap M$. Let T_M denote this Steiner tree, and refer to it as the *internal tree* for moat M. The following lemma was proved in Agrawal et al. [1] and Goemans and Williamson [8], and hence is stated here without proof.

LEMMA 3.3. *For any capped moat $M \in \mathscr{C}_k$, we have $2\tau_M + \sum_{e \in T_M} c(e) \le 2 \sum_{M' \subseteq M} z_M$.*

LEMMA 3.4. *For any capped moat $M \in \mathscr{C}_k$, we have $\sum_{M' \subseteq M} z_M \le 2 \sum_{e \in M} c(e) x_e^k$.*

PROOF. We will prove a stronger statement: For any moat that belonged to $\mathscr{M}$ at any point in the algorithm, we have $\sum_{M' \subseteq M} z_M \le 2 \sum_{e \in M} c(e) x_e^k$. The proof is by induction on increments of the time counter τ. Clearly the claim holds at the beginning of the algorithm, since all moats consist of singleton vertices.

Consider an iteration and a moat $M \in \mathscr{M}$. The dual value z_M increases by ϵ. However, Condition (iv) guarantees that there exists some terminal $t \in \hat{S}_k \cap M$ such that $\sum_{p \in P_t : p(tv_p)} f(p) \le 1/2$. This means that at least half a unit of flow from terminal t is leaving moat M on recourse edges, that is, $\sum_{e \in \delta(M)} x_e^k \ge 1/2$. Let dM denote the edges that got included in M when it grew in the current iteration. These edges constitute disjoint fragments of the flow paths in P_t of length at least ϵ and with $\sum_{e \in dM} x_e^k \ge 1/2$. Hence, the total increase in the quantity $\sum_{e \in M} c(e) x_e^k$ is at least $\epsilon/2$, thus completing the proof of the claim. □

3.2.6. Recourse Steiner trees for scenarios: Connecting paths. The internal trees are sufficient for connecting terminals within each moat in $\hat{S}_k \cap M$. It still remains to connect these trees to the first-stage solution. The edges we use for these connections are called *connecting paths*, denoted p_M. We now describe the construction of these connecting paths and bound their cost. The connecting path of each moat is constructed separately, depending on its capping condition. Let $M \in \mathscr{C}_k$ be a capped moat.

Case (i) *M is R-capped*: According to Proposition 3.4, there exist $v \in R^0$ and $t \in \hat{S}_k \cap M$ such that $c(t, v) \le \tau_M$. Let the connecting path p_M for M be a shortest path between v and t. This is illustrated in Figure 4, with M, v, and t represented by M_1, v_1, and t_1 respectively.

LEMMA 3.5. $\sum_{e\in T_M\cup p_M} c(e) \le 4\sum_{e\in M} c(e)x_e^k$.

PROOF. Because the cost of the path p_M is $\sum_{e\in p_M} c(e) \le \tau_M$, the total cost is $\sum_{e\in T_M\cup p_M} c(e) \le \tau_M + \sum_{e\in T_M} c(e)$; now applying Lemmas 3.3 and 3.4 completes the proof. □

Case (ii) *M is f-capped*: In the iteration immediately preceding the iteration when M got capped, there must have been at least one terminal $t \in \hat{S}_k \cap M$ such that $\sum_{p\in P_t:\, p(tt_p)\in M} f(p) \le 1/2$. Recall from Proposition 3.3 that there exists a representative $\text{rep}(t) \in R^0$ for t; let the connecting path p_M be a shortest path from t to $\text{rep}(t)$. In Figure 4, this is illustrated using M_2 and t_2 to represent M and t, respectively.

LEMMA 3.6. $\sum_{e\in T_M\cup p_M} c(e) \le 20\sum_{e\in M} c(e)x_e^k$.

PROOF. Using Proposition 3.3, it follows that $c(t,\text{rep}(t)) \le 2\gamma(t) + 2\gamma(\text{rep}(t)) \le 4\gamma(t)$. Recall our definition of the critical radius $\gamma(t)$: It is the distance at which balls centered at t contain the critical points of paths with at least one-half unit of recourse flow. However, the specification of f-capped moats (Condition (iv)) implies that at time τ_M, for terminal t in the moat, paths p with one-half unit of recourse flow have their entire recourse segments $p(tv_p)$ within the moat M. By definition of the critical radius $\gamma(t)$, this means that

$$\gamma(t) \le \max_{v_p:\, p(tv_p)\in M} c(t,v_p) \le \sum_{e\in T_M} c(e) + \tau_M$$

The right-hand side in the above inequality is an upper bound on the maximum distance from any terminal t in M to the boundary of M when it is f-capped, and thus an upper bound on the distance to the transition points of any one of the trapped recourse segments from this terminal t. Because the connecting path p_M has length of at most $4\times\gamma(t)$, we have $\sum_{e\in T_M\cup p_M} c(e) \le 5\sum_{e\in T_M} c(e) + 4\tau_M < 20\sum_{e\in M} c(e)x_e^k$, where the last inequality follows from Lemmas 3.3 and 3.4. □

Case (iii) *M is C-capped*: In this case, there must exist $M' \in C_k$ such that $v \in M \cap M'$. By Proposition 3.4, there exists $t \in M$ such that $c(t,v) \le \tau_M$ and $t' \in M'$ such that $c(t',v) \le \tau_{M'}$. Define the connecting path p_M to be the union of a shortest path from t to v and a shortest path from v to t'. In Figure 4, this is illustrated with $\{M, M', t, t', v\}$ represented by $\{M_3, M_2, t_3, t_2, v_3\}$.

LEMMA 3.7. $\sum_{e\in T_M\cup p_M} c(e) \le 4\sum_{e\in M} c(e)x_e^k$.

PROOF. By definition of the algorithm, it follows that $\tau_{M'} \le \tau_M$. Therefore, $\sum_{e\in p_M} c(e) \le c(t,v) + c(t',v) \le \tau_M + \tau_{M'} \le 2\tau_M$. The lemma now follows from Lemmas 3.3 and 3.4. □

3.2.7. The final solution. Our complete second-stage solution for scenario k is given by

$$T^k = \bigcup_{M\in\mathcal{C}_k} (T_M \cup p_M).$$

That is, the second-stage tree is the union of internal trees and connecting paths of the capped moats.

THEOREM 3.1. *The LP relaxation of* (8)–(14) *can be rounded within a factor of* 20 *in polynomial time.*

PROOF. First, we show that $T^0 \cup T^k$ is a Steiner tree that spans all vertices in S_k. Every terminal in $\hat{S}_k \cap R^0$ belongs to precisely one moat in $\mathcal{C}_k$. If the moat was R-capped or f-capped, then the internal tree added to the connecting path clearly connects all terminals in the moat to r. If the moat was C-capped, then the connection to r follows because the terminals in the moat are connected to another moat that was either R-capped, or f-capped, or C-capped with a termination time smaller than the current moat's, allowing for a lexicographic ordering to break ties.

Next, observe that the moats in $\mathcal{C}_k$ are disjoint. Therefore, we can bound the cost of the second-stage trees using Lemmas 3.5, 3.6, and 3.7 as follows: $\sum_{M\in\mathcal{C}_k}\sum_{e\in T_M\cup p_M} c(e) \le \sum_{M\in\mathcal{C}_k} 20\sum_{e\in M} c(e)x_e^k \le 20\sum_{e\in E} c(e)x_e^k$. Finally, using Lemma 3.2 to bound the cost of T^0, we obtain the guarantee claimed in this theorem. □

Recall that the formulation given in (8)–(14) was for a monotone near-optimal solution; now using Lemma 3.1 to relate this to the cost of the true optimum, we have the following result for the stochastic Steiner tree problem.

COROLLARY 3.1. *The stochastic Steiner tree problem can be approximated within a factor of* 40 *in polynomial time.*

4. Stochastic network design.

4.1. Problem definition. Recall the definition of the stochastic network design problem from §2: In this problem, a scenario consists of a set of clients, each of whom want to ship one unit of flow to the root r, and the cost incurred by this flow is a concave function of the total flow on an edge. In particular, in the

nonstochastic version of the problem, if the flow on the edge e is $f(e)$, then the cost incurred for that edge is $c(e)(\sigma_0 + \delta_0 f(e))$, where σ_0 and δ_0 are parameters specified by the problem instance. This can be viewed as modeling the situation when using the edge incurs a *fixed* or *building* cost of $\sigma_0 c(e)$, and there is an *incremental* or *routing* cost of $\delta_0 c(e)$ incurred for each unit of flow sent over the edge. In the stochastic version, the first stage behaves exactly the same, whereas in the second stage, the parameters σ_0 and δ_0 are replaced by σ_k and δ_k in case scenario k materializes. Once again, the goal is to minimize the expected cost of the solution.

Before we describe our algorithm, let us mention a slightly different version of our problem. Because all the routing takes place in the second stage, an alternative formulation of the problem might prescribe the per-unit routing-cost in scenario k to be $\delta_k c(e)$, regardless of whether the cable was installed in the first stage or the second stage. (Note that if the cable is installed in the first stage in our formulation, the per-unit routing cost is $\delta_0 c(e)$.) By scaling the demand at each terminal in scenario S_k to δ_0/δ_k, and redefining the second-stage routing-cost multiplier to be δ_k^2/δ_0, this version of the problem becomes equivalent to the problem we study, albeit with nonunit demands at each node. While we can duplicate nodes to handle this problem, it would be interesting to obtain a strongly polynomial approximation algorithm for stochastic network design when terminals want to send arbitrary amounts of flow to the root.

In the deterministic approximation algorithms literature, an oft-used formulation of the single-cable single-sink network design problem assumes that each cable has a fixed cost ($\sigma_0 c(e)$) and a capacity (u_0), such that only integral quantities of the cable may be installed on any edge. This formulation has an optimal value within a constant factor of the formulation we consider, and several papers have used this "equivalence" to develop approximation algorithms for network design problems Garg et al. [7], Guha et al. [9], and Talwar [28]. However, it is an open question whether this equivalence holds in our stochastic versions, and whether our results can be used to obtain approximation algorithms for the alternate version.

4.2. Preliminaries. At the time of the writing of this paper, it is an open question whether there exists a near-optimal solution for the stochastic network design problem where the first-stage solution is a tree. Note that Lemma 3.1 does not hold because it does not account for the routing costs.

However, recall that our formulation for stochastic Steiner tree (8)–(14) contains a monotonicity constraint on the flow from terminals to the root. Our formulation for stochastic network design (shown below) also has a similar constraint. This constraint ensures that in an optimal solution, the first-stage component (without loss of generality) is a tree.

Our algorithm for the network design problem proceeds along the lines of the stochastic Steiner tree algorithm. We begin by solving the linear relaxation of an IP formulation of the problem, rounding it to trees as before, and adding an additional step (described later) to account for the routing costs. The integer program formulation we will use is shown below, where $r_e^0(t)$ and $r_e^k(t)$ denote the flow along edge e of terminal t on first-stage and recourse cables, respectively, with $t \in S_k$. As in §3.2 and the IP formulation (8)–(14) for the stochastic Steiner tree, the flow variables $r_e^k(t)$ are directed, while the edge installation variables $x_e^k(v)$ are undirected.

$$\min \sum_{e\in E} c(e)\left[\sigma_0 x_e^0 + \sum_{k=1}^{m} p_k\left(\sigma_k x_e^k + \sum_{t\in S_k}(\delta_0 r_e^0(t) + \delta_k r_e^k(t))\right)\right] \tag{19}$$

$$\sum_{e\in\delta_+(t)} (r_e^0(t) + r_e^k(t)) \geq 1 \quad \forall\, t \in S_k, \ \ \forall\, k \tag{20}$$

$$\sum_{e\in\delta_-(v)} (r_e^0(t) + r_e^k(t)) - \sum_{e\in\delta_+(v)} (r_e^0(t) + r_e^k(t)) = 0 \quad \forall\, v \notin \{t, r\}, \ \ \forall\, t \in S_k, \ \ \forall\, k \tag{21}$$

$$\sum_{e\in\delta_-(v)} r_e^0(t) - \sum_{e\in\delta_+(v)} r_e^0(t) \leq 0 \quad \forall\, v \notin \{t, r\}, \ \ \forall\, t \in S_k, \ \ \forall\, k \tag{22}$$

$$r_e^k(t) \leq x_e^k \quad \forall\, e, \ \ \forall\, t \in S_k, \ \ \forall\, k \tag{23}$$

$$r_e^k(t), x_e^k \geq 0 \quad \forall\, e, \ \ \forall\, t \in S_k, \ \ \forall\, k \tag{24}$$

$$r_e^k(t), x_e^k \in \{0, 1\} \quad \forall\, e, \ \ \forall\, t \in S_k, \ \ \forall\, k. \tag{25}$$

The constraints (20)–(25) are identical to constraints (9)–(14) of the Steiner tree IP formulation; they encode the same basic idea of requiring a tree that supports a monotone flow path from each terminal to the root. The main difference lies in the objective function: If the scenario k materializes, and f_e units of flow are pushed through edge e, then our net cost on account of this flow-edge pair is $(\sigma_0 c(e) + \delta_0 f_e c(e))$ if the edge was

purchased in the first stage, and $(\sigma_k c(e) + \delta_k f_e c(e))$ if it was purchased in the second stage. That is, purchasing an edge in the first stage guarantees the ability to use it with an incremental cost of $\delta_0 c(e)$ per unit flow.

This problem is a strict generalization of stochastic Steiner tree, which can be obtained by setting $\delta_0 = \delta_k = 0$ for all k. In fact, we will often be using our algorithm for stochastic Steiner tree to obtain a partial solution for which the σ component of the cost can be easily accounted. By scaling the parameters suitably, we can assume that $\delta_0 = 1$.

Our main tool to account for the routing (or incremental flow) cost of the solution is a powerful theorem due to Khuller et al. [19]. The theorem is stated below without proof, adapted to our context. For any two vertices u, v and a subgraph H, let $c_H(u, v)$ denote the length of a shortest path between u and v using edges only in H. As before, $c(u, v)$ continues to denote $c_G(u, v)$.

THEOREM 4.1. *There exist constants $\alpha > 1$ and $\beta > 1$ such that given a graph G and a tree T rooted at r, there exists a tree $L(T)$ computable in polynomial time such that*:

(i) *$L(T)$ is also a tree spanning the vertices in T;*
(ii) *the weight of $L(T)$ is $\sum_{e \in L(T)} c(e) \le \alpha \sum_{e \in T} c(e)$;*
(iii) *for every vertex $t \in T$, the distance from t to r in $L(T)$ is $c_{L(T)}(r, t) \le \beta c_G(r, t)$; and*
(iv) *the parameter $\alpha = 1 + 2/(\beta - 1)$.*

Loosely, the above theorem implies that the tree $L(T)$ (which is known as a *Light Approximate Shortest-Path Tree*, or LAST), has the remarkable property that it closely approximates both a minimum Steiner tree (on $R \cup \{r\}$), as well as a shortest-path tree rooted at r. Because the two components of our cost function are (a) the weight of the tree and (b) the sum of routing costs from the terminals to the root, it is no surprisse that LASTs play a crucial role in our algorithms. Recall that our assumption that the graph is complete and all distances satisfy the triangle inequality continues to hold for this section.

4.3. Special case: $\delta_k = 1$ for all k. We first analyze a constant-factor approximation for the special case when $\delta_k = 1$ for all k. This yields the main ideas, which are developed further to provide an approximation algorithm for the general case.

Algorithm. The algorithm is fairly straightforward: We begin by solving the linear relaxation (19)–(24) of the formulation. As before, we let (x, r) denote the optimal solution to the linear relaxation, which we will use as a lower bound for the cost of our solution.

We then ignore the routing-cost component, and use the Stochastic Steiner Tree algorithm of the previous section to compute a first stage tree T_0, and a forest T^k for each scenario k, such that $T^0 \cup T^k$ is a tree for the scenario-k terminals S_k. While we will subsequently prove this formally, it should not be surprising that the σ (building) component of the cost function can be bounded using this strategy.

We continue by applying Theorem 4.1 to T^0 to get the corresponding LAST $L(T^0)$: The first-stage component of our solution will be this LAST $L(T^0)$. We then consider each scenario separately. When considering the kth scenario, we first contract $L(T^0)$ into a singleton "root" vertex r_k, updating the metric appropriately. In this contracted graph, the forest T^k appears as a Steiner *tree* rooted at r_k, and spanning the vertices in $S_k \setminus L(T^0)$. We now apply Theorem 4.1 to this tree to obtain the corresponding LAST, which we call $L(T^k)$. We then uncontract the graph, which may cause the LAST $L(T^k)$ to again split into many trees: With a slight abuse of notation, we use $L(T^k)$ to refer to the forest thus obtained. We output this forest $L(T^k)$ as the second-stage component of our solution for scenario k. An illustration is shown in Figure 5, where the heavy lines denote the first stage, and the lighter lines denote the second stage; the solid lines denote the original trees, and the dotted lines denote the corresponding LASTs.

The analysis. We begin by computing a bound on the cost of the σ component of our solution. The proposition below follows immediately from Theorems 3.1 and 4.1.

PROPOSITION 4.1. *For $k = 0, 1, \ldots, m$, we have $\sum_{e \in L(T^k)} c(e) \le 20\alpha \sum_{e \in E} c(e) x_e^k$.*

We now proceed to bound the routing-cost component of the solution. Because we are considering the special case of $\delta_k = 1 = \delta_0$ for all k, we will not use the linear programming bound; however, it will be used when we extend our algorithm to handle general δ_k in the next subsection.

LEMMA 4.1. *For any scenario k and any terminal $v \in S_k$, the distance $c_{L(T^0) \cup L(T^k)}(v, r)$ from v to the root in the tree $L(T^0) \cup L(T^k)$ is at most $\beta(\beta + 2) c(v, r)$.*

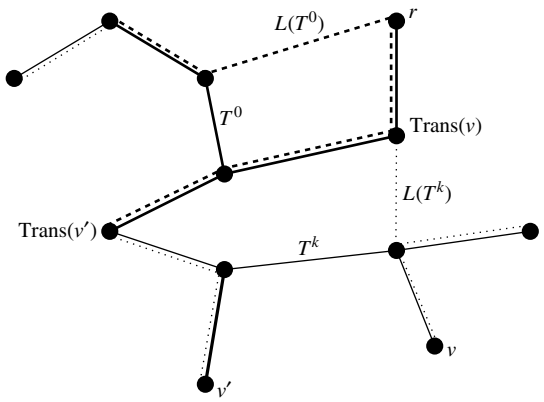

FIGURE 5. Illustration of the algorithm for stochastic network design.

PROOF. Consider the path from v to r in $L(T^0) \cup L(T^k)$, and let trans(v) be the first vertex in $L(T^0)$ in this path. Clearly, $c_{L(T^0)\cup L(T^k)}(v, r) = c_{L(T^0)}(\text{trans}(v), r) + c_{L(T^k)}(v, \text{trans}(v))$. (It may be useful to refer to Figure 5.)

Using the properties of the LAST from Theorem 4.1, and the triangle inequality, we can obtain the following two bounds:

$$\begin{aligned} c_{L(T^0)}(\text{trans}(v), r) &\le \beta c(\text{trans}(v), r) \le \beta(c(v, \text{trans}(v)) + c(v, r)), \\ c_{L(T^k)}(v, \text{trans}(v)) &= c_{L(T^k)}(v, r_k) \le \beta c(v, r_k) \le \beta c(v, r). \end{aligned} \tag{26}$$

Therefore, $c_{L(T^0)\cup L(T^k)}(v, r) \le \beta(c(v, \text{trans}(v)) + 2c(v, r))$. However, because $L(T^k)$ was constructed with r_k representing a singleton vertex obtained from contracting $L(T^0)$, note that the distance of v in the LAST $L(T^k)$ from r_k (and hence from all vertices in $L(T^0)$) must be close to its shortest-path distance; i.e., we must have

$$c(v, \text{trans}(v)) \le c_{L(T^k)}(v, \text{trans}(v)) \le \beta c(v, r).$$

Putting these all together, we obtain that $c_{L(T^0)\cup L(T^k)}(v, r) \le \beta(\beta + 2)c(v, r)$, as claimed. □

THEOREM 4.2. *The solution $L(T^0), L(T^1), \ldots, L(T^k)$ constitutes an 30.952-approximation for the stochastic network design problem for the special case $\delta_k = 1\ \forall k$.*

PROOF. We set $\alpha = 1.5476$, which results in $\beta = 4.6523$ using Theorem 4.1. Summing over all scenarios in Proposition 4.1 and over all terminals in Lemma 4.1 with these values of α and β yields the claimed approximation guarantee. □

4.4. General case. We now consider the case of general incremental costs, where each δ_k is different, some greater than one, and some smaller. Surprisingly enough, the algorithm is identical to the special case of constant incremental costs discussed above, barring one exception.

Algorithm description. As before, we begin by solving the LP formulation (19)–(24). Let $I_< = \{k\colon \delta_k < 1\}$ be the set of scenarios where routing costs are cheaper in recourse. For scenarios $k \in I_<$, let $S'_k = \{v \in S_k\colon r \in \mathbf{B}(v, 2\gamma(v))\}$. The exception is created by these terminals, and handled separately as follows. We construct approximate minimum Steiner trees $\widehat{T}^k$ using recourse edges for each S'_k, as described in (3.2.4)–(3.2.6) in the previous section, where we define $R^0 = \{r\}$, $\gamma(r) = 0$ and $\text{rep}(v) = v$ for every $v \in S'_k$. We then use Theorem 4.1 to compute $L(\widehat{T}^k)$ for each $k \in I_<$, using possibly different settings of parameters α' and β'. Note that unlike in §4.3, when creating the LASTs, we do not contract T^0—in fact, we have not even computed T^0 yet. Instead, $L(\widehat{T}^k)$ is computed by applying Theorem 4.1 directly on $\widehat{T}^k$, in the original graph G, with respect to the root r. We will bound the cost of these trees $L(\widehat{T}^k)$ in Lemma 4.4.

Once this is done, we then proceed in much the same way as for the case $\delta_k = 1$, except that when considering scenario k, the set of terminals is now S_k if $\delta_k \ge 1$, and is $S_k \setminus S'_k$ if $\delta_k < 1$. In other words, we again build the tree T^0, find the LAST $L(T^0)$ and contract it, and build the LASTs $L(T^k)$ for $k = 1, 2, \ldots, m$ as described in §4.3. For the rest of the analysis, we will use R^0 to denote the terminals contained in the tree T^0. Let us specify the final solution: The first-stage solution we produce is $L(T^0)$. For scenarios with $\delta_k \ge 1$, our second-stage solution is $L(T^k)$, whereas for the scenarios with $\delta_k < 1$, our second-stage solution is $L(T^k) \cup L(\widehat{T}^k)$.

Analysis of algorithm. Proposition 4.1 continues to hold and serves to bound the fixed-cost component of our solution, except for the cost of the trees $\widehat{T}^k$, which we bound in Lemma 4.4. However, before we do that, let us first bound the routing costs, which requires a new analysis that crucially uses the lower bounds given by the linear program. For the next two lemmas (4.2 and 4.3), we will abuse notation slightly and use $L(T^k)$ to refer to the union of $L(T^k)$ and $L(\widehat{T}^k)$ if $L(\widehat{T}^k)$ exists. This is simply for ease of notation, because we are only bounding the routing cost of the second-stage terminals.

Let us fix a scenario k, and consider a terminal $v \in S_k$. Define the *LP cost* of terminal v thus:

$$c^*(v) \doteq \sum_{e \in E} (\delta_0 r_e^0(v) + \delta_k r_e^k(v)) = \sum_{e \in E} (r_e^0(v) + \delta_k r_e^k(v)).$$

As in Lemma 4.1, let $\text{trans}(v)$ represent the first vertex in $L(T^0)$ encountered on the path from v to r in the solution $L(T^k) \cup L(T^0)$. Define the *routing cost* for the terminal v as:

$$c(v) \doteq \delta_k c_{L(T^k)}(v, \text{trans}(v)) + c_{L(T^0)}(\text{trans}(v), r).$$

We use $c^*(v)$ to bound $c(v)$, using two distinct arguments depending on whether $\delta_k \geq 1$ or $\delta_k < 1$. Note that while the bounds proved below are described for terminals $v \notin R^0$, they continue to hold for terminals $v \in R^0$ by simply defining $\text{rep}(v) = \text{trans}(v) = v$ for every $v \in R^0$.

LEMMA 4.2. *For a terminal $v \in S_k$ such that $\delta_k \geq 1$, the routing cost for v is $c(v) \leq (8\beta^2 + 9\beta)c^*(v)$.*

PROOF. We use two different lower bounds for $c^*(v)$ in this argument. Firstly, because $\delta_k \geq 1$, $c^*(v) \geq c(v, r) \cdot \delta_k \geq c(v, r)$. Secondly, because the critical radius $\gamma(v)$ is defined such that at least half a unit of flow uses flow paths whose transition points are at distance at least $\gamma(v)$ from v, the LP cost of v is at least $c^*(v) \geq (\delta_k/2)\gamma(v)$.

However, Proposition 3.3 guarantees that there must exist $\text{rep}(v) \in R^0$ at a distance $c(v, \text{rep}(v)) \leq 4\gamma(v)$. Using the guarantee from Theorem 4.1, we now have $c_{L(T^k)}(v, \text{trans}(v)) \leq \beta c(v, \text{rep}(v)) \leq 4\beta\gamma(v)$. Hence,

$$\delta_k c_{L(T^k)}(v, \text{trans}(v)) \leq \delta_k \cdot 4\beta\gamma(v) \leq 8\beta c^*(v). \tag{27}$$

It remains to bound the distance $c_{L(T^0)}(\text{trans}(v), r)$. Using Theorem 4.1 and the triangle inequality (in a manner identical to that used for (26)), we infer that $c_{L(T^0)}(\text{trans}(v), r) \leq \beta c(\text{trans}(v), r) \leq \beta(c(v, \text{trans}(v)) + c(v, r))$. Now, because $\delta_k \geq 1$, we have $c(v, \text{trans}(v)) \leq \delta_k c_{L(T^k)}(v, \text{trans}(v))$. Finally, using the lower bound $c(v, r) \leq c^*(v)$, and putting all these inequalities together yields

$$c_{L(T^0)}(\text{trans}(v), r) \leq \beta(8\beta + 1)c^*(v). \tag{28}$$

Adding this bound to (27) yields the bound for $c(v)$ claimed in the lemma. □

The following lemma now bounds the routing costs for all the terminals $v \in S_k$ for which $\delta_k < 1$, but where the terminals were at distance at least $2\gamma(v)$ from the root.

LEMMA 4.3. *For a terminal $v \in S_k$ such that $\delta_k < 1$ and $r \notin \mathbf{B}(v, 2\gamma(v))$, we have $c(v) \leq (8\beta^2 + 12\beta)c^*(v)$.*

PROOF. Again, we need to prove lower bounds on the LP cost $c^*(v)$: Note that the bound $c^*(v) \geq (\delta_k/2)\gamma(v)$ continues to hold as in Lemma 4.2, and hence we still have $\delta_k c_{L(T^k)}(v, \text{trans}(v)) \leq 8\beta c^*(v)$ given by (27). However, because the argument used to bound $c_{L(T^0)}(\text{trans}(v), r)$ in (28) depended crucially on δ_k being at least one, we need a new lower bound on $c^*(v)$. To this end, note that since $\sum_{p \in P_v : v_p \in \mathbf{B}(v, \gamma(v))} f(p) \geq 1/2$, at least half the flow must travel a distance of $c(v, r) - \gamma(v)$ while incurring the first-stage routing cost:

$$c^*(v) \geq \delta_0 \cdot \frac{1}{2}(c(v, r) - \gamma(v)) = \frac{1}{2}(c(v, r) - \gamma(v)). \tag{29}$$

Now because $r \notin \mathbf{B}(v, 2\gamma(v))$ is the same as saying $c(v, r) \geq 2\gamma(v)$, the inequality (29) implies $c^*(v) \geq \gamma(v)/2$. Yet again, as in (26), we use that

$$c_{L(T^0)}(\text{trans}(v), r) \leq \beta c(\text{trans}(v), r) \leq \beta(c(\text{trans}(v), v) + c(v, r)). \tag{30}$$

Furthermore, recall that each $v \in S_k$ has a representative in R^0 at a distance at most $4\gamma(v)$ from it. Now the distance $c(\text{trans}(v), v)$ is at most $c_{L(T^k)}(\text{trans}(v), v)$; because both $\text{trans}(v)$ and $\text{rep}(v)$ are contained in R^0, and $L(T^k)$ is a LAST when R^0 is contracted to a single node, this implies that

$$c(\text{trans}(v), v) \leq c_{L(T^k)}(\text{trans}(v), v) \leq \beta \times c(\text{rep}(v), v) \leq 4\beta\gamma(v). \tag{31}$$

Moreover, using $\gamma(v) \le 2c^*(v)$ in (29) results in

$$c(v, r) \le 4c^*(v). \tag{32}$$

Substituting these inequalities into (30) gives us that $c_{L(T^0)}(\text{trans}(v), r) \le \beta(4+8\beta)c^*(v)$, and adding this to the bound of $8\beta c^*(v)$ on $\delta_k c_{L(T^k)}(v, \text{trans}(v))$ completes the proof of the lemma. □

It remains to prove bounds on the cost of building the trees $L(\widehat{T}^k)$ for the sets S'_k; remember, the set S'_k was defined when $\delta_k < 1$, and consisted of all the terminals $v \in S_k$ that were close to (i.e., at distance at most $2\gamma(v)$ from) the root r. Furthermore, for these terminals, we need to bound their routing cost, because Lemma 4.3 does not consider them.

LEMMA 4.4. *For scenarios $k \in I_<$, we have $\sum_{e \in L(\widehat{T}^k)} c(e) \le 20\alpha' \sum_{e \in E} c(e)x^k_e$. Furthermore, for terminals $v \in S'_k$, we have $c(v) \le 4\beta' c^*(v)$; here $\alpha' = 1 + 2/(\beta' - 1)$, as in Theorem 4.1.*

PROOF. To begin, let us note that defining $R^0 = \{r\}$ and $\gamma(r) = 0$, and defining $\text{rep}(v) = r$ for all $v \in S'_k$ was a valid choice of the parameters when building the trees $\widehat{T}^k$, because indeed these parameters satisfy the conditions of Proposition 3.3. Thus, the result in Theorem 3.1 holds, bounding the total cost of each tree in $\{\widehat{T}^k\}_{k \in I_<}$ by $20 \sum_{e \in E} c(e)x^k_e$. Because one can make a LAST out of the tree by increasing the cost of the tree by at most a factor of α', one gets the claimed bound of $20\alpha' \sum_{e \in E} c(e)x^k_e$.

To analyze the routing cost, let us use the simplest lower bound on the LP cost $c^*(v)$; as in Lemmas 4.2 and 4.3, we infer that $c^*(v) \ge (\delta_k/2)\gamma(v)$. However, the routing cost of v is

$$c(v) = \delta_k c_{L(\widehat{T}^k)}(v, r) \le \delta_k \beta' c(v, r) \le \delta_k \beta' 2\gamma(v) \le 4\beta' c^*(v), \tag{33}$$

which completes the proof. □

THEOREM 4.3. *The solution $\{L(T^k)\colon 0 \le k \le m\} \cup \{L(\widehat{T}^k)\colon k \in I_<\}$ constitutes a 72.435-approximation for the stochastic network design problem in which the first stage is required to be a tree.*

PROOF. The routing cost of all terminals $v \in S_k \setminus L(T^0)$ are bounded above by $\max\{8\beta^2 + 12\beta, 4\beta'\}c^*(v)$ for all k using Lemmas 4.2, 4.3, and 4.4, and the quantities $c^*(v)$ are disjoint terms in the solution to the linear program (19)–(24). Proposition 4.1 continues to hold and bounds the building cost of the trees $L(T^k)$ by 20α times the cost in the linear programming solution, while the corresponding factor for the building cost of the trees $L(\widehat{T}^k)$ is $20\alpha'$. Hence, the total building cost of the tree in each scenario is bounded above by $20(\alpha + \alpha')$ times the corresponding cost in the linear programming solution.

We now apply Theorem 4.1 with $\alpha = 2.4802$ and $\alpha' = 1.1169$. This results in $\beta = 2.3512$ and $\beta' = 18.109$, yielding an overall approximation ratio of $\max\{20(\alpha + \alpha'), 8\beta^2 + 12\beta, 4\beta'\} = 72.435$. □

5. Risk-bounded network design. While typical stochastic optimization algorithms minimize the overall expected cost, a natural question to ask is that, given a particular scenario, how much is the algorithm requiring us to pay? A reasonable solution might require that the cost incurred in the second stage is comparable to the requirement of the second stage. A general way of modeling this is to assign *budgets* B_k for each scenario k, saying, "If scenario k materializes, the solution must not cost more than B_k in the second stage." Such a budget is a means for guarding the *downside risk*: the worst cost that could be incurred in any scenario.

A similar budget could also be specified for the first stage, although of course that could lead to an infeasible problem. (Even though portions of the routing cost are incurred using first-stage cables, we model this as wholly being incurred in the second-stage, and take this into account only in the second-stage budgets.) A powerful feature of the finite-scenario model and our solution technique is that this version of downside risk can be explicitly modeled, and our algorithm provides a solution that guards against it.

5.1. Steiner tree. We first consider bounding the risk of the solution for the stochastic Steiner tree problem. Formally, let $\{B_k\colon k \in I\}$ be a set of budgets, where $I \subseteq \{0, 1, \ldots, m\}$ is an index set, such that the following is also required (where $\sigma_0 = 1$):

$$\sum_{e \in E} \sigma_k c(e)x^k_e \le B_k \quad \forall k \in I. \tag{34}$$

That is, the budgeted stochastic Steiner tree problem is defined by adding Constraint (34) to the IP formulation (8)–(14). Notice that in the version of the budgeted Steiner tree problem, because we are using the IP formulation (8)–(14) to model the Steiner tree problem, we are implicitly assuming that the first-stage solution is required to be connected and contain the root. This is something we did not necessarily require of the Steiner tree problem,

but ensured anyway with an overhead of a factor of two via Lemma 3.1. This is because the transformation in Lemma 3.1 uses second-stage costs to bound the first-stage costs of the monotone fractional solution.

Our main result for this problem is stated below.

THEOREM 5.1. *There is a polynomial-time algorithm for the budgeted stochastic Steiner tree problem that either proves that the problem instance is infeasible* (*some/all of the budgets are too low*) *or provides a solution where each budget is violated by at most a factor of* 20.

PROOF. We begin by solving the linear relaxation of the problem, given by (8)–(13), (34). If this linear program is found to be infeasible, then the problem can be declared to be infeasible because some budgets are too low. Note that the problem can always be made feasible by making B_0 appropriately high, for any values of second-stage budgets.

If a feasible LP solution is found, we proceed by using exactly the same algorithm as for the stochastic Steiner tree. The key observation is that the algorithm bounds the cost of each scenario *locally*, using only its corresponding components of the LP solution: Theorem 3.1 bounds the cost of T^0 by $20\sum_{e\in E} c(e)x_e^0$, the cost of each component of T^k by $20\sigma_k \sum_{e\in E} c(e)x_e^k$. □

5.2. Network design. Before we study risk-bounding for stochastic network design, we need to clarify an accounting issue. Specifically, what costs are incurred at what points in time? Recall that the notion of buying a first-stage cable on edge e in this problem corresponds to *paying $\sigma_0 c_e$ in the present to reserve the right to transport goods at the rate of $\delta_0 c_e$ per unit flow in the future*. That is, all routing costs are only incurred in the second stage, after the demands have been realized. The building costs are accounted for as in the risk-bounded study of the stochastic Steiner tree.

Formally, we have a first-stage budget defined as follows, where B_0 may be set to ∞ if a first-stage budget is not required to be imposed.

$$\sum_{e\in E} \sigma_0 c(e)x_e^0 \le B_0 \tag{35}$$

Once again, we let $I \subseteq \{1, 2, \ldots, m\}$ be an index set for second-stage budgets specified by $\{B_k\colon k \in I\}$, with the following constraint:

$$\sum_{e\in E} c(e)\left(\sigma_k x_e^k + \sum_{t\in S_k} (\delta_0 r_e^0(t) + \delta_k r_e^k(t))\right) \le B_k \quad \forall k \in I. \tag{36}$$

The budgeted stochastic network design problem is obtained by adding Constraints (35)–(36) to the formulation (19)–(25). As before, by using the monotone LP formulation, we insist that the first-stage solution must be a connected graph containing the root node in this budgeted version. Our main result is the following theorem.

THEOREM 5.2. *There is a polynomial-time algorithm for the budgeted stochastic network design problem that either proves that the problem instance is infeasible* (*some/all of the budgets are too low*) *or provides a solution where each budget is violated by at most a factor of* 72.435.

PROOF. The proof closely follows that of Theorem 5.1. We solve the linear program given by (19)–(24), (35)–(36), and report problem infeasibility if the linear program is found to be infeasible. Once again, the fact that in Theorem 4.3 the bounds for the various components of the solution are obtained using their corresponding linear programming solution components results in this theorem. □

In fact, we can prove a (perhaps) more useful corollary. Suppose the network designer is flush with funds in the present, and is willing to tolerate excess first-stage expenditure. However, the second-stage risk must be controlled, meaning that the factor by which the second-stage budgets are overshot must be minimized. Our algorithmic techniques allow us to prove the following.

COROLLARY 5.1. *Given a feasible instance of the budgeted stochastic network design problem, for any $\rho > 40$, we can find a solution $L^0, L^1, \ldots, L^m$ such that the following hold*:

(i) *Any constraint of the form* (36) *is violated at most by a factor of ρ,*

(ii) *Constraint* (35) (*if specified*) *is violated at most by a factor of* $20(\sqrt{4\rho+9}-2)/(\sqrt{4\rho+9}-10)$.

(iii) *The overall approximation factor of the solution is* $\max\{\rho, 20((\sqrt{4\rho+9}-2)/(\sqrt{4\rho+9}-10)+2)\}$.

PROOF. Recall the proof of Theorem 4.3 where the bounds on the various components of the solution were computed. The overall bound on the second-stage component is $\max\{20\alpha', 8\beta^2+12\beta, 4\beta'\}$. For any $\rho > 40$, we can compute $\beta = (\sqrt{4\rho+9}-6)/4 > 1$ to satisfy $\rho = 8\beta^2+12\beta$. For such β, several values of α' and β' exist (in particular, $\alpha' = 2$ and $\beta' = 3$ so that $\max\{20\alpha', 8\beta^2+12\beta, 4\beta'\} = \rho$, proving (i).

Using Theorem 4.1 to compute α from the β computed above results in $\alpha = (\sqrt{4\rho+9}-2)/(\sqrt{4\rho+9}-10)$. The first-stage budget is violated by a factor of at most 20α, which proves (ii).

Finally, the overall approximation ratio is $\max\{20(\alpha+\alpha'), 8\beta^2+12\beta, 4\beta'\}$. Using the values of α, β, α', and β' computed above completes the proof. □

We note that the bounds above (and indeed, in our other main results as well) can be improved further by defining the critical radius more carefully. We chose not to do so for expositional clarity because the main goals of this paper were to show that these problems are constant-factor approximable, and describe our rounding procedure, which overlays a primal-dual subroutine on an optimal fractional solution.

6. Future work. We believe that our technique of solving the linear relaxation of an IP and using it to guide a primal-dual subroutine has applicability to other problems; it would be interesting to see if this is true. There are several issues considered in the stochastic programming community for which approximation algorithms are nonexistent or sparse at this point: multistage stochastic optimization, chance-constrained models, and scenario reduction and approximation are a few. Approximation algorithms are known for several complex versions of network design in a deterministic framework; it would be interesting to see if the stochastic versions of such problems can also be well approximated.

Acknowledgments. This work was done while the third author (Amitabh Sinha) was a graduate student at the Tepper School of Business at Carnegie Mellon University. R. Ravi and Amitabh Sinha were supported in part by NSF Grant CCR-0105548 and ITR Grant CCR-0122581 (The ALADDIN project). An earlier version of this paper (Gupta et al. [11]) appeared in the proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science. The authors thank the reviewers for their valuable comments. They also thank Nikhil Devanur for pointing out an error in Gupta et al. [11] (Lemma 3.4 in this version).

References

[1] Agrawal, Ajit, Philip Klein, R. Ravi. 1995. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Comput.* **24**(3) 440–456. (Preliminary version in *23rd STOC*, 1991.)
[2] Awerbuch, Baruch, Yossi Azar. 1997. Buy-at-bulk network design. *Proc. 38th Annual IEEE Sympos. Foundations Comput. Sci., Miami, FL*, IEEE Computer Society, 542–547.
[3] Beale, E. M. L. 1955. On minimizing a convex function subject to linear inequalities. *J. Roy. Statist. Soc. Ser. B.* **17** 173–184; discussion, 194–203. (Symposium on linear programming.)
[4] Birge, John R., François Louveaux. 1997. *Introduction to Stochastic Programming. Springer Series in Operations Research*. Springer-Verlag, New York.
[5] Dantzig, George B. 1955. Linear programming under uncertainty. *Management Sci.* **1** 197–206.
[6] Dye, Shane, Leen Stougie, Asgeir Tomasgard. 2003. The stochastic single resource service-provision problem. *Naval Res. Logist.* **50**(8) 869–887.
[7] Garg, Naveen, Rohit Khandekar, Goran Konjevod, R. Ravi, F. Sibel Salman, Amitabh Sinha. 2001. On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design formulation. *Proc. 8th Integer Programming Combin. Optim. Conf., Utrecht, The Netherlands*, *Lecture Notes in Computer Science*, Vol. 2081, Springer, 170–184.
[8] Goemans, Michel X., David P. Williamson. 1995. A general approximation technique for constrained forest problems. *SIAM J. Comput.* **24**(2) 296–317.
[9] Guha, Sudipto, Adam Meyerson, Kamesh Mungala. 2001. A constant factor approximation for the single sink edge installation problems. *Proc. 33rd Annual ACM Sympos. Theory Comput.* (*STOC*), *Crete, Greece*, ACM Press, New York, 383–388.
[10] Gupta, Anupam, Amit Kumar, Tim Roughgarden. 2003. Simpler and better approximation algorithms for network design. *Proc. 35th Annual ACM Sympos. Theory Comput., San Diego, CA*, ACM Press, New York, 365–372.
[11] Gupta, Anupam, R. Ravi, Amitabh Sinha. 2004. An edge in time saves nine: LP rounding approximation algorithms for stochastic network design. *Proc. 45th Annual IEEE Sympos. Foundations Comput. Sci., Rome, Italy*, IEEE Computer Society, 218–227.
[12] Gupta, Anupam, Martin Pál, R. Ravi, Amitabh Sinha. 2004. Boosted sampling: Approximation algorithms for stochastic optimization problems. *Proc. 36th Annual ACM Sympos. Theory Comput., Chicago, IL*, ACM Press, New York, 417–425.
[13] Gupta, Anupam, Martin Pál, R. Ravi, Amitabh Sinha. 2005. What about Wednesday? Approximation algorithms for multistage stochastic optimization. *Proc. 9th Internat. Workshop Approximation Algorithms Combin. Optim. Problems* (*APPROX*), *Berkeley, CA*, Springer, 86–98.
[14] Gupta, Anupam, Amit Kumar, Jon Kleinberg, Rajeev Rastogi, Bülent Yener. 2001. Provisioning a virtual private network: A network design problem for multicommodity flow. *Proc. 33rd Annual ACM Sympos. Theory Comput., Crete, Greece*, ACM Press, New York, 389–398.
[15] Hassin, Refael, R. Ravi, F. S. Salman. 2000. Approximation algorithms for a capacitated network design problem. *Internat. Workshop on Approximation Algorithms for Combin. Optim. Problems* (*APPROX*), *Saarbrücken, Germany*, Springer, 167–176.
[16] Hayrapetyan, Ara, Chaitanya Swamy, Éva Tardos. 2005. Network design for information networks. *Proc. 15th Annual ACM-SIAM Sympos. Discrete Algorithms, Vancouver, Canada*, SIAM, Philadelphia, PA, 933–942.
[17] Immorlica, Nicole, David Karger, Maria Minkoff, Vahab Mirrokni. 2004. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. *Proc. 15th Annual ACM-SIAM Sympos. Discrete Algorithms, New Orleans, LA*, SIAM, Philadelphia, PA, 684–693.

[18] Kall, Peter, Stein W. Wallace. 1994. *Stochastic Programming. Wiley-Interscience Series in Systems and Optimization*. John Wiley & Sons Ltd., Chichester, UK.
[19] Khuller, Samir, Balaji Raghavachari, Neal E. Young. 1995. Balancing minimum spanning and shortest path trees. *Algorithmica* **14**(4) 305–322.
[20] Klein Haneveld, Willem K., Maarten H. van der Vlerk. 1999. Stochastic integer programming: General models and algorithms. *Ann. Oper. Res.* **85** 39–57.
[21] Klein Haneveld, Willem K., Maarten H. van der Vlerk. 2003. *Stochastic Programming*. Department of Econometrics, University of Groningen, Netherlands.
[22] Mansour, Yishay, David Peleg. 2000. An approximation algorithm for minimum-cost network design. *Robust Communication Networks: Interconnection and Survivability* (*New Brunswick, NJ, 1998*). *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 53. American Mathematical Society, Providence, RI, 97–106.
[23] Ravi, R., F. S. Salman. 1999. Approximation algorithms for the traveling purchaser problem and its variants in network design. *Algorithms—ESA '99* (*Prague*), *Lecture Notes in Computer Science*, Vol. 1643. Springer, Berlin, Germany, 29–40.
[24] Ravi, R., Amitabh Sinha. 2006. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Math. Programming A* **108**(1) 97–114.
[25] Robins, Gabriel, Alexander Zelikovsky. 2000. Improved Steiner tree approximation in graphs. *Proc. 11th Annual ACM-SIAM Sympos. Discrete Algorithms, San Francisco, CA*, SIAM, Philadelphia, PA, 770–779.
[26] Salman, F. Sibel, Joseph Cheriyan, R. Ravi, Sairam Subramanian. 1997. Buy-at-bulk network design: Approximating the single-sink edge installation problem. *Proc. 8th Annual ACM-SIAM Sympos. Discrete Algorithms, New Orleans, LA*, SIAM, Philadelphia, PA, 619–628.
[27] Shmoys, David, Chaitanya Swamy. 2004. Stochastic optimization is (almost) as easy as deterministic optimization. *Proc. 45th Annual IEEE Sympos. Foundations of Comput. Sci., Rome, Italy*, IEEE Computer Society, 228–237.
[28] Talwar, Kunal. 2002. Single-sink buy-at-bulk LP has constant integrality gap. *Proc. 9th Integer Programming Combin. Optim. Conf., Cambridge, MA*, *Lecture Notes in Computer Science*, Vol. 2337, Springer, 475–486.
[29] Vazirani, Vijay V. 2001. *Approximation Algorithms*. Springer-Verlag, Berlin, Germany.