# Two-Stage Robust Network Design with Exponential Scenarios

Rohit Khandekar[1], Guy Kortsarz[2] ⋆, Vahab Mirrokni[3] ⋆⋆, and Mohammad R. Salavatipour[4] ⋆⋆⋆

[1] IBM T.J.Watson research center.
rkhandekar@gmail.com
[2] Department of Computer Science, Rutgers University-Camden. Currently visiting IBM Research at Yorktown Heights
guyk@crab.rutgers.edu
[3] Google research, New York, NY, USA
mirrokni@theory.csail.mit.edu
[4] Dept. of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada
mreza@cs.ualberta.ca

**Abstract.** We study two-stage *robust* variants of combinatorial optimization problems like Steiner tree, Steiner forest, and uncapacitated facility location. The robust optimization problems, previously studied by Dhamdhere et al. [1], Golovin et al. [6], and Feige et al. [4], are two-stage planning problems in which the requirements are revealed after some decisions are taken in stage one. One has to then complete the solution, at a higher cost, to meet the given requirements. In the robust Steiner tree problem, for example, one buys some edges in stage one after which some terminals are revealed. In the second stage, one has to buy more edges, at a higher cost, to complete the stage one solution to build a Steiner tree on these terminals. The objective is to minimize the total cost under the worst-case scenario. In this paper, we focus on the case of *exponentially many* scenarios given implicitly. A scenario consists of any subset of $k$ terminals (for Steiner tree), or any subset of $k$ terminal-pairs (for Steiner forest), or any subset of $k$ clients (for facility location). We present the first constant-factor approximation algorithms for the robust Steiner tree and robust uncapacitated facility location problems. For the robust Steiner forest problem on trees and with uniform inflation, we present a constant approximation and show that the problem on general graphs and with two inflation factors is impossible to approximate within $O(\log^{1/2-\epsilon} n)$ factor, for any constant $\epsilon > 0$, unless NP has randomized quasi-polynomial time algorithms. Finally, we show APX-hardness of the robust min-cut problem (even with singleton-set scenarios), resolving an open question by [1] and [6].

## 1 Introduction

In a classical optimization problem, we are usually given a system with some known parameters and constraints and the goal is to find a feasible solution of minimum cost

⋆ Partially supported by NSF Award Grant number 072887.
⋆⋆ Part of this work was done when the author was at Microsoft research.
⋆⋆⋆ Supported by NSERC and an Alberta Ingenuity New Faculty award.

(or maximum profit) with respect to the constraints. These parameters and constraints, which heavily influence the optimum solution, are assumed to be precisely known. However, in reality, often it is very costly (or maybe impossible) to have an accurate picture about the values of the parameters or even the constraints of the optimization problem at hand at the time of planning. Two of the common approaches studied in the literature to address this uncertainty about future are referred to as *robust optimization* and *stochastic optimization*.

**Robust and Stochastic optimization.** Robust optimization has been studied in both decision theory [10] and Mathematical Programming [2] and deals with the uncertainty in data. In a typical data-robust model, we have a finite set of scenarios that can materialize and each scenario contains one possible set of data values. The goal is to find a solution that is good with respect to all or most scenarios. One example in this category is min-max regret, in which the goal is to minimize the maximum regret over all possible scenarios, where the regret of a scenario is defined as the difference between the cost of the solution in that scenario with respect to optimal solution for that scenario.

In Stochastic optimization, we are provided with a probability distribution on the possible scenarios and the goal is then to find a solution that minimizes the expected cost over this distribution. This approach is useful if we have a good idea about the probability distribution (which may be a strong requirement), and we have a repeated decision making framework. One particular version of stochastic optimization, that has attracted much attention in the last decade, is two-stage (or multi-stage) stochastic optimization, where the solution is built in two stages: in the first stage we have to decide to build a partial solution based on the probability distribution of possible scenarios. In stage two, once the actual scenario is revealed, we have to complete our partial solution to a feasible solution for the given scenario. There has been considerable amount of research focused on two-stage (or multi-stage) stochastic version of classical optimization problems such as set-cover, Steiner tree, vertex cover, facility location, cut problems, and other network design problems [11, 13, 7, 8] and efficient approximation algorithms have been developed for many of these problem. In some cases, the set of possible scenarios and the corresponding probabilities are given explicitly [11, 8], and some papers study a more general model in which the sets of possible scenarios are given implicitly as the product of a set of independent trials, or by an oracle [1, 7, 9, 13].

**Demand-robust optimization.** More recently, a new notion of robustness has been introduced by Dhamdhere et al. [1] which can be viewed as the worst-case analogue of the (two-stage) stochastic optimization problem. This model, called *demand-robust optimization* (and we simply call robust optimization in the rest of the paper), deals with both uncertainty in data as well as constraints of the problem. In a two-stage robust optimization problem, similar to a two-stage stochastic optimization, we are given a set of possible scenarios (which can be explicit or implicit) and the goal is to compute a solution in two stages while minimizing the maximum cost over all possible scenarios. The major difference of this model w.r.t. data-robust model is that each possible scenario might have a different set of constraints to be satisfied. For example, in the robust Steiner tree problem, we are given a graph $G = (V, E)$ with a cost function $c : E \rightarrow \mathbb{R}^+$ on the edges. In the second stage one of $m$ possible scenarios material-

izes; scenario $i$ consists of a set $S_i \subseteq V$ of terminals that need to be connected to each other. We also have an inflation factor $\lambda_i$ for edge costs. Each edge $e$ costs $c_e$ in the first stage and $\lambda_i \cdot c_e$ in the second stage if scenario $i$ materializes. Our goal is to select a subset of edges $E_1 \subseteq E$ in the first stage, and a set $E_2(i) \subseteq E$ in the second stage if scenario $i$ is revealed, so that $E_1 \cup E_2(i)$ is a feasible solution for the Steiner tree problem with terminal set $S_i$; the overall cost paid in scenario $i$ is $c(E_1) + \lambda_i \cdot c(E_2(i))$. The objective function is to minimize this cost over all possible scenarios, i.e., to minimize $c(E_1) + \max_i \lambda_i \cdot c(E_2(i))$. Since in this model of robustness, each scenario has a different requirement, it allows one to handle uncertainty in the set of constraints. It also provides a worst-case guarantee unlike only the expected-cost guarantee as in the two-stage stochastic optimization model. In a very recent work [4] authors consider a more general model of (two-stage) robust optimization in which scenarios are given implicitly, and therefore, one can have an exponentially large set of possible scenarios.

### 1.1 Previous work

We are aware of only three other papers [1, 6, 4] which have studied (demand) robust optimization. In [1, 6], authors consider the robust problems with polynomially-many explicitly given scenarios. They present constant factor approximation algorithms for robust versions of Steiner tree, vertex cover, and Facility location problems. They also give polylogarithmic approximation for robust min-cut and multi-cut. In the problem of robust min-cut, we are given a edge-weighted graph $G = (V, E)$, a source $s \in V$, and inflation factor $\lambda_i(e)$; scenario $i$ consist of a terminal $t_i \in V$. The goal is to find a subset $E_1 \subseteq E$ for stage one and $E_2(i) \subseteq E$ for stage two (if scenario $i$ arrives) so that $E_1 \cup E_2(i)$ is a $s, t_i$-cut. In [1], the authors present an $O(\log m)$-approximation where $m$ is the number of scenarios. This was improved to a $(1 + \sqrt{2})$-approximation in [6]. In the robust multi-cut, each scenario consists of a set of pairs of nodes that form a multi-cut problem. For this problem [1] present an $O(\log rm \cdot \log \log rm)$ where $m$ is the number of scenarios and $r$ is the maximum number of pairs in any scenario.

Feige et al. [4] consider covering problems (such as set cover) where the set of possible scenarios is given implicitly, and therefore can be exponentially large. For example, in the robust set cover problem, we are given a universe of elements $U = \{e_1, \ldots, e_m\}$ and a collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$, where $S_i \subseteq U$ and has cost $c(S_i)$, an inflation factor $\lambda$, and an integer $k$. Each scenario is a subset $U' \subset U$ of size $k$ that needs to be covered. We have to purchase a collection of sets $\mathcal{S}_1 \subseteq \mathcal{S}$ in stage one. Once a set $U'$ of elements is given in stage two, we have to purchase some (possibly none) other sets $\mathcal{S}_2(U') \subset \mathcal{S}$ where the cost of each set now is inflated by $\lambda$, so that $\mathcal{S}_1 \cup \mathcal{S}_2(U')$ is a set cover for the given set $U'$. The goal is to minimize the maximum total cost over all possible scenarios. Using an LP rounding method, they give a general framework for designing approximation algorithms for a class of robust covering problems using competitive algorithms for online variants of the problems. However, this framework does not apply to robust network design problems like robust Steiner tree, and gives logarithmic approximation for robust uncapacitated metric facility location problem.

### 1.2 Our results

The model we study in this paper is the (demand) robust optimization model with (possibly) implicit sets of scenarios (which can be exponentially many). We study Steiner tree, Steiner forest, uncapacitated facility location, and min-cut problems under this model and present some approximation algorithms and hardness results.

Specifically, we provide the first constant factor approximation algorithms for the (exponential scenarios) robust Steiner tree and robust facility location problems. Our algorithms are combinatorial in nature and are based on nice structural properties of the stage one solution of a near-optimum algorithm.

**Theorem 1.** *There exists a polynomial-time $5.55$-approximation algorithm for two-stage robust Steiner tree problem with a uniform inflation factor. Here a scenario consists of any $k$ terminals out of given terminals.*

**Theorem 2.** *There exists a polynomial-time $10$-approximation algorithm for robust uncapacitated facility location problem in which the inflation factor may depend on the facility. Here a scenario consists of any $k$ clients (perhaps co-located) out of given clients.*

We then present a constant approximation for robust Steiner forest problem when the underlying graph is a tree. Our algorithm is based on first solving a standard LP relaxation using a dynamic-programming based separation oracle and then rounding it to a near-optimum integral solution.

**Theorem 3.** *There exists a polynomial-time $3$-approximation algorithm for two-stage robust Steiner forest problem on trees with a uniform inflation factor. Here a scenario consists of any $k$ terminal-pairs out of given terminal-pairs.*

We next show that the robust Steiner forest problem on general graphs on $n$ vertices and with two inflation factors is impossible to approximate within a factor of $O(\log^{\frac{1}{2}-\epsilon} n)$, for any constant $\epsilon > 0$, unless NP has randomized quasi-polynomial time algorithms. We emphasize that our 3-approximation algorithm of Theorem 3 holds for exponentially many scenarios (i.e., $k$ source-sink pairs need to be connected in stage 2), and our hardness result holds for even polynomially many scenarios, in which we have to connect only *one source-sink pair* in the second stage. However in the hardness result, we work with general graphs and allow two possible values for the inflation factors on the edge-costs.

**Theorem 4.** *For any constant $\epsilon > 0$, there is no $O(\log^{\frac{1}{2}-\epsilon} n)$-approximation for robust Steiner forest on general graphs on $n$ vertices in which only one pair arrives in stage two, we have only two (distinct) edge costs and two (distinct) inflation factors, unless $NP$ has randomized quasi-polynomial time algorithms.*

Finally, we resolve an open question posed by Dhamdhere et al. [1] and Golovin et al. [6] about the hardness of the two-stage robust min-cut problem by proving the following theorem.

**Theorem 5.** *The two-stage robust min-cut problem is APX-hard even with a uniform inflation factor and which consists of a single source and three sinks.*

**Organization.** The remainder of the paper is organized as follows. In the next section, we present our 5.55-approximation algorithm for robust Steiner tree and then 3-approximation for Steiner forest on trees. Finally we present the proof of Theorem 5. Due to space limit, we postpone the proofs of Theorems 2 and 4 to the full version of our paper.

## 2 A constant approximation for robust Steiner tree problem

In this section we prove Theorem 1. Recall that the input to the Steiner tree problem is an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}^+$, and a subset $T \subseteq V$ called "terminals". The objective is to find a connected subgraph $H$ that includes all the terminals $T$ and has minimum cost $c(H) := \sum_{e \in H} c_e$.

In the robust version of the Steiner tree problem, the input also contains an integer $k$ and a real number $\lambda \geq 1$. There are two stages. In the first stage the algorithm has to identify a subset $E_1 \subseteq E$ of edges to buy. In the second stage, the cost of each edge in $E \setminus E_1$ increases by a factor of $\lambda$ and a subset $T' \subseteq T$ of at most $k$ terminals is revealed. We refer to $T'$ as a "scenario". The algorithm, in the second stage, has to augment the solution $E_1$ by buying edges $E_2(T')$ so that the resulting graph $E_1 \cup E_2(T')$ includes a Steiner tree on terminals $T'$. The choice of edges $E_2(T')$ is allowed to depend on the subset $T'$. The overall cost of this solution is thus $\sum_{e \in E_1} c_e + \lambda \cdot \sum_{e \in E_2(T')} c_e$. The objective is to minimize the maximum overall cost over all scenarios, i.e., to minimize

$$\sum_{e \in E_1} c_e + \max_{T' \subseteq T, |T'| \leq k} \lambda \cdot \sum_{e \in E_2(T')} c_e.$$

The edge-costs $c_e$ induce a shortest-path metric on the vertices $V$: for any two vertices $u, v \in V$, we use $d_G(u, v)$ to denote the length of the shortest path between $u$ and $v$, under costs $c_e$ in graph $G$.

### 2.1 The algorithm

Let $E_1^*$ and $E_2^*(T')$ be the set of edges the optimum buys in the first stage and the second stage for scenario $T'$ respectively. Let $\mathrm{OPT} = \mathrm{OPT}_1 + \lambda \cdot \mathrm{OPT}_2$ be the overall cost of the optimum, where $\mathrm{OPT}_1 = \sum_{e \in E_1^*} c_e$ is its cost in the first stage and $\mathrm{OPT}_2 = \max_{T' \subseteq T, |T'| \leq k} \sum_{e \in E_2^*(T')} c_e$ is the maximum cost in the second stage divided by $\lambda$.

**First stage.** Our algorithm, in the first stage, guesses (an upper bound on) the value of $\mathrm{OPT}_2$.[5] It then computes a subset of terminals $\mathcal{C} = \{c_1, c_2, \ldots, c_p\} \subseteq T$ called "centers" and an assignment $\pi : T \to \mathcal{C}$ that satisfy:

[5] The algorithm in fact tries all guesses of $\mathrm{OPT}_2$ that are powers of $(1 + \epsilon)$, for a small constant $\epsilon > 0$, and takes the cheapest solution for any of these guesses. We also point out that we can, in polynomial time, estimate the cost of our solution for a given guess on $\mathrm{OPT}_2$. That is, in polynomial-time, we can find a scenario that maximizes the cost of the solution (or its upper bound proved). To simplify the presentation, we assume that the guess on $\mathrm{OPT}_2$ is exact.

– The centers are far apart: $d_G(c_i, c_j) > \frac{r \cdot \mathrm{OPT}_2}{k}$ for all $i \neq j$, and
– Each terminal is close to its assigned center: $d_G(t, \pi(t)) \leq \frac{r \cdot \mathrm{OPT}_2}{k}$ for all $t \in T$,

where $r > 1$ is a constant to be determined later. Such a clustering can be computed as follows. Pick any terminal and name it $c_1$. Assign all terminals within a distance of $\frac{r \cdot \mathrm{OPT}_2}{k}$ from $c_1$ to $c_1$ and remove these terminals. Pick any one of the remaining terminals and name it $c_2$, and so on.

The algorithm then computes an approximate minimum-cost Steiner tree $\mathcal{T}$ in $G$ on the centers $\mathcal{C}$ under the costs $c_e$. Currently, the best known polynomial-time algorithm for the Steiner tree problem is $\gamma$-approximate, where $\gamma < 1.55$ [12]. The algorithm buys the edges in the Steiner tree in the first stage.

**Second stage.** In the second stage, a subset $T'$ of at most $k$ terminals is revealed. The algorithm, in the second stage, buys the shortest path from each terminal $t \in T'$ to its assigned center $\pi(t)$. It is easy to see that the algorithm computes a feasible solution to the problem.

### 2.2 The analysis

We first introduce the notion of a "ball" of certain radius around a vertex in a graph. Consider the graph $G = (V, E)$ with edge-costs $c_e$. We think of each edge $e$ as a continuous interval of length $c_e$. For a vertex $v$ and a radius $R > 0$, let $B_G(v, R)$ denote, intuitively speaking, the "moat" of radius $R$ around $v$. More precisely, $B(v, R)$ contains

– all the vertices $u$ such that $d_G(u, v) \leq R$,
– all edges $e = (u, w)$ such that $d_G(u, v) \leq R$ and $d_G(w, v) \leq R$, and
– for the edges $e = (u, w)$ such that $d_G(u, v) \leq R$ and $d_G(w, v) > R$, the sub-interval of edge $e$ of length $R - d_G(u, v)$ adjacent to vertex $u$.

Note that since $d_G(c_i, c_j) > \frac{r \cdot \mathrm{OPT}_2}{k}$ for any two distinct centers in $\mathcal{C}$, the balls $B_G(c_i, \frac{r \cdot \mathrm{OPT}_2}{2k})$ and $B_G(c_j, \frac{r \cdot \mathrm{OPT}_2}{2k})$ are disjoint. It is easy to see that the algorithm pays at most $\lambda \cdot r \cdot \mathrm{OPT}_2$ in the second stage. This holds since the distance of any terminal to its assigned center is at most $\frac{r \cdot \mathrm{OPT}_2}{k}$. Since at most $k$ terminals need to be connected to their centers, the total cost of these connections is at most $\lambda \cdot k \cdot \frac{r \cdot \mathrm{OPT}_2}{k}$. We now bound the cost of the algorithm in stage one using the following lemma.

**Lemma 1.** *Assuming $r > 4$, there exists a Steiner tree on centers $\mathcal{C}$ in $G$ that has cost at most $\frac{r}{r-4} \cdot \mathrm{OPT}_1 + \mathrm{OPT}_2$.*

*Proof.* Recall that $E_1^*$ is the set of edges optimum buys in stage one and $\mathrm{OPT}_1 = \sum_{e \in E_1^*} c_e$. Let $H$ be a graph obtained from $G$ by shrinking the edges in $E_1^*$. We now perform another clustering of the centers $\mathcal{C}$ in the shortest-path metric on $\mathcal{C}$ induced by the graph $H$. For centers $c_i, c_j \in \mathcal{C}$, let $d_H(c_i, c_j)$ denote the shortest-path length under lengths $c_e$ in $H$. We identify a subset of centers $\mathcal{L} = \{l_1, l_2, \ldots, l_t\}$ called "leaders" and a mapping $\phi : \mathcal{C} \to \mathcal{L}$ such that

– The leaders are far apart: $d_H(l_i, l_j) > 2\text{OPT}_2/k$ for all $i \neq j$, and
– Each center is close to its mapped leader: $d_H(c, \phi(c)) \leq 2\text{OPT}_2/k$ for all centers $c \in \mathcal{C}$.

Such a clustering can be computed as follows. Pick any center and name it $l_1$. For all centers $c \in \mathcal{C}$ with $d_H(c, l_1) \leq 2\text{OPT}_2/k$, define $\phi(c) = l_1$. Remove all such centers from $\mathcal{C}$ and repeat.

Analogous to $B_G(v, R)$, we use $B_H(v, R)$ to denote the ball of radius $R$ centered at $v$ in the graph $H$ with length $c_e$ for $e \in H$. Note that the balls of radii $\frac{\text{OPT}_2}{k}$ around the leaders in $\mathcal{L}$ are disjoint in $H$.

*Claim.* The following inequality holds: $|\mathcal{L}| \leq k$.

*Proof.* Assume on the contrary that $|\mathcal{L}| > k$ and let $T' \subseteq \mathcal{L}$ be any subset of size $k+1$. Consider the scenario $T'$. Since even after shrinking the edges in $E_1^*$ that optimum bought in the first stage, the balls of radii $\frac{\text{OPT}_2}{k}$ centered at the centers in $T'$ in the graph $H$ are disjoint. Therefore the minimum Steiner tree on $T'$ in $H$ has cost more than $\text{OPT}_2$. This is a contradiction since the optimum pays at most $\lambda \cdot \text{OPT}_2$ in the second phase to connect all the centers in $T'$ after shrinking the edges in $E_1^*$. Thus the claim holds.

Since $|\mathcal{L}| \leq k$, we now consider scenario $\mathcal{L}$. There exists a Steiner tree $E_{\mathcal{L}}^*$ on $\mathcal{L}$ in $H$ with cost at most $\text{OPT}_2$. Thus $E_1^* \cup E_{\mathcal{L}}^*$ has cost at most $\text{OPT}_1 + \text{OPT}_2$ and contains a Steiner tree on $\mathcal{L}$ in $G$. We now show how to extend this into a subgraph with low cost and which contains a Steiner tree on $\mathcal{C}$ in $G$.

Now recall that the balls of radii $\frac{r \cdot \text{OPT}_2}{2k}$ around the centers $\mathcal{C}$ are disjoint in $G$. Note however that $d_H(c, \phi(c)) \leq \frac{2\text{OPT}_2}{k}$ for all centers $c \in \mathcal{C}$. Thus at least $\frac{r \cdot \text{OPT}_2}{2k} - \frac{2\text{OPT}_2}{k} = \left(\frac{r}{2} - 2\right) \cdot \frac{\text{OPT}_2}{k}$ cost of $E_1^*$ must lie inside the ball of radius $\frac{r \cdot \text{OPT}_2}{2k}$ around each center $c \in \mathcal{C}$. We can thus extend the subgraph $E_1^* \cup E_{\mathcal{L}}^*$ by adding shortest paths from each $c$ to $\phi(c)$ in $H$ and charge this additional cost to the contribution of $E_1^*$ in the respective balls around centers $c \in \mathcal{C}$. The resulting subgraph clearly contains a Steiner tree on $\mathcal{C}$ in $G$. The overall cost of this subgraph is thus at most $\text{OPT}_1 + \text{OPT}_2 + \frac{2}{\frac{r}{2}-2} \cdot \text{OPT}_1 = \frac{r}{r-4} \cdot \text{OPT}_1 + \text{OPT}_2$. We remind the reader that $OPT_2$ is the cost of computing a Steiner tree on the leaders. Hence the proof.

Since we use a $\gamma$-approximation algorithm to compute a Steiner tree in stage one, the overall cost of stage one is at most $\frac{\gamma \cdot r}{r-4} \cdot \text{OPT}_1 + \gamma \cdot \text{OPT}_2$. Combining this with the second stage cost, the overall cost of our solution is:

$$\frac{\gamma \cdot r}{r-4} \cdot \text{OPT}_1 + (\gamma + \lambda r) \cdot \text{OPT}_2. \tag{1}$$

A trivial strategy for solving the robust Steiner tree is to select nothing in stage 1 and make all the selections in stage 2. Given that every edge is inflated by $\lambda$ and we use a $\gamma$-approximation for Steiner tree, this strategy will have an approximation factor of $\lambda \cdot \gamma$. Using the best known approximation algorithm for Steiner tree [12], which has approximation 1.55, we get a $1.55\lambda$-approximation. For values of $\lambda \leq 3.51$ we use this

trivial strategy which gives an approximation factor of $5.45$. For values of $\lambda > 3.51$ we use the above algorithm with parameter $r$ defined below.

Let $r = r^*$ to be the solution of: $\frac{\gamma \cdot r}{r-4} = \frac{\gamma}{\lambda} + r$. Then the two factors in front of $\text{OPT}_1$ and $\text{OPT}_2$ in the ratio of our algorithm calculated in Equation (1) become equal at $r = r^* = \frac{\gamma\lambda - \gamma + 4\lambda + \sqrt{\gamma^2\lambda^2 - 2\gamma^2\lambda + 8\gamma\lambda^2 + \gamma^2 + 8\gamma\lambda + 16\lambda^2}}{2\lambda}$. Therefore, for $r = r^*$ and with $\gamma = 1.55$, the ratio of our algorithm becomes: $\frac{5.55\lambda - 1.55 + \sqrt{30.8025\lambda^2 + 7.595\lambda + 2.4025}}{2\lambda}$. It can be verified that this expression is upper bounded by $5.55$ (it has a limit of $5.55$). Thus, for values of $\lambda > 3.51$, by choosing $r = r^*$, the ratio of our algorithm presented will be at most $5.55$ and for smaller values of $\lambda$ we use the trivial strategy which has ratio at most $5.55$ as well. This completes the proof of Theorem 1.

## 3 A constant approximation for robust Steiner forest problem on trees

In this section, we prove Theorem 3. The input to the Steiner forest problem is an undirected graph $G = (V, E)$ with non-negative edge-costs $c_e$. We are also given a set of terminal-pairs $T \subseteq V \times V$. Similar to the robust Steiner tree problem, the input also has an integer $k$ and a real number $\lambda \geq 1$. There are two stages. In the first stage the algorithm has to identify a subset $E_1 \subseteq E$ of edges to buy. In the second stage, the cost of each edge in $E \setminus E_1$ increases by a factor of $\lambda$ and a subset $T' \subseteq T$ of at most $k$ terminal-pairs is revealed. We refer to $T'$ as a "scenario". The algorithm, in the second stage, has to augment the solution $E_1$ by buying edges $E_2(T')$ so that the resulting graph $E_1 \cup E_2(T')$ includes a Steiner forest on terminal-pairs $T'$, i.e., $E_1 \cup E_2(T')$ contains a path between each terminal-pair in $T'$. The objective is to minimize the maximum overall cost over all scenarios, i.e., to minimize $\sum_{e \in E_1} c_e + \max_{T' \subseteq T, |T'| \leq k} \lambda \cdot \sum_{e \in E_2(T')} c_e$.

In this section, we focus our attention on the special case when the graph $G$ is a tree $\mathcal{T}$ with edge-costs $c_e$. Let $d_{\mathcal{T}}(u, v)$ denotes the length of the unique path between $u$ and $v$ in $\mathcal{T}$.

For a scenario $T' \subseteq T$ of at most $k$ terminal pairs, let $E(T')$ denote the union of the unique paths between the terminal-pairs in $T'$. We now consider the following integer linear programming formulation of our problem. Let $x_e \in \{0, 1\}$ denote an integer variable that takes value 1 if edge $e$ is picked in stage one, and 0 otherwise. Note that any edge $e \in E(T')$ is picked in stage two for scenario $T'$ if and only if $x_e = 0$. Thus the stage two cost for scenario $T'$ is $\lambda \cdot \sum_{e \in E(T')} c_e \cdot (1 - x_e)$. It is now easy to see that the following integer program is identical to our problem.

$$\begin{array}{lll} \min & \sum_e c_e \cdot x_e + \lambda \cdot C_2 & \\ s.t. & \sum_{e \in E(T')} c_e \cdot (1 - x_e) \leq C_2 & \forall \text{ scenarios } T' \\ & x_e \in \{0, 1\} & \forall \text{ edges } e \\ & C_2 \geq 0 & \end{array} \tag{2}$$

A linear relaxation of the above integer program is obtained by replacing the integrality constraints $x_e \in \{0, 1\}$ by $0 \leq x_e \leq 1$ for each edge $e$. This linear program has

polynomially many variables and exponentially many constraints. We now give an approximate separation oracle for this program and solve it using the ellipsoid algorithm.

**The separation oracle:** The separation oracle for the above linear program needs to solve the following problem: given $x_e \in [0,1]$ for each edge $e$, find a scenario $T'$ such that $\sum_{e \in E(T')} y_e$ is maximized, where $y_e = c_e \cdot (1 - x_e)$. Recall that a scenario $T'$ consists of at most $k$ terminal pairs from $T$ and $E(T')$ denotes the union of the paths between the terminal-pairs in $T'$. Thus the separation oracle can be viewed as the following problem. Given a set of paths $T$ on a tree $\mathcal{T}$ with edge-profits $y_e \geq 0$, find a subset of at most $k$ paths that maximizes the total profit in the union of the paths.

We now give a dynamic programming based 2-approximation algorithm for the above problem. Pick any vertex $r \in \mathcal{T}$ in the tree to be the "root" and imagine that $\mathcal{T}$ is hung from $r$. Thus we get a natural ancestor-descendant relation between the vertices of $\mathcal{T}$: vertex $u$ is called an ancestor of vertex $v$ if $u$ lies on the unique path between $v$ and root $r$; and vertex $v$ is called a descendant of vertex $u$ if $u$ is an ancestor of $v$.

Now any path $p \in T$ can be expressed as a disjoint union of two paths $p_1$ and $p_2$ such that the end-points of both $p_1$ and $p_2$ satisfy the ancestor-descendant relation. We call such paths "up-paths". We now solve our profit maximization problem on this collection of up-paths. It is easy to see that the maximum profit of at most $k$ up-paths obtained in a manner given above is at least *half* of the maximum profit of at most $k$ paths in the original problem.

The maximum profit collection of $k$ up-paths can be computed by dynamic programming as follows. In what follows, we say that a path $p$ "covers" an edge $e$ if $e \in p$. For every vertex $v \in \mathcal{T}$, let $\mathcal{T}_v$ be the subtree rooted at $v$. For each $v \in \mathcal{T}$, for each of its ancestors $u \in \mathcal{T}$, and for each integer $0 \leq l \leq k$, let $\mathtt{p}(v,u,l)$ denote the maximum profit that can be accrued in the subtree $\mathcal{T}_v$ by at most $l$ paths that together cover each edge on the path between $v$ and $u$ in $\mathcal{T}$. We compute the values of $\mathtt{p}$ from leaves up. Below, we only consider triplets $(v,u,l)$ where $u$ is an ancestor of $v$ (possibly, $u = v$) and $l$ is an integer (possibly $l < 0$ to simplify the description). We first initialize the values of $\mathtt{p}(v,u,l)$ to $-\infty$.

To simplify the exposition, we assume that each vertex has at most two children. This assumption can be made without loss of generality as described below. Consider a vertex $v$ with $c > 2$ children $v_1, \ldots, v_c$. We expand $v$ into a binary tree with $c$ leaves corresponding to its $c$ children. The profit of any new edge on this binary tree is set to zero. The original paths can be extended naturally. It is easy to see that the maximum achievable profit in the new instance is same as that in the original instance.

Now for the base case of the dynamic program, we set $\mathtt{p}(v,u,l)$ where $v$ is a leaf and $l \geq 0$ to 0. Now consider any internal vertex $v \in \mathcal{T}$ and assume that we have already computed (and stored) the values of $\mathtt{p}(v',u,l)$ for all children $v'$ of $v$.

We first explain how to compute $\mathtt{p}(v,u,l)$ when $v$ has only one child $v_1$. Let $e = (v,v_1)$. If $u = v$, we set $\mathtt{p}(v,v,l) = \max\{\mathtt{p}(v_1,v_1,l), \mathtt{p}(v_1,v,l) + y_e, \mathtt{p}(v_1,v_1,l-1) + y_e\}$. For $u \neq v$, we let $\mathtt{p}(v,u,l) = \max\{\mathtt{p}(v_1,u,l) + y_e, \max_{u'} \mathtt{p}(v_1,u',l-1) + y_e\}$ where the maximum is taken over vertices $u'$ on the path between $v_1$ and $u$ such that there is a path between $u'$ and one of its ancestor that covers the path between $u'$ and $u$.

Now we explain how to compute $\mathtt{p}(v,u,l)$ when $v$ has exactly two children $v_1$ and $v_2$. First consider the case when $u = v$. Let $e_1 = (v,v_1)$ and $e_2 = (v,v_2)$.

We set $\mathrm{p}(v, v, l)$ to be the maximum of the following different ways of accruing a profit. Below the maximum is taken over $l'$ where $0 \leq l' \leq l$. The maximum profit without covering edges $e_1$ and $e_2$ is $\max_{l'}(\mathrm{p}(v_1, v_1, l') + \mathrm{p}(v_2, v_2, l - l'))$. The maximum profit covering $e_1$ but not $e_2$ is $\max_{l'}(\max\{\mathrm{p}(v_1, v, l'), \mathrm{p}(v_1, v_1, l' - 1)\} + y_{e_1} + \mathrm{p}(v_2, v_2, l - l'))$. Similarly, the maximum profit covering $e_2$ but not $e_1$ is maximum of $\max_{l'}(\max\{\mathrm{p}(v_2, v, l'), \mathrm{p}(v_2, v_2, l' - 1)\} + y_{e_2} + \mathrm{p}(v_1, v_1, l - l'))$. Similarly, the maximum profit covering both $e_1$ and $e_2$ is $\max_{l'}(\max\{\mathrm{p}(v_1, v, l'), \mathrm{p}(v_1, v_1, l' - 1)\} + y_{e_1} + \max\{\mathrm{p}(v_2, v, l - l'), \mathrm{p}(v_2, v_2, l - l' - 1)\} + y_{e_2})$.

Now consider the case when $u \neq v$. Again let $e_1 = (v, v_1)$ and $e_2 = (v, v_2)$. We set $\mathrm{p}(v, u, l)$ to be the maximum of the following different ways of accruing a profit. Below the maximum is taken over $l'$ where $0 \leq l' \leq l$. The maximum profit without covering edges $e_1$ and $e_2$ is $\max_{l'}(\mathrm{p}(v_1, v_1, l') + \mathrm{p}(v_2, v_2, (l - l'') - l'))$ if $l''$ is the minimum number of paths needed to cover the edges on path between $v$ and $u$. The maximum profit covering $e_1$ but not $e_2$ is $\max_{l'}(\mathrm{p}(v_1, u, l') + y_{e_1} + \mathrm{p}(v_2, v_2, l - l'))$. Similarly, the maximum profit covering $e_2$ but not $e_1$ is maximum of $\max_{l'}(\mathrm{p}(v_2, u, l') + y_{e_2} + \mathrm{p}(v_1, v_1, l - l'))$. Similarly, the maximum profit covering both $e_1$ and $e_2$ is the maximum of $\max_{l'}(\mathrm{p}(v_1, u, l') + y_{e_1} + \mathrm{p}(v_2, v, l - l') + y_{e_2})$ and $\max_{l'}(\mathrm{p}(v_2, u, l') + y_{e_2} + \mathrm{p}(v_1, v, l - l') + y_{e_1})$.

**The rounding:** Since there is a 2-approximation to the separation oracle, we can compute, using the ellipsoid algorithm, a feasible solution $(\{x_e^*\}, C_2^*)$ to (2) such that $\sum_e c_e \cdot x_e^* + \lambda \cdot \frac{C_2^*}{2} \leq \text{OPT}^* \leq \sum_e c_e \cdot x_e^* + \lambda \cdot C_2^*$ where $\text{OPT}^*$ denotes the cost of the optimum fractional solution to (2). We round this solution to an integral feasible solution to the Steiner forest problem on trees as follows: pick $e \in E$ in stage one if and only if $x_e^* \geq \frac{1}{3}$. In stage two, given a scenario $T'$, pick the remaining edges in $E(T')$ to form a feasible solution.

The cost of the stage one of our solution is $\sum_{e:x_e^* \geq 1/3} c_e \leq 3 \sum_e c_e \cdot x_e^*$. The stage two cost of scenario $T'$ is $\lambda \cdot \sum_{e:x_e^* < 1/3} c_e \leq \frac{3}{2}\lambda \cdot \sum_{e:x_e^* < 1/3} c_e \cdot (1 - x_e^*) \leq \frac{3}{2}\lambda \cdot C_2^*$. Thus the overall cost of our solution is at most $3 \sum_e c_e \cdot x_e^* + \frac{3}{2}\lambda \cdot C_2^* \leq 3 \cdot \text{OPT}^*$. Since $\text{OPT}^*$ is at most the optimum integral solution, our algorithm is a 3-approximation.

## 4 APX-hardness of the robust min-cut problem

In this section, we prove Theorem 5. In the robust min-cut problem we are given an undirected graph $G = (V, E)$ with edge-costs $c_e \geq 0$, a source $s \in V$, a collection of sinks $T \subseteq V$, and a inflation factor $\lambda_i \geq 1$ for every $t_i \in T$. There are two stages in the algorithm. The algorithm has to choose edges $E_0 \subseteq E$ in the first stage. We are then given a *single* sink $t_i \in T$. We call $t_i$ a "scenario". In such a scenario, the cost of each edge $e \in E \setminus E_0$ becomes $\lambda_i \cdot c_e$. The algorithm, then, has to pick edges $E_1(t_i) \subseteq E \setminus E_0$ such that $s$ and $t_i$ are not connected in the graph $(V, E \setminus \{E_0 \cup E_1(t_i)\})$. The objective is to minimize the maximum cost of the solution under any scenario: $c(E_0) + \max_{t_i \in T} \lambda_i \cdot c(E_1(t_i))$, where $c(X) = \sum_{e \in X} c_e$ for $X \subseteq E$.

In [11], the authors give $(1 + \sqrt{2})$-approximation algorithm for this problem and pose as an open question to determine if this problem is NP-hard. We show that the special case, in which there are only *three* sinks and all inflation factors $\lambda_i$ are equal,

is already APX-hard. We reduce the APX-hard problem of finding *multi-way cut* to our problem. The input to the multi-way cut problem is an undirected graph $G = (V, E)$ with edge-costs $c_e \geq 0$ and a collection $T \subseteq V$ of terminals. The problem is to find a subset $E' \subseteq E$ of minimum total cost $c(E')$ such that all terminals in $T$ lie in different connected components in $(V, E \setminus E')$. In [5] the following theorem is proved.

**Theorem 6.** [5] *There exists a universal constant $\alpha > 0$, value of which is known, such that given an instance of the multi-way cut problem on 3 terminals, it is NP-hard to distinguish between the following cases: (i)"yes-instance": there exists a multi-way cut of cost at most 1, or (ii)"no-instance": all multi-way cuts have cost at least $1 + \alpha$.*

Given an instance of the multi-way cut problem $\mathcal{I} = \{G = (V, E), \{c_e\}, T = \{t_1, t_2, t_3\}\}$, we construct a new graph $G'$ from $G$ by adding a new vertex $s$ and edges $e_1 = (s, t_1), e_2 = (s, t_2), e_3 = (s, t_3)$. We let $\lambda = 2$. In the instance for the robust min-cut problem, $s$ serves as a source, $T$ serves as a collection of terminals, and the edge-costs as given by $c_e$ for $e \in E$ and $c_{e_1} = c_{e_2} = c_{e_3} = 1 + \alpha$, where $\alpha$ is the constant from Theorem 6. Let $\beta = 1 + \alpha$.

**Lemma 2.** *If $\mathcal{I}$ is a yes-instance then the optimum cost of the robust min-cut is at most $1 + 2\beta$.*

*Proof.* Let $E^*$ be the minimum multi-way cut in $G$. We pick $E^*$ in stage one. Then given any terminal $t_i \in T$ as a scenario, we pick the edge $e_i$ in stage two. This clearly forms a feasible solution with cost $c(E^*) + \lambda \cdot \beta \leq 1 + 2\beta$.

**Lemma 3.** *If $\mathcal{I}$ is a no-instance then the optimum cost of the robust min-cut is at least $\min\{3\beta, 1 + 2\beta + \alpha\}$.*

*Proof.* Fix an optimum algorithm, say OPT. We consider four cases depending upon whether OPT picks zero, one, two, or three of the edges $e_1, e_2, e_3$ in stage one. If OPT picks exactly one edge, say $e_1$, in stage one, we consider scenario $t_2$. Since OPT has to pick $e_2$ in stage two for this scenario, the overall cost is at least $c_{e_1} + \lambda \cdot c_{e_2} = \beta + 2\beta = 3\beta$. If OPT picks exactly two edges, say $\{e_1, e_2\}$, in stage one, we consider scenario $t_3$. Since OPT has to pick $e_3$ in stage two for this scenario, the overall cost is at least $2\beta + \lambda \cdot \beta = 4\beta$. Similarly, if OPT picks three edges in stage one, its cost is at least $3\beta$.

Now consider the case where OPT does not pick any edge out of $e_1, e_2, e_3$ in stage one. Let $E_0$ be the set of edges OPT picks in stage one. Let $H = (V, E \setminus E_0)$. Let $E_{123} \subseteq E \setminus E_0$ the minimum multi-way cut separating $t_1, t_2, t_3$ in $H$. Note that $c(E_0) + c(E_{123}) \geq 1 + \alpha$ and hence $c(E_{123}) \geq 1 + \alpha - c(E_0)$. For $i = 1, 2, 3$, let $E_i$ denote the minimum cut separating $t_i$ from the other two terminals in $H$. Note that each of $E_1 \cup E_2$, $E_2 \cup E_3$, and $E_3 \cup E_1$ form a multi-way cut separating the terminals in $H$. Therefore, $c(E_1) + c(E_2) \geq c(E_{123})$, $c(E_2) + c(E_3) \geq c(E_{123})$, and $c(E_3) + c(E_1) \geq c(E_{123})$. Thus $c(E_1) + c(E_2) + c(E_3) \geq \frac{3}{2} \cdot c(E_{123})$ and hence $\max_i c(E_i) \geq c(E_{123})/2 \geq (1 + \alpha - c(E_0))/2$.

Without loss of generality, let $c(E_1) = \max_i c(E_i)$. Now consider scenario $t_1$. In stage two, OPT must pick edge $e_1$. Moreover OPT either picks a cut separating $t_1$ from the other terminals in $H$ or picks at least one edge out of $e_2, e_3$. If OPT picks a cut, its overall cost is at least $c(E_0) + \lambda \cdot c_{e_1} + \lambda c(E_1) \geq c(E_0) + 2\beta + 2 \cdot (1 +$

$\alpha - c(E_0))/2 = 2\beta + 1 + \alpha$. In the other case, the overall cost of OPT is at least $c(E_0) + \lambda \cdot c_{e_1} + \lambda \min\{c_{e_2}, c_{e_3}\} \geq 4\beta$. This completes the proof.

Since $\beta = 1 + \alpha$, we get that the ratio of costs of the robust min-cuts in a yes-instance and a no-instance is at least $\frac{3+3\alpha}{3+2\alpha}$. This completes the proof of Theorem 5.

## Acknowledgments

We thank Viswanath Nagarajan for pointing out an error in an earlier draft of the paper.

## References

1. K. Dhamdhere, V. Goyal, R. Ravi, and M. Singh, *How to pay, come what may: Approximation Algorithms for Demand-Robust Covering Problems*, In Proc. of 46th IEEE FOCS, 2005.
2. G. B. Dantzig, *Linear programming under uncertainty*, Management Sci., 1:197-206, 1955.
3. J. Fakcharoenphol, S. Rao, and K. Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, J. Comput. Syst. Sci. 69(3): 485–497, 2004.
4. U. Feige, K. Jain, M. Mahdian, and V. Mirrokni, *Robust Combinatorial Optimization with Exponential Scenarios*, In Proc. of IPCO 2007.
5. E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour, and. M Yannakakis. The Complexity of Multiterminal Cuts. *SIAM J. Comput.* 23(4): 864-894 1994
6. D. Golovin, V. Goyal, and R. Ravi, *Pay Today for a Rainy Day: Improved Approximation Algorithms for Demand-Robust Min-Cut and Shortest Path Problems,* In Proc. of STACS ,206-217,2006
7. A. Gupta, M. Pál, R. Ravi, and A. Sinha, *Boosted sampling: approximation algorithms for stochastic optimization*, In Proc. of 36th ACM STOC, 2004.
8. A. Gupta, R. Ravi, and A. Sinha, *An edge in time Saves nine: LP Rounding Approximation Algorithms for Stochastic Network Design*, In Proc. of 45th IEEE FOCS, 2004.
9. N. Immorlica, D. Karger, M. Minkoff, and V. Mirrokni, *On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems*, In Proc. of SODA 2004.
10. J. W. Milnor, *Games against nature*, in R. M. Thrall, C. H. Coomb, and R. L. Davis, editors, Decision Processes. Wiley.
11. R. Ravi and A. Sinha, *Hedging uncertainty: Approximation algorithms for stochastic optimization problems*, In Proc. of IPCO 2004.
12. G. Robins and A. Zelikovsky, *Improved Steiner tree approximation in graphs*, In Proc. of SODA 2000, 770-779.
13. D. Shmoys and C. Swamy, *Stochastic optimization is (almost) as easy as deterministic optimization*, In Proc. of 45th IEEE FOCS 2004.