Two-stage Robust Network Design with Exponential Scenarios*

Rohit Khandekar† Guy Kortsarz‡ Vahab Mirrokni§ Mohammad R. Salavatipour¶

Abstract

We study two-stage *robust* variants of combinatorial optimization problems on undirected graphs, like Steiner tree, Steiner forest, and uncapacitated facility location. Robust optimization problems, previously studied by Dhamdhere et al. [3], Golovin et al. [9], and Feige et al. [7], are two-stage planning problems in which the requirements are revealed after some decisions are taken in Stage 1. One has to then complete the solution, at a higher cost, to meet the given requirements. In the robust $k-$Steiner tree problem, for example, one buys some edges in Stage 1. Then k terminals are revealed in Stage 2 and one has to buy more edges, at a higher cost, to complete the Stage 1 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 $k-$Steiner tree), or any subset of k terminal-pairs (for $k-$Steiner forest), or any subset of k clients (for facility location). Feige et al. [7] give an LP-based general framework for approximation algorithms for a class of two stage robust problems. Their framework cannot be used for network design problems like $k-$Steiner tree (see later elaboration). Their framework can be used for the robust facility location problem, but gives only a logarithmic approximation.

We present the first constant-factor approximation algorithms for the robust $k-$Steiner tree (with exponential number of scenarios) and robust uncapacitated facility location problems. Our algorithms are combinatorial and are based on *guessing* the optimum cost and clustering to aggregate nearby vertices. For the robust $k-$Steiner forest problem on trees and with uniform multiplicative increase factor for Stage 2 (also known as inflation), we present a constant approximation. We show APX-hardness of the robust min-cut problem (even with singleton-set scenarios), resolving an open question of [3] and [9].

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 (or maximum profit) with respect to the constraints. Often these parameters and constraints, which heavily influence the optimal solution, are assumed to be precisely known. However, in reality, often it is very costly (or maybe impossible) to have an accurate picture of the values of the parameters or even the constraints of the optimization problem at the time of planning. There are two common approaches studied in the literature to address this uncertainty. One is the *robust optimization*

*A preliminary version of this paper appeared in the Proceedings of 16th Annual European Symposium on Algorithms (ESA) 2008.

†IBM T.J.Watson research center. email: rkhandekar@gmail.com.

‡Department of Computer Science, Rutgers University-Camden. Currently visiting IBM Research at Yorktown Heights. Partially supported by NSF Award Grant number 072887. email: guyk@crab.rutgers.edu.

§Google Research, New York, USA. email: mirrokni@gmail.com

¶Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. email: mreza@cs.ualberta.ca. Supported by NSERC and an Alberta Ingenuity New Faculty award.

and the second is *stochastic optimization*.

Robust optimization. This has been studied in both decision theory [14] and Mathematical Programming [5]. 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 absolute robustness or min-max, where the goal is to find a solution such that the maximum cost over all possible scenarios is minimized. Another example is *min-max regret*, in which the goal is to minimize the maximum regret over all possible scenarios. Here, the regret of a scenario is defined as the value of the solution for that scenario minus the optimal solution cost for that scenario.

Stochastic optimization. In this model, we are provided with a probability distribution on the possible scenarios and the goal is 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 build a partial solution based on the probability distribution of possible scenarios. In Stage 2, 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 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 [15, 18, 10, 11]. Efficient approximation algorithms have also been developed for many of these problem. In some cases, the set of possible scenarios and the corresponding probabilities are given explicitly [15, 11], and some papers study a more general model in which the sets of possible scenarios are given implicitly (rather than explicitly) as the product of a set of independent trials, or by an oracle [3, 10, 13, 18].

Demand-robust optimization. More recently, a new notion of robustness has been introduced by Dhamdhere et al. [3] 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 uncertainty in both data as well as the constraints of the problem. To see the difference, as an example, consider two different formulations of the shortest-path problem from a root node r: in the data-robust model we know the input graph as well as the other end-point s to which we like to find a shortest path from r. The uncertainty is on the values of the costs of the edges. In a typical demand-robust version the other end-point of the path is not known in advance; instead each potential node is given as a possible scenario. 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 two-stage robust Steiner tree problem, we are given a graph $G = (V, E)$ with a cost function $c : E \to \mathbb{R}^+$ on the edges. In the second stage one of m possible scenarios materializes; 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 λ_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 $c(E_1) + \max_i \lambda_i \cdot c(E_2(i))$. In the robust $k-$Steiner tree problem, each set S_i is a subset of

size k of terminals.

This model of robustness allows one to handle uncertainty in the input and provides a worst-case guarantee unlike the expected-cost guarantee as in the two-stage stochastic optimization model. In very recent work [7] authors consider a more general model of (two-stage) robust optimization in which scenarios are given implicitly, instead of explicitly as in the original model introduced in [3]. The difference is that in the more general version, one can have an exponentially large set of possible scenarios. For example, for the robust $k-$Steiner tree problem (with exponential scenarios), instead of having a total of m possible scenarios $S_1, \ldots, S_m$ ($S_i \subseteq V$) given explicitly, we can have each set $S \subset V$ of size k as one possible scenario and have the inflation factor λ_i to be uniformly equal to λ. In other words, in Stage 2, an arbitrary subset of size k of vertices is revealed as the terminals that need to be connected. Clearly the number of scenarios is exponential in the input k, and for this reason, (demand) robust optimization problems typically become more difficult in this more general setting.

1.1 Previous work

Robust optimization has been studied in [3, 9, 7, 12]. In [3, 9], the 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. They also give polylogarithmic approximation for robust min-cut and multi-cut. In the problem of robust min-cut, we are given an 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 1 and $E_2(i) \subseteq E$ for Stage 2 (if scenario i arrives) so that $E_1 \cup E_2(i)$ is a s, t_i-cut. In [3], 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 [9]. In the robust multi-cut problem, each scenario consists of a set of pairs of nodes that form a multi-cut problem. For this problem [3] 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. [7] 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 $c(S_i)$, a multiplicative increase factor λ, and an integer k. Each scenario can be a subset $U' \subset U$ of elements with size k that need to be covered. We have to purchase a collection of sets $\mathcal{S}_1 \subseteq \mathcal{S}$ in Stage 1. Once a set U' of elements is given in Stage 2, 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 λ, 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. This framework gives an $O(\log m \log n)$-approximation algorithm for the two-stage robust set cover problem, and a constant-factor approximation for the robust vertex cover problem. They also prove a hardness of factor $\Omega(\frac{\log m}{\log\log m})$ for the two-stage robust set cover under some plausible complexity assumptions. This framework does not apply to robust network design problems like the robust $k-$Steiner tree problem. In [7] the authors give logarithmic approximation for the robust uncapacitated metric facility location problem.

1.2 Our results

We only consider problems on undirected graphs. The model we study in this paper is the (demand) robust optimization model with (possibly) implicit sets of scenarios (which can be exponentially large). We study robust $k-$Steiner tree (in which a scenario consists of any k terminals out of given terminals), robust $k-$Steiner forest (in which a scenario consists of any k terminal-pairs out of given terminal-pairs), uncapacitated facility location (in which a scenario consists of any k clients—possibly located at the same location—out of given clients), and min-cut (in which a scenario consists of any *single* terminal from a given set of terminals) 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 $k-$Steiner tree and robust facility location problems. Our algorithms are combinatorial in nature and are based on nice structural properties of the Stage 1 solution of a near-optimal algorithm.

Theorem 1.1 *There exists a polynomial time 5.34-approximation algorithm for the two-stage robust $k-$Steiner tree problem with a uniform inflation factor. Here a scenario consists of any k terminals out of given terminals.*

Theorem 1.2 *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 factor approximation algorithm for the robust $k-$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 1.3 *There exists a polynomial time 3-approximation algorithm for the two-stage robust $k-$Steiner forest problem on trees with a uniform inflation factor. Here a scenario consists of any k terminal-pairs out of given terminal-pairs.*

Finally, we resolve an open question posed by [3] and [9]. These papers posed the question if the two-stage robust min-cut problem is NP-complete, or could be solved in polynomial time. We prove that the problem is hard to approximate within some constant factor $c > 1$ (namely, we show APX-hardness). See Theorem 1.4. The open question in [3, 9] was regarding a more difficult question in which the input contains many sinks and there is a different multiplicative inflation factor for every sink. We show that a much simpler problem is already APX-hard.

Theorem 1.4 *The two-stage robust min-cut problem is APX-hard even with a uniform inflation factor and in which a scenario consists of a single source and only 3 sinks.*

Organization. The remainder of the paper is organized as follows. In the next section, we present our 5.34-approximation algorithm for robust $k-$Steiner tree. Then we present our approximation algorithms for robust $k-$Steiner forest on trees and for the facility location problem in subsequent sections. Finally, we prove Theorem 1.4 in Section 5

2 A constant approximation for the robust $k-$Steiner tree problem

In this section we prove Theorem 1.1. 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 Steiner tree* problem, after our choice of edges in Stage 1 *any* subset of terminals can be presented by an adversary in Stage 2 and then the algorithm has to complete the choice in the first stage to a feasible Steiner tree for the set of terminals presented. We study a special case of robust Steiner tree, called *Robust $k-$Steiner tree*. The input to this problem is an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}^+$, and subset $T \subseteq V$ of "terminals". There is also an integer k and a real number $\lambda \geq 1$ called inflation factor. The solution is built in 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 λ 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 an optimal solution buys in the first stage and the second stage for scenario T', respectively. Let $\text{OPT} = \text{OPT}_1 + \lambda \cdot \text{OPT}_2$ be the overall cost of the optimum, where $\text{OPT}_1 = \sum_{e \in E_1^*} c_e$ is its cost in the first stage and $\text{OPT}_2 = \max_{T' \subseteq T, |T'| \leq k} \sum_{e \in E_2^*(T')} c_e$ is the maximum cost in the second stage divided by λ.

First Stage. Our algorithm, in the first stage, guesses an upper bound on the value of OPT_2 and then outputs some Steiner tree $\mathcal{T}$. More specifically, the algorithm tries values of the form $(1+\epsilon)^i$ as the guess for a good upper bound on OPT_2, for increasing values of i. Clearly, there is a value of i such that $(1+\epsilon)^i < \text{OPT}_2$, and $(1+\epsilon)^{i+1} \geq \text{OPT}_2$. Let μ be the guess value $(1+\epsilon)^{i+1}$ for OPT_2 for which the above condition holds. We later prove the following lemma:

Lemma 2.1 *If $\mu \geq \text{OPT}_2$ then algorithm returns a solution whose cost is at most*

$$c(\mathcal{T}) + \lambda \cdot r \cdot \mu \tag{1}$$

with r a universal constant.

We also show that $c(\mathcal{T}) = O(\text{OPT})$. This implies that the overall cost of the solution is small if and only if μ is close to OPT_2, which holds true, by construction. Indeed, the algorithm may fail to output a solution for $\mu_1 = (1+\epsilon)^i < \text{OPT}_2$ but we are guaranteed by Lemma 2.1 to output a solution for $\mu_2 = (1+\epsilon)^{i+1} \geq \text{OPT}_2$. Since $(1+\epsilon)^i < \text{OPT}_2 \leq (1+\epsilon)^{i+1}$ the guessed value μ_2 is a tight estimate of OPT_2 and a constant ratio follows from Equation (1).

To simplify the presentation, we assume that the guessed value μ for OPT_2 is exact.

Given the assumption that we know OPT_2 exactly, our algorithm 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:

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

where $r > 1$ is the above 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 $\frac{r \cdot \text{OPT}_2}{k}$ 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 γ-approximate, where $\gamma < 1.39$ [1]. The algorithm buys the edges in the Steiner tree in the first 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 $e = uw$ such that $d_G(u, v) \leq R$ and $d_G(w, v) \leq R$, and
- for the edges $e = uw$ 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 $d_G(c_i, c_j) > \frac{r \cdot \text{OPT}_2}{k}$ for any two distinct centers in $\mathcal{C}$, the balls $B_G(c_i, \frac{r \cdot \text{OPT}_2}{2k})$ and $B_G(c_j, \frac{r \cdot \text{OPT}_2}{2k})$ are disjoint. It is easy to see that the algorithm pays at most

$$\lambda \cdot r \cdot \text{OPT}_2 \tag{2}$$

in the second stage. This holds since the distance of any terminal to its assigned center is at most $\frac{r \cdot \text{OPT}_2}{k}$. Since at most k terminals need to be connected to their centers, the total cost of these connections is at most $\lambda \cdot k \cdot \frac{r \cdot \text{OPT}_2}{k}$. We now bound the cost of the algorithm in Stage 1 using the following lemma. Observe that since we use a γ-approximation to compute a Steiner tree in Stage 1 on centers $\mathcal{C}$, the following lemma together with (2) implies Lemma 2.1.

Lemma 2.2 *Assuming $r > 4$, there exists a Steiner tree on centers $\mathcal{C}$ in G that has cost at most $\frac{r}{r-4} \cdot \text{OPT}_1 + \text{OPT}_2$.*

Proof: Recall that E_1^* is the set of edges that optimum buys in Stage 1 and $\text{OPT}_1 = \sum_{e \in E_1^*} c_e$. Let H be a graph obtained from G by contracting 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 2.3 $|\mathcal{L}| < k$.

Proof: Assume on the contrary that $|\mathcal{L}| \geq k$ and let $T' \subseteq \mathcal{L}$ be any subset of size k. Consider scenario T'. After contracting 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 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 contracting the edges in E_1^*. Thus the claim holds. □

We now consider scenario $\mathcal{L}$. There exists a Steiner tree $E_{\mathcal{L}}^*$ on $\mathcal{L}$ in H with cost at most 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 which contains a Steiner tree on $\mathcal{C}$ in G.

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 $\frac{r \cdot \text{OPT}_2}{2k} - \frac{2\text{OPT}_2}{k} = \left(\frac{r}{2} - 2\right) \cdot \frac{\text{OPT}_2}{k}$ of the cost of each ball around a center in $\mathcal{C}$ belongs to E_1^*. 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 $c \in \mathcal{C}$. More precisely, $c(E_1^*) = \text{OPT}_1 \geq k \cdot \left(\frac{r}{2} - 2\right) \cdot \frac{\text{OPT}_2}{k} = \left(\frac{r}{2} - 2\right) \text{OPT}_2$. On the other hand, the overall cost of extending each $c \in \mathcal{C}$ to $\phi(c)$ is at most $k \cdot \frac{2\text{OPT}_2}{k} = 2\text{OPT}_2 \leq \frac{2}{\frac{r}{2}-2} \cdot \text{OPT}_1$. Thus, the resulting subgraph clearly contains a Steiner tree on $\mathcal{C}$ in G. The overall cost of this subgraph is thus at most $\text{OPT}_1 + \text{OPT}_2 + \frac{2}{\frac{r}{2}-2} \cdot \text{OPT}_1 = \frac{r}{r-4} \cdot \text{OPT}_1 + \text{OPT}_2$. Hence the proof. □

Since we use a γ-approximation algorithm to compute a Steiner tree in Stage 1, the overall cost of Stage 1 is at most $\frac{\gamma \cdot r}{r-4} \cdot \text{OPT}_1 + \gamma \cdot \text{OPT}_2$, which is $O(\text{OPT})$. Combining this with the second stage cost, which is bounded by $\lambda \cdot r \cdot \text{OPT}_2$ (see (2)) the overall cost of our solution is:

$$\frac{\gamma \cdot r}{r-4} \cdot \text{OPT}_1 + (\gamma + \lambda r) \cdot \text{OPT}_2. \tag{3}$$

This completes the proof of Lemme 2.1. Now we describe what the value of r should be. A trivial strategy for solving the robust $k-$Steiner tree is to select nothing in Stage 1 and make all the selections in Stage 2. Given that every edge is inflated by λ and we use a γ-approximation for Steiner tree, this strategy will have an approximation factor of $\lambda \cdot \gamma$. Using the best known approximation algorithm for Steiner tree [17], which has approximation 1.39, we get a 1.39λ-approximation. For values of $\lambda \leq 3.84$ we use this trivial strategy which gives an approximation factor of 5.33. For values of $\lambda > 3.84$ we use the above algorithm with parameter r defined below.

Let $r = r^*$ be the solution of: $\frac{\gamma \cdot r}{r-4} = \frac{\gamma}{\lambda} + r$. Then at $r = r^* = \frac{\gamma\lambda - \gamma + 4\lambda + \sqrt{\gamma^2\lambda^2 - 2\gamma^2\lambda + 8\gamma\lambda^2 + \gamma^2 + 8\gamma\lambda + 16\lambda^2}}{2\lambda}$, the two factors in front of OPT_1 and OPT_2 in the ratio of our algorithm calculated in Equation (3) become equal. Therefore, for $r = r^*$ and with $\gamma = 1.39$, the ratio of our algorithm becomes: $\frac{5.39\lambda - 1.39 + \sqrt{29.0521\lambda^2 + 7.2558\lambda + 1.9321}}{2\lambda}$.

It can be verified by a standard calculus argument that this expression is upper bounded by 5.335. Thus, for values of $\lambda > 3.84$, by choosing $r = r^*$, the ratio of our algorithm presented will be at most 5.34 and for smaller values of λ we use the trivial strategy which has ratio at most 5.33 as well. This completes the proof of Theorem 1.1.

3 A constant approximation for the robust facility location problem

In this section, we prove Theorem 1.2. In the (classical) uncapacitated facility location problem (UFL), the input is a set F of facilities, a set C of clients sites, and a metric ℓ on $F \cup C$. We assume that the distances ℓ_{ij} for $i, j \in F \cup C$ are symmetric and satisfy the triangle inequality. Each facility $i \in F$ has an *opening cost* f_i and the cost of connecting client $j \in C$ to facility $i \in F$ is ℓ_{ij} (also denoted by $\ell(i, j)$). The goal is to find a subset $F' \subseteq F$ of facilities to open such that the sum of total facility cost and the total service cost is minimized. For a client $j \in C$ and a subset of facilities $F' \subseteq F$, let $\ell(j, F') = \min_{i \in F'} \ell_{ij}$ denote the cost of serving j by the nearest facility in F'. Thus the objective is to find a subset F' of facilities to minimize

$$\sum_{i \in F'} f_i + \sum_{j \in C} \ell(j, F').$$

In the robust version, the input contains an integer k and an inflation factor $\lambda_i \geq 1$ for each facility i. There are two stages in solving the problem. In the first stage the algorithm has to open some facilities $F_1 \subseteq F$. After this choice is made, the cost of every un-opened facility $i \in F \setminus F_1$ is increased by a factor of λ_i to $\lambda_i \cdot f_i$. The adversary then chooses a (multi) set $\tilde{C} \subseteq C$ of at most k clients that actually need to be serviced (other clients do not need service). We assume that the adversary can pick multiple clients at a client site $j \in C$ so long as the total number of clients (counted with multiplicities) is at most k. We call $\tilde{C}$ a *scenario*. The algorithm has to choose another (possibly empty) set of facilities $F_2(\tilde{C}) \subseteq F \setminus F_1$ to be opened (at the higher costs) so that the total cost for this scenario is $\sum_{i \in F_1} f_i + \sum_{i \in F_2(\tilde{C})} \lambda_i \cdot f_i + \sum_{j \in \tilde{C}} \ell(j, F_1 \cup F_2(\tilde{C}))$. The objective is to minimize the maximum total cost over all scenarios, i.e., to minimize

$$\sum_{i \in F_1} f_i + \max_{\tilde{C} \subseteq C, |\tilde{C}| \leq k} \left(\sum_{i \in F_2(\tilde{C})} \lambda_i \cdot f_i + \sum_{j \in \tilde{C}} \ell(j, F_1 \cup F_2(\tilde{C})) \right).$$

We re-emphasize that the maximum in the above expression is taken over the *multi-sets* $\tilde{C}$ of size at most k.

Note that the service costs do not increase in the second stage. Note also that the number of potential scenarios $\tilde{C}$ is $|C|^k + |C|^{k-1} + \ldots + 1$, which is exponential in k. We denote this robust version of the problem (with exponential number of scenarios) by R-FL.

3.1 The algorithm

In what follows, we distinguish between client *sites* C and the actual clients $\tilde{C}$ that are realized in a particular scenario. Note that multiple clients may correspond to the same client site.

Let F_1^* be the set of facilities the optimum opens in Stage 1 and let $F_2^*(\tilde{C})$ be the set of facilities the optimum opens in Stage 2 when scenario $\tilde{C}$ occurs. Thus

$$\text{OPT} = \sum_{i \in F_1^*} f_i + \max_{\tilde{C}} \left(\sum_{i \in F_2^*(\tilde{C})} \lambda_i \cdot f_i + \sum_{j \in \tilde{C}} \ell(j, F_1^* \cup F_2^*(\tilde{C})) \right)$$

is the cost of the optimum.

First stage. Our algorithm, in the first stage, guesses a tight upper bound of OPT (as explained in the Steiner tree case). To simplify the presentation below, we assume that the guess on OPT is exact. It then computes a subset of client sites $\mathcal{C} = \{c_1, c_2, \ldots, c_p\} \subseteq C$ called "centers" and an assignment $\pi : C \to \mathcal{C}$ that satisfy:

- The centers are far apart: $\ell(c_i, c_j) > \frac{4\text{OPT}}{k}$ for all $i \neq j$, $c_i, c_j \in \mathcal{C}$, and
- Each client site is close to its assigned center: $\ell(j, \pi(j)) \leq \frac{4\text{OPT}}{k}$ for all $j \in C$.

Such a clustering can be computed as follows. Pick any client site and name it c_1. Assign all client sites within a distance of $\frac{4\text{OPT}}{k}$ from c_1 to site c_1 and remove these client sites. Pick any one of the remaining client site and name it c_2, and so on.

Now for a client site $j \in C$ and a real number $R > 0$, let $B(j, R) = \{i \in F \mid \ell(i, j) \leq R\}$ be the set of facilities within a distance R from j. We often call $B(j, \frac{2\text{OPT}}{k})$ the *ball* around j. Note that the balls around centers $c \in \mathcal{C}$ are disjoint.

Definition 3.1 *We call a center $c \in \mathcal{C}$ "cheap" if there is a facility in the ball around c with the Stage-2 cost at most $\frac{2\text{OPT}}{k}$, i.e., $\lambda_i \cdot f_i \leq \frac{2\text{OPT}}{k}$. We call a center "expensive" otherwise.*

The algorithm, in Stage 1, opens the cheapest facilities (w.r.t. Stage-1 costs) in the balls around the all expensive centers $c \in \mathcal{C}$. More precisely, for each expensive center $c \in \mathcal{C}$, the algorithm opens a facility $i = \operatorname{argmin}\{f_i \mid i \in B(c, \frac{2\text{OPT}}{k})\}$ where ties are broken arbitrarily.

Second stage. Let $\tilde{C}$ be the realized scenario. For every client $j \in \tilde{C}$ such that $\pi(j)$ is a cheap center, the algorithm opens the cheapest facility (w.r.t. Stage-2 costs) $i = \operatorname{argmin}\{\lambda_i \cdot f_i \mid i \in B(\pi(j), \frac{2\text{OPT}}{k})\}$ where ties are broken arbitrarily. The algorithm, then, serves each client $j \in \tilde{C}$ by an open facility in the ball around $\pi(j)$.

It is easy to see that the algorithm computes a feasible solution to the problem.

3.2 The analysis

It is easy to see that our solution has low service cost.

Lemma 3.2 *The total service cost of our algorithm, for any scenario $\tilde{C}$, is at most* 6OPT.

Proof: Fix a scenario $\tilde{C}$ and consider a client $j \in \tilde{C}$. Whether $\pi(j)$ is cheap or expensive, there is a open facility $i \in B(\pi(j), \frac{2\text{OPT}}{k})$. Since $\ell(j, \pi(j)) \leq \frac{4\text{OPT}}{k}$ and $\ell(\pi(j), i) \leq \frac{2\text{OPT}}{k}$, by the triangle inequality the service cost of client j is $\ell(j, i) \leq \frac{4\text{OPT}}{k} + \frac{2\text{OPT}}{k} = \frac{6\text{OPT}}{k}$. Because there are at most k clients in $\tilde{C}$, the total service cost is at most 6OPT. □

Now we bound the total facility cost.

Lemma 3.3 *The total facility cost of the facilities opened in Stage 2 by our algorithm is at most* 2OPT.

Proof: Recall that there are at most k clients in the scenario $\tilde{C}$. Thus in Stage 2, the algorithm opens the cheapest facilities in the balls around at most k cheap centers. From the definition of cheap centers, the total Stage-2 cost of these facilities is at most $k \cdot \frac{2\text{OPT}}{k} = 2\text{OPT}$. □

Now we bound the cost of the facilities opened in Stage 1 of the algorithm.

Definition 3.4 *We call a center $c \in \mathcal{C}$ "Stage-1" if the optimum opens a facility in the ball around c in Stage 1. We call a center "Stage-2" otherwise.*

Claim 3.5 *There are less than $k/2$ centers which are both expensive and Stage-2.*

Proof: Suppose, to the contrary, that there are $t \geq k/2$ expensive Stage-2 centers, say, $c_1, c_2, \ldots, c_t$. Now consider a scenario which consists of two copies each of $c_1, c_2, \ldots, c_{k/2}$. Consider how the optimum serves this scenario. By definition, the optimum does not open any facility in the balls around c_i for $i = 1, \ldots, k/2$ in Stage 1. Now suppose that in Stage 2, the optimum opens facilities in the balls around c_i for $i = 1, \ldots, t'$ and does not open any facilities in the balls around c_i for $i = t'+1, \ldots, k/2$. From the definition of expensive centers, the Stage-2 facility cost of the optimum is more than $t' \cdot \frac{2\text{OPT}}{k}$. Also the service cost of clients at the sites $c_{t'+1}, \ldots, c_{k/2}$ is more than $(k/2 - t') \cdot \frac{2\text{OPT}}{k}$. Thus the total cost of the optimum is more than $t' \cdot \frac{2\text{OPT}}{k} + (k/2 - t') \cdot \frac{2\text{OPT}}{k} = \text{OPT}$, which is a contradiction. □

Lemma 3.6 *The total facility cost of the facilities opened in Stage 1 by our algorithm is at most* 2OPT.

Proof: Recall that the algorithm opens the cheapest facilities in the balls around the expensive centers in Stage 1. Note that the total cost of the cheapest facilities in the balls around Stage-1 centers is at most OPT. This holds since the optimum itself opens at least one facility in each of the balls around Stage-1 centers in Stage 1.

Now we bound the total facility cost inside the balls around the expensive Stage-2 centers, say, $c_1, \ldots, c_t$. From Claim 3.5, we know that $t < k/2$. Let $f(j) = \min\{\lambda_i \cdot f_i \mid i \in B(c_j, \frac{2\text{OPT}}{k})\}$ denote the Stage-2 cost of the cheapest facility in the ball around center c_j, for $1 \leq j \leq t$. Consider a scenario in which the adversary requests

$$N(i) = \left\lceil \frac{f(i)}{\sum_{j=1}^{t} f(j)} \cdot \frac{k}{2} \right\rceil$$

clients at the client site c_i for $1 \leq i \leq t$. Note that $\sum_{i=1}^{t} N(i) < \sum_{i=1}^{t} \left(1 + \frac{f(i)}{\sum_{j=1}^{t} f(j)} \cdot \frac{k}{2}\right) < \frac{k}{2} + \frac{k}{2} = k$. Thus the total number of clients in this scenario is at most k.

Now consider how the optimum serves this scenario. Recall that the optimum does not open any facilities (in Stage 1) in the ball around a Stage-2 center. Suppose now that in Stage 2, the optimum opens facilities in the balls around centers $c_1, \ldots, c_{t'}$ and does not open any facilities in the balls around centers $c_{t'+1}, \ldots, c_t$. Thus the total facility cost of the optimum is at least $\sum_{i=1}^{t'} f(i)$ and the total service cost of the optimum is *strictly* more than $\sum_{i=t'+1}^{t} N(i) \cdot \frac{2\text{OPT}}{k} \geq \sum_{i=t'+1}^{t} \frac{f(i)}{\sum_{j=1}^{t} f(j)} \cdot \frac{k}{2} \cdot \frac{2\text{OPT}}{k} = \text{OPT} \cdot \sum_{i=t'+1}^{t} \frac{f(i)}{\sum_{j=1}^{t} f(j)}$. Since the total optimum cost is at most OPT, we have

$$\sum_{i=1}^{t'} f(i) + \text{OPT} \cdot \sum_{i=t'+1}^{t} \frac{f(i)}{\sum_{j=1}^{t} f(j)} < \text{OPT} = \text{OPT} \cdot \sum_{i=1}^{t} \frac{f(i)}{\sum_{j=1}^{t} f(j)}.$$

Simplifying the above expression, we get

$$\sum_{i=1}^{t'} f(i) < \text{OPT} \cdot \sum_{i=1}^{t'} \frac{f(i)}{\sum_{j=1}^{t} f(j)}.$$

This in particular implies that $\sum_{i=1}^{t'} f(i) > 0$ and that $\sum_{i=1}^{t} f(i) < \text{OPT}$. Since our algorithm opens the cheapest facilities (w.r.t. Stage-1 costs) in the balls around expensive centers in Stage 1, it pays at most $f(i)$ for each such center c_i. Hence the cost of these facilities is less than OPT. This completes the proof. □

Combining the above lemmas, we get our main theorem.

4 A constant approximation for the robust $k-$Steiner forest problem on trees

In this section, we prove Theorem 1.3. The input to the $k-$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 $k-$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 λ 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 $\sum_{e \in E_1} c_e + \max_{T' \subseteq T, |T'| \leq k} \lambda \cdot \sum_{e \in E_2(T')} c_e$.

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 1, and 0 otherwise. Note that any edge $e \in E(T')$ is picked in Stage 2 for scenario T' if and only if $x_e = 0$. Thus the Stage-2 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.

$$\begin{array}{lrcll} \min & \sum_e c_e \cdot x_e + \lambda \cdot C_2 & & & \\ s.t. & \sum_{e \in E(T')} c_e \cdot (1 - x_e) & \leq & C_2 & \forall \text{ scenarios } T' \\ & x_e & \in & \{0, 1\} & \forall \text{ edges } e \\ & C_2 & \geq & 0 & \end{array} \tag{4}$$

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.

4.1 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 $\mathsf{p}(v, u, l)$ 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 $\mathcal{T}$. It is crucial to note that the profit in this definition does not contain the profit in the path between the parent of v and u. We compute the values of 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 $\mathsf{p}(v, u, l)$ 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.

For the base case of the dynamic program, we set $\mathsf{p}(v, u, l)$ 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 $\mathsf{p}(v', u, l)$ for all children v' of v.

We first explain how to compute $\mathsf{p}(v, u, l)$ when v has only one child v_1. Let $e = (v, v_1)$. If $u = v$, we set $\mathsf{p}(v, v, l) = \max\{\mathsf{p}(v_1, v_1, l), \mathsf{p}(v_1, v, l) + y_e, \mathsf{p}(v_1, v_1, l-1) + y_e\}$. Note that we may add y_e in the third term as in $p(v, v, \ell)$ we are guaranteed that the path will go at least as high as v. For $u \neq v$, we let $\mathsf{p}(v, u, l) = \max\{\mathsf{p}(v_1, u, l) + y_e, \max_{u'} \mathsf{p}(v_1, u', l-1) + y_e\}$ 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 $\mathsf{p}(v, u, l)$ 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 $\mathsf{p}(v, v, l)$ 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 $\max_{l'}(\mathsf{p}(v_1, v_1, l') + \mathsf{p}(v_2, v_2, l - l'))$. The maximum profit covering e_1 but not e_2 is $\max_{l'}(\max\{\mathsf{p}(v_1, v, l'), \mathsf{p}(v_1, v_1, l' - 1)\} + y_{e_1} + \mathsf{p}(v_2, v_2, l - l'))$. Similarly, the maximum profit covering e_2 but not e_1 is maximum of $\max_{l'}(\max\{\mathsf{p}(v_2, v, l'), \mathsf{p}(v_2, v_2, l' - 1)\} + y_{e_2} + \mathsf{p}(v_1, v_1, l - l'))$. Similarly, the maximum profit covering both e_1 and e_2 is $\max_{l'}(\max\{\mathsf{p}(v_1, v, l'), \mathsf{p}(v_1, v_1, l' - 1)\} + y_{e_1} + \max\{\mathsf{p}(v_2, v, l - l'), \mathsf{p}(v_2, v_2, l - l' - 1)\} + y_{e_2})$.

Next consider the case when $u \neq v$. Again let $e_1 = (v, v_1)$ and $e_2 = (v, v_2)$. We set $\mathsf{p}(v, u, l)$ 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 $\max_{l'}(\mathsf{p}(v_1, v_1, l') + \mathsf{p}(v_2, v_2, (l - l'') - l'))$ 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 $\max_{l'}(\mathsf{p}(v_1, u, l') + y_{e_1} + \mathsf{p}(v_2, v_2, l - l'))$. Similarly, the maximum profit covering e_2 but not e_1 is maximum of $\max_{l'}(\mathsf{p}(v_2, u, l') + y_{e_2} + \mathsf{p}(v_1, v_1, l - l'))$. Similarly, the maximum profit covering

both e_1 and e_2 is the maximum of $\max_{l'}(\mathsf{p}(v_1, u, l') + y_{e_1} + \mathsf{p}(v_2, v, l - l') + y_{e_2})$ and $\max_{l'}(\mathsf{p}(v_2, u, l') + y_{e_2} + \mathsf{p}(v_1, v, l - l') + y_{e_1})$.

4.2 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 (4) 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 OPT^* denotes the cost of the optimum fractional solution to (4). We round this solution to an integral feasible solution to the Steiner forest problem on trees as follows: pick $e \in E$ in Stage 1 if and only if $x_e^* \geq \frac{1}{3}$. In Stage 2, given a scenario T', pick the remaining edges in $E(T')$ to form a feasible solution.

The cost of the Stage 1 of our solution is $\sum_{e:x_e^* \geq 1/3} c_e \leq 3 \sum_e c_e \cdot x_e^*$. The Stage 2 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 OPT^* is at most the optimum integral solution, our algorithm is a 3-approximation.

5 APX-hardness of the robust min-cut problem

In this section, we prove Theorem 1.4. 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 $\lambda \geq 1$. There are two stages and the algorithm has to first choose edges $E_1 \subseteq E$ in the first stage, after which the cost of each edge $e \in E \setminus E_1$ becomes $\lambda \cdot c_e$. We are then given a *single* sink $t \in T$. We call t a "scenario". The algorithm, then, has to pick edges $E_2(t) \subseteq E \setminus E_1$ such that s and t are not connected in the graph $(V, E \setminus \{E_1 \cup E_2(t)\})$. The objective is to minimize the maximum cost of the solution under any scenario: $c(E_1) + \max_{t \in T} \lambda \cdot c(E_2)$, where $c(X) = \sum_{e \in X} c_e$ for $X \subseteq E$.

In [15], the authors give a $(1 + \sqrt{2})$-approximation algorithm for this problem and pose as an open question to determine if this problem is NP-hard. We answer this question and in fact show that the problem is 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 [8] the following theorem is proved.

Theorem 5.1 [8] *There exists a universal (known in advance) constant $\alpha > 0$, 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 the source, T serves as the collection of terminals, and the edge-costs are given by c_e for $e \in E$ and $c_{e_1} = c_{e_2} = c_{e_3} = \beta$ where $\beta = 1 + \alpha$ where α is the constant from Theorem 5.1.

Lemma 5.2 *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 1. Then given any terminal $t_i \in T$ as a

scenario, we pick the edge e_i in Stage 2. This clearly forms a feasible solution with cost $c(E^*) + \lambda \cdot \beta \leq 1 + 2\beta$. □

Lemma 5.3 *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 1. If OPT picks exactly one edge, say e_1, in Stage 1, we consider scenario t_2. Since OPT has to pick e_2 in Stage 2 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 1, we consider scenario t_3. Since OPT has to pick e_3 in Stage 2 for this scenario, the overall cost is at least $2\beta + \lambda \cdot \beta = 4\beta$. Similarly, if OPT picks three edges in Stage 1, its cost is at least 3β.

Now consider the case where OPT does not pick any edge out of e_1, e_2, e_3 in Stage 1. Let E_1 be the set of edges OPT picks in Stage 1. Let $H = (V, E \setminus E_1)$. Let $E_{123} \subseteq E \setminus E_1$ be a minimum multi-way cut separating t_1, t_2, t_3 in H. Note that $c(E_1) + c(E_{123}) \geq 1 + \alpha$ and hence $c(E_{123}) \geq 1 + \alpha - c(E_1)$. For $i = 1, 2, 3$, let F_i denote the minimum cut separating t_i from the other two terminals in H. Note that each of $F_1 \cup F_2$, $F_2 \cup F_3$, and $F_3 \cup F_1$ form a multi-way cut separating the terminals in H. Therefore, $c(F_1) + c(F_2) \geq c(E_{123})$, $c(F_2) + c(F_3) \geq c(E_{123})$, and $c(F_3) + c(F_1) \geq c(E_{123})$. Thus $c(F_1) + c(F_2) + c(F_3) \geq \frac{3}{2} \cdot c(E_{123})$ and hence $\max_i c(F_i) \geq c(E_{123})/2 \geq (1 + \alpha - c(E_1))/2$.

Without loss of generality, let $c(F_1) = \max_i c(F_i)$. Now consider scenario t_1. In Stage 2, 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 $c(E_1) + \lambda \cdot c_{e_1} + \lambda c(F_1) \geq 2\beta + c(E_1) + 2 \cdot (1 + \alpha - c(E_1))/2 = 2\beta + 1 + \alpha$. In the other case, the overall cost of OPT is at least $c(E_1) + \lambda \cdot c_{e_1} + \lambda \min\{c_{e_2}, c_{e_3}\} \geq 4\beta$. 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 1.4.

Acknowledgments: We would like to thank anonymous referees for their comments and suggestions.

References

[1] J. Byrka, F. Grandoni, R. Thomas, and S. Laura *An improved LP-based approximation for steiner tree*, In Proceedings of the 42nd ACM Symposium on Theory of computing, 2010, pages 583-592.

[2] Jaroslaw Byrka, Karen Aardal, *An Optimal Bifactor Approximation Algorithm for the Metric Uncapacitated Facility Location Problem*, SIAM J. Comput. 39(6): 2212-2231, 2010.

[3] 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 Annual IEEE Symposium on Foundations of Computer Science (FOCS'05), pages 367-378.

[4] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour,, and M. Yannakakis, *The complexity of multiterminal cuts*, SIAM J. Comput. 23(4): 864-894, 1994.

[5] G. B. Dantzig, *Linear programming under uncertainty*, Management Sci., 1:197-206, 1955.

[6] 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.

[7] U. Feige, K. Jain, M. Mahdian, and V. Mirrokni, *Robust combinatorial optimization with exponential scenarios*, In Proc. of the 12th International Integer Programming and Combinatorial Optimization conference, Lecture Notes in Computer Science, 2007, Volume 4513, pages 439-453.

[8] E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour and M Yannakakis. *The complexity of multiterminal Cuts*, SIAM J. Comput., 23(4): 864-894, 1994.

[9] 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 the 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS) 2006, Lecture Notes in Computer Science, Volume 3884, pages 206-217.

[10] A. Gupta, M. Pál, R. Ravi, and A. Sinha, *Boosted sampling: approximation algorithms for stochastic optimization*, In Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (STOC) 2004, pages 86-98.

[11] 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 Annual IEEE Symposium on Foundations of Computer Science (FOCS) 2004, pages 218-227.

[12] A. Gupta, V. Nagarajan, and R. Ravi. *Thresholded covering algorithms for robust and max-min optimization* In Proc. of the 37th International Colloquium on Automata, Languages and Programming (ICALP) 2010, Lecture Notes in Computer Science, Volume 6198, pages 262-274.

[13] N. Immorlica, D. Karger, M. Minkoff, and V. Mirrokni, *On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems*, In Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms SODA 2004, pages 691-700.

[14] J. W. Milnor, *Games against nature*, in R. M. Thrall, C. H. Coomb, and R. L. Davis, editors, Decision Processes. Wiley.

[15] R. Ravi and A. Sinha, *Hedging uncertainty: Approximation algorithms for stochastic optimization problems*, Math. Program. 108(1): 97-114, 2006.

[16] R. Raz, *A parallel repetition theorem*, SIAM J. of Computing, 27(3): 763-803, 1998.

[17] G. Robins and A. Zelikovsky, *Improved Steiner tree approximation in graphs*, In Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms (SODA) 2000, pages 770-779.

[18] D. Shmoys and C. Swamy, *Stochastic optimization is (almost) as easy as deterministic optimization*, In Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS) 2004, pages 228-237.

[19] V. Vazirani, *Approximation algorithms*, Springer, 2001.