Robust Algorithms for TSP and Steiner Tree

Arun Ganesh
Department of Electrical Engineering and Computer Sciences, UC Berkeley, USA
arunganesh@berkeley.edu

Bruce M. Maggs
Department of Computer Science, Duke University, USA and Emerald Innovations, USA
bmm@cs.duke.edu

Debmalya Panigrahi
Department of Computer Science, Duke University, USA
debmalya@cs.duke.edu

Abstract

Robust optimization is a widely studied area in operations research, where the algorithm takes as input a range of values and outputs a single solution that performs well for the entire range. Specifically, a robust algorithm aims to minimize *regret*, defined as the maximum difference between the solution's cost and that of an optimal solution in hindsight once the input has been realized. For graph problems in **P**, such as shortest path and minimum spanning tree, robust polynomial-time algorithms that obtain a constant approximation on regret are known. In this paper, we study robust algorithms for minimizing regret in **NP**-hard graph optimization problems, and give constant approximations on regret for the classical traveling salesman and Steiner tree problems.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases Robust optimization, Steiner tree, traveling salesman problem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.54

Funding *Arun Ganesh*: Supported in part by NSF Award CCF-1535989.
Bruce M. Maggs: Supported in part by NSF Award CCF-1535972.
Debmalya Panigrahi: Supported in part by NSF grants CCF-1535972, CCF-1955703, an NSF CAREER Award CCF-1750140, and the Indo-US Virtual Networked Joint Center on Algorithms under Uncertainty.

1 Introduction

In many graph optimization problems, the inputs are not known precisely and the algorithm is desired to perform well over a range of inputs. For instance, consider the following situations. Suppose we are planning the delivery route of a vehicle that must deliver goods to n locations. Due to varying traffic conditions, the exact travel times between locations are not known precisely, but a range of possible travel times is available from historical data. Can we design a tour that is nearly optimal for *all* travel times in the given ranges? Consider another situation where we are designing a telecommunication network to connect a set of locations. We are given cost estimates on connecting every two locations in the network but these estimates might be off due to unexpected construction problems. Can we design the network in a way that is nearly optimal for *all* realized construction costs?

These questions have led to the field of *robust* graph algorithms. Given a range of weights $[\ell_e, u_e]$ for every edge e, the goal is to find a solution that minimizes *regret*, defined as the maximum difference between the algorithm's cost and the optimal cost for any edge weights. In other words, the goal is to obtain: $\min_{\text{SOL}} \max_{\mathbf{I}}(\text{SOL}(\mathbf{I}) - \text{OPT}(\mathbf{I}))$, where $\text{SOL}(\mathbf{I})$ (resp. $\text{OPT}(\mathbf{I})$) denotes the cost of SOL (resp. the optimal solution) in instance $\mathbf{I}$, SOL ranges over all feasible solutions, and $\mathbf{I}$ ranges over all realizable inputs. We emphasize that SOL is a fixed solution (independent of $\mathbf{I}$) whereas the solution determining $\text{OPT}(\mathbf{I})$ is dependent on the input $\mathbf{I}$. The solution that achieves this minimum is called the *minimum regret solution* (MRS), and its regret is the *minimum regret* (MR). In many cases, however, minimizing regret turns out to be **NP**-hard, in which case one seeks an approximation guarantee. Namely, a β-approximation algorithm satisfies, for all input realizations $\mathbf{I}$, $\text{SOL}(\mathbf{I}) - \text{OPT}(\mathbf{I}) \leq \beta \cdot \text{MR}$, i.e., $\text{SOL}(\mathbf{I}) \leq \text{OPT}(\mathbf{I}) + \beta \cdot \text{MR}$.

It is known that minimizing regret is **NP**-hard for shortest path [34] and minimum cut [1] problems, and using a general theorem for converting exact algorithms to robust ones, 2-approximations are known for these problems [12, 23]. In some cases, better results are known for special classes of graphs, e.g., [24]. Robust minimum spanning tree (MST) has also been studied, although in the context of making exponential-time exact algorithms more practical [33]. Moreover, robust optimization has been extensively researched for other (non-graph) problem domains in the operations research community, and has led to results in clustering [5, 3, 6, 27], linear programming [21, 28], and other areas [4, 23]. More details can be found in the book by Kouvelis and Yu [26] and the survey by Aissi *et al.* [2].

To the best of our knowledge, all previous work in polynomial-time algorithms for minimizing regret in robust graph optimization focused on problems in **P**. In this paper, we study robust graph algorithms for minimizing regret in **NP**-hard optimization problems. In particular, we study robust algorithms for the classical traveling salesman (TSP) and Steiner tree (STT) problems, that model e.g. the two scenarios described at the beginning of the paper. As a consequence of the **NP**-hardness, we cannot hope to show guarantees of the form: $\text{SOL}(\mathbf{I}) \leq \text{OPT}(\mathbf{I}) + \beta \cdot \text{MR}$, since for $\ell_e = u_e$ (i.e., $\text{MR} = 0$), this would imply an exact algorithm for an **NP**-hard optimization problem. Instead, we give guarantees: $\text{SOL}(\mathbf{I}) \leq \alpha \cdot \text{OPT}(\mathbf{I}) + \beta \cdot \text{MR}$, where α is (necessarily) at least as large as the best approximation guarantee for the optimization problem. We call such an algorithm an (α, β)-robust algorithm. If both α and β are constants, we call it a constant-approximation to the robust problem. In this paper, our main results are constant approximation algorithms for the robust traveling salesman and Steiner tree problems. We hope that our work will lead to further research in the field of robust approximation algorithms, particularly for other **NP**-hard optimization problems in graph algorithms as well as in other domains.

1.1 Problem Definition and Results

We first define the Steiner tree (STT) and traveling salesman problems (TSP). In both problems, the input is an undirected graph $G = (V, E)$ with non-negative edge costs. In Steiner tree, we are also given a subset of vertices called *terminals* and the goal is to obtain a minimum cost connected subgraph of G that spans all the terminals. In traveling salesman, the goal is to obtain a minimum cost tour that visits every vertex in V[1]. In the robust versions of these problems, the edge costs are ranges $[\ell_e, u_e]$ from which any cost may realize.

Our main results are the following:

▶ **Theorem 1.** *(Robust Approximations.) There exist constant approximation algorithms for the robust traveling salesman and Steiner tree problems.*

Remark: The constants we are able to obtain for the two problems are very different: $(4.5, 3.75)$ for TSP (in Section 3) and $(2755, 64)$ for STT (in Section 4). While we did not attempt to optimize the precise constants, obtaining small constants for STT comparable to the TSP result requires new ideas beyond our work and is an interesting open problem.

We complement our algorithmic results with lower bounds. Note that if $\ell_e = u_e$, we have MR $= 0$ and thus an (α, β)-robust algorithm gives an α-approximation for precise inputs. So, hardness of approximation results yield corresponding lower bounds on α. More interestingly, we show that hardness of approximation results also yield lower bounds on the value of β (see Section 5 for details):

▶ **Theorem 2.** *(APX-hardness.) A hardness of approximation of ρ for* TSP *(resp.,* STT*) under* $\mathbf{P} \neq \mathbf{NP}$ *implies that it is* **NP***-hard to obtain $\alpha \leq \rho$ (irrespective of β) and $\beta \leq \rho$ (irrespective of α) for robust* TSP *(resp., robust* STT*).*

1.2 Our Techniques

We now give a sketch of our techniques. Before doing so, we note that for problems in $\mathbf{P}$ with linear objectives, it is known that running an exact algorithm using weights $\frac{\ell_e+u_e}{2}$ gives a $(1, 2)$-robust solution [12, 23]. One might hope that a similar result can be obtained for **NP**-hard problems by replacing the exact algorithm with an approximation algorithm in the above framework. Unfortunately, there exists robust TSP instances where using a 2-approximation for TSP with weights $\frac{\ell_e+u_e}{2}$ gives a solution that is **not** (α, β)-robust for *any* $\alpha = o(n), \beta = o(n)$. More generally, a black-box approximation run on a fixed realization could output a solution including edges that have small weight relative to OPT for that realization (so including these edges does not violate the approximation guarantee), but these edges could have large weight relative to MR and OPT in other realizations, ruining the robustness guarantee. This establishes a qualitative difference between robust approximations for problems in $\mathbf{P}$ considered earlier and **NP**-hard problems being considered in this paper, and demonstrates the need to develop new techniques for the latter class of problems.

LP relaxation. We denote the input graph $G = (V, E)$. For each edge $e \in E$, the input is a range $[\ell_e, u_e]$ where the actual edge weight d_e can realize to any value in this range. The robust version of a graph optimization problem is is then described by the LP

$$\min\{r : \mathbf{x} \in P; \sum_{e \in E} d_e x_e \leq \text{OPT}(\mathbf{d}) + r,\ \forall \mathbf{d}\},$$

[1] There are two common and equivalent assumptions made in the TSP literature in order to achieve reasonable approximations. In the first assumption, the algorithms can visit vertices multiple times in the tour, while in the latter, the edges satisfy the metric property. We use the former in this paper.

where P is the standard polytope for the optimization problem, and $\textsc{opt}(\mathbf{d})$ denotes the cost of an optimal solution when the edge weights are $\mathbf{d} = \{d_e : e \in E\}$. That is, this is the standard LP for the problem, but with the additional constraint that the fractional solution $\mathbf{x}$ must have regret at most r for any realization of edge weights. We call the additional constraints the *regret constraint set*. Note that setting $\mathbf{x}$ to be the indicator vector of MRS and r to MR gives a feasible solution to the LP; thus, the LP optimum is at most MR.

Solving the LP. We assume that the constraints in P are separable in polynomial time (e.g., this is true for most standard optimization problems including STT and TSP). So, designing the separation oracle comes down to separating the regret constraint set, which requires checking that:

$$\max_{\mathbf{d}} \left[\sum_{e\in E} d_e x_e - \textsc{opt}(\mathbf{d})\right] =$$

$$\max_{\mathbf{d}} \max_{\textsc{sol}} \left[\sum_{e\in E} d_e x_e - \textsc{sol}(\mathbf{d})\right] = \max_{\textsc{sol}} \max_{\mathbf{d}} \left[\sum_{e\in E} d_e x_e - \textsc{sol}(\mathbf{d})\right] \le r.$$

Thus, given a fractional solution $\mathbf{x}$, we need to find an integer solution SOL and a weight vector $\mathbf{d}$ that maximizes the regret of $\mathbf{x}$ given by $\sum_{e\in E} d_e x_e - \textsc{sol}(\mathbf{d})$. Once SOL is fixed, finding $\mathbf{d}$ that maximizes the regret is simple: If SOL does not include an edge e, then to maximize $\sum_{e\in E} d_e x_e - \textsc{sol}(\mathbf{d})$, we set $d_e = u_e$; else if SOL includes e, we set $d_e = \ell_e$. Note that in these two cases, edge e contributes $u_e x_e$ and $\ell_e x_e - \ell_e$ respectively to the regret. The above maximization thus becomes:

$$\max_{\textsc{sol}} \left[\sum_{e\notin \textsc{sol}} u_e x_e + \sum_{e\in \textsc{sol}} (\ell_e x_e - \ell_e)\right] = \sum_{e\in E} u_e x_e - \min_{\textsc{sol}} \sum_{e\in \textsc{sol}} (u_e x_e - \ell_e x_e + \ell_e). \qquad (1)$$

Thus, SOL is exactly the optimal solution with edge weights $a_e := u_e x_e - \ell_e x_e + \ell_e$. (For reference, we define the *derived* instance of the problem as one with edge weights a_e.)

Now, if we were solving a problem in **P**, we would simply need to solve the problem on the derived instance. Indeed, we will show later that this yields an alternative technique for obtaining robust algorithms for problems in **P**, and recover existing results in [23]. However, we cannot hope to find an optimal solution to an **NP**-hard problem. Our first compromise is that we settle for an *approximate* separation oracle. More precisely, our goal is to show that there exists some fixed constants $\alpha', \beta' \ge 1$ such that if $\sum_e d_e x_e > \alpha' \cdot \textsc{opt}(\mathbf{d}) + \beta' \cdot r$ for some $\mathbf{d}$, then we can find $\textsc{sol}, \mathbf{d}'$ such that $\sum_e d'_e x_e > \textsc{sol}(\mathbf{d}') + r$. Since the LP optimum is at most MR, we can then obtain an (α', β')-robust *fractional* solution using the standard ellipsoid algorithm.

For TSP, we show that the above guarantee can be achieved by the classic MST-based 2-approximation on the derived instance. The details appear in Section 3 and the full paper. Although STT also admits a 2-approximation based on the MST solution, this turns out to be insufficient for the above guarantee. Instead, we use a different approach here. We note that the regret of the fractional solution against any fixed solution SOL (i.e., the argument over which Eq. (1) maximizes) can be expressed as the following difference:

$$\sum_{e\notin \textsc{sol}} (u_e x_e - \ell_e x_e + \ell_e) - \sum_{e\in E} (\ell_e - \ell_e x_e) = \sum_{e\notin \textsc{sol}} a_e - \sum_{e\in E} b_e, \text{ where } b_e := \ell_e - \ell_e x_e.$$

The first term is the weight of edges in the derived instance that are *not* in SOL. The second term corresponds to a new STT instance with different edge weights b_e. It turns out that the overall problem now reduces to showing the following approximation guarantees on these two STT instances (c_1 and c_2 are constants):

$$\text{(i)} \sum_{e \in \text{ALG} \setminus \text{SOL}} a_e \le c_1 \cdot \sum_{e \in \text{SOL} \setminus \text{ALG}} a_e \quad \text{and} \quad \text{(ii)} \sum_{e \in \text{ALG}} b_e \le c_2 \cdot \sum_{e \in \text{SOL}} b_e.$$

Note that guarantee (i) on the derived instance is an unusual "difference approximation" that is stronger than usual approximation guarantees. Moreover, we need these approximation bounds to *simultaneously* hold, i.e., hold for the same ALG. Obtaining these dual approximation bounds simultaneously forms the most technically challenging part of our work; a high level overview is given in Section 4 and technical details are deferred to the full paper.

Rounding the fractional solution. After applying our approximate separation oracles, we have a fractional solution $\mathbf{x}$ such that for all edge weights $\mathbf{d}$, we have $\sum_e d_e x_e \le \alpha' \cdot \text{OPT}(\mathbf{d}) + \beta' \cdot \text{MR}$. Suppose that, ignoring the regret constraint set, the LP we are using has integrality gap at most δ for precise inputs. Then a natural rounding approach is to search for an integer solution ALG that has minimum regret with respect to the specific solution $\delta\mathbf{x}$, i.e., ALG satisfies:

$$\text{ALG} = \operatorname*{argmin}_{\text{SOL}} \max_{\mathbf{d}} \left[\text{SOL}(\mathbf{d}) - \delta \sum_{e \in E} d_e x_e \right]. \tag{2}$$

Since the integrality gap is at most δ, we have $\delta \cdot \sum_{e \in E} d_e x_e \ge \text{OPT}(\mathbf{d})$ for any $\mathbf{d}$. This implies that:

$$\text{MRS}(\mathbf{d}) - \delta \cdot \sum_{e \in E} d_e x_e \le \text{MRS}(\mathbf{d}) - \text{OPT}(\mathbf{d}) \le \text{MR}.$$

Hence, the regret of MRS with respect to δx is at most MR. Since ALG has minimum regret with respect to $\delta\mathbf{x}$, ALG's regret is also at most MR. Note that $\delta\mathbf{x}$ is a $(\delta\alpha', \delta\beta')$-robust solution. Hence, ALG is a $(\delta\alpha', \delta\beta' + 1)$-robust solution.

If we are solving a problem in **P**, finding ALG that satisfies Eq. (2) is easy. So, using an integral LP formulation (i.e., integrality gap of 1), we get a $(1, 2)$-robust algorithm overall for these problems. This exactly matches the results in [23], although we are using a different set of techniques. Of course, for **NP**-hard problems, finding a solution ALG that satisfies Eq. (2) is **NP**-hard as well. It turns out, however, that we can design a generic rounding algorithm that gives the following guarantee:

▶ **Theorem 3.** *There exists a rounding algorithm that takes as input an (α, β)-robust fractional solution to* STT *(resp.* TSP*) and outputs a $(\gamma\delta\alpha, \gamma\delta\beta + \gamma)$-robust integral solution, where γ and δ are respectively the best approximation factor and integrality gap for (classical)* STT *(resp.,* TSP*).*

We remark that while we stated this rounding theorem for STT and TSP here, we actually give a more general version (Theorem 4) in Section 2 that applies to a broader class of covering problems including set cover, survivable network design, etc. and might be useful in future research in this domain.

1.3 Related Work

We have already discussed the existing literature in robust optimization for minimizing regret. Other robust variants of graph optimization have also been studied in the literature. In the *robust combinatorial optimization* model proposed by Bertsimas and Sim [7], edge costs are given as ranges as in this paper, but instead of optimizing for all realizations of costs within the ranges, the authors consider a model where at most k edge costs can be set to their maximum value and the remaining are set to their minimum value. The objective is to minimize the maximum cost over all realizations. In this setting, there is no notion of regret and an approximation algorithm for the standard problem translates to an approximation algorithm for the robust problem with the same approximation factor.

In the *data-robust model* [13], the input includes a polynomial number of explicitly defined "scenarios" for edge costs, with the goal of finding a solution that is approximately optimal for all given scenarios. That is, in the input one receives a graph and a polynomial number of scenarios $\mathbf{d}^{(1)}, \mathbf{d}^{(2)} \ldots \mathbf{d}^{(k)}$ and the goal is to find ALG whose maximum cost across all scenarios is at most some approximation factor times $\min_{\text{SOL}} \max_{i\in[k]} \sum_{e\in\text{SOL}} d_e^{(i)}$. In contrast, in this paper, we have exponentially many scenarios and look at the maximum of $\text{ALG}(\mathbf{d}) - \text{OPT}(\mathbf{d})$ rather than $\text{ALG}(\mathbf{d})$. A variation of this is the *recoverable robust model* [9], where after seeing the chosen scenario, the algorithm is allowed to "recover" by making a small set of changes to its original solution.

Dhamdhere *et al.* [13] also studies the *demand-robust model*, where edge costs are fixed but the different scenarios specify different connectivity requirements of the problem. The algorithm now operates in two phases: In the first phase, the algorithm builds a partial solution T' and then one of the scenarios (sets of terminals) T_i is revealed to the algorithm. In the second phase, the algorithm then adds edges to T' to build a solution T, but must pay a multiplicative cost of σ_k on edges added in the second phase. The demand-robust model was inspired by a two-stage stochastic optimization model studied in, e.g., [30, 29, 31, 13, 14, 25, 18, 19, 20, 8] where the scenario is chosen according to a distribution rather than an adversary.

Another related setting to the data-robust model is that of *robust network design*, introduced to model uncertainty in the demand matrix of network design problems (see the survey by Chekuri [10]). This included the well-known VPN conjecture (see, e.g., [17]), which was eventually settled in [15]. In all these settings, however, the objective is to minimize the maximum cost over all realizations, whereas in this paper, our goal is to minimize the maximum *regret* against the optimal solution.

2 Generalized Rounding Algorithm

We start by giving the rounding algorithm of Theorem 3, which is a corollary of the following, more general theorem:

▶ **Theorem 4.** *Let $\mathcal{P}$ be an optimization problem defined on a set system $\mathcal{S} \subseteq 2^E$ that seeks to find the set $S \in \mathcal{S}$ that minimizes $\sum_{e\in S} d_e$, i.e., the sum of the weights of elements in S. In the robust version of this optimization problem, we have $d_e \in [\ell_e, u_e]$ for all $e \in E$.*

Consider an LP formulation of $\mathcal{P}$ (called $\mathcal{P}$-LP) given by: $\{\min \sum_{e\in E} d_e x_e : \boldsymbol{x} \in X, \boldsymbol{x} \in [0,1]^E\}$, where X is a polytope containing the indicator vector χ_S of all $S \in \mathcal{S}$ and not containing χ_S for any $S \notin \mathcal{S}$. The corresponding LP formulation for the robust version (called $\mathcal{P}_{\text{robust}}$-LP) is given by: $\{\min r : \boldsymbol{x} \in X, \boldsymbol{x} \in [0,1]^E, \sum_{e\in E} d_e x_e \le \text{OPT}(\boldsymbol{d}) + r \ \forall \boldsymbol{d}\}$.

Now, suppose we have the following properties:

- *There is a γ-approximation algorithm for $\mathcal{P}$.*
- *The integrality gap of $\mathcal{P}$-LP is at most δ.*
- *There is a feasible solution $\boldsymbol{x}^*$ to $\mathcal{P}$-LP that satisfies: $\forall \boldsymbol{d} : \sum_{e\in E} d_e x_e^* \le \alpha \cdot \text{OPT}(\boldsymbol{d}) + \beta \cdot \text{MR}$.*

Then, there exists an algorithm that outputs an integer solution SOL for $\mathcal{P}$ that satisfies:

$$\forall \boldsymbol{d} : \text{SOL}(\boldsymbol{d}) \le (\gamma\delta\alpha) \cdot \text{OPT}(\boldsymbol{d}) + (\gamma\delta\beta + \gamma) \cdot \text{MR}.$$

Proof. The algorithm is as follows: Construct an instance of $\mathcal{P}$ which uses the same set system $\mathcal{S}$ and where element e has weight $\max\{u_e(1-\delta x_e^*), \ell_e(1-\delta x_e^*)\} + \delta \ell_e x_e^*$. Then, use the γ-approximation algorithm for $\mathcal{P}$ on this instance to find an integral solution S, and output it.

Given a feasible solution S to $\mathcal{P}$, note that:

$$\max_{\mathbf{d}}[\sum_{e\in S} d_e - \delta \sum_{e\in E} d_e x_e^*] = \sum_{e\in S} \max\{u_e(1-\delta x_e^*), \ell_e(1-\delta x_e^*)\} - \sum_{e\notin S} \delta \ell_e x_e^*$$

$$= \sum_{e\in S}[\max\{u_e(1-\delta x_e^*), \ell_e(1-\delta x_e^*)\} + \delta \ell_e x_e^*] - \sum_{e\in E} \delta \ell_e x_e^*.$$

Now, note that since S was output by a γ-approximation algorithm, for any feasible solution S':

$$\sum_{e\in S}[\max\{u_e(1-\delta x_e^*), \ell_e(1-\delta x_e^*)\} + \delta \ell_e x_e^*] \leq \gamma \sum_{e\in S'}[\max\{u_e(1-\delta x_e^*), \ell_e(1-\delta x_e^*)\} + \delta \ell_e x_e^*] \implies$$

$$\sum_{e\in S}[\max\{u_e(1-\delta x_e^*), \ell_e(1-\delta x_e^*)\} + \delta \ell_e x_e^*] - \gamma \sum_{e\in E} \delta \ell_e x_e^*$$

$$\leq \gamma[\sum_{e\in S'}[\max\{u_e(1-\delta x_e^*), \ell_e(1-\delta x_e^*)\} + \delta \ell_e x_e^*] - \sum_{e\in E} \delta \ell_e x_e^*]$$

$$= \gamma \max_{\mathbf{d}}[\sum_{e\in S'} d_e - \delta \sum_{e\in E} d_e x_e^*].$$

Since $\mathcal{P}$-LP has integrality gap δ, for any fractional solution $\mathbf{x}$, $\forall \mathbf{d}: \textsc{opt}(\mathbf{d}) \leq \delta \sum_{e\in E} d_e x_e$. Fixing S' to be the set of elements used in the minimum regret solution then gives:

$$\max_{\mathbf{d}}[\sum_{e\in S'} d_e - \delta \sum_{e\in E} d_e x_e^*] \leq \max_{\mathbf{d}}[\textsc{mrs}(\mathbf{d}) - \textsc{opt}(\mathbf{d})] = \textsc{mr}.$$

Combined with the previous inequality, this gives:

$$\sum_{e\in S}[\max\{u_e(1-\delta x_e^*), \ell_e(1-\delta x_e^*)\} + \delta \ell_e x_e^*] - \gamma \sum_{e\in E} \delta \ell_e x_e^* \leq \gamma \textsc{mr} \implies$$

$$\sum_{e\in S}[\max\{u_e(1-\delta x_e^*), \ell_e(1-\delta x_e^*)\} + \delta \ell_e x_e^*] - \sum_{e\in E} \delta \ell_e x_e^* \leq \gamma \textsc{mr} + (\gamma - 1) \sum_{e\in E} \delta \ell_e x_e^* \implies$$

$$\max_{\mathbf{d}}[\sum_{e\in S} d_e - \delta \sum_{e\in E} d_e x_e^*] \leq \gamma \textsc{mr} + (\gamma - 1) \sum_{e\in E} \delta \ell_e x_e^*.$$

This implies:

$$\forall \mathbf{d}: \textsc{sol}(\mathbf{d}) = \sum_{e\in S} d_e \leq \delta \sum_{e\in E} d_e x_e^* + \gamma \textsc{mr} + (\gamma - 1) \sum_{e\in E} \delta \ell_e x_e^*$$

$$\leq \delta \sum_{e\in E} d_e x_e^* + \gamma \textsc{mr} + (\gamma - 1) \sum_{e\in E} \delta d_e x_e^*$$

$$= \gamma\delta \sum_{e\in E} d_e x_e^* + \gamma \textsc{mr} \leq \gamma\delta[\alpha \textsc{opt}(\mathbf{d}) + \beta \textsc{mr}] + \gamma \textsc{mr} = \gamma\delta\alpha \cdot \textsc{opt}(\mathbf{d}) + (\gamma\delta\beta + \gamma) \cdot \textsc{mr}.$$

◀

Minimize r **subject to**

$$\begin{array}{ll}
\forall \emptyset \neq S \subset V: & \sum_{u \in S, v \in V \setminus S} y_{uv} \geq 2 \\
\forall u \in V: & \sum_{v \neq u} y_{uv} = 2 \\
\forall \emptyset \neq S \subset V, u \in S, v \in V \setminus S: & \sum_{e \in \delta(S)} x_{e,u,v} \geq y_{uv} \\
\forall \mathbf{d}: & \sum_{e \in E} d_e x_e \leq \text{OPT}(\mathbf{d}) + r \\
\forall u, v \in V, u \neq v: & 0 \leq y_{uv} \leq 1 \\
\forall e \in E, u, v \in V, v \neq u: & 0 \leq x_{e,u,v} \leq 1 \\
\forall e \in E: & x_e \leq 2
\end{array} \tag{3}$$

Figure 1 The Robust TSP Polytope

3 Algorithm for the Robust Traveling Salesman Problem

In this section, we give a robust algorithm for the traveling salesman problem:

▶ **Theorem 5.** *There exists a* $(4.5, 3.75)$*-robust algorithm for the traveling salesman problem.*

Recall that we consider the version of the problem where we are allowed to use edges multiple times in TSP. We present a high level sketch of our ideas here, the details are deferred to the full paper. We recall that any TSP tour must contain a spanning tree, and an Eulerian walk on a doubled MST is a 2-approximation algorithm for TSP (known as the "double-tree algorithm"). One might hope that since we have a $(1, 2)$-robust algorithm for robust MST, one could take its output and apply the double-tree algorithm to get a $(2, 4)$-robust solution to robust TSP. Unfortunately, this algorithm is not (α, β)-robust for any $\alpha = o(n), \beta = o(n)$. Nevertheless, we are able to leverage the connection to MST to arrive at a $(4.5, 3.75)$-robust algorithm for TSP.

3.1 Approximate Separation Oracle

We use the LP relaxation of robust traveling salesman in Figure 1. This is the standard subtour LP (see e.g. [32]), but augmented with variables specifying the edges used to visit each new vertex, as well as with the regret constraint set. Integrally, y_{uv} is 1 if splitting the tour into subpaths at each point where a vertex is visited for the first time, there is a subpath from u to v (or vice-versa). That is, y_{uv} is 1 if between the first time u is visited and the first time v is visited, the tour only goes through vertices that were already visited before visiting u. $x_{e,u,v}$ is 1 if on this subpath, the edge e is used. We use x_e to denote $\sum_{u,v \in V} x_{e,u,v}$ for brevity. A discussion of why the constraints other than the regret constraint set in (3) are identical to the standard TSP polytope is included in the full paper.

We now describe the separation oracle RRTSP-ORACLE used to separate (3). All constraints except the regret constraint set can be separated in polynomial time by solving a min-cut problem. Recall that exactly separating the regret constraint set involves finding an "adversary" SOL that maximizes $\max_{\mathbf{d}}[\sum_{e \in E} d_e x_e - \text{SOL}(\mathbf{d})]$, and seeing if this quantity exceeds r. However, since TSP is **NP**-hard, finding this solution in general is **NP**-hard. Instead, we will only consider a solution SOL if it is a walk on some spanning tree T, and find the one that maximizes $\max_{\mathbf{d}}[\sum_{e \in E} d_e x_e - \text{SOL}(\mathbf{d})]$.

Fix any SOL that is a walk on some spanning tree T. For any e, if e is not in T, the regret of $\mathbf{x}, \mathbf{y}$ against SOL is maximized by setting e's length to u_e. If e is in T, then SOL is paying $2d_e$ for that edge whereas the fractional solution pays $d_e x_e \leq 2d_e$, so to maximize the

$$\textbf{Minimize} \quad r \textbf{ subject to}$$

$$\forall S \subset V \text{ such that } \emptyset \subset S \cap T \subset T: \quad \textstyle\sum_{e \in \delta(S)} x_e \geq 1 \tag{4}$$

$$\forall \mathbf{d} \text{ such that } d_e \in [\ell_e, u_e]: \quad \textstyle\sum_{e \in E} d_e x_e \leq \text{OPT}(\mathbf{d}) + r \tag{5}$$

$$\forall e \in E: \quad x_e \in [0,1] \tag{6}$$

Figure 2 The Robust Steiner Tree Polytope

fractional solution's regret, d_e should be set to ℓ_e. This gives that the regret of fractional solution $\mathbf{x}$ against any SOL that is a spanning tree walk on T is

$$\sum_{e \in T}(\ell_e x_e - 2\ell_e) + \sum_{e \notin T} u_e x_e = \sum_{e \in E} u_e x_e - \sum_{e \in T}(u_e x_e - (\ell_e x_e - 2\ell_e)).$$

The quantity $\sum_{e \in E} u_e x_e$ is fixed with respect to T, so finding the spanning tree T that maximizes this quantity is equivalent to finding T that minimizes $\sum_{e \in T}(u_e x_e - (\ell_e x_e - 2\ell_e))$. But this is just an instance of the minimum spanning tree problem where edge e has weight $u_e x_e - (\ell_e x_e - 2\ell_e)$, and thus we can find T in polynomial time. After finding this spanning tree, RRTSP-ORACLE checks if the regret of $\mathbf{x}, \mathbf{y}$ against the walk on T is at least r, and if so outputs this as a violated inequality. If there is some $\text{SOL}, \mathbf{d}$ such that $\sum_{e \in E} d_e x_e > 2 \cdot \text{SOL}(\mathbf{d}) + r$, then the regret of the fractional solution against a walk on a spanning tree contained in SOL (which has cost at most $2 \cdot \text{SOL}(\mathbf{d})$ in realization $\mathbf{d}$) must be at least r, and thus its regret against T must also be at least r. This gives the following lemma:

▶ **Lemma 6.** *For any instance of robust traveling salesman there exists an algorithm* RRTSP-ORACLE *that given a solution* $(\boldsymbol{x}, \boldsymbol{y}, r)$ *to* (3) *either:*
- *Outputs a separating hyperplane for* (3)*, or*
- *Outputs "Feasible", in which case* $(\boldsymbol{x}, \boldsymbol{y})$ *is feasible for the (non-robust) TSP LP and* $\forall \boldsymbol{d} : \sum_{e \in E} d_e x_e \leq 2 \cdot \text{OPT}(\boldsymbol{d}) + r$.

The formal description of RRTSP-ORACLE and the proof of Lemma 6 are given in the full paper. By using the ellipsoid method with separation oracle RRTSP-ORACLE and the fact that (3) has optimum at most MR, we get a $(2, 1)$-robust fractional solution. Applying Theorem 3 as well as the fact that the TSP polytope has integrality gap $3/2$ (see e.g. [32]) and the TSP problem has a $3/2$-approximation gives Theorem 5.

4 Algorithm for the Robust Steiner Tree Problem

In this section, our goal is to find a fractional solution to the LP in Fig. 2 for robust Steiner tree. By Theorem 3 and known approximation/integrality gap results for Steiner Tree, this gives the following theorem:

▶ **Theorem 7.** *There exists a* $(2755, 64)$*-robust algorithm for the Steiner tree problem.*

It is well-known that the standard Steiner tree polytope admits an exact separation oracle (by solving the s, t-min-cut problem using edge weights x_e for all $s, t \in T$) so it is sufficient to find an approximate separation oracle for the regret constraint set. Unlike TSP, we do not know how to leverage the approximation for STT via solving an instance of MST, since this approximation uses information about shortest paths in the STT distance which are not well-defined when the weights are unknown. In turn, a more nuanced separation oracle and analysis is required. We present the main ideas of the separation oracle here, and defer the details to the full paper.

First, we create the derived instance of the Steiner tree problem which is a copy G' of the input graph G with edge weights $u_e x_e + \ell_e - \ell_e x_e$. As noted earlier, the optimal Steiner tree T^* on the derived instance maximizes the regret of the fractional solution $\mathbf{x}$. However, since Steiner tree is **NP**-hard, we cannot hope to exactly find T^*. We need a Steiner tree $\hat{T}$ such that the regret caused by it can be bounded against that caused by T^*. The difficulty is that the regret corresponds to the total weight of edges *not* in the Steiner tree (plus an offset that we will address later), whereas standard Steiner tree approximations give guarantees in terms of the total weight of edges in the Steiner tree. We overcome this difficulty by requiring a stricter notion of "difference approximation" – that the weight of edges $\hat{T} \setminus T^*$ be bounded against those in $T^* \setminus \hat{T}$. Note that this condition ensures that not only is the weight of edges in $\hat{T}$ bounded against those in T^*, but also that the weight of edges *not in* $\hat{T}$ is bounded against that of edges *not in* T^*. We show the following lemma to obtain the difference approximation:

▶ **Lemma 8.** *For any $\epsilon > 0$, there exists a polynomial-time algorithm for the Steiner tree problem such that if* $\textsc{opt}$ *denotes the set of edges in the optimal solution and $c(S)$ denotes the total weight of edges in S, then for any input instance of Steiner tree, the output solution* $\textsc{alg}$ *satisfies $c(\textsc{alg} \setminus \textsc{opt}) \leq (4 + \epsilon) \cdot c(\textsc{opt} \setminus \textsc{alg})$.*

The algorithm proving Lemma 8 is a local search procedure proposed by [16] (who considered the more general Steiner forest) that considers local moves of the following form: For the current solution $\textsc{alg}$, a local move consists of adding any path f whose endpoints are vertices in $\textsc{alg}$ and whose intermediate vertices are not in $\textsc{alg}$, and then deleting from $\textsc{alg}$ a subpath a in the resulting cycle such that $\textsc{alg} \cup f \setminus a$ remains feasible. We extend the results in [16] by showing that such an algorithm is 4-approximate for Steiner tree. We can further extend this argument to show that such an algorithm, in fact, satisfies the stricter difference approximation requirement in Lemma 8 (see the full paper for details).

Recall that the regret caused by T is not exactly the weight of edges not in T, but includes a fixed offset of $\sum_{e \in E}(\ell_e - \ell_e x_e)$. If $\ell_e = 0$ for all edges, i.e., the offset is 0, then we can recover a robust algorithm from Lemma 8 alone with much better constants than in Theorem 7 (we defer the discussion/proof of this result to the full paper). In general though, the approximation guarantee given in Lemma 8 alone does not suffice because of the offset. We instead rely on a stronger notion of approximation formalized in the next lemma that provides simultaneous approximation guarantees on two sets of edge weights: $c_e = u_e x_e - \ell_e x_e + \ell_e$ and $c'_e = \ell_e - \ell_e x_e$. The guarantee on $\ell_e - \ell_e x_e$ can then be used to handle the offset.

▶ **Lemma 9.** *Let G be a graph with terminals T and two sets of edge weights c and c'. Let* $\textsc{sol}$ *be any Steiner tree connecting T. Let $\Gamma' > 1$, $\kappa > 0$, and $0 < \epsilon < \frac{4}{35}$ be fixed constants. Then there exists a constant Γ (depending on $\Gamma', \kappa, \epsilon$) and an algorithm that obtains a collection of Steiner trees* $\textsc{alg}$*, at least one of which (let us call it $\textsc{alg}_i$) satisfies:*

- $c(\textsc{alg}_i \setminus \textsc{sol}) \leq 4\Gamma \cdot c(\textsc{sol} \setminus \textsc{alg}_i)$, *and*
- $c'(\textsc{alg}_i) \leq (4\Gamma' + \kappa + 1 + \epsilon) \cdot c'(\textsc{sol})$.

The fact that Lemma 9 generates multiple solutions (but only polynomially many) is fine because as long as we can show that one of these solutions causes sufficient regret, our separation oracle can just iterate over all solutions until it finds one that causes sufficient regret.

We give a high level sketch of the proof of Lemma 9 here, and defer details to the full paper. The algorithm uses the idea of *alternate minimization*, alternating between a "forward phase" and a "backward phase". The goal of each forward phase/backward phase pair is to keep $c'(\textsc{alg})$ approximately fixed while obtaining a net decrease in $c(\textsc{alg})$. In the forward phase, the algorithm greedily uses local search, choosing swaps that decrease c and increase c' at the best "rate of exchange" between the two costs (i.e., the maximum ratio of decrease in c to increase in c'), until $c'(\textsc{alg})$ has increased past some upper threshold. Then, in the backward phase, the algorithm greedily chooses swaps that decrease c' while increasing c at the best rate of exchange, until $c'(\textsc{alg})$ reaches some lower threshold, at which point we start a new forward phase.

We guess the value of $c'(\textsc{sol})$ (we can run many instances of this algorithm and generate different solutions based on different guesses for this purpose) and set the upper threshold for $c'(\textsc{alg})$ appropriately so that we satisfy the approximation guarantee for c'. For c we show that as long as $\textsc{alg}$ is not a 4Γ-difference approximation with respect to c then a forward/backward phase pair reduces $c(\textsc{alg})$ by a non-negligible amount (of course, if $\textsc{alg}$ is a 4Γ-difference approximation then we are done). This implies that after enough iterations, $\textsc{alg}$ must be a 4Γ-difference approximation as $c(\textsc{alg})$ can only decrease by a bounded amount. To show this, we claim that while $\textsc{alg}$ is not a 4Γ-difference approximation, for sufficiently large Γ the following bounds on rates of exchange hold:

- For each swap in the forward phase, the ratio of decrease in $c(\textsc{alg})$ to increase in $c'(\textsc{alg})$ is at least some constant k_1 times $\frac{c(\textsc{alg} \setminus \textsc{sol})}{c'(\textsc{sol} \setminus \textsc{alg})}$.
- For each swap in the backward phase, the ratio of increase in $c(\textsc{alg})$ to decrease in $c'(\textsc{alg})$ is at most some constant k_2 times $\frac{c(\textsc{sol} \setminus \textsc{alg})}{c'(\textsc{alg} \setminus \textsc{sol})}$.

Before we discuss how to prove this claim, let us see why this claim implies that a forward phase/backward phase pair results in a net decrease in $c(\textsc{alg})$. If this claim holds, suppose we set the lower threshold for $c'(\textsc{alg})$ to be, say, $101c'(\textsc{sol})$. That is, throughout the backward phase, we have $c'(\textsc{alg}) > 101c'(\textsc{sol})$. This lower threshold lets us rewrite our upper bound on the rate of exchange in the backward phase in terms of the lower bound on rate of exchange for the forward phase:

$$k_2 \frac{c(\textsc{sol} \setminus \textsc{alg})}{c'(\textsc{alg} \setminus \textsc{sol})} \leq k_2 \frac{c(\textsc{sol} \setminus \textsc{alg})}{c'(\textsc{alg}) - c'(\textsc{sol})} \leq k_2 \frac{c(\textsc{sol} \setminus \textsc{alg})}{100c'(\textsc{sol})} \leq k_2 \frac{c(\textsc{sol} \setminus \textsc{alg})}{100c'(\textsc{sol} \setminus \textsc{alg})}$$
$$\leq k_2 \frac{1}{4\Gamma} \frac{c(\textsc{alg} \setminus \textsc{sol})}{100c'(\textsc{sol} \setminus \textsc{alg})} = \frac{k_2}{400\Gamma k_1} \cdot k_1 \frac{c(\textsc{alg} \setminus \textsc{sol})}{c'(\textsc{sol} \setminus \textsc{alg})}.$$

In other words, the bound in the claim for the rate of exchange in the forward phase is larger than the bound for the backward phase by a multiplicative factor of $\Omega(1) \cdot \Gamma$. While these bounds depend on $\textsc{alg}$ and thus will change with every swap, let us make the simplifying assumption that through one forward phase/backward phase pair these bounds remain constant. Then, the change in $c(\textsc{alg})$ in one phase is just the rate of exchange for that phase times the change in $c'(\textsc{alg})$, which by definition of the algorithm is roughly equal in absolute value for the forward and backward phase. So this implies that the decrease in $c(\textsc{alg})$ in the forward phase is $\Omega(1) \cdot \Gamma$ times the increase in $c(\textsc{alg})$ in the backward phase, i.e., the net change across the phases is a non-negligible decrease in $c(\textsc{alg})$ if Γ is sufficiently large. Without the simplifying assumption, we can still show that the decrease in $c(\textsc{alg})$ in the forward phase is larger than the increase in $c(\textsc{alg})$ in the backward phase for large enough Γ using a much more technical analysis. In particular, our analysis will show there is a net decrease as long as:

$$\min\left\{\frac{4\Gamma-1}{8\Gamma}, \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\right\} - (e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)}-1) > 0, \tag{7}$$

where

$$\zeta' = \frac{4(1+\epsilon)\Gamma'}{(\sqrt{\Gamma'}-1)(\sqrt{\Gamma'}-1-\epsilon)(4\Gamma'-1)(4\Gamma-1)}.$$

Note that for any positive $\epsilon, \kappa, \Gamma'$, there exists a sufficiently large value of Γ for (7) to hold, since as $\Gamma \to \infty$, we have $\zeta' \to 0$, so that

$$(e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)}-1) \to 0 \text{ and}$$

$$\min\left\{\frac{4\Gamma-1}{8\Gamma}, \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\right\} \to \min\{1/2, \kappa/(4+4\epsilon)\}.$$

So, the same intuition holds: as long as we are willing to lose a large enough Γ value, we can make the increase in $c(\text{ALG})$ due to the backward phase negligible compared to the decrease in the forward phase, giving us a net decrease.

It remains to argue that the claimed bounds on rates of exchange hold. Let us argue the claim for $\Gamma = 4$, although the ideas easily generalize to other choices of Γ. We do this by generalizing the analysis of the local search algorithm. This analysis shows that if ALG is a locally optimal solution, then

$$c(\text{ALG} \setminus \text{SOL}) \le 4 \cdot c(\text{SOL} \setminus \text{ALG}),$$

i.e., ALG is a 4-difference approximation of SOL. The contrapositive of this statement is that if ALG is not a 4-difference approximation, then there is at least one swap that will improve it by some amount. We modify the approach of [16] by weakening the notion of locally optimal. In particular, suppose we define a solution ALG to be "approximately" locally optimal if at least half of the (weighted) swaps between paths a in ALG $\setminus$ SOL and paths f in SOL $\setminus$ ALG satisfy $c(a) \le 2c(f)$ (as opposed to $c(a) \le c(f)$ in a locally optimal solution; the choice of 2 for both constants here implies $\Gamma = 4$). Then a modification of the analysis of the local search algorithm, losing an additional factor of 4, shows that if ALG is approximately locally optimal, then

$$c(\text{ALG} \setminus \text{SOL}) \le 16 \cdot c(\text{SOL} \setminus \text{ALG}).$$

The contrapositive of this statement, however, has a stronger consequence than before: if ALG is not a 16-difference approximation, then a weighted half of the swaps satisfy $c(a) > 2c(f)$, i.e. reduce $c(\text{ALG})$ by at least

$$c(a) - c(f) > c(a) - c(a)/2 = c(a)/2.$$

The decrease in $c(\text{ALG})$ due to all of these swaps together is at least $c(\text{ALG} \setminus \text{SOL})$ times some constant. In addition, since a swap between a and f increases $c'(\text{ALG})$ by at most $c'(f)$, the total increase in c' due to these swaps is at most $c'(\text{SOL} \setminus \text{ALG})$ times some other constant. An averaging argument then gives the rate of exchange bound for the forward phase in the claim, as desired. The rate of exchange bound for the backward phase follows analogously.

This completes the algorithm and proof summary, although more detail is needed to formalize these arguments. Moreover, we also need to show that the algorithm runs in polynomial time. These details are given in the full paper.

We now formally define our separation oracle RRST-Oracle in Fig. 3 and prove that it is an approximate separation oracle in the lemma below:

```
RRST-ORACLE(G(V,E), {[ℓ_e, u_e]}_{e∈E}, (x, r))
   Data: Undirected graph G(V,E), lower and upper bounds on edge lengths
         {[ℓ_e, u_e]}_{e∈E}, solution (x = {x_e}_{e∈E}, r) to the LP in Fig. 2
1  Check all constraints of the LP in Fig. 2 except regret constraint set, return any
    violated constraint that is found;
2  G' ← copy of G where c_e = u_e x_e − ℓ_e x_e + ℓ_e, c'_e = ℓ_e − ℓ_e x_e;
3  ALG ← output of algorithm from Lemma 9 on G';
4  for ALG_i ∈ ALG do
5  |  if Σ_{e∉ALG_i} u_e x_e + Σ_{e∈ALG_i} ℓ_e x_e − Σ_{e∈ALG_i} ℓ_e > r then
6  |  |  return Σ_{e∉ALG_i} u_e x_e + Σ_{e∈ALG_i} ℓ_e x_e − Σ_{e∈ALG_i} ℓ_e ≤ r;
7  |  end
8  end
9  return "Feasible";
```

Figure 3 Separation Oracle for LP in Fig. 2

▶ **Lemma 10.** *Fix any $\Gamma' > 1, \kappa > 0, 0 < \epsilon < 4/35$ and let Γ be the constant given in Lemma 9. Let $\alpha = (4\Gamma' + \kappa + 2 + \epsilon)4\Gamma + 1$ and $\beta = 4\Gamma$. Then there exists an algorithm* RRST-ORACLE *that given a solution $(\boldsymbol{x}, r)$ to the LP in Fig. 2 either:*

- *Outputs a separating hyperplane for the LP in Fig. 2, or*
- *Outputs "Feasible", in which case $\boldsymbol{x}$ is feasible for the (non-robust) Steiner tree LP and*

$$\forall \boldsymbol{d} : \sum_{e \in E} d_e x_e \leq \alpha \cdot \text{OPT}(\boldsymbol{d}) + \beta \cdot r.$$

Proof. It suffices to show that if there exists $\mathbf{d}, \text{SOL}$ such that

$$\sum_{e \in E} d_e x_e > \alpha \cdot \text{SOL}(\mathbf{d}) + \beta \cdot r, \text{ i.e., } \sum_{e \in E} d_e x_e - \alpha \cdot \text{SOL}(\mathbf{d}) > \beta \cdot r$$

then RRST-ORACLE outputs a violated inequality on line 6, i.e., the algorithm finds a Steiner tree T' such that

$$\sum_{e \notin T'} u_e x_e + \sum_{e \in T'} \ell_e x_e - \sum_{e \in T'} \ell_e > r.$$

Notice that in the inequality

$$\sum_{e \in E} d_e x_e - \alpha \cdot \text{SOL}(\mathbf{d}) > \beta \cdot r,$$

replacing $\mathbf{d}$ with $\mathbf{d}'$ where $d'_e = \ell_e$ when $e \in \text{SOL}$ and $d'_e = u_e$ when $e \notin \text{SOL}$ can only increase the left hand side. So replacing $\mathbf{d}$ with $\mathbf{d}'$ and rearranging terms, we have

$$\sum_{e \in \text{SOL}} \ell_e x_e + \sum_{e \notin \text{SOL}} u_e x_e > \alpha \sum_{e \in \text{SOL}} \ell_e + \beta \cdot r = \sum_{e \in \text{SOL}} \ell_e + \left[(\alpha - 1) \sum_{e \in \text{SOL}} \ell_e + \beta \cdot r\right].$$

In particular, the regret of the fractional solution against SOL is at least $(\alpha - 1)\sum_{e \in \text{SOL}} \ell_e + \beta \cdot r$.

Let T' be the Steiner tree satisfying the conditions of Lemma 9 with $c_e = u_e x_e - \ell_e x_e + \ell_e$ and $c'_e = \ell_e - \ell_e x_e$. Let $E_0 = E \setminus (\text{SOL} \cup T')$, $E_S = \text{SOL} \setminus T'$, and $E_T = T' \setminus \text{SOL}$. Let $c(E')$ for $E' = E_0, E_S, E_T$ denote $\sum_{e \in E'} (u_e x_e - \ell_e x_e + \ell_e)$, i.e., the total weight of the edges E' in G'. Now, note that the regret of the fractional solution against a solution using edges E' is:

$$\sum_{e \notin E'} u_e x_e + \sum_{e \in E'} \ell_e x_e - \sum_{e \in E'} \ell_e = \sum_{e \notin E'} (u_e x_e - \ell_e x_e + \ell_e) - \sum_{e \in E} (\ell_e - \ell_e x_e)$$
$$= c(E \setminus E') - \sum_{e \in E} (\ell_e - \ell_e x_e).$$

Plugging in $E' = \text{SOL}$, we then get that:

$$c(E_0) + c(E_T) - \sum_{e \in E} (\ell_e - \ell_e x_e) > (\alpha - 1) \sum_{e \in \text{SOL}} \ell_e + \beta \cdot r.$$

Isolating $c(E_T)$ then gives:

$$c(E_T) > (\alpha - 1) \sum_{e \in \text{SOL}} \ell_e + \beta \cdot r - \sum_{e \in E_0} (u_e x_e - \ell_e x_e + \ell_e) + \sum_{e \in E} (\ell_e - \ell_e x_e)$$
$$= (\alpha - 1) \sum_{e \in \text{SOL}} \ell_e + \beta \cdot r - \sum_{e \in E_0} u_e x_e + \sum_{e \notin E_0} (\ell_e - \ell_e x_e).$$

Since $\beta = 4\Gamma$, Lemma 9 along with an appropriate choice of ϵ gives that $c(E_T) \leq \beta c(E_S)$, and thus:

$$c(E_S) > \frac{1}{\beta} \left[(\alpha - 1) \sum_{e \in \text{SOL}} \ell_e + \beta \cdot r - \sum_{e \in E_0} u_e x_e + \sum_{e \notin E_0} (\ell_e - \ell_e x_e) \right].$$

Recall that our goal is to show that $c(E_0) + c(E_S) - \sum_{e \in E} (\ell_e - \ell_e x_e) > r$, i.e., that the regret of the fractional solution against T' is at least r. Adding $c(E_0) - \sum_{e \in E} (\ell_e - \ell_e x_e)$ to both sides of the previous inequality, we can lower bound $c(E_0) + c(E_S) - \sum_{e \in E} (\ell_e - \ell_e x_e)$ as follows:

$$c(E_0) + c(E_S) - \sum_{e \in E} (\ell_e - \ell_e x_e)$$
$$> \frac{1}{\beta} \left[(\alpha - 1) \sum_{e \in \text{SOL}} \ell_e + \beta \cdot r - \sum_{e \in E_0} u_e x_e + \sum_{e \notin E_0} (\ell_e - \ell_e x_e) \right]$$
$$+ \sum_{e \in E_0} (u_e x_e - \ell_e x_e + \ell_e) - \sum_{e \in E} (\ell_e - \ell_e x_e)$$
$$= r + \frac{\alpha - 1}{\beta} \sum_{e \in \text{SOL}} \ell_e + \frac{1}{\beta} \sum_{e \notin E_0} (\ell_e - \ell_e x_e) + \frac{\beta - 1}{\beta} \sum_{e \in E_0} u_e x_e - \sum_{e \notin E_0} (\ell_e - \ell_e x_e)$$
$$\geq r + \frac{\alpha - 1 - \beta}{\beta} \sum_{e \in \text{SOL}} \ell_e + \frac{1}{\beta} \sum_{e \notin E_0} (\ell_e - \ell_e x_e) + \frac{\beta - 1}{\beta} \sum_{e \in E_0} u_e x_e - \sum_{e \in E_T} (\ell_e - \ell_e x_e) \geq r.$$

Here, the last inequality holds because by our setting of α, we have

$$\frac{\alpha - 1 - \beta}{\beta} = 4\Gamma' + \kappa + 1 + \epsilon,$$

and thus Lemma 9 gives that

$$\sum_{e \in E_T} (\ell_e - \ell_e x_e) \leq \frac{\alpha - 1 - \beta}{\beta} \sum_{e \in \text{SOL}} (\ell_e - \ell_e x_e) \leq \frac{\alpha - 1 - \beta}{\beta} \sum_{e \in \text{SOL}} \ell_e.$$

◀

By using Lemma 10 with the ellipsoid method and the fact that the LP optimum is at most MR, we get an (α, β)-robust fractional solution. Then, Theorem 3 and known approximation/integrality gap results give us the following theorem, which with appropriate choice of constants gives Theorem 7:

▶ **Theorem 11.** *Fix any $\Gamma' > 1, \kappa > 0, 0 < \epsilon < 4/35$ and let Γ be the constant given in Lemma 9. Let $\alpha = (4\Gamma' + \kappa + 2 + \epsilon)4\Gamma + 1$ and $\beta = 4\Gamma$. Then there exists a polynomial-time $(2\alpha \ln 4 + \epsilon, 2\beta \ln 4 + \ln 4 + \epsilon)$-robust algorithm for the Steiner tree problem.*

5 Lower Bounds

To contextualize our approximation guarantees, we give the following generalized hardness result for a family of problems which includes many graph optimization problems:

▶ **Theorem 12.** *Let $\mathcal{P}$ be any robust covering problem whose input includes a weighted graph G where the lengths d_e of the edges are given as ranges $[\ell_e, u_e]$ and for which the non-robust version of the problem, $\mathcal{P}'$, has the following properties:*

- *A solution to an instance of $\mathcal{P}'$ can be written as a (multi-)set S of edges in G, and has cost $\sum_{e \in S} d_e$.*
- *Given an input including G to $\mathcal{P}'$, there is a polynomial-time approximation-preserving reduction from solving $\mathcal{P}'$ on this input to solving $\mathcal{P}'$ on some input including G', where G' is the graph formed by taking G, adding a new vertex v^*, and adding a single edge from v^* to some $v \in V$ of weight 0.*
- *For any input including G to $\mathcal{P}'$, given any spanning tree T of G, there exists a feasible solution only including edges from T.*

Then, if there exists a polynomial time (α, β)-robust algorithm for $\mathcal{P}$, there exists a polynomial-time β-approximation algorithm for $\mathcal{P}$.

Before proving Theorem 12, we note that robust traveling salesman and robust Steiner tree are examples of problems that Theorem 12 implicitly gives lower bounds for. For both problems, the first property clearly holds.

For traveling salesman, given any input G, any solution to the problem on input G' as described in Theorem 12 can be turned into a solution of the same cost on input G by removing the new vertex v^* (since v^* was distance 0 from v, removing v^* does not affect the length of any tour), giving the second property. For any spanning tree of G, a walk on the spanning tree gives a valid TSP tour, giving the third property.

For Steiner tree, for the input with graph G' and the same terminal set, for any solution containing the edge (v, v^*) we can remove this edge and get a solution for the input with graph G that is feasible and of the same cost. Otherwise, the solution is already a solution for the input with graph G that is feasible and of the same cost, so the second property holds. Any spanning tree is a feasible Steiner tree, giving the third property.

We now give the proof of Theorem 12.

Proof of Theorem 12. Suppose there exists a polynomial time (α, β)-robust algorithm A for $\mathcal{P}$. The β-approximation algorithm for $\mathcal{P}'$ is as follows:

1. From the input instance $\mathcal{I}$ of $\mathcal{P}$ where the graph is G, use the approximation-preserving reduction (that must exist by the second property of the theorem) to construct instance $\mathcal{I}'$ of $\mathcal{P}'$ where the graph is G'.

2. Construct an instance $\mathcal{I}''$ of $\mathcal{P}$ from $\mathcal{I}'$ as follows: For all edges in G', their length is fixed to their length in $\mathcal{I}'$. In addition, we add a "special" edge from v^* to all vertices besides v with length range $[0, \infty]$[2].
3. Run A on $\mathcal{I}''$ to get a solution SOL. Treat this solution as a solution to $\mathcal{I}'$ (we will show it only uses edges that appear in $\mathcal{I}$). Use the approximation-preserving reduction to convert SOL into a solution for $\mathcal{I}$ and output this solution.

Let O denote the cost of the optimal solution to $\mathcal{I}'$. Then, $\textsc{mr} \le O$. To see why, note that the optimal solution to $\mathcal{I}'$ has cost O in all realizations of demands since it only uses edges of fixed cost, and thus its regret is at most O. This also implies that for all $\mathbf{d}$, $\textsc{opt}(\mathbf{d})$ is finite. Then for all $\mathbf{d}$, $\textsc{sol}(\mathbf{d}) \le \alpha \cdot \textsc{opt}(\mathbf{d}) + \beta \cdot \textsc{mr}$, i.e. $\textsc{sol}(\mathbf{d})$ is finite in all realizations of demands, so SOL does not include any special edges, as any solution with a special edge has infinite cost in some realization of demands.

Now consider the realization of demands $\mathbf{d}$ where all special edges have length 0. The special edges and the edge (v, v^*) span G', so by the third property of $\mathcal{P}'$ in the theorem statement there is a solution using only cost 0 edges in this realization, i.e. $\textsc{opt}(\mathbf{d}) = 0$. Then in this realization, $\textsc{sol}(\mathbf{d}) \le \alpha \cdot \textsc{opt}(\mathbf{d}) + \beta \cdot \textsc{mr} \le \beta \cdot O$. But since SOL does not include any special edges, and all edges besides special edges have fixed cost and their cost is the same in $\mathcal{I}''$ as in $\mathcal{I}'$, $\textsc{sol}(\mathbf{d})$ also is the cost of SOL in instance $\mathcal{I}'$, i.e. $\textsc{sol}(\mathbf{d})$ is a β-approximation for $\mathcal{I}'$. Since the reduction from $\mathcal{I}$ to $\mathcal{I}'$ is approximation-preserving, we also get a β-approximation for $\mathcal{I}$.

◀

From [11, 22] we then get the following hardness results:

▶ **Corollary 13.** *Finding an (α, β)-robust solution for Steiner tree where $\beta < 96/95$ is NP-hard.*

▶ **Corollary 14.** *Finding an (α, β)-robust solution for TSP where $\beta < 121/120$ is NP-hard.*

6 Conclusion

In this paper, we designed constant approximation algorithms for the robust Steiner tree and traveling salesman problems. To the best of our knowledge, this is the first instance of robust polynomial-time algorithms being developed for **NP**-complete graph problems. While our approximation bounds for TSP are small constants, that for STT are very large constants. A natural question is whether these constants can be made smaller, e.g. of the same scale as classic approximation bounds for STT. While we did not seek to optimize our constants, obtaining truly small constants for STT appears to be beyond our techniques, and is an interesting open question. More generally, robust algorithms are a key component in the area of optimization under uncertainty that is of much practical and theoretical significance. We hope that our work will lead to more research in robust algorithms for other fundamental problems in combinatorial optimization, particularly in graph algorithms.

[2] We use ∞ to simplify the proof, but it can be replaced with a sufficiently large finite number. For example, the total weight of all edges in G suffices and has small bit complexity.

References

1 H. Aissi, C. Bazgan, and D. Vanderpooten. Complexity of the min–max (regret) versions of min cut problems. *Discrete Optimization*, 5(1):66 – 73, 2008.

2 Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427 – 438, 2009. URL: `http://www.sciencedirect.com/science/article/pii/S0377221708007625`, `doi:https://doi.org/10.1016/j.ejor.2008.09.012`.

3 I. Averbakh and Oded Berman. Minimax regret p-center location on a network with demand uncertainty. *Location Science*, 5(4):247 – 254, 1997. URL: `http://www.sciencedirect.com/science/article/pii/S0966834998000333`, `doi:https://doi.org/10.1016/S0966-8349(98)00033-3`.

4 Igor Averbakh. On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, 90(2):263–272, Apr 2001.

5 Igor Averbakh. The minmax relative regret median problem on networks. *INFORMS Journal on Computing*, 17(4):451–461, 2005.

6 Igor Averbakh and Oded Berman. Minmax regret median location on a network under uncertainty. *INFORMS Journal on Computing*, 12(2):104–110, 2000. URL: `https://doi.org/10.1287/ijoc.12.2.104.11897`, `arXiv:https://doi.org/10.1287/ijoc.12.2.104.11897`, `doi:10.1287/ijoc.12.2.104.11897`.

7 Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1):49–71, Sep 2003. URL: `https://doi.org/10.1007/s10107-003-0396-4`, `doi:10.1007/s10107-003-0396-4`.

8 Moses Charikar, Chandra Chekuri, and Martin Pál. Sampling bounds for stochastic optimization. In *Proceedings of the 8th International Workshop on Approximation, Randomization and Combinatorial Optimization Problems, and Proceedings of the 9th International Conference on Randamization and Computation: Algorithms and Techniques*, APPROX'05/RANDOM'05, pages 257–269, Berlin, Heidelberg, 2005. Springer-Verlag. URL: `http://dx.doi.org/10.1007/11538462_22`, `doi:10.1007/11538462_22`.

9 André Chassein and Marc Goerigk. On the recoverable robust traveling salesman problem. *Optimization Letters*, 10, 09 2015. `doi:10.1007/s11590-015-0949-5`.

10 Chandra Chekuri. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News*, 38(3):106–129, 2007.

11 Miroslav Chlebík and Janka Chlebíková. Approximation hardness of the Steiner tree problem on graphs. In Martti Penttonen and Erik Meineche Schmidt, editors, *Algorithm Theory — SWAT 2002*, pages 170–179, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

12 Eduardo Conde. On a constant factor approximation for minmax regret problems using a symmetry point scenario. *European Journal of Operational Research*, 219(2):452 – 457, 2012. URL: `http://www.sciencedirect.com/science/article/pii/S0377221712000069`, `doi:https://doi.org/10.1016/j.ejor.2012.01.005`.

13 Kedar Dhamdhere, Vineet Goyal, R. Ravi, and Mohit Singh. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 367–378, 2005.

14 Uriel Feige, Kamal Jain, Mohammad Mahdian, and Vahab S. Mirrokni. Robust combinatorial optimization with exponential scenarios. In *Integer Programming and Combinatorial Optimization, 12th International IPCO Conference, Ithaca, NY, USA, June 25-27, 2007, Proceedings*, pages 439–453, 2007.

15 Navin Goyal, Neil Olver, and F. Bruce Shepherd. The VPN conjecture is true. *J. ACM*, 60(3):17:1–17:17, 2013.

16 Martin Groß, Anupam Gupta, Amit Kumar, Jannik Matuschke, Daniel R. Schmidt, Melanie Schmidt, and José Verschae. A local-search algorithm for Steiner forest. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge,*

MA, USA, pages 31:1–31:17, 2018. URL: `https://doi.org/10.4230/LIPIcs.ITCS.2018.31`, `doi:10.4230/LIPIcs.ITCS.2018.31`.

17 Anupam Gupta, Jon M. Kleinberg, Amit Kumar, Rajeev Rastogi, and Bülent Yener. Provisioning a virtual private network: a network design problem for multicommodity flow. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 389–398, 2001.

18 Anupam Gupta, Viswanath Nagarajan, and R. Ravi. Thresholded covering algorithms for robust and max-min optimization. *Math. Program.*, 146(1-2):583–615, 2014.

19 Anupam Gupta, Viswanath Nagarajan, and R. Ravi. Robust and maxmin optimization under matroid and knapsack uncertainty sets. *ACM Trans. Algorithms*, 12(1):10:1–10:21, 2016.

20 Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Boosted sampling: Approximation algorithms for stochastic optimization. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 417–426, New York, NY, USA, 2004. ACM. URL: `http://doi.acm.org/10.1145/1007352.1007419`, `doi:10.1145/1007352.1007419`.

21 Masahiro Inuiguchi and Masatoshi Sakawa. Minimax regret solution to linear programming problems with an interval objective function. *European Journal of Operational Research*, 86(3):526 – 536, 1995. URL: `http://www.sciencedirect.com/science/article/pii/037722179400092Q`, `doi:https://doi.org/10.1016/0377-2217(94)00092-Q`.

22 Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. In Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam, editors, *Algorithms and Computation*, pages 568–578, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

23 Adam Kasperski and PawełZieliński. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Inf. Process. Lett.*, 97(5):177–180, March 2006. URL: `http://dx.doi.org/10.1016/j.ipl.2005.11.001`, `doi:10.1016/j.ipl.2005.11.001`.

24 Adam Kasperski and Pawel Zieliński. On the existence of an FPTAS for minmax regret combinatorial optimization problems with interval data. *Oper. Res. Lett.*, 35:525–532, 2007.

25 Rohit Khandekar, Guy Kortsarz, Vahab S. Mirrokni, and Mohammad R. Salavatipour. Two-stage robust network design with exponential scenarios. *Algorithmica*, 65(2):391–408, 2013.

26 P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Springer US, 1996.

27 Panos Kouvelis and Gang Yu. *Robust 1-Median Location Problems: Dynamic Aspects and Uncertainty*, pages 193–240. Springer US, Boston, MA, 1997. URL: `https://doi.org/10.1007/978-1-4757-2620-6_6`, `doi:10.1007/978-1-4757-2620-6_6`.

28 Helmut E. Mausser and Manuel Laguna. A new mixed integer formulation for the maximum regret problem. *International Transactions in Operational Research*, 5(5):389 – 403, 1998. URL: `http://www.sciencedirect.com/science/article/pii/S0969601698000239`, `doi:https://doi.org/10.1016/S0969-6016(98)00023-9`.

29 David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM*, 53(6):978–1012, November 2006. URL: `http://doi.acm.org/10.1145/1217856.1217860`, `doi:10.1145/1217856.1217860`.

30 Chaitanya Swamy and David B. Shmoys. Approximation algorithms for 2-stage stochastic optimization problems. *SIGACT News*, 37(1):33–46, 2006.

31 Chaitanya Swamy and David B. Shmoys. Sampling-based approximation algorithms for multistage stochastic optimization. *SIAM J. Comput.*, 41(4):975–1004, 2012.

32 Jens Vygen. New approximation algorithms for the tsp.

33 H. Yaman, O. E. Karaşan, and M. Ç. Pinar. The robust spanning tree problem with interval data. *Operations Research Letters*, 29(1):31 – 40, 2001.

34 P. Zieliński. The computational complexity of the relative robust shortest path problem with interval data. *European Journal of Operational Research*, 158(3):570 – 576, 2004.