Robust Algorithms for TSP and Steiner Tree

ARUN GANESH, UC Berkeley, USA
BRUCE M. MAGGS, Duke University and Emerald Innovations, USA
DEBMALYA PANIGRAHI, Duke University, USA

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.

CCS Concepts: • **Theory of computation → Graph algorithms analysis;**

Additional Key Words and Phrases: Steiner tree, traveling salesman

ACM Reference format:
Arun Ganesh, Bruce M. Maggs, and Debmalya Panigrahi. 2023. Robust Algorithms for TSP and Steiner Tree. *ACM Trans. Algor.* 19, 2, Article 12 (March 2023), 37 pages.
https://doi.org/10.1145/3570957

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?

Arun Ganesh was supported in part by NSF Award CCF-1535989. Bruce M. Maggs was supported in part by NSF Award CCF-1535972. Debmalya Panigrahi was 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.
Authors' addresses: A. Ganesh, UC Berkeley, Soda Hall, Berkeley, California, USA, 94709; email: arunganesh@berkeley.edu; B. M. Maggs, Duke University and Emerald Innovations, 308 Research Drive, Durham, North Carolina, USA, 27710; email: bmm@cs.duke.edu; D. Panigrahi, Duke University, 308 Research Drive, Durham, North Carolina, USA, 27710; email: debmalya@cs.duke.edu.

1549-6325/2023/03-ART12 $15.00
https://doi.org/10.1145/3570957

These questions have led to the field of *robust* graph algorithms. To define a robust graph algorithm, we start with a "standard" optimization problem $\mathcal{P}$ defined by a set system $\mathcal{S} \subseteq 2^E$ over edges E with weights d_e. For example, if $\mathcal{P}$ is the minimum spanning tree problem, $\mathcal{S}$ would be the set of all sets of edges comprising spanning trees. Given these inputs, the goal of the standard version of $\mathcal{P}$ is to find the set $S \in \mathcal{S}$ that minimizes[1] $\sum_{e \in S} d_e$. In the robust version of $\mathcal{P}$, given a range of weights $[\ell_e, u_e]$ for every edge e, we want a solution that is good for all realizations of edge weights simultaneously. To quantify how good a solution is, we define its *regret* as the maximum difference between the algorithm's cost and the optimal cost for any vector of edge weights $\mathbf{d}$. In other words, the regret of SOL is:

$$\max_{\mathbf{d}}(\textsc{sol}(\mathbf{d}) - \textsc{opt}(\mathbf{d})), \qquad \textsc{sol}(\mathbf{d}) := \sum_{e \in \textsc{sol}} d_e, \qquad \textsc{opt}(\mathbf{d}) := \min_{S \in \mathcal{S}} \sum_{e \in S} d_e.$$

Here, $\textsc{sol}(\mathbf{d})$ (resp. $\textsc{opt}(\mathbf{d})$) denotes the cost of SOL (resp. the cost of the optimal solution) in instance $\mathbf{d}$, and $\mathbf{d}$ ranges over all realizable inputs, i.e., inputs such that $\ell_e \le d_e \le u_e$ for all e. We emphasize that SOL is a fixed solution (independent of $\mathbf{d}$) whereas the solution determining $\textsc{opt}(\mathbf{d})$ is dependent on the input $\mathbf{d}$.

Now, the goal of the robust version of $\mathcal{P}$ is to find a solution that minimizes regret. 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 $\mathbf{d}$, $\textsc{sol}(\mathbf{d}) - \textsc{opt}(\mathbf{d}) \le \beta \cdot \textsc{mr}$, i.e., $\textsc{sol}(\mathbf{d}) \le \textsc{opt}(\mathbf{d}) + \beta \cdot \textsc{mr}$.

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: $\textsc{sol}(\mathbf{d}) \le \textsc{opt}(\mathbf{d}) + \beta \cdot \textsc{mr}$, since for $\ell_e = u_e$ (i.e., $\textsc{mr} = 0$), this would imply an exact algorithm for an **NP**-hard optimization problem. Instead, we give guarantees: $\textsc{sol}(\mathbf{d}) \le \alpha \cdot \textsc{opt}(\mathbf{d}) + \beta \cdot \textsc{mr}$, 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 minimizing regret in other **NP**-hard optimization problems in graph algorithms as well as in other domains.

1.1 Related Work

Robust graph algorithms have been extensively studied in the operations research community. It is known that minimizing regret is **NP**-hard for shortest path [25] and minimum cut [1] problems, and using a general theorem for converting exact algorithms to robust ones, 2-approximations are known for these problems [11, 16]. In some cases, better results are known for special classes of graphs, e.g., [17]. Robust **minimum spanning tree (MST)** has also been studied, although in the context of making exponential-time exact algorithms more practical [24]. 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 [4–6, 19], linear programming

[1]We focus on minimization problems over sets of edges in this paper, but one can easily extend the definition to maximization problems and problems over arbitrary set systems.

[14, 20], and other areas [3, 16]. More details can be found in the book by Kouvelis and Yu [18] and the survey by Aissi et al. [2].

Other robust variants of graph optimization where one does not know the edge costs ahead of time 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* [12], 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.

1.2 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.[2] 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 1.1 (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 5). 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 $\text{MR} = 0$ 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 6 for details):

THEOREM 1.2 (APX-HARDNESS). *A hardness of approximation of ρ for TSP (resp., STT) under* $\mathbf{P} \neq \mathbf{NP}$ *implies that it is* $\mathbf{NP}$*-hard to obtain $\alpha \leq \rho$ (irrespective of β) and $\beta \leq \rho$ (irrespective of α) for robust TSP (resp., robust STT).*

[2]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.

1.3 Our Techniques

We now give a sketch of our techniques. Before doing so, we note that for problems in **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 [11, 16]. 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, we show in Section 3 that this is not true in general. In particular, we give a robust TSP instance 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 **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 $\mathcal{P}$ then has the LP relaxation

$$\min\left\{r : \mathbf{x} \in \mathcal{S}; \sum_{e\in E} d_e x_e \le \textsc{opt}(\mathbf{d}) + r,\ \forall \mathbf{d}\right\},$$

where P is the standard polytope for $\mathcal{P}$, 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, i.e., the optimal solution to the LP gives a lower bound for the regret minimization problem.

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:

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

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 a robust graph problem as the instance with edge weights a_e.) Note that these weights are non-negative as $u_e > \ell_e$ and $x_e \ge 0$.

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 [16]. 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 $\alpha', \beta' \geq 1$ 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 $\textsc{mr}$, we can then obtain an (α', β')-robust *fractional* solution using the standard ellipsoid algorithm.

For $\textsc{tsp}$, we show that the above guarantee can be achieved by the classic 2-approximation based on the $\textsc{mst}$ solution of the derived instance. The details appear in Section 3. Although $\textsc{stt}$ also admits a 2-approximation based on the $\textsc{mst}$ 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 $\textsc{sol}$ (i.e., the argument over which Equation (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 $\textsc{sol}$. The second term corresponds to a new $\textsc{stt}$ 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 $\textsc{stt}$ instances (c_1 and c_2 are constants):

$$\text{(i)} \sum_{e \in \textsc{alg} \setminus \textsc{sol}} a_e \leq c_1 \cdot \sum_{e \in \textsc{sol} \setminus \textsc{alg}} a_e \qquad \text{and} \qquad \text{(ii)} \sum_{e \in \textsc{alg}} b_e \leq c_2 \cdot \sum_{e \in \textsc{sol}} b_e.$$

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 $\textsc{alg}$. Obtaining these dual approximation bounds simultaneously forms the most technically challenging part of our work, and is given in Section 5.

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 $\sum_e d_e x_e \leq \alpha' \cdot \textsc{opt}(\mathbf{d}) + \beta' \cdot \textsc{mr}$. Suppose that, ignoring the regret constraint set, the LP we are using has integrality gap at most δ for precise inputs. Then we can bound the difference between the cost of $\textsc{mrs}$ and $\delta\mathbf{x}$ in every realization: Since the integrality gap is at most δ, we have $\delta \cdot \sum_{e \in E} d_e x_e \geq \textsc{opt}(\mathbf{d})$ for any $\mathbf{d}$. This implies that:

$$\textsc{mrs}(\mathbf{d}) - \delta \cdot \sum_{e \in E} d_e x_e \leq \textsc{mrs}(\mathbf{d}) - \textsc{opt}(\mathbf{d}) \leq \textsc{mr}.$$

Hence, the regret of $\textsc{mrs}$ with respect to δx is at most $\textsc{mr}$. Then a natural rounding approach is to try to match this property of $\textsc{mrs}$, i.e., search for an integer solution $\textsc{alg}$ that does not cost much more than $\delta\mathbf{x}$ in any realization. Suppose we choose $\textsc{alg}$ that satisfies:

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

Since $\textsc{alg}$ has minimum regret with respect to $\delta\mathbf{x}$, $\textsc{alg}$'s regret is also at most $\textsc{mr}$. Note that $\delta\mathbf{x}$ is a $(\delta\alpha', \delta\beta')$-robust solution. Hence, $\textsc{alg}$ is a $(\delta\alpha', \delta\beta' + 1)$-robust solution.

If we are solving a problem in **P**, the $\textsc{alg}$ that satisfies Equation (2) is the optimal solution with weights $\max\{\ell_e, u_e - (u_e - \ell_e)\delta x_e\}$ and thus can be found in polynomial time. 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 [16], although we are using a different set of techniques.[3] Of

[3]They obtain $(1, 2)$-robust algorithms by choosing $\textsc{alg}$ as the optimal solution with edge weights $\ell_e + u_e$. For any $\mathbf{d}$, consider $\mathbf{d}'$ with weights $d'_e = u_e + \ell_e - d_e$. By optimality of $\textsc{alg}$, $\textsc{alg}(\mathbf{d}) + \textsc{alg}(\mathbf{d}') \leq \textsc{mrs}(\mathbf{d}) + \textsc{mrs}(\mathbf{d}')$. Rearranging, we get $\textsc{alg}(\mathbf{d}) - \textsc{mrs}(\mathbf{d}) \leq \textsc{mrs}(\mathbf{d}') - \textsc{alg}(\mathbf{d}') \leq \textsc{mr}$, so $\textsc{alg}$'s cost exceeds $\textsc{mrs}$' regret by at most $\textsc{mr}$ in every realization.

course, for **NP**-hard problems, finding a solution ALG that satisfies Equation (2) is **NP**-hard as well. It turns out, however, that we can design a generic rounding algorithm that gives the following guarantee:

THEOREM 1.3. *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 2.1) in Section 2 that applies to a broader class of covering problems including set cover, survivable network design, and so on, and might be useful in future research in this domain.

1.4 Roadmap

We present the general rounding algorithm for robust problems in Section 2. In Section 3, we use this rounding algorithm to give a robust algorithm for the Traveling Salesman problem. Section 4 gives a local search algorithm for the Steiner Tree problem. Both the local search algorithm and the rounding algorithm from Section 2 are then used to give a robust algorithm for the Steiner Tree problem in Section 5. The hardness results for robust problems appear in Section 6. Finally, we conclude with some interesting directions of future work in Section 7.

2 A GENERAL ROUNDING ALGORITHM FOR ROBUST PROBLEMS

In this section we give the rounding algorithm of Theorem 1.3, which is a corollary of the following, more general theorem:

THEOREM 2.1. *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: $\{\min r : \boldsymbol{x} \in X, \boldsymbol{x} \in [0,1]^E, \sum_{e \in E} d_e x_e \leq \textsc{opt}(\boldsymbol{d}) + r \ \forall \boldsymbol{d}\}$.

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: $\forall \boldsymbol{d} : \sum_{e \in E} d_e x_e^* \leq \alpha \cdot \textsc{opt}(\boldsymbol{d}) + \beta \cdot \textsc{mr}$.*

Then, there exists an algorithm that outputs a $(\gamma\delta\alpha, \gamma\delta\beta + \gamma)$-robust SOL for $\mathcal{P}$.

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:

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

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

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

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

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

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

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

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

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

Combined with the previous inequality, this gives:

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

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

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

This implies:

$$\begin{aligned}
\forall \mathbf{d} : \textsc{sol}(\mathbf{d}) = \sum_{e\in S} d_e &\le \delta \sum_{e\in E} d_e x_e^* + \gamma \textsc{mr} + (\gamma-1) \sum_{e\in E} \delta \ell_e x_e^* \\
&\le \delta \sum_{e\in E} d_e x_e^* + \gamma \textsc{mr} + (\gamma-1) \sum_{e\in E} \delta d_e x_e^* \\
&= \gamma\delta \sum_{e\in E} d_e x_e^* + \gamma \textsc{mr} \\
&\le \gamma\delta[\alpha \textsc{opt}(\mathbf{d}) + \beta \textsc{mr}] + \gamma \textsc{mr} \\
&= \gamma\delta\alpha \cdot \textsc{opt}(\mathbf{d}) + (\gamma\delta\beta + \gamma) \cdot \textsc{mr}.
\end{aligned}$$

i.e., $\textsc{sol}$ is $(\gamma\delta\alpha, \gamma\delta\beta + \gamma)$-robust as desired. □

3 ALGORITHM FOR THE ROBUST TRAVELING SALESMAN PROBLEM

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

THEOREM 3.1. *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 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, we

show in Section 3.1 that 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, given in Section 3.3.

3.1 Failure of Double-Tree Algorithm

The black-box reduction of [16] for turning exact algorithms into $(1, 2)$-robust algorithms simply uses the exact algorithm to find the optimal solution when all d_e are set to $\frac{\ell_e+u_e}{2}$ and outputs this solution (see [16] for details on its analysis). We give an example of a robust TSP instance where applying the double-tree algorithm to the $(1, 2)$-robust MST generated by this algorithm does not give a robust TSP solution. Since the doubling of this MST is a 2-approximation for TSP when all d_e are set to $\frac{\ell_e+u_e}{2}$, this example will also show that using an approximation algorithm instead of an exact algorithm in the black-box reduction fails to give any reasonable robustness guarantee as stated in Section 1.

Consider an instance of robust TSP with vertices $V = \{v', v_1 \dots v_n\}$, where there is a "type-1" edge from v' to v_i with length $1 - \epsilon$ for some $\epsilon > \frac{1}{2(n-1)}$, and where there is a "type-2" edge from v_i to v_{i+1} for all i, as well as from v_n to v_1, with length in the range $[0, 2 - \frac{1}{n-1}]$.

Consider MRS, which uses $n-1$ type-2 edges and two type-1 edges to connect v' to the rest of the tour.[4] Its regret is maximized in the realization where all the type-2 edges it is using have length $2 - \frac{1}{n-1}$ and the type-2 edge it is not using has length 0. Note that if a solution contains a type-2 edge of length $2 - \frac{1}{n-1}$, we can replace it with the two type-1 edges it is adjacent to and the cost of the solution decreases since we set $\epsilon > \frac{1}{2(n-1)}$. In turn, the optimum solution for this realization uses the type 2-edge with length 0, the two type-1 edges adjacent to this type-2 edge once, and then the other $n - 2$ type-1 edges twice. So MRS has cost $(n - 1)(2 - \frac{1}{n-1}) + 2(1 - \epsilon) \leq 2(n - 1)$ whereas OPT has cost $2(n - 1)(1 - \epsilon)$. Then, the regret of this solution is at most $n\epsilon$.

When all edge costs are set to $\frac{\ell_e+u_e}{2}$, since $\epsilon > \frac{1}{2(n-1)}$ the minimum spanning tree of the graph is a star centered at v', i.e., all the length $1 - \epsilon$ edges. So the $(1, 2)$-approximation algorithm outputs this tree for MST. Doubling this tree gives a solution to the robust TSP instance that costs $2n(1 - \epsilon)$ in all realizations of demands.

Consider the realization $\mathbf{d}$ where all type-2 edges have length 0. MRS costs $2 - 2\epsilon$ and is also the optimal solution. If the double-tree solution is (α, β)-robust we get that:

$$2n(1 - \epsilon) \leq \alpha \cdot \text{OPT}(\mathbf{d}) + \beta \cdot \text{MR} \leq \alpha \cdot (2 - 2\epsilon) + \beta n\epsilon.$$

Setting ϵ to e.g., $1/n$ gives that one of α, β is $\Omega(n)$.

3.2 LP Relaxation

We use the LP relaxation of robust traveling salesman in Figure 1. This is the standard subtour LP (see e.g., [22]), 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. We discuss in this subsection why the constraints other than the regret constraint set in (3) are identical to the standard TSP

[4]We do not prove this is MRS: Even if it is not, it suffices to upper bound MR by this solution's regret.

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

Fig. 1. The robust TSP polytope.

polytope. This discussion may be skipped without affecting the readability of the rest of the paper.

The standard LP for TSP is the *subtour LP* (see e.g., [22]), which is as follows:

$$\begin{array}{lll}
\min & \sum_{(u,v) \in E} c_{uv} y_{uv} & \\
\text{s.t.} & \forall \emptyset \neq S \subset V: & \sum_{(u,v) \in \delta(S)} y_{uv} \geq 2 \\
& \forall u \in V: & \sum_{(u,v) \in E} y_{uv} = 2 \\
& \forall (u,v) \in E: & 0 \leq y_{uv} \leq 1
\end{array} \tag{4}$$

where $\delta(S)$ denotes the set of edges with one endpoint in S. Note that because the graph is undirected, the order of u and v in terms such as (u, v), c_{uv}, and y_{uv} is immaterial, e.g., there is no distinction between edge (u, v) and edge (v, u) and y_{uv} and y_{vu} denote the same variable. This LP is written for the problem formulation where the triangle inequality holds, and thus we only need to consider tours that are cycles that visit every vertex exactly once. We are concerned, however, with the formulation where the triangle inequality does not necessarily hold, but tours can revisit vertices and edges multiple times. To modify the subtour LP to account for this formulation, we instead let y_{uv} be an indicator variable for whether our solution connects u to v using some path in the graph. Using this definition for y_{uv}, the subtour LP constraints then tell us that we must buy a set of paths such that a set of edges directly connecting the endpoints of the paths would form a cycle visiting every vertex exactly once. Then, we introduce variables $x_{e,u,v}$ denoting that we are using the edge e on the path from u to v. For ease of notation, we let $x_e = \sum_{u,v \in V} x_{e,u,v}$ denote the number of times a fractional solution uses the edge e in paths. We can then use standard constraints from the canonical shortest path LP to ensure that in an integer solution y_{uv} is set to 1 only if for some path from u to v, all edges e on the path have $x_{e,u,v}$ set to 1.

Lastly, note that the optimal tour does not use an edge more than twice. Suppose a tour uses the edge $e = (u, v)$ thrice. By fixing a start/end vertex for the tour, we can split the tour into e, P_1, e, P_2, e, P_3 where P_1 is the part of the tour between the first and second use of e, P_2 is the part of the tour between the second and third use of e, and P_3 is the part of the tour after the third use of e. Because the tour starts and ends at the same vertex (u or v), and each of the three uses of edge e goes from u to v or vice versa, the number of P_1, P_2, and P_3 that go from u to v or vice-versa (as opposed to going from u to u or v to v) must be odd, and hence not zero. Without loss of generality, we can assume P_1 goes from u to v. Then, the tour $\overline{P_1}, P_2, e, P_3$, where $\overline{P_1}$ denotes the reversal of P_1, is a valid tour and costs strictly less than the original tour. So any tour using an edge more than twice is not optimal. This lets us add the constraint $x_e \leq 2$ to the LP without affecting the optimal solution.

This gives the formulation for TSP without triangle inequality but with repeated edges allowed:

$$\begin{array}{lll} \min & \sum_{e\in E} c_e x_e & \\ \text{s.t.} & \forall \emptyset \neq S \subset V: & \sum_{u\in S, v\in V\setminus S} y_{uv} \geq 2 \\ & \forall u \in V: & \sum_{v\neq u} y_{uv} = 2 \\ & \forall \emptyset \neq S \subset V, u \in S, v \in V\setminus S: & \sum_{e\in\delta(S)} x_{e,u,v} \geq y_{uv} \\ & \forall u, v \in V, v \neq u: & 0 \leq y_{uv} \leq 1 \\ & \forall e \in E, u, v \in V, v \neq u: & 0 \leq x_{e,u,v} \leq 1 \\ & \forall e \in E: & x_e \leq 2 \end{array} \tag{5}$$

By integrality of the shortest path polytope, if we let p_{uv} denote the length of the shortest path from u to v, then $\sum_{e\in E, u,v\in V} c_e x_{e,u,v} \geq \sum_{u,v\in V} p_{uv} y_{uv}$. In particular, if we fix the value of y_{uv} the optimal setting of $x_{e,u,v}$ values is to set $x_{e,u,v}$ to y_{uv} for every e on the shortest path from u to v. So (5) without the triangle inequality assumption is equivalent to (4) with the triangle inequality assumption. In particular, the integrality gap of (5) is the same as the integrality gap of (4), which is known to be at most 3/2 [23]. Then, adding a variable r for the fractional solution's regret and the regret constraint set gives (3).

3.3 Approximate Separation Oracle

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 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. This gives the following lemma:

Lemma 3.2. *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 $\forall \boldsymbol{d}$: $\sum_{e\in E} d_e x_e \leq 2 \cdot \text{opt}(\boldsymbol{d}) + r$.*

Proof of Lemma 3.2. RRTSP-Oracle is given in Figure 2. All inequalities except the regret constraint set can be checked exactly by RRTSP-Oracle. Consider the tree T' computed in RRTSP-Oracle and $\mathbf{d}'$ with $d'_e = \ell_e$ for $e \in T'$ and $d'_e = u_e$ for $e \notin T'$. The only other violated inequality RRTSP-Oracle can output is the inequality $\sum_{e\in T'}(\ell_e x_e - 2\ell_e) + \sum_{e\notin T'} u_e x_e \leq r$ in line 5, which is equivalent to $\sum_{e\in E} d'_e x_e \leq 2\sum_{e\in T} d'_e + r$. Since $2\sum_{e\in T} d'_e$ is the cost of a tour in realization $\mathbf{d}'$ (the tour that follows a DFS on the spanning tree T), this inequality is implied by the inequality

Algorithm 1 RRTSP-ORACLE($G(V,E)$, $\{(\ell_e, u_e)\}_{e\in E}$, $(\mathbf{x}, \mathbf{y}, r)$)

Input: Undirected graph $G(V,E)$, lower and upper bounds on edge lengths $\{(\ell_e, u_e)\}_{e\in E}$, solution $(\mathbf{x} = \{x_{e,u,v}\}_{e\in E, u,v\in V}, \mathbf{y} = \{y_{uv}\}_{u,v\in V}, r)$ to (3)

1: Check all constraints of (3), **return** any violated constraint that is found
2: $G' \leftarrow$ copy of G where e has weight $u_e x_e - (\ell_e x_e - 2\ell_e)$
3: $T' \leftarrow$ minimum spanning tree of G'
4: **if** $\sum_{e\in T'}(\ell_e x_e - 2\ell_e) + \sum_{e\notin T'} u_e x_e > r$ **then**
5: **return** $\sum_{e\in T'}(\ell_e x_e - 2\ell_e) + \sum_{e\notin T'} u_e x_e \le r$
6: **else**
7: **return** "Feasible"
8: **end if**

Fig. 2. Separation Oracle for (3).

$\sum_{e\in E} d'_e x_e \le \text{OPT}(\mathbf{d}') + r$ from the regret constraint set. Furthermore, RRTSP-ORACLE only outputs this inequality when it is actually violated.

So, it suffices to show that if there exists $\mathbf{d}$ such that $\sum_{e\in E} d_e x_e > 2\text{OPT}(\mathbf{d}) + r$ then RRTSP-ORACLE outputs a violated inequality on line 5. Since $\text{OPT}(\mathbf{d})$ visits all vertices, it contains some spanning tree T, such that $\text{OPT}(\mathbf{d}) \ge \sum_{e\in T} d_e$. Combining these inequalities gives

$$\sum_{e\in E} d_e x_e > 2\sum_{e\in T} d_e + r.$$

Since all x_e are at most 1, setting $d_e = \ell_e$ for $e \in T$ and $d_e = u_e$ otherwise can only increase $\sum_{e\in E} d_e x_e - 2\sum_{e\in T} d_e$, so

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

Then RRTSP-ORACLE finds a minimum spanning tree T' on G', i.e., T' such that

$$\sum_{e\in T'}(u_e x_e - (\ell_e x_e - 2\ell_e)) \le \sum_{e\in T}(u_e x_e - (\ell_e x_e - 2\ell_e)).$$

which combined with the previous inequality gives

$$\sum_{e\in E} u_e x_e - \sum_{e\in T'}(u_e x_e - (\ell_e x_e - 2\ell_e)) > r \implies \sum_{e\in T'}(\ell_e x_e - 2\ell_e) + \sum_{e\notin T'} u_e x_e > r.$$

□

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 1.3 as well as the fact that the TSP polytope has integrality gap 3/2 (see e.g., [22]) and the TSP problem has a 3/2-approximation gives Theorem 3.1.

4 A LOCAL SEARCH ALGORITHM FOR STEINER TREE

In this section, we describe a local search algorithm for the Steiner tree, given in [13]. By a simplified version of the analysis appearing in [13], we show that the algorithm is 4-approximate. As with many local search algorithms, this algorithm could run in superpolynomial time in the worst case. Standard tricks can be used to modify this algorithm into a polynomial time $(4+\epsilon)$-approximation. This algorithm will serve as a primitive in the algorithms we design in Section 5.

The local moves considered by the algorithm are all *path swaps*, defined as follows: If the current Steiner tree is T, the algorithm can pick any two vertices u, v in T such that there exists a path from

u to v where all vertices except u and v on the path are not in T (and thus all edges on the path are not in T). The algorithm may take any path f from u to v of this form. It suffices to consider only the shortest path of this form. The path f is added to T, inducing a cycle. The algorithm then picks a subpath a in the cycle and removes it from T, while maintaining that T is a feasible Steiner tree. It suffices to consider only maximal subpaths. These are just the subpaths formed by splitting the cycle at every vertex with degree at least 3 in $T \cup \{f\}$. Let n denote the number of nodes and m the number of edges in the graph. Since there are at most $\binom{n}{2}$ pairs of vertices u, v, and a shortest path f between u and v can be found in $O(m + n \log n)$ time, and all maximal subpaths in the induced cycle in $T \cup \{f\}$ can be found in $O(n)$ time, if there is a move that improves the cost of T, we can find it in polynomial time.

We will use the following lemma to show the approximation ratio.

Lemma 4.1. *For any tree the fraction of vertices with degree at most 2 is strictly greater than* $1/2$.

Proof. This follows from a Markov bound on the random variable X defined as the degree of a uniformly random vertex minus one. A tree with n vertices has average degree $\frac{2n-2}{n} < 2$, so $\mathbb{E}[X] < 1$. In turn, the fraction of vertices with degree 3 or greater is $\Pr[X \geq 2] < 1/2$. □

Theorem 4.2. *Let A be a solution to an instance of Steiner tree such that no path-swap reduces the cost of A. Then A is a 4-approximation.*

Proof. Consider any other solution F to the Steiner tree instance. We partition the edges of A into the subpaths such that these subpaths' endpoints are (i) vertices with degree 3 or larger in A, or (ii) vertices in A and F (which might also have degree 3 or larger in A). Besides the endpoints, all other vertices in each subpath have degree 2 and are in A but not in F. Note that a vertex may appear as the endpoint of more than one subpath. Note also that the set of vertices in F includes the terminals, which, without loss of generality includes all leaves in A. This along with condition (i) for endpoints ensures the partition into subpaths is well-defined, i.e., if a subpath ends at a leaf of A, that leaf is in F.

We also decompose F into subpaths, but some edges may be contained in two of these subpaths. To decompose F into subpaths, we first partition the edges of F into maximal connected subgraphs of F whose leaves are vertices in A (including terminals) and whose internal vertices are not in A. Note that some vertices may appear in more than one subgraph, e.g., an internal vertex in F that is in A becomes a leaf in multiple subgraphs. Since these subgraphs are not necessarily paths, we next take any DFS walk on each of these subgraphs starting from one of their leaves (that is, one of the vertices in A). We take the route traversed by the DFS walk, and split it into subpaths at every point where the DFS walk reaches a leaf. This now gives a set of subpaths in F such that each subpaths' endpoints are vertices in A, no other vertices on a subpath are in A, and no edge appears in more than two subpaths. Let $\mathcal{A}$ and $\mathcal{F}$ denote the set of subpaths we decomposed A and F into, respectively.

For $a \in \mathcal{A}$ let $N(a) \subseteq \mathcal{F}$ be the set of subpaths f in $\mathcal{F}$ such that $A \setminus a \cup f$ is a feasible Steiner tree, i.e., f can be swapped for a, and let $N(X) = \cup_{a \in X} N(a)$. We will show that for any $X \subseteq \mathcal{A}$, $|N(X)| \geq \frac{1}{2}|X|$. By an extension of Hall's Theorem (Fact 15 in [13]) this implies the existence of a weight function $\alpha : \mathcal{A} \times \mathcal{F} \to \mathbb{R}^+$ such that:

(1) $\alpha(a, f) > 0$ only if f can be swapped for a
(2) For all subpaths $a \in \mathcal{A}$, $\sum_{f \in N(a)} \alpha(a, f) = 1$.
(3) For all subpaths $f \in \mathcal{F}$, $\sum_{a \in N^{-1}(f)} \alpha(a, f) \leq 2$.

Minimize r subject to

$$\forall S \subset V \text{ such that } \emptyset \subset S \cap T \subset T: \quad \textstyle\sum_{e \in \delta(S)} x_e \geq 1 \tag{6}$$

$$\forall \mathbf{d} \text{ such that } d_e \in [\ell_e, u_e]: \quad \textstyle\sum_{e \in E} d_e x_e \leq \textsc{opt}(\mathbf{d}) + r \tag{7}$$

$$\forall e \in E: \quad x_e \in [0, 1] \tag{8}$$

Fig. 3. The robust Steiner tree polytope.

This weight function then gives:

$$c(A) = \sum_{a \in \mathcal{A}} c(a) = \sum_{a \in \mathcal{A}} \sum_{f \in N(a)} c(a)\alpha(a, f) \leq \sum_{f \in \mathcal{F}} \sum_{a \in N^{-1}(f)} c(f)\alpha(a, f) \leq \sum_{f \in \mathcal{F}} 2c(f) \leq 4c(F),$$

where $N^{-1}(f) = \{a \in \mathcal{A} : f \in N(a)\}$. The first inequality holds by the assumption in the lemma statement that no swaps reduce the cost of A, so for $a \in \mathcal{A}$ and $f \in N(a)$, $c(a) \leq c(f)$. The last inequality follows by the fact that every edge in F appears in at most two subpaths in $\mathcal{F}$.

We now turn towards proving that for any $X \subseteq \mathcal{A}$, $|N(X)| \geq \frac{1}{2}|X|$. Fix any $X \subseteq \mathcal{A}$. Suppose that we remove all of the edges on paths $a \in X$ from A, and also remove all vertices on these paths except their endpoints. After removing these nodes and edges, we are left with $|X| + 1$ connected components. Let T' be a tree with $|X| + 1$ vertices, one for each of these connected components, with an edge between any pair of vertices in T' whose corresponding components are connected by a subpath $a \in X$. Consider any vertices of T' with degree at most 2. We claim the corresponding component contains a vertex in F. Let V' denote the set of vertices in the corresponding component that are endpoints of subpaths in $\mathcal{A}$. There must be at least one such vertex in V'. Furthermore, it is not possible that all of the vertices in V' are internal vertices of A with degree at least 3, since at most two subpaths leave this component and there are no cycles in A. The only other option for endpoints is vertices in F, so this component must contain some vertex in F.

Applying Lemma 4.1, strictly more than $(|X| + 1)/2$ (i.e., at least $|X|/2 + 1$) of the components have degree at most 2, and by the previous argument contain a vertex in F. These vertices are connected by F, and since each subpath in $\mathcal{F}$ does not have internal vertices in A, no subpath in $\mathcal{F}$ passes through more than two of these components. Hence, at least $|X|/2$ of the subpaths in $\mathcal{F}$ have endpoints in two different components because at least $|X|/2$ edges are required to connect $|X|/2 + 1$ vertices. In turn, any of these $|X|/2$ paths f could be swapped for one of the subpaths $a \in X$ that is on the path between the components containing f's endpoints. This shows that $|N(X)| \geq |X|/2$ as desired. □

5 ALGORITHM FOR THE ROBUST STEINER TREE PROBLEM

In this section, our goal is to find a fractional solution to the LP in Figure 3 for the robust Steiner tree. By Theorem 1.3 and known approximation/integrality gap results for Steiner Tree, this will give the following theorem:

Theorem 5.1. *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. So, we focus on this section in deriving an approximate separation oracle. Doing so is the most technically difficult part of the paper, so we break the section up into multiple parts as follows: In Section 5.1, we start with a simpler case where $\ell_e = 0$ for all edges, and show how the local search algorithm of the previous section can help design the separation oracle in this case. In Section 5.2, we state our main technical lemma (Lemma 5.5), give a high-level overview of its proof, and show how it implies Theorem 5.1.

In Section 5.3, we give the algorithm of Lemma 5.5. In Section 5.4, we analyze this algorithm's guarantees, deferring some proofs that are highly technical and already covered at a high level in Section 5.2. In Section 5.5, we give the deferred proofs.

We believe the high-level overview in Section 5.2 captures the main ideas of Sections 5.3, 5.4, and 5.5, and thus a reader who just wants to understand the algorithm and analysis at a high level can stop reading after Section 5.2. A reader who wants a deeper understanding of the algorithm's design choices and implementation but is not too concerned with the details of the analysis can stop reading after Section 5.3. A reader who wants a deeper understanding of the analysis can stop reading after Section 5.4 and still have a strong understanding of the analysis.

5.1 Special Case where the Lower Bounds on All Edge Lengths Are $\ell_e = 0$

In this section, we give a simple algorithm/analysis for the special case when $\ell_e = 0$ for all edges.

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 5.2. *For any $\epsilon > 0$, there exists a polynomial-time algorithm for the Steiner tree problem such that if* 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* ALG *satisfies $c(\text{ALG} \setminus \text{OPT}) \leq (4 + \epsilon) \cdot c(\text{OPT} \setminus \text{ALG})$.*

Proof. The algorithm we use is the local search algorithm described in Section 4, which finds ALG such that $c(\text{ALG}) \leq 4 \cdot c(\text{OPT})$. Suppose that the cost of each edge $e \in \text{ALG} \cap \text{OPT}$ is now changed from its initial value to 0. After this change, ALG remains locally optimal because for every feasible solution F that can be reached by making a local move from ALG, the amount by which the cost of ALG has decreased by setting edge costs to zero is at least the amount by which F has decreased. Hence, no local move causes a decrease in cost. Thus, ALG remains a 4-approximation, which implies that $c(\text{ALG} \setminus \text{OPT}) \leq 4 \cdot c(\text{OPT} \setminus \text{ALG})$.

We also need to show that the algorithm converges in polynomially many iterations. The authors in [13] achieve this convergence by discretizing all the edge costs to the nearest multiple of $\frac{\epsilon}{kn} c(\text{APX})$ for an initial solution APX such that $c(\text{OPT}) \leq c(\text{APX}) \leq kc(\text{OPT})$ (e.g., a simple way to do so is to start with a solution formed by the union of shortest paths between terminals, and then remove edges which cause cycles arbitrarily. This solution has cost between $c(\text{OPT})$ and $n^2 \cdot c(\text{OPT})$. See Section B.3 of [13] for more details). This guarantees that the algorithm converges in $\frac{kn}{\epsilon}$ iterations, at an additive $\epsilon \cdot c(\text{OPT})$ cost. For a standard approximation algorithm this is not an issue, but for an algorithm that aims for a guarantee of the form $c(\text{ALG} \setminus \text{OPT}) \leq O(1) \cdot c(\text{OPT} \setminus \text{ALG})$ an additive $\epsilon \cdot c(\text{OPT})$ might be too much.

We modify the algorithm as follows to ensure that it converges in polynomially many iterations: We only consider swapping out a for f if the decrease in cost is at least $\epsilon/4$ times the

Algorithm 2 RRST-ORACLE-ZLB($G(V,E)$, $\{u_e\}_{e\in E}$, $(\mathbf{x}, r)$)

Input: Undirected graph $G(V,E)$, upper bounds on edge lengths $\{u_e\}_{e\in E}$, solution ($\mathbf{x} = \{x_e\}_{e\in E}, r$) to the LP in Fig. 3

Check all constraints of the LP in Fig. 3 except the regret constraint set, **return** any violated constraint that is found
$G' \leftarrow$ copy of G where e has cost $u_e x_e$
$T' \leftarrow$ output of algorithm from Lemma 5.2 on G'
if $\sum_{e\notin T'} u_e x_e > r$ **then**
  return $\sum_{e\notin T'} u_e x_e \le r$
else
  return "Feasible"
end if

Fig. 4. Separation Oracle for the LP in Figure 3 when $\ell_e = 0$, $\forall e$.

cost of a, and we always choose the swap of this kind that decreases the cost by the largest amount.[5]

We now show the algorithm converges. Later in the section, we will prove two claims, so for brevity's sake we will not include the proofs of the claims here. The first claim is that as long as $c(\text{ALG} \setminus \text{OPT}) > (4+\epsilon)c(\text{OPT} \setminus \text{ALG})$, there is a swap between ALG and OPT where decrease in cost is at least $\epsilon/4$ times the cost of the path being swapped out, and is at least $\epsilon/4n^2$ times $c(\text{ALG} \setminus \text{OPT})$ (the proof follows similarly to Lemma 5.10 in Section 5.3). The second claim is that in any swap the quantity $c(\text{ALG} \setminus \text{OPT}) - c(\text{OPT} \setminus \text{ALG})$ decreases by the same amount that $c(\text{ALG})$ does (see Lemma 5.12 in Section 5.4).

So, we use $c(\text{ALG}\setminus\text{OPT}) - c(\text{OPT}\setminus\text{ALG})$ as a potential to bound the number of swaps. This potential is initially at most $n \max_e c_e$, is always at least $\min_e c_e$ as long as $c(\text{ALG} \setminus \text{OPT}) > c(\text{OPT} \setminus \text{ALG})$, and each swap decreases it multiplicatively by at least a factor of $(1 - \epsilon/4n^2)$ as long as $c(\text{ALG}\setminus\text{OPT}) > (4+\epsilon)c(\text{OPT}\setminus\text{ALG})$. Thus, the algorithm only needs to make $\frac{\log(n \max_e c_e / \min_e c_e)}{-\log(1-\epsilon/4n^2)}$ swaps to arrive at a solution that is a $(4+\epsilon)$-approximation, which is a polynomial in the input size. □

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 5.2 alone with much better constants than in Theorem 5.1:

LEMMA 5.3. *For any instance of robust Steiner tree for which all $\ell_e = 0$, for every $\epsilon > 0$ there exists an algorithm RRST-ORACLE-ZLB which, given a solution $(\boldsymbol{x}, r)$ to the LP in Figure 3, either:*

- *Outputs a separating hyperplane for the LP in Figure 3, or*
- *Outputs "Feasible", in which case $\boldsymbol{x}$ is feasible for the (non-robust) Steiner tree LP and $\forall \boldsymbol{d}$: $\sum_{e\in E} d_e x_e \le \text{OPT}(\boldsymbol{d}) + (4+\epsilon)r$.*

RRST-ORACLE-ZLB is given in Figure 4. Via the ellipsoid method this gives a $(1, 4+\epsilon)$-robust fractional solution. Using Theorem 1.3, the fact that the integrality gap of the LP we use is 2 [21], and that there is a $(\ln 4 + \epsilon) \approx 1.39$-approximation for Steiner tree [8], with appropriate choice of ϵ we get the following corollary:

[5]Note that $c(a) - c(f)$ and $\frac{c(a)-c(f)}{c(a)}$ are both maximized by maximizing $c(a)$ and minimizing $c(f)$. Any path f from u to v that we consider adding is independent of the paths a we can consider removing, since f by definition does not intersect with our solution. So in finding a swap satisfying these conditions if one exists, it still suffices to only consider swaps between shortest paths f and longest paths a in the resulting cycles as before.

Corollary 5.4. *There exists a* $(2.78, 12.51)$*-robust algorithm for Steiner tree when* $\ell_e = 0$ *for all* $e \in E$.

Proof of Lemma 5.3. All inequalities except the regret constraint set can be checked exactly by RRST-Oracle-ZLB. Consider the tree T' computed in RRST-Oracle-ZLB and $\mathbf{d}'$ with $d'_e = 0$ for $e \in T'$ and $d'_e = u_e$ for $e \notin T'$. The only other violated inequality RRST-Oracle-ZLB can output is the inequality $\sum_{e \notin T'} u_e x_e \leq r$ in line 5, which is equivalent to $\sum_{e \in E} d'_e x_e \leq T'(\mathbf{d}') + r$, an inequality from the regret constraint set. Furthermore, RRST-Oracle-ZLB only outputs this inequality when it is actually violated. So, it suffices to show that if there exists $\mathbf{d}$, sol such that $\sum_{e \in E} d_e x_e > \text{sol}(\mathbf{d}) + (4+\epsilon)r$ then RRST-Oracle-ZLB outputs a violated inequality on line 5, i.e., finds Steiner tree T' such that $\sum_{e \notin T'} u_e x_e > r$.

Suppose there exists $\mathbf{d}$, sol such that $\sum_{e \in E} d_e x_e > \text{sol}(\mathbf{d}) + (4+\epsilon)r$. Let $\mathbf{d}^*$ be the vector obtained from $\mathbf{d}$ by replacing d_e with u_e for edges not in sol and with 0 for edges in sol. Replacing $\mathbf{d}$ with $\mathbf{d}^*$ can only increase $\sum_{e \in E} d_e x_e - \text{sol}(\mathbf{d})$, i.e.,:

$$\sum_{e \notin \text{sol}} u_e x_e = \sum_{e \in E} d^*_e x_e > \text{sol}(\mathbf{d}^*) + (4+\epsilon)r = (4+\epsilon)r. \tag{9}$$

Consider the graph G' made by RRST-Oracle-ZLB. We'll partition the edges into four sets, E_0, E_S, E_T, E_{ST} where $E_0 = E \setminus (\text{sol} \cup T')$, $E_S = \text{sol} \setminus T'$, $E_T = T' \setminus \text{sol}$, $E_{ST} = \text{sol} \cap T'$. Let $c(E')$ for $E' = E_0, E_S, E_T, E_{ST}$ denote $\sum_{e \in E'} u_e x_e$, i.e., the total cost of the edge set E' in G'. Since $\mathbf{d}^*$ has $d_e = 0$ for $e \in \text{sol}$, from (9) we get that $c(E_0) + c(E_T) > (4+\epsilon)r$.

Now note that $\sum_{e \notin T'} u_e x_e = c(E_0) + c(E_S)$. Lemma 5.2 gives that $(4+\epsilon)c(E_S) \geq c(E_T)$. Putting it all together, we get that:

$$\sum_{e \notin T'} u_e x_e = c(E_0) + c(E_S) \geq c(E_0) + \frac{c(E_T)}{4+\epsilon} \geq \frac{c(E_0) + c(E_T)}{4+\epsilon} > \frac{(4+\epsilon)r}{4+\epsilon} = r.$$

□

5.2 General Case for Arbitrary Lower Bounds on Edge Lengths: High Level Overview

In this section, we give our main lemma (Lemma 5.5), a high-level overview of the algorithm and analysis proving this lemma, and show how the lemma implies Theorem 5.1.

In the general case, the approximation guarantee given in Lemma 5.2 alone does not suffice because of the offset of $\sum_{e \in E}(\ell_e - \ell_e x_e)$. 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 c' can then be used to handle the offset.

Lemma 5.5. *Let* G *be a graph with terminals* T *and two sets of edge weights* c *and* c'*. Let* 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* alg*, at least one of which (let us call it* alg_i*) satisfies:*

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

The fact that Lemma 5.5 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 5.5 here, and defer the full details to Section 5.4. 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:

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

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:

$$\min\left\{\frac{4\Gamma-1}{8\Gamma}, \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\right\} - \left(e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)}-1\right) > 0, \tag{10}$$

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 (10) to hold, since as $\Gamma \to \infty$, we have $\zeta' \to 0$, so that

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

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

So, the same intuition holds: as long as we are willing to lose a large enough Γ value, we can make the increase in $c(\textsc{alg})$ 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 giving Lemma 5.2. This analysis shows that if $\textsc{alg}$ is a locally optimal solution, then

$$c(\textsc{alg} \setminus \textsc{sol}) \leq 4 \cdot c(\textsc{sol} \setminus \textsc{alg}),$$

i.e., $\textsc{alg}$ is a 4-difference approximation of $\textsc{sol}$. The contrapositive of this statement is that if $\textsc{alg}$ 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 [13] by weakening the notion of locally optimal. In particular, suppose we define a solution $\textsc{alg}$ to be "approximately" locally optimal if at least half of the (weighted) swaps between paths a in $\textsc{alg} \setminus \textsc{sol}$ and paths f in $\textsc{sol} \setminus \textsc{alg}$ satisfy $c(a) \leq 2c(f)$ (as opposed to $c(a) \leq c(f)$ 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 $\textsc{alg}$ is approximately locally optimal, then

$$c(\textsc{alg} \setminus \textsc{sol}) \leq 16 \cdot c(\textsc{sol} \setminus \textsc{alg}).$$

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

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

The decrease in $c(\textsc{alg})$ due to all of these swaps together is at least $c(\textsc{alg} \setminus \textsc{sol})$ times some constant. In addition, since a swap between a and f increases $c'(\textsc{alg})$ by at most $c'(f)$, the total increase in c' due to these swaps is at most $c'(\textsc{sol} \setminus \textsc{alg})$ 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 details are needed to formalize these arguments. Moreover, we also need to show that the algorithm runs in polynomial time.

We now formally define our separation oracle RRST-Oracle in Figure 5 and prove that it is an approximate separation oracle in the lemma below:

Lemma 5.6. *Fix any $\Gamma' > 1, \kappa > 0, 0 < \epsilon < 4/35$ and let Γ be the constant given in Lemma 5.5. 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 Figure 3 either:*

- *Outputs a separating hyperplane for the LP in Figure 3, or*
- *Outputs "Feasible", in which case $\boldsymbol{x}$ is feasible for the (non-robust) Steiner tree LP and*

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

Algorithm 3 RRST-ORACLE($G(V,E)$, $\{[\ell_e, u_e]\}_{e\in E}$, $(\mathbf{x}, r)$)

Input: Undirected graph $G(V,E)$, lower and upper bounds on edge lengths $\{[\ell_e, u_e]\}_{e\in E}$, solution $(\mathbf{x} = \{x_e\}_{e\in E}, r)$ to the LP in Fig. 3

1: Check all constraints of the LP in Fig. 3 except regret constraint set, **return** any violated constraint that is found
2: $G' \leftarrow$ copy of G where $c_e = u_e x_e - \ell_e x_e + \ell_e$, $c'_e = \ell_e - \ell_e x_e$
3: ALG $\leftarrow$ output of algorithm from Lemma 5.5 on G'
4: **for** $\text{ALG}_i \in$ ALG **do**
5: **if** $\sum_{e\notin \text{ALG}_i} u_e x_e + \sum_{e\in\text{ALG}_i} \ell_e x_e - \sum_{e\in\text{ALG}_i} \ell_e > r$ **then**
6: **return** $\sum_{e\notin \text{ALG}_i} u_e x_e + \sum_{e\in\text{ALG}_i} \ell_e x_e - \sum_{e\in\text{ALG}_i} \ell_e \le r$
7: **end if**
8: **end for**
9: **return** "Feasible"

Fig. 5. Separation Oracle for LP in Figure 3.

PROOF. It suffices to show that if there exists $\mathbf{d}$, 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 $e \in$ SOL and $d'_e = u_e$ when $e \notin$ 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 5.5 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 $c(E') = \sum_{e\in E'}(u_e x_e - \ell_e x_e + \ell_e)$, 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:

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

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:

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

Since $\beta = 4\Gamma$, Lemma 5.5 along with an appropriate choice of ϵ gives that $c(E_T) \le \beta c(E_S)$, 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:

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

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 5.5 gives that

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

□

By using Lemma 5.6 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 1.3 and known approximation/integrality gap results give us the following theorem, which with an appropriate choice of constants gives Theorem 5.1:

THEOREM 5.7. *Fix any $\Gamma' > 1, \kappa > 0, 0 < \epsilon < 4/35$ and let Γ be the constant given in Lemma 5.5. 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.*

PROOF OF THEOREM 5.7. By using the ellipsoid method with Lemma 5.6 we can compute a feasible (α,β)-robust fractional solution to the Steiner tree LP (as the robust Steiner tree LP has optimum at most MR). Then, the theorem follows from Theorem 1.3, and the fact that the polytope in Figure 3 has integrality gap $\delta = 2$ and there is a $\gamma = (\ln 4 + \epsilon)$-approximation for the Steiner tree

problem due to [8] (The error parameters can be rescaled appropriately to get the approximation guarantee in the theorem statement). □

Optimizing for α in Theorem 5.7 subject to the constraints in (10), we get that for a fixed (small) ϵ, α is minimized by setting $\Gamma \approx 9.284 + f_1(\epsilon), \Gamma' \approx 5.621 + f_2(\epsilon), \kappa \approx 2.241 + f_3(\epsilon)$ (for monotonically increasing f_1, f_2, f_3 which approach 0 as ϵ approaches 0). Plugging in these values gives Theorem 5.1.

5.3 Algorithm Description

In this section we give the algorithm description for DoubleApprox, as well as a few lemmas that motivate our algorithm's design and certify it is efficient. We will again use local search to find moves that are improving with respect to c. However, now our goal is to show that we can do this without blowing up the cost with respect to c'. We can start to show this via the following lemma, which generalizes the arguments in Section 4. Informally, it says that as long as a significant fraction $(1/\theta)$ of the swaps (rather than all the swaps) that the local search algorithm can make between its solution A and an adversarial solution F do not improve its objective by some factor λ (rather than by any amount at all), A's cost can still be bounded by $4\lambda\theta$ times F's cost.

From Lemma 5.8 to Lemma 5.11 we will refer to the cost functions on edges by w, w' instead of c, c'. This is because these lemmas are agnostic to the cost functions they are applied to and will be applied with both $w = c, w' = c'$ and $w = c', w' = c$ in our algorithm. We also define $\mathcal{A}, \mathcal{F}, \alpha, N(\cdot), N^{-1}(\cdot)$ as in the proof of Theorem 4.2 for these lemmas.

Lemma 5.8. *Let A and F be solutions to an instance of Steiner tree with edge costs w such that if all edges in $A \cap F$ have their costs set to 0, then for $\lambda \geq 1, \theta \geq 1$, we have*

$$\sum_{a \in \mathcal{A}, f \in N(a): w(a) \leq \lambda w(f)} \alpha(a, f) w(a) \geq \frac{1}{\theta} \sum_{a \in \mathcal{A}, f \in N(a)} \alpha(a, f) w(a).$$

Then $w(A \setminus F) \leq 4\lambda\theta w(F \setminus A)$.

Proof. This follows by generalizing the argument in the proof of Theorem 4.2. After setting costs of edges in $A \cap F$ to 0, note that $w(A) = w(A \setminus F)$ and $w(F) = w(F \setminus A)$. Then:

$$\begin{aligned}
w(A \setminus F) &= \sum_{a \in \mathcal{A}} w(a) \\
&\leq \sum_{a \in \mathcal{A}} \sum_{f \in N(a)} \alpha(a, f) w(a) \\
&= \sum_{f \in F} \sum_{a \in N^{-1}(f)} \alpha(a, f) w(a) \\
&\leq \theta \sum_{f \in F} \sum_{a \in N^{-1}(f): w(a) \leq \lambda w(f)} \alpha(a, f) w(a) \\
&\leq \lambda\theta \sum_{f \in F} \sum_{a \in N^{-1}(f): w(a) \leq \lambda w(f)} \alpha(a, f) w(f) \\
&\leq \lambda\theta \sum_{f \in F} \sum_{a \in N^{-1}(f)} \alpha(a, f) w(f) \leq 4\lambda\theta w(F \setminus A).
\end{aligned}$$

□

Corollary 5.9. *Let A, F be solutions to an instance of Steiner tree with edge costs w such that for parameters $\lambda \geq 1, \theta \geq 1$, $w(A \setminus F) > 4\lambda\theta w(F \setminus A)$. Then after setting the cost of all edges in $A \cap F$ to 0,*

$$\sum_{a \in \mathcal{A}, f \in N(a): w(a) > \lambda w(f)} \alpha(a, f) w(a) > \frac{\theta - 1}{\theta} \sum_{a \in \mathcal{A}, f \in N(a)} \alpha(a, f) w(a).$$

The corollary effectively tells us that if $w(A \setminus F)$ is sufficiently larger than $w(F \setminus A)$, then there are many local swaps between S in A and f in F that decrease $w(A)$ by a large fraction of $w(a)$. The next lemma then shows that one of these swaps also does not increase $w'(A)$ by a large factor (even if instead of swapping in f, we swap in an approximation of f), and reduces $w(A)$ by a non-negligible amount.

Lemma 5.10. *Let A and F be solutions to an instance of Steiner tree with two sets of edge costs, w and w', such that for parameter $\Gamma > 1$, $w(A \setminus F) > 4\Gamma \cdot w(F \setminus A)$. Fix any $0 < \epsilon < \sqrt{\Gamma} - 1$. Then there exists a swap between $a \in \mathcal{A}$ and a path f between two vertices in A such that $\frac{(1+\epsilon)w'(f) - w'(a)}{w(a) - (1+\epsilon)w(f)} \leq \frac{4(1+\epsilon)\Gamma}{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)} \cdot \frac{w'(F \setminus A)}{w(A \setminus F)}$ and $w(a) - (1+\epsilon)w(f) \geq \frac{1}{n^2} w(A \setminus F)$.*

Proof. We use an averaging argument to prove the lemma. Consider the quantity

$$R = \frac{\sum_{a \in \mathcal{A}, f \in N(a): w(a) > \sqrt{\Gamma} w(f), w(a) - (1+\epsilon) w(f) \geq \frac{1}{n^2} w(A \setminus F)} \alpha(a, f)[(1+\epsilon)w'(f) - w'(a)]}{\sum_{a \in \mathcal{A}, f \in N(a): w(a) > \sqrt{\Gamma} w(f), w(a) - (1+\epsilon) w(f) \geq \frac{1}{n^2} w(A \setminus F)} \alpha(a, f)[w(a) - (1+\epsilon)w(f)]},$$

which is the ratio of the weighted average of increase in c' to the weighted average of decrease in c over all swaps where $w(a) > \sqrt{\Gamma} w(f)$ and $w(a) - (1+\epsilon)w(f) \geq \frac{1}{n^2} w(A \setminus F)$.

For any edge e in $A \cap F$, it is also a subpath $f \in \mathcal{A} \cap \mathcal{F}$ for which the only $a \in \mathcal{A}$ such that $A \cup f \setminus a$ is feasible is $a = f$. So for all such e we can assume that α is defined such that $\alpha(e, e) = 1$, $\alpha(e, f) = 0$ for $f \neq e$, and $\alpha(a, e) = 0$ for $a \neq e$. Clearly $w(a) > \sqrt{\Gamma} w(a)$ does not hold, so no swap with a positive α value in either sum involves edges in $A \cap F$. So we can now set the cost with respect to both c, c' of edges in $A \cap F$ to 0, and doing so does not affect the quantity R.

Then, the numerator can be upper bounded by $4(1+\epsilon)w'(F \setminus A)$. For the denominator, we first observe that

$$\begin{aligned}
&\sum_{a \in \mathcal{A}, f \in N(a): w(a) > \sqrt{\Gamma} w(f), w(a) - (1+\epsilon) w(f) \geq \frac{1}{n^2} w(A \setminus F)} \alpha(a, f)[w(a) - (1+\epsilon)w(f)] \\
&\geq \sum_{a \in \mathcal{A}, f \in N(a): w(a) > \sqrt{\Gamma} w(f)} \alpha(a, f)[w(a) - (1+\epsilon)w(f)] \\
&\quad - \sum_{a \in \mathcal{A}, f \in N(a): w(a) - (1+\epsilon) w(f) < \frac{1}{n^2} w(A \setminus F)} \alpha(a, f)[w(a) - (1+\epsilon)w(f)]. \qquad (11)
\end{aligned}$$

The second term on the right-hand side of (11) is upper bounded by:

$$\begin{aligned}
&\sum_{a \in \mathcal{A}, f \in N(a): w(a) - (1+\epsilon) w(f) < \frac{1}{n^2} w(A \setminus F)} \alpha(a, f)[w(a) - (1+\epsilon)w(f)] \\
&\leq \frac{1}{n^2} \sum_{a \in \mathcal{A}, f \in N(a): w(a) - (1+\epsilon) w(f) < \frac{1}{n^2} w(A \setminus F)} \alpha(a, f) w(A \setminus F) \leq \frac{1}{n} w(A \setminus F).
\end{aligned}$$

The inequality follows because there are at most n different $a \in \mathcal{A}$, and for each one we have $\sum_{f \in F} \alpha(a, f) = 1$. We next use Corollary 5.9 (setting both parameters to $\sqrt{\Gamma}$) to get the following

lower bound on the first term in (11):

$$\begin{aligned}
&\sum_{a\in\mathcal{A}, f\in N(a): w(a)>\sqrt{\Gamma}w(f)} \alpha(a,f)[w(a)-(1+\epsilon)w(f)] \\
&\geq \sum_{a\in\mathcal{A}, f\in N(a): w(a)>\sqrt{\Gamma}w(f)} \alpha(a,f)\left[w(a)-\frac{1+\epsilon}{\sqrt{\Gamma}}w(a)\right] \\
&= \frac{\sqrt{\Gamma}-1-\epsilon}{\sqrt{\Gamma}} \sum_{a\in\mathcal{A}, f\in N(a): w(a)>\sqrt{\Gamma}w(f)} \alpha(a,f)w(a) \\
&\geq \frac{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)}{\Gamma} \sum_{a\in\mathcal{A}, f\in N(a)} \alpha(a,f)w(a) \\
&= \frac{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)}{\Gamma} w(A\setminus F).
\end{aligned}$$

This lower bounds the denominator of R by $(\frac{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)}{\Gamma}-1/n)\cdot w(A\setminus F)$. By properly choosing of ϵ, for sufficiently large n we can ignore the $1/n$ term. Then, combining the bounds implies that R is at most $\frac{4(1+\epsilon')\Gamma}{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon')}\cdot\frac{w'(F\setminus A)}{w(A\setminus F)}$. In turn, one of the swaps being summed over in R satisfies the lemma statement. □

We now almost have the tools to state our algorithm and prove Lemma 5.5. However, the local search process is now concerned with two edge costs, so just considering adding the shortest path with respect to c between each pair of vertices and deleting a subset of vertices in the induced cycle will not suffice. We instead use the following lemma:

Lemma 5.11. *Given a graph $G=(V,E)$ with edge costs w and w', two vertices s and t, and input parameter W', let p be the shortest path from s to t with respect to w whose cost with respect to c' is at most W'. For all $\epsilon>0$, there is a polynomial time algorithm that finds a path from s to t whose cost with respect to w is at most $w(p)$ and whose cost with respect to w' is at most $(1+\epsilon)W'$.*

Proof. If all edge lengths with respect to w' are multiples of Δ, an optimal solution can be found in time $\text{poly}(|V|,|E|,W'/\Delta)$ via dynamic programming: Let $\ell(v,i)$ be the length of the shortest path from s to v with respect to w whose cost with respect to w' is at most $i\cdot\Delta$. Using $\ell(s,i)=0$ for all i and the recurrence $\ell(v,i)\leq\ell(u,i-(w'_{uv}/\Delta))+w_{uv}$ for edge (u,v), we can compute $\ell(v,i)$ for all v,i and use backtracking from $\ell(t,W')$ to retrieve p in $\text{poly}(|V|,|E|,W'/\Delta)$ time.

To get the runtime down to polynomial, we use a standard rounding trick, rounding each w'_e down to the nearest multiple of $\epsilon W'/|V|$. After rounding, the runtime of the dynamic programming algorithm is $\text{poly}(|V|,|E|,\frac{W'}{\epsilon W'/|V|})=\text{poly}(|V|,|E|,\frac{1}{\epsilon})$. Any path has at most $|V|$ edges, and so its cost decreases by at most $\epsilon W'$ in this rounding process, i.e., all paths considered by the algorithm have cost with respect to w' of at most $(1+\epsilon)W'$. Lastly, since p's cost with respect to w' only decreases, $w(p)$ still upper bounds the cost of the shortest path considered by the algorithm with respect to w. □

The idea is to run a local search with respect to c starting with a good approximation with respect to c'. Our algorithm alternates between a "forward" and "backward" phase. In the forward phase, we use Lemma 5.11 to decide which paths can be added to the solution in local search moves. The local search takes any swap that causes both $c(\text{ALG})$ and $c'(\text{ALG})$ to decrease if any exists. Otherwise, it picks the swap between $S\in\text{ALG}$ and f that among all swaps where $c(f)<c(a)$ and

Algorithm 4 DoubleApprox($G = (V, E)$, χ, χ', Γ, Γ', κ)

Input: A graph $G = (V, E)$ with terminal set T and cost functions c, c' for which all c'_e are a multiple of $\frac{\epsilon}{n}\chi'$, χ such that $\chi \in [c(\text{SOL}), (1+\epsilon) \cdot c(\text{SOL})]$, χ' such that $\chi' \in [c'(\text{SOL}), (1+\epsilon) \cdot c'(\text{SOL})]$, constants Γ, Γ', κ.

1: $i \leftarrow 0$
2: $\text{ALG}^{(0)} \leftarrow \alpha$-approximation of optimal Steiner tree with respect to c' for $\alpha < 4\Gamma'$
3: **while** $i = 0$ or $c(\text{ALG}^{(i)}) < c(\text{ALG}^{(i-2)})$ **do**
4: **for** $\rho \in \{\min_e c_e, (1+\epsilon)\min_e c_e, \ldots (1+\epsilon)^{\log_{1+\epsilon} n \frac{\max_e c_e}{\min_e c_e}} \min_e c_e\}$ **do** {Iterate over guesses for $c(\text{ALG}^{(i)} \setminus \text{SOL})$}
5: $\text{ALG}_\rho^{(i+1)} \leftarrow \text{ALG}^{(i)}$ {Forward phase}
6: **while** $c'(\text{ALG}_\rho^{(i+1)}) \leq (4\Gamma' + \kappa)\chi'$ and $c(\text{ALG}_\rho^{(i+1)}) > c(\text{ALG}^{(i)}) - \rho/2$ **do**
7: $(\text{ALG}_\rho^{(i+1)}, stop) \leftarrow \text{GreedySwap}(\text{ALG}_\rho^{(i+1)}, c, c', \chi', \frac{1}{10n^2}\rho)$
8: **if** $stop = 1$ **then**
9: **break** while loop starting on line 6
10: **end if**
11: **end while**
12: $\text{ALG}_\rho^{(i+2)} \leftarrow \text{ALG}_\rho^{(i+1)}$ {Backward phase}
13: **while** $c'(\text{ALG}_\rho^{(i+2)}) \geq 4\Gamma'\chi'$ **do**
14: $(\text{ALG}_\rho^{(i+2)}, \sim) \leftarrow \text{GreedySwap}(\text{ALG}_\rho^{(i+2)}, c', c, \chi, \frac{\epsilon}{n}\chi')$
15: **end while**
16: **end for**
17: $\text{ALG}^{(i+2)} \leftarrow \text{argmin}_{\text{ALG}_\rho^{(i+2)}} c(\text{ALG}_\rho^{(i+2)})$
18: $i \leftarrow i + 2$
19: **end while**
20: **return** all values of $\text{ALG}_\rho^{(i)}$ stored for any value of i, ρ

Fig. 6. Algorithm DoubleApprox, which finds ALG such that $c(\text{ALG} \setminus \text{SOL}) \leq O(1) \cdot c(\text{SOL} \setminus \text{ALG})$ and $c'(\text{ALG}) \leq O(1) \cdot c'(\text{SOL})$.

$c'(f) \leq c'(\text{SOL})$ minimizes the ratio $\frac{c'(f)-c'(a)}{c(a)-c(f)}$ (we assume we know the value of $c'(\text{SOL})$, as we can guess many values, and our algorithm will work for the right value for $c'(\text{SOL})$).

If the algorithm only made swaps of this form, however, $c'(\text{ALG})$ might become a very poor approximation of $c'(\text{SOL})$. To control for this, when $c'(\text{ALG})$ exceeds $(4\Gamma' + \kappa) \cdot c'(\text{SOL})$ for some constant $\Gamma' > 1$, we begin a "backward phase": We take the opposite approach, greedily choosing either swaps that improve both c and c' or that improve c' and minimize the ratio $\frac{c(f)-c(a)}{c'(a)-c'(f)}$, until $c'(\text{ALG})$ has been reduced by at least $\kappa \cdot c'(\text{SOL})$. At this point, we begin a new forward phase.

The intuition for the analysis is as follows: If, throughout a forward phase, $c(\text{ALG} \setminus \text{SOL}) \geq 4\Gamma \cdot c(\text{SOL} \setminus \text{ALG})$, Lemma 5.10 tells us that there is swap where the increase in $c'(\text{ALG})$ will be very small relative to the decrease in $c(\text{ALG})$. (Note that our goal is to reduce the cost of $c(\text{ALG} \setminus \text{SOL})$ to something below $4\Gamma \cdot c(\text{SOL} \setminus \text{ALG})$.) Throughout the subsequent backward phase, we have $c'(\text{ALG}) > 4\Gamma' \cdot c'(\text{SOL})$, which implies $c'(\text{ALG} \setminus \text{SOL}) > 4\Gamma' \cdot c'(\text{SOL} \setminus \text{ALG})$. So Lemma 5.10 also implies that the total increase in $c(\text{ALG})$ will be very small relative to the decrease in $c'(\text{ALG})$. Since the absolute change in $c'(\text{ALG})$ is similar between the two phases, one forward and one backward phase should decrease $c(\text{ALG})$ overall.

The formal description of the backward and forward phase is given as algorithm DoubleApprox in Figure 6. For the lemmas/corollaries in the following section, we implicitly assume that we know

Algorithm 5 GreedySwap(ALG, w, w', χ', ρ)

Input: Solution ALG, cost functions w, w' on the edges, $\chi' \in [w'(\text{SOL}), (1+\epsilon)w'(\text{SOL})]$, minimum improvement per swap ρ

```
swaps ← ∅
for W' ∈ {1, 1+ε, (1+ε)^2, ..., (1+ε)^⌈log_{1+ε} χ'⌉} do
  for s, t ∈ ALG do
    Find a (1+ε)-approximation f of the shortest path f̂ from s to t with respect to w such
    that w'(f̂) ≤ W',
      f̂ ∩ ALG = {s, t} (via Lemma 5.11)
    for all maximal a ⊆ ALG such that ALG ∪ a \ f is feasible, w(a) − w(f) ≥ ρ do
      swaps ← swaps ∪ {(a, f)}
    end for
  end for
end for
if swaps = ∅ then
  return (ALG, 1)
end if
(a*, f*) ← argmin_{(a,f)∈swaps} (w'(f) − w'(a)) / (w(a) − w(f))
return (ALG ∪ a* \ f*, 0)
```

Fig. 7. Algorithm GreedySwap, which finds a swap with the properties described in Lemma 5.10.

values of χ and χ' satisfying the conditions of DoubleApprox. When we conclude by proving Lemma 5.5, we will simply call DoubleApprox for every reasonable value of χ, χ' that is a power of $1+\epsilon$, and one of these runs will have χ, χ' satisfying the conditions. Furthermore, there are multiple error parameters in our algorithm and its analysis. For simplicity of presentation, we use the same value ϵ for all error parameters in the algorithm and its analysis.

5.4 Algorithm Analysis and Proof of Lemma 5.5

In this section we analyze DoubleApprox and give the proof of Lemma 5.5. We skip the proof of some technical lemmas whose main ideas have been covered already. We first make some observations. The first lets us relate the decrease in cost of a solution ALG to the decrease in the cost of ALG $\setminus$ SOL.

Lemma 5.12. *Let ALG, ALG′, SOL be any Steiner tree solutions to a given instance. Then*

$$c(\text{ALG}) - c(\text{ALG}') = [c(\text{ALG} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG})] - [c(\text{ALG}' \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}')].$$

Proof. By symmetry, the contribution of edges in ALG $\cap$ ALG′ and edges in neither ALG nor ALG′ to both the left and right hand side of the equality is zero, so it suffices to show that all edges in ALG $\oplus$ ALG′ contribute equally to the left and right hand side.

Consider any $e \in \text{ALG} \setminus \text{ALG}'$. Its contribution to $c(\text{ALG}) - c(\text{ALG}')$ is $c(e)$. If $e \in \text{ALG} \setminus \text{SOL}$, then e contributes $c(e)$ to $c(\text{ALG}\setminus\text{SOL}) - c(\text{SOL}\setminus\text{ALG})$ and 0 to $-[c(\text{ALG}'\setminus\text{SOL}) - c(\text{SOL}\setminus\text{ALG}')]$. If $e \in \text{ALG}\cap\text{SOL}$, then e contributes 0 to $c(\text{ALG} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG})$ and $c(e)$ to $-[c(\text{ALG}' \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}')]$. So the total contribution of e to $[c(\text{ALG} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG})] - [c(\text{ALG}' \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}')]$ is $c(e)$.

Similarly, consider $e \in \text{ALG}' \setminus \text{ALG}$. Its contribution to $c(\text{ALG}) - c(\text{ALG}')$ is $-c(e)$. If $e \in \text{SOL} \setminus \text{ALG}$, then e contributes $-c(e)$ to $c(\text{ALG}\setminus\text{SOL}) - c(\text{SOL}\setminus\text{ALG})$ and 0 to $[c(\text{ALG}'\setminus\text{SOL}) - c(\text{SOL}\setminus\text{ALG}')]$. If $e \notin \text{SOL}$, then e contributes 0 to $c(\text{ALG}\setminus\text{SOL}) - c(\text{SOL}\setminus\text{ALG})$ and $-c(e)$ to $-[c(\text{ALG}'\setminus\text{SOL}) - c(\text{SOL}\setminus\text{ALG}')]$. So the total contribution of e to $[c(\text{ALG} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG})] - [c(\text{ALG}' \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}')]$ is $-c(e)$. □

Lemma 5.12 is useful because Lemma 5.10 relates the ratio of change in c, c' to $c(\text{ALG} \setminus \text{SOL})$, but it is difficult to track how $c(\text{ALG} \setminus \text{SOL})$ changes as we make swaps that improve $c(\text{ALG})$. For example, $c(\text{ALG} \setminus \text{SOL})$ does not necessarily decrease with swaps that cause $c(\text{ALG})$ to decrease (e.g., consider a swap that adds a light edge not in SOL and removes a heavy edge in SOL). Whenever $c(\text{ALG} \setminus \text{SOL}) \gg c(\text{SOL} \setminus \text{ALG})$ (if this doesn't hold, we have a good approximation and are done), $c(\text{ALG} \setminus \text{SOL})$ and $c(\text{ALG} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG})$ are off by a multiplicative factor that is very close to 1, and thus we can relate the ratio of changes in Lemma 5.10 to $c(\text{ALG} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG})$ instead at a small loss in the constant, and by Lemma 5.12 changes in this quantity are much easier to track over the course of the algorithm, simplifying our analysis greatly.

The next lemma lets us assume that any backward phase uses polynomially many calls to GreedySwap.

Lemma 5.13. *Let χ' be any value such that $\chi' \in [c'(\text{SOL}), (1+\epsilon)c'(\text{SOL})]$, and suppose we round all c'_e up to the nearest multiple of $\frac{\epsilon}{n}\chi'$ for some $0 < \epsilon < 1$. Then any γ-approximation of SOL with respect to c' using the rounded c'_e values is an $\gamma(1+2\epsilon)$-approximation of SOL with respect to c' using the original edge costs.*

Proof. This follows because the rounding can only increase the cost of any solution, and the cost increases by at most $\epsilon\chi' \le \epsilon(1+\epsilon)c'(\text{SOL}) \le 2\epsilon c'(\text{SOL})$. □

Via this lemma, we will assume all c'_e are already rounded. The following two lemmas formalize the intuition given in Section 5.2; in particular, by using bounds on the "rate of exchange", they show that the decrease in c in the forward phase can be lower bounded, and the increase in c in the backward phase can be upper bounded. Their proofs are highly technical and largely follow the intuition given in Section 5.2, so we defer them to the following section.

Lemma 5.14 (Forward Phase Analysis). *For any even i in algorithm DoubleApprox, let ρ be the power of $(1+\epsilon)$ times $\min_e c_e$ such that $\rho \in [c(\text{ALG}^{(i)} \setminus \text{SOL}), (1+\epsilon)c(\text{ALG}^{(i)} \setminus \text{SOL})]$. Suppose all values of $\text{ALG}_\rho^{(i+1)}$ and the final value of $\text{ALG}^{(i)}$ in DoubleApprox satisfy $c(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}) > 4\Gamma \cdot c(\text{SOL} \setminus \text{ALG}_\rho^{(i+1)})$ and $c(\text{ALG}^{(i)} \setminus \text{SOL}) > 4\Gamma \cdot c(\text{SOL} \setminus \text{ALG}^{(i)})$. Then for $0 < \epsilon < 2/3 - 5/12\Gamma$, the final values of $\text{ALG}^{(i)}, \text{ALG}_\rho^{(i+1)}$ satisfy*

$$c\left(\text{ALG}^{(i)}\right) - c\left(\text{ALG}_\rho^{(i+1)}\right) \ge \min\left\{\frac{4\Gamma - 1}{8\Gamma}, \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\right\} \cdot c\left(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}\right).$$

Lemma 5.15 (Backward Phase Analysis). *Fix any even $i+2$ in algorithm DoubleApprox and any value of ρ. Suppose all values of $\text{ALG}_\rho^{(i+2)}$ satisfy $c(\text{ALG}_\rho^{(i+2)} \setminus \text{SOL}) > 4\Gamma \cdot c(\text{SOL} \setminus \text{ALG}_\rho^{(i+2)})$. Let $T = \frac{c'(\text{ALG}_\rho^{(i+1)}) - c'(\text{ALG}_\rho^{(i+2)})}{c'(\text{SOL})}$. Then for*

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

the final values of $\text{ALG}_\rho^{(i+1)}, \text{ALG}_\rho^{(i+2)}$ satisfy

$$c\left(\text{ALG}_\rho^{(i+2)}\right) - c\left(\text{ALG}_\rho^{(i+1)}\right) \le \left(e^{\zeta' T} - 1\right) \cdot c\left(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}\right).$$

By combining the two preceding lemmas, we can show that as long as ALG is a poor difference approximation, a combination of the forward and backward phase collectively decrease $c(\text{ALG}^{(i)})$.

Corollary 5.16. *Fix any positive even value of $i+2$ in algorithm DoubleApprox, and let ρ be the power of $(1+\epsilon)$ times $\min_e c_e$ such that $\rho \in [c(\text{ALG}^{(i)} \setminus \text{SOL}), (1+\epsilon)c(\text{ALG}^{(i)} \setminus \text{SOL})]$. Suppose all values of $\text{ALG}_\rho^{(i+1)}$ and the final value of $\text{ALG}^{(i)}$ in DoubleApprox satisfy $c(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}) > 4\Gamma \cdot c(\text{SOL} \setminus \text{ALG}_\rho^{(i+1)})$*

and $c(\textsc{alg}^{(i)} \setminus \textsc{sol}) > 4\Gamma \cdot c(\textsc{sol} \setminus \textsc{alg}^{(i)})$. *Then for* $0 < \epsilon < 2/3 - 5/12\Gamma$ *and* ζ' *as defined in Lemma 5.15, the final values of* $\textsc{alg}^{(i+2)}, \textsc{alg}^{(i)}$ *satisfy*

$$c(\textsc{alg}^{(i)}) - c(\textsc{alg}^{(i+2)})$$
$$\geq \left[\min\left\{\frac{4\Gamma - 1}{8\Gamma}, \frac{(4\Gamma - 1)(\sqrt{\Gamma} - 1)(\sqrt{\Gamma} - 1 - \epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\right\} - (e^{\zeta'(4\Gamma' + \kappa + 1 + \epsilon)} - 1)\right] \cdot c\left(\textsc{alg}_\rho^{(i+1)} \setminus \textsc{sol}\right).$$

Proof. It suffices to lower bound $c(\textsc{alg}^{(i)}) - c(\textsc{alg}_\rho^{(i+2)})$ for this value of ρ, since $c(\textsc{alg}^{(i)}) - c(\textsc{alg}^{(i+2)})$ must be at least this large. After rescaling ϵ appropriately, we have

$$c'\left(\textsc{alg}_\rho^{(i+1)}\right) - c'\left(\textsc{alg}_\rho^{(i+2)}\right) \leq c'\left(\textsc{alg}_\rho^{(i+1)}\right) \leq (4\Gamma' + \kappa + 1 + \epsilon)c'(\textsc{sol}),$$

because the algorithm can increase its cost with respect to c' by at most $(1+\epsilon)c'(\textsc{sol})$ in any swap in the forward phase (by line 5 of GreedySwap, which bounds the increase $w'(\hat{f}) \leq W \leq (1+\epsilon)w'(\textsc{sol})$), so it exceeds the threshold $(4\Gamma' + \kappa)\chi' \leq (4\Gamma' + \kappa)(1+\epsilon)c'(\textsc{sol})$ on line 13 of DoubleApprox by at most this much. Then applying Lemma 5.14 to $c(\textsc{alg}^{(i)}) - c(\textsc{alg}_\rho^{(i+1)})$ and Lemma 5.15 to $c(\textsc{alg}_\rho^{(i+1)}) - c(\textsc{alg}_\rho^{(i+2)})$ (using $T \leq 4\Gamma' + \kappa + 1 + \epsilon$) gives:

$$c\left(\textsc{alg}^{(i)}\right) - c\left(\textsc{alg}_\rho^{(i+2)}\right) = \left[c\left(\textsc{alg}^{(i)}\right) - c\left(\textsc{alg}_\rho^{(i+1)}\right)\right] + \left[c\left(\textsc{alg}_\rho^{(i+1)}\right) - c\left(\textsc{alg}_\rho^{(i+2)}\right)\right]$$
$$\geq \left[\min\left\{\frac{4\Gamma - 1}{8\Gamma}, \frac{(4\Gamma - 1)(\sqrt{\Gamma} - 1)(\sqrt{\Gamma} - 1 - \epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\right\} - (e^{\zeta'(4\Gamma' + \kappa + 1 + \epsilon)} - 1)\right] \cdot c\left(\textsc{alg}_\rho^{(i+1)} \setminus \textsc{sol}\right).$$

□

Now, we can chain Corollary 5.16 multiple times to show that after sufficiently many iterations of DoubleApprox, if all intermediate values of $\textsc{alg}^{(i)}$ are poor difference approximations, over all these iterations $c(\textsc{alg}^{(i)})$ must decrease multiplicatively by more than $\frac{n \max_e c_e}{\min_e c_e}$, which is a contradiction as this is the ratio between an upper and lower bound on the cost of every Steiner tree. In turn, some intermediate value of $\textsc{alg}^{(i)}$ must have been a good difference approximation:

Lemma 5.17. *Suppose* Γ, Γ', κ, *and* ϵ *are chosen such that for* ζ' *as defined in Lemma 5.15,*

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

and $0 < \epsilon < 2/3 - 5/12\Gamma$. *Let* η *equal*

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

Assume $\eta > 0$ *and let* $I = 2(\lceil \log \frac{n \max_e c_e}{\min_e c_e} / \log(1+\eta) \rceil + 1)$. *Then there exists some intermediate value* $\textsc{alg}^*$ *assigned to* $\textsc{alg}_\rho^{(i)}$ *by the algorithm for some* $i \leq I$ *and* ρ *such that* $c(\textsc{alg}^* \setminus \textsc{sol}) \leq 4\Gamma c(\textsc{sol} \setminus \textsc{alg}^*)$ *and* $c'(\textsc{alg}^*) \leq (4\Gamma' + \kappa + 1 + \epsilon)c'(\textsc{sol})$.

Proof. Let $\Phi(i) := c(\textsc{alg}^{(i)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}^{(i)})$ for even i. Assume that the lemma is false. Since algorithm DoubleApprox guarantees that $c'(\textsc{alg}_\rho^{(i)}) \leq (4\Gamma' + \kappa + 1 + \epsilon)c'(\textsc{sol})$, if the lemma is false it must be that for all i and ρ, $c(\textsc{alg}_\rho^{(i)} \setminus \textsc{sol}) > 4\Gamma c(\textsc{sol} \setminus \textsc{alg}_\rho^{(i)})$. By Corollary 5.16, and the assumption on $\Gamma, \Gamma', \kappa, \epsilon$ in the statement of this lemma, for all i $c(\textsc{alg}^{(i)}) < c(\textsc{alg}^{(i-2)})$, so the while loop on Line 3 of DoubleApprox never breaks. This means that for all even $i \leq I$, $\textsc{alg}^{(i)}$ is assigned a value in DoubleApprox. We will show that this implies that for the final value of $\textsc{alg}^{(I)}$, $\Phi(I) = c(\textsc{alg}^{(I)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}^{(I)}) < \frac{4\Gamma-1}{4\Gamma} \min_e c_e$. The inequality

$c(\text{ALG}^{(I)} \setminus \text{SOL}) > 4\Gamma c(\text{SOL} \setminus \text{ALG}^{(I)})$ implies $c(\text{ALG}^{(I)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}^{(I)}) > \frac{4\Gamma-1}{4\Gamma} c(\text{ALG}^{(I)} \setminus \text{SOL})$. The value of $c(\text{ALG}^{(I)} \setminus \text{SOL})$ must be positive (otherwise $c(\text{ALG}^{(I)} \setminus \text{SOL}) \leq 4\Gamma c(\text{SOL} \setminus \text{ALG}^{(I)})$ trivially), and hence it must be at least $\min_e c_e$. These two inequalities conflict, which implies a contradiction. Hence, the lemma must be true.

We now analyze how the quantity $\Phi(i)$ changes under the assumption that the lemma is false. Of course $\Phi(0) \leq n \max_e c_e$. Lemma 5.12 gives that $\Phi(i) - \Phi(i+2)$ is exactly equal to $c(\text{ALG}^{(i)}) - c(\text{ALG}^{(i+2)})$. For the value of ρ such that $\rho \in [c(\text{ALG}^{(i)} \setminus \text{SOL}), (1+\epsilon)c(\text{ALG}^{(i)} \setminus \text{SOL})]$, by Corollary 5.16 and the assumption that the lemma is false, for even i we have

$$\begin{aligned}
&\Phi(i) - \Phi(i+2) \\
&\geq \left[\min\left\{\frac{4\Gamma-1}{8\Gamma}, \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\right\} - (e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)} - 1)\right] \cdot c(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}) \\
&\geq \left[\min\left\{\frac{4\Gamma-1}{8\Gamma}, \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\right\} - (e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)} - 1)\right] \\
&\qquad \cdot [c(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}_\rho^{(i+1)})]. \qquad (12)
\end{aligned}$$

Lemma 5.15 (using the proof from Corollary 5.16 that $T \leq 4\Gamma' + \kappa + 1 + \epsilon$), Lemma 5.12, and the inequality $c(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}_\rho^{(i+1)}) > \frac{4\Gamma-1}{4\Gamma} c(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL})$ give:

$$\begin{aligned}
&\Phi(i+2) - [c(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}_\rho^{(i+1)})] \\
&\leq (e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)} - 1) c(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}) \\
&< \frac{4\Gamma-1}{4\Gamma}(e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)} - 1)[c(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}^{(i+1)})] \\
\implies &[c(\text{ALG}_\rho^{(i+1)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}_\rho^{(i+1)})] > \frac{1}{1 + \frac{4\Gamma-1}{4\Gamma}(e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)} - 1))} \Phi(i+2).
\end{aligned}$$

Plugging this into (12) gives:

$$\begin{aligned}
\Phi(i+2) &< \left(1 + \frac{\min\{\frac{4\Gamma-1}{8\Gamma}, \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\} - (e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)} - 1)}{1 + \frac{4\Gamma-1}{4\Gamma}(e^{\zeta'(4\Gamma+\kappa+1+\epsilon)} - 1)}\right)^{-1} \Phi(i) \\
&= (1+\eta)^{-1}\Phi(i).
\end{aligned}$$

Applying this inductively gives:

$$\Phi(i) \leq (1+\eta)^{-i/2}\Phi(0) \leq (1+\eta)^{-i/2} n \max_e c_e.$$

Plugging in $i = I = 2(\lceil \log \frac{n \max_e c_e}{\min_e c_e} / \log(1+\eta) \rceil + 1)$ gives $\Phi(I) \leq (1+\eta)^{-1} \min_e c_e < \min_e c_e$ as desired. □

To prove Lemma 5.5, we now just need to certify that it suffices to guess multiple values of χ, χ', and that the algorithm is efficient.

Proof of Lemma 5.5. If we have $\chi \in [c(\text{SOL}), (1+\epsilon) \cdot c(\text{SOL})]$ and $\chi' \in [c'(\text{SOL}), (1+\epsilon) \cdot c'(\text{SOL})]$, and the c'_e values are multiples of $\frac{\epsilon}{n}\chi'$, then the conditions of DoubleApprox are met. As long as (10) holds, that is:

$$\min\left\{\frac{4\Gamma-1}{8\Gamma}, \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\right\} - (e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)} - 1) > 0, \qquad (10)$$

then we have $\eta > 0$ in Lemma 5.17, thus giving the approximation guarantee in Lemma 5.5. For any positive $\epsilon, \kappa, \Gamma'$, there exists a sufficiently large value of Γ for (10) to hold, since as $\Gamma \to \infty$, we have $\zeta' \to 0$, $(e^{\zeta'(4\Gamma'+\kappa+1+\epsilon)} - 1) \to 0$, and $\min\{\frac{4\Gamma-1}{8\Gamma}, \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)\kappa}{16(1+\epsilon)\Gamma^2}\} \to \min\{1/2, \kappa/(4+4\epsilon)\}$, so for any fixed choice of $\epsilon, \kappa, \Gamma'$, a sufficiently large value of Γ causes $\eta > 0$ to hold as desired.

Some value in $\{\min_e c_e, (1+\epsilon)\min_e c_e, \ldots (1+\epsilon)^{\lceil \log_{1+\epsilon} \frac{n \max_e c_e}{\min_e c_e} \rceil} \min_e c_e\}$ satisfies the conditions for χ, and there are polynomially many values in this set. The same holds for χ' in $\{\min_e c'_e, \ldots (1+\epsilon)^{\lceil \log_{1+\epsilon} \frac{n \max_e c'_e}{\min_e c'_e} \rceil} \min_e c'_e\}$. So we can run DoubleApprox for all pairs of χ, χ' (paying a polynomial increase in runtime), and output the union of all outputs, giving the guarantee of Lemma 5.5 by Lemma 5.17. For each χ' we choose, we can round the edge costs to the nearest multiple of $\frac{\epsilon}{n}\chi'$ before running DoubleApprox, and by Lemma 5.13 we only pay an additive $O(\epsilon)$ in the approximation factor with respect to c'. Finally, we note that by setting ϵ appropriately in the statement of Lemma 5.17, we can achieve the approximation guarantee stated in Lemma 5.5 for a different value of ϵ.

Then, we just need to show DoubleApprox runs in polynomial time. Lemma 5.17 shows that the while loop of Line **??** only needs to be run a polynomial number (I) of times. The while loop for the forward phase runs at most $O(n^2)$ times since each call to GreedySwap decreases the cost with respect to c by at least $\frac{1}{10n^2}\rho$, and once the total decrease exceeds $\rho/2$ the while loop breaks. The while loop for the backward phase runs at most $(\kappa+1+\epsilon)\frac{n}{\epsilon}$ times, since the initial cost with respect to c' is at most $(4\Gamma+\kappa+1+\epsilon)\chi'$, the while loop breaks when it is less than $4\Gamma'\chi'$, and each call to GreedySwap improves the cost by at least $\frac{\epsilon}{n}\chi'$. Lastly, GreedySwap can be run in polynomial time as the maximal a which needs to be enumerated and can be computed in polynomial time as described in Section 4. □

5.5 Proofs of Lemmas 5.14 and 5.15

Proof of Lemma 5.14. Let $\text{ALG}_{\rho,j}^{(i+1)}$ denote the value of $\text{ALG}_{\rho}^{(i+1)}$ after j calls to GreedySwap on $\text{ALG}_{\rho}^{(i+1)}$, and let J be the total number of calls of GreedySwap on $\text{ALG}_{\rho}^{(i+1)}$. Then $\text{ALG}_{\rho,0}^{(i+1)}$ is the final value of $\text{ALG}^{(i)}$, and the final value of $\text{ALG}_{\rho}^{(i+1)}$ is $\text{ALG}_{\rho,J}^{(i+1)}$. Any time GreedySwap is invoked on $\text{ALG}_{\rho}^{(i+1)}$, by line 6 of DoubleApprox and the assumption $\rho \le (1+\epsilon)c(\text{ALG}^{(i)} \setminus \text{SOL})$ in the lemma statement, we have:

$$c\left(\text{ALG}_{\rho}^{(i+1)}\right) > c(\text{ALG}^{(i)}) - \rho/2 \ge c(\text{ALG}^{(i)}) - \frac{1+\epsilon}{2}c(\text{ALG}^{(i)} \setminus \text{SOL}).$$

Then, by Lemma 5.12 and the assumption $c(\text{ALG}^{(i)} \setminus \text{SOL}) > 4\Gamma \cdot c(\text{SOL} \setminus \text{ALG}^{(i)})$ in the lemma statement, we have:

$$\begin{aligned} c(\text{ALG}_{\rho}^{(i+1)} \setminus \text{SOL}) &\ge c(\text{ALG}_{\rho}^{(i+1)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}_{\rho}^{(i+1)}) \\ &= c(\text{ALG}^{(i)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}^{(i)}) + c(\text{ALG}_{\rho}^{(i+1)}) - c(\text{ALG}^{(i)}) \\ &\ge c(\text{ALG}^{(i)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}^{(i)}) - \frac{1+\epsilon}{2}c(\text{ALG}^{(i)} \setminus \text{SOL}) \\ &\ge \left(\frac{1-\epsilon}{2} - \frac{1}{4\Gamma}\right)c(\text{ALG}^{(i)} \setminus \text{SOL}), \end{aligned}$$

For $\epsilon < 2/3 - 5/12\Gamma$, $c(\text{ALG}_{\rho}^{(i+1)} \setminus \text{SOL})/n^2 \ge \frac{1}{10n^2}\rho$. So by Lemma 5.10 GreedySwap never outputs a tuple where $stops = 1$, and thus we can ignore lines 8-10 of DoubleApprox under the conditions in the lemma statement.

Suppose $\textsc{alg}_{\rho,J}^{(i+1)}$ satisfies $c(\textsc{alg}_{\rho,J}^{(i+1)}) \le c(\textsc{alg}^{(i)}) - \rho/2$, a condition that causes the while loop at line 6 of DoubleApprox to exit and the forward phase to end. Then

$$\begin{aligned} c(\textsc{alg}^{(i)}) - c(\textsc{alg}_{\rho,J}^{(i+1)}) &\ge \rho/2 \ge \frac{1}{2} c(\textsc{alg}^{(i)} \setminus \textsc{sol}) \\ &\ge \frac{1}{2}[c(\textsc{alg}^{(i)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}^{(i)})] \\ &= \frac{1}{2}[c(\textsc{alg}_{\rho,0}^{(i+1)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}_{\rho,0}^{(i+1)})] \\ &\ge \frac{1}{2}[c(\textsc{alg}_{\rho,J}^{(i+1)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}_{\rho,J}^{(i+1)})] \\ &\ge \frac{4\Gamma - 1}{8\Gamma} c(\textsc{alg}_{\rho,J}^{(i+1)} \setminus \textsc{sol}). \end{aligned}$$

The second-to-last inequality is using Lemma 5.12, which implies $c(\textsc{alg}_{\rho,j}^{(i+1)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+1)})$ is decreasing with swaps, and the last inequality holds by the assumption $c(\textsc{alg}_{\rho}^{(i+1)} \setminus \textsc{sol}) > 4\Gamma \cdot c(\textsc{sol} \setminus \textsc{alg}_{\rho}^{(i+1)})$ in the lemma statement. Thus, if $c(\textsc{alg}_{\rho,J}^{(i+1)}) \le c(\textsc{alg}^{(i)}) - \rho/2$, the lemma holds.

Now assume instead that $c(\textsc{alg}_{\rho,J}^{(i+1)}) > c(\textsc{alg}^{(i)}) - \rho/2$ when the forward phase ends. We want a lower bound on

$$c(\textsc{alg}_{\rho,0}^{(i+1)}) - c(\textsc{alg}_{\rho,J}^{(i+1)}) = \sum_{j=0}^{J-1} [c(\textsc{alg}_{\rho,j}^{(i+1)}) - c(\textsc{alg}_{\rho,j+1}^{(i+1)})].$$

We bound each $c(\textsc{alg}_{\rho,j}^{(i+1)}) - c(\textsc{alg}_{\rho,j+1}^{(i+1)})$ term using Lemma 5.10 and Lemma 5.11. By Lemma 5.10 and the assumption in the lemma statement that $c(\textsc{alg}_{\rho}^{(i+1)} \setminus \textsc{sol}) > 4\Gamma \cdot c(\textsc{sol} \setminus \textsc{alg}_{\rho}^{(i+1)})$, we know there exists a swap between $a \in \textsc{alg}_{\rho,j}^{(i+1)}$ and $f \in \textsc{sol}$ such that

$$\frac{(1+\epsilon)c'(f) - c'(a)}{c(a) - (1+\epsilon)c(f)} \le \frac{4(1+\epsilon)\Gamma}{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)} \cdot \frac{c'(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+1)})}{c(\textsc{alg}_{\rho,j}^{(i+1)} \setminus \textsc{sol})}.$$

By Lemma 5.11, we know that when G' is set to a value in $[c'(f), (1+\epsilon) \cdot c'(f)]$ in line 2 of GreedySwap, the algorithm finds a path f' between the endpoints of f such that $c(f') \le (1+\epsilon)c(f)$ and $c'(f') \le (1+\epsilon)c'(f)$. Thus, $(a, f') \in \mathit{swaps}$ and the swap (a^*, f^*) chosen by the $(j+1)$th call to GreedySwap satisfies:

$$\begin{aligned} \frac{c'(f^*) - c'(a^*)}{c(a^*) - c(f^*)} \le \frac{c'(f') - c'(a)}{c(a) - c(f)} &\le \frac{(1+\epsilon)c'(f) - c'(a)}{c(a) - (1+\epsilon)c(f)} \\ &\le \frac{4(1+\epsilon)\Gamma}{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)} \cdot \frac{c'(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+1)})}{c(\textsc{alg}_{\rho,j}^{(i+1)} \setminus \textsc{sol})}. \end{aligned}$$

Rearranging terms and observing that $c'(\textsc{sol}) \ge c'(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+1)})$ gives:

$$\begin{aligned} c(\textsc{alg}_{\rho,j}^{(i+1)}) - c(\textsc{alg}_{\rho,j+1}^{(i+1)}) &= c(a^*) - c(f^*) \\ &\ge \frac{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)}{4(1+\epsilon)\Gamma} \cdot c(\textsc{alg}_{\rho,j}^{(i+1)} \setminus \textsc{sol}) \frac{c'(f^*) - c'(a^*)}{c'(\textsc{sol})} \\ &= \frac{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)}{4(1+\epsilon)\Gamma} \cdot c(\textsc{alg}_{\rho,j}^{(i+1)} \setminus \textsc{sol}) \frac{c'(\textsc{alg}_{\rho,j+1}^{(i+1)}) - c'(\textsc{alg}_{\rho,j}^{(i+1)})}{c'(\textsc{sol})}. \end{aligned}$$

This in turn gives:

$$
\begin{aligned}
c(\textsc{alg}^{(i+1)}_{\rho,0}) - c(\textsc{alg}^{(i+1)}_{\rho,J}) &= \sum_{j=0}^{J-1}[c(\textsc{alg}^{(i+1)}_{\rho,j}) - c(\textsc{alg}^{(i+1)}_{\rho,j+1})] \\
&\geq \sum_{j=0}^{J-1} \frac{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)}{4(1+\epsilon)\Gamma} \cdot c(\textsc{alg}^{(i+1)}_{\rho,j} \setminus \textsc{sol}) \\
&\qquad \cdot \frac{c'(\textsc{alg}^{(i+1)}_{\rho,j+1}) - c'(\textsc{alg}^{(i+1)}_{\rho,j})}{c'(\textsc{sol})} \\
&\geq \frac{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)}{4(1+\epsilon)\Gamma} \sum_{j=0}^{J-1}[c(\textsc{alg}^{(i+1)}_{\rho,j} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}^{(i+1)}_{\rho,j})] \\
&\qquad \cdot \frac{c'(\textsc{alg}^{(i+1)}_{\rho,j+1}) - c'(\textsc{alg}^{(i+1)}_{\rho,j})}{c'(\textsc{sol})} \\
&\geq \frac{(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)}{4(1+\epsilon)\Gamma} \sum_{j=0}^{J-1}[c(\textsc{alg}^{(i+1)}_{\rho,J} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}^{(i+1)}_{\rho,J})] \\
&\qquad \cdot \frac{c'(\textsc{alg}^{(i+1)}_{\rho,j+1}) - c'(\textsc{alg}^{(i+1)}_{\rho,j})}{c'(\textsc{sol})} \\
&\geq \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)}{16(1+\epsilon)\Gamma^2} c(\textsc{alg}^{(i+1)}_{\rho,J} \setminus \textsc{sol}) \\
&\qquad \cdot \sum_{j=0}^{J-1} \frac{c'(\textsc{alg}^{(i+1)}_{\rho,j+1}) - c'(\textsc{alg}^{(i+1)}_{\rho,j})}{c'(\textsc{sol})} \\
&= \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)}{16(1+\epsilon)\Gamma^2} c(\textsc{alg}^{(i+1)}_{\rho,J} \setminus \textsc{sol}) \\
&\qquad \cdot \frac{c'(\textsc{alg}^{(i+1)}_{\rho,J}) - c'(\textsc{alg}^{(i+1)}_{\rho,0})}{c'(\textsc{sol})} \\
&\geq \frac{(4\Gamma-1)(\sqrt{\Gamma}-1)(\sqrt{\Gamma}-1-\epsilon)\kappa}{16(1+\epsilon)\Gamma^2} c(\textsc{alg}^{(i+1)}_{\rho,J} \setminus \textsc{sol}).
\end{aligned}
$$

The third-to-last inequality is using Lemma 5.12, which implies $c(\textsc{alg}^{(i+1)}_{\rho,j} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}^{(i+1)}_{\rho,j})$ is decreasing with swaps. The second-to-last inequality is using the assumption $c(\textsc{alg}^{(i+1)}_{\rho} \setminus \textsc{sol}) > 4\Gamma \cdot c(\textsc{sol} \setminus \textsc{alg}^{(i+1)}_{\rho})$ in the statement the lemma. The last inequality uses the fact that the while loop on line 6 of DoubleApprox terminates because $c'(\textsc{alg}^{(i+1)}_{\rho,J}) > (4\Gamma' + \kappa)\chi'$ (by the assumption that $c(\textsc{alg}^{(i+1)}_{\rho,J}) > c(\textsc{alg}^{(i)}) - \rho/2$), and lines 2 and 13 of DoubleApprox give that $c'(\textsc{alg}^{(i+1)}_{\rho,0}) \leq 4\Gamma'\chi'$. □

Proof of Lemma 5.15. Because $c'(\textsc{alg}^{(i+2)}_{\rho}) > 4\Gamma'\chi'$ in every backwards phase and $\chi' \geq c'(\textsc{sol})$, by Lemma 5.10 whenever GreedySwap is called on $\textsc{alg}^{(i+2)}_{\rho}$ in line 14 of DoubleApprox, at least one swap is possible. Since all edge costs are multiples of $\frac{\epsilon}{n}\chi'$, and the last argument to GreedySwap is $\frac{\epsilon}{n}\chi'$ (which lower bounds the decrease in $c'(\textsc{alg}^{(i+2)}_{\rho})$ due to any improving swap), GreedySwap always makes a swap.

Let $\textsc{alg}_{\rho,j}^{(i+2)}$ denote the value of $\textsc{alg}^{(i+2)}$ after j calls to GreedySwap on $\textsc{alg}^{(i+2)}$, and let J be the total number of calls of GreedySwap on $\textsc{alg}^{(i+2)}$. Then $\textsc{alg}_{\rho,0}^{(i+2)}$ is the final value of $\textsc{alg}^{(i+1)}$ and the final value of $\textsc{alg}^{(i+2)}$ is $\textsc{alg}_{\rho,J}^{(i+2)}$. We want to show that

$$c(\textsc{alg}_{\rho,J}^{(i+2)}) - c(\textsc{alg}_{\rho,0}^{(i+2)}) = \sum_{j=0}^{J-1}[c(\textsc{alg}_{\rho,j+1}^{(i+2)}) - c(\textsc{alg}_{\rho,j}^{(i+2)})] \leq (e^{\zeta' T} - 1)c(\textsc{alg}_{\rho,0}^{(i+2)} \setminus \textsc{sol}).$$

We bound each $c(\textsc{alg}_{j+1}^{(i+2)}) - c(\textsc{alg}_{j}^{(i+2)})$ term using Lemma 5.10 and Lemma 5.11. Since in a backward phase we have $c'(\textsc{alg}_{\rho}^{(i+2)}) > 4\Gamma' c'(\textsc{sol})$, by Lemma 5.10 we know there exists a swap between $a \in \textsc{alg}_{\rho,j}^{(i+2)}$ and $f \in \textsc{sol}$ such that

$$\frac{(1+\epsilon)c(f) - c(a)}{c'(a) - (1+\epsilon)c'(f)} \leq \frac{4(1+\epsilon)\Gamma'}{(\sqrt{\Gamma'}-1)(\sqrt{\Gamma'}-1-\epsilon)} \cdot \frac{c(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+2)})}{c'(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol})}.$$

By Lemma 5.11, we know that when G' is set to the value in $[c(f), (1+\epsilon)\cdot c(f)]$ in line 2 of GreedySwap, the algorithm finds a path f' between the endpoints of f such that $c'(f') \leq (1+\epsilon)c'(f)$ and $c(f') \leq (1+\epsilon)c(f)$. Thus, $(a, f') \in \mathit{swaps}$ and we get that the swap (a^*, f^*) chosen by the $(j+1)$th call to GreedySwap satisfies:

$$\begin{aligned}
\frac{c(f^*) - c(a^*)}{c'(a^*) - c'(f^*)} &\leq \frac{c(f') - c(a)}{c'(a) - c'(f)} \leq \frac{(1+\epsilon)c(f) - c(a)}{c'(a) - (1+\epsilon)c'(f)} \\
&\leq \frac{4(1+\epsilon)\Gamma'}{(\sqrt{\Gamma'}-1)(\sqrt{\Gamma'}-1-\epsilon)} \cdot \frac{c(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+2)})}{c'(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol})} \\
&\leq \frac{4(1+\epsilon)\Gamma'}{(\sqrt{\Gamma'}-1)(\sqrt{\Gamma'}-1-\epsilon)(4\Gamma'-1)(4\Gamma)} \cdot \frac{c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol})}{c'(\textsc{sol})}.
\end{aligned}$$

The last inequality is derived using the assumption $c(\textsc{alg}_{\rho}^{(i+2)} \setminus \textsc{sol}) > 4\Gamma \cdot c(\textsc{sol} \setminus \textsc{alg}_{\rho}^{(i+2)})$ in the statement of the lemma, as well as the fact that for all $j < J$, $c'(\textsc{alg}_{\rho,j}^{(i+2)}) \geq 4\Gamma c'(\textsc{sol}) \implies c'(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol}) \geq c'(\textsc{alg}_{\rho,j}^{(i+2)}) - c'(\textsc{sol}) \geq (4\Gamma'-1)c'(\textsc{sol})$. This in turn gives:

$$\begin{aligned}
c(\textsc{alg}_{\rho,J}^{(i+2)}) - c(\textsc{alg}_{\rho,0}^{(i+2)}) &= \sum_{j=0}^{J-1}[c(\textsc{alg}_{\rho,j+1}^{(i+2)}) - c(\textsc{alg}_{\rho,j}^{(i+2)})] \\
&= \sum_{j=0}^{J-1} \frac{c(\textsc{alg}_{\rho,j+1}^{(i+2)}) - c(\textsc{alg}_{\rho,j}^{(i+2)})}{c'(\textsc{alg}_{\rho,j}^{(i+2)}) - c'(\textsc{alg}_{\rho,j+1}^{(i+2)})} \cdot [c'(\textsc{alg}_{\rho,j}^{(i+2)}) - c'(\textsc{alg}_{\rho,j+1}^{(i+2)})] \\
&\leq \frac{4(1+\epsilon)\Gamma'}{(\sqrt{\Gamma'}-1)(\sqrt{\Gamma'}-1-\epsilon)(4\Gamma'-1)(4\Gamma)} \sum_{j=0}^{J-1}[c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol})] \\
&\qquad \cdot \frac{c'(\textsc{alg}_{\rho,j}^{(i+2)}) - c'(\textsc{alg}_{\rho,j+1}^{(i+2)})}{c'(\textsc{sol})}
\end{aligned}$$

$$\leq \frac{4(1+\epsilon)\Gamma'}{(\sqrt{\Gamma'}-1)(\sqrt{\Gamma'}-1-\epsilon)(4\Gamma'-1)(4\Gamma-1)} \cdot \sum_{j=0}^{J-1}[c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+2)})] \cdot \frac{c'(\textsc{alg}_{\rho,j}^{(i+2)}) - c'(\textsc{alg}_{\rho,j+1}^{(i+2)})}{c'(\textsc{sol})}$$

$$= \zeta' \sum_{j=0}^{J-1}[c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+2)})] \tag{13}$$

$$\cdot \frac{c'(\textsc{alg}_{\rho,j}^{(i+2)}) - c'(\textsc{alg}_{\rho,j+1}^{(i+2)})}{c'(\textsc{sol})}. \tag{14}$$

The last inequality is proved using the assumption $c(\textsc{alg}_{\rho}^{(i+2)} \setminus \textsc{sol}) > 4\Gamma \cdot c(\textsc{sol} \setminus \textsc{alg}_{\rho}^{(i+2)})$ in the statement of the lemma, which implies

$$c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol}) = \frac{4\Gamma}{4\Gamma-1}c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol}) - \frac{1}{4\Gamma-1}c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol})$$
$$< \frac{4\Gamma}{4\Gamma-1}c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol}) - \frac{4\Gamma}{4\Gamma-1}c(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+2)}).$$

It now suffices to show

$$\sum_{j=0}^{J-1}[c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+2)})] \cdot \frac{c'(\textsc{alg}_{\rho,j}^{(i+2)}) - c'(\textsc{alg}_{\rho,j+1}^{(i+2)})}{c'(\textsc{sol})}$$
$$\leq \frac{e^{\zeta' T}-1}{\zeta'}c(\textsc{alg}_{\rho,0}^{(i+2)} \setminus \textsc{sol}).$$

To do so, we view the series of swaps as occurring over a continuous timeline, where for $j = 0, 1, \ldots J-1$ the $(j+1)$th swap takes time $\tau(j) = \frac{c'(\textsc{alg}_{\rho,j}^{(i+2)}) - c'(\textsc{alg}_{\rho,j+1}^{(i+2)})}{c'(\textsc{sol})}$, i.e., occurs from time $\sum_{j'<j}\tau(j')$ to time $\sum_{j'\leq j}\tau(j')$. The total time taken to perform all swaps in the sum is the total decrease in c' across all swaps, divided by $c'(\textsc{sol})$, i.e., exactly T. Using this definition of time, let $\Phi(t)$ denote $c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+2)})$ for the value of j satisfying $\Phi(t) \in [\sum_{j'<j}\tau(j'), \sum_{j'\leq j}\tau(j'))$. Using this definition, we get:

$$\sum_{j=0}^{J-1}[c(\textsc{alg}_{\rho,j}^{(i+2)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}_{\rho,j}^{(i+2)})] \cdot \frac{c'(\textsc{alg}_{\rho,j}^{(i+2)}) - c'(\textsc{alg}_{\rho,j+1}^{(i+2)})}{c'(\textsc{sol})} = \int_0^{\to T} \Phi(t)\, dt.$$

We conclude by claiming $\Phi(t) \leq e^{\zeta' t}c(\textsc{alg}_{\rho,0}^{(i+2)} \setminus \textsc{sol})$. Given this claim, we get:

$$\int_0^{\to T} \Phi(t)\, dt \leq c(\textsc{alg}_{\rho,0}^{(i+2)} \setminus \textsc{sol}) \int_0^{\to T} e^{\zeta' t}\, dt = \frac{e^{\zeta' T}-1}{\zeta'}c(\textsc{alg}_{\rho,0}^{(i+2)} \setminus \textsc{sol}).$$

Which completes the proof of the lemma. We now focus on proving the claim. Since $\Phi(t)$ is fixed in the interval $[\sum_{j'<j}\tau(j'), \sum_{j'\leq j}\tau(j'))$, it suffices to prove the claim only for t which are equal to $\sum_{j'<j}\tau(j')$ for some j, so we proceed by induction on j. The claim clearly holds for $j = 0$ since $\sum_{j'<0}\tau(j') = 0$ and $\Phi(0) = c(\textsc{alg}_{\rho,0}^{(i+2)} \setminus \textsc{sol}) - c(\textsc{sol} \setminus \textsc{alg}_{\rho,0}^{(i+2)}) \leq c(\textsc{alg}_{\rho,0}^{(i+2)} \setminus \textsc{sol})$.

Assume that for $t' = \sum_{j'<j} \tau(j')$, we have $\Phi(t') \le e^{\zeta' t'} c(\text{ALG}_{\rho,0}^{(i+2)} \setminus \text{SOL})$. For $t'' = t' + \tau(j)$, by induction we can prove the claim by showing $\Phi(t'') \le e^{\zeta' \tau(j)} \Phi(t')$.

To show this, we consider the quantity

$$\begin{aligned}\Phi(t'') - \Phi(t') &= [c(\text{ALG}_{\rho,j+1}^{(i+2)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}_{\rho,j+1}^{(i+2)})] - [c(\text{ALG}_{\rho,j}^{(i+2)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}_{\rho,j}^{(i+2)})] \\ &= [c(\text{ALG}_{\rho,j+1}^{(i+2)} \setminus \text{SOL}) - c(\text{ALG}_{\rho,j}^{(i+2)} \setminus \text{SOL})] + [c(\text{SOL} \setminus \text{ALG}_{\rho,j}^{(i+2)}) - c(\text{SOL} \setminus \text{ALG}_{\rho,j+1}^{(i+2)})].\end{aligned}$$

By Lemma 5.12 and reusing the bound in (14), we have:

$$\begin{aligned}\Phi(t'') - \Phi(t') &= c(\text{ALG}_{\rho,j+1}^{(i+2)}) - c(\text{ALG}_{\rho,j}^{(i+2)}) \\ &\le \zeta' \frac{[c(\text{ALG}_{\rho,j}^{(i+2)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}_{\rho,j}^{(i+2)})]}{c'(\text{SOL})} [c'(\text{ALG}_{\rho,j}^{(i+2)}) - c'(\text{ALG}_{\rho,j+1}^{(i+2)})] \\ &= \zeta' \cdot [c(\text{ALG}_{\rho,j}^{(i+2)} \setminus \text{SOL}) - c(\text{SOL} \setminus \text{ALG}_{\rho,j}^{(i+2)})] \cdot \tau(j) = \zeta' \cdot \Phi(t') \cdot \tau(j).\end{aligned}$$

Rearranging terms we have:

$$\Phi(t'') \le (1 + \zeta' \cdot \tau(j))\, \Phi(t') \le e^{\zeta' \tau(j)} \Phi(t'),$$

where we use the inequality $1 + x \le e^x$. This completes the proof of the claim. □

6 HARDNESS RESULTS FOR ROBUST PROBLEMS

We give the following general hardness result for a family of problems that includes many graph optimization problems:

THEOREM 6.1. *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 6.1, we note that robust traveling salesman and robust Steiner tree are examples of problems that Theorem 6.1 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 6.1 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 6.1.

Proof of Theorem 6.1. 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 $[0, \infty]$.[6]
(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, $\text{mr} \leq O$. 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 $\mathbf{d}$, $\text{opt}(\mathbf{d})$ is finite. Then for all $\mathbf{d}$, $\text{sol}(\mathbf{d}) \leq \alpha \cdot \text{opt}(\mathbf{d}) + \beta \cdot \text{mr}$, i.e., $\text{sol}(\mathbf{d})$ 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., $\text{opt}(\mathbf{d}) = 0$. Then in this realization, $\text{sol}(\mathbf{d}) \leq \alpha \cdot \text{opt}(\mathbf{d}) + \beta \cdot \text{mr} \leq \beta \cdot O$. 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 $\mathcal{I}'$, $\text{sol}(\mathbf{d})$ also is the cost of sol in instance $\mathcal{I}'$, i.e., $\text{sol}(\mathbf{d})$ 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 [10, 15] we then get the following hardness results:

Corollary 6.2. *Finding an (α, β)-robust solution for Steiner tree where $\beta < 96/95$ is NP-hard.*

Corollary 6.3. *Finding an (α, β)-robust solution for TSP where $\beta < 121/120$ is NP-hard.*

7 CONCLUSION

In this paper, we designed constant approximation algorithms for the robust Steiner tree (stt) and traveling salesman problems (tsp). More precisely, our algorithms take as input a range of possible edge lengths in a graph and obtain a single solution for the problem at hand that can be compared to the optimal solution for any realization of edge lengths in the given ranges. While our approximation bounds for tsp are small constants, those 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. Indeed, as mentioned in our survey of related work, several different models of robust algorithms have been considered in the literature.

[6] ∞ is used to simplify the proof, but 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.

Optimizing over input ranges is one of the most natural models in robust optimization, but has been restricted in the past to polynomial-time solvable problems because of definitional limitations. We circumvent this by setting regret minimization as our goal, and creating the (α, β)-approximation framework, which then allows us to consider a large variety of interesting combinatorial optimization problems in this setting. We hope that our work will lead to more research in robust algorithms for other fundamental problems in combinatorial optimization, particularly in algorithmic graph theory.

REFERENCES

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

[2] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. 2009. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research* 197, 2 (2009), 427–438. https://doi.org/10.1016/j.ejor.2008.09.012

[3] Igor Averbakh. 2001. On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming* 90, 2 (01 Apr. 2001), 263–272.

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

[5] I. Averbakh and Oded Berman. 1997. Minimax regret p-center location on a network with demand uncertainty. *Location Science* 5, 4 (1997), 247–254. arXiv: https://doi.org/10.1016/S0966-8349(98)00033-3

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

[7] Dimitris Bertsimas and Melvyn Sim. 2003. Robust discrete optimization and network flows. *Mathematical Programming* 98, 1 (01 Sep. 2003), 49–71. https://doi.org/10.1007/s10107-003-0396-4

[8] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. 2010. An improved LP-based approximation for Steiner tree. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5–8 June 2010*. 583–592.

[9] André Chassein and Marc Goerigk. 2015. On the recoverable robust traveling salesman problem. *Optimization Letters* 10 (09 2015). https://doi.org/10.1007/s11590-015-0949-5

[10] Miroslav Chlebík and Janka Chlebíková. 2002. Approximation hardness of the Steiner tree problem on graphs. In *Algorithm Theory — SWAT 2002*, Martti Penttonen and Erik Meineche Schmidt (Eds.). Springer Berlin, Berlin, 170–179.

[11] Eduardo Conde. 2012. On a constant factor approximation for minmax regret problems using a symmetry point scenario. *European Journal of Operational Research* 219, 2 (2012), 452–457. https://doi.org/10.1016/j.ejor.2012.01.005

[12] Kedar Dhamdhere, Vineet Goyal, R. Ravi, and Mohit Singh. 2005. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05), 23–25 October 2005, Pittsburgh, PA, USA, Proceedings*. 367–378.

[13] Martin Groß, Anupam Gupta, Amit Kumar, Jannik Matuschke, Daniel R. Schmidt, Melanie Schmidt, and José Verschae. 2018. A local-search algorithm for Steiner forest. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11–14, 2018, Cambridge, MA, USA*. 31:1–31:17. https://doi.org/10.4230/LIPIcs.ITCS.2018.31

[14] Masahiro Inuiguchi and Masatoshi Sakawa. 1995. Minimax regret solution to linear programming problems with an interval objective function. *European Journal of Operational Research* 86, 3 (1995), 526–536. https://doi.org/10.1016/0377-2217(94)00092-Q

[15] Marek Karpinski, Michael Lampis, and Richard Schmied. 2013. New inapproximability bounds for TSP. In *Algorithms and Computation*, Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam (Eds.). Springer Berlin, Berlin, 568–578.

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

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

[18] P. Kouvelis and G. Yu. 1996. *Robust Discrete Optimization and Its Applications*. Springer US.

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

[20] Helmut E. Mausser and Manuel Laguna. 1998. A new mixed integer formulation for the maximum regret problem. *International Transactions in Operational Research* 5, 5 (1998), 389–403. https://doi.org/10.1016/S0969-6016(98)00023-9

[21] V. Vazirani. 2001. *Approximation Algorithms*. Springer-Verlag, Berlin.
[22] Jens Vygen. [n. d.]. New Approximation Algorithms for the TSP.
[23] Laurence A. Wolsey. 1980. Heuristic analysis, linear programming and branch and bound. In *Combinatorial optimization II*, V. J. Rayward-Smith (Ed.). Springer Berlin, Berlin, 121–134. https://doi.org/10.1007/BFb0120913
[24] H. Yaman, O. E. Karaşan, and M. Ç. Pinar. 2001. The robust spanning tree problem with interval data. *Operations Research Letters* 29, 1 (2001), 31–40.
[25] P. Zieliński. 2004. The computational complexity of the relative robust shortest path problem with interval data. *European Journal of Operational Research* 158, 3 (2004), 570–576.

Received 14 October 2021; revised 30 August 2022; accepted 23 October 2022