Reoptimization of Steiner Trees: Changing the Terminal Set ⋆

Hans-Joachim Böckenhauer [a], Juraj Hromkovič [a,*], Richard Královič [a], Tobias Mömke [a], Peter Rossmanith [b,1]

[a] *Department of Computer Science, ETH Zurich, Switzerland*

[b] *Department of Computer Science, RWTH Aachen, Germany*

Abstract

Given an instance of the Steiner tree problem together with an optimal solution, we consider the scenario where this instance is modified locally by adding one of the vertices to the terminal set or removing one vertex from it. In this paper, we investigate the problem whether the knowledge of an optimal solution to the unaltered instance can help in solving the locally modified instance. Our results are as follows: (i) We prove that these reoptimization variants of the Steiner tree problem are NP-hard, even if edge costs are restricted to values from $\{1, 2\}$. (ii) We design 1.5-approximation algorithms for both variants of local modifications. This is an improvement over the currently best known approximation algorithm for the classical Steiner tree problem which achieves an approximation ratio of $1+\ln(3)/2 \approx 1.55$. (iii) We present a PTAS for the subproblem in which the edge costs are natural numbers $\{1, \ldots, k\}$ for some constant k.

⋆ This work was partially supported by SBF grant C 06.0108 as part of the COST 293 (GRAAL) project funded by the European Union.

* Corresponding author. Address: Department of Computer Science, ETH Zurich, Universitätsstrasse 6, 8092 Zurich, Switzerland.

Email addresses: hjb@inf.ethz.ch (Hans-Joachim Böckenhauer), juraj.hromkovic@inf.ethz.ch (Juraj Hromkovič), richard.kralovic@inf.ethz.ch (Richard Královič), tobias.moemke@inf.ethz.ch (Tobias Mömke), rossmani@informatik.rwth-aachen.de (Peter Rossmanith).

[1] This author was visiting ETH Zurich while part of this work was done.

1 Introduction

In traditional optimization theory, one considers the task to find an optimal or near-optimal solution to an input instance of an optimization problem, where little or nothing is known in advance about this instance. But in many applications it might be necessary to recompute a solution for some locally modified input instance, where the local modification reflects some small change in the environment.

Technically, this leads to the theory of reoptimization problems. For an optimization problem U and a type of local modifications LM, like, e.g., adding or deleting a single vertex in a graph or changing the cost of an edge in a edge-weighted graph, the corresponding *reoptimization problem* LM-U is defined as follows: The input consists of an input instance I for U together with an optimal solution for I, and a second instance I' for U that is a local modification of I according to LM, and the objective is to find an optimal solution for I'. Reoptimization problems have been studied for example for the TSP for various types of local modifications in [1, 2, 4, 5].

In this paper, we will deal with the reoptimization of the Steiner tree problem. The input for the Steiner tree problem consists of a complete edge-weighted graph and a subset of the vertex set, the so-called *terminals*. The objective is now to compute a minimum-cost tree connecting all terminals, and possibly containing some of the other vertices of the graph, called *non-terminals*. The Steiner tree problem is a very prominent optimization problem with many practical applications, see for example [10, 12]. It is known to be APX-hard even if the edge costs are restricted to be 1 or 2 [3], and the best known approximation algorithm achieves an approximation ratio of $1+\ln(3)/2 \approx 1.55$ for general edge costs and 1.28 for edge costs from $\{1, 2\}$ [13].

Two reoptimization variants of the Steiner tree problem where the local modification consists of adding one vertex to the graph or deleting one vertex from the graph, respectively, were considered in [8]. In this paper, we will deal with another type of local modifications which does not change the underlying graph, namely adding one vertex to the terminal set or deleting one vertex from it. Our main results are as follows.

(i) We prove that these reoptimization variants of the Steiner tree problem are NP-hard, even if edge costs are restricted to values from $\{1, 2\}$.
(ii) We design 1.5-approximation algorithms for both adding and deleting a terminal. This shows that reoptimization really helps, since this is an improvement over the currently best known approximation algorithms for the classical Steiner tree problem which achieve an approximation ratio of about 1.55.

(iii) We show that reoptimization can help even more in the case of restricted edge costs: We present a PTAS for the subproblem where the edge costs are natural numbers from $\{1, \ldots, k\}$ for some constant k, whereas the classical Steiner tree problem is APX-hard even with edge costs restricted to values from $\{1, 2\}$.

The paper is organized as follows. In Section 2, we formally define the problem and present some observations on Steiner trees. Section 3 contains our hardness results. In Sections 4 and 5, we present approximation algorithms for decreasing and increasing the size of the terminal set, respectively. Section 6 presents the PTAS for the case of restricted edge costs and Section 7 concludes the paper.

2 Preliminaries

Given a simple graph G, we denote its set of vertices with $V(G)$ and its set of edges with $E(G)$. The degree of a vertex $v \in V(G)$ is $\deg_G(v)$. If G' is a subgraph of G, we write $G' \subseteq G$. We consider weighted graphs where a cost function c_G is defined on the edges. For simplicity, we omit the index G if it is clear from the context. The notation $c(G)$ denotes the sum of all edge costs in the graph G. For a graph G without an edge e or with an extra edge e, we simply write $G - e$ or $G + e$ respectively; analogously, for a graph G without a vertex v and without all edges incident to v, we write $G - v$. For two graphs G_1 and G_2, we denote by $G_1 + G_2$ the graph $(V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$.

We describe a path in a graph as a sequence of vertices. The length of a path is its number of edges. In a shortest path, the length of the path is minimized whereas in a cheapest path its cost is minimized. Thus the cheapest path connecting two vertices is at least as long as the shortest path.

We call a complete graph $G = (V, E)$ with edge cost function $c : E \to \mathbb{Q}^+$ *metric*, if the edge costs satisfy the *triangle inequality*

$$c(\{u, v\}) \leq c(\{u, w\}) + c(\{w, v\})$$

for all $u, v, w \in V$. For a complete graph $G = (V, E)$ with an arbitrary edge cost function $c : E \to \mathbb{Q}^+$, we define the *metric closure* of c as the function $\tilde{c} : E \to \mathbb{Q}^+$ where $\tilde{c}(\{u, v\})$ is defined as the length of the shortest path in (G, c) from u to v. Observe that $(G, \tilde{c})$ is metric.

Definition 1 *Given a graph $G = (V, E)$, a cost function $c : E \to \mathbb{Q}^+$, and a set of* terminals *$S \subseteq V$, a* Steiner tree *for (G, S, c) is any subgraph T of G such that T is a tree and $S \subseteq V(T)$.*

A tree T is a minimum Steiner tree *for (G, S, c), if T is a Steiner tree and $c(T) \leq c(T')$, for all Steiner trees T' for (G, S, c).*

The *minimum Steiner tree problem* on complete edge-weighted graphs (MinSTP for short) is the problem to find a minimum Steiner tree for an input instance (G, S, c) where G is a complete graph.

Lemma 1 *Let $G = (V, E)$ be a complete graph with edge cost function $c : E \to \mathbb{Q}^+$, let $S \subseteq V$ be a set of terminals. A minimum Steiner tree T for (G, S, c) is also a minimum Steiner tree for the metric closure $(G, S, \tilde{c})$. Moreover, any minimum Steiner tree T' for $(G, S, \tilde{c})$ can be transformed into a minimum Steiner tree for (G, S, c) by replacing any edge $\{u, v\}$ of T' by the shortest path from u to v in G according to c.*

Proof. To be found in [12]. □

We now formally define the reoptimization variants of MinSTP which we consider in this paper, namely changing the set of terminals.

Definition 2 *The* minimum Steiner tree reoptimization problem with reduced terminal set*, MinSTRP-RedTerm for short, is the following optimization problem: Given a complete graph $G = (V, E)$ with edge cost function $c : E \to \mathbb{Q}^+$, two terminal sets $S_N \subset S_O \subseteq V$ such that $|S_O - S_N| = 1$, and a minimum Steiner tree T_O for (G, S_O, c), find a minimum Steiner tree T_N for (G, S_N, c).*

The minimum Steiner tree reoptimization problem with augmented terminal set*, MinSTRP-AugTerm for short, is the following optimization problem: Given a complete graph $G = (V, E)$ with edge cost function $c : E \to \mathbb{Q}^+$, two terminal sets $S_O \subset S_N \subseteq V$ such that $|S_N - S_O| = 1$, and a minimum Steiner tree T_O for (G, S_O, c), find a minimum Steiner tree for (G, S_N, c).*

Our algorithms, as well as most known approximation algorithms for the classical MinSTP, only work for metric graphs G. This is no restriction for MinSTP since the Steiner tree for an arbitrarily edge-weighted graph (G, c) and for its metric closure $(G, \tilde{c})$ are related by Lemma 1. Therefore, also for the problems MinSTRP-AugTerm and MinSTRP-RedTerm we can assume in the following without loss of generality that the given edge cost function c is metric. Furthermore, we assume in the following without loss of generality that the given minimum Steiner tree T_O does not contain any non-terminal of degree 2. Due to the metricity of the considered input instances, such non-terminals can always be bypassed by a direct edge without increasing the cost.

For the Steiner tree problem, and also for the reoptimization variants as defined above, we can also define a subproblem where the cost of the edges is restricted by a constant-size set of integers. In the following we write $[i]$ for

the set of natural numbers $\{1, 2, \ldots, i\}$. We denote the respective variants restricted to edge costs from $[r]$ by r-MinSTP, r-MinSTRP-AugTerm, and r-MinSTRP-RedTerm.

Throughout the paper, we use the following notation: The input instances for our problems will consist of a complete graph $G = (V, E)$, two terminal sets S_O and S_N, and a metric edge cost function c. The given minimum Steiner tree will always be denoted by T_O and a minimum Steiner tree for the modified instances will be called T_N. By Opt_O and Opt_N we denote the costs of T_O and T_N, respectively. We refer to a Steiner tree computed by one of our algorithms as T_A and denote its costs with $c(T_A)$.

Before we start investigating the reoptimization variants, we observe some fundamental properties of minimal Steiner trees.

Lemma 2 *Let G be a complete graph with edge cost function c, let $S \subseteq V(G)$ be a terminal set, let T be a minimum Steiner tree for (G, S, c). Let $e = \{x, y\} \in E(T)$ such that $x \in S$. Let T_x be the connected component of $T - e$ containing x, i.e., an inclusion-maximal subtree of T rooted at x. Let G_x be the subgraph of G induced by $V(T_x)$. Then T_x is a minimum Steiner tree for $(G_x, S \cap V(T_x), c)$.*

Proof. To be found in [11]. □

The following lemma allows us to express the time complexity of some algorithms by using the number of terminals rather than by using the total number of vertices.

Lemma 3 *Let T be a minimal Steiner tree for the input instance (G, S, c). If $\deg_T(v) \neq 2$ for all non-terminals $v \in V(T)$, then $|V(T)| < 2 \cdot |S|$.*

Proof. Each leaf of T is a terminal since otherwise we would obtain a better solution by cutting that leaf from T. We want to prove that T has at least as many leaves as non-terminals. Let us first assume that T does not contain any terminal of degree 2. Then T does not have any vertices of degree 2 at all. We will show that in this case at least half of the vertices have to be leaves. Let V' denote the set of inner vertices of T. Then $\sum_{v \in V'} \deg_T(v) \geq 3 \cdot |V'|$. On the other hand, the restriction of T to the vertices from V' obviously is again a tree which we denote by T'. The tree T' has $|V'| - 1$ edges and thus $\sum_{v \in V'} \deg_{T'}(v) \leq 2 \cdot (|V'| - 1)$. This implies that T has at least $3 \cdot |V'| - 2 \cdot |V'| + 2 = |V'| + 2$ leaves, i.e., more than half of the vertices are leaves.

We now deal with the general case of trees including terminals of degree 2. We can obtain a new tree $\tilde{T}$ from T by sequentially replacing each terminal t of degree 2 together with its incident edges $\{t, x\}$ and $\{t, y\}$ by the direct edge $\{x, y\}$. Using the same argument as above, we can show that at least half of

the vertices of $\tilde{T}$ are leaves, and thus terminals. As T does not contain more non-terminals than $\tilde{T}$, the claim follows. □

3 Hardness results

We provide a framework which enables us to show the NP-hardness of various reoptimization variants of the Steiner tree problem.

Lemma 4 *Let LM be a type of local modifications such that a deterministic algorithm can transform some efficiently solvable input instance (G', S', c') for MinSTP into any input instance (G, S, c) for 2-MinSTP using a polynomial number of local modifications of type LM, then the problem MinSTP-LM is NP-hard.*

Proof. We reduce 2-MinSTP to MinSTP-LM by the means of a polynomial-time Turing reduction, i.e., we assume that we have a polynomial-time algorithm for MinSTP-LM and use it for constructing a polynomial-time algorithm for 2-MinSTP. Since 2-MinSTP is known to be NP-hard in the strong sense [9], such a reduction implies the NP-hardness of MinSTP-LM.

Let (G, S, c) be an arbitrary instance for 2-MinSTP. Let l be the number of local modifications of type LM needed for transforming the efficiently solvable MinSTP instance (G', S', c') into (G, S, c). We assume that we have a polynomial-time algorithm A solving MinSTP-LM. Starting with (G', S', c') and applying A exactly $l-1$ times, we can find an optimal solution for (G, S, c). This is obviously possible in polynomial time. □

Note that we use a polynomial-time Turing reduction (also called Cook reduction) for showing the NP-hardness. Therefore, strictly speaking, from our proof we cannot conclude NP-completeness, see [9].

Theorem 1 *The problems MinSTRP-AugTerm and MinSTRP-RedTerm are strongly NP-hard.*

Proof. Let (G, S, c) be an arbitrary input instance for 2-MinSTP. For both problems MinSTRP-AugTerm and MinSTRP-RedTerm, we will give an efficiently solvable input instance and a simple algorithm to transform the instance into (G, S, c). Applying Lemma 4 then directly implies the NP-hardness of these problems. Let $S = \{s_1, s_2, \ldots, s_l\}$.

To show the hardness of MinSTRP-AugTerm, we consider the graph $(\{s_1\}, \emptyset)$, which is the unique minimum Steiner tree for $(G, \{s_1\}, c)$. Adding all vertices s_2 to s_l successively to the terminal set, i.e., solving $l - 1$ instances of MinSTRP-AugTerm, is sufficient to transform $(G, \{s_1\}, c)$ into (G, S, c).

Analogously, the problem MinSTRP-RedTerm is NP-hard since we can solve the instance $(G, V(G), c)$ of MinSTP optimally by computing a minimum spanning tree. Obviously, successively removing the $|V(G)\backslash S|$ vertices not belonging to S from the terminal set is sufficient and we can again apply Lemma 4.

We only considered cost functions that assign costs 1 or 2 to the edges. This shows, that even the variants of the problems with restricted edge costs are NP-hard. Following [9], this implies the strong NP-hardness of the problems. □

4 Decreasing the Number of Terminals

Our approximation algorithm for MinSTRP-RedTerm is based on the following idea: It removes the most expensive edges from the given minimum Steiner tree, contracts those components of the resulting forest that contain terminals into vertices and calculates a minimum Steiner tree to cover those vertices. Finally, it uses the calculated Steiner tree to re-connect the forest which yields a feasible solution for the given input.

Formally, we define the contraction of subgraphs similarly to the standard contraction of edges (see e.g. [14]).

Definition 3 *Let G be a complete graph with edge cost function c. The* contraction *of a subgraph C of G into a vertex v yields the multigraph G' with $V(G') = (V(G) - V(C)) \cup \{v\}$. Each edge between two vertices of C is transformed into a loop at v, and the edges from vertices in C to a vertex y outside C are transformed into multiedges from v to y.*

Obviously, when contracting a subgraph C of G each edge from $E(G)$ can be bijectively mapped to an edge of the resulting multigraph G'. Therefore, we do not distinguish between the two edge sets and the cost functions of the two graphs.

The Steiner tree problem for multigraphs can be treated analogously to the Steiner tree problem for simple graphs since we can ignore the loops and consider only the cheapest simple edge from each multiedge.

This contraction technique will be used in Algorithm 1 for approximating MinSTRP-RedTerm.

Theorem 2 *Algorithm 1 runs in $O(|V(G)|^{4.17})$ time and achieves an approximation ratio of* 1.5 *for MinSTRP-RedTerm.*

Algorithm 1 A terminal becomes a non-terminal

Input: A metric graph G, a cost function c, a terminal set $S_O \subseteq V(G)$, a minimum Steiner tree $T_O \subseteq G$ for (G, S_O, c) and a new terminal set $S_N := S_O - \{t\}$ for some terminal $t \in S$.

1: Obtain a forest F by removing one edge incident to t and the $\min\{2 \cdot \lceil \log_2 |V(G)| \rceil, |E(T_O)| - 1\}$ most expensive of the remaining edges from T_O.

2: Remove all components without vertices from S_N from F. Let l be the number of remaining components.

3: Obtain the multigraph G' by contracting the components $C_1, C_2, \ldots, C_l$ of F to $v_1, v_2, \ldots, v_l$ in G.

4: Calculate a minimum Steiner tree T' for $(G', \{v_1, v_2, \ldots, v_l\}, c')$.

Output: $T_A = T' \cup F$

Proof. In the following, we assume that $|E(T_O)| > 2 \cdot \lceil \log_2 n \rceil + 1$ and thus $2 \cdot \lceil \log_2 n \rceil + 1$ edges are removed, where $n = |V(G)|$ is the number of vertices of the input graph. Otherwise, Algorithm 1 yields an optimal solution. We distinguish three cases depending on the degree of the vertex $t \in S_O - S_N$ in T_O.

Case 1: Assume $\deg_{T_O}(t) \geq 3$. In this case, we show that T_O — and therefore T_A — is a 1.5-approximate solution: note that the deleted edges in Algorithm 1 form a feasible Steiner tree for $(G', \{v_1, v_2, \ldots, v_l\}, c')$ and therefore $c(T_A) \leq \mathrm{Opt}_O$. Let p be the cheapest path from t to another terminal. Then, since $\deg_{T_O}(t) \geq 3$, there exist three edge-disjoint paths from t to terminals in T_O and thus $c(T_O) \geq 3 \cdot c(p)$. On the other hand, since $T_N + p$ is a solution for (G, S, c), we know $\mathrm{Opt}_N + c(p) \geq \mathrm{Opt}_O$. Therefore, $\mathrm{Opt}_N \geq 2 \cdot c(p)$ and thus

$$\frac{\mathrm{Opt}_O}{\mathrm{Opt}_N} \leq \frac{\mathrm{Opt}_N + c(p)}{\mathrm{Opt}_N} = 1 + \frac{c(p)}{\mathrm{Opt}_N} \leq 1 + \frac{c(p)}{2c(p)} = 1.5.$$

Case 2: Assume $\deg_{T_O}(t) = 2$. The Steiner tree T_O where $d(t) = 2$ has the form as depicted in Figure 1.

The trees T_a and T_b are the left and the right subtrees of T_O rooted at t. Let p be the cheapest path from t to some terminal in T_O. Without loss of generality, let p contain the edge a. We define analogously p' as the cheapest path to a terminal in T_O starting at t and containing the edge b. We distinguish two sub-cases concerning the cost of p.

Case 2.1: Assume $c(p) \leq \mathrm{Opt}_N/2$. Since $T_N + p$ is a solution for (G, S_O, c), $\mathrm{Opt}_O \leq \mathrm{Opt}_N + c(p) \leq 3\mathrm{Opt}_N/2$ and thus Opt_O is at least a 1.5-approximation for (G, S_N, c).

Case 2.2: Assume $\mathrm{Opt}_N/2 < c(p)$. In this case, $c(p) = \mathrm{Opt}_N/2 + \alpha \cdot \mathrm{Opt}_N$ for some $0 < \alpha$. Let $\tilde{p}$ and $\tilde{p}'$ be the shortest (with respect to the number of edges) paths from t to a terminal, containing the edges a and b respectively. Let e be a function determining the length of a path, i.e., its

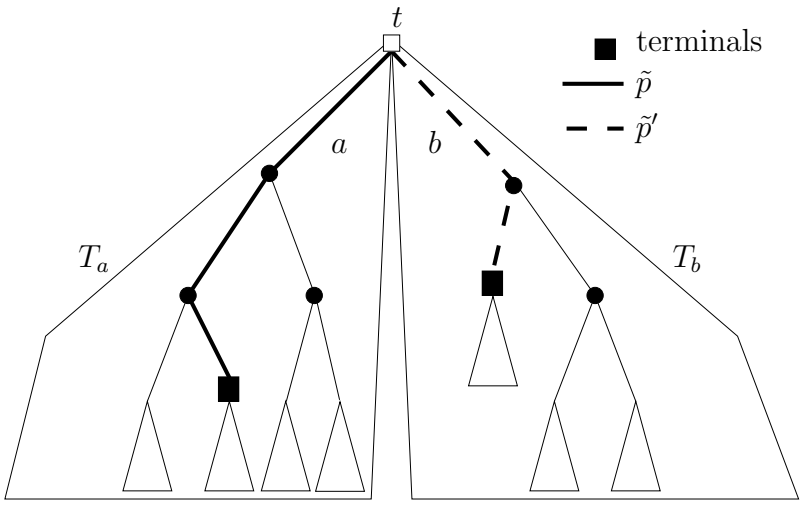

Fig. 1. A Steiner tree with $d(t) = 2$.

number of edges. Obviously,

$$c(\tilde{p}) + c(\tilde{p}') \geq 2 \cdot (1/2 + \alpha) \cdot \mathrm{Opt}_N.$$

The trees T_a and T_b have at least $2^{e(\tilde{p})}$ and $2^{e(\tilde{p}')}$ vertices, respectively, since all non-terminals have degree at least 3. Therefore, the length of $\tilde{p}$, as well as the length of $\tilde{p}'$, is at most $\lceil \log_2 n \rceil$. Let E_r denote the set of edges removed from T_O by the algorithm. Then $c(E_r) \geq c(\tilde{p}) + c(\tilde{p}')$ holds since E_r contains the $2 \cdot \lceil \log_2 n \rceil$ most expensive edges from T_O.

Removing one edge from a forest increases the number of components exactly by one. Therefore, $T_O - E_r$ has exactly $|E_r| + 1$ components. Let $\hat{T}$ be the Steiner tree re-connecting these components (Line 4 in Algorithm 1). Certainly, the cost $c(\hat{T})$ cannot be larger than Opt_N. Therefore, the costs of the new tree composed from all components and $\hat{T}$ are at most

$$\mathrm{Opt}_O - 2 \cdot \frac{1}{2} \cdot \mathrm{Opt}_N - 2\alpha \cdot \mathrm{Opt}_N + \mathrm{Opt}_N = \mathrm{Opt}_O - 2\alpha \cdot \mathrm{Opt}_N.$$

Since

$$\begin{aligned}
&\frac{\mathrm{Opt}_O - 2\alpha \cdot \mathrm{Opt}_N}{\mathrm{Opt}_N} \\
\leq &\frac{\mathrm{Opt}_N + c(p) - 2\alpha \cdot \mathrm{Opt}_N}{\mathrm{Opt}_N} \\
= &\frac{\mathrm{Opt}_N + \mathrm{Opt}_N/2 + (\alpha - 2\alpha) \cdot \mathrm{Opt}_N}{\mathrm{Opt}_N} \\
= &\frac{3\mathrm{Opt}_N/2 - \alpha \cdot \mathrm{Opt}_N}{\mathrm{Opt}_N} \\
< &\, 3/2,
\end{aligned}$$

we can be sure that the new solution is at least a 1.5-approximation.

Thus, by removing edges E_r from T_O and re-connecting in the described manner, Algorithm 1 yields a 1.5-approximation.

Case 3: Assume $\deg_{T_O}(t) = 1$. In this case, there is exactly one $v \in V(G)$ such that $e = \{t, v\} \in E(G)$ is incident to t. Moreover, $e \in E_r$ since at least one edge incident to t is removed by the algorithm. We distinguish two cases depending on whether v is a terminal or not.

Case 3.1: Assume that v is a terminal. Then $T_O - e$ is a minimum Steiner tree for $(G, (S \cup \{v\}) - \{t\}, c)$ according to Lemma 2. Moreover, the Steiner tree computed by the algorithm cannot be more expensive than $T_O - e$.

Case 3.2: Assume that v is a non-terminal. Since we excluded non-terminals of degree 2, we know that $\deg_{T_O}(v) \geq 3$ and thus $\deg_{T_O - \{t\}}(v) \geq 2$. Since T_O is a minimum Steiner tree for (G, S_O, c) containing v, it is also optimal for $(G, S_O \cup \{v\}, c)$. Lemma 2 shows that $T_O - e$ is a minimum Steiner tree for $(G, (S_O \cup \{v\}) - \{t\}, c)$. Noting that the proof of case 2.2 only considered $2 \cdot \lceil \log_2 n \rceil$ of the edges from E_r, i.e., all edges except e, we can conclude, analogously to the cases 1 and 2, that the algorithm computes a 1.5-approximate Steiner tree for $(G, ((S_O \cup \{v\}) - \{t\}) - \{v\}, c)$. Obviously, $S_N = S_O - \{t\} = ((S_O \cup \{v\}) - \{t\}) - \{v\}$ which completes the proof.

The time complexity of Algorithm 1 is dominated by the exact calculation of a minimum Steiner tree connecting the components.

In order to keep the time complexity polynomial, we exploit the fact that the number of components is small and that each of them corresponds to exactly one terminal in the contracted graph. Let n be the number of vertices and k be the number of terminals. Then the Dreyfus-Wagner algorithm for computing the minimum Steiner tree [7] runs in time

$$O\left(n^3 + n^2 \cdot 2^k + n \cdot 3^k\right).$$

We have to compute a Steiner tree with $2 \cdot \lceil \log_2 n \rceil + 1$ terminals and at most n vertices. Thus, the asymptotic time complexity of the computation is

$$O\left(n^3 + n^2 \cdot 2^{2 \cdot \log_2 n} + n \cdot 3^{2 \cdot \log_2 n}\right).$$

The dominant part is

$$n \cdot 3^{2 \cdot \log_2 n} = n \cdot (2^{\log_2 3})^{2 \cdot \log_2 n} = n^{2 \cdot \log_2 3 + 1} < n^{4.17}$$

and thus the time complexity is at most $O(n^{4.17})$. It is easy to see that no other step of Algorithm 1 increases the asymptotic time complexity. □

5 Increasing the Number of Terminals

Instead of removing a terminal from the set S, we can also add a non-terminal to S. For this local modification we present a very simple linear-time 1.5-approximation algorithm. Our focus is on the time complexity since a 1.5-approximation follows easily from the results in [8] where an approximation algorithm for adding vertices to the graph is presented. We can assume that the new terminal is not in the given optimal solution, because otherwise that solution is already optimal for the modified instance. On the other hand, removing the new terminal from the graph and adding it again using the approximation algorithm from [8] yields the desired result. However, this algorithm has superquadratic time complexity.

We again assume without loss of generality that the given minimum Steiner tree T_O does not contain non-terminals of degree 2.

The idea of our algorithm is simply to take the old solution and to add the cheapest edge that connects the new terminal with the given tree.

Algorithm 2 Declaring a non-terminal to be a terminal

Input: A metric graph G, a cost function c, a terminal set $S_O \subseteq V(G)$, a minimum Steiner tree $T_O \subseteq G$ for (G, S_O, c) and a new terminal set $S_N := S_O \cup \{t\}$ for some non-terminal $t \in V(G) \backslash S_O$.

1: **if** $t \in V(T_O)$ **then**
2: $\quad T_A := T_O$
3: **else**
4: $\quad$ Let $e = \{u, t\}$ be an edge of minimum cost such that $u \in V(T_O)$
5: $\quad T_A := T_O + e$
6: **end if**

Output: T_A

Theorem 3 *Algorithm 2 runs in $O(|S_N|)$ time and achieves an approximation ratio of 1.5 for MinSTRP-AugTerm.*

Proof. Let $f = (s, t)$ be the cheapest edge connecting a terminal $s \in S_O$ with t and let $c(T_A)$ be the cost of the solution calculated by Algorithm 2.

If $c(f) \leq \mathrm{Opt}_O/2$ holds, then $c(T_A) \leq 3\mathrm{Opt}_O/2 \leq 3\mathrm{Opt}_N/2$ since $c(e) \leq c(f)$ and $\mathrm{Opt}_O \leq \mathrm{Opt}_N$. Thus, in the remainder of the proof we assume that $c(f) > \mathrm{Opt}_O/2$.

We first consider the case where t has exactly one neighbor in T_N. If that neighbor is a terminal, then Lemma 2 shows that Algorithm 2 calculates an optimal solution. If that neighbor is a non-terminal, say v, its degree is, as already mentioned above, at least three. Since $d(v) \geq 3$, there are two edge-

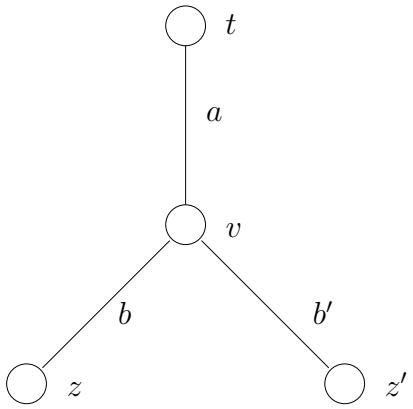

Fig. 2. The situation in the proof of Theorem 3 where $\deg_{T_N}(t) = 1$ holds and the neighbor of t is a non-terminal.

disjoint paths from v to terminals z and z' from S_O. Let $b = (v, z)$ and $b' = (v, z')$ be the edges connecting v with these terminals, see Figure 2. Due to the metricity, these edges are at most as expensive as the corresponding paths. We denote the edge $\{v, t\}$ by a.

Since $T_N - a$ is a solution for (G, S_O, c),

$$\mathrm{Opt}_N \geq \mathrm{Opt}_O + c(a). \tag{1}$$

Since as well $a + b$ as $a + b'$ connects t with a terminal,

$$c(T_A) \leq \mathrm{Opt}_O + c(a) + c(b). \tag{2}$$

We assume without loss of generality that $c(b') \geq c(b)$ holds. Since b and b' are shortcuts of two edge-disjoint paths in T_N, we know that $\mathrm{Opt}_N \geq c(b) + c(b')$ and thus

$$c(b) \leq \mathrm{Opt}_N/2. \tag{3}$$

We get

$$c(T_A) \underset{(2)}{\leq} \mathrm{Opt}_O + c(a) + c(b) \underset{(1)}{\leq} \mathrm{Opt}_N + c(b) \underset{(3)}{\leq} 3\mathrm{Opt}_N/2.$$

The resulting approximation ratio is obviously $3/2$.

The remaining case where t has more than one neeighbor in T_N can easily be reduced to the previous case by introducing a new non-terminal v such that

$c(\{v,w\}) = c(\{t,w\})$ for all $w \in V(G)$ and $c(\{v,t\}) := 0$.[2] Now t has only one neighbor, namely v, and all edges incident to t in T_N now are replaced by edges incident to v. □

6 A PTAS for Restricted Edge Costs

In this section, we consider the restricted subproblems r-MinSTRP-AugTerm and r-MinSTRP-RedTerm, i.e., the variants restricted to input instances with edge costs from $[r]$. For any $\varepsilon > 0$, we construct a $(1+\varepsilon)$-approximative algorithm for r-MinSTRP-AugTerm and r-MinSTRP-RedTerm, respectively. The algorithm either calculates an optimal solution without using the old optimum if the number of terminals is small, or the old optimum (augmented by an extra edge for r-MinSTRP-AugTerm) is a good approximation.

Algorithm 3 PTAS for r-MinSTRP-AugTerm and r-MinSTRP-RedTerm

Input: A metric graph G, a constant $r \in \mathbb{N}$, a cost function $c : E(G) \to [r]$, a terminal set $S_O \subseteq V(G)$, a minimum Steiner tree $T_O \subseteq G$ for (G, S_O, c), and either, for r-MinSTRP-AugTerm, a vertex $t \in V(G) - S_O$ and a new terminal set $S_N = S_O \cup \{t\}$ or, for r-MinSTRP-RedTerm, a vertex $t \in S_O$, and a new terminal set $S_N = S_O - \{t\}$.

1: Let $k := \lceil 1/\varepsilon \rceil$.
2: **if** $|S_N| \leq r \cdot k$ **then**
3: Use the Dreyfus-Wagner algorithm to compute an optimal solution T_A for (G, S_N, c).
4: **else**
5: Let $T_A := T_O$.
6: In the case of r-MinSTRP-AugTerm, add t and an (arbitrary) edge from $\{\{t,v\} \mid v \in V(T_O)\}$ to T_A.
7: **end if**

Output: T_A

We show that Algorithm 3 is even an efficient PTAS in the sense of the following definition from [6].

Definition 4 *An approximation algorithm for an optimization problem is an* efficient PTAS *if it computes a $(1+\varepsilon)$-approximative solution in time $O\left(f(\varepsilon) \cdot n^c\right)$ for some computable function f and some constant c.*

Theorem 4 *Algorithm 3 is an efficient PTAS for r-MinSTRP-AugTerm and for r-MinSTRP-RedTerm.*

[2] Note that we use a slightly expanded definition of the MinSTP here since edge cost of 0 are usually not allowed.

Proof. For showing that the approximation ratio of Algorithm 3 is $1+\varepsilon$, we can assume without loss of generality that $|S_N| > r \cdot k$ holds, because otherwise the algorithm computes an optimal solution. In this case, the cost of an optimal solution is at least $r \cdot k$, because the cost of each edge is at least one.

For r-MinSTRP-AugTerm, adding one edge costs at most r, i.e., $c(T_A) \leq \mathrm{Opt}_O + r$. Thus, the approximation ratio is

$$\frac{c(T_A)}{\mathrm{Opt}_N} \leq \frac{\mathrm{Opt}_O + r}{\mathrm{Opt}_N} \leq \frac{\mathrm{Opt}_N + r}{\mathrm{Opt}_N} = 1 + \frac{r}{\mathrm{Opt}_N} \leq 1 + \frac{r}{r \cdot k} \leq 1 + \varepsilon.$$

For r-MinSTRP-RedTerm, we know that $\mathrm{Opt}_O \leq \mathrm{Opt}_N + r$ since adding one edge to T_N yields a valid solution for (G, S_O, c). The approximation ratio of Algorithm 3 for r-MinSTRP-RedTerm is thus

$$\frac{c(T_A)}{\mathrm{Opt}_N} \leq \frac{\mathrm{Opt}_N + r}{\mathrm{Opt}_N} = 1 + \frac{r}{\mathrm{Opt}_N} \leq 1 + \frac{r}{r \cdot k} = 1 + \varepsilon.$$

According to [7], the time complexity of calculating an optimal solution in line 3 of Algorithm 3 is in $O(n^2 \cdot 3^{r \cdot k})$. The time complexity of the remaining parts is negligible. Since r is a constant and we fix k as soon as we choose ε, all requirements for an efficient PTAS are fulfilled. □

7 Conclusion

In real applications, one usually has experience with solving situations that are similar to the current one. The question of principal interest is how much such experience can help in searching for a good solution to an actual problem instance. As we have shown in our simplified scenario here, the answer may differ from problem to problem, and it may also depend on the way how the hardness of a problem is measured.

For the Steiner tree problem with bounded edge costs, one can move from the APX-hardness of the original problem to a PTAS for the reoptimization variant. On the other hand, the reoptimization problem with its additional knowledge can be as hard as the original problem from some point of view: Reoptimizing the Steiner tree problem remains NP-hard even for edge costs restricted to the values 1 and 2.

Hence, the main research focus is on investigating (i) what kind of additional knowledge (in the form of experience in solving similar problem instances) can be helpful for efficiently getting high-quality solutions for discrete optimization

problems, and (ii) for which optimization problems such additional knowledge helps at all.

Acknowledgments

The authors would like to thank Martin Dietzfelbinger and Sebastian Seibert for helpful discussions.

References

[1] C. Archetti, L. Bertazzi, M. G. Speranza, Reoptimizing the traveling salesman problem, Networks 42 (3) (2003) 154–159.

[2] G. Ausiello, B. Escoffier, J. Monnot, V. T. Paschos, Reoptimization of minimum and maximum traveling salesman's tours, in: Algorithm theory—SWAT 2006, vol. 4059 of Lecture Notes in Comput. Sci., Springer, Berlin, 2006, pp. 196–207.

[3] M. W. Bern, P. E. Plassmann, The Steiner problem with edge lengths 1 and 2., Inf. Process. Lett. 32 (4) (1989) 171–176.

[4] H.-J. Böckenhauer, L. Forlizzi, J. Hromkovič, J. Kneis, J. Kupke, G. Proietti, P. Widmayer, On the approximability of TSP on local modifications of optimally solved instances, Algorithmic Operations Research (to appear).

[5] H.-J. Böckenhauer, L. Forlizzi, J. Hromkovič, J. Kneis, J. Kupke, G. Proietti, P. Widmayer, Reusing optimal TSP solutions for locally modified input instances (extended abstract), in: Fourth IFIP International Conference on Theoretical Computer Science—TCS 2006, vol. 209 of IFIP Int. Fed. Inf. Process., Springer, New York, 2006, pp. 251–270.

[6] R. G. Downey, M. R. Fellows, Parameterized Complexity, Monographs in Computer Science, Springer-Verlag, New York, 1999.

[7] S. E. Dreyfus, R. A. Wagner, The Steiner problem in graphs, Networks 1 (1971/72) 195–207.

[8] B. Escoffier, M. Milanic, V. T. Paschos, Simple and fast reoptimizations for the Steiner tree problem, Tech. Rep. 2007-01, DIMACS (2007).

[9] M. R. Garey, D. S. Johnson, Computers and Intractability, W. H. Freeman and Co., San Francisco, Calif., 1979.

[10] F. Hwang, D. Richards, P. Winter, The Steiner Tree Problems, vol. 53 of Annals of Discrete Mathematics, North-Holland, 1992.

[11] D. Mölle, S. Richter, P. Rossmanith, A faster algorithm for the Steiner tree problem, in: B. Durand, W. Thomas (eds.), STACS, vol. 3884 of Lecture Notes in Computer Science, Springer, 2006.

[12] H. J. Prömel, A. Steger, The Steiner Tree Problem, Advanced Lectures in Mathematics, Friedr. Vieweg & Sohn, Braunschweig, 2002.

[13] G. Robins, A. Zelikovsky, Improved Steiner tree approximation in graphs, in: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 2000), ACM, New York, 2000.

[14] D. B. West, Introduction to Graph Theory, 2nd ed., Prentice Hall Inc., Upper Saddle River, NJ, 2000.