Algorithmica (2020) 82:1966–1988
https://doi.org/10.1007/s00453-020-00682-x

Robust Reoptimization of Steiner Trees

Keshav Goyal[1] · **Tobias Mömke**[2]

Received: 6 December 2017 / Accepted: 14 January 2020 / Published online: 27 January 2020

Abstract
In reoptimization, one is given an optimal solution to a problem instance and a (locally) modified instance. The goal is to obtain a solution for the modified instance. We aim to use information obtained from the given solution in order to obtain a better solution for the new instance than we are able to compute from scratch. In this paper, we consider Steiner tree reoptimization and address the optimality requirement of the provided solution. Instead of assuming that we are provided an optimal solution, we relax the assumption to the more realistic scenario where we are given an approximate solution with an upper bound on its performance guarantee. We show that for Steiner tree reoptimization there is a clear separation between local modifications where optimality is crucial for obtaining improved approximations and those instances where approximate solutions are acceptable starting points. For some of the local modifications that have been considered in previous research, we show that for every fixed $\varepsilon > 0$, approximating the reoptimization problem with respect to a given $(1+\varepsilon)$-approximation is as hard as approximating the Steiner tree problem itself. In contrast, with a given optimal solution to the original problem it is known that one can obtain considerably improved results. Furthermore, we provide a new algorithmic technique that, with some further insights, allows us to obtain improved performance guarantees for Steiner tree reoptimization with respect to all remaining local modifications that have been considered in the literature: a required node of degree more than one becomes a Steiner node; a Steiner node becomes a required node; the cost of one edge is increased.

Keywords Reoptimization · Approximation algorithms · Steiner tree problem

A preliminary version of this paper has appeared in Proc. 35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015.

✉ Tobias Mömke
moemke@cs.uni-saarland.de

Keshav Goyal
keshav.goyal@gravitontrading.com

[1] Graviton Research Capital LLP, Gurugram, Haryana, India

[2] Saarland University, Saarbrücken, Germany

1 Introduction

The Steiner tree problem (STP) is one of the most studied problems in the area of network design. We are given a graph G with nodes $V(G)$, edges $E(G)$, and a cost function $c : E(G) \to \mathbb{R}_{\geq 0}$, as well as a set $R \subseteq V(G)$ of required nodes (also called regular nodes or terminals). The objective is to find a minimum-cost tree T within G such that $R \subseteq V(T)$. The Steiner tree problem is known to be APX-hard [8], and the currently best approximation algorithm has a performance guarantee of $\ln 4 + \varepsilon \approx 1.387$ [25].

We consider the Steiner tree problem with respect to reoptimization, a framework for dynamic algorithms in the context of NP-hard problems. We are given two related instances I and I' of an algorithmic problem together with a solution Sol to the instance I, and our goal is to compute a solution to I'. The relation between I and I' is determined by an operation that we call *local modification*.

The concept of reoptimization is motivated by the observation that instead of computing new solutions from scratch, oftentimes we can reuse the effort spent to solve problems similar to the one at hand. For instance, let us consider a large circuit where certain components have to be connected. The components are the required nodes and there are points that may be used by several connections, the Steiner nodes. Now suppose that a long and costly computation has led to an almost optimal solution. Afterwards the requirements change: either an additional component has to be placed to a point that previously was a Steiner node or a component is removed, which turns a required node into a Steiner node. In such a situation it would seem wasteful to discard the entire previous effort.

Classically, when considering reoptimization problems one assumes that Sol is an optimal solution. The reason for this assumption is that assuming optimality considerably reduces the formal overhead and therefore facilitates to concentrate on the main underlying properties of the reoptimization problem. We show, however, that assuming optimality is not without loss of generality. Let us assume that $c(\text{Sol})$ is a $(1 + \varepsilon)$ factor larger than the cost of an optimal solution. Then we say that a Steiner tree reoptimization algorithm is *robust*, if it is an approximation algorithm and its performance guarantee is $\alpha \cdot (1 + O(\varepsilon))$, where α is its performance guarantee when $\varepsilon = 0$. Intuitively, this definition ensures that for $\varepsilon \to 0$, the performance guarantee converges smoothly towards α, independent of the given instance. We consider robustness of reoptimization algorithms to be a crucial feature, since in real-world applications close-to-optimal solutions are much more frequent than optimal solutions.

We address all local modifications that have previously been considered for Steiner tree reoptimization. We classify these modifications into two groups, according to their robustness. The first group contains those problems where obtaining a robust reoptimization algorithm implies to provide an approximation algorithm for the (non-reoptimization) Steiner tree problem with matching performance guarantee. The second group of problems allows for improved robust reoptimization algorithms compared to STP approximation algorithms.

Table 1 Comparison of approximation ratios of the Steiner Tree Reoptimization problem for the different types of local modifications

Local modification	Our results solution sol: $(1+\varepsilon)$-approx		Previous results solution sol: optimal solution		Subsequent results [9]
	Expression	Value	Expression	Value	
STP^{R-} (internal node)	$\frac{10\beta-7}{7\beta-4}$	1.204	$\frac{3\beta-2}{2\beta-1}$ [14]	1.219	$1+\varepsilon$
STP^{R-} (leaf node)	Not robust	1.204 if $\varepsilon=0$	$\frac{3\beta-2}{2\beta-1}$ [14]	1.219	$1+\varepsilon$
STP^{R+}	$\frac{10\beta-7}{7\beta-4}$	1.204	$\frac{3\beta-2}{2\beta-1}$ [13]	1.219	$1+\varepsilon$
STP^{E+}	$\frac{7\beta-4}{4\beta-1}$	1.256	$\frac{2\beta-1}{\beta}$ [14]	1.29	$1+\varepsilon$
STP^{E-}	Not robust	1.387	$\frac{5\beta-3}{3\beta-1}$ [11] Assuming metricity	1.246	$1+\varepsilon$
Add node	Not robust	1.387	Without [25]: 1.5 [28]	1.387	
Remove node	Not robust	1.387	As hard as STP approx [17]	1.387	

To increase the readability, all values ε, δ in the approximation ratios are omitted. The numerical values are rounded up at the third digit and we assume $\beta = 1.387$, the approximation ratio $\ln(4)+\varepsilon$ of the Steiner tree approximation algorithm of Byrka et al. [25] with small enough ε. The subsequent results of Bilò [9] assume Sol to be an optimal solution but translate to robust reoptimization results for all local modifications except those that we show to be not permit robust reoptimization

For all reoptimization problems of the second group that have previously been considered (and that are known to be NP-hard [17]), we provide robust reoptimization algorithms that, for $\varepsilon \to 0$, obtain better performance guarantees than the previous results with optimality assumption [13, 14].

After the journal submission of our manuscript, Bilò [9] published polynomial time approximation schemes for Steiner tree reoptimization with respect to four of the most important local modifications (see Table 1). The algorithms use some of our techniques as a building block.

1.1 Local Modifications and Our Contribution

There are ten local modifications that previously have been considered for the Steiner tree problem. The two most studied modifications address the set of required nodes: we either declare a required node to be a Steiner node, or a Steiner node to be a required node. Here, STP^{R-} resp. STP^{R+} denote the corresponding reoptimization problems. We show, in Sect. 4, that finding a robust reoptimization algorithm for STP^{R-} is as hard as finding a Steiner tree approximation algorithm with matching approximation ratio. If one, however, excludes that the node t declared to be a Steiner node is a leaf in the given instance, we provide a robust reoptimization algorithm with improved performance ratio (see Table 1 for an overview of the achieved improvements). We show that in contrast to STP^{R-}, STP^{R+} always allows for improved robust reoptimization algorithms. The next interesting type of local modification is to modify the cost of a single edge. We do not require the cost function to be metric. In particular, in the shortest path metric induced by the modified edge cost, the cost of several edges may be changed. We call the modification where

the cost of one edge is increased STP^{E+}, and the converse local modification where the cost of one edge is decreased is STP^{E-}. We provide an improved robust reoptimization algorithm for STP^{E+} and show that robust reoptimization for STP^{E-} is as hard as approximating the Steiner tree problem itself (analogous to general STP^{R-}). The two local modifications to remove an edge from the graph and to add an edge to the graph reduce to STP^{E+} resp. STP^{E-} in a straightforward manner.

The remaining four local modifications are the removal or addition of a required node or a Steiner node. It is known that the local modification where required or Steiner nodes are removed is as hard as Steiner tree approximation, even if we are given an optimal solution to the old problem [17]. We show that adding a required node or a Steiner node to the graph causes robust reoptimization to be as hard as STP approximation.

One of the key insights that leads to our improved algorithms is that for all local modifications that allow for robust reoptimization algorithms, we can replace the given Steiner tree by a k-restricted Steiner tree of about the same cost. A k-restricted Steiner tree has a limited size of the subgraphs where all required nodes are leaves (see Sect. 2 for a precise definition). At the same time, we have the promise that there is an almost optimal Steiner tree for the modified instance that is k-restricted. This property allows us to handle certain subgraphs of Steiner trees called full components. (i) We remove entire full components from the given Steiner tree and perform optimal computations to obtain a feasible solution to the modified instance, and (ii) we *guess* entire full components of the Steiner tree that we aim to compute. The new insights simplify and generalize the previous approaches to Steiner tree reoptimization and therefore give raise to more sophisticated analyses than before.[1]

1.2 Related Work

The concept of reoptimization was first mentioned by Schäffter [31] in the context of postoptimality analysis for a scheduling problem. Since then, the concept of reoptimization has been investigated for several different problems, including the traveling salesman problem [1, 5, 7, 15, 16, 30], the minimum latency problem [4, 26], the rural postman problem [3], fast reoptimization of the spanning tree problem [23], the knapsack problem [2], covering problems [12], the shortest common superstring problem [10], maximum-weight induced hereditary problems [22], and scheduling [6, 21, 31]. There are several overviews on reoptimization [4, 19, 24, 33].

The Steiner tree reoptimization problem in general weighted graphs was previously investigated in [11, 13, 14, 17, 18, 28], see Table 1.

[1] We note that with some additional effort, it would also be possible to adapt the technique of Bilò and Zych [14] and use them for our results.

2 Preliminaries

We denote a Steiner tree instance by (G, R, c), where G is an undirected graph, $R \subseteq V(G)$ is the set of required nodes, and $c : E(G) \to \mathbb{R}_{\geq 0}$ is a cost function. The *Steiner nodes* of (G, R, c) are the nodes $S = V(G) \backslash R$.

Since c is symmetric, we sometimes use the simplified notation $c(u, v) = c(v, u)$ instead of $c(\{u, v\})$.

For two graphs G, G', we define $G \cup G'$ to be the graph with node set $V(G) \cup V(G')$ and edge set $E(G) \cup E(G')$ (i.e., we do not keep multiple edges). For an edge e, $G - e$ is G with e removed from $E(G)$. We define $G - G'$ to be the graph with node set $V(G) \backslash (V(G') \cap S)$ and edge set $E(G) \backslash E(G')$. We emphasize that we do *not* remove required vertices.

In Steiner tree algorithms, it is standard to consider the edge-costs to be metric. The reason is that forming the metric closure (i.e., using the shortest path metric) does not change the cost of an optimal solution: if we replace an edge of a Steiner tree by the shortest path between the two ends, we obtain a valid Steiner tree again.

In the context of reoptimization, however, we cannot assume the cost function to be metric without loss of generality, because the triangle inequality restricts the effect of local changes. Therefore in the following we have to carefully distinguish between metric and general cost functions.

For a given Steiner tree, its *full components* are exactly those maximal subtrees that have all leaves are in R and all internal nodes are in S. Note that for a given Steiner tree T, we may remove leaves if they are not in R; we still have a Steiner tree, and its cost did not increase. Therefore we may assume that T is composed of full components. A *k-restricted Steiner tree* is a Steiner tree where each full component has at most k nodes from R.

Lemma 1 (Borchers and Du [20]) *For an arbitrary $\varepsilon > 0$, let $k \geq 2^{1/\varepsilon}$. Then for all Steiner tree instances (G, R, c) with optimal solution* Opt *of cost* opt, *there is a k-restricted Steiner tree T of cost at most* $(1 + \varepsilon)$opt *which can be obtained from* Opt *in polynomial time.*

Let T be a k-restricted Steiner tree in graph G. We replace the edge costs c by the shortest-path metric of G. In particular, since G is connected, it is a complete graph with respect to the changed cost function c. We assume that within T, the Steiner nodes $v \in V(T) \cap S$ have a degree of $\deg(v) \geq 3$. This is without loss of generality, since $\deg(v) \geq 2$ by the definition of k-restricted Steiner trees; if $\deg(v) = 2$ and u, w are the neighbors of v, $c(u, v) + c(v, w) \geq c(u, w)$. We replace $\{u, v\}, \{v, w\}$ by $\{u, w\}$ without increasing the cost of T and without changing the property that T is k-restricted.

Within the entire text, Opt denotes an optimal solution and opt denotes the cost of an optimal solution. We will often add sub- and superscripts to Opt and opt in order to distinguish between various types of (close to) optimal solutions.

3 Connecting Forests and Guessing Components

We state two algorithms that we will use repeatedly within the subsequent sections. The first algorithm, Connect, was introduced by Böckenhauer et al. [18] and has been used in all previous Steiner tree reoptimization results. The algorithm connects components of a Forest F of G in order to obtain a feasible Steiner tree T. The idea is that we start from a partial solution with few components that together contain all required vertices, and we use an exact computation to complete the solution. In Connect we use the following notation. Denote by G/V' for $V' \subseteq V(G)$ the contraction of V' in G. We write G/F instead of $G/V(F)$, if F is a subgraph of G. Note that after contracting a component there may be multiedges. Here, we treat multigraphs as simple graphs, where we only consider the cheapest edge of each multiedge. For ease of presentation, we slightly abuse notation and use the cost function c for both the graph before and the graph after the contraction.

Input : A Steiner tree instance (G,R,c),
a forest F in G with Steiner trees $F_1,F_2,\ldots,F_\ell$
Output : A Steiner tree T
Set $G' := G/F$ such that F_i is contracted to v_i;
Set $R' := \{v_i : V(F_i) \cap R \neq \emptyset\}$;
Compute a minimum Steiner tree T' of (G',R',c);
Obtain T from T' by expanding F.

Algorithm 1: Connect

Clearly, the graph T computed by Connect is a Steiner tree. If the number of components ℓ of the forest F given as input is a constant and we use the Dreyfus–Wagner algorithm [27][2] to compute T', Connect runs in polynomial time. The graph T computed by Connect is the minimum-cost Steiner tree that contains F, since all Steiner trees that contain F determine feasible solutions T'.

The second algorithm of this section is called Guess. It is motivated by the CONNECT algorithm of [14] and we present it here in a different manner. The algorithm Guess provides a mechanism to profit from guessing full components of an optimal k-restricted Steiner tree: we contract the guessed full components S_i to single vertices r_i and this way we obtain a new instance to which we apply known approximation algorithms. We call Guess by simply writing Guess(ℓ), if the instance and k are clear from the context and $\mathscr{A}$ is a β-approximation algorithm. For example Guess($3k$) means that $\ell = 3k$.

[2] We refer to Hougardy et al. [29] for an overview of further exact Steiner tree algorithms that, depending on the given parameters, may be faster than Dreyfus–Wagner.

Input : A Steiner tree instance (G,R,c) with c metric, numbers $\ell,k \in \mathbb{N}$, and a Steiner tree approximation algorithm $\mathscr{A}$
Output : A Steiner tree T
Run $\mathscr{A}$ on (G,R,c) and obtain a Steiner tree T;
foreach $\mathscr{S} = \{S_1, S_2, \ldots, S_\ell\}$ *such that* $S_i \subseteq V$ *with* $|S_i| \leq 2k$ *and* $2 \leq |S_i \cap R| \leq k$ *for* $1 \leq i \leq \ell$ **do**
  For each i, compute a minimum spanning tree T_i with $V(T_i) = S_i$;
  Contract each T_i to a required node r_i;
  Run $\mathscr{A}$ on the resulting instance;
  Obtain T' by expanding the contracted components of each r_i;
  if $c(T') < c(T)$ **then**
    Replace T by T'.

Algorithm 2: GUESS

Lemma 2 *For an arbitrary $\varepsilon > 0$, let k be the parameter obtained from Lemma 1. Let $\mathscr{A}$ be a polynomial-time β-approximation algorithm for the Steiner tree problem. Furthermore, let OPT_k be an optimal k-restricted solution of cost opt_k to the Steiner tree instance (G, R, c) where c is a metric. Then, for $\ell \in O_\varepsilon(1)$, GUESS runs in polynomial time and computes a Steiner tree T of cost at most $(1+\varepsilon)(\beta - \beta\zeta + \zeta)\mathrm{opt}$, where $\zeta\mathrm{opt}_k$ is the total cost of the ℓ maximum-weight full components of OPT_k and opt is the cost of an optimal solution.*

The index ε in the notation $O_\varepsilon(\cdot)$ means that ε is a constant.[3]

Proof We first analyze the running time of the algorithm. Since $\mathscr{A}$ runs in polynomial time, we only have to consider the number of families $\mathscr{S}$ that we have to test. This number is bounded from above by $\left(\sum_{i=2}^{2k} \binom{n}{i}\right)^\ell$, since we only choose sets of size at most $2k$. Since both k and ℓ are constants, this number is polynomial in n.

Next we analyze the cost of T. Since we assume that for each Steiner node $v \in S \cap V(\mathrm{OPT}_k)$, $\deg(v) \geq 3$, we conclude that all full components of OPT_k have at most $2k$ nodes. Therefore there is a family $\mathscr{S}$ considered by GUESS such that the classes of $\mathscr{S}$ are exactly the node sets of the ℓ maximum-weight full components of OPT_k. Contracting a minimum spanning tree T_i is equivalent to contracting the full component with required nodes $R \cap S_i$ in OPT_k. We finish the proof by applying a standard argument that was used, for instance, by Böckenhauer et al. [16]. The cost of an optimal Steiner tree before expanding the full components is bounded from above by $\mathrm{opt}_k - \zeta\mathrm{opt}_k$, and expanding the full components adds $\zeta\mathrm{opt}_k$. Therefore we obtain $c(T) \leq \beta(\mathrm{opt}_k - \zeta\mathrm{opt}_k) + \zeta\mathrm{opt}_k = (\beta - \beta\zeta + \zeta)\mathrm{opt}_k$.

[3] Therefore $k \in O_\varepsilon(1)$ means that there is a computable function $f : \mathbb{R} \to \mathbb{R}$ such that $k \leq f(\varepsilon)$.

By our choice of k and Lemma 1, $\text{opt}_k \leq (1+\varepsilon)\text{opt}$ and therefore $c(T) \leq (1+\varepsilon)(\beta - \beta\zeta + \zeta)\text{opt}$. □

In the subsequent proofs, we will repeatedly obtain a value η such that $\zeta \geq (\alpha - 1 - \varepsilon)\eta$, where α is the actual performance ratio of the considered approximation algorithm. By simple arithmetics and assuming that $(1+\varepsilon)(\beta - \beta\zeta + \zeta)$ tends to $(\beta - \beta\zeta + \zeta)$ for ε chosen sufficiently small, Lemma 2 implies

$$\alpha \leq \frac{\beta + \beta\eta - \eta + \varepsilon(\beta\eta - \eta)}{1 + \beta\eta - \eta}. \tag{1}$$

The reason for our assumption is that we can choose k in Lemma 1 and therefore the additional error is arbitrarily small.[4] We avoid complicated formalisms and instead slightly relax the approximation ratios in theorem statements by adding a value $O_{\beta,\varepsilon}(\delta)$ for a $\delta > 0$ that can be chosen arbitrarily small whenever the proofs use (1).

4 A Required Node Becomes a Steiner Node

The variant of the minimum Steiner tree reoptimization problem where a node is declared to be a Steiner node ($\text{STP}^{R-}_{\varepsilon}$) is defined as follows.

Given: A parameter $\varepsilon > 0$, a Steiner tree instance (G, R, c), a solution $\text{OPT}^{\text{old}}_{\varepsilon}$ to (G, R, c) such that $\text{opt}^{\text{old}}_{\varepsilon} \leq (1+\varepsilon)\text{opt}^{\text{old}}$, and a node $t \in R$.
Solution: A Steiner tree solution to $(G, R\backslash\{t\}, c)$.

An instance of $\text{STP}^{R-}_{\varepsilon}$ is a tuple $\left(G, R, c, \text{OPT}^{\text{old}}_{\varepsilon}, t\right)$. If $\varepsilon = 0$, we skip the index and write STP^{R-}. Without loss of generality we assume that c is a metric: we may use the metric closure since the local modification does not change G or c.

The algorithm DeclareSteiner starts with reducing the instance to one where the changed required node has a degree of at least two, using a known technique. Afterwards it transforms the given solution to a k-restricted Steiner tree (note that the order of these two steps is important). The remaining algorithm outputs the best of three solutions that intuitively can be described as follows: we either keep the old solution; or we remove up to three full components incident to t to obtain a partial solution that we complete again using Connect; or we guess a partial solution that is at least as large as the $3k$ largest full-components of an optimal solution and complete these components to a solution using the best available approximation algorithm.

[4] Note that in contrast to the error from Lemma 1, we *cannot* control the error of the given solutions.

Input : An instance $(G, R, c, \text{OPT}_\varepsilon^{\text{old}}, t)$ of $\text{STP}_\varepsilon^{R-}$
Output : A Steiner tree T
while $\deg_{\text{OPT}_\varepsilon^{\text{old}}}(t) = 1$ **do** `// We assume that either` $\varepsilon = 0$ `or` $\deg_{\text{OPT}_\varepsilon^{\text{old}}}(t) > 1$
  Set $t' := \text{child}(t)$; `// The node adjacent to` t `in` $\text{OPT}_\varepsilon^{\text{old}}$
  Remove t from $\text{OPT}_\varepsilon^{\text{old}}$ and R, rename t' to t, and set $t \in R$;
  `// Now` $(G, R, c, \text{OPT}_\varepsilon^{\text{old}}, t)$ `is the changed instance`
Transform $\text{OPT}_\varepsilon^{\text{old}}$ to a k-restricted solution $\text{OPT}_{\varepsilon,k}^{\text{old}}$ such that $\text{opt}_{\varepsilon,k}^{\text{old}} \leq (1+\varepsilon_k)\text{opt}_\varepsilon^{\text{old}}$, where ε_k tends to 0 for large enough k;
Set $T_1 := \text{OPT}_\varepsilon^{\text{old}}$; `// Note that` $\text{opt}_\varepsilon^{\text{old}} \leq \text{opt}_{\varepsilon,k}^{\text{old}}$
Let $C_1^t, C_2^t, \ldots$ be the full components of $\text{OPT}_{\varepsilon,k}^{\text{old}}$ such that $t \in V(C_i^t)$ for all i and $c(C_i^t) \leq c(C_j^t)$ for $i < j$;
Set $F := \text{OPT}_{\varepsilon,k}^{\text{old}} - C_1^t - C_2^t - C_3^t$; `// Ignore` C_3^t `if it does not exist`
Set $T_2 := \text{CONNECT}(F)$;
Set $T_3 := \text{GUESS}(3k)$;
Set $T = T_i$ with $i = \min \arg_{j \in \{1,2,3\}} \{c(T_j)\}$.

Algorithm 3: DECLARESTEINER

The following theorem indicates that in general we have to require $\varepsilon = 0$ for instances of $\text{STP}_\varepsilon^{R-}$ with $\deg(t) = 1$.

Theorem 1 *For an arbitrary $\varepsilon > 0$, let $\mathscr{A}$ be a polynomial-time α-approximation algorithm for $\text{STP}_\varepsilon^{R-}$. Then there is a polynomial-time α-approximation algorithm for the Steiner tree problem.*

Proof Given a Steiner tree instance (G, R, c), let opt^{new} be the cost of an optimal solution. We construct a $\text{STP}_\varepsilon^{R-}$ instance $\left(G', R', c', \text{OPT}_\varepsilon^{\text{old}}, t\right)$ from (G, R, c). We first compute a minimum spanning tree $\tilde{T}$ of $G[R]$. Note that $G[R]$ is a complete graph since we assume c to be metric, and $c(\tilde{T}) \leq 2\text{opt}^{\text{new}}$, as shown by Takahashi and Matsuyama [32]; we assume w.l.o.g. that $\alpha < 2$. We obtain G' by combining G and a new node t as follows. We set $V(G') := V(G) \cup \{t\}$ and $E(G') = E(G) \cup \{t, t'\}$ for a node $t' \in R$. Then we obtain c' from c by setting $c'(t, t') = c(\tilde{T}) \cdot \max\{1, (1-\varepsilon)/\varepsilon\}$ and forming the metric closure.[5] We set $R' = R \cup \{t\}$ and obtain a solution to (G', R', c') by adding $\{t, t'\}$ to $\tilde{T}$. We show later that the computed solution is $\text{OPT}_\varepsilon^{\text{old}}$. Finally, we obtain the Steiner tree T by applying $\mathscr{A}$ to $\left(G', R', c', \text{OPT}_\varepsilon^{\text{old}}, t\right)$.

Observe that T cannot contain an edge incident to t, since all of those edges have at least the cost of $\tilde{T}$. Therefore T is a Steiner tree of (G, R, c). Conversely, all Steiner trees of (G, R, c) are feasible solutions to $\left(G', R', c', \text{OPT}_\varepsilon^{\text{old}}, t\right)$. We conclude that T provides an α approximation, i.e., T is a feasible solution to (G, R, c) and $c(T) \leq \alpha\text{opt}^{\text{new}}$.

To finish the proof, we have to show that $\text{OPT}_\varepsilon^{\text{old}}$ was a valid solution given to $\mathscr{A}$, i.e., its cost $\text{opt}_\varepsilon^{\text{old}}$ is at most a factor $(1+\varepsilon)$ larger than optimum. Clearly, $\text{OPT}_\varepsilon^{\text{old}}$

[5] Taking the maximum of the two values ensures that $c'(t, t') \geq c(\tilde{T})$, even if $\varepsilon \geq 1/2$.

is a Steiner tree of (G', R', c'). Let opt^{old} be the cost of an optimal Steiner trees for (G', R', c').

$$\frac{\text{opt}_{\varepsilon}^{\text{old}}}{\text{opt}^{\text{old}}} = \frac{c(\tilde{T}) + c(t,t')}{\text{opt}^{\text{new}} + c(t,t')} \leq \frac{2\text{opt}^{\text{new}} + c(t,t')}{\text{opt}^{\text{new}} + c(t,t')} \leq 1 + \frac{\text{opt}^{\text{new}}}{\text{opt}^{\text{new}} + \text{opt}^{\text{new}} \cdot \frac{1-\varepsilon}{\varepsilon}} = 1 + \varepsilon.$$

□

For all remaining cases, DeclareSteiner profits from knowing $\text{Opt}_{\varepsilon}^{\text{old}}$.

Theorem 2 *Let* $\left(G, R, c, \text{Opt}_{\varepsilon}^{\text{old}}, t\right)$ *be an instance of* $\text{STP}_{\varepsilon}^{R-}$ *with* $\deg_{\text{Opt}_{\varepsilon}^{\text{old}}}(t) \geq 2$ *or* $\varepsilon = 0$. *Then, for an arbitrary* $\delta > 0$, DeclareSteiner *is an approximation algorithm for* $\text{STP}_{\varepsilon}^{R-}$ *with performance guarantee*

$$\frac{(10\beta - 7 + 2\varepsilon - 2\varepsilon\beta)(1+\varepsilon)}{7\beta - 4 + 5\varepsilon - 2\varepsilon\beta} + O_{\beta,\varepsilon}(\delta).$$

For the approximation ratio $\beta = \ln(4) + \varepsilon''$ from [25] with ε'' and δ chosen sufficiently small, we obtain an approximation ratio of less than $1.204 \cdot (1 + \varepsilon)$.

4.1 Proof of Theorem 2

Since k is a constant, all steps of DeclareSteiner except for the call of Connect clearly run in polynomial time. To see that also the call of Connect does, observe that removing the edges and Steiner nodes of a full component increases the number of connected components by at most $k - 1$.

We continue with showing the claimed upper bound on the performance guarantee. Before we show the main result, we introduce two simplification steps. First, we show that we can restrict our attention to the case $\deg(t) = 2$ in $\text{Opt}_{\varepsilon,k}^{\text{old}}$. Our analysis simultaneously gives a new proof for the previous best reoptimization result [14]. Subsequently we reduce the class of considered instances to those where all optimal solution to $(G, R\backslash\{t\}, c)$ have a special structure.

We start with analyzing the case where $\deg(t) = 1$. If this case appears in the while loop, by our assumption we have $\varepsilon = 0$ and thus $\text{Opt}_{\varepsilon}^{\text{old}}$ is an optimal solution. The transformation of DeclareSteiner within the while loop reduces the instance to one where $\deg(t) \geq 2$ [18]. When transforming the resulting solution $\text{Opt}_{\varepsilon}^{\text{old}}$ to $\text{Opt}_{\varepsilon,k}^{\text{old}}$, generally t could become a degree-one vertex. We use, however, that this is not the case when applying the algorithm of Borchers and Du [20]: The algorithm considers the full components separately, which implies that initially the degrees of all required vertices are one. Each full component is replaced by a graph where each required vertex has a degree of at least one. Consequently, the degree of no required vertex is decreased.

For the remaining proof, we assume $\deg_{\text{Opt}_{\varepsilon,k}^{\text{old}}}(t) \geq 2$. We prove the following technical lemma, which is needed for our subsequent argumentation.

Lemma 3 *There is a collection $\mathscr{C}$ of at most $3k$ full components of $\textsc{Opt}_k^{\text{new}}$ such that $F \cup \mathscr{C}$ is a connected graph.*

Proof Observe that F has less than $3k$ connected components, and each of them contains nodes from R. We use that the full components of $\textsc{Opt}_k^{\text{new}}$ only intersect in R. Since $\textsc{Opt}_k^{\text{new}}$ is connected, by the pigeonhole principle it has a full component C that contains required nodes from two distinct components of F. Thus adding C to F reduces the number of components. Now the claim follows inductively. □

Let $\alpha = c(T)/\text{opt}^{\text{new}} \geq 1$ be the performance ratio of DeclareSteiner. Thus, in the following we want to determine an upper bound on α. We may assume

$$\text{opt}_{\epsilon,k}^{\text{old}} \geq \alpha \text{opt}^{\text{new}} \tag{2}$$

since otherwise, T_1 already gives an approximation ratio better than α.

We define $\gamma = c(\textsc{Connect}(F)) - c(F)$, the cost to connect F. Let d be the number of full components removed from $\textsc{Opt}_{\epsilon,k}^{\text{old}}$ to obtain F, i.e., $d \in \{2, 3\}$.

Lemma 4 *For an arbitrary $\delta > 0$, the performance ratio α of* DeclareSteiner *is bounded from above by $1 + \frac{\beta-1+\epsilon(\beta-1)(d+1)}{1+(\beta-1)d+\epsilon} + O_{\beta,\epsilon}(\delta)$.*

Proof We have $c(C_1^t) + c(C_2^t) + c(C_3^t) \geq d \cdot c(C_1^t)$ assuming $c(C_1^t) \leq c(C_i^t)$ for $i \leq d$.

We determine the following constraints. Since $C_1^t + \textsc{Opt}^{\text{new}}$ contains a feasible solution to (G, R, c),

$$\text{opt}^{\text{new}} + c(C_1^t) \geq \text{opt}^{\text{old}}. \tag{3}$$

Furthermore,

$$\text{opt}_{\epsilon,k}^{\text{old}} - c(C_1^t) - c(C_2^t) - c(C_3^t) + \gamma \geq \alpha \text{opt}^{\text{new}} \tag{4}$$

since $c(T_2)$ is at most as large as the left hand side of (4).

We use (3) to replace $\text{opt}_{\epsilon,k}^{\text{old}}$ in (2) to obtain

$$c(C_1^t) \geq \frac{\alpha - 1 - \varepsilon}{(1+\varepsilon)(1+\delta)} \text{opt}^{\text{new}}. \tag{5}$$

By applying (3) and (5) to (4), we obtain

$$\gamma \geq \frac{d}{(1+\varepsilon)(1+\delta)} \cdot (\alpha - 1 - \varepsilon)\text{opt}^{\text{new}}. \tag{6}$$

Finally, $\zeta \geq \gamma/\text{opt}_k^{\text{new}}$, by Lemma 3. Therefore, due to Lemma 2,

$$(1+\delta)\left(\beta - \beta\gamma/\text{opt}_k^{\text{new}} + \gamma/\text{opt}_k^{\text{new}}\right) \geq \alpha. \tag{7}$$

Now the claim follows if we replace γ in (7) by the right hand side of (6), where we used that $\beta \geq 1$. □

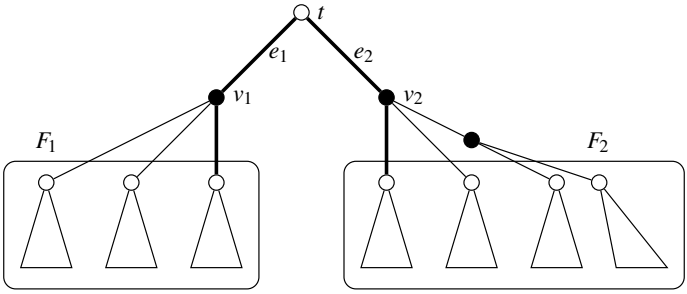

Fig. 1 Structure of $\text{OPT}^{\text{old}}_{\varepsilon,k}$. The paths P_1 and P_2 are drawn with thick lines

We note that for $d = 2$ and $\varepsilon = 0$, the upper bound on the performance guarantee due to Lemma 4 matches the previously best performance guarantee [14]. For $d = 3$, the value is better than the aimed-for value from Theorem 2. Observe that a straightforward extension of DeclareSteiner would allow us to consider values of d larger than three.

Due to Lemma 4, in the following we may assume that $\deg(t) = 2$. Next, we analyze the structure of $\text{OPT}^{\text{old}}_{\varepsilon,k}$ and OPT^{new}. Let $R_1 = (R \cap V(C_1^t))\backslash\{t\}$ and $R_2 = (R \cap V(C_2^t))\backslash\{t\}$. We partition F into forests F_1 and F_2 such that F_1 contains exactly the trees T of F with $V(T) \cap R_1 \neq \emptyset$ and F_2 contains the remaining trees T', with $V(T') \cap R_2 \neq \emptyset$ (see Fig. 1).

Let $v_1 \in V(C_1^t)$ and $v_2 \in V(C_2^t)$ such that $e_1 = \{t, v_1\} \in E(C_1^t)$ and $e_2 = \{t, v_2\} \in E(C_2^t)$. Let P_1 be a minimum-cost path in $\text{OPT}^{\text{old}}_{\varepsilon,k}$ from t to R_1 and let P_2 be a minimum-cost path in $\text{OPT}^{\text{old}}_{\varepsilon,k}$ from t to R_2. Observe that P_1 contains e_1 and that P_2 contains e_2. We define $\kappa_1 := c(P_1)$, $\kappa_1' := c(e_1)$, and $\kappa_1'' = c(P_1) - c(e_1)$. Analogously, $\kappa_2 := c(P_2)$, $\kappa_2' := c(e_2)$, and $\kappa_2'' = c(P_2) - c(e_2)$. Note that we do not exclude that $v_1 \in R_1$ or $v_2 \in R_2$. In this case κ_1'' resp. κ_2'' are zero.

To simplify the presentation, we define $\kappa' := (\kappa_1' + \kappa_2')/2$ and $\kappa'' := (\kappa_1'' + \kappa_2'')/2$. Since P_1, P_2 are minimum-cost paths, $c(C_1^t) \geq \kappa_1' + 2\kappa_1''$ and $c(C_2^t) \geq \kappa_2' + 2\kappa_2''$, which implies

$$c(C_1^t) + c(C_2^t) \geq 2\kappa' + 4\kappa''. \tag{8}$$

We have $\text{opt}^{\text{new}} + \kappa_1' + \kappa_1'' \geq \text{opt}^{\text{old}}$ and $\text{opt}^{\text{new}} + \kappa_2' + \kappa_2'' \geq \text{opt}^{\text{old}}$. Therefore,

$$\text{opt}^{\text{new}} + \kappa' + \kappa'' \geq \text{opt}^{\text{old}}. \tag{9}$$

Lemma 5 *Suppose there are at least two edge-disjoint paths in* $\text{OPT}^{\text{new}}_k$ *between* $V(F_1)$ *and* $V(F_2)$*. Then, for an arbitrary* $\delta > 0$*, the performance guarantee of* DeclareSteiner *is bounded from above by* $\frac{(11\beta-8)(1+\varepsilon)}{8\beta-5+3\varepsilon} + O_{\beta,\varepsilon}(\delta)$.

Proof Let P' and P'' be two edge-disjoint paths within $\text{OPT}^{\text{new}}_k$ between $V(F_1)$ and $V(F_2)$ such that none of their internal nodes are in $V(\text{OPT}^{\text{old}}_{\varepsilon,k})$. Without loss of

generality, we assume that $c(P') \leq c(P'')$. Then, additionally to the previous constraints, we obtain the following.

$$\text{opt}^{\text{old}}_{\varepsilon,k} - 2\kappa' + c(P') = \text{opt}^{\text{old}}_{\varepsilon,k} - \kappa'_1 - \kappa'_2 + c(P') \geq \alpha \text{opt}^{\text{new}} \tag{10}$$

$$\zeta \cdot \text{opt}^{\text{new}}_k \geq c(P') + c(P'') \geq 2c(P') \tag{11}$$

From (9) and (10), we obtain

$$(1+\delta) \cdot c(P') \geq (\alpha - 1 - \varepsilon)\text{opt}^{\text{new}} + (1-\varepsilon)\kappa' - (1+\varepsilon)\kappa'', \tag{12}$$

and thus, due to (11) and (2),(9),

$$\begin{aligned} \zeta \text{opt}^{\text{new}}_k &\geq 2((\alpha - 1 - \varepsilon)\text{opt}^{\text{new}} + (1-\varepsilon)\kappa' - (1+\varepsilon)\kappa'')/(1+\delta) \\ &\geq \frac{4}{(1+\varepsilon)(1+\delta)}(\alpha - 1 - \varepsilon)\text{opt}^{\text{new}} - 4\kappa''. \end{aligned} \tag{13}$$

Furthermore, by using (8) and (9) in (4), we obtain

$$(1 + O(\delta))\gamma \geq (\alpha - 1 - \varepsilon)\text{opt}^{\text{new}} + (1-\varepsilon)\kappa' + (3-\varepsilon)\kappa''.$$

and thus, due to (2) and (9) and the fact that $\zeta \text{opt}^{\text{new}}_k \geq \gamma$,

$$(1 + O(\delta))\zeta \text{opt}^{\text{new}}_k \geq \frac{2}{(1+\varepsilon)}(\alpha - 1 - \varepsilon)\text{opt}^{\text{new}} + 2\kappa''. \tag{14}$$

A linear combination of (13) and (14) with coefficients one and two gives $(1 + O(\delta))\zeta \geq \frac{8(\alpha-1-\varepsilon)}{3(1+\varepsilon)}$. Using (1) we obtain

$$\alpha \leq \frac{(11\beta - 8)(1+\varepsilon)}{8\beta - 5 + 3\varepsilon} + O_{\beta,\varepsilon}(\delta).$$

□

Since the value obtained by Lemma 5 is better than the aimed-for ratio, from now on we can restrict our focus to instances where in $\text{OPT}^{\text{new}}_k$, there are no two edge-disjoint paths between F_1 and F_2. In particular, this means that there is exactly one full component L in $\text{OPT}^{\text{new}}_k$ that connects F_1 and F_2. Since we assumed that there are no Steiner nodes of degree two in $\text{OPT}^{\text{new}}_k$, there is exactly one edge e_L in L such that removing e_L leaves two connected components of $\text{OPT}^{\text{new}}_k$, one containing R_1 and the other one containing R_2. Let P_L be a minimum-cost path between $V(F_1)$ and $V(F_2)$ in L (and thus P_L clearly contains e_L). Let P^1_L be the subpath of P_L between F_1 and e_L and let P^2_L be the subpath of P_L between F_2 and e_L. We define $\lambda := c(P_L)$, $\lambda' := c(e_L)$, $\lambda''_1 := c(P^1_L)$, and $\lambda''_2 := c(P^2_L)$. Similar to above, we define $\lambda'' := (\lambda''_1 + \lambda''_2)/2$. Note that $\lambda - \lambda' = 2\lambda''$. It follows easily that $c(L) \geq \lambda' + 4\lambda''$. Let L' be a forest with a minimum number of full components from $\text{OPT}^{\text{new}}_k$ such that $\text{OPT}^{\text{old}}_{\varepsilon,k} - C^t_1 - C^t_2 + L'$ is connected. From Lemma 3, we obtain that L' contains at most $3k$ full components and thus

we considered guessing L' when computing T_3 in DeclareSteiner. We define $\xi := c(L') - \lambda' - 4\lambda''$. Since L' contains L, ξ is non-negative.

To find an upper bound on the value of α, we maximize α subject to the constraints (2), (9) and the following constraints.

By removing e_L from $\text{Opt}_k^{\text{new}}$ and adding the paths P_1 and P_2, we obtain a feasible solution to (G, R, c); conversely, by removing e_1 and e_2 from $\text{Opt}_{\varepsilon,k}^{\text{old}}$ and adding P_L, we obtain a feasible solution to $(G, R\backslash t, c)$ that is considered in T_2. Therefore

$$\text{opt}_k^{\text{new}} + 2\kappa' + 2\kappa'' - \lambda' \geq \text{opt}^{\text{old}}, \tag{15}$$

$$\text{opt}_{\varepsilon,k}^{\text{old}} - 2\kappa' + \lambda' + 2\lambda'' \geq \alpha \text{opt}^{\text{new}}. \tag{16}$$

In T_2 we also consider to remove C_1^t, C_2^t completely and to add L'. Therefore

$$\text{opt}_{\varepsilon,k}^{\text{old}} - 2\kappa' - 4\kappa'' + \lambda' + 4\lambda'' + \xi \geq \alpha \text{opt}^{\text{new}}. \tag{17}$$

Due to Lemma 2, we may assume

$$\beta - \beta\zeta + \zeta \geq \alpha/(1+\delta). \tag{18}$$

In T_3, one of the considered guesses is L' and therefore

$$\zeta \cdot \text{opt}_k^{\text{new}} \geq \lambda' + 4\lambda'' + \xi. \tag{19}$$

We scale the values such that $\text{opt}^{\text{new}} = 1$. Then we perform the following replacements. We replace opt^{old} in (9) and in (15) by using (2); we use (9) to replace opt^{old} in (16); we use (9) to replace opt^{old} in (17). We keep (18) and (19). This way we obtain a linear program that maximizes α subject to the following constraints, where we ignore the dependence on δ.

$$\begin{aligned}
-\kappa' - \kappa'' + \alpha/(1+\varepsilon) &\leq 1 \\
-2\kappa' - 2\kappa'' + \lambda' + \alpha/(1+\varepsilon) &\leq 1 \\
(1-\varepsilon)\kappa' - (1+\varepsilon)\kappa'' - \lambda' - 2\lambda'' + \alpha &\leq 1+\varepsilon \\
(1-\varepsilon)\kappa' + (3-\varepsilon)\kappa'' - \lambda' - 4\lambda'' - \xi + \alpha &\leq 1+\varepsilon \\
(\beta-1)\zeta + \alpha &\leq \beta \\
\lambda' + 4\lambda'' + \xi - \zeta &\leq 0
\end{aligned}$$

Now we obtain the dual linear program

$$\begin{aligned}
\text{minimize} \quad & y_1 + y_2 + (1+\varepsilon)y_3 + (1+\varepsilon)y_4 + \beta y_5 & \\
\text{s.t.:} \quad & -y_1 - 2y_2 + (1-\varepsilon)y_3 + (1-\varepsilon)y_4 & \geq 0 \\
& -y_1 - 2y_2 - (1+\varepsilon)y_3 + (3-\varepsilon)y_4 & \geq 0 \\
& y_2 - y_3 - y_4 + y_6 & \geq 0 \\
& -2y_3 - 4y_4 + 4y_6 & \geq 0 \\
& -y_4 + y_6 & \geq 0 \\
& (\beta-1)y_5 - y_6 & \geq 0 \\
& y_1/(1+\varepsilon) + y_2/(1+\varepsilon) + y_3 + y_4 + y_5 & \geq 1
\end{aligned}$$

To finish the proof, we consider the following feasible solution. We set

$$y_1 = \frac{2(\beta-1)(1+\varepsilon)(1-2\varepsilon)}{7\beta-4+5\varepsilon-2\varepsilon\beta}; \qquad y_2 = y_1/2;$$
$$y_3 = y_1/(1-2\varepsilon); \qquad y_4 = y_1/(1-2\varepsilon);$$
$$y_5 = \frac{3(1+\varepsilon)(1-2\varepsilon)}{(1-2\varepsilon)(7\beta-4+5\varepsilon-2\varepsilon\beta)}; \qquad y_6 = \frac{3y_1}{2(1-2\varepsilon)}.$$

With these values, the objective function value matches the claimed value in Theorem 2. By weak duality, we obtained an upper bound on the value of α in the primal linear program, which finishes the proof.

5 A Steiner Node Becomes a Required Node

In this section, instead of removing nodes from R, we consider the problem to add nodes to R, i.e., we declare a Steiner node to be a required node. Formally the problem $\text{STP}^{R+}_{\varepsilon}$ is defined as follows.

Given: A parameter $\varepsilon > 0$, a Steiner tree instance (G, R, c), a solution $\text{OPT}^{\text{old}}_{\varepsilon}$ to (G, R, c) such that $\text{opt}^{\text{old}}_{\varepsilon} \leq (1+\varepsilon)\text{opt}^{\text{old}}$, and a Steiner node $t \in V(G)\backslash R$.

Solution: A Steiner tree solution to $(G, R \cup \{t\}, c)$.

Input : An instance $(G, R, c, \text{OPT}^{\text{old}}_{\varepsilon}, t)$ of $\text{STP}^{R+}_{\varepsilon}$
Output : A Steiner tree T
Transform $\text{OPT}^{\text{old}}_{\varepsilon}$ to a k-restricted solution $\text{OPT}^{\text{old}}_{\varepsilon,k}$ such that $\text{opt}^{\text{old}}_{\varepsilon,k} \leq (1+\varepsilon_k)\text{opt}^{\text{old}}_{\varepsilon}$, where ε_k tends to 0 for large k ;
Set $e := \{v,t\}$ where $v = \arg\min_{v' \in V(\text{OPT}^{\text{old}}_{\varepsilon,k})} c(\{v',t\})$;
Set $T_1 := \text{OPT}^{\text{old}}_{\varepsilon,k} + e$;
Set $T_2 := T_1$;
foreach *Full component* C *of* $\text{OPT}^{\text{old}}_{\varepsilon,k}$ **do**
  Set $F := \text{OPT}^{\text{old}}_{\varepsilon,k} - C$;
  if $c(\text{CONNECT}(F)) < c(T_2)$ *(for* $R \cup \{t\}$*)* **then**
    $T_2 := \text{CONNECT}(F)$;
Set $T_3 := \text{GUESS}(k+1)$;
Set $T = T_i$ with $i = \min \arg_{j \in \{1,2,3\}} \{c(T_j)\}$.

Algorithm 4: DECLAREREQUIRED

Unlike in the case of terminal removal, in the following theorem we may allow $\text{OPT}^{\text{old}}_{\varepsilon}$ with $\varepsilon > 0$ without any exceptions.

Theorem 3 *Let* $\left(G, R, c, \text{OPT}^{\text{old}}_{\varepsilon}, t\right)$ *be an instance of* $\text{STP}^{R+}_{\varepsilon}$. *Then, for an arbitrary* $\delta > 0$, DECLAREREQUIRED *is a* $\left(\frac{(10+7\varepsilon)\beta-7\cdot(1+\varepsilon)}{7\beta-4} + O_{\beta,\varepsilon}(\delta)\right)$*-approximation algorithm for* $\text{STP}^{R+}_{\varepsilon}$.

5.1 Proof of Theorem 3

We prove the theorem in three steps, according to the degree of t in $\text{OPT}_k^{\text{new}}$.

Let us first assume that $\deg(t) = 1$. Then there is a well defined edge $\{t, t'\}$ in $\text{OPT}_k^{\text{new}}$. Thus we may solve $(G, R, c, \text{OPT}_\varepsilon^{\text{old}}, t')$ and add $\{t, t'\}$ afterwards if t is not included. If $t' \in R$, $\text{OPT}_\varepsilon^{\text{old}}$ is good enough already. Otherwise $\deg(t') \geq 2$ in the k-restricted optimal solution to $(G, R, c, \text{OPT}_\varepsilon^{\text{old}}, t')$ that we obtain from $\text{OPT}_k^{\text{new}}$ by removing t. Adding $\{t, t'\}$ after obtaining a solution to $(G, R, c, \text{OPT}_\varepsilon^{\text{old}}, t')$ does not increase the approximation ratio since both the computed and the optimal solution are increased by the same amount. Note that in DeclareRequired, the transformation is implicitly contained. If $t' \in R$, T_1 provides a $(1+\varepsilon)$-approximation. Since for both instances we consider the same full components C, T_2 is at most as large as the solution obtained for $(G, R, c, \text{OPT}_\varepsilon^{\text{old}}, t')$ together with $\{t, t'\}$. In T_3, we implicitly considered $\deg(t) = 1$ by choosing $k+1$ as parameter, whereas k is sufficient for the remaining proof. Therefore, in the following we assume $\deg(t) \geq 2$.

For the remaining proof where $\deg(t) \geq 2$ in $\text{OPT}_k^{\text{new}}$, we introduce the following notation. Let us fix a close-to-optimal solution $\text{OPT}_k^{\text{new}}$ to $(G, R \cup \{t\}, c)$.

We consider t as the root of $\text{OPT}_k^{\text{new}}$ and fix two subtrees F_1 and F_2 such that there is a full component L of $\text{OPT}_\varepsilon^{\text{old}}$ with $V(L) \cap V(F_1) \neq \emptyset$ and $V(L \cap V(F_2) \neq \emptyset$. Thus L connects F_1 and F_2. We define $\lambda = c(L)$. The trees F_1 and F_2 exist, because $\text{OPT}_\varepsilon^{\text{old}}$ is a connected graph. Let us now consider the graph $H = F_1 \cup F_2 \cup L$. Note that since nodes not in R have a degree of at least three, there is at most one edge e_L in L that is a cut edge of H. If e_L exists, we set $\lambda' = c(e_L)$; otherwise, $\lambda' = 0$. If there are more than one components of $\text{OPT}_\varepsilon^{\text{old}}$ connecting F_1 and F_2, we change the names such that λ' is minimal.

We set $R_1 := R \cap V(F_1)$ and $R_2 := R \cap V(F_2)$. Then $\rho := \min\{c(\{v_1, v_2\}) : v_1 \in R_1, v_2 \in R_2\}$. We define $p_1 = \min\{c(\{r, t\}) : r \in R_1\}$, $p_2 = \min\{c(\{r, t\}) : r \in R_2\}$, and $p := \min\{c(\{r, t\}) : r \in R\}$. Furthermore, let P_1 and P_2 be the minimum-cost paths from t to R_1 resp. R_2 in $\text{OPT}_k^{\text{new}}$. Let e_1 and e_2 be the edges of P_1 resp. P_2 incident to t. We define, for $i \in \{1, 2\}$, $p_i' := c(e_i)$ and $p_i'' := c(P_i) - p_i'$. Then $p' := (p_1' + p_2')/2$ and $p'' := (p_1'' + p_2'')/2$. In particular,

$$p \leq p' + p''. \tag{20}$$

$$p_1 + p_2 \leq 2(p' + p''). \tag{21}$$

Before we continue with the second case, let us state two general constraints. Since OPT^{new} is a feasible solution to (G, R), we have

$$\text{opt}^{\text{new}} \geq \text{opt}^{\text{old}}. \tag{22}$$

We get

$$\text{opt}_\varepsilon^{\text{old}} + p \geq \alpha \text{opt}^{\text{new}}, \tag{23}$$

since otherwise T_1 is a good enough solution. The constraints (22) and (23) imply

$$p \geq (\alpha - 1 - \varepsilon)\mathrm{opt}^{\mathrm{new}}. \tag{24}$$

Next we reduce the problem to $\deg(t) = 2$.

Lemma 6 *For an arbitrary $\delta > 0$,* DeclareRequired *is a $\left(\frac{4\beta-3+(3\beta-3)\varepsilon}{3\beta-2} + O_{\beta,\varepsilon}(\delta)\right)$-approximation algorithm for instances $(G, R, c, \mathrm{OPT}_{\varepsilon}^{\mathrm{old}}, t)$ with $\deg(t) > 2$ in $\mathrm{OPT}_{k}^{\mathrm{new}}$.*

Proof By definition of p, we conclude that $\zeta \mathrm{opt}_{k}^{\mathrm{new}} \geq 3p$. Using (24), we conclude that $(1+\delta)\zeta \geq 3(\alpha - 1 - \varepsilon)$. Therefore, by (1), we obtain the claimed result. □

For the last case where $\deg(t) = 2$, we obtain the following additional constraints. Since L contains a path between R_1 and R_2 of length at least ρ, and e_L is the only cut edge (if there is any).

$$\lambda \geq 2 \cdot \rho - \lambda' \tag{25}$$

In other words, if we double e_L then there are two edge-disjoint paths between R_1 and R_2 in $\mathrm{OPT}_{k}^{\mathrm{new}}$. Furthermore,

$$\mathrm{opt}_{k}^{\mathrm{new}} - 2p' + \rho \geq \mathrm{opt}^{\mathrm{old}} \tag{26}$$

since adding a direct edge between F_1 and F_2 reconnects the graph after removing e_1 and e_2. From (23) and (26), we obtain

$$(1+\delta)p \geq (\alpha - 1 - \varepsilon)\mathrm{opt}^{\mathrm{new}} + (1+\varepsilon)(2p' - \rho). \tag{27}$$

To get a feasible solution to $(G, R \cup \{t\}, c)$, we can remove e_L from L (if e_L exists) and connect the graph again by adding minimum-cost edges from R_1 and R_2 to t. Therefore T_2 is good enough unless

$$\mathrm{opt}_{\varepsilon,k}^{\mathrm{old}} - \lambda' + p_1 + p_2 \geq \alpha \mathrm{opt}^{\mathrm{new}}. \tag{28}$$

We obtain the following constraint from (22) and (28).

$$p_1 + p_2 \geq (\alpha - 1 - \varepsilon)\mathrm{opt}^{\mathrm{new}}/(1+\delta) + \lambda'. \tag{29}$$

Finally, in T_3 we guess a collection of trees that combine the connected components of $\mathrm{OPT}_{\varepsilon,k}^{\mathrm{old}} - L$, and therefore

$$\zeta \mathrm{opt}_{k}^{\mathrm{new}} \geq \lambda. \tag{30}$$

Also from T_3 we obtain

$$\zeta \mathrm{opt}_{k}^{\mathrm{new}} \geq 2p' + 4p'' \geq 2p + 2p''. \tag{31}$$

From (25) and (30) we obtain

$$\zeta \mathrm{opt}_{k}^{\mathrm{new}} \geq 2\rho - \lambda'. \tag{32}$$

From (21), (29), (32), we obtain

$$\rho/O_\varepsilon(1+\delta) \le p' + p'' + (\zeta + 1 - \alpha + \varepsilon)\mathrm{opt}^{\mathrm{new}}/2. \tag{33}$$

Then from (20), (27), (33) we obtain

$$(1+\varepsilon)(1+O_\varepsilon(\delta))p'' \ge (\alpha(3+\varepsilon) - (3+\varepsilon)(1+\varepsilon) - \zeta(1+\varepsilon))\mathrm{opt}^{\mathrm{new}}/4 + \varepsilon p/2. \tag{34}$$

From (34), together with (24) and (31), we get $(1+O_\varepsilon(\delta))\zeta \ge 7(\alpha - 1 - \varepsilon)/3$. Therefore applying (1) finishes the proof.

6 Increased Edge Cost

We now consider the reoptimization variant where the edge cost of one edge is increased, $\mathrm{STP}^{E+}_\varepsilon$. If e is the edge of G with increased cost, we define $c^{\mathrm{new}} : E(G) \to \mathbb{R}_{\ge 0}$ as $c^{\mathrm{new}}(e') = c(e')$ for all edges $e' \in E(G)\backslash\{e\}$ and $c^{\mathrm{new}}(e)$ is the increased cost. Then the formal definition of the reoptimization variant is as follows.

Given: A parameter $\varepsilon > 0$, a Steiner tree instance (G, R, c), a solution $\mathrm{OPT}^{\mathrm{old}}_\varepsilon$ to (G, R, c) such that $\mathrm{opt}^{\mathrm{old}}_\varepsilon \le (1+\varepsilon)\mathrm{opt}^{\mathrm{old}}$, and a cost $c^{\mathrm{new}}(e) \ge c(e)$ for an edge $e \in E(G)$.
Solution: A Steiner tree solution to (G, R, c^{new}).

Observe that the cost function obtained after applying the local modification in general is not a metric, and $\mathrm{OPT}^{\mathrm{new}}_{\varepsilon,k}$ is assumed to live in the metric closure according to the new cost function.

Input : An instance $(G, R, c, \mathrm{OPT}^{\mathrm{old}}_\varepsilon, e, c^{\mathrm{new}}(e))$ of $\mathrm{STP}^{E+}_\varepsilon$
Output : A Steiner tree T
Transform $\mathrm{OPT}^{\mathrm{old}}_\varepsilon$ to a k-restricted solution $\mathrm{OPT}^{\mathrm{old}}_{\varepsilon,k}$ such that $\mathrm{opt}^{\mathrm{old}}_{\varepsilon,k} \le (1+\varepsilon_k)\mathrm{opt}^{\mathrm{old}}_\varepsilon$ where ε_k tends to 0 for large enough k;
Set $T_1 := \mathrm{OPT}^{\mathrm{old}}_\varepsilon$;
Set $T_2 := \mathrm{GUESS}(k+1)$, with respect to $c^{\mathrm{new}}(e)$;
Set $T = T_i$ with $i = \min \arg_{j\in\{1,2\}}\{c(T_j)\}$.

Algorithm 5: EDGEINCREASE

Theorem 4 *Let* $\left(G, R, c, \mathrm{OPT}^{\mathrm{old}}_\varepsilon, e, c^{\mathrm{new}}(e)\right)$ *be an instance of* $\mathrm{STP}^{E+}_\varepsilon$. *Then, for an arbitrary* $\delta > 0$, EDGEINCREASE *is a* $\left(\frac{7\beta-4+\varepsilon(4\beta-4)}{4\beta-1} + O_{\beta,\varepsilon}(\delta)\right)$*-approximation algorithm for* $\mathrm{STP}^{E+}_\varepsilon$.

Proof Let us introduce the following notation. To emphasize which of the two instances we consider, we write $c^{\mathrm{old}}(e)$ instead of $c(e)$, where e is the edge with increased cost. We assume that $e \in E(\mathrm{OPT}^{\mathrm{old}}_{\varepsilon,k})$, as otherwise T_1 would be good enough already. Therefore the graph $\mathrm{OPT}^{\mathrm{old}}_{\varepsilon,k} - e$ has exactly two connected

components F_1 and F_2. Similar to the previous proof, we define $R_1 := R \cap V(F_1)$ and $R_2 := R \cap V(F_2)$.

In $\text{OPT}^{\text{old}}_{\varepsilon,k}$, let K be the full component that contains e. Let P be a minimum-cost path from R_1 to R_2 within K. Then we set $\kappa := c(P) - c^{\text{old}}(e)$.

In $\text{OPT}^{\text{new}}_{k}$, there is a full component L of cost λ such that $V(L)$ contains nodes from both R_1 and R_2. If L has two edge-disjoint paths between R_1 and R_2, we define $\lambda' = 0$. Otherwise there is an edge $e_L \in E(L)$ such that e_L is a cut edge in $F_1 \cup F_2 \cup L$, and $\lambda' := c(e_L)$. We obtain the following inequalities, where as before $\alpha = c(T)/\text{opt}^{\text{new}}$.

Removing e and adding a shortest path between R_1 and R_2 within L gives a feasible solution to (G, R, c^{new}). Therefore T_2 is good enough unless

$$\text{opt}^{\text{old}}_{\varepsilon,k} - c^{\text{old}}(e) + \lambda/2 + \lambda'/2 \geq \alpha \text{opt}^{\text{new}}. \tag{35}$$

One feasible solution to the original instance is to remove e_L and to add P. Therefore we obtain

$$\text{opt}^{\text{new}}_{k} - \lambda' + c^{\text{old}}(e) + \kappa \geq \text{opt}^{\text{old}}. \tag{36}$$

From (35) and (36), we obtain

$$(1+\varepsilon)(1+\delta)\kappa \geq (\alpha - 1 - \varepsilon)\text{opt}^{\text{new}} + \varepsilon\lambda' + \lambda'/2 - \lambda/2. \tag{37}$$

Since clearly $\text{opt}^{\text{new}} \geq \text{opt}^{\text{old}}$, (35) implies

$$\lambda' \geq 2(\alpha - 1 - \varepsilon)\text{opt}^{\text{new}}_{\varepsilon,k} - \lambda. \tag{38}$$

In (37) and (38), we have used that $c^{\text{old}}(e) \geq 0$ and therefore it can be omitted.

We obtain an additional constraint by observing that in addition to using e_L, within $\text{OPT}^{\text{new}}_{k}$ the required vertices of K have to be connected. Let K_1 be the tree of $K - e$ that contains $R_1 \cap V(K)$. We see K_1 as a rooted tree with the root r_1 contained in $e = \{r_1, r_2\}$. Let us fix any two vertices $u \neq u' \in V(K_1)\backslash\{r\}$, with parents v, v'. Then the minimum distance between the two subtrees rooted at u, u' is at least $\max\{c(u,v), c(u',v')\}$. The same argumentation holds for K_2, which we define analogous to K_1 (it contains $R_2 \cap V(K)$, and has the root r_2). By traversing a path from $V(K_1) \cap R_1$ to r_1 within K_1 and from $V(K_2) \cap R_2$ to r_2, and adding the distances, we conclude that there is a collection of at most k full components in $\text{OPT}^{\text{new}}_{k}$ that without counting e_L have a total cost of at least κ. Therefore, using T_2,

$$\zeta \text{opt}^{\text{new}}_{k} \geq \lambda' + \kappa,$$

which due to (37) implies

$$(1+\varepsilon)(1+\delta)\zeta\text{opt}^{\text{new}}_{k} \geq (\alpha - 1 - \varepsilon)\text{opt}^{\text{new}}_{\varepsilon,k} + ((3+4\varepsilon)/2)\lambda' - \lambda/2,$$

and with (38) we obtain

$$(1+\delta)\zeta\text{opt}^{\text{new}}_{k} \geq 4(\alpha - 1 - \varepsilon)\text{opt}^{\text{new}}_{\varepsilon,k} - 2\lambda. \tag{39}$$

By definition of T_2,

$$\zeta \mathrm{opt}_k^{\mathrm{new}} \geq \lambda. \tag{40}$$

The value ζ is minimized subject to the constraints (39) and (40) if

$$\lambda = 4(\alpha - 1 - \varepsilon)/3.$$

Now the claim follows from (1). □

7 Decreased Edge Cost

We consider the reoptimization variant where the edge cost of one edge is decreased, $\mathrm{STP}_{\varepsilon}^{E-}$. Similar to the previous section, if e is the edge of G with decreased cost, we define $c^{\mathrm{new}} : E(G) \to \mathbb{R}_{\geq 0}$ as $c^{\mathrm{new}}(e') = c(e')$ for all edges $e' \in E(G)\backslash\{e\}$ and $c^{\mathrm{new}}(e)$ is the decreased cost. Then the formal definition of the reoptimization variant is as follows.

Given: A parameter $\varepsilon > 0$, a Steiner tree instance (G, R, c), a solution $\mathrm{OPT}_{\varepsilon}^{\mathrm{old}}$ to (G, R, c) such that $\mathrm{opt}_{\varepsilon}^{\mathrm{old}} \leq (1+\varepsilon)\mathrm{opt}^{\mathrm{old}}$, and a cost $c^{\mathrm{new}}(e) \leq c(e)$ for an edge $e \in E(G)$.
Solution: A Steiner tree solution to (G, R, c^{new}).

We show that decreasing the edge cost does not allow for a robust reoptimization better than STP approximation, similar to the case considered in Theorem 1. The proof of the following theorem is analogous to the proof of Theorem 1.

Theorem 5 *For an arbitrary $\varepsilon > 0$, let $\mathscr{A}$ be a polynomial-time α-approximation algorithm for $\mathrm{STP}_{\varepsilon}^{E-}$. Then there is a polynomial-time α-approximation algorithm for the Steiner tree problem.*

Proof Given a Steiner tree instance (G, R, c), we construct a $\mathrm{STP}_{\varepsilon}^{E-}$ instance $(G', R', c', e, c^{\mathrm{new}}(e))$. We first compute a minimum spanning tree $\tilde{T}$ of $G[R]$. As in the proof of Theorem 1, $G[R]$ is a complete graph since w.l.o.g. we assume c to be metric, and $c(\tilde{T}) \leq 2\mathrm{opt}^{\mathrm{new}}$, as shown by Takahashi and Matsuyama [32]. Again, we assume w.l.o.g. that $\alpha < 2$.

We obtain G' by combining G and a new node t as follows. We set $V(G') := V(G) \cup \{t\}$ and $E(G') = E(G) \cup \{t, t'\}$ for a node $t' \in R$. Then we obtain c' from c by setting $c'(t, t') = c(\tilde{T}) \cdot \max\{1, (1-\varepsilon)/\varepsilon\}$ and forming the metric closure. We set $R' = R \cup \{t\}$ and obtain $\mathrm{OPT}_{\varepsilon}^{\mathrm{old}}$ by adding $\{t, t'\}$ to $\tilde{T}$. Finally, we obtain the Steiner tree T by applying $\mathscr{A}$ to $(G', R', c', \{t, t'\}, 0)$.

Observe that reducing the cost in this case does not influence any edge costs in G' except for those edges incident to t. Therefore the new instance is equivalent to (G, R).

To finish the proof, we have to show that $\text{OPT}_\varepsilon^{\text{old}}$ was a valid solution given to $\mathscr{A}$, i.e., $\text{opt}_\varepsilon^{\text{old}}$ is at most a factor $(1+\varepsilon)$ larger than optimum. This step is identical with the corresponding step in the proof of Theorem 1. □

8 Changing the Node Set

Finally, we consider the local modifications where we remove nodes from the graph or add nodes to the graph. The results of this section are much simpler than the results of the previous sections and we sketch them for completeness, without formal definitions of the considered reoptimization problems.

Removing a required node or a Steiner node gives a reoptimization problem that is as hard as the original Steiner tree problem [17]. The idea is that if (G, R, c) is a Steiner tree instance that we want to solve, then we add a node t that has cost zero edges to all nodes in G, and therefore there is a trivial optimal Steiner tree of cost zero. Removing t, however, leaves us with the original problem instance; in this case the hardness does not even depend on the robustness, the problem is hard even if we are given an optimal solution.

Adding a node is at least as hard as decreasing the edge cost. If a Steiner node t is added, this is straightforward: if we want to decrease the cost of $e = \{u, v\}$ in a Steiner tree instance (G, R, c), we add a Steiner node v' with $c(v, v') = 0$, $c(u, v')$ the reduced cost, and $c(v', w) = c(v, w)$ for all other $w \in V(G)$.

If a required node t is added, we have to consider an additional step. Observe that decreasing the cost of $e = \{u, v\}$ can only have an impact, if it is contained in all optimal solutions to the modified problem, as otherwise keeping the given solution is close to optimal. Therefore we can solve $\text{STP}_\varepsilon^{E-}$ by taking the best of the following two solutions: we either keep the given old optimal solution or we reduces the edge cost analogous to the case where t is a Steiner node.

Acknowledgements Open Access funding provided by Projekt DEAL. We would like to thank Anna Zych for some helpful comments and the anonymous reviewers for helping to improve the presentation.

Funding Research partially funded by Deutsche Forschungsgemeinschaft Grants BL511/10-1 and MO2889/1-1, and by the Indo-German Max Planck Center for Computer Science (IMPECS). The results were obtained when Keshav Goyal was affiliated with IIT Delhi, India.

References

1. Archetti, C., Bertazzi, L., Speranza, M.G.: Reoptimizing the traveling salesman problem. Networks **42**(3), 154–159 (2003)
2. Archetti, C., Bertazzi, L., Speranza, M.G.: Reoptimizing the 0–1 knapsack problem. Discrete Appl. Math. **158**(17), 1879–1887 (2010)
3. Archetti, C., Guastaroba, G., Speranza, M.G.: Reoptimizing the rural postman problem. Comput. Oper. Res. **40**(5), 1306–1313 (2013)
4. Ausiello, G., Bonifaci, V., Escoffier, B.: Complexity and Approximation in Reoptimization. Imperial College Press/World Scientific, London (2011)
5. Ausiello, G., Escoffier, B., Monnot, J., Paschos, V.T.: Reoptimization of minimum and maximum traveling salesman's tours. J. Discrete Algorithms **7**(4), 453–463 (2009)
6. Bender, M.A., Farach-Colton, M., Fekete, S.P., Fineman, J.T., Gilbert, S.: Reallocation problems in scheduling. In: Proceedings of the 25th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2013), pp. 271–279. ACM (2013)
7. Berg, T., Hempel, H.: Reoptimization of traveling salesperson problems: changing single edge-weights. In: Proceedings of the Third International Conference on Language and Automata Theory and Applications (LATA 2009), LNCS, vol. 5457, pp. 141–151. Springer (2009)
8. Bern, M.W., Plassmann, P.E.: The Steiner problem with edge lengths 1 and 2. Inf. Process. Lett. **32**(4), 171–176 (1989)
9. Bilò, D.: New algorithms for Steiner tree reoptimization. In: ICALP, LIPIcs, vol. 107, pp. 19:1–19:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018)
10. Bilò, D., Böckenhauer, H., Komm, D., Královic, R., Mömke, T., Seibert, S., Zych, A.: Reoptimization of the shortest common superstring problem. Algorithmica **61**(2), 227–251 (2011)
11. Bilò, D., Böckenhauer, H.J., Hromkovič, J., Královič, R., Mömke, T., Widmayer, P., Zych, A.: Reoptimization of Steiner trees. In: Proceedings of the 11th Scandinavian Workshop on Algorithm Theory (SWAT 2008), LNCS, vol. 5124, pp. 258–269 (2008)
12. Bilò, D., Widmayer, P., Zych, A.: Reoptimization of weighted graph and covering problems. In: Proceedings of the 6th International Workshop on Approximation and Online Algorithms (WAOA 2008), Lecture Notes in Computer Science, vol. 5426, pp. 201–213. Springer (2008)
13. Bilò, D., Zych, A.: New reoptimization techniques employed to Steiner tree problem. In: Proceedings of the 6th Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS 11) (2011)
14. Bilò, D., Zych, A.: New advances in reoptimizing the minimum Steiner tree problem. In: Mathematical Foundations of Computer Science 2012, pp. 184–197. Springer (2012)
15. Böckenhauer, H., Komm, D.: Reoptimization of the metric deadline TSP. J. Discrete Algorithms **8**(1), 87–100 (2010)
16. Böckenhauer, H.J., Forlizzi, L., Hromkovič, J., Kneis, J., Kupke, J., Proietti, G., Widmayer, P.: On the approximability of TSP on local modifications of optimally solved instances. Algorithmic Oper. Res. **2**(2), 83–93 (2007)
17. Böckenhauer, H.J., Freiermuth, K., Hromkovič, J., Mömke, T., Sprock, A., Steffen, B.: The Steiner tree reoptimization problem in graphs with sharpened triangle inequality. J. Discrete Algorithms **11**, 73–86 (2012)
18. Böckenhauer, H.J., Hromkovič, J., Královič, R., Mömke, T., Rossmanith, P.: Reoptimization of Steiner trees: changing the terminal set. Theor. Comput. Sci. **410**(36), 3428–3435 (2009)
19. Böckenhauer, H.J., Hromkovič, J., Mömke, T., Widmayer, P.: On the hardness of reoptimization. In: Proceedings of the 34th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2008), LNCS, vol. 4910, pp. 50–65 (2008)
20. Borchers, A., Du, D.Z.: The k-Steiner ratio in graphs. SIAM J. Comput. **26**(3), 857–869 (1997)
21. Boria, N., Della Croce, F.: Reoptimization in machine scheduling. Theor. Comput. Sci. **540**, 13–26 (2014)
22. Boria, N., Monnot, J., Paschos, V.T.: Reoptimization of maximum weight induced hereditary subgraph problems. Theor. Comput. Sci. **514**, 61–74 (2013)
23. Boria, N., Paschos, V.T.: Fast reoptimization for the minimum spanning tree problem. J. Discrete Algorithms **8**(3), 296–310 (2010)
24. Boria, N., Paschos, V.T.: A survey on combinatorial optimization in dynamic environments. RAIRO-Oper. Res. **45**(03), 241–294 (2011)

25. Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L.: Steiner tree approximation via iterative randomized rounding. J. ACM **60**(1), 6 (2013)
26. Dai, W.: Reoptimization of minimum latency problem. In: COCOON, Lecture Notes in Computer Science, vol. 10392, pp. 162–174. Springer (2017)
27. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks **1**, 195–207 (1971)
28. Escoffier, B., Milanič, M., Paschos, V.T.: Simple and fast reoptimizations for the Steiner tree problem. Algorithmic Oper. Res. **4**(2), 86–94 (2009)
29. Hougardy, S., Silvanus, J., Vygen, J.: Dijkstra meets Steiner: a fast exact goal-oriented Steiner tree algorithm. ArXiv preprint arXiv:1406.0492 (2014)
30. Monnot, J.: A note on the traveling salesman reoptimization problem under vertex insertion. Inf. Process. Lett. **115**(3), 435–438 (2015)
31. Schäffter, M.W.: Scheduling with forbidden sets. Discrete Appl. Math. **72**(1), 155–166 (1997)
32. Takahashi, H., Matsuyama, A.: An approximate solution for the Steiner problem in graphs. Math. Japonica **24**(6), 573–577 (1979/1980)
33. Zych, A.: Reoptimization of NP-Hard Problems. Ph.D. Thesis, Dissertations, Eidgenössische Technische Hochschule ETH Zürich, Nr. 20257, 2012 (2012)