



New Algorithms for Steiner Tree Reoptimization

Davide Bilò¹

Received: 23 February 2022 / Accepted: 14 May 2024 / Published online: 29 May 2024
© The Author(s) 2024

Abstract

Reoptimization is a setting in which we are given a good approximate solution of an optimization problem instance and a local modification that slightly changes the instance. The main goal is that of finding a good approximate solution of the modified instance. We investigate one of the most studied scenarios in reoptimization known as *Steiner tree reoptimization*. Steiner tree reoptimization is a collection of strongly NP-hard optimization problems that are defined on top of the classical Steiner tree problem and for which several constant-factor approximation algorithms have been designed in the last decades. In this paper we improve upon all these results by developing a novel technique that allows us to design *polynomial-time approximation schemes*. Remarkably, prior to this paper, no approximation algorithm better than recomputing a solution from scratch was known for the elusive scenario in which the cost of a single edge decreases. Our results are best possible since none of the problems addressed in this paper admits a fully polynomial-time approximation scheme, unless $P = NP$

Keywords Reoptimization · Steiner tree · Approximation algorithms · Polynomial-time approximation scheme

1 Introduction

Reoptimization is a setting in which we are given an instance I of an optimization problem paired with an either optimal or approximate solution S for I and a local modification that, once applied to I , generates a new instance I' that slightly differs from I . The main goal is that of finding a good approximate solution of I' . The idea beyond reoptimization is to exploit the structural properties of S when computing a good solution for I' . This approach leads to the design of reoptimization algorithms that, when compared to classical algorithms that compute feasible solutions from

A preliminary version of this work appeared in the proceeding of the 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).

✉ Davide Bilò
davide.bilo@univaq.it

¹ Department of Information Engineering Computer Science, and Mathematics, University of L'Aquila, Via Vetoio 0 Coppito, 67100 L'Aquila, AQ, Italy

scratch by ignoring both S and I , are much more efficient in terms of running time and/or output solutions that guarantee a better approximation ratio. Thus, reoptimization finds applications in many settings, like scheduling problems or network design problems, where prior knowledge is often at our disposal and a problem instance can arise from a local modification of a previously solved problem instance (for example, a scheduled job is canceled or some links of the network fail).

The term reoptimization was mentioned for the first time in the paper of Schäffter [35], where the author addressed an NP-hard scheduling problem with forbidden sets in the scenario of adding/removing a forbidden set. Since then, reoptimization has been successfully applied to other NP-hard problems including¹: the Steiner tree problem [8, 11, 13–15, 31, 32, 37], the traveling salesman problem and some of its variants [1, 5, 7, 12, 15–19, 34], scheduling problems [6, 21], the minimum latency problem [28, 29], the minimum spanning tree problem [24], the rural postman problem [3], many covering problems [10], the shortest superstring problem [9], the knapsack problem [2], the maximum-weight induced hereditary subgraph problems [22], and the maximum P_k subgraph problem [23]. Some overviews on other NP-hard reoptimization problems can be found in [4, 25, 36].

1.1 Our Results

In this paper we address *Steiner tree reoptimization*. Steiner tree reoptimization is a collection of optimization problems that are built on top of the famous *Steiner tree problem* (*STP* for short). An *STP* instance consists of an n -vertex, connected undirected graph G with non-negative edge costs, where each vertex can be either *terminal* or *Steiner*. The goal in *STP* is to find a minimum-cost tree contained in G that spans all the terminals. It is well-known that *STP* is not approximable within a factor better than $96/95$, unless $P = NP$ [27], but can be approximated, in polynomial time, within a factor of $\sigma = \ln 4 + \varepsilon \approx 1.387$, for every constant $\varepsilon > 0$ [26].

In the Steiner tree reoptimization problems we address in this paper we modify the instance by either changing the cost of an edge (the edge cost can either increase or decrease) or by changing the status of a vertex, from Steiner to terminal or from terminal to Steiner. We observe that the edge-cost modifications also capture the scenarios in which we either add an edge to the graph or we delete an edge from the graph. In fact, the addition of an edge can be modeled by changing the cost of the edge from a very large value to the desired value, while the deletion of an edge can be modeled by changing the cost of the edge to a very large value.

All the Steiner tree reoptimization problems addressed in this paper are strongly NP-hard [14, 15], which implies that none of them admits a fully polynomial-time approximation scheme, unless $P = NP$. In this paper we improve upon all the constant-factor approximation algorithms that have been developed in the last decades (see [8, 11, 13–15, 31, 32, 37]) by designing *polynomial-time approximation schemes*. More precisely, for a given ρ -approximate solution S of the instance I and for every constant $\varepsilon > 0$, all our algorithms compute a $(\rho + \varepsilon)$ -approximate solution of I' in polynomial time. For the scenarios in which the cost of an edge decreases or a terminal vertex

¹ In this paper we focus only on reoptimization problems that are NP-hard.

Table 1 The state of the art of the approximability of Steiner tree reoptimization before and after our paper

Local modification	Before this paper	This paper
A terminal becomes a Steiner vertex*	$\frac{10\sigma-7}{7\sigma-4} + \varepsilon$ [32]	$1 + \varepsilon$
A Steiner vertex becomes terminal	$\frac{10\sigma-7}{7\sigma-4} + \varepsilon$ [32]	$1 + \varepsilon$
An edge cost increases	$\frac{7\sigma-4}{4\sigma-1} + \varepsilon$ [32]	$1 + \varepsilon$
An edge cost decreases*	$\frac{5\sigma-3}{3\sigma-1}$ [8]	$1 + \varepsilon$
A vertex is added to the graph*	?	?
A vertex is deleted from the graph	As hard as <i>STP</i> [32]	As hard as <i>STP</i> [32]

The value σ denotes the best approximation ratio we can achieve for *STP* with a polynomial time algorithm. For the sake of readability, all the bounds are provided for the case in which $\rho = 1$. The bound of $\frac{5\sigma-3}{3\sigma-1}$ for the scenario in which the cost of an edge decreases holds when the modified edge can affect only its cost in the shortest-path metric closure of the input graph. Local modifications marked with an asterisk are those for which the condition $\rho = 1$ is necessary as otherwise the reoptimization problem would be as hard to approximate as the *STP*. Question marks represent open problems. As $\sigma \leq \ln 4 + \varepsilon \approx 1.387$ [26], we obtain $\frac{10\sigma-7}{7\sigma-4} \approx 1.204$, $\frac{7\sigma-4}{4\sigma-1} \approx 1.256$, and $\frac{5\sigma-3}{3\sigma-1} \approx 1.246$

becomes Steiner, our polynomial-time approximation schemes hold under the assumption that S is an optimal solution of I , i.e., $\rho = 1$. We observe that this assumption is somehow necessary since Goyal and Mömke [32] proved that in both scenarios, unless $\rho = 1$, any σ -approximation algorithm for the reoptimization problem with polynomial running time could be used to design a polynomial-time σ -approximation algorithm for the *STP*. Remarkably, prior to this paper, no approximation algorithm better than recomputing a $(\ln 4 + \varepsilon)$ -approximate solution from scratch using the algorithm in [26] was known for the elusive scenario in which the cost of a single edge decreases.

The current state of the art of Steiner tree reoptimization is summarized in Table 1. We observe that the only problem that remains open is the design of a reoptimization algorithm for the scenario in which a vertex, that can be either terminal or Steiner, is added to the graph. In fact, as proved by Goyal and Mömke [32], the scenario in which a vertex is removed from the graph is as hard to approximate as the *STP*.²

1.2 Used Techniques

All the polynomial-time approximation schemes proposed in this paper make a clever use of the following three algorithms:

- the algorithm **CONNECT** that takes as inputs an *STP* instance I and a Steiner forest of I . The algorithm augments the Steiner forest with a minimum-cost set of edges to obtain a Steiner tree of I . This algorithm has been introduced by Böckenhauer et al. [14] and has been subsequently used in several other papers on Steiner tree reoptimization [11, 13, 32, 37].

² Think of any *STP* instance I' and augment it with a vertex v that is connected to each terminal by an edge of cost 0 to generate the instance I . An optimal solution S of cost 0 for I is the star centered at v that spans all the terminals. The new instance I' is obtained by removing v from the graph.

- the algorithmic proof of Borchers and Du [20] that, for every $\xi > 0$, converts a Steiner tree S into an k -restricted Steiner tree S_ξ , with $k = 2^{\lceil 1/\xi \rceil}$, such that the cost of S_ξ is at most a $(1 + \xi)$ -factor away from the cost of S . This algorithm has been used for the first time in Steiner tree reoptimization by Goyal and Mömke [32];
- a novel algorithm, that we call BUILDST, that takes a k -restricted Steiner forest S_ξ of S with q trees and a positive integer h as inputs and computes a minimum-cost Steiner tree S' among those solutions that can be obtained by swapping all the edges contained in up to h full components of S_ξ with a minimum-cost set of edges. The running time of this algorithm is polynomial when all the three parameters k , q , and h are constant.

The algorithm BUILDST computes several feasible solutions S_1, \dots, S_ℓ for I' and returns the cheapest of them. We prove that the average cost of the computed feasible solutions is at most a $\rho + \varepsilon$ factor away from the cost of an optimal solution for I' . This clearly implies that the returned solution is $(\rho + \varepsilon)$ -approximate. Intuitively, each S_i is obtained by swapping up to h suitable full components, say H_i , of a suitable k -restricted version of a forest in S , say S_ξ , with a minimal set of full components, say H'_i , of a k' -restricted version of a fixed optimal solution OPT' . This implies that we can easily bound the cost of each S_i by a function that depends on the cost of S_ξ , the cost of H_i , and the cost of H'_i . However, to prove that the computed solution is indeed a $(\rho + \varepsilon)$ -approximation of OPT' , we need to define H_{i+1} as a function of both OPT' and H'_i , and H'_i as a function of both S_ξ and H_i . In this way, we obtain different upper bounds on the cost of the computed solutions for which we can easily compute the average.

The paper is organized as follows: in Sect. 2 we provide the basic definitions and some preliminaries; in Sect. 3 we describe the three main tools that are used in all our algorithms; in Sects. 4, 5, 6, and 7 we describe and analyze the algorithms for all the four local modifications addressed in this paper. Section 8 concludes the paper with a list of open problems in the field.

2 Basic Definitions and Preliminaries

The *Steiner tree problem* (STP for short) is defined as follows: The input is a triple $I = \langle G, c, R \rangle$. $G = (V(G), E(G))$ is an n -vertex connected undirected graph, where $V(G)$ and $E(G)$ denote the set of vertices and the set of edges, respectively, c is a function that associates a real value $c(e) \geq 0$ to each edge $e \in E(G)$, and $R \subseteq V(G)$ is the set of *terminal* vertices. The problem asks to compute a minimum-cost Steiner tree of I , i.e., a tree contained in G that spans R and that minimizes the overall sum of its edge costs.

In *Steiner tree reoptimization* we are given a triple $\langle I, S, I' \rangle$, where I is an STP instance, S is a ρ -approximate Steiner tree of I , and I' is another STP instance which slightly differs from I . The problem asks to find a ρ -approximate Steiner tree of I' .

The vertices in $V(G) \setminus R$ are also called *Steiner* vertices. With a slight abuse of notation, for any (not necessarily proper) subgraph H of G , we denote by $c(H) := \sum_{e \in E(H)} c(e)$ the cost of H .

For a forest F of G and a set of edges $E' \subseteq E(G)$, we denote by $F + E'$ a forest yielded by the addition of all the edges in $E' \setminus E(F)$ to F , i.e., we scan all the edges of E' one by one in any fixed order and we add the currently scanned edge to the forest only if the resulting graph remains a forest. Analogously, we denote by $F - E'$ the forest yielded by the removal of all the edges in $E' \cap E(F)$ from F . We use the shortcuts $F + e$ and $F - e$ to denote $F + \{e\}$ and $F - \{e\}$, respectively. Furthermore, if F' is another forest of G we also use the shortcuts $F + F'$ and $F - F'$ to denote $F + E(F')$ and $F - E(F')$, respectively. For a set of vertices $U \subseteq V(G)$, we define $F + U := (V(F) \cup U, E(F))$ and use the shortcut $F + v$ to denote $F + \{v\}$.

W.l.o.g., in this paper we tacitly assume that all the leaves of the Steiner trees we compute are terminals. Indeed, if a Steiner tree S contains a leaf which is not a terminal, then we can always remove such a leaf from S to obtain another Steiner tree S' such that $c(S') \leq c(S)$. Analogously, we tacitly assume that all the leaves of a *Steiner forest*, i.e., a forest spanning all the terminals, are terminals. A *full component* of a Steiner forest F – so a Steiner tree as well – is a maximal (w.r.t. edge insertion) subtree of F whose leaves are all terminals and whose internal vertices are all Steiner vertices. We observe that each tree of a Steiner forest can always be decomposed into full components. A *k-restricted* Steiner forest is a Steiner forest where each of its full components has at most k terminals.

In the rest of the paper, for any superscript s (the empty string is included), we denote by I^s an *STP* instance (we tacitly assume that $I^s = \langle G^s, c^s, R^s \rangle$ as well as that G^s has n vertices), by OPT^s a fixed optimal solution of I^s , by d^s the shortest-path metric induced by c^s in G^s (i.e. $d^s(u, v)$ is equal to the cost of a shortest path between u and v in G^s w.r.t. cost function c^s), by K^s the complete graph on $V(G^s)$ with edge costs d^s , and by K_n^s the complete graph consisting of n copies of K^s , where the cost of an edge between a vertex and any of its copies is equal to 0, while the cost of an edge (u, v) , with u not being a copy of v , is $d^s(u, v)$. With a little abuse of notation, we use d^s to denote the cost function of K_n^s . Moreover, we say that a subgraph H of K_n^s *spans* a vertex $v \in V(G^s)$ if H spans – according to the standard terminology used in graph theory – any of the n copies of v . Finally, with a little abuse of notation, we denote by I_n^s the *STP* instance $I_n^s = \langle K_n^s, d^s, R^s \rangle$.

Let I be an *STP* instance. We observe that any forest F of G is also a forest of K such that $d(F) \leq c(F)$. Moreover, any forest \bar{F} of K can be transformed in polynomial time into a forest F of G spanning $V(\bar{F})$ and such that $c(F) \leq d(\bar{F})$. In fact, F can be build from an empty graph on $V(G)$ by iteratively adding – according to our definition of graph addition – a shortest path between u and v in G , for every edge $(u, v) \in E(\bar{F})$.

We also observe that any forest F of K can be viewed as a forest of K_n . Conversely, any forest F of K_n can be transformed in polynomial time into a forest of K spanning $V(F)$ and having the same cost of F : it suffices to identify each of the 0-cost edges of F between any vertex and any of its copies. Therefore, we have polynomial-time algorithms that convert any forest F of any graph in $\{G, K, K_n\}$ into a forest F' of any graph in $\{G, K, K_n\}$ such that F' always spans $V(F)$ and the cost of F' (according to the cost function associated with the corresponding graph) is less than or equal to the cost of F (according to the cost function associated with the corresponding graph). All these polynomial-time algorithms will be implicitly used in the rest of the paper; for example, we can say that a Steiner tree of I_n is also a Steiner tree of I or that a

forest of I_n that spans v is also a forest of I that spans v . However, we observe that some of these polynomial-time algorithms do not preserve useful structural properties; for example, when transforming a k -restricted Steiner tree of I_n into a Steiner tree of I we may lose the property that the resulting tree is k -restricted. Therefore, whenever we refer to a structural property of a forest, we assume that such a property holds only w.r.t. the underlying graph the forest belongs to.

3 General Tools

In this section we describe the three main tools that are used by all our algorithms.

The first of them is the algorithm `CONNECT` that has been introduced by Böckenhauer *et al.* in [14]. This algorithm takes an *STP* instance I together with a Steiner forest F of I as inputs and computes a minimum-cost set of edges of G whose addition to F yields a Steiner tree of I . The algorithm `CONNECT` transforms I into a smaller *STP* instance, where each terminal vertex in the reduced instance corresponds to a tree of F . The algorithm then uses the famous Dreyfus-Wagner algorithm in [30] to find an optimal solution of the reduced instance.³ The running time of the algorithm `CONNECT` is $O^*(3^q)$, where q is the number of terminals in the reduced instance.⁴ We observe that the algorithm `CONNECT` has polynomial running time when q is polylogarithmic in n . The call of the algorithm `CONNECT` with input parameters I and F is denoted by `CONNECT`(I, F).

The second tool is the algorithmic proof of Borchers and Du on k -restricted Steiner trees [20] that has already been used in Steiner tree reoptimization by Goyal and Mömke [32]. In their paper, Borchers and Du proved that, for every *STP* instance I and every $\xi > 0$, there exists a k -restricted Steiner tree S_ξ of I_n , with $k = 2^{\lceil 1/\xi \rceil}$, such that $d(S_\xi) \leq (1 + \xi)c(OPT)$. Furthermore, S_ξ can be constructed in polynomial time if we know OPT . As the following theorem shows, the algorithmic proof of Borchers and Du immediately extends to any (not necessarily optimal) Steiner tree of I_n .

Theorem 1 [20] *Let I be an *STP* instance, S a Steiner tree of I , and $\xi > 0$ a constant value. There is a polynomial time algorithm that computes a k -restricted Steiner tree S_ξ of I_n , with $k = 2^{\lceil 1/\xi \rceil}$, such that $d(S_\xi) \leq (1 + \xi)c(S)$.*

Proof Consider the *STP* instance $I' = \langle G', d', R' \rangle$ in which G' is the complete graph on $V(S)$, d' is the shortest path metric induced by c in S , and $R' = R$. By construction, $d'(S) = c(S)$. We claim that S is an optimal solution for I' . Before proving this fact, we show its implications. As shown by Borchers and Du [20], we can compute a k -restricted Steiner tree S_ξ of I' , with $k = 2^{\lceil 1/\xi \rceil}$, such that $d'(S_\xi) \leq (1 + \xi)d'(S) = (1 + \xi)c(S)$ in polynomial time. As S is a subgraph of G , for any two vertices $u, v \in V(S)$, we have $d(u, v) \leq d'(u, v)$. Therefore, $d(S_\xi) \leq d'(S_\xi)$. Hence, $d(S_\xi) \leq (1 + \xi)c(S)$.

Therefore, to prove the theorem statement, it remains to show that S is an optimal solution for I' . Let S' be a Steiner tree for I' . For any edge e' of S' , let $P_{e'}$ be the unique path in S between the endvertices of e' . By construction, we have that $d'(e') = d'(P_{e'})$.

³ We refer to Hougardy [33] for further exact algorithms for the *STP*.

⁴ The O^* notation hides polynomial factors.

Therefore, to show that $d'(S) \leq d'(S')$, it is enough to prove that any edge e of S is also an edge of $P_{e'}$, for some edge e' of S' . To see this, consider the forest $S - e$ containing two trees, say T_1 and T_2 , each of which spans some terminal vertices of R . As S' also spans R , there is an edge e' of S' that has one endvertex in T_1 and the other endvertex in T_2 . As a consequence, e is an edge of $P_{e'}$. \square

The call of Borchers and Du algorithm with input parameters I , S , and ξ , is denoted by $\text{RESTRICTEDST}(I, S, \xi)$. By construction, the algorithm of Borchers and Du also guarantees the following properties:

- (a) if v is a Steiner vertex that has degree at least 3 in S , then S_ξ spans v ;
- (b) if the degree of a terminal t in S is 2, then the degree of t in S_ξ is at most 4.

As shown in the following corollary, these two properties are useful if we need that a specific vertex of S is also contained in the k -restricted Steiner tree S_ξ of I_n .

Corollary 1 *Let I be an STP instance, S a Steiner tree of I , v a vertex of S , and $\xi > 0$ a constant value. There is a polynomial time algorithm that computes a k -restricted Steiner tree S_ξ of I_n , with $k = 2^{2+\lceil 1/\xi \rceil}$, such that $d(S_\xi) \leq (1 + \xi)c(S)$ and $v \in V(S_\xi)$.*

Proof The cases in which $v \in R$ immediately follows from Theorem 1. Moreover, the claim trivially holds by property (a) when v is a Steiner vertex of degree greater than or equal to 3 in S . Therefore we assume that v is a Steiner vertex of degree 2 in S . Let $I' = \langle G, c, R \cup \{v\} \rangle$. In this case, the call of $\text{RESTRICTEDST}(I', S, \xi)$ outputs a k' -restricted Steiner tree S_ξ of I'_n , with $k' = 2^{\lceil 1/\xi \rceil}$, such that $d(S_\xi) \leq (1 + \xi)c(S)$. Since v is a terminal in I' , by property (b) the degree of v in S_ξ is at most 4; in other words, v is contained in at most 4 full components of S_ξ . Therefore, S_ξ is a k -restricted Steiner tree of I , with $k = 4k' = 2^{2+\lceil 1/\xi \rceil}$. \square

The call of Borchers and Du algorithm with parameters I , S , ξ , and v , where we want to force vertex v to be in the restricted Steiner tree that is computed by the algorithm, is denoted by $\text{RESTRICTEDST}(I, S, \xi, v)$.

The third and last tool, which is the novel idea of this paper, is the algorithm BUILDST (see Algorithm 1 for the pseudocode). The algorithm BUILDST takes as inputs an STP instance I , a Steiner forest F of I_n , and a positive integer value h . The algorithm computes a Steiner tree of I by selecting, among several feasible solutions, the one of minimum cost. More precisely, the algorithm computes a feasible solution for each forest \bar{F} that is obtained by removing from F all the edges of h full components of F (when F contains fewer than h full components the algorithm computes an optimal solution from scratch). The Steiner tree of I associated with \bar{F} is obtained by the call of $\text{CONNECT}(I_n, \bar{F})$. If F contains at most h full components, then the algorithm computes a minimum-cost Steiner tree of I from scratch. The call of the algorithm BUILDST with input parameters I , F , and h is denoted by $\text{BUILDST}(I, F, h)$. The proof of the following lemma is immediate.

Lemma 1 *Let I be an STP instance, F a Steiner forest of I_n , \bar{F} a Steiner forest of I_n that is obtained from F by removing up to h of its full components, and H a minimum-cost subgraph of K_n such that $\bar{F} + H$ is a Steiner tree of I . Then the cost of the Steiner tree of I returned by the call of $\text{BUILDST}(I, F, h)$ is at most $c(\bar{F}) + c(H)$.*

Algorithm 1: The pseudocode of the algorithm BUILDST(I, F, h).

```

1 if  $F$  contains less than  $h$  full components then return CONNECT( $I, (R, \emptyset)$ );
2  $S' \leftarrow \perp$ ;
3 for every set  $H$  of  $h$  full components of  $F$  do
4    $\bar{F} \leftarrow F - H$ ;
5    $\bar{S} \leftarrow$  CONNECT( $I_n, \bar{F}$ );
6   if  $S' = \perp$  or  $c(\bar{S}) < c(S')$  then  $S' \leftarrow \bar{S}$ ;
7 return  $S'$ ;

```

Moreover, the following lemmas hold.

Lemma 2 *Let I be an STP instance and F a forest of G spanning R . If F contains q trees, then every minimal (w.r.t. the deletion of entire full components) subgraph H of any graph in $\{G, K, K_n\}$ such that $F + H$ is a Steiner tree of I contains at most $q - 1$ full components.*

Proof Since H is minimal, each full component of H contains at least 2 vertices each of which respectively belongs to one of two distinct trees of F . Therefore, if we add the full components of H to F one after the other, the number of connected components of the resulting graph strictly decreases at each iteration because of the minimality of H . Hence, H contains at most $q - 1$ full components. \square

Lemma 3 *Let I be an STP instance and F a Steiner forest of I_n of q trees. If each tree of F is k -restricted, then the call of BUILDST(I, F, h) outputs a Steiner tree of I in time $O^*(|R|^{h3^{q+hk}})$.*

Proof If F contains at most h full components, then the number of terminals is at most $q + hk$ and the call of CONNECT($I, (R, \emptyset)$) outputs a Steiner tree of I in time $O^*(3^{q+hk})$. Therefore, we assume that F contains a number of full components that is greater than or equal to h . In this case, the algorithm BUILDST(I, F, h) computes a feasible solution for each forest that is obtained by removing up to h full components of F from itself. As the number of full components of F is at most $|R|$, the overall number of feasible solutions evaluated by the algorithm is $O(h|R|^h)$. Let \bar{F} be any forest that is obtained from F by removing h of its full components. Since F is k -restricted, \bar{F} contains at most $q + hk$ trees. Therefore the call of CONNECT(I_n, \bar{F}) requires $O^*(3^{q+hk})$ time to output a solution. Hence, the overall time needed by the call of BUILDST(I, F, h) to output a solution is $O^*(|R|^{h3^{q+hk}})$. \square

For the rest of the paper, we denote by $\langle I, S, I' \rangle$ an instance of the Steiner tree reoptimization problem, where $I = \langle G, c, R \rangle$ and $I' = \langle G', c', R' \rangle$. Furthermore, we also assume that $\rho \leq 2$ as well as $\varepsilon \leq 1$, as otherwise we can use classical time-efficient algorithms to compute a 2-approximate solution of I' .

4 A Steiner Vertex Becomes a Terminal

In this section we consider the local modification in which a Steiner vertex $t \in V(G) \setminus R$ becomes a terminal. Clearly, $G' = G$, $c' = c$, $d' = d$, whereas $R' = R \cup \{t\}$. Therefore, for the sake of readability, we will drop the superscripts from G' , c' , and d' .

Let $\xi = \varepsilon/10$ and $h = 2^{2^{\lceil 2/\varepsilon \rceil \lceil 1/\xi \rceil}}$. Algorithm 2 computes a Steiner tree of I' first by calling $\text{RESTRICTEDST}(I, S, \xi)$ to obtain a k -restricted Steiner tree S_ξ of I_n , with $k = 2^{\lceil 1/\xi \rceil}$, and then by calling $\text{BUILDST}(I', S_\xi + t, h)$. Intuitively, the algorithm *guesses* the full components of S_ξ to be replaced with the corresponding full components of a suitable k' -restricted version of a new (fixed) optimal solution, for suitable values of k' .

Algorithm 2: A Steiner vertex $t \in V(G) \setminus R$ becomes a terminal.

```

1  $\xi \leftarrow \varepsilon/10$ ;
2  $h \leftarrow 2^{2^{\lceil 2/\varepsilon \rceil \lceil 1/\xi \rceil}}$ ;
3  $S_\xi \leftarrow \text{RESTRICTEDST}(I, S, \xi)$ ;
4  $S' \leftarrow \text{BUILDST}(I', S_\xi + t, h)$ ;
5 return  $S'$ ;

```

Theorem 2 *Let $\langle I, S, I' \rangle$ be an instance of Steiner tree reoptimization, where S is a ρ -approximate solution of I and I' is obtained from I by changing the status of a single vertex from Steiner to terminal. Then Algorithm 2 computes a $(\rho + \varepsilon)$ -approximate Steiner tree of I' in polynomial time.*

Proof Theorem 1 implies that the computation of S_ξ requires polynomial time. Since S_ξ is a Steiner tree of I_n , $S_\xi + t$ is a Steiner forest of I'_n of at most two trees (observe that S_ξ may already contain t). Therefore, using Lemma 3 and the fact that h is a constant value, the call of $\text{BUILDST}(I', S_\xi + t, h)$ outputs a solution S' in polynomial time. Hence, the overall time complexity of Algorithm 2 is polynomial.

In the following we show that Algorithm 2 returns a $(\rho + \varepsilon)$ -approximate solution. Let OPT' be a fixed optimal solution of I' . Let S'_ξ denote the Steiner tree of I'_n that is returned by the call of $\text{RESTRICTEDST}(I', \text{OPT}', \xi)$ and let H'_0 be any fixed full component of S'_ξ that spans t (ties are broken arbitrarily). For every $i = 1, \dots, \ell$, with $\ell = \lceil 2/\varepsilon \rceil$, we define:

- H_i as the forest consisting of a minimal set of full components of S_ξ whose addition to $S'_\xi - H'_{i-1}$ yields a Steiner tree of I_n (see Fig. 1 (b) and (d));
- H'_i as the forest consisting of a minimal set of full components of S'_ξ whose addition to $S_\xi - H_i$ yields a Steiner tree of I'_n (see Fig. 1 (c)).

For the rest of the proof, we denote by S_i the Steiner tree of I' yielded by the addition of H'_i to $S_\xi - H_i$.

Let $k = 2^{\lceil 1/\xi \rceil}$ and observe that both S_ξ and S'_ξ are k -restricted Steiner trees of I_n and I'_n , respectively (see Theorem 1). Therefore H'_0 spans at most k terminals. Furthermore, by repeatedly using Lemma 2, H_i contains at most k^{2i-1} full components and spans

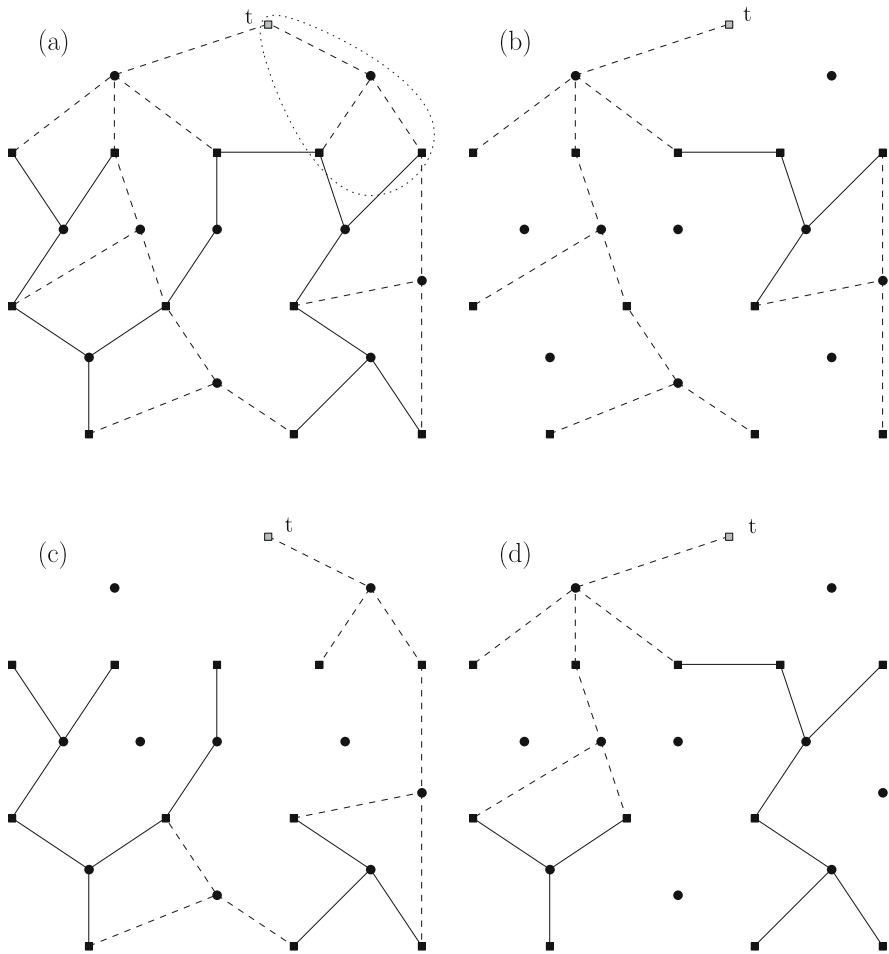


Fig. 1 An example that shows how H_i and H'_i are defined. In (a) it is depicted a Steiner tree reoptimization instance where square vertices represent the terminals, while circle vertices represent the Steiner vertices. The Steiner vertex t of I that becomes a terminal in I' is represented as a terminal vertex of grey color. Solid edges are the edges of S_ξ , while dashed edges are the edges of S'_ξ . The full component H'_0 of S'_ξ is highlighted in (a). The forests $S'_\xi - H'_0$ and H_1 are shown in (b). The forests $S_\xi - H_1$ and H'_1 are shown in (c). Finally, the forests $S'_\xi - H'_1$ and H_2 are shown in (d)

at most k^{2i} terminals, while H'_i contains at most k^{2i} full components and spans at most k^{2i+1} terminals. This implies that the number of the full components of H_i , with $i = 1, \dots, \ell$, is at most

$$k^{2i-1} \leq k^{2\ell} = 2^{2\lceil 2/\varepsilon \rceil \lceil 1/\xi \rceil} = h.$$

Therefore, by Lemma 1, the cost of the solution returned by the call of the algorithm BUILDST($I', S_\xi + t, h$) is at most $c(S_i)$. As a consequence

$$c(S') \leq \frac{1}{\ell} \sum_{i=1}^{\ell} c(S_i). \tag{1}$$

Now we prove an upper bound to the cost of each S_i . Let $\Delta_{OPT} = (1 + \xi)c(OPT') - c(OPT)$. As any Steiner tree of I' is also a Steiner tree of I , $c(OPT') \geq c(OPT)$; therefore $\Delta_{OPT} \geq 0$. Since the addition of H_i to $S'_\xi - H'_{i-1}$ yields a Steiner tree of I , the cost of this tree is lower bounded by the cost of OPT . As a consequence, using Theorem 1 in the second inequality that follows, $c(OPT) \leq d(S'_\xi) - d(H'_{i-1}) + d(H_i) \leq (1 + \xi)c(OPT') - d(H'_{i-1}) + d(H_i)$, i.e.,

$$d(H_i) \geq d(H'_{i-1}) - \Delta_{OPT}. \tag{2}$$

Using Theorem 1 in the first inequality that follows we have that

$$d(S_\xi) \leq (1 + \xi)c(S) \leq \rho(1 + \xi)c(OPT) \leq \rho(1 + \xi)^2c(OPT') - \Delta_{OPT}. \tag{3}$$

Using both (2) and (3) in the second inequality that follows, we can upper bound the cost of S_i with

$$c(S_i) \leq d(S_\xi) - d(H_i) + d(H'_i) \leq \rho(1 + \xi)^2c(OPT') - d(H'_{i-1}) + d(H'_i). \tag{4}$$

Observe that the overall sum of the last two terms on the right-hand side of (4) for every $i \in \{1, \dots, \ell\}$ is a telescoping sum. As a consequence, if for every $i \in \{1, \dots, \ell\}$ we substitute the term $c(S_i)$ in (1) with the corresponding upper bound in (4), and use Theorem 1 to derive that $d(H'_\ell) \leq d(S'_\xi) \leq (1 + \xi)c(OPT')$, we obtain

$$c(S') \leq \frac{1}{\ell} \sum_{i=1}^{\ell} c(S_i) \leq \rho(1 + \xi)^2c(OPT') + \frac{1 + \xi}{\ell}c(OPT') \leq (\rho + \varepsilon)c(OPT'),$$

where last inequality holds by the choice of ξ and ℓ , and because $\rho \leq 2$ and $\varepsilon \leq 1$. This completes the proof. □

5 The Cost of an Edge Increases

In this section we consider the local modification in which the cost of an edge $e = (u, v) \in E(G)$ increases by $\Delta > 0$. More formally, we have that $G' = G$, $R' = R$, whereas, for every $e' \in E(G)$,

$$c'(e') = \begin{cases} c(e) + \Delta & \text{if } e' = e; \\ c(e') & \text{otherwise.} \end{cases}$$

For the sake of readability, in this section we drop the superscripts from G' and R' . Furthermore, we denote by c_{-e} the edge-cost function c restricted to $G - e$ and by d_{-e} the corresponding shortest-path metric.

Let $\xi = \varepsilon/10$ and $h = 2^{2(1+\lceil 1/\xi \rceil)\lceil 2/\varepsilon \rceil}$. The algorithm, whose pseudocode is reported in Algorithm 3, first checks whether $e \in E(S)$. If this is not the case, then the algorithm returns S . If $e \in E(S)$, then let S_u and S_v denote the two trees of $S - e$ containing u and v , respectively. The algorithm first computes a k -restricted Steiner tree $S_{u,\xi}$ (resp., $S_{v,\xi}$), with $k = 2^{2+\lceil 1/\xi \rceil}$, of S_u (resp., S_v) such that $u \in V(S_{u,\xi})$ (resp., $v \in V(S_{v,\xi})$). Then the algorithm computes a solution S' via the call of BUILDST(I' , $S_{u,\xi} + S_{v,\xi}$, h), and, finally, it returns the cheapest solution between S and S' . As we will see, the removal of e from S is necessary to guarantee that the cost of the processed Steiner forest of I' (i.e., $S - e$) is upper bounded by the cost of OPT' .

Algorithm 3: The cost of an edge $e = (u, v)$ increases by $\Delta > 0$.

```

1  $\xi \leftarrow \varepsilon/10$ ;
2  $h \leftarrow 2^{2(1+\lceil 1/\xi \rceil)\lceil 2/\varepsilon \rceil}$ ;
3 if  $e \notin E(S)$  then
4    $S' \leftarrow S$ ;
5 else
6   let  $S_u$  and  $S_v$  be the tree of  $S - e$  containing  $u$  and  $v$ , respectively;
7    $I_u \leftarrow (G - e, c_{-e}, V(S_u) \cap R)$ ;
8    $I_v \leftarrow (G - e, c_{-e}, V(S_v) \cap R)$ ;
9    $S_{u,\xi} \leftarrow \text{RESTRICTEDST}(I_u, S_u, \xi, u)$ ;
10   $S_{v,\xi} \leftarrow \text{RESTRICTEDST}(I_v, S_v, \xi, v)$ ;
11   $S_\xi \leftarrow S_{u,\xi} + S_{v,\xi}$ ;
12   $S' \leftarrow \text{BUILDST}(I', S_\xi, h)$ ;
13  if  $c'(S) \leq c'(S')$  then  $S' \leftarrow S$ ;
14 return  $S'$ ;

```

Theorem 3 Let $\langle I, S, I' \rangle$ be an instance of Steiner tree reoptimization, where S is a ρ -approximate solution of I and I' is obtained from I by increasing the cost of a single edge. Then Algorithm 3 computes a $(\rho + \varepsilon)$ -approximate Steiner tree of I' in polynomial time.

Proof Corollary 1 implies that the computation of both $S_{u,\xi}$ and $S_{v,\xi}$ requires polynomial time. Therefore, using Lemma 3 and the fact that h is a constant value, the call of BUILDST(I' , S_ξ , h) outputs a solution S' in polynomial time. Hence, the overall time complexity of Algorithm 3 is polynomial.

In the following we show that Algorithm 3 returns a $(\rho + \varepsilon)$ -approximate solution. Observe that the local modification does not change the set of feasible solutions, i.e., a tree is a Steiner tree of I iff it is a Steiner tree of I' . This implies that OPT' is a Steiner tree of I as well. Thanks to this observation, if $e \notin E(S)$ we have that

$$c'(S) = c(S) \leq \rho c(OPT) \leq \rho c(OPT') \leq \rho c'(OPT'),$$

i.e., the solution returned by the algorithm is a ρ -approximate solution. Moreover, if $e \in E(OPT')$, then $c'(OPT') = c(OPT') + \Delta$ or, equivalently, $c(OPT') = c'(OPT') - \Delta$. As a consequence,

$$c'(S) \leq c(S) + \Delta \leq \rho c(OPT) + \Delta \leq \rho c(OPT') + \Delta \leq \rho c'(OPT'),$$

i.e., once again, the solution returned by the algorithm is a ρ -approximate solution. Therefore, in the rest of the proof, we assume that $e \in E(S)$ as well as $e \notin E(OPT')$.

Let S'_ξ denote the Steiner tree of $\langle K_n, d_{-e}, R \rangle$ that is returned by the call of $\text{RESTRICTEDST}(\langle G - e, c_{-e}, R \rangle, OPT', \xi)$. Using Theorem 1 and the fact that $e \notin E(OPT')$ we have that

$$d_{-e}(S'_\xi) \leq (1 + \xi)c_{-e}(OPT') = (1 + \xi)c'(OPT'). \tag{5}$$

Let H'_0 be a full component of S'_ξ whose addition to S_ξ yields a Steiner tree of I'_n (ties are broken arbitrarily). For every $i = 1, \dots, \ell$, with $\ell = \lceil 2/\varepsilon \rceil$, we define:

- H_i as the forest consisting of a minimal set of full components of S_ξ such that the addition of H_i and e to $S'_\xi - H'_{i-1}$ yields a Steiner tree of I'_n ⁵;
- H'_i as the forest consisting of a minimal set of full components of S'_ξ whose addition to $S_\xi - H_i$ yields a Steiner tree of I'_n .

For the rest of the proof, we denote by S_i the Steiner tree of I' obtained by augmenting $S_\xi - H_i$ with H'_i . Observe that S_i does not contain e . Therefore

$$c'(S_i) \leq c_{-e}(S_i) \leq d_{-e}(S_\xi) - d_{-e}(H_i) + d_{-e}(H'_i). \tag{6}$$

Let $k = 2^{2+\lceil 1/\xi \rceil}$ and $r = 2^{\lceil 1/\xi \rceil}$. Observe that S_ξ is a k -restricted Steiner forest of I_n (see Corollary 1), while S'_ξ is an r -restricted Steiner tree of I'_n (see Theorem 1). Therefore, H'_0 spans at most r terminals. Moreover, by repeatedly using Lemma 2, H_i contains at most $k^{i-1}r^i$ full components and spans at most $k^i r^i$ terminals, while H'_i contains at most $k^i r^i$ full components and spans at most $k^i r^{i+1}$ terminals. This implies that the number of full components of H_i , with $i = 1, \dots, \ell$, is at most

$$k^{i-1}r^i \leq k^\ell r^\ell = 2^{2(1+\lceil 1/\xi \rceil)\lceil 2/\varepsilon \rceil} = h.$$

Therefore, by Lemma 1, the cost of the solution returned by the call of algorithm $\text{BUILDST}(I', S'_\xi, h)$ is at most the cost of S_i . As a consequence

$$c'(S') \leq \frac{1}{\ell} \sum_{i=1}^{\ell} c'(S_i). \tag{7}$$

Now we prove an upper bound to the cost of each S_i , with $i \in \{1, \dots, \ell\}$. Let $\Delta_{OPT} = (1 + \xi)c'(OPT') - c(OPT)$ and observe that $\Delta_{OPT} \geq 0$. Since the addition

⁵ We observe that the existence of H_i is guarantee by the fact that $S_\xi + e$ is a Steiner tree of I_n : in fact, $S_\xi = S_{u,\xi} + S_{v,\xi}$ is a forest of two trees such that $u \in S_{u,\xi}$ and $v \in S_{v,\xi}$.

of H_i and e to $S'_\xi - H'_{i-1}$ yields a Steiner tree of I , the cost of this tree is lower bounded by the cost of OPT . As a consequence, using (5) together with Theorem 1 in the third inequality that follows,

$$\begin{aligned} c(OPT) &\leq d(S'_\xi - H'_{i-1}) + d(H_i) + d(e) \leq d_{-e}(S'_\xi - H'_{i-1}) + d_{-e}(H_i) + c(e) \\ &= d_{-e}(S'_\xi) - d_{-e}(H'_{i-1}) + d_{-e}(H_i) + c(e) \\ &\leq (1 + \xi)c'(OPT') - d_{-e}(H'_{i-1}) + d_{-e}(H_i) + c(e), \end{aligned}$$

from which we derive

$$d_{-e}(H_i) \geq d_{-e}(H'_{i-1}) - \Delta_{OPT} - c(e). \tag{8}$$

Using Corollary 1 twice in the first inequality that follows we have that

$$\begin{aligned} d_{-e}(S_\xi) &= d_{-e}(S_{u,\xi}) + d_{-e}(S_{v,\xi}) \leq (1 + \xi)c_{-e}(S_u) + (1 + \xi)c_{-e}(S_v) \\ &= (1 + \xi)c(S) - (1 + \xi)c(e) \leq \rho(1 + \xi)c(OPT) - c(e) \\ &\leq \rho(1 + \xi)^2c'(OPT') - \Delta_{OPT} - c(e). \end{aligned} \tag{9}$$

Starting from (6) and using both (8) and (9) in the second inequality that follows, we can upper bound the cost of S_i , for every $i \in \{1, \dots, \ell\}$, with

$$\begin{aligned} c'(S_i) &\leq d_{-e}(S_\xi) - d_{-e}(H_i) + d_{-e}(H'_i) \\ &\leq \rho(1 + \xi)^2c'(OPT') - d_{-e}(H'_{i-1}) + d_{-e}(H'_i). \end{aligned} \tag{10}$$

As a consequence, if for every $i \in \{1, \dots, \ell\}$ we substitute the term $c'(S_i)$ in (7) with the corresponding upper bound in (10), and use (5) to derive $d_{-e}(H'_\ell) \leq d_{-e}(S'_\xi) \leq (1 + \xi)c'(OPT')$, we obtain

$$c'(S') \leq \frac{1}{\ell} \sum_{i=1}^{\ell} c'(S_i) \leq \rho(1 + \xi)^2c'(OPT') + \frac{1 + \xi}{\ell}c'(OPT') \leq (\rho + \varepsilon)c'(OPT'),$$

where last inequality holds by the choice of ξ and ℓ , and because $\rho \leq 2$ and $\varepsilon \leq 1$. This completes the proof. □

6 A Terminal Becomes a Steiner Vertex

In this section we consider the local modification in which a terminal vertex $t \in R$ becomes a Steiner vertex. More precisely, the input instance satisfies the following: $G' = G$, $c' = c$, whereas $R' = R \setminus \{t\}$. Therefore, for the sake of readability, in this section we drop the superscripts from c' , d' , and G' . We recall that, unless $\rho = 1$, the problem is as hard to approximate as the *STP* [32]. Therefore, in this section we assume that $S = OPT$ is an optimal solution, i.e., $\rho = 1$.

Algorithm 4: A terminal t of G becomes a Steiner vertex.

```

1  $\xi \leftarrow \varepsilon/10$ ;
2  $h \leftarrow (1 + \lceil 1/\xi \rceil)2^{2(1+\lceil 1/\xi \rceil)\lceil 2/\varepsilon \rceil}$ ;
3 let  $P^*$  be a maximal (w.r.t. hop-length) path of  $OPT$  starting from  $t$  and such that  $OPT - P^*$  is a
  Steiner forest of  $I'$  with at most  $1 + \lceil 1/\xi \rceil$  trees;
4 let  $S_\xi$  be the empty graph;
5 for every  $v \in V(P^*) \cap V(OPT - P^*)$  do
6   let  $S_v$  be the tree of  $OPT - P^*$  that spans  $v$ ;
7    $S_{v,\xi} \leftarrow \text{RESTRICTEDST}(I, S_v, \xi, v)$ ;
8    $S_\xi \leftarrow S_\xi + S_{v,\xi}$ ;
9  $S' \leftarrow \text{BUILDST}(I', S_\xi, h)$ ;
10 if  $c(OPT) \leq c(S')$  then return  $OPT$  else return  $S'$ 
```

Let $\xi = \varepsilon/10$ and $h = (1 + \lceil 1/\xi \rceil)2^{2(1+\lceil 1/\xi \rceil)\lceil 2/\varepsilon \rceil}$. As $c(OPT')$ may be much smaller than $c(OPT)$, we first need to remove a path from OPT to ensure that the obtained Steiner forest of I' has a constant number of trees and a cost upper bounded by $(1 + \xi)c(OPT')$. The algorithm, whose pseudocode is reported in Algorithm 4, precomputes a maximal (w.r.t. hop-length) path P^* of OPT having t as one of its endvertices and such that $OPT - P^*$ is a Steiner forest of I' with at most $1 + \lceil 1/\xi \rceil$ trees (we recall that only trees of the forest spanning one or more terminals are kept; therefore, the Steiner vertices of P^* having degree 2 in S are not contained in $OPT - P^*$). As the following lemma shows, P^* is exactly the path we are searching for.

Lemma 4 $OPT - P^*$ is a Steiner forest of I' with at most $1 + \lceil 1/\xi \rceil$ trees and $c(P^*) \geq c(OPT) - (1 + \xi)c(OPT')$.

Proof By definition of P^* , $OPT - P^*$ is a Steiner forest of I' with at most $1 + \lceil 1/\xi \rceil$ trees. To show that $c(P^*) \geq c(OPT) - (1 + \xi)c(OPT')$, we split the proof into two cases, depending on whether P^* spans additional terminals other than t .

We consider the case in which P^* spans additional terminals other than t . In this case $OPT' + P^*$ is a Steiner tree of I . Therefore, $c(OPT) \leq c(OPT') + c(P^*)$, from which we derive $c(P^*) \geq c(OPT) - c(OPT') \geq c(OPT) - (1 + \xi)c(OPT')$.

Now, we consider the case in which P^* does not span another terminal other than t . In this case $OPT - P^*$ yields a Steiner forest of I' of exactly $1 + \lceil 1/\xi \rceil$ trees. This implies that $OPT - P^*$ contains at least $1 + \lceil 1/\xi \rceil$ pairwise edge-disjoint paths connecting vertices of P^* with terminals in R' . Let P be any of these paths that minimizes its cost (ties are broken arbitrarily). As the addition of both P^* and P to OPT' yields a Steiner tree of I , $(1 + \lceil 1/\xi \rceil)c(P) + c(P^*) \leq c(OPT) \leq c(OPT') + c(P) + c(P^*)$, from which we derive $c(P) \leq \xi c(OPT')$. As a consequence $c(OPT) \leq c(OPT') + c(P) + c(P^*) \leq c(OPT') + \xi c(OPT') + c(P^*)$, i.e., $c(P^*) \geq c(OPT) - (1 + \xi)c(OPT')$. \square

The algorithm then computes a k -restricted Steiner forest S_ξ of $OPT - P^*$, with $k = 2^{2+\lceil 1/\xi \rceil}$, such that $S_\xi + P^*$ yields a Steiner tree of I . Finally, the algorithm calls $\text{BUILDST}(I', S_\xi, h)$ to compute a feasible solution S' and returns the cheapest solution between OPT and S' .

Theorem 4 *Let $\langle I, OPT, I' \rangle$ be an instance of Steiner tree reoptimization, where OPT is an optimal solution of I and I' is obtained from I by changing the status of a single vertex from terminal to Steiner. Then Algorithm 4 computes a $(1 + \varepsilon)$ -approximate Steiner tree of I' in polynomial time.*

Proof Corollary 1 implies that the computation of each $S_{v,\xi}$, with $v \in V(P^*)$, requires polynomial time. By Lemma 4, S_ξ is a Steiner forest of I_n with a constant number of trees. Therefore, using Lemma 3 and the fact that h is a constant value, the call of $BUILDST(I', S_\xi, h)$ outputs a solution in polynomial time. Hence, the overall time complexity of Algorithm 4 is polynomial.

In the following we show that Algorithm 4 returns a $(1 + \varepsilon)$ -approximate solution. W.l.o.g., we can assume that $c(OPT) > (1 + \varepsilon)c(OPT')$ as otherwise the solution returned by the algorithm, whose cost is less than or equal to $c(OPT)$, would already be a $(1 + \varepsilon)$ -approximate solution.

Let S'_ξ be the Steiner tree of I'_n that is returned by the call of algorithm $RESTRICTEDST(I', OPT', \xi)$ and let H'_0 be the forest consisting of a minimal set of full components of S'_ξ whose addition to S_ξ yields a Steiner tree of I'_n . For every $i = 1, \dots, \ell$, with $\ell = \lceil 2/\varepsilon \rceil$, we define:

- H_i as the forest consisting of a minimal set of full components of S_ξ such that the addition of H_i and P^* to $S'_\xi - H'_{i-1}$ yields a Steiner tree of I_n ⁶;
- H'_i as the forest consisting of a minimal set of full components of S'_ξ whose addition to $S_\xi - H_i$ yields a Steiner tree of I'_n .

For the rest of the proof, we denote by S_i the Steiner tree of I' obtained by augmenting $S_\xi - H_i$ with H'_i .

Let $k = 2^{2+\lceil 1/\xi \rceil}$ and $r = 2^{\lceil 1/\xi \rceil}$. Observe that S_ξ is a k -restricted Steiner forest of I_n (see Corollary 1), while S'_ξ is an r -restricted Steiner tree of I'_n (see Theorem 1). Using Lemma 2, H'_0 contains at most $(1 + \lceil 1/\xi \rceil)$ full components and spans at most $(1 + \lceil 1/\xi \rceil)r$ terminals. By repeatedly using Lemma 2, H_i contains at most $(1 + \lceil 1/\xi \rceil)k^{i-1}r^i$ full components and spans at most $(1 + \lceil 1/\xi \rceil)k^i r^i$ terminals, while H'_i contains at most $(1 + \lceil 1/\xi \rceil)k^i r^i$ full components and spans at most $(1 + \lceil 1/\xi \rceil)k^i r^{i+1}$ terminals. This implies that the number of full components of H_i , with $i = 1, \dots, \ell$, is at most

$$(1 + \lceil 1/\xi \rceil)k^{i-1}r^i \leq (1 + \lceil 1/\xi \rceil)k^\ell r^\ell = (1 + \lceil 1/\xi \rceil)2^{2(1+\lceil 1/\xi \rceil)\lceil 2/\varepsilon \rceil} = h.$$

Therefore, by Lemma 1, the cost of the solution returned by the algorithm is at most the cost of S_i , for every $i = 1, \dots, \ell$. As a consequence

$$c(S') \leq \frac{1}{\ell} \sum_{i=1}^{\ell} c(S_i). \tag{11}$$

Now we prove an upper bound to the cost of each S_i . Let $\Delta_{OPT} = c(OPT) - (1 + \varepsilon)c(OPT')$ and observe that $\Delta_{OPT} \geq 0$ by our assumption. Since the addition of H_i

⁶ We observe that the existence of H_i is guarantee by the fact that $S_\xi + P^*$ is a Steiner tree of I_n .

and P^* to $S'_\xi - H'_{i-1}$ yields a Steiner tree of I , the cost of this tree is lower bounded by the cost of OPT . As a consequence, using Theorem 1 in the second inequality that follows,

$$\begin{aligned} c(OPT) &\leq d(S'_\xi) - d(H'_{i-1}) + d(H_i) + d(P^*) \\ &\leq (1 + \xi)c(OPT') - d(H'_{i-1}) + d(H_i) + c(P^*), \end{aligned}$$

from which we derive

$$d(H_i) \geq d(H'_{i-1}) + \Delta_{OPT} - c(P^*). \tag{12}$$

Using Corollary 1 and Lemma 4 in the first and last inequality that follows, respectively, we have that

$$\begin{aligned} d(S_\xi) &= \sum_{v \in V(P^*)} d(S_{v,\xi}) \leq (1 + \xi)c(OPT - P^*) \leq (1 + \xi)c(OPT) - (1 + \xi)c(P^*) \\ &\leq (1 + \xi)^2c(OPT') + (1 + \xi)\Delta_{OPT} - (1 + \xi)c(P^*) \\ &\leq (1 + \xi)^2c(OPT') + \Delta_{OPT} - c(P^*). \end{aligned} \tag{13}$$

Using both (12) and (13) in the second inequality that follows, we can upper bound the cost of S_i , for every $i \in \{1, \dots, \ell\}$, with

$$c(S_i) \leq d(S_\xi) - d(H_i) + d(H'_i) \leq (1 + \xi)^2c(OPT') - d(H'_{i-1}) + d(H'_i). \tag{14}$$

Therefore, if for every $i \in \{1, \dots, \ell\}$ we substitute the term $c(S_i)$ in (11) with the corresponding upper bound in (14), and use Theorem 1 to derive $d(H'_\ell) \leq d(S'_\xi) \leq (1 + \xi)c(OPT')$, we obtain

$$c(S') \leq \frac{1}{\ell} \sum_{i=1}^{\ell} c(S_i) \leq (1 + \xi)^2c(OPT') + \frac{1 + \xi}{\ell}c(OPT') \leq (1 + \varepsilon)c(OPT'),$$

where last inequality holds by the choice of ξ and ℓ , and because $\varepsilon \leq 1$. This completes the proof. □

7 The Cost of an Edge Decreases

In this section we consider the local modification in which the cost of an edge $e = (u, v) \in E$ decreases by $0 < \Delta \leq c(e)$. More precisely, the input instance satisfies the following: $G' = G, R' = R$, whereas for every $e' \in E$,

$$c'(e') = \begin{cases} c(e) - \Delta & \text{if } e' = e; \\ c(e') & \text{otherwise.} \end{cases}$$

Algorithm 5: The cost of an edge $e = (u, v)$ of G decreases by $0 < \Delta \leq c(e)$.

```

1  $\xi \leftarrow \varepsilon/10$ ;
2  $h \leftarrow 2(1 + \lceil 1/\xi \rceil)2^{2(2+\lceil 1/\xi \rceil)\lceil 2/\varepsilon \rceil}$ ;
3  $S' \leftarrow OPT$ ;
4 let  $\mathcal{H}$  be the set of all the paths of  $OPT$  such that, for every  $P \in \mathcal{H}$ ,  $OPT - P$  is a Steiner forest of  $I$ 
   with at most  $2(1 + \lceil 1/\xi \rceil)$  trees;
5 for every  $P \in \mathcal{H}$  do
6   let  $S_\xi$  be the empty graph;
7   for every  $v \in V(P) \cap V(OPT - P)$  do
8     let  $S_v$  be the tree of  $OPT - P$  that spans  $v$ ;
9      $S_{v,\xi} \leftarrow \text{RESTRICTEDST}((G - e, c_{-e}, R), S_v, \xi, v)$ ;
10     $S_\xi \leftarrow S_\xi + S_{v,\xi}$ ;
11   $S'' \leftarrow \text{BUILDST}(I', S_\xi, h)$ ;
12  if  $c'(S'') \leq c'(S')$  then  $S' \leftarrow S''$ ;
13 return  $S'$ ;

```

For the sake of readability, in this section we drop the superscripts from G' and R' . Furthermore, for the rest of this section, we denote by c_{-e} the edge-cost function c restricted to $G - e$ and by d_{-e} the corresponding shortest-path metric. We recall that, unless $\rho = 1$, the problem is as hard to approximate as the *STP* [32]. Therefore, in this section we assume that $S = OPT$ is an optimal solution, i.e., $\rho = 1$.

Let $\xi = \varepsilon/10$ and $h = 2(1 + \lceil 1/\xi \rceil)2^{2(2+\lceil 1/\xi \rceil)\lceil 2/\varepsilon \rceil}$. As $c'(OPT')$ may be much smaller than $c'(OPT)$, we first need to remove a path from OPT to ensure that the obtained Steiner forest of I has a constant number of trees and a cost upper bounded by $(1 + \xi)c'(OPT')$. The algorithm, whose pseudocode is reported in Algorithm 5, precomputes a set \mathcal{H} containing all the paths of OPT such that, for every $P \in \mathcal{P}$, $OPT - P$ is a Steiner forest of I with at most $2(1 + \lceil 1/\xi \rceil)$ trees. As the following lemma shows, there exists $P^* \in \mathcal{H}$, which is exactly the path we are searching for when $c'(OPT')$ is much smaller than $c'(OPT)$.

Lemma 5 *If $e \in E(OPT')$ and $e \notin E(OPT)$, then there exists $P^* \in \mathcal{H}$ s.t. $c'(P^*) \geq c(OPT) - (1 + \xi)c'(OPT')$.*

Proof Let S'_u and S'_v denote the trees of $OPT' - e$ containing u and v , respectively. Let $R_u = V(S'_u) \cap R$ and $R_v = V(S'_v) \cap R$. As $e \notin E(OPT)$, there exists a path in OPT , say \bar{P} , that connects a terminal in R_u , say u' , with a terminal in R_v , say v' , and such that all the internal vertices of \bar{P} are Steiner vertices. Let σ be the number of Steiner vertices of \bar{P} whose corresponding degrees in OPT are all greater than or equal to 3. We divide the proof into two complementary cases depending on the value of σ . In both cases, we prove that P^* is a subpath of \bar{P} .

We consider the case in which $\sigma \leq 2(1 + \lceil 1/\xi \rceil)$. Let $P^* = \bar{P}$. Clearly $P^* \in \mathcal{H}$ and $OPT' - e + P^*$ is a Steiner tree of I . Therefore, $c(OPT) \leq c'(OPT' - e) + c'(P^*) \leq c'(OPT') + c'(P^*)$, from which we derive $c'(P^*) \geq c(OPT) - c'(OPT') \geq c(OPT) - (1 + \xi)c'(OPT')$.

Now, we consider the case in which $\sigma > 2(1 + \lceil 1/\xi \rceil)$. Let X be the set of the σ Steiner vertices of \bar{P} whose corresponding degrees in OPT are all greater than or equal to 3. Let x_1, \dots, x_σ be the vertices in X in order in which they are encountered

while traversing \bar{P} from u' to v' . Finally, let $x_0 = u'$ and $x_{\sigma+1} = v'$. We associate a label $\lambda(i) \in \{u, v\}$ with each x_i to indicate the existence of a path in OPT between x_i and a terminal in $R_{\lambda(i)}$ that is edge-disjoint from \bar{P} (if $\lambda(i)$ can assume both values, we arbitrarily choose one of them). We set $\lambda(0) = u$ and $\lambda(\sigma + 1) = v$ to indicate that u' and v' have the trivial paths towards themselves, respectively. For $i = 0, \dots, \sigma - 2\lceil 1/\xi \rceil$, let $L_i = \lambda(i), \dots, \lambda(i + 1 + 2\lceil 1/\xi \rceil)$ be the sequence of $2(1 + \lceil 1/\xi \rceil)$ consecutive labels starting from i . One of the following holds:

- (a) L_0 contains at least $1 + \lceil 1/\xi \rceil$ occurrences of v ;
- (b) $L_{\sigma - 2\lceil 1/\xi \rceil}$ contains at least $1 + \lceil 1/\xi \rceil$ occurrences of u ;
- (c) there is an index i such that the sequence L_i contains exactly $1 + \lceil 1/\xi \rceil$ occurrences of u and $1 + \lceil 1/\xi \rceil$ occurrences of v .

Indeed, let μ_i and ν_i denote the occurrences of u and v in L_i , respectively. Observe that $|\mu_i - \mu_{i+1}| \in \{0, 1\}$ as well as $|\nu_i - \nu_{i+1}| \in \{0, 1\}$. More precisely:

- $\mu_i = \mu_{i+1}$ iff $\nu_i = \nu_{i+1}$;
- $\mu_i = \mu_{i+1} + 1$ iff $\nu_i = \nu_{i+1} - 1$;
- $\mu_i = \mu_{i+1} - 1$ iff $\nu_i = \nu_{i+1} + 1$.

Therefore, if neither (a) nor (b) holds, i.e., $\mu_0 > \nu_0$ and $\mu_{\sigma - 2\lceil 1/\xi \rceil} < \nu_{\sigma - 2\lceil 1/\xi \rceil}$, then (c) must hold.

Let P^* be the path containing exactly the Steiner vertices corresponding to a sequence L_i that satisfies any of the above three conditions (ties are broken arbitrarily). Observe that $P^* \in \mathcal{H}$. Let P_u (resp., P_v) be a minimum-cost path between a vertex of P^* and a vertex of R_u (resp., R_v) that is edge-disjoint from P^* . W.l.o.g., if P^* contains u' , then we can assume that P_u is the path containing only vertex u ; similarly, if P^* contains v' , then we can assume that P_v is the path containing only vertex v . As the addition of both P^* , P_u and P_v to $OPT' - e$ yields a Steiner tree of I , $(1 + \lceil 1/\xi \rceil)(c'(P_u) + c'(P_v)) + c'(P^*) = (1 + \lceil 1/\xi \rceil)(c(P_u) + c(P_v)) + c(P^*) \leq c(OPT) \leq c'(OPT') + c'(P_u) + c'(P_v) + c'(P^*)$, from which we derive $c'(P_u) + c'(P_v) \leq \xi c'(OPT')$. As a consequence $c(OPT) \leq c'(OPT') + c'(P_u) + c'(P_v) + c'(P^*) \leq c'(OPT') + \xi c'(OPT') + c'(P^*)$, i.e., $c'(P^*) \geq c(OPT) - (1 + \xi)c(OPT')$. \square

The algorithm always returns a solution whose cost is upper bounded by $c'(OPT)$. This allows us to guarantee that the output of the algorithm is a $(1 + \varepsilon)$ -approximate solution when $e \notin E(OPT')$ or $e \in E(OPT)$. In the complementary case in which $e \in E(OPT')$ and $e \notin E(OPT)$ the algorithm computes a $(1 + \varepsilon)$ -approximate solution using the path P^* that satisfies the conditions stated in Lemma 5. More precisely, the algorithm computes a k -restricted Steiner forest S_ξ of $OPT - P^*$ w.r.t. the STP instance $\langle G - e, c_{-e}, R \rangle$, with $k = 2^{2 + \lceil 1/\xi \rceil}$, such that $S_\xi + P^*$ yields a Steiner tree of I . Finally, the algorithm calls $BUILDST(I', S_\xi, h)$ to compute a feasible solution S'' and returns a solution whose cost is upper bounded by $c'(S'')$.

Theorem 5 *Let $\langle I, OPT, I' \rangle$ be an instance of Steiner tree reoptimization, where OPT is an optimal solution of I and I' is obtained from I by decreasing the cost of a single edge. Then Algorithm 5 computes a $(1 + \varepsilon)$ -approximate Steiner tree of I' in polynomial time.*

Proof Corollary 1 implies that the computation of each $S_{v,\xi}$, with $v \in V(P)$, requires polynomial time. Observe that $OPT - P$ is a Steiner forest of I of at most $2(1 + \lceil 1/\xi \rceil)$ trees. As a consequence S_ξ is a Steiner forest of I_n with at most $2(1 + \lceil 1/\xi \rceil)$ trees. Therefore, using Lemma 3 and the fact that h is a constant value, the call of $BUILDST(I', S_\xi, h)$ outputs a solution in polynomial time. Finally, since \mathcal{H} contains $O(n^2)$ paths (one path for each pair of vertices in the worst case), the overall time complexity of Algorithm 5 is polynomial.

In the following we show that Algorithm 5 returns a $(1 + \varepsilon)$ -approximate solution. W.l.o.g., we can assume that $c(OPT) > (1 + \varepsilon)c(OPT')$, as otherwise, the solution returned by the algorithm, whose cost is always less than or equal to $c'(OPT) \leq c(OPT)$, would already be a $(1 + \varepsilon)$ -approximate solution. As a consequence, we have that $e \notin E(OPT)$ as well as $e \in E(OPT')$: indeed, if even one of these two conditions were not satisfied, then OPT would be an optimal solution of I' .

Let S'_u and S'_v be the two trees of $OPT' - e$ that contain u and v , respectively. Let $S'_{u,\xi}$ and $S'_{v,\xi}$ be the two k -restricted trees of I'_n , with $k = 2^{2+1/\xi}$, that are returned by the call $RESTRICTEDST((G - e, c_{-e}, R), S'_u, \xi, u)$ and $RESTRICTEDST((G - e, c_{-e}, R), S'_v, \xi, v)$, respectively. Finally, let $S'_\xi = S'_{u,\xi} + S'_{v,\xi}$, $P^* \in \mathcal{H}$ the path satisfying the conditions stated in Lemma 5, and S_ξ the k -restricted Steiner forest of I that is computed by the algorithm when $P = P^*$. Let H'_0 be the forest consisting of a minimal set of full components of S'_ξ such that the addition of H'_0 and e to $S_\xi + e$ yields a Steiner tree of I'_n . For every $i = 1, \dots, \ell$, with $\ell = \lceil 2/\varepsilon \rceil$, we define:

- H_i as the forest consisting of a minimal set of full components of S_ξ such that the addition of H_i and P^* to $S'_\xi - H'_{i-1}$ yields a Steiner tree of I_n ⁷;
- H'_i as the forest consisting of a minimal set of full components of S'_ξ such that the addition of H'_i and e to $S_\xi - H_i$ yields a Steiner tree of I'_n ⁸.

For the rest of the proof, we denote by S_i the Steiner tree of I' obtained by augmenting $S_\xi - H_i$ with H'_i and e .

Let $k = 2^{2+\lceil 1/\xi \rceil}$. Observe that S_ξ and S'_ξ are k -restricted Steiner forests of I_n and I'_n , respectively (see Corollary 1). Observe that $S_\xi + e$ is a forest of at most $1 + 2(1 + \lceil 1/\xi \rceil)$ trees. Therefore, using Lemma 2, H'_0 contains at most $2(1 + \lceil 1/\xi \rceil)$ full components and spans at most $2(1 + \lceil 1/\xi \rceil)k$ terminals. Therefore, by repeatedly using Lemma 2, H_i contains at most $2(1 + \lceil 1/\xi \rceil)k^{2i-1}$ full components and spans at most $2(1 + \lceil 1/\xi \rceil)k^{2i}$ terminals, while H'_i contains at most $2(1 + \lceil 1/\xi \rceil)k^{2i}$ full components and spans at most $2(1 + \lceil 1/\xi \rceil)k^{2i+1}$ terminals. This implies that the number of full components of H_i , with $i = 1, \dots, \ell$, is at most

$$2(1 + \lceil 1/\xi \rceil)k^{2i-1} \leq 2(1 + \lceil 1/\xi \rceil)k^{2\ell} = 2(1 + \lceil 1/\xi \rceil)2^{2(2+\lceil 1/\xi \rceil)\lceil 2/\varepsilon \rceil} = h.$$

⁷ We observe that the existence of H_i is guaranteed as $S_\xi + P^*$ is a Steiner tree of I_n .

⁸ We observe that the existence of H_i is guaranteed as $S'_\xi + e$ is a Steiner tree of I'_n .

Therefore, by Lemma 1, the cost of the solution returned by the algorithm cannot be worse than the cost of S_i , for every $i = 1, \dots, \ell$. As a consequence

$$c'(S') \leq \frac{1}{\ell} \sum_{i=1}^{\ell} c'(S_i). \tag{15}$$

Now we prove an upper bound to the cost of each S_i . Let $\Delta_{OPT} = c(OPT) - (1 + \xi)c'(OPT')$ and observe that $\Delta_{OPT} \geq 0$ by our assumption. Since the addition of H_i and P^* to $S'_\xi - H'_{i-1}$ yields a Steiner tree of I , the cost of this tree is lower bounded by the cost of OPT . As a consequence, using Corollary 1 in the third inequality that follows,

$$\begin{aligned} c(OPT) &\leq d_{-e}(S'_\xi) - d_{-e}(H'_{i-1}) + d_{-e}(H_i) + d_{-e}(P^*) \\ &\leq (1 + \xi)c_{-e}(OPT' - e) - d_{-e}(H'_{i-1}) + d_{-e}(H_i) + d_{-e}(P^*) \\ &\leq (1 + \xi)c'(OPT') - c'(e) - d_{-e}(H'_{i-1}) + d_{-e}(H_i) + d_{-e}(P^*) \\ &\leq (1 + \xi)c'(OPT') - d_{-e}(H'_{i-1}) + d_{-e}(H_i) + c'(P^*) - c'(e), \end{aligned}$$

from which we derive

$$d_{-e}(H_i) \geq d_{-e}(H'_{i-1}) + \Delta_{OPT} - c'(P^*) + c'(e). \tag{16}$$

Using Corollary 1 and Lemma 5 respectively in the first and last inequality that follows we have that

$$\begin{aligned} d_{-e}(S'_\xi) &= \sum_{v \in V(P^*)} d_{-e}(S_{v,\xi}) \leq (1 + \xi)c_{-e}(OPT - P^*) \\ &\leq (1 + \xi)c(OPT) - (1 + \xi)c(P^*) \\ &\leq (1 + \xi)^2c'(OPT') + (1 + \xi)\Delta_{OPT} - (1 + \xi)c'(P^*) \\ &\leq (1 + \xi)^2c'(OPT') + \Delta_{OPT} - c'(P^*). \end{aligned} \tag{17}$$

Using both (16) and (17) in the second inequality that follows, we can upper bound the cost of S_i , for every $i \in \{1, \dots, \ell\}$, with

$$\begin{aligned} c'(S_i) &\leq d_{-e}(S'_\xi) - d_{-e}(H_i) + d_{-e}(H'_i) + c'(e) \\ &\leq (1 + \xi)^2c'(OPT') - d_{-e}(H'_{i-1}) + d_{-e}(H'_i). \end{aligned} \tag{18}$$

Therefore, if for every $i \in \{1, \dots, \ell\}$ we substitute the term $c(S_i)$ in (15) with the corresponding upper bound in (18), and use Corollary 1 to derive $d_{-e}(H'_\ell) + c'(e) \leq (1 + \xi)c_{-e}(OPT' - e) + c'(e) \leq (1 + \xi)c'(OPT')$, we obtain

$$c'(S') \leq \frac{1}{\ell} \sum_{i=1}^{\ell} c'(S_i) \leq (1 + \xi)^2c'(OPT') + \frac{1 + \xi}{\ell}c'(OPT') \leq (1 + \varepsilon)c'(OPT'),$$

where last inequality holds by the choice of ξ and ℓ , and because $\varepsilon \leq 1$. This completes the proof. \square

8 Conclusions

In this paper we developed polynomial-time approximation schemes for some Steiner tree reoptimization problems and consider the local modifications in which the cost of a single edge can change or a single vertex can change its status, from terminal to Steiner or from Steiner to terminal. We believe that our algorithms can be unified to provide polynomial-time approximation schemes for more involved scenarios in which the local modification affects, at the same time, the cost of a constant number of edges and the status of a constant number of vertices. We leave this extension as an interesting open problem.

It is known that the Steiner tree problem can be optimally solved when the number of terminals is polylogarithmic in n using, for instance, the famous algorithm by Dreyfus and Wagner [30]. Therefore, a very challenging open problem is that of understanding whether polynomial-time approximation schemes still exist when the local modification affects a number of edge-costs and vertices of the graph that are polylogarithmic in n .

Finally, albeit our results are interesting from a theoretical perspective as they are best possible in terms of approximation factor guarantee, our algorithms are too slow to be used in the practice. Unfortunately, this is somehow a weakness of all the most recent approximation algorithms that have been developed for the Steiner tree problem and Steiner tree reoptimization. Therefore, an interesting scenario is that of designing reoptimization algorithms that guarantee a hopefully good constant approximation ratio, but are also efficient in terms of running time and can also be used for practical applications.

Funding Open access funding provided by Università degli Studi dell'Aquila within the CRUI-CARE Agreement.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Archetti, C., Bertazzi, L., Speranza, M.G.: Reoptimizing the traveling salesman problem. *Networks* **42**(3), 154–159 (2003). <https://doi.org/10.1002/net.10091>
2. Archetti, C., Bertazzi, L., Speranza, M.G.: Reoptimizing the 0–1 knapsack problem. *Discrete Appl. Math.* **158**(17), 1879–1887 (2010). <https://doi.org/10.1016/j.dam.2010.08.003>

3. Archetti, C., Guastaroba, G., Speranza, M.G.: Reoptimizing the rural postman problem. *Comput. OR* **40**(5), 1306–1313 (2013). <https://doi.org/10.1016/j.cor.2012.12.010>
4. Ausiello, G., Bonifaci, V., Escoffier, B.: Complexity and Approximation in Reoptimization, pp. 101–129. Imperial College Press, London (2012). https://doi.org/10.1142/9781848162778_0004
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). <https://doi.org/10.1016/j.jda.2008.12.001>
6. Bender, M.A., Farach-Colton, M., Fekete, S.P., Fineman, J.T., Gilbert, S.: Reallocation problems in scheduling. *Algorithmica* **73**(2), 389–409 (2015). <https://doi.org/10.1007/s00453-014-9930-4>
7. Berg, T., Hempel, H.: Reoptimization of traveling salesperson problems: changing single edge-weights. In: Dediu, A., Ionescu, A., Martín-Vide, C. (eds.) *Language and Automata Theory and Applications*. , Third International Conference, LATA 2009, Tarragona, Spain, April 2–8, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5457, pp. 141–151. Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-00982-2_12
8. Bilò, D., Böckenhauer, H., Hromkovic, J., Kráľovic, R., Mömke, T., Widmayer, P., Zych, A.: Reoptimization of Steiner trees. In: Gudmundsson, J. (ed.) *Algorithm Theory—SWAT 2008*, 11th Scandinavian Workshop on Algorithm Theory, Gothenburg, Sweden, July 2–4, 2008, Proceedings, Lecture Notes in Computer Science, vol. 5124, pp. 258–269. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-69903-3_24
9. Bilò, D., Böckenhauer, H., Komm, D., Kráľovic, R., Mömke, T., Seibert, S., Zych, A.: Reoptimization of the shortest common superstring problem. *Algorithmica* **61**(2), 227–251 (2011). <https://doi.org/10.1007/s00453-010-9419-8>
10. Bilò, D., Widmayer, P., Zych, A.: Reoptimization of weighted graph and covering problems. In: Bampis, E., Skutella, M. (eds.) *Approximation and Online Algorithms*, 6th International Workshop, WAOA 2008, Karlsruhe, Germany, September 18–19, 2008. Revised Papers, Lecture Notes in Computer Science, vol. 5426, pp. 201–213. Springer, Berlin (2008) https://doi.org/10.1007/978-3-540-93980-1_16 (2008)
11. Bilò, D., Zych, A.: New advances in reoptimizing the minimum Steiner tree problem. In: Rován, B., Sassone, V., Widmayer, P. (eds.) *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012*, Bratislava, Slovakia, August 27–31, 2012. Proceedings, Lecture Notes in Computer Science, vol. 7464, pp. 184–197. Springer, Berlin https://doi.org/10.1007/978-3-642-32589-2_19(2012)
12. Böckenhauer, H., Forlizzi, L., Hromkovic, 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)
13. Böckenhauer, H., Freiermuth, K., Hromkovic, J., Mömke, T., Sprock, A., Steffen, B.: Steiner tree reoptimization in graphs with sharpened triangle inequality. *J. Discrete Algorithms* **11**, 73–86 (2012). <https://doi.org/10.1016/j.jda.2011.03.014>
14. Böckenhauer, H., Hromkovic, J., Kráľovic, R., Mömke, T., Rossmanith, P.: Reoptimization of Steiner trees: changing the terminal set. *Theor. Comput. Sci.* **410**(36), 3428–3435 (2009). <https://doi.org/10.1016/j.tcs.2008.04.039>
15. Böckenhauer, H., Hromkovic, J., Mömke, T., Widmayer, P.: On the hardness of reoptimization. In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková M. (eds.) *SOFSEM 2008: Theory and Practice of Computer Science*, 34th Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 19–25, 2008, Proceedings, Lecture Notes in Computer Science, vol. 4910, pp. 50–65. Springer, Berlin https://doi.org/10.1007/978-3-540-77566-9_5 (2008)
16. Böckenhauer, H., Hromkovic, J., Sprock, A.: Knowing all optimal solutions does not help for TSP reoptimization. In: Kelemen, J., Kelemenová, A. (eds.) *Computation, Cooperation, and Life - Essays Dedicated to Gheorghe Paun on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science, vol. 6610, pp. 7–15. Springer, Berlin (2011) https://doi.org/10.1007/978-3-642-20000-7_2
17. Böckenhauer, H., Hromkovic, J., Sprock, A.: On the hardness of reoptimization with multiple given solutions. *Fundam. Inform.* **110**(1–4), 59–76 (2011). <https://doi.org/10.3233/FI-2011-528>
18. Böckenhauer, H., Kneis, J., Kupke, J.: Approximation hardness of deadline-TSP reoptimization. *Theor. Comput. Sci.* **410**(21–23), 2241–2249 (2009). <https://doi.org/10.1016/j.tcs.2009.02.016>
19. Böckenhauer, H., Komm, D.: Reoptimization of the metric deadline TSP. *J. Discrete Algorithms* **8**(1), 87–100 (2010). <https://doi.org/10.1016/j.jda.2009.04.001>

20. Borchers, A., Du, D.: The k -Steiner ratio in graphs. *SIAM J. Comput.* **26**(3), 857–869 (1997). <https://doi.org/10.1137/S0097539795281086>
21. Boria, N., Croce, F.D.: Reoptimization in machine scheduling. *Theor. Comput. Sci.* **540**, 13–26 (2014). <https://doi.org/10.1016/j.tcs.2014.04.004>
22. Boria, N., Monnot, J., Paschos, V.T.: Reoptimization of maximum weight induced hereditary subgraph problems. *Theor. Comput. Sci.* **514**, 61–74 (2013). <https://doi.org/10.1016/j.tcs.2012.10.037>
23. Boria, N., Monnot, J., Paschos, V.T.: Reoptimization under vertex insertion: max p_k -free subgraph and max planar subgraph. *Discrete Math. Alg. Appl.* (2011). <https://doi.org/10.1142/S1793830913600045>
24. Boria, N., Paschos, V.T.: Fast reoptimization for the minimum spanning tree problem. *J. Discrete Algorithms* **8**(3), 296–310 (2010). <https://doi.org/10.1016/j.jda.2009.07.002>
25. Boria, N., Paschos, V.T.: A survey on combinatorial optimization in dynamic environments. *RAIRO Oper. Res.* **45**(3), 241–294 (2011). <https://doi.org/10.1051/ro/2011114>
26. Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L.: Steiner tree approximation via iterative randomized rounding. *JACM* **60**(1), 6:1–6:33 (2013). <https://doi.org/10.1145/2432622.2432628>
27. Chlebík, M., Chlebíková, J.: The steiner tree problem on graphs: inapproximability results. *Theor. Comput. Sci.* **406**(3), 207–214 (2008). <https://doi.org/10.1016/j.tcs.2008.06.046>
28. Dai, W.: Reoptimization of minimum latency problem. In: Cao, Y., Chen, J. (eds.) *Computing and Combinatorics—23rd International Conference, COCOON 2017, Hong Kong, China, August 3–5, 2017, Proceedings, Lecture Notes in Computer Science*, vol. 10392, pp. 162–174. Springer, Berlin (2017) https://doi.org/10.1007/978-3-319-62389-4_14
29. Dai, W., Yang, Y.: Reoptimization of minimum latency problem revisited: don't panic when asked to revisit the route after local modifications. *J. Comb. Optim.* **37**(2), 601–619 (2019). <https://doi.org/10.1007/s10878-018-0317-3>
30. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. *Networks* **1**(3), 195–207 (1971). <https://doi.org/10.1002/net.3230010302>
31. Escoffier, B., Milanic, M., Paschos, V.T.: Simple and fast reoptimizations for the Steiner tree problem. *Algorithmic Oper. Res.* **4**(2), 86–94 (2009)
32. Goyal, K., Mömke, T.: Robust reoptimization of Steiner trees. *Algorithmica* **82**(7), 1966–1988 (2020). <https://doi.org/10.1007/s00453-020-00682-x>
33. Hougardy, S., Silvanus, J., Vygen, J.: Dijkstra meets Steiner: a fast exact goal-oriented Steiner tree algorithm. *Math. Program. Comput.* **9**(2), 135–202 (2017). <https://doi.org/10.1007/s12532-016-0110-1>
34. Monnot, J.: A note on the traveling salesman reoptimization problem under vertex insertion. *Inf. Process. Lett.* **115**(3), 435–438 (2015). <https://doi.org/10.1016/j.ipl.2014.11.003>
35. Schäffter, M.W.: Scheduling with forbidden sets. *Discrete Appl. Math.* **72**(1–2), 155–166 (1997). [https://doi.org/10.1016/S0166-218X\(96\)00042-X](https://doi.org/10.1016/S0166-218X(96)00042-X)
36. Zych, A.: Reoptimization of NP-hard problems. Ph.D. thesis, Dissertation, Eidgenössische Technische Hochschule ETH Zürich, Nr. 20257 (2012)
37. Zych, A., Bilò, D.: New reoptimization techniques applied to Steiner tree problem. *Electron. Notes Discrete Math.* **37**, 387–392 (2011). <https://doi.org/10.1016/j.endm.2011.05.066>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.