New algorithms for Steiner tree reoptimization

Davide Bilò
Department of Humanities and Social Sciences, University of Sassari
Via Roma 151, 07100 Sassari (SS), Italy
davide.bilo@uniss.it
https://orcid.org/0000-0003-3169-4300

Abstract

Reoptimization is a setting in which we are given an (near) optimal solution of a problem instance and a local modification that slightly changes the instance. The main goal is that of finding an (near) optimal 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 decade. 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.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Steiner tree problem, reoptimization, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.19

Related Version A full version of the paper is available at `https://arxiv.org/abs/1804.10791`.

1 Introduction

Reoptimization is a setting in which we are given an instance I of an optimization problem together with an (near) optimal solution S for I and a local modification that, once applied to I, generates a new instance I' which slightly differs from I. The main goal is that of finding an (near) optimal solution of I'. Since reoptimization problems defined on top of NP-hard problems are usually NP-hard, the idea beyond reoptimization is to compute a good approximate solution of I' by exploiting the structural properties of S. This approach leads to the design of reoptimization algorithms that, when compared to classical algorithms that compute feasible solutions from scratch, are less expensive in terms of running time or output solutions which guarantee a better approximation ratio. Thus, reoptimization finds applications in many real 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 previous 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 [33], 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 including: the Steiner tree problem [8, 11, 13, 14, 15, 29, 30, 35], the traveling salesman problem and some of its variants [1, 5, 7, 12, 15, 16, 17, 18, 19, 32],

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 19; pp. 19:1–19:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

scheduling problems [6, 21], the minimum latency problem [27], 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 reoptimization can be found in [4, 25, 34].[1]

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), a problem that given an edge-weighted graph as input, where each vertex can be either terminal or Steiner, asks to find a minimum-cost tree spanning all the terminals. More precisely, we study the four local modifications in which the cost of an edge changes (it can either increase or decrease) or a vertex changes its status (from Steiner to terminal and viceversa).[2] All the problems addressed are strongly NP-hard [14, 15], which implies that none of them admits a fully polynomial-time approximation scheme, unless $\mathsf{P} = \mathsf{NP}$.

In this paper we improve upon all the constant-factor approximation algorithms that have been developed in the last 10 years by designing *polynomial-time approximation schemes*. More precisely, if S is a ρ-approximate solution of I, then all our algorithms compute a $(\rho + \epsilon)$-approximate solution of I'. For the scenarios in which the cost of an edge decreases or a terminal vertex becomes Steiner, our polynomial-time approximation schemes hold under the assumption that $\rho = 1$, i.e., the provided solution is optimal. We observe that this assumption is somehow necessary since Goyal and Mömke (see [30]) proved that in both scenarios, unless $\rho = 1$, any σ-approximation algorithm for the reoptimization problem would be also a σ-approximation algorithm for the *STP*! 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.

The 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 [30], the scenario in which a vertex is removed from the graph is as hard to approximate as the *STP* (think of the removal of a vertex that is connected to each terminal by an edge of cost 0).

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 computes a Steiner tree in polynomial time by augmenting a Steiner forest having a constant number of trees with a minimum-cost set of edges. This algorithm has been introduced by Böckenhauer et al. in [14] and has been subsequently used in several other papers on Steiner tree reoptimization [11, 13, 30, 35];
- the algorithmic proof of Borchers and Du [20] that, for every $\xi > 0$, converts a Steiner tree S into a $f(\xi)$-*restricted Steiner tree* S_ξ, for some function $f(\xi)$ that depends on ξ only, whose cost is 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 [30];

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

[2] We observe that the insertion of an edge can be modeled by decreasing the cost of the edge from $+\infty$ to its real value. Analogously, the deletion of an edge can be modeled by increasing the cost of the edge to $+\infty$.

Table 1 The state of the art of the approximability of Steiner tree reoptimization before and after our paper. The value σ is the best approximation ratio we can achieve for *STP* (actually $\sigma = \ln 4 + \epsilon \approx 1.387$ [26]). If we substitute $\sigma = \ln 4 + \epsilon$, we get $\frac{10\sigma-7}{7\sigma-4} \approx 1.204$, $\frac{7\sigma-4}{4\sigma-1} \approx 1.256$, and $\frac{5\sigma-3}{3\sigma-1} \approx 1.246$. 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. To increase readability, all the bounds are provided for the case in which $\rho = 1$. 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*. Finally, the question mark means that the problem is still open.

Local modification	**before our paper**	**after our paper**
a terminal becomes a Steiner vertex*	$\frac{10\sigma-7}{7\sigma-4} + \epsilon$ [30]	$1 + \epsilon$
a Steiner vertex becomes terminal	$\frac{10\sigma-7}{7\sigma-4} + \epsilon$ [30]	$1 + \epsilon$
an edge cost increases	$\frac{7\sigma-4}{4\sigma-1} + \epsilon$ [30]	$1 + \epsilon$
an edge cost decreases*	$\frac{5\sigma-3}{3\sigma-1}$ [8]	$1 + \epsilon$
a vertex is added to the graph*	?	?
a vertex is deleted from the graph	as hard as the *STP* [30]	as hard as the *STP* [30]

- a novel algorithm, that we call BuildST, that takes a $f(\xi)$-restricted Steiner forest S_ξ of S with q trees, generated with the algorithm of Borchers and Du, and a positive integer h as inputs and computes a minimum-cost Steiner tree S' w.r.t. the set of all the feasible solutions that can be obtained by swapping up to h *full components* of S_ξ with a minimum-cost set of edges (that is computed using the algorithm Connect). The running time of this algorithm is polynomial when all the three parameters $f(\xi), q$, and h are bounded by a constant.

We prove that the approximation ratio of the solution S' returned by the algorithm BuildST is $\rho+\epsilon$ (i) by showing that the cost of S' is at most the cost of other feasible solutions $S_1, \ldots, S_\ell$ and (ii) by proving useful upper bounds to the cost of each S_i's. Intuitively, each S_i is obtained by swapping up to h suitable full components, say H_i, of a suitable $f(\xi)$-restricted version of a forest in S, say S_ξ, with a minimal set of full components, say H'_i, of a $g(\xi)$-restricted version of a fixed optimal solution OPT'. This implies that we can easily bound the cost of each S_i w.r.t. the cost of S_ξ, the cost of H_i, and the cost of H'_i. Unfortunately, none of these bounds can be used by itself to prove the claim. We address this issue by carefully defining 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. This trick allows us to derive better upper bounds: we prove that the cost of each S_i is at most $\rho g(\xi)^2$ times the cost of OPT' plus the i-th term of a telescoping sum. As each term of the telescoping sum can be upper bounded by $g(\xi)$ times the cost of OPT', the bound of $(\rho + \epsilon)$ on the approximation ratio of S' then follows by averaging the costs of the S_i's and because of the choices of the parameters.

The paper is organized as follows: in Section 2 we provide the basic definitions and some preliminaries; in Section 3 we describe the three main tools that are used in all our algorithms; in Sections 4 and 5 we describe and analyze the algorithms for the cases in which a Steiner vertex becomes terminal and the cost of an edge increases, respectively. Due to the lack of space, the local modifications in which a terminal becomes a Steiner vertex and the cost of an edge decreases can be found in the full version of the paper.

2 Basic definitions and preliminaries

The Steiner tree problem (*STP* for short) is defined as follows:

- the input is a triple $I = \langle G, c, R\rangle$ where $G = (V(G), E(G))$ is a connected undirected graph with $|V(G)| = n$ vertices, 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 a set of *terminal* vertices;
- the problem asks to compute a minimum-cost Steiner tree of I, i.e., a tree that spans R and that minimizes the overall sum of its edge costs.

The vertices in $V(G) \setminus R$ are also called *Steiner* vertices. With a slight abuse of notation, for any subgraph H of G, we denote by $c(H) := \sum_{e\in E(H)} c(e)$ the cost of H.

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'.

For a forest F of G (i.e., with $E(F) \subseteq E(G)$) and a set of edges $E' \subseteq E(G)$, we denote by $F + E'$ a (fixed) 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 a Steiner tree are terminals: 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 whose cost is upper bounded by the cost of 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 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^s_n the complete graph obtained by adding $n - 1$ copies of K^s to K^s and where the cost of an edge between a vertex and any of its copies is equal to 0. With a little abuse of notation, we use d^s to denote the cost function of K^s_n. Moreover, we say that a subgraph H of K^s_n *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^s_n the *STP* instance $I^s_n = \langle K^s_n, d^s, R^s\rangle$.

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 no 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 tool 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 reduces the *STP* instance to a smaller *STP* instance, where each terminal vertex corresponds to a tree of F, and then uses the famous Dreyfus-Wagner algorithm (see [28]) to find an optimal solution of the reduced instance.[3] If F contains q trees, the running time of the algorithm Connect is $O^*(3^q)$;[4] this implies that the algorithm has polynomial running time when q is constant. 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 in the recent paper of Goyal and Mömke [30]. 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 $d(S_\xi) \leq (1+\xi)c(OPT)$. 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 real 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 $d(S_\xi) \leq (1+\xi)c(S)$.*

The call of Borchers and Du algorithm with input parameters I, S, and ξ, is denoted by RestrictedST(I, S, ξ). 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 want that a specific vertex of S would also be a vertex of the k-restricted Steiner tree S_ξ of I_n.

[3] We refer to Hougardy [31] for further exact algorithms for the *STP*.

[4] The O^* notation hides polynomial factors.

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, ∅));
2 S' ←⊥;
3 for every set H of up to h full components of F do
4 |   F̄ ← F − H;
5 |   S̄ ← CONNECT(I_n, F̄);
6 |   if S' =⊥ or c(S̄) < c(S') then S' ← S̄;
7 return S';
```

▶ **Corollary 2.** *Let I be an STP instance, S a Steiner tree of I, v a vertex of S, and $\xi > 0$ a real 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 $v \in R$ has already been proved in 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 RESTRICTEDST(I', S, ξ) 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 $k = 4k' = 2^{2+\lceil 1/\xi \rceil}$. ◀

The call of Borchers and Du algorithm with parameters I, S, ξ, and v is denoted by RESTRICTEDST(I, S, ξ, v).

The third 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 up to h of its full components. The Steiner tree of I associated with $\bar{F}$ is obtained by the call of CONNECT$(I_n, \bar{F})$. If F contains a number of full components which is strictly less than h, 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 BUILDST(I, F, h). The proof of the following lemma is immediate.

▶ **Lemma 3.** *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* BUILDST(I, F, h) *is at most $c(\bar{F}) + c(H)$.*

Moreover, the following lemmas hold.

▶ **Lemma 4.** *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. ◀

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

```
1 ξ ← ε/10;
2 h ← 2^{2⌈2/ε⌉⌈1/ξ⌉};
3 S_ξ ← RestrictedST(I, S, ξ);
4 S' ← BuildST(I', S_ξ + t, h);
5 return S';
```

▶ **Lemma 5.** *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 $O^*(|R|^h 3^{q+hk})$.*

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 $O(|R|^{h+1})$. 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 $O^*(|R|^h 3^{q+hk})$. ◀

For the rest of the paper, we denote by $\langle I, S, I' \rangle$ an instance of the Steiner tree reoptimization problem. Furthermore, we also assume that $\rho \leq 2$ as well as $\epsilon \leq 1$, 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 $\xi = \epsilon/10$ and $h = 2^{2\lceil 2/\epsilon \rceil \lceil 1/\xi \rceil}$. The algorithm, whose pseudocode is reported in Algorithm 2, computes a Steiner tree of I' first by calling RestrictedST(I, S, ξ) to obtain a k-restricted Steiner tree S_ξ of I_n, with $k = 2^{\lceil 1/\xi \rceil}$, and then by calling 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'.

▶ **Theorem 6.** *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 $(\rho + \epsilon)$-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 5 and the fact that h is a constant value, the call of 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 $(\rho + \epsilon)$-approximate solution. Let S'_ξ denote the Steiner tree of I'_n that is returned by the call of RestrictedST(I', OPT', ξ). Let

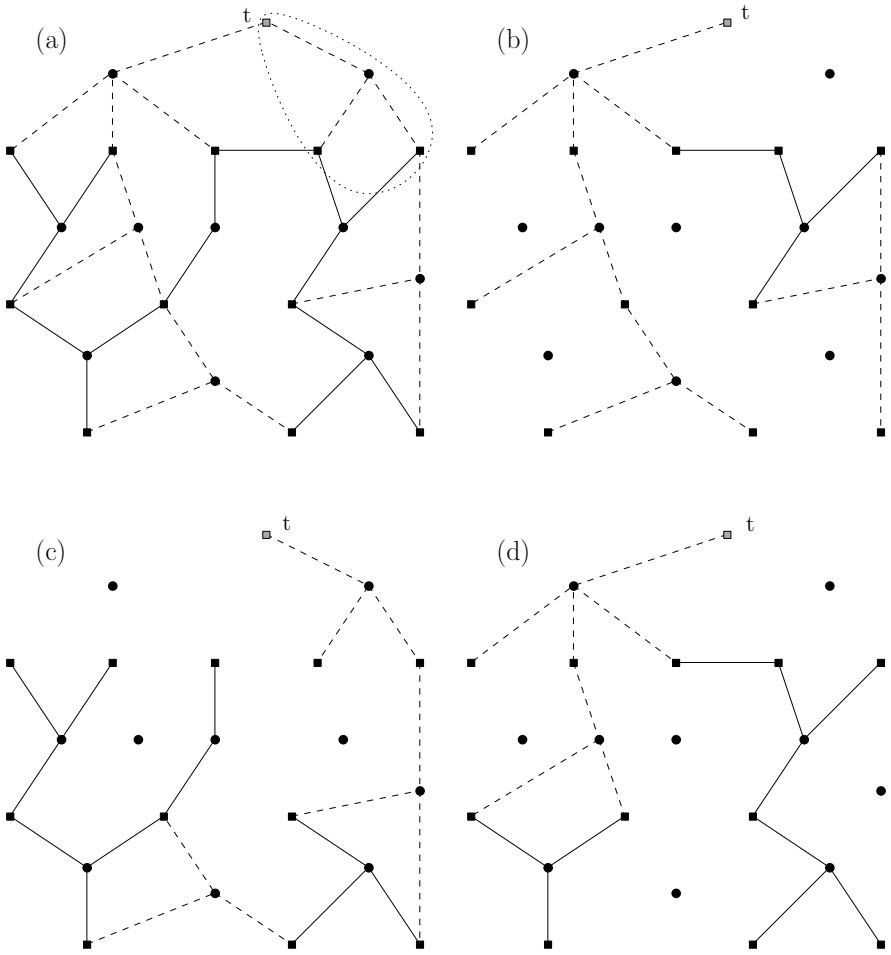

Figure 1 An example that shows how H_i and H'_i are defined. In (a) it is depicted a Steiner tree reoptimization instance where square vertices represent the terminals, while circle vertices represent the Steiner vertices. The Steiner vertex t of I that becomes a terminal in I' is represented as a terminal 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).

H'_0 be a full component of S'_ξ that spans t (ties are broken arbitrarily). For every $i = 1, \ldots, \ell$, with $\ell = \lceil 2/\epsilon \rceil$, 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 Figure 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 Figure 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 4, H_i contains at most k^{2i-1} full components and spans 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 $i = 1, \ldots, \ell$, is at most

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

Therefore, by Lemma 3, the cost of the solution returned by the call of $\textsc{BuildST}(I', S_\xi + t, h)$ 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

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

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

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

Observe that the overall sum of the last two terms on the right-hand side of (4) for every $i \in \{1, \ldots, \ell\}$ is a telescoping sum. As a consequence, if for every $i \in \{1, \ldots, \ell\}$ 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

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

where last inequality holds by the choice of ξ and ℓ, and because $\rho \leq 2$ and $\epsilon \leq 1$. 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.

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

$\xi \leftarrow \epsilon/10$;
$h \leftarrow 2^{2(1+\lceil 1/\xi \rceil)\lceil 2/\epsilon \rceil}$;
if $e \notin E(S)$ **then**
  $S' \leftarrow S$;
else
  let S_u and S_v be the tree of $S - e$ containing u and v, respectively;
  $I_u \leftarrow \langle G - e, c_{-e}, V(S_u) \cap R \rangle$;
  $I_v \leftarrow \langle G - e, c_{-e}, V(S_v) \cap R \rangle$;
  $S_{u,\xi} \leftarrow \textsc{RestrictedST}(I_u, S_u, \xi, u)$;
  $S_{v,\xi} \leftarrow \textsc{RestrictedST}(I_v, S_v, \xi, v)$;
  $S_\xi \leftarrow S_{u,\xi} + S_{v,\xi}$;
  $S' \leftarrow \textsc{BuildST}(I', S_\xi, h)$;
  if $c'(S) \leq c'(S')$ **then** $S' \leftarrow S$;
return S';

Let $\xi = \epsilon/10$ and $h = 2^{2(1+\lceil 1/\xi \rceil)\lceil 2/\epsilon \rceil}$. 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 $\textsc{BuildST}(I', S_{u,\xi} + S_{v,\xi}, h)$, 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'.

▶ **Theorem 7.** *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 $(\rho + \epsilon)$-approximate Steiner tree of I' in polynomial time.*

Proof. Corollary 2 implies that the computation of both $S_{u,\xi}$ and $S_{v,\xi}$ requires polynomial time. Therefore, using Lemma 5 and the fact that h is a constant value, the call of $\textsc{BuildST}(I', S_\xi, h)$ 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 $(\rho+\epsilon)$-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 $\textsc{RestrictedST}(\langle G - e, c_{-e}, R \rangle, OPT', \xi)$. 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 $i = 1, \ldots, \ell$, with $\ell = \lceil 2/\epsilon \rceil$, 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 2), 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 4, 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 $i = 1, \ldots, \ell$, is at most

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

Therefore, by Lemma 3, the cost of the solution returned by the call of BuildST(I', S_ξ, 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 $i \in \{1, \ldots, \ell\}$. Let $\Delta_{OPT} = (1+\xi)c'(OPT') - c(OPT)$ and observe that $\Delta_{OPT} \geq 0$. Since the addition 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 2 twice in the first inequality that follows we have that

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

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 $i \in \{1, \ldots, \ell\}$, with

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

[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 $v \in S_{v,\xi}$.

As a consequence, if for every $i \in \{1, \ldots, \ell\}$ 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

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

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

References

1 Claudia Archetti, Luca Bertazzi, and Maria Grazia Speranza. Reoptimizing the traveling salesman problem. *Networks*, 42(3):154–159, 2003. doi:10.1002/net.10091.

2 Claudia Archetti, Luca Bertazzi, and Maria Grazia Speranza. Reoptimizing the 0-1 knapsack problem. *Discrete Applied Mathematics*, 158(17):1879–1887, 2010. doi:10.1016/j.dam.2010.08.003.

3 Claudia Archetti, Gianfranco Guastaroba, and Maria Grazia Speranza. Reoptimizing the rural postman problem. *Computers & OR*, 40(5):1306–1313, 2013. doi:10.1016/j.cor.2012.12.010.

4 Giorgio Ausiello, Vincenzo Bonifaci, and Bruno Escoffier. *Complexity and Approximation in Reoptimization*, pages 101–129. Imperial College Press, 2012.

5 Giorgio Ausiello, Bruno Escoffier, Jérôme Monnot, and Vangelis Th. Paschos. Reoptimization of minimum and maximum traveling salesman's tours. *J. Discrete Algorithms*, 7(4):453–463, 2009. doi:10.1016/j.jda.2008.12.001.

6 Michael A. Bender, Martin Farach-Colton, Sándor P. Fekete, Jeremy T. Fineman, and Seth Gilbert. Reallocation problems in scheduling. *Algorithmica*, 73(2):389–409, 2015. doi:10.1007/s00453-014-9930-4.

7 Tobias Berg and Harald Hempel. Reoptimization of traveling salesperson problems: Changing single edge-weights. In Adrian-Horia Dediu, Armand-Mihai Ionescu, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications, Third International Conference, LATA 2009, Tarragona, Spain, April 2-8, 2009. Proceedings*, volume 5457 of *Lecture Notes in Computer Science*, pages 141–151. Springer, 2009. doi:10.1007/978-3-642-00982-2_12.

8 Davide Bilò, Hans-Joachim Böckenhauer, Juraj Hromkovic, Richard Královic, Tobias Mömke, Peter Widmayer, and Anna Zych. Reoptimization of Steiner trees. In Joachim Gudmundsson, editor, *Algorithm Theory - SWAT 2008, 11th Scandinavian Workshop on Algorithm Theory, Gothenburg, Sweden, July 2-4, 2008, Proceedings*, volume 5124 of *Lecture Notes in Computer Science*, pages 258–269. Springer, 2008. doi:10.1007/978-3-540-69903-3_24.

9 Davide Bilò, Hans-Joachim Böckenhauer, Dennis Komm, Richard Královic, Tobias Mömke, Sebastian Seibert, and Anna Zych. Reoptimization of the shortest common superstring problem. *Algorithmica*, 61(2):227–251, 2011. doi:10.1007/s00453-010-9419-8.

10 Davide Bilò, Peter Widmayer, and Anna Zych. Reoptimization of weighted graph and covering problems. In Evripidis Bampis and Martin Skutella, editors, *Approximation and Online Algorithms, 6th International Workshop, WAOA 2008, Karlsruhe, Germany, September 18-19, 2008. Revised Papers*, volume 5426 of *Lecture Notes in Computer Science*, pages 201–213. Springer, 2008. doi:10.1007/978-3-540-93980-1_16.

11 Davide Bilò and Anna Zych. New advances in reoptimizing the minimum Steiner tree problem. In Branislav Rovan, Vladimiro Sassone, and Peter Widmayer, editors, *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012,*

Bratislava, Slovakia, August 27-31, 2012. Proceedings, volume 7464 of *Lecture Notes in Computer Science*, pages 184–197. Springer, 2012. doi:10.1007/978-3-642-32589-2_19.

12 Hans-Joachim Böckenhauer, Luca Forlizzi, Juraj Hromkovic, Joachim Kneis, Joachim Kupke, Guido Proietti, and Peter Widmayer. On the approximability of TSP on local modifications of optimally solved instances. *Algorithmic Operations Research*, 2(2):83–93, 2007. URL: http://journals.hil.unb.ca/index.php/AOR/article/view/2803.

13 Hans-Joachim Böckenhauer, Karin Freiermuth, Juraj Hromkovic, Tobias Mömke, Andreas Sprock, and Björn Steffen. Steiner tree reoptimization in graphs with sharpened triangle inequality. *J. Discrete Algorithms*, 11:73–86, 2012. doi:10.1016/j.jda.2011.03.014.

14 Hans-Joachim Böckenhauer, Juraj Hromkovic, Richard Královic, Tobias Mömke, and Peter Rossmanith. Reoptimization of Steiner trees: Changing the terminal set. *Theor. Comput. Sci.*, 410(36):3428–3435, 2009. doi:10.1016/j.tcs.2008.04.039.

15 Hans-Joachim Böckenhauer, Juraj Hromkovic, Tobias Mömke, and Peter Widmayer. On the hardness of reoptimization. In Viliam Geffert, Juhani Karhumäki, Alberto Bertoni, Bart Preneel, Pavol Návrat, and Mária Bieliková, editors, *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*, volume 4910 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2008. doi:10.1007/978-3-540-77566-9_5.

16 Hans-Joachim Böckenhauer, Juraj Hromkovic, and Andreas Sprock. Knowing all optimal solutions does not help for TSP reoptimization. In Jozef Kelemen and Alica Kelemenová, editors, *Computation, Cooperation, and Life - Essays Dedicated to Gheorghe Paun on the Occasion of His 60th Birthday*, volume 6610 of *Lecture Notes in Computer Science*, pages 7–15. Springer, 2011. doi:10.1007/978-3-642-20000-7_2.

17 Hans-Joachim Böckenhauer, Juraj Hromkovic, and Andreas Sprock. On the hardness of reoptimization with multiple given solutions. *Fundam. Inform.*, 110(1-4):59–76, 2011. doi:10.3233/FI-2011-528.

18 Hans-Joachim Böckenhauer, Joachim Kneis, and Joachim Kupke. Approximation hardness of deadline-TSP reoptimization. *Theor. Comput. Sci.*, 410(21-23):2241–2249, 2009. doi:10.1016/j.tcs.2009.02.016.

19 Hans-Joachim Böckenhauer and Dennis Komm. Reoptimization of the metric deadline TSP. *J. Discrete Algorithms*, 8(1):87–100, 2010. doi:10.1016/j.jda.2009.04.001.

20 Al Borchers and Ding-Zhu Du. The k-Steiner ratio in graphs. *SIAM J. Comput.*, 26(3):857–869, 1997. doi:10.1137/S0097539795281086.

21 Nicolas Boria and Federico Della Croce. Reoptimization in machine scheduling. *Theor. Comput. Sci.*, 540:13–26, 2014. doi:10.1016/j.tcs.2014.04.004.

22 Nicolas Boria, Jérôme Monnot, and Vangelis Th. Paschos. Reoptimization of maximum weight induced hereditary subgraph problems. *Theor. Comput. Sci.*, 514:61–74, 2013. doi:10.1016/j.tcs.2012.10.037.

23 Nicolas Boria, Jérôme Monnot, and Vangelis Th. Paschos. Reoptimization under vertex insertion: Max p_k-free subgraph and max planar subgraph. *Discrete Math., Alg. and Appl.*, 5(2), 2013. doi:10.1142/S1793830913600045.

24 Nicolas Boria and Vangelis Th. Paschos. Fast reoptimization for the minimum spanning tree problem. *J. Discrete Algorithms*, 8(3):296–310, 2010. doi:10.1016/j.jda.2009.07.002.

25 Nicolas Boria and Vangelis Th. Paschos. A survey on combinatorial optimization in dynamic environments. *RAIRO - Operations Research*, 45(3):241–294, 2011. doi:10.1051/ro/2011114.

26 Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013. doi:10.1145/2432622.2432628.

27 Wenkai Dai. Reoptimization of minimum latency problem. In Yixin Cao and Jianer Chen, editors, *Computing and Combinatorics - 23rd International Conference, COCOON 2017, Hong Kong, China, August 3-5, 2017, Proceedings*, volume 10392 of *Lecture Notes in Computer Science*, pages 162–174. Springer, 2017. doi:10.1007/978-3-319-62389-4_14.

28 S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/net.3230010302.

29 Bruno Escoffier, Martin Milanic, and Vangelis Th. Paschos. Simple and fast reoptimizations for the Steiner tree problem. *Algorithmic Operations Research*, 4(2):86–94, 2009. URL: http://journals.hil.unb.ca/index.php/AOR/article/view/5653.

30 Keshav Goyal and Tobias Mömke. Robust reoptimization of Steiner trees. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 10–24. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.10.

31 Stefan Hougardy, Jannik Silvanus, and Jens Vygen. Dijkstra meets Steiner: a fast exact goal-oriented Steiner tree algorithm. *Math. Program. Comput.*, 9(2):135–202, 2017. doi:10.1007/s12532-016-0110-1.

32 Jérôme Monnot. A note on the traveling salesman reoptimization problem under vertex insertion. *Inf. Process. Lett.*, 115(3):435–438, 2015. doi:10.1016/j.ipl.2014.11.003.

33 Markus W. Schäffter. Scheduling with forbidden sets. *Discrete Applied Mathematics*, 72(1-2):155–166, 1997. doi:10.1016/S0166-218X(96)00042-X.

34 Anna Zych. *Reoptimization of NP-hard problems.* PhD thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20257, 2012.

35 Anna Zych and Davide Bilò. New reoptimization techniques applied to Steiner tree problem. *Electronic Notes in Discrete Mathematics*, 37:387–392, 2011. doi:10.1016/j.endm.2011.05.066.