New Advances in Reoptimizing the Minimum Steiner Tree Problem

Davide Bilò[1] and Anna Zych[2]

[1] Dip.to di Teorie e Ricerche dei Sistemi Culturali, University of Sassari, Italy
[2] Uniwersytet Warszawski
davide.bilo@uniss.it

Abstract. In this paper we improve the results in the literature concerning the problem of computing the minimum Steiner tree given the minimum Steiner tree for a similar problem instance. Using a σ-approximation algorithm computing the minimum Steiner tree from scratch, we provide a $\left(\frac{3\sigma-1}{2\sigma-1}+\epsilon\right)$ and a $\left(\frac{2\sigma-1}{\sigma}+\epsilon\right)$ -approximation algorithm for altering the instance by removing a vertex from the terminal set and by increasing the cost of an edge, respectively. If we use the best up to date $\sigma = \ln 4+\epsilon$, our ratios equal 1.218 and 1.279 respectively.

1 Introduction

The concept of reoptimization adds an interesting twist to hard optimization problems. It is motivated by the fact, that when confronted with a hard problem in reality, one often has some additional information about an instance at hand. Imagine for example a train station where the train traffic is regulated via a certain time schedule. The schedule is computed when the station is ready to operate and, provided that an unexpected event does not occur, there is no need to compute it again. Unfortunately an unexpected event, such as a delayed train, is in a long run inevitable. Reoptimization addresses this scenario, asking whether knowing a solution for a certain problem instance is beneficial when computing a solution for a similar instance. When dealing with relatively stable environments, i. e., where changing conditions alter the environment only for a short period of time, it seems reasonable to spend even a tremendous amount of time on computing a good solution for the undisturbed instance, and profit from it when confronted with a temporal change.

The term reoptimization was mentioned for the first time in [20] and applied to the problem of scheduling with forbidden sets for the scenarios of adding or removing a forbidden set. Since then, the concept of reoptimization has been successfully applied to various problems like the Traveling Salesman problem [3,1,8,11], the Steiner Tree problem [22,5,9,13,16], the Knapsack problem [2], the Weighted Minimum Set Cover problem [17], various covering problems [7], and the Shortest Common Superstring problem [6]. A survey of reoptimization problems can be found in [10,15].

The Minimum Steiner Tree (*SMT*) problem is a very prominent optimization problem with many practical applications, especially in network design. It is

APX-complete [4]. The best up to date approximation ratio is due to a randomized $\ln 4 + \epsilon$-approximation algorithm [12]. The best deterministic algorithm achieves an approximation ratio of 1.55 [19]. The problem of reoptimizing *SMT* where the modification of an instance consists of adding or removing one vertex to/from the input graph was considered in [14]. The modifications of adding or removing a vertex to/from the terminal set and increasing or decreasing the cost of an edge was addressed in [9,5]. All these reoptimization problems are strongly NP-hard [9].

We improve in this paper the best up to date results for *SMT* under removing a terminal from the terminal set and for *SMT* under increasing the cost of an edge, complementing the new results in [22] for *SMT* under adding a terminal to the terminal set. We achieve a 1.218 approximation ratio for the scenario of removing a terminal and 1.279 for the scenario of edge cost increase. For the terminal addition and edge cost decrease the best ratios remain 1.218 [22] and 1.246 [5] respectively.[1] It is worth to remark that, contrary to [5], for the modification of decreasing the cost of an edge we do not assume that the modified graph remains metric.

For both reoptimization scenarios considered in this paper we employ the technique developed in [22]. Nevertheless, for both scenarios, the technique cannot be applied directly as for the scenario of adding a terminal to the terminal set. The heart of the paper is Lemma 3, which settles an important property that allows successfully applying the technique.

2 Preliminaries

Let us begin with a formal definition of the Minimum Steiner Tree Problem (*SMT*). We call a complete graph $G = (V, E, \text{cost})$ with the cost function $\text{cost} : E \to \mathbb{Q}^+$ *metric* if the edge weights satisfy the *triangle inequality*, i.e, $\text{cost}(\{u, v\}) \leq \text{cost}(\{u, w\}) + \text{cost}(\{w, v\})$ for all $u, v, w \in V$.

Definition 1. ***Minimum Steiner Tree Problem (SMT)***

Instance: *a metric graph* $G = (V, E, \text{cost})$ *and a terminal set* $S \subseteq V$*;*
Solution: *a Steiner tree, i.e., a subtree of* G *containing* S*;*
Objective: *minimize the sum of the costs of the edges in the solution.*

We view an algorithm as a function from the instance space to the solution space. Some of our results use an approximation algorithm for *SMT* as a subroutine and refer to it as ApprSMT. Any of the approximation algorithms for *SMT* can be used as ApprSMT.

1. ***The Minimum Steiner Tree Terminal Removal problem***
   Instance: a metric graph $G = (V, E, \text{cost})$, a terminal set S, an optimal solution Opt to (G, S), and a modified terminal set $S' = S \setminus \{t\}$ for some $t \in S$;
   Solution: a Steiner tree for (G, S').

[1] In [5] the authors provide a $\frac{5\sigma-3}{3\sigma-1}$-approximation algorithm, where σ is the approximation ratio of an algorithm for the Steiner tree problem.

2. ***The minimum Steiner Tree Edge Cost Increase problem***
   Instance: a metric graph $G = (V, E, \text{cost})$, a terminal set $S \subseteq V$, an optimal solution Opt to (G, S), and a modified graph $G' = (V, E, \text{cost}')$, where cost$'$ coincides with cost on all edges but one edge e^* and $\text{cost}(e^*) < \text{cost}'(e^*)$;
   Solution: a Steiner tree for (G', S).

We introduce the notation used in the paper. Given a simple graph G, we denote the set of its vertices by $V(G)$ and the set of its edges by $E(G)$. We denote an (undirected) edge $e \in E(G)$ by $\{u, v\}$, where $u, v \in V(G)$ are the endpoints of e. If H is a subgraph of G, we write $H \subseteq G$. An edge $\{u, v\} \in E(G)$ can be identified with a subgraph $H \subseteq G$ with $V(H) = \{u, v\}$ and $E(H) = \{\{u, v\}\}$. The neighborhood $\Gamma_G(v)$ of a vertex $v \in V(G)$ in a graph G is the set of vertices adjacent to v in $E(G)$. The degree of a vertex $v \in V(G)$ is defined as the size of its neighborhood: $deg_G(v) = |\Gamma_G(v)|$. For two subgraphs $G_1, G_2 \subseteq G$ we denote by $G_1 \cup G_2$ a graph G' such that $V(G') = V(G_1) \cup V(G_2)$ and $E(G') = E(G_1) \cup E(G_2)$. For two subgraphs $G_1, G_2 \subseteq G$ we denote by $G_1 - G_2$ a graph G' such that $E(G') = E(G_1) \setminus E(G_2)$ and $V(G')$ is $V(G_1)$ without isolated vertices. To analyze our algorithms, we often refer to costs of graphs rather than edges. The cost of graph $G_1 \subseteq G$ is denoted by $\text{cost}(G_1)$ and is the sum of the costs of all edges in $E(G_1)$.

A path $P = (V(P), E(P))$ is a connected graph in which two vertices have degree one (called the endpoints of the path) and the remaining ones have degree two. The length of a path is its number of edges. In a shortest path, the length of the path is minimized whereas in a cheapest path its cost is minimized. A forest is a graph that consists of a union of disjoint trees: $F = T_1 \cup \cdots \cup T_m$. We denote the number of trees in F as $|F|_T$.

For a graph $G = (V, E, \text{cost})$ with an arbitrary edge weight function cost : $E \to \mathbb{Q}^+$, we denote by $\overline{G} = (V, \{\{u, v\} \mid u, v \in V, u \neq v\}, \overline{\text{cost}})$ the *metric closure* of G, i.e., the complete graph on $V(G)$ where $\overline{\text{cost}}(\{u, v\})$ is defined as the cost of the cheapest path in G from u to v. Observe that $\overline{G}$ is metric. The optimal Steiner tree for an arbitrarily edge weighted graph G and for its metric closure $\overline{G}$ coincide due to Lemma 1, whose proof can be found in [18].

Lemma 1. *Let $G = (V, E, \text{cost})$ be an edge-weighted graph and $S \subseteq V$ be a set of terminals. If a tree T is a minimum Steiner tree for (G, S) then it is also a minimum Steiner tree for $(\overline{G}, S)$. Moreover, any Steiner tree T' for $(\overline{G}, S)$ can be easily transformed to a Steiner tree for (G, S) of cost less than or equal to the cost of T'.*

The lemma above implies that for both *SMT* problem and *SMT* under removing a vertex from the terminal set the restriction on the input graph to be metric and complete does not cause any loss of generality. A similar restriction for the scenario of edge cost increase is discussed in Section 4.

Observation 1. *Due to the metricity and completeness of the graph we may assume without loss of generality that $deg_{\text{Opt}}(v) \geq 3$ for an optimal solution* Opt *to any SMT problem instance (G, S) and any $v \in V(\text{Opt}) \setminus S$.*

3 The *SMT* Terminal Removal Problem

Before we move on to the *SMT* Terminal Removal problem, we introduce a few techniques and observations that will be used later. We start with the technique introduced in [21], which relies on executing an approximation algorithm for *SMT* on a reduced instance and expanding the obtained solution. The technique is based on the procedure $\mathcal{S}_{\text{ApprSMT}}$ presented in Algorithm 1. This procedure, for a given instance (G, S) of *SMT* and a forest F of trees contained in G, contracts the edges of F. Routine Δ, given an *SMT* instance and an edge, computes another, reduced instance of *SMT*, where the edge is contracted to a single node which becomes a terminal in the reduced instance. Additionally, if the contraction creates multiple edges, the cheapest one is chosen for each pair of vertices (in other words, the metric closure is computed). Then a σ-approximation algorithm ApprSMT for *SMT* is applied on the obtained reduced instance. After adding the contracted edges to the resulting solution, the modified solution becomes feasible for the original instance.

Algorithm 1. $\mathcal{S}_{\text{ApprSMT}}$

```
Input: A metric graph $G$, a terminal set $S \subseteq V(G)$, a forest $F = T_1 \cup \cdots \cup T_k$
1: $(G_0, S_0) := (G, S)$
2: for all edges $e$ in $E(F)$ do
3:     $(G_j, S_j) := \Delta((G_{j-1}, S_{j-1}), e_j)$
4: end for
5: $T :=$ ApprSMT$(\overline{G_k}, S_k)$
Output: $T \cup F$
```

Lemma 2. *If $F \subseteq \text{Opt}(G, S)$ then the algorithm $\mathcal{S}_{\text{ApprSMT}}((G, S), F)$ returns a solution of cost at most $\sigma\text{Opt}(G, S) - (\sigma - 1)\text{cost}(F)$ where σ is the approximation ratio of* ApprSMT. *Proof to be found in [5,21].*

The next remark states that it is enough to consider the case when the removed terminal t has at least two neighbors in Opt.

Remark 1. Let Opt be the optimal solution to (G, S) given in the input of the *SMT* under removing a vertex from the terminal set. Let $t \in S$ be the terminal removed from S, i.e., $S' = S \setminus \{t\}$. We may assume that $deg_{\text{Opt}}(t) \geq 2$.

Proof. If there is only one edge $e = \{t, v\}$ adjacent to t in Opt, then Opt $- e$ is an optimal solution to $(G, S \setminus \{t\} \cup \{v\})$. If $v \in S'$ then Opt $- e$ is optimal to (G, S'). Otherwise we build a new instance for the *SMT* under removing a vertex from the terminal set. We let the non-modified instance to be $(G, S \setminus \{t\} \cup \{v\})$ with the optimal solution Opt $- e$. Then we let the terminal to be removed be v, so the modified set of terminals is $S \setminus \{t\}$ which is equal to S'. For the instance we have it holds that $deg_{\text{Opt}-e}(v) \geq 2$. □

Below we introduce a lemma that settles an important property of $\textsc{Opt}'$. The lemma holds for any optimal Steiner tree, but we formulate it using $\textsc{Opt}'$, because we want to apply it to $\textsc{Opt}'$. We define a binary tree as a tree where there is just one vertex of degree 2 called root, and every other vertex has either degree 1 (a leaf) or 3 (an inner vertex). The set of leaves of a tree T is denoted by $\mathrm{Leaf}(T)$ and the set of inner vertices by $\mathrm{Inner}(T)$.

Definition 2. *Let* $\textsc{Opt}'$ *be a minimum Steiner tree for an SMT problem instance* (G, S'). *A binary subtree* $B \subseteq \textsc{Opt}'$ *of* $\textsc{Opt}'$ *is* A*-appropriate for a set* $A \subseteq S'$ *if and only if* $\mathrm{Leaf}(B) \subseteq A$ *and* $\mathrm{Inner}(B) \cap S' = \emptyset$.

In Figure 1 we illustrate Definition 2 with an example of an optimal tree $\textsc{Opt}'$ for an *SMT* instance (G, S'), containing an A-appropriate subtree B for some $A \subseteq S'$. The next lemma shows that for any instance (G, S') of the *SMT* problem and any partition of the terminal set S' into $S' = S_1 \cup S_2$ (i. e., $S_2 = S' \setminus S_1$), an optimal solution $\textsc{Opt}'$ (satisfying Observation 1) contains a subtree $P^{(3)} \cup B_1 \cup B_2$. The subtree $P^{(3)} \cup B_1 \cup B_2$ we prove to exist consists of two binary trees B_1 and B_2, that are S_1 and S_2-appropriate respectively, and a path $P^{(3)}$ of at most three edges connecting these two trees. Figure 2 shows the desired subtree.

Lemma 3. *Let* $\textsc{Opt}'$ *be a minimum Steiner tree for* (G, S') *satisfying Observation 1. Let* $S' = S_1 \cup S_2$, *where* $S_2 = S' \setminus S_1$, *be any partition of a terminal set into two disjoint subsets. Then* $\textsc{Opt}'$ *contains a subtree* $P^{(3)} \cup B_1 \cup B_2$ *such that*

- $|P^{(3)}| \leq 3$,
- B_1 *is* S_1*-appropriate tree rooted at* r_1, *where* $r_1 \in \mathrm{Leaf}(P^{(3)})$,
- B_2 *is* S_2*-appropriate tree rooted at* r_2, *where* $r_2 \neq r_1$, $r_2 \in \mathrm{Leaf}(P^{(3)})$.

Proof. Let $(\overline{S_1}, \overline{S_2})$ be minimum size cut in $\textsc{Opt}'$ between S_1 and S_2, i.e., a partition of vertices $V(\textsc{Opt}') = \overline{S_1} \cup \overline{S_2}$ into two disjoint subsets $\overline{S_1} \cap \overline{S_2} = \emptyset$ such that $S_1 \subseteq \overline{S_1}$ and $S_2 \subseteq \overline{S_2}$ and the number of edges of $\textsc{Opt}'$ having one endpoint in $\overline{S_1}$ and the other in $\overline{S_2}$ is minimum. Let C be a bipartite graph consisting of edges crossing the cut and their endpoints. Hence, $C \subseteq \textsc{Opt}'$ is a (not necessarily

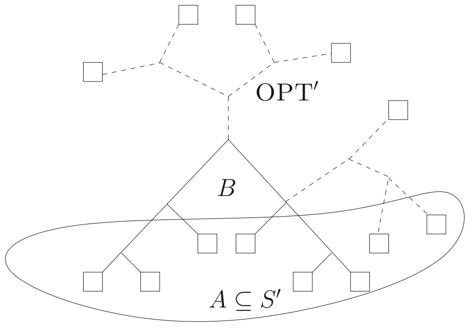

Fig. 1. An A-appropriate subtree $B \subseteq \textsc{Opt}'$, marked with solid line

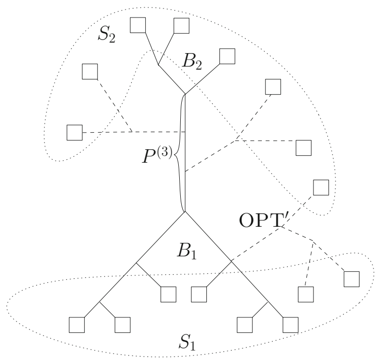

Fig. 2. A subtree $P^{(3)} \cup B_1 \cup B_2 \subseteq$ Opt′ proven to exist in Lemma 3

induced) subgraph of Opt′. Let $W_1 = V(C) \cap \overline{S_1}$ and $W_2 = V(C) \cap \overline{S_2}$ be the partition of vertices in C into two sets independent in C. Note that W_1 and W_2, although independent in C, do not have to be independent in Opt′. For each vertex $v \in W_j$, $j \in \{1, 2\}$, let $D(v)$ be the set of edges in Opt′ between v and the vertices of $\overline{S_j}$. Let $D(W_j) = \bigcup_{v \in W_j} D(v), j \in \{1, 2\}$. We illustrate the situation in Figure 3. We say that an edge $e \in D(W_j)$, $j \in \{1, 2\}$ is binary, if its endpoint not in W_j is a root of an S_j-appropriate subtree. For the sake of convenience, if $v \in W_j$ is a terminal, we let it have an imaginary binary edge in $D(v)$ incident only to v. We aim to prove that there are two binary edges, $f_1 \in D(W_1)$ and $f_2 \in D(W_2)$, both adjacent to the same edge $f_c \in C$.

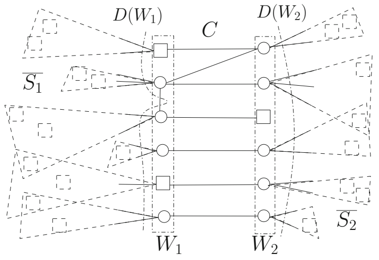

Fig. 3. Illustration to Lemma 3: Opt′ with its corresponding structures

Observe, that for each non-terminal vertex $v \in W_j$, $j \in \{1, 2\}$, it holds that $|D(v)| \geq 2$: if v is an endpoint of one edge in C, then it must have two adjacent edges in $D(v)$ as $deg_{\text{OPT}'}(v) \geq 3$; otherwise, due to the minimality of the cut C, vertex v must have at least as many adjacent edges outside the cut as inside the cut.

We now define an equivalence relation $\approx_j$, $j \in \{1, 2\}$ on the vertices of W_j. We say that $v \approx_j w$ for $v, w \in W_j$ if there is a path from v to w in Opt′ that contains only vertices in $\overline{S_j}$. We call such a path an $\overline{S_j}$-path. Hence, the vertices in W_j are partitioned into equivalence classes $[\cdot]_{\approx_j}$ with respect to relation $\approx_j$. As a first step, we prove that for each class $R \in [\cdot]_{\approx_j}$, the set of edges $D(R) = \bigcup_{v \in R} D(v)$ contains at least one binary edge.

First observe that, if R contains a terminal vertex, than we are done because of the imaginary binary edges. So assume that $R \cap S' = \emptyset$. Let us start with vertex $v_1 \in R$ and consider an edge $e_1 \in D(v_1)$. If e_1 is a binary edge, we are done. Otherwise, there is an $\overline{S_j}$-path starting with e_1 from v_1 to another vertex v_2 in R . We follow this path to v_2 and consider all the vertices visited. We enter vertex v_2 with edge $f_2 \in D(v_2)$. Now, since $|D(v_2)| \geq 2$, there is another edge $e_2 \in D(v_2)$. If e_2 is binary, we are done again. Otherwise, there is an $\overline{S_j}$-path starting with e_2 from v_2 to some $v_3 \in R$. We continue following the paths until either we found a binary edge, or we hit a vertex that we already visited. Since we always leave a vertex via a different edge as we entered, the latter implies a cycle in OPT' and hence a contradiction.

Now let $[\cdot]_{\approx_1} = \{\zeta_1, \ldots \zeta_l\}$ and $[\cdot]_{\approx_2} = \{\xi_1 \ldots \xi_m\}$ be the equivalence classes on W_1 and W_2 respectively. Note that there can be at most one edge in C between ζ_i and ξ_j, as otherwise OPT' contains a cycle. Recall that any vertex in W_1 is connected with an edge in C to a vertex in W_2 and vice-versa. We call a vertex binary if it is adjacent to a binary edge. We proved that every ζ_i and every ξ_j contains at least one binary vertex. Let $v \in \zeta_1$ be a binary vertex in ζ_1. It is adjacent via an edge $e \in C$ to some vertex $w \in \xi_j$. If w is binary, we let $f_c = e$ and the proof is completed. Otherwise, there is another vertex in ξ_j that is binary, call it w'. Vertex w' is connected to v' in ζ_i, $i \neq 1$. If v' is not binary, there must be another connection going from a binary vertex in ζ_i to another yet unvisited ξ_j. We can continue traversing the equivalence classes in this way until either we hit a binary vertex, or a class that was already visited. The latter is a contradiction, as it implies that there is a cycle in OPT'. □

Once we have the above lemma at our disposal, we propose the approximation algorithm for The Minimum Steiner Tree Terminal Removal problem. We start with an intuitive description of the proposed algorithm. The algorithm is parametrized with parameters Z and Y. Let OPT be rooted at terminal t that we want to remove. Due to Remark 1, we may assume that t has at least two sons t_1 and t_2 in OPT. Each of these two sons determines a subtree of OPT, i. e. the subtree containing t_i, $i \in \{1, 2\}$, obtained by removing the edge between t and t_i from OPT. We let $S'_i \subseteq S'$, $i \in \{1, 2\}$, be the set of terminals contained in the subtree of t_i. Clearly, $S'_1 \cap S'_2 = \emptyset$. Let P_t be the cheapest path from t to S'_1 in OPT and let R_t be the cheapest path from t to S'_2 in OPT. Let $P_t^{(Y)} \subseteq P_t$, respectively $R_t^{(Y)} \subseteq R_t$ be the path composed of the first Y edges on P_t, respectively R_t, starting from t. Let P'_t and R'_t be the remaining parts of P_t and R_t, i. e., $P_t = P_t^{(Y)} \cup P'_t$ and $R_t = R_t^{(Y)} \cup R'_t$. If P_t (respectively R_t) is shorter than Y, we let $P_t^{(Y)} = P_t$ (respectively $R_t^{(Y)} = R_t$) and P'_t (respectively R'_t) be empty. The reoptimization algorithm removes paths $P_t^{(Y)}$ and $R_t^{(Y)}$ from OPT, creating a forest F of at most $2Y + 1$ trees. Note that every tree in F contains at least one terminal vertex. The algorithm then applies the procedure Connect, described further in more detail (see Algorithm 2), to connect the obtained forest. The procedure Connect on the one hand guesses how $\textsc{Opt}'$ connects the trees in F, on the other hand it uses $\mathcal{S}_{\textsc{ApprSMT}}$ to self-reduce the instance using the guessed edges of $\textsc{Opt}'$. At the end Connect returns the best solution among those it computes. The reoptimization algorithm then returns a better solution among Opt and the output of Connect.

We now describe the procedure Connect in more detail. It takes as an input an *SMT* instance (G, S'), a forest $F = T_1 \cup \cdots \cup T_m$ and a parameter Z. The precondition for Connect is that each tree in F has to contain a terminal vertex from S'. Procedure Connect computes a number of Steiner trees for (G, S') and returns the best among the computed trees. It connects the trees in F in a recursive manner. At each recursive call it decreases the number of trees in F by connecting the last tree T_m, i.e., the tree with the highest index among the trees in F, to some other tree $T_i, i < m$. This is done with the help of Lemma 3. Since every tree T_i contains a terminal, we can partition the terminal set S' into two non-empty sets of terminals: the terminals in the last tree T_m and the terminals in the remaining trees of F. To be more precise, we set $S_1 = S' \cap V(T_m)$ and $S_2 = S' \cap V(F - T_m)$ to be the partition of S'. Then $\textsc{Opt}'$ contains a subtree $B_1 \cup P^{(3)} \cup B_2$ as in Lemma 3. Procedure Connect guesses $B_1^{(Z)} \cup P^{(3)} \cup B_2^{(Z)}$ by iterating through all trees of appropriately bounded size. There $B_j^{(Z)}$, $j \in \{1, 2\}$, is the subtree of the first Z floors of the tree B_j. If the height of B_j is smaller than Z, we let $B_j^{(Z)} = B_j^{(h)}$, where $h < Z$ is the height of B_j. In other words, if there is a terminal within the first Z floors of B_j, we only guess as far as to reach that terminal. The number of edges in the tree $B_1^{(Z)} \cup P^{(3)} \cup B_2^{(Z)}$ to be guessed is bounded by $|E(B_1^{(Z)} \cup P^{(3)} \cup B_2^{(Z)})| \leq 2^{Z+1} + 3$. Hence, we let $\overline{T}_m$ run trough all trees of size bounded by $2^{Z+1} + 3$. For each $\overline{T}_m$ we compute the cheapest edges e_m and f_m connecting $\overline{T}_m$ with S_1 and S_2 respectively, and use $\overline{T}_m \cup e_m \cup f_m$ to connect T_m to some other tree $T_i, i < m$. We proceed recursively with a forest of a smaller size. Once F becomes a single connected tree T_1 (at the bottom of recursion), two solutions are computed: $\textsc{Sol}_j = T_1$ and $\textsc{Sol}'_j = \mathcal{S}_{\textsc{ApprSMT}}((G, S'), \bigcup_{\overline{T}_i \in Q} \overline{T}_i)$. There, j is incremented whenever a new pair of solutions is computed and Q is a stack that stores the trees from higher level recursive calls. We present the pseudo-code in Algorithm 2. We allow two operations on the stack Q: $\leftarrow T$ puts a tree T on the top of the stack, whereas $\rightarrow$ pops a tree from the top of the stack.

In the following lemma we give the bounds on the costs of solutions computed by Connect.

Lemma 4. *Let (G, S') be an instance of the SMT problem where G is metric and $\textsc{Opt}'$ be an optimal solution to it satisfying Observation 1. Let a forest $F = T_1 \cup \cdots \cup T_m$ be such that for every tree $T_i, i \in \{1, \ldots, m\}$, we have $V(T_i) \cap S' \neq \emptyset$. Let Z be a positive integer. Let $\textsc{Sol}$ and $\textsc{Sol}'$ be the best solution among $\textsc{Sol}_j$ and $\textsc{Sol}'_j$, respectively. Then there exists a forest $\overline{F} \subseteq \textsc{Opt}'$ of $\textsc{Opt}'$ such that:*

Algorithm 2. Recursive procedure Connect

```
Input: An SMT instance $(G, S')$, $F = \bigcup_{i=1}^{m} T_i$, parameter $Z$
 1: if $m := |F|_T > 1$ then
 2:     $S_1 := S' \cap V(T_m)$
 3:     $S_2 := S' \cap V(\bigcup_{i=1}^{m-1} T_i)$
 4:     $\mathcal{G}_m :=$ all subtrees of $G$ of at most $2^{Z+1} + 3$ edges
 5:     for each tree $\overline{T}_m \in \mathcal{G}_m$ do
 6:         $Q \leftarrow \overline{T}_m$
 7:         $e_m :=$ the cheapest edge between $\overline{T}_m$ and $S_1$
 8:         $f_m :=$ the cheapest edge between $\overline{T}_m$ and $S_2$
 9:         Let $i$ be the index of $T_i \in F$ which contains an endpoint of $f_m$
10:         $T_i := T_i \cup T_m \cup e_m \cup f_m \cup \overline{T}_m$
11:         Connect$((G, S'), F - T_m, Z)$
12:         $Q \rightarrow$
13:     end for
14: else
15:     $j := j + 1$; $\text{Sol}_j := T_1$;
16:     $\text{Sol}'_j := \mathcal{S}_{\text{ApprSMT}}((G, S'), \bigcup_{\overline{T}_i \in Q} \overline{T}_i)$;
17: end if
Output: the best among the computed solutions $\text{Sol}_j$ and $\text{Sol}'_j$
```

$$\text{cost}(\text{Sol}) \leq \text{cost}(F) + \text{cost}(\overline{F}) + |F|_T \frac{\text{cost}(\text{Opt}')}{2^Z},$$
$$\text{cost}(\text{Sol}') \leq \sigma \text{cost}(\text{Opt}') - (\sigma - 1)\text{cost}(\overline{F}).$$

Moreover, the running time of Connect *is polynomial if Z and the number of trees in F are constant.*

Proof. At each recursive call the respective $\overline{T}_i$ runs through all trees of size at most $2^{Z+1}+3$. Due to Lemma 3, for the partition $S' = S_1 \cup S_2$ computed in that recursive call, there exist a tree $B_1 \cup P^{(3)} \cup B_2 \subseteq \text{Opt}'$. Since $B_1^{(Z)} \cup P^{(3)} \cup B_2^{(Z)} \subseteq B_1 \cup P^{(3)} \cup B_2 \subseteq \text{Opt}'$ is a subtree of size at most $2^{Z+1} + 3$, at some point $\overline{T}_i$ is set to $B_1^{(Z)} \cup P^{(3)} \cup B_2^{(Z)}$. Hence, at some point at the bottom of the recursion we end up with every $\overline{T}_i \in Q$ set to its respective $B_1^{(Z)} \cup P^{(3)} \cup B_2^{(Z)}$. In what follows, we analyze the solutions computed exactly at that point. Let j be such that Sol_j and Sol'_j were computed at that point and let $\overline{F} = \bigcup_{i=1}^{|F|_T} \overline{T}_i$ be the forest composed of the corresponding selection of trees on Q. The following bound holds:

$$\text{cost}(\text{Sol}_j) \leq \text{cost}(F) + \text{cost}(\overline{F}) + \sum_{i=1}^{|Q|} (\text{cost}(e_i) + \text{cost}(f_i)).$$

Note that if there is a terminal leaf in $B_1^{(Z)} \subseteq \overline{T}_i$ then $\text{cost}(e_i) = 0$. If there is a terminal leaf in $B_2^{(Z)} \subseteq \overline{T}_i$ then $\text{cost}(f_i) = 0$. If in $B_i^{(Z)}$, $i \in \{1, 2\}$, there is no terminal, due to Lemma 3 there are two paths branching from each of 2^Z non-terminal leaves of $B_i^{(Z)}$. So $\text{cost}(e_i) \leq \frac{\text{cost}(\text{Opt}')}{2^{Z+1}}$ (and analogously $\text{cost}(f_i) \leq \frac{\text{cost}(\text{Opt}')}{2^{Z+1}}$). In any case $\max\{\text{cost}(e_i), \text{cost}(f_i)\} \leq \frac{\text{cost}(\text{Opt}')}{2^{Z+1}}$. Since $|Q| \leq |F|_T$ and $\text{cost}(\text{Sol}) \leq \text{cost}(\text{Sol}_j)$, the upper bound on cost(Sol) as claimed in the lemma follows. The second inequality claimed in the lemma is a consequence of Lemma 2 and the fact that $\overline{F} \subseteq \text{Opt}'$.

Finally, let us analyze the running time of the procedure Connect. At each recursive call Connect runs through at most $|E(G)|^{(2^{Z+1}+3)}$ trees and makes a recursive call for each of these trees. The depth of the recursion is bounded by the number of trees in the input forest minus one, i. e., $|F|_T - 1$. Hence, we obtain a running time of $\mathcal{O}(|I|^{(2^{Z+1}+3)(|F|_T-1)} \cdot \text{Poly}(|I|))$ for some polynomial function Poly where $|I|$ is the size of the instance. If Z and $|F|_T$ are constant, then the running time is polynomial. □

The next lemma states the main result of this section.

Lemma 5. *For any constant $\epsilon > 0$, there is a polynomial-time $(\frac{3\sigma-2}{2\sigma-1} + \epsilon)$-approximation algorithm for SMT under removing a terminal.*

Proof. The final output of the reoptimization algorithm for the modification of removing a terminal is the best solution among $\text{Connect}((G, S'), \text{Opt} - (P_t^{(Y)} \cup R_t^{(Y)}), Z)$ and Opt.

Due to Lemma 4 and because $|F|_T \leq 2Y + 1$, the bounds on the computed solutions are as follows:

$$\begin{aligned}
\text{cost}(\text{Sol}) &\leq \text{cost}(\text{Opt} - (P_t^{(Y)} \cup R_t^{(Y)})) + \text{cost}(\overline{F}) + \frac{(2Y+1)\text{cost}(\text{Opt}')}{2^Z},\\
\text{cost}(\text{Sol}') &\leq \sigma\text{cost}(\text{Opt}') - (\sigma - 1)\text{cost}(\overline{F}),\\
\text{cost}(\text{Opt}) &\leq \text{cost}(\text{Opt}') + \min\{\text{cost}(P_t), \text{cost}(R_t)\}.
\end{aligned}$$

Taking into account the bound $\text{cost}(\text{Opt}) \leq \text{cost}(\text{Opt}') + \text{cost}(R_t)$ it holds that

$$\begin{aligned}
&\text{cost}(\text{Opt} - (P_t^{(Y)} \cup R_t^{(Y)}))\\
&\leq \text{cost}(\text{Opt}) - \text{cost}(P_t^{(Y)}) - \text{cost}(R_t^{(Y)})\\
&\leq \text{cost}(\text{Opt}') + \text{cost}(R_t) - \text{cost}(R_t^{(Y)}) + \text{cost}(P_t) - \text{cost}(P_t^{(Y)}) - \text{cost}(P_t)\\
&= \text{cost}(\text{Opt}') + \text{cost}(R_t') + \text{cost}(P_t') - \text{cost}(P_t).
\end{aligned}$$

Since w.l.o.g. $\text{cost}(R_t') \leq \text{cost}(P_t')$, we can rewrite

$$\text{cost}(\text{Sol}) \leq (1 + \frac{2Y+1}{2^Z})\text{cost}(\text{Opt}') - \text{cost}(P_t) + 2\text{cost}(P_t') + \text{cost}(\overline{F}),$$

where $\text{cost}(P_t') \leq \frac{\text{cost}(\text{Opt})}{Y} \leq \frac{\text{cost}(\text{Opt}')}{Y} + \frac{\text{cost}(P_t)}{Y}$ (due to Observation 1 and the fact that P_t was the cheapest, there have to be $Y - 1$ paths in Opt more expensive than P_t' and disjoint with it) and the bound on the cost of Opt given by $\text{cost}(\text{Opt}) \leq \text{cost}(\text{Opt}') + \text{cost}(P_t)$.

As a result of these calculations, we obtain the following bounds on the considered solutions:

$$\begin{aligned}
&\mathrm{cost}(\textsc{Sol}) \le (1+\frac{2}{Y}+\frac{2Y+1}{2^Z})\mathrm{cost}(\textsc{Opt}') - (1-\frac{2}{Y})\mathrm{cost}(P_t) + \mathrm{cost}(\overline{F}),\\
&\mathrm{cost}(\textsc{Sol}') \le \sigma \mathrm{cost}(\textsc{Opt}') - (\sigma-1)\mathrm{cost}(\overline{F}),\\
&\mathrm{cost}(\textsc{Opt}) \le \mathrm{cost}(\textsc{Opt}') + \mathrm{cost}(P_t).
\end{aligned}$$

Hence the cost of the overall solution is bounded by minimum among these three bounds. Observe that the first two bounds are functions of $\mathrm{cost}(\overline{F})$, increasing and decreasing with $\mathrm{cost}(\overline{F})$ respectively. Setting them equal allows computing the maximum value of the minimum among the first two bounds and the value of $\mathrm{cost}(\overline{F})$ where the maximum is reached as a function of $\mathrm{cost}(\textsc{Opt}')$ and $\mathrm{cost}(P_t)$. Similarly we eliminate $\mathrm{cost}(P_t)$ using the second and the third bound. The approximation ratio we obtain is

$$\frac{3\sigma - 2 + (\sigma-1)\frac{2Y+1}{2^Z}}{2\sigma - 1 - (\sigma-1)\frac{2}{Y}}.$$

This expression converges to $\frac{3\sigma-2}{2\sigma-1}$ as Z and Y grow to infinity. It is clear that with the right choice of constant parameters Z and Y we will obtain a value within $\frac{3\sigma-2}{2\sigma-1}+\epsilon$ for any $\epsilon > 0$. Polynomial running time follows from the fact, that Connect runs in a polynomial time. □

4 Edge Cost Increase

In this section we consider the reoptimization scenario of increasing the cost of one edge. We provide a polynomial-time 1.281-approximation algorithm for this reoptimization variant.

Recall that the *SMT* Edge Cost Increase problem takes as input two *SMT* instances (G,S) and (G',S) together with an optimal solution Opt for (G,S), where $G=(V,E,\mathrm{cost})$, $G'=(V,E,\mathrm{cost}')$ and cost coincides with cost′ on all the edges but one edge e^*, where $\mathrm{cost}(e^*) < \mathrm{cost}'(e^*)$. We will show that without loss of generality G and G' may be assumed to be metric. For a graph G, we call an edge $f \in E(G)$ *implied*, if its cost in the metric closure $\overline{G}$ is the cost of some path between its endpoints, and this path is not just a single edge f.

Remark 2. Without loss of generality we may assume that the input instance to *SMT* under edge cost increase consists of two *SMT* problem instances (G,S) and (G',S) and a solution Opt for (G,S), where $G=(V,E,\mathrm{cost})$ and $G'=(V,E,\mathrm{cost}')$ are metric and complete, Opt satisfies Observation 1 and for every $f \in \textsc{Opt} - e^*$ it holds that $\mathrm{cost}(f) = \mathrm{cost}'(f)$ whereas $\mathrm{cost}(e^*) < \mathrm{cost}'(e^*)$.

Proof. Assume G and G' are not metric and complete. Observe that Opt does not contain implied edges. Since cost and cost′ coincide on all the edges but e^*, for every $f \subseteq \textsc{Opt} - e^*$ it holds that $\overline{\mathrm{cost}}(f) = \overline{\mathrm{cost}'}(f)$ whereas $\overline{\mathrm{cost}}(e^*) <$

$\overline{\mathrm{cost}'}(e^*)$. We solve the reoptimization problem for the instance $(\overline{G}, S)$, $(\overline{G'}, S)$ and an optimal solution Opt for $(\overline{G}, S)$. This instance satisfies Remark 2. Note that when transforming Opt to its counterpart satisfying Observation 1 we preserve the invariant that at most one edge in Opt increases its cost in $\overline{G'}$. The approximate solution for $(\overline{G'}, S)$ can easily be transformed to the solution for (G', S) with the same cost. □

Lemma 6. *For any constant $\epsilon > 0$, there is a polynomial-time $\left(\frac{2\sigma-1}{\sigma} + \epsilon\right)$-approximation algorithm for the SMT Edge Cost Increase.*

Proof. Let Z be a parameter. Let $S = S_1 \cup S_2$ be a partition of the terminals depending on to which tree in $\mathrm{Opt} - e^*$ they belong. Let Opt' be an optimal solution to (G', S). Consider $P^{(3)} \cup B_1 \cup B_2 \subseteq \mathrm{Opt}'$ as in Lemma 3 for S_1 and S_2. Let $B_i^{(Z)}$, $i \in \{1, 2\}$, be the first Z levels of B_i starting from the root and let $T^{(Z)} = P^{(3)} \cup B_1^{(Z)} \cup B_2^{(Z)}$. For a tree T, let $f_1(T)$ be the cheapest edge connecting it with S_1 and $f_2(T)$ be the cheapest edge connecting it with S_2. The reoptimization algorithm guesses $T^{(Z)}$, and returns a better solution among $\mathrm{Opt} - e^* \cup T^{(Z)} \cup f_1(T^{(Z)}) \cup f_2(T^{(Z)})$ and $\mathcal{S}_{\mathrm{ApprSMT}}((G', S), T^{(Z)})$. By "guesses" we mean that it iterates over all trees T of size bounded by $2^{Z+1} + 3$. For each T, it computes $\mathrm{Opt} - e^* \cup T \cup f_1(T) \cup f_2(T)$ and $\mathcal{S}_{\mathrm{ApprSMT}}((G', S), T)$. It returns the best of the computed solutions.

Note that $\mathrm{cost}'(\mathrm{Opt} - e^*) \leq \mathrm{cost}'(\mathrm{Opt}')$ because

$$\mathrm{cost}'(\mathrm{Opt} - e^*) = \mathrm{cost}(\mathrm{Opt} - e^*) \leq \mathrm{cost}(\mathrm{Opt}) \leq \mathrm{cost}(\mathrm{Opt}') \leq \mathrm{cost}'(\mathrm{Opt}').$$

Due to Lemma 3,

$$\mathrm{cost}'(f_1(T^{(Z)})), \mathrm{cost}'(f_2(T^{(Z)})) \leq \frac{\mathrm{cost}'(\mathrm{Opt}')}{2^{Z+1}}.$$

Therefore,

$$\begin{aligned}\mathrm{cost}'(\mathrm{Sol}) &\leq \mathrm{cost}'(\mathrm{Opt} - e^* \cup T^{(Z)} \cup f_1(T^{(Z)}) \cup f_2(T^{(Z)})) \\ &\leq (1 + \frac{1}{2^Z})\mathrm{cost}'(\mathrm{Opt}') + \mathrm{cost}'(T^{(Z)}).\end{aligned}$$

On the other hand, by the power of Lemma 3,

$$\mathrm{cost}'(\mathcal{S}_{\mathrm{ApprSMT}}((G', S), T^{(Z)})) \leq \sigma \mathrm{cost}'(\mathrm{Opt}') - (\sigma - 1)\mathrm{cost}'(T^{(Z)}),$$

where σ is the approximation ratio of an algorithm ApprSMT for *SMT*. Again, we view these two bounds on the overall solution as functions of $\mathrm{cost}'(T^{(Z)})$, increasing and decreasing respectively. We compute the maximum value of the minimum of these two bounds by setting them equal. The resulting approximation ratio is

$$\frac{2\sigma - 1 + \frac{\sigma-1}{2^Z}}{\sigma}.$$

This ratio can be arbitrarily close to $\frac{2\sigma-1}{\sigma}$ with the right choice of constant parameter Z. □

5 Concluding Remarks

To conclude the paper, we present the state of the art on the *SMT* reoptimization in Table 1. An open question concerning the reoptimization variants of *SMT* studied here is the existence of a PTAS for them. In [16], the authors proved that there cannot be a PTAS for the modification of adding a vertex to the graph. They also provided a PTAS for all the modifications studied here if the input graph is β-metric for some constant $\beta < 1$. A PTAS for the metric case remains an interesting open problem.

Table 1. Different types of *SMT* modifications with the corresponding best up to date approximation ratios

The *SMT* problem under:			
Terminal		Edge cost	
addition	removal	decrease	increase
1.218 [22]	1.218 here	1.246 [5]	1.279 here

References

1. Archetti, C., Bertazzi, L., Speranza, M.G.: Reoptimizing the traveling salesman problem. Networks 42(3), 154–159 (2003)
2. Archetti, C., Luca, B., Speranza, M.G.: Reoptimizing the 0-1 knapsack problem. Technical Report 267, University of Brescia (2006)
3. Ausiello, G., Escoffier, B., Monnot, J., Paschos, V.T.: Reoptimization of Minimum and Maximum Traveling Salesman's Tours. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 196–207. Springer, Heidelberg (2006)
4. Bern, M.W., Plassmann, P.E.: The Steiner problem with edge lengths 1 and 2. Inf. Process. Lett. 32(4), 171–176 (1989)
5. Bilò, D., Böckenhauer, H.-J., Hromkovič, J., Královič, R., Mömke, T., Widmayer, P., Zych, A.: Reoptimization of Steiner Trees. In: Gudmundsson, J. (ed.) SWAT 2008. LNCS, vol. 5124, pp. 258–269. Springer, Heidelberg (2008)
6. Bilò, D., Böckenhauer, H.-J., Komm, D., Královič, R., Mömke, T., Seibert, S., Zych, A.: Reoptimization of the shortest common superstring problem. Algorithmica 61(2), 227–251 (2011)
7. Bilò, D., Widmayer, P., Zych, A.: Reoptimization of Weighted Graph and Covering Problems. In: Bampis, E., Skutella, M. (eds.) WAOA 2008. LNCS, vol. 5426, pp. 201–213. Springer, Heidelberg (2009)
8. Böckenhauer, H.-J., Forlizzi, L., Hromkovič, J., Kneis, J., Kupke, J., Proietti, G., Widmayer, P.: Reusing Optimal TSP Solutions for Locally Modified Input Instances (Extended Abstract). In: Navarro, G., Bertossi, L., Kohayakawa, Y. (eds.) TCS 2006. IFIP, vol. 209, pp. 251–270. Springer, Boston (2006)
9. Böckenhauer, H.-J., Hromkovič, J., Královič, R., Mömke, T., Rossmanith, P.: Reoptimization of Steiner trees: Changing the terminal set, vol. 410, pp. 3428–3435. Elsevier Science Publishers Ltd. (August 2009)

10. Böckenhauer, H.-J., Hromkovič, 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. LNCS, vol. 4910, pp. 50–65. Springer, Heidelberg (2008)
11. Böckenhauer, H.-J., Komm, D.: Reoptimization of the Metric Deadline TSP. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 156–167. Springer, Heidelberg (2008)
12. Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L.: An Improved LP-based Approximation for Steiner Tree. In: STOC 2010, Best Paper Award (2010)
13. Escoffier, B., Milanic, M., Paschos, V.T.: Simple and fast reoptimizations for the Steiner tree problem. Technical Report 2007-01, DIMACS (2007)
14. Escoffier, B., Milanič, M., Paschos, V.T.: Simple and fast reoptimizations for the Steiner tree problem 4(2), 86–94 (2009)
15. Escoffier, B., Ausiello, G., Bonifaci, V.: Complexity and Approximation in Reoptimization. In: Computability in Context: Computation and Logic in the Real World. Imperial College Press (2011)
16. Böckenhauer, H.-J., Freiermuth, K., Hromkovič, J., Mömke, T., Sprock, A., Steffen, B.: The Steiner Tree Reoptimization Problem with Sharpened Triangle Inequality. In: Calamoneri, T., Diaz, J. (eds.) CIAC 2010. LNCS, vol. 6078, pp. 180–191. Springer, Heidelberg (2010)
17. Mikhailyuk, V.A.: Reoptimization of set covering problems. Cybernetics and Sys. Anal. 46, 879–883 (2010)
18. Prömel, H.J., Steger, A.: The Steiner Tree Problem. Advanced Lectures in Mathematics. Friedr. Vieweg & Sohn, Braunschweig (2002)
19. Robins, G., Zelikovsky, A.: Improved Steiner tree approximation in graphs. In: ACM-SIAM Symposium on Discrete Algorithms, pp. 770–779. ACM (2000)
20. Schäffter, M.W.: Scheduling with forbidden sets. Discrete Applied Mathematics 72(1-2), 155–166 (1997)
21. Zych, A.: Reoptimization of NP-hard problems. Ph.D. thesis, ETH Zürich
22. Zych, A., Bilò, D.: New reoptimization techniques applied to Steiner tree problem. Electronic Notes in Discrete Mathematics 37, 387–392 (2011); LAGOS 2011 - VI Latin-American Algorithms, Graphs and Optimization Symposium