Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
CNRS UMR 7024

CAHIER DU LAMSADE
279

Novembre 2008

# Fast reoptimization for the minimum spanning tree problem

Nicolas Boria, Vangelis Th. Paschos

# Fast reoptimization for the minimum spanning tree problem

Nicolas Boria Vangelis Th. Paschos

LAMSADE, CNRS UMR 7024 and Université Paris-Dauphine, France

{boria,paschos}@lamsade.dauphine.fr

November 7, 2008

**Abstract**

Minimum spanning tree is a classical polynomial problem very well known in operational research and in theoretical computer science. In this paper, we settle the reoptimization versions of this problem, which can be formulated as follows: given an instance of the problem for which we already know some optimal solution, and given some "small" perturbations on this initial instance, is it possible to compute a new (optimal or at least near-optimal) solution for the modified instance without ex nihilo computation? We focus on two kinds of modifications: node-insertions and node-deletions. For the former type of modifications, where $k$ new nodes are inserted together with their incident edges, we first propose a fast strategy with complexity $O(kn)$ which provides a $\max\{2, 3-(2/(k-1))\}$-approximation ratio, in complete metric graphs. We then devise a more elaborated strategy that computes optimal solutions in any graph with complexity $O(kn\log n)$. When $k$ nodes are deleted, we devise a strategy which in $O(n)$ achieves approximation ratio bounded above by $2\lceil |L_{\max}|/2\rceil$ in complete metric graphs, where $L_{\max}$ is the longest deleted path and $|L_{\max}|$ is the number of its edges. For any of the approximation strategies, we also provide lower bounds on their approximation ratios.

## 1 Introduction

The minimum spanning tree problem (MIN SPANNING TREE) is a celebrated problem, very useful to model any kind of networks in transports, communications, energy, logistics, etc. It consists, given a graph $G(V,E)$ with weights on its edges, of determining a tree of minimum total edge-weight spanning $V$. MIN SPANNING TREE is polynomial. One of the algorithms solving it is the seminal Kruskal's algorithm ([10]) which consists of sorting edges in non increasing order with respect to their weights, and of inserting them in the current solution on the condition that they do not create cycles with those that have been inserted previously. The complexity of this algorithm $O(m\log n)$.

In this paper, we settle reoptimization versions for MIN SPANNING TREE. The reoptimization setting can be generally described as follows: given an instance $I$ of an optimization problem $\Pi$ for which we already know an optimal solution $S^*$, if we slightly modify $I$ into a new instance $I_x$, is it possible to compute a new solution $S_x$ for $I_x$ that is either optimal or at least a good approximation of the optimal solution with complexity better than that needed to solve $\Pi$ ex nihilo in $I_x$? The most common modifications considered in the literature are insertions or deletions of a few vertices.

Reoptimization issue is already used for studying several optimization problems such as scheduling problems ([11, 3], or [4] for practical applications), MIN TSP ([1, 2, 5]), MAX TSP ([2, 5]), Steiner tree problem ([8, 6]), as well as for classical polynomial problems where the goal is to recompute the optimum solution as fast as possible ([7, 9]). In particular, in [9], MIN SPANNING TREE is tackled but in a different setting from the ours. More precisely, reoptimization strategies are devised there by using not only the solution on the initial instance itself but also the sorted lists that were used to compute this solution, while our approach consists of using only the optimal solution as available information. Using these lists enables the authors of [9] to formulate an optimal reoptimization strategy that computes the optimal solution in $O(\sqrt{n})$ for any edge's insertion, deletion, or modification.

Consider an initial instance $I = G(V, E, \vec{w}_m)$ of MIN SPANNING TREE, i.e., a graph $G$ on a set $V$ of $n$ vertices, a set $E$ of $m$ edges and a vector $\vec{w}$ of edge-weights. The problems settled in the sequel are the following:

1. MIN SPANNING TREE$_k$+ where the instance $I_x$, consists of a graph $G_x(V_x, E_x)$, where $V_x = V \cup X$, with $X$ a vector of $k$ nodes $(x_1, x_2, \ldots, x_k)$ and $E_x = E \cup E(X)$, where $E(X)$ is the set of edges adjacent to the nodes of $X$;

2. MIN SPANNING TREE$_k-$, $k < n$, where the instance $I_x$, consists of a graph $G_x(V_x, E_x)$, where $V_x = V \setminus X$, with $X \subset V$ a vector of $k$ nodes $(x_1, x_2, \ldots, x_k)$ and $E_x = E \setminus E(X)$, where $E(X)$ is the set of edges adjacent to the nodes of $X$.

## 2 Node insertions

In this section, we study two reoptimization strategies for MIN SPANNING TREE$_k$+, namely `REOPT1+` and `REOPT2+`.

### 2.1 Algorithm `REOPT1+`

We first propose an easy algorithm (called `REOPT1+`) that extends $T^*$ into a tree spanning the whole $V_x$. It works as follows:

1. for each $x_i$ in $X$, let $e_i^* = (x_i, v_i^*)$ be the lightest edge linking $x_i$ to a node of $V$; set $E^* = \{e_1^*, \ldots, e_k^*\}$, $V^* = \{v_1^*, \ldots, v_k^*\}$ (note that $|E^*| = k$ and $|V^*| \leqslant k$);

2. build an artificial node $v'$ as the contraction of all nodes in $V^*$, so that, $\forall i$, $e_i' = (v', x_i)$, and $w(e_i') = w(e_i^*)$; set $E' = \{e_1', \ldots, e_k'\}$;

3. run Kruskal's algorithm on the graph $H = (X \cup \{v'\}, (X \times X) \cup E')$; let $\Psi'$ be the resulting tree;

4. replace each edge $e_i'$ in $\Psi'$ with the corresponding edge $e_i^*$ and denote by $\Psi^*$ the resulting set;

5. output $T_{\text{reopt}} = T^* \cup \Psi^*$.

It is easy to see that, due to steps 1 and 3 the running time of Algorithm `REOPT1+` is $O(kn + k \log k)$.

We now prove that the solution $T_{\text{reopt}}$ computed in step 5 of `REOPT1+` is indeed a spanning tree. For this we show the following three claims:

Claim 1. $T_{\text{reopt}}$ spans $V_x$;

Claim 2. $T_{\text{reopt}}$ is connected;

Claim 3. $T_{\text{reopt}}$ is acyclic.

In order to prove Claim 1, just observe that $T^*$ spans all nodes of $V$ and each node of $X$ is spanned by $\Psi^*$.

In order to prove Claim 2, we need to show that the three following items hold:

Item 1. $\forall y, z \in V$, there exists a path $P(y, z)$ from $y$ to $z$ in $T_{\text{reopt}}$;

Item 2. $\forall x_i, x_j \in X$, there exists a path $P(x_i, x_j)$ from $x_i$ to $x_j$ in $T_{\text{reopt}}$;

Item 3. $\exists y \in V, x \in X$, such that $(y, z) \in T_{\text{reopt}}$.

For Item 1, just remark that $T_{\text{reopt}}$ includes $T^*$ that is a tree spanning $V$.

Let us now prove Item 2. Denote by $P'(x_i, x_j)$ the path from $x_i$ to $x_j$ in $\Psi'$. If node $v'$ built at step 2 is not on $P'(x_i, x_j)$, then this path exists also in $\Psi^*$ and thus in $T_{\text{reopt}}$. If, on the contrary, $v'$ is on $P'(x_i, x_j)$, then this path can be written as:

$$P'(x_i, x_j) = P'(x_i, x_s) \cup (x_s, v') \cup (v', x_t) \cup P'(x_t, x_j)$$

so that, in $\Psi^*$, $P'(x_i, x_j)$ corresponds to the union of two distinct and potentially disconnected paths $P(x_i, v_s^*)$ and $P(v_t^*, x_j)$. Since both $v_s^*$ and $v_t^*$ are nodes of the initial graph, according to Item 1, there exists a path $P(v_s^*, v_t^*)$ in $T_{\text{reopt}}$. So, there exists a path from $x_i$ to $x_j$ in $T_{\text{reopt}}$, and this path is:

$$P(x_i, x_j) = P(x_i, v_s^*) \cup P(v_s^*, v_t^*) \cup P(v_t^*, x_j)$$

completing the proof of Item 2

For Item 3, since the node $v'$ is spanned by $\Psi'$, then there exists an edge $e \in E'$ such that $e \in \Psi'$ and also there exists an edge $e \in E^*$ such that $e \in \Psi^*$. So, there exists at least an edge in $T_{\text{reopt}}$ which connects the two sets $T^*$ and $\Psi^*$ completing so the proof of Item 3.

Items 1, 2 and 3 put together, derive connectivity of $T_{\text{reopt}}$.

Let us now show Claim 3. Since both $T^*$ and $\Psi^*$ are trees, a cycle $C$ in $T_{\text{reopt}}$ should necessarily be the union of two paths $P(y, z)$, and $P'(y, z)$, where $P(y, z)$ is the unique path from $y$ to $z$ in $T^*$, and $P'(y, z)$ is the unique path from $y$ to $z$ in $\Psi^*$. Nodes $y$ and $z$ are necessarily nodes of the initial graph, since they are spanned by $T^*$. They are also nodes of the set $V^*$, defined at step 1 of `REOPT1+`, since they are spanned by $\Psi^*$. But since nodes of $V^*$ are contracted in a single node in $\Psi'$, a path from $y$ to $z$ in $\Psi^*$ is equivalent to a path from $v'$ to $v'$ in $\Psi'$, a contradiction since $\Psi'$ is acyclic.

Putting together Claims 1, 2 and 3, derives that $T_{\text{reopt}}$ returned by `REOPT1+` is a tree spanning $V_x$.

**Proposition 1.** *There exist arbitrarily large instances of* MIN SPANNING TREE$_k$+ *where the approximation ratio of* ***REOPT1+*** *is at least* $n$ *(the order of* $G$*).*

**Proof.** More precisely we will show that proposition's statement holds even for MINIMUM SPANNING TREE$_1$+. Consider a complete graph $G$ on $n$ vertices with all edge weights equal to $n+1$. Here, any spanning tree $T$ has the same weight $w(T) = (n-1)(n+1) = n^2 - 1$. Assume that a new node $x$ is inserted so that the new graph $G_x$ remains complete. Assume also that the weights in $E(\{x\})$ are all equal to 1.

The new optimal spanning tree is obviously $E(x)$ (a star rooted at $x$), and its weight is $w(T_x^*) = n$. It is easy to see that `REOPT1+` will compute a spanning tree $T_{\text{reopt}}$ of $G_x$ with $w(T_{\text{reopt}}) = n^2$ so that the ratio between its value and that of $T_x^*$ equals $n$. ∎

We now restrict ourselves to a particular but natural classe of graphs, namely to complete metric graphs. We prove that in such graphs `REOPT1+`, although non optimal, behaves much better. More precisely, we prove the following result.

**Proposition 2.** *In complete metric graphs, `REOPT1+` achieves approximation ratio bounded above by* $\max\{2, 3 - (2/(k-1))\}$.

The proof of Proposition 2 requires the following Lemmata 1, 2 and 3.

**Lemma 1.** *`REOPT1+` computes a smallest spanning tree on $G_x$ that includes $T^*$.*

**Proof.** First, we will show that $\Psi'$, the solution computed by step 3 of `REOPT1+` is the same as the tree computed by the following algorithm `OTHER`: contract all the nodes of $V$ into a single node $v'$ (this contraction results in a multigraph $H$) and run Kruskal's Algorithm on $H$.

The main difference between `REOPT1+` and `OTHER` relies upon the set of candidate edges: in `OTHER`, this set is the whole set $E(X)$, whereas in `REOPT1+` this set is $(X \times X) \cup E^*$, but the set of edges that are taken as candidates by `OTHER` but not by `REOPT1+` will never be included in the solution by `OTHER`. Indeed, when Kruskal's Algorithm runs on a multigraph, it will never pick a non-minimum weight edge between two nodes, so that in `OTHER`, only edges of $(X \times X) \cup E^*$ are "real" candidates since $E^*$ is the set of minimum weight edges for each pair of nodes $(v', x_i)$.

Assume that $T_{\text{reopt}}$ is not one of the minimum spanning trees on $G_x$ that includes $T^*$. This straightforwardly implies that there exists a tree $\tilde{T}$ including $T^*$ "lighter" than $T_{\text{reopt}}$. The existence of $\tilde{T}$ induces the existence of a set $\tilde{\Psi}$ resulting from the contraction of $T^*$ in $\tilde{T}$, so that $w(\tilde{\Psi}) < w(\Psi')$, a contradiction since, as we showed, $\Psi'$ coincides with a spanning tree computed by Kruskal's algorithm on $H$, hence $\Psi'$ is a minimum spanning tree of $H$. ∎

**Lemma 2.** *Given an rooted tree $T$ spanning the vertices of a complete and metric graph $G$, denoting by $\varphi$ the path that links the leaves of $T$ in their order of appearance in a depth-first-search (dfs) of $T$ and by $P$ the path inside $T$ connecting the first visited leaf with the last one, $w(\varphi) \leqslant 2w(T) - w(P)$.*

**Proof.** Let $l$ be the number of leaves in $T$, and $x_i$ be the $i$th leaf to be visited in a dfs of $T$. Thus, $x_1$ is the first visited leaf, and $x_l$ is the last one.

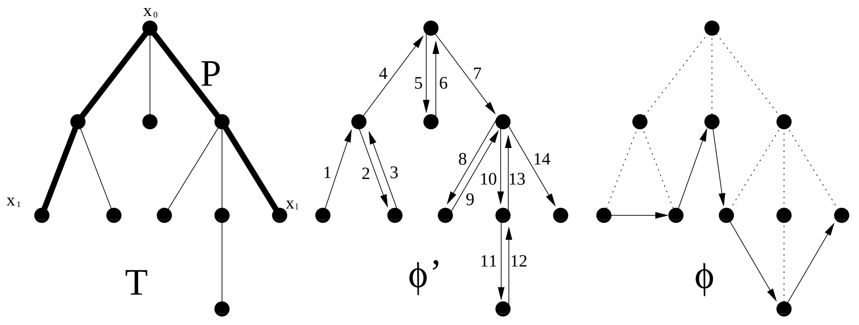

Figure 1: Building the path $\varphi$ in $T$.

The tree $T$ is clearly an outerplanar graph (it can be drawn in the plane so that no edges cross, and each node of $T$ lies in the outer face). Then, we can easily find a non elementary path $\varphi'$ between $x_1$ and $x_l$ spanning all nodes of $T$. This path consists of the outer face of $T$ between $x_1$ and $x_l$ (Figure 1). Another way to represent $\varphi'$ is to see it as the union of paths $P(x_1, x_2)$, $P(x_2, x_3)$, ..., $P(x_{l-1}, x_l)$, where $P(x_i, x_j)$ is the unique path between $x_i$ and $x_j$ in $T$. So, $w(\varphi') = 2w(T) - w(P)$, where $P$ is the unique path between $x_1$ and $x_l$ in $T$. Indeed, each edge of $T$ appears exactly twice on the path, except for the edges of $P$ which appear only once.

This path goes through each leaf exactly once. So, if we delete each series of internal nodes in $\varphi'$ and replace them with corresponding shortcuts, we get a path $\varphi$ going through all leaves.

Given the metric property of the graph, each shortcut is lighter than the subpath that it replaces, so that $w(\varphi) \leqslant w(\varphi') = 2w(T) - w(P)$, qed. ∎

**Lemma 3.** *Given a rooted tree $T$ with at most $k \geqslant 2$ nodes, the heaviest path of $T$ has weight at least $2w(T)/(k-1)$.*

**Proof.** Let us first notice that inside a tree, the heaviest path necessarily connects two leaves. Denote by $P$ the heaviest path in $T$. The set $P \cup \varphi'$ (where $\varphi'$ is as in Lemma 2), can be partitioned into at most $k-1$ sub-paths (recall that as we noticed above, $\varphi'$ is the disjoint union of $l-1$ subpaths connecting two consecutive leaves). The first element of this partition is the path in $T$ connecting the first leaf with the second one, the second element is the path connecting the second leaf with the third one, and so on. Finally, the last element is the path connecting the last leaf with the first one, i.e., the path $P$ itself. With respect to Figure 1, the partition of $P \cup \varphi'$ is: $\{(1,2),(3,4,5),(6,7,8),(9,10,11),(12,13,14),P\}$.

Since $P$ is the heaviest element of the partition, it is necessarily heavier than the average weight of the partition, i.e., $w(P) \geqslant w(P \cup \varphi')/|P \cup \varphi'|$.

Finally, let us notice that $w(P \cup \varphi') = 2w(T)$ and that if $T$ has $l$ leaves, then $|P \cup \varphi'| = l$. Considering that $l \leqslant k-1$ (one of the $k$ nodes of $T$ is the root), we get:

$$w\left(P\right) \geqslant \frac{2}{l} w\left(T\right) \geqslant \frac{2}{k-1} w\left(T\right)$$

as claimed. ∎

We are well-prepared now to continue the proof of Proposition 2. For this, we first try to analyze the structure of an optimal solution $T_x^*$. The edges of this tree can be split into the following three distinct subsets:

1. $E^0 = T_x^* \cap V \times V$, i.e., the subset of edges of $T_x^*$ with no endpoints in $X$;
2. $E^1 = T_x^* \cap V \times X$, i.e., the subset of edges of $T_x^*$ with one endpoint in $X$ one endpoint in $V$;
3. $E^2 = T_x^* \cap X \times X$, i.e., the subset of edges of $T_x^*$ with two endpoints in $X$.

In general, all of these subsets are forests. In particular, $E^2$ can be considered as the union of its distinct maximal connected components (trees). Each of these components will be defined as $E_j^2 \cup L_j$, where $L_j$ is the set of edges such that there is no path in $T_x^*$ connecting two nodes of $V$ that contains them (in other words, $L_j$ is the set of edges that do not belong to any path of $T_x^*$ connecting two nodes of $V$). For each tree $E_j^2 \cup L_j$, we will denote by $E_j^1$ the set of edges of $E^1$ incident to nodes of this tree.

For example, consider the tree of Figure 2. There are five different maximal connected components $E_j^2 \cup L_j$; the cuts associated to each of these components define the sets $E_j^1$. Inside each component, edges of $L_j$ are shown boldfaced. To identify them, one must check for each edge $e$ of $E^2$ if its removal disconnects in $T_x^*$ some pair of nodes of $V \times V$. If not, then $e$ belongs to some set $L_j$, else it belongs to some set $E_j^2$. In other words, sets $L_j$ can be seen as the “last” edges of the paths of $T_x^*$ leading to its leaves and belonging to $E^2$.

Fix some $j$. Inside tree $E_j^2$, denote by $P_j^{\max}$ the heaviest path connecting two leaves (of $E_j^2$), denoted by $xs_j$ and $xt_j$. Since $E_j^2$ and $L_j$ are disjoint, both $xs_j$ and $xt_j$ have at least one neighbor in $V$. Pick at random a node $vs_j$ among the neighbors of $xs_j$ in $V$, a node $vt_j$ among the neighbors of $xt_j$ and build a path $\varphi_j$ from $vs_j$ to $vt_j$ which links all leaves of the tree $E_j^2 \cup E_j^1$ as shown in Figure 3. Path $\varphi_j$ meets the hypothesis of Lemma 2. So:

$$\begin{aligned} w\left(\varphi_j\right) &\leqslant 2\left(w\left(E_j^2\right) + w\left(E_j^1\right)\right) - \left(w\left(vs_j, xs_j\right) + w\left(P_j^{\max}\right) + w\left(vt_j, xt_j\right)\right) \\ &\Downarrow \qquad (1) \\ w\left(\varphi_j\right) &\leqslant 2\left(w\left(E_j^2\right) + w\left(E_j^1\right)\right) - w\left(e_j^*\right) - w\left(P_j^{\max}\right) \qquad (2) \end{aligned}$$

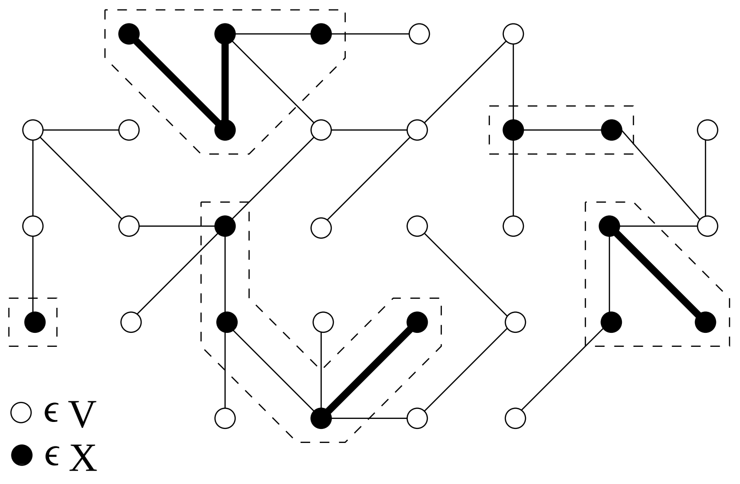

Figure 2: Structure of $T_x^*$.

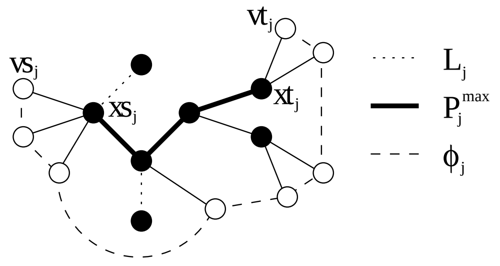

Figure 3: A tree $E_j^2 \cup E_j^1$ and the corresponding path $\varphi_j$.

where, in order to simplify equations, we denote by $e_j^*$ the lightest among edges $(vs_j, xs_j)$ and $(vt_j, xt_j)$; this notation will be also adopted for the sequel.

Since $P_j^{\max}$ is assumed to be the heaviest path connecting terminal nodes of the tree $E_j^2$, we get by Lemma 3:

$$w\left(P_j^{\max}\right) \geqslant \frac{2w\left(E_j^2\right)}{k-1} \tag{3}$$

where, if $k = 1$, then $w(E_j^2) = w(P_j^{\max}) = 0$. Combining (3) and (2), some algebra leads to the following:

$$\begin{aligned} w\left(\varphi_j\right) + w\left(e_j^*\right) + w\left(E_j^2\right) &\leqslant \left(3 - \frac{2}{k-1}\right) w\left(E_j^2\right) + 2w\left(E_j^1\right) \\ w\left(\varphi_j\right) + w\left(e_j^*\right) + w\left(E_j^2 \cup L_j\right) &\leqslant \left(3 - \frac{2}{k-1}\right) w\left(E_j^2 \cup L_j\right) + 2w\left(E_j^1\right) \qquad (4) \end{aligned}$$

Considering that the different trees $E_j^2 \cup L_j \cup E_j^1$ are distinct, we sum (4) for each $j$ and get the following:

$$\begin{aligned} w\left(\Phi\right) + w\left(E^*\right) + w\left(E^2\right) &\leqslant \left(3 - \frac{2}{k-1}\right) w\left(E^2\right) + 2w\left(E^1\right) \\ &\Downarrow \\ w\left(E^0\right) + w\left(\Phi\right) + w\left(E^*\right) + w\left(E^2\right) &\leqslant \left(3 - \frac{2}{k-1}\right) w\left(E^2\right) + 2w\left(E^1\right) + w\left(E^0\right) \end{aligned}$$

Let us now take a closer look at the set $E^0 \cup \Phi$. Remark that the set $E^0$ is a forest spanning nodes of the initial graph $G$. In $T_x^*$, each node $u$ of $V$ is connected to any other node $v$ of $V$ by a path $P(u,v)$. This path may include several nodes of $X$, thus it might not exist in $E^0$. We will show that set $\Phi$ reconnects each component of $E^0$.

Fix such a path $P(u,v)$ and denote by $X_j' = \{x_{j_1}, x_{j_2}, \dots, x_{j_l}\}$, the consecutive nodes of $X$ encountered in $P(u,v)$. Then:

$$P(u,v) = P\left(u,u_j\right) \cup \left(u_j, x_{j_1}\right) \cup P\left(x_{j_1}, x_{j_l}\right) \cup \left(x_{j_l}, v_j\right) \cup P\left(v_j, v\right)$$

where paths $P(u,u_j)$ and $P(v_j,v)$ can contain vertices of both $V$ and $X$, while path $P(x_{j_1}, x_{j_l})$ only contains vertices from $X$. Also, edges $(u_j, x_j), (x_{j_l}, v_j) \in E^1$ and path $P(x_{j_1}, x_{j_l}) \in E^2$. Obviously, this last path is surrounded by two nodes of $V$, here $u_j$ and $v_j$. Path $(u_j, x_{j_1}) \cup P(x_{j_1}, x_{j_l}) \cup (x_{j_l}, v_j)$ is then included in a tree $E_j^2 \cup E_j^1$ in which nodes $u_i$ and $v_i$ are leaves. By construction, the path $\varphi_j$ (which is included in $\Phi$) goes through all the leaves of a given tree $E_j^2 \cup E_j^1$; so there exists a path between $u_i$ and $v_i$ in $\Phi$. The above argument holds for any path of consecutive vertices of $X$ encountered in $P(u,v)$. Hence, there exists a path between any pair of nodes $u$ and $v$ of $V$ in $E^0 \cup \Phi$.

In all, what we have shown is that $E^0 \cup \Phi$ is a connected spanning set of $V$, for which we can directly derive that it is heavier than the minimum spanning tree of $V$, namely $T^*$:

$$\begin{array}{rcl} w\left(E^0\right) + w\left(\Phi\right) & \geqslant & w\left(T^*\right) \\ & \Downarrow & \\ w\left(T^*\right) + w\left(E^*\right) + w\left(E^2\right) & \leqslant & \left(3 - \dfrac{2}{k-1}\right) w\left(E^2\right) + 2w\left(E^1\right) + w\left(E^0\right) \end{array}$$

We now focus ourselves on set $T^* \cup E^* \cup E^2$. Remark that set $E^2$ is a forest spanning nodes of $X$. For each connected component $E_j^2$ of $E^2$, we have identified a single edge $e_j^*$ which connects this component to a node $vs_j$ of $V$, so that for each node $x$ inside any of these components, there exists a path in $E^* \cup E^2$ connecting $x$ to a node of $V$. From this we can easily conclude that $T^* \cup E^* \cup E^2$ is a connected spanning set of $V \cup X$.

Recall that, by Lemma 1, $T_{\text{reopt}}$ is one of the minimum spanning trees of $G_x$ that includes $T^*$. So, $T^* \cup E^* \cup E^2$ being a spanning set of $G_x$ that includes $T^*$, is heavier than $T_{\text{reopt}}$:

$$\begin{array}{rcl} w\left(T_{\text{reopt}}\right) & \leqslant & w\left(T^*\right) + w\left(E^*\right) + w\left(E^2\right) \\ & \Downarrow & \\ w\left(T_{\text{reopt}}\right) & \leqslant & \left(3 - \dfrac{2}{k-1}\right) w\left(E^2\right) + 2w\left(E^1\right) + w\left(E^0\right) \end{array}$$

Using the fact that $w(T_x^*) = w(E^2) + w(E^1) + w(E^0)$, it is immediate to yield the approximation ratio claimed:

$$\frac{w\left(T_{\text{reopt}}\right)}{w\left(T_x^*\right)} \leqslant \max\left\{2,\, 3 - \frac{2}{k-1}\right\}$$

that completes the proof of Proposition 2

Remark that for $k \leqslant 3$, the approximation ratio is 2. This ratio is asymptotically tight. Consider the instance of Proposition 1. To make it metric, we just set all weights of the initial graph $G$ to 2 instead of $n+1$. The approximation ratio becomes then $2 - (1/n)$.

For $k \geqslant 4$ we prove a lower bound of $3 - 3/\sqrt{k-1} - 1/(k-1)$. This bound is attained in the following instance: consider two groups of nodes; inside each group all the edges have weight 0 while, between the two groups, all the edges have weight 4. Hence, the optimum takes only one edge with weight 4 to connect the subtrees spanning each of the two groups, both of these trees having weight 0; thus, $w(T^*) = 4$.

Five nodes are inserted which form a path between the two groups as shown in Figure 4 where, all edges that are not drawn are as heavy as the metric hypothesis allows. A new optimum is thus given by the union of the initial groups and of this path; so, $w(T_x^*) = 4$.

`REOPT1+` gathers $T^*$ and adds two edges with weight 0 (edges of $E^1$), and three edges with weight 1 to span the recently inserted nodes; thus, $w(T_x^*) = 7$.

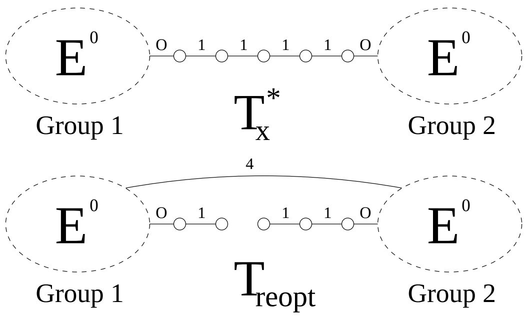

Figure 4: Lower bound for the approximation ratio of `REOPT1+` for $k = 5$.

The lower bound for the approximation ratio with $k = 5$ is then 7/4, and the upper bound is 10/4.

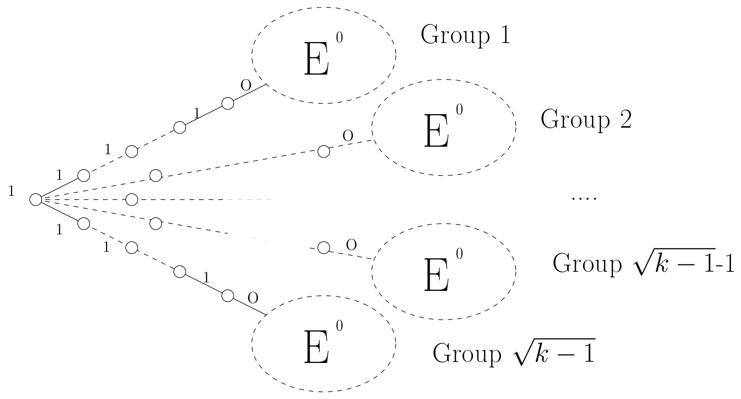

Figure 5: Lower bound for the approximation ratio of `REOPT1+` for $k \geqslant 5$.

The construction given just above can be generalized for any $k \geqslant 5$ as shown in Figure 5. The initial graph is composed of $\sqrt{k-1}$ groups of nodes; inside each group, all edges have weight 0 and between any pair of groups edges have weight $2\sqrt{k-1}$. The initial optimum includes $\sqrt{k-1}-1$ edges with weight $2\sqrt{k-1}$ to connect the several trees spanning the several groups whose weights are equal to 0. Then $k$ nodes are inserted as shown in Figure 5. Each of the $\sqrt{k-1}$ groups of initial nodes is connected to the central inserted node by a path with $\sqrt{k-1}$ new nodes. All edges not drawn in the figure are as heavy as the metric hypothesis allows. On this instance, the following result holds and concludes the section.

**Proposition 3.** *The approximation ratio of `REOPT1+` is bounded below by* $3 - (3/\sqrt{k-1}) + (1/(k-1))$.

## 2.2 Algorithm REOPT2+

We now propose the following reoptimization algorithm denoted by REOPT2+ that determines an optimal solution on $G_x$: build the sub-graph $H(V \cup X, T^* \cup_{i=1}^{k} E(x_i))$ of $G_x$ and run the Kruskal's Algorithm on it. Note that $H$ does not contain edges of $G$ that do not belong to $T^*$.

The complexity of REOPT2+ is $O((k^2 + nk)\log(n+k)$, that is better than the complexity of Kruskal's algorithm run on the entire graph $G_x$, $O((n+k)^2 \log(n+k))$.

**Proposition 4.** *REOPT2+ computes an optimal solution for* MIN SPANNING TREE$_k$+ *on* $G_x$.

In order to prove optimality of the solution computed by REOPT2+, we need the following Lemma 4.

**Lemma 4.** *A tree $T$ is a minimum spanning tree of a graph $G(V, E)$ iff for any $(a, b) \in E/T$ and for any $(x, y) \in P(a, b)$ (the unique path from $a$ to $b$ in $T$, $w(x, y) \leqslant w(a, b)$.*

**Proof.** $\Rightarrow$ If there exists an edge $(x, y)$ on some $P(a, b)$ for which $w(a, b) \leqslant w(x, y)$, then by deleting $(x, y)$ from $T$ and replacing it by $(a, b)$ to reconnect the tree, we are able to compute a spanning tree $T'$ lighter than $T$, a contradiction.

$\Leftarrow$ Let $T$ be a spanning tree of $G$ that verifies the property stated by the lemma, and assume that it is not a minimum spanning tree of $G$. Fix a minimum spanning tree $T^*$ of $G$. Since $T$ is different from $T^*$, the set $T^*/T$ is non-empty. Let $(a, b)$ be an element of this set. Assume that $T^*$ is computed by Kruskal's Algorithm. When $(a, b)$ has been inserted in the solution it did not create any cycle with the edges already inserted, so at least one edge $(x_i, y_i)$ of $P(a, b)$ (the path from $a$ to $b$ in $T$) was not inserted yet and, moreover, at least one of these edges was still a candidate edge. Indeed, if none of these edges is candidate, then insertion of any of these edges should create a cycle, so $\forall(x_i, y_i) \in P(a, b)$, there exists a path from $x_i$ to $y_i$ in the current solution of Kruskal's Algorithm, so that there should exist a path from $a$ to $b$ in this solution, and $(a, b)$ should not be a candidate edge.

Furthermore, when $(a, b)$ has been inserted in the solution, it was the lightest candidate edge, a fortiori it was lighter than at least one edge of $P(a, b)$, that contradicts the hypothesis made on the tree $T$.

So if a spanning tree $T$ verifies the property of the lemma, then $T$ is a minimum spanning tree and the proof of the lemma is completed. ▮

Based upon Lemma 4, one can claim that any edge that does not belong to $T^*$ is a heaviest edge on at least one cycle of $G$ and, consequently, also on at least one cycle in $G_x$, so that it cannot no more be part of $T_x^*$. REOPT2+ simply removes all these edges from the set of candidate edges, and runs Kruskal's Algorithm, on the surviving graph (that is smaller than $G_x$). Henceforth it is optimal for MIN SPANNING TREE$_k$+, qed.

## 3 Node deletions

We now settle a complementary problem, where modification of the instance consists of deletion of $k$ nodes. The deleted nodes form the vector $X = \{x_1, \ldots, x_k\}$. Notations are basically the same as those used in Section 2.

In this section, we propose and evaluate two reoptimization strategies, denoted by REOPT1- and REOPT2-, respectively.

Let us first fix some notations used in the sequel. The initial minimum spanning tree $T^*$ can be split into the following three distinct subsets:

1. $E^0 = T_x^* \cap V \times V$, the subset of edges of $T^*$ with no endpoint in $X$;

2. $E^1 = T^*_x \cap V \times X$, the subset of edges of $T^*$ with exactly one endpoint in $X$ and one endpoint in $V$;

3. $E^2 = T^*_x \cap X \times X$, the subset of edges of $T^*$ which with both endpoints in $X$.

All these subsets are included in $T^*$, thus they are forests. As before, we will consider the partition $\{E^2_1, E^2_2, \ldots, E^2_j, ...\}$ of the set $E^2$, where each subset $E^2_j$ is a maximal tree of the forest $E^2$. The set of edges of $E^1$ which are connected to the tree $E^2_j$ will be denoted by $E^1_j$, so that $\{E^1_1, E^1_2, \ldots, E^1_j, \ldots\}$ defines a partition of the set $E^1$. Finally, the subtrees induced by $E^2$ contain possibly non-empty sets $L_j$, which, as in Section 2, are the sets of edges that do not belong on any path connecting two non deleted nodes. Our proofs will always exclude the sets $L_j$, so our notations $E^2_j$ and $E_j$ will always mean respectively $E^2_j \setminus L_j$ and $E^2 \setminus L$.

As previously, we will denote by $X_j$ the set of deleted nodes spanned by the tree $E^2_j$, and by $V_j$ the set of non deleted nodes spanned by the set $E^1_j$.

## 3.1 Algorithm `REOPT1-`

`REOPT1-` is an easy reoptimization strategy, which computes a solution in $O(n+k)$ and works as follows:

1. for each connected component $E^2_j \cup E^1_j$, build a path $\text{reopt}_j$ connecting all the non deleted nodes spanned by $E^1_j$ as follows:
   - build a cycle connecting all nodes of $V_j$ in the order of their occurrence in a breadth-first-search (bfs) on the tree $E^2_j \cup E^1_j$;
   - delete the heaviest edge of this cycle in order to get the path $\text{reopt}_j$;
2. output $T_{\text{reopt}} = E^0 \cup_j \text{reopt}_j$.

We first show that the computed solution $T_{\text{reopt}}$ is a spanning tree. For this we prove the following three points:

Point 1. $T_{\text{reopt}}$ is a spanning set of $V \setminus X$;

Point 2. $T_{\text{reopt}}$ is connected;

Point 3. $T_{\text{reopt}}$ is acyclic.

In order to prove Point 1, observe that only nodes of $V \setminus X$ need to be spanned, so that $E^0$ (which is included in $T_{\text{reopt}}$) is already a spanning forest.

For Point 2, let us consider two nodes $v_1$ and $v_2$ of $V \setminus X$. These two nodes may be in the same connected component of $E_0$, and thus there may exist a path inside $T_{\text{reopt}}$ connecting $v_1$ and $v_2$ (since $E_0 \subseteq T_{\text{reopt}}$). If not ($v_1$ and $v_2$ are not in the same connected component of $E_0$), then at least one sequence of consecutive vertices $x_i$ of $X$ were found on the unique path from $v_1$ to $v_2$ in $T^*$. Each of these sequences is part of a set $X^2_j$ and thus, it is necessarily surrounded by two surviving nodes spanned by $E^1_j$, which are connected by the path $\text{reopt}_j$; hence, there exists a path from $v_1$ to $v_2$ in $T_{\text{reopt}}$.

Let us now prove Point 3. By construction, the existence of a path spanning a given set of nodes in $T_{\text{reopt}}$ implies the existence of a path spanning the same set of nodes (plus potentially some deleted nodes) in $T^*$. A cycle in $T_{\text{reopt}}$ can be seen as two distinct paths between a given pair of nodes. According to how `REOPT1-` works, the existence of such a cycle would imply the existence of a cycle in $T^*$, impossible.

**Proposition 5.** *The approximation ratio of* `REOPT1-` *is unbounded.*

**Proof.** We build an instance of MIN SPANNING TREE$_1-$ where the approximation ratio of `REOPT1-` is $\Theta(n)$. Let $G$ be an instance of MIN SPANNING TREE, such that in $T^*$, node $x$ has $d = n/2$ neighbors. Thus, deletion of $x$ splits $T^*$ into $d$ trees $F_i$, each of them having exactly 2 nodes: $v_i$, the neighbor of $x$, and $z_i$ a leaf of $T^*$. Weights are set as follows:

$$\begin{cases} w\left(x, v_i\right) = 1 & i \in \{1, \ldots, d\} \\ w\left(v_i, v_j\right) = n & i \in \{1, \ldots, d\},\ j \in \{1, \ldots, d\},\ i \neq j \\ w\left(v_i, z_i\right) = 1 & i \in \{1, \ldots, d\} \\ w\left(z_i, z_j\right) = 2 & i \in \{1, \ldots, d\},\ j \in \{1, \ldots, d\},\ i \neq j \end{cases}$$

Figure 6 illustrates such an instance where $w(T^*) = n + (n/2) = 3n/2$, $w(T_x^*) = 2n - 2 < 2n$ and $w(T_{\text{reopt}}) = n + n((n/2) - 1) = n^2/2$.

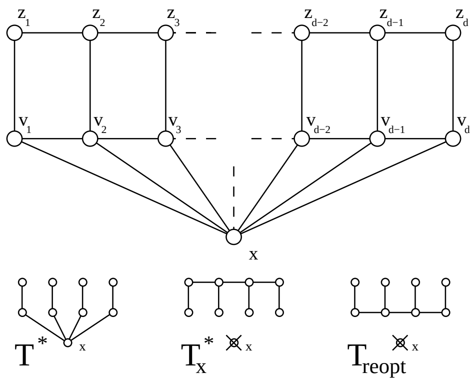

Figure 6: The construction of Proposition 5.

Thus, the approximation ratio of `REOPT1-` on this instance is equal to $n/4$, which concludes the proof. ▮

As in Section 2 we restrict, in what follows, ourselves in metric and complete graphs. We will prove the following result.

**Proposition 6.** *In metric and complete graphs,* `REOPT1-` *achieves approximation ration bounded above by* $2\lceil |L_{\max}|/2 \rceil$*, where* $L_{\max}$ *is the longest deleted path (measured in terms of edges). This bound is tight.*

Based once more upon Lemma 2, for each connected component $E_j^0 \cup E_j^1$ that is deleted, it is possible to bound the weight of the path $\text{reopt}_j$, by means of which `REOPT1-` reconnects the tree. According to Lemma 2, for any $j$:

$$w\left(\text{reopt}_j\right) \leqslant 2\left(1 - \frac{1}{\left|E_j^1\right|}\right) w\left(E_j^2 \cup E_j^1\right) \tag{5}$$

Now, let us take a look at the optimal solution $T_x^*$. This tree is the union of edges that were not deleted, which form the set $E^0$, and of sets of edges $\varphi_i^*$ that have been inserted to the new optimum $T_x^*$, while they did not belong to the initial one $T^*$.

To prove that set $E^0$ belongs to the new optimum, let us just see that according to Lemma 4, all these edges are of non-maximum weight on any cycle of $E$ to which they belong. Since the new graph is included to the initial one, all these edges remain non-maximal on any cycle to which they belong. Once again, according to Lemma 4, these edges are part of the new optimum.

Each of the edges $\varphi_i^*$ forms a cycle if added to $T^*$ and, since they were not part of the initial optimum, they are necessarily heaviest edges on these cycles. In particular, they are heavier than all the edges of $E^2 \cup E^1$ on the cycle.

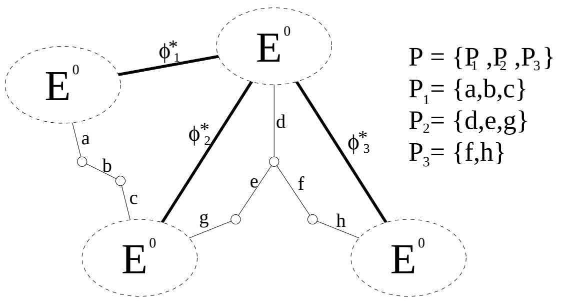

Figure 7: An example of a partition $P$ satisfying Properties 1 to 4.

We now define a partition $P = \{P_1, P_2, \dots, P_{|\Phi^*|}\}$ of the set $E^2 \cup E^1$ that verifies the following properties (an example of such a partition is given in Figure 7):

Property 1. $\cup_j P_j = E^2 \cup E^1$;

Property 2. $\forall i, j$, $P_j \cap P_i = \emptyset$;

Property 3. $\forall i$, all edges of $P_i$ belong to the cycle induced by $\varphi_i^*$ in $T^*$;

Property 4. $\forall i$, all edges of $P_i$ belong to the same connected component $E_j^2 \cup E_j^1$ of the forest $E^2 \cup E^1$.

Let us note that if we can define a set $P$ verifying all the other properties except Property 2, we just have to arbitrarily assign edges appearing to more than one $P_i$'s to one of them. So, we will first prove that a partition verifying Properties 1, 3 and 4 exists.

Combination of Properties 3 and 4 enables us to give a specific form for each set $P_i$. Property 3 forces all edges of $P_i$ to be on the same path in $T^*$, while Property 4 forces all edges of $P_i$ to be in the same connected component of the forest $E^2 \cup E^1$. Thus, a given $P_i$ can always be defined as a path between two non-deleted nodes spanning only deleted ones of the same deleted component. Notice finally that any path $P_i$ can be defined by the pair $(\varphi_i^*, E_j^2)$. Indeed, once one knows in which path of $T^*$ (namely the cycle induced by adding $\varphi_i^*$) and in which connected component (namely $E_j^2 \cup E_j^1$) the path is located, the intersection of these two sets gives exactly the path $P_i$.

Revisit the example of Figure 7 and set: $E^2_{\text{left}} = \{b\}$, $E^1_{\text{left}} = \{a, c\}$, $E^2_{\text{right}} = \{e, f\}$ and $E^1_{\text{right}} = \{d, g, h\}$. Then, we could say that the set $P_1$ is defined by the pair $(\varphi_1^*, E^2_{\text{left}})$, thus, it is the intersection of the cycle induced by $\varphi_1^*$ in $T^*$ with the set $E^2_{\text{left}} \cup E^1_{\text{left}}$. Similarly, $P_2$ is defined by the pair $(\varphi_2^*, E^2_{\text{right}})$ and $P_3$ by $(\varphi_3^*, E^2_{\text{right}})$.

So a way to define partition $P$ can be to assign each edge $\varphi_i^*$ to a unique connected component $E_j^2$. For instance, in Figure 7, $\varphi_1^*$ is assigned to the left deleted connected component, and both $\varphi_2^*$ and $\varphi_3^*$ are assigned to the right deleted connected component. This partition verifies Properties 1 to 4.

We now propose a method to assign edges $\varphi_i^*$ to a component $E_j^2$. Observe first that a given edge $\varphi_i^*$ cannot be assigned to any component $E_j^2$ but only to some component that contains edges of the cycle that $\varphi_i^*$ induces when added to $T^*$. In the example of Figure 7, $\varphi_3^*$ cannot be attributed to the left component, since its induced cycle only contains edges $d$, $f$ and $h$ as deleted edges, none of which are in the left component $\{a, b, c\}$. The method works as follows:

- identify, for each deleted component, the list of candidate edges $\varphi_i^*$;
- for each deleted component, identify the set $E_j^1$ that contains all edges of $T^*$ adjacent to one deleted node of $E_j^2$ and to one non-deleted node (in Figure 7, $E_{\text{left}}^1 = \{a, b\}$ and $E_{\text{right}}^1 = \{d, g, h\}$);
- if for a given deleted component $E_j^2 \cup E_j^1$, the list $L$ of candidate edges contains exactly $|E_j^1| - 1$ elements, then delete these elements from the other lists of candidate edges; the edges of $L$ are definitely attributed to the component $E_j^2 \cup E_j^1$;
- if no component verifies the above condition (so, if each component $E_j^2 \cup E_j^1$ has at least $|E_j^1|$ edges in its candidate set), then delete randomly a redundant edge from one of the lists;
- repeat the two previous steps until the list of candidate edges contains exactly $|E_j^1| - 1$ elements for each deleted component.

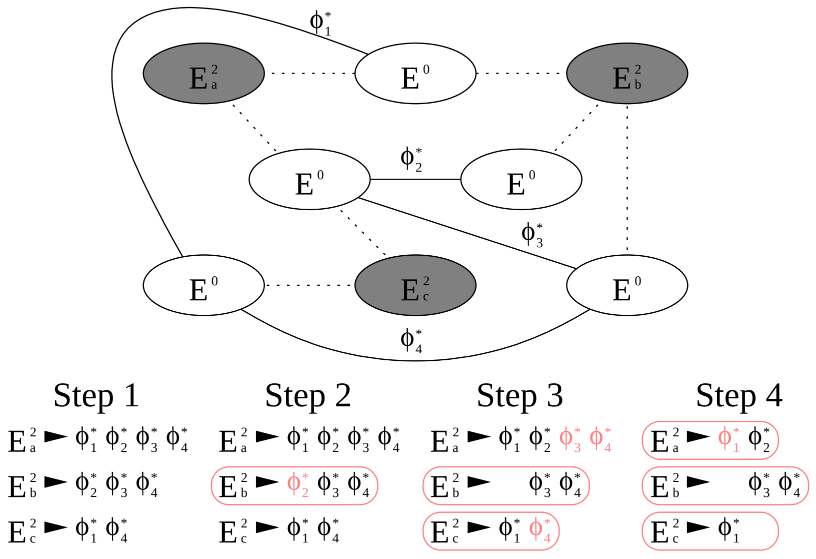

Figure 8: An application of the assignment method.

Figure 8, where edges $\varphi_i^*$ removed from candidate sets are written in grey, provides an example of the above method:

Step 1: all candidate edges are listed for each deleted tree; we recall that a given edge $\varphi_i^*$ is a candidate edge for a given deleted component, if and only if the cycle induced by $\varphi_i^*$ contains edges of the component; sets $E_j^1$ (which are the cuts associated to sets $E_j^2$) are identified, so we get $|E_a^1| = 2$, $|E_b^1| = 3$ and $|E_c^1| = 2$;

Step 2: since no candidate list contains exactly $|E_j^1| - 1$ edges $j = a, b, c$, one redundant edge is randomly chosen for deletion in one list, for instance, $\varphi_2^*$ (which becomes grey in the list of $E_b^2$ in Figure 8); now this list contains exactly $|E_b^1| - 1 = 2$ elements (recall that, at the beginning, $|E_b^1| = 3$); so, the edges of the candidate set are definitely attributed to the component: the fixed candidate set is then displayed in a circle in Figure 8;

Step 3: all the edges of the fixed list (namely, $\varphi_3^*$ and $\varphi_4^*$) are then removed from all the other lists in which they appear and are also displayed in grey; now the list of $E_c^2$ contains exactly $|E_c^1|-1=1$ elements, so the edges of the candidate set are definitely attributed to the component displayed as mentioned in Step 2 above;

Step 4: the only edge of the fixed list (namely, $\varphi_1^*$) is removed from all the other lists in which it appears, and is also displayed in grey; now the list of $E_a^2$ contains exactly $|E_a^1|-1=1$ elements; the edges of the candidate set are definitely attributed to the component once more displayed as above; now all candidate sets contain $|E_j^1|-1$ edges.

Let us recall that the aim of the assignment that we are describing is to define a partition of the deleted edges, so that each edge of $\varphi_i^*$ is heavier than all the deleted edges of its corresponding set in the partition. To do so, we consider the cycle $C_i$ induced by adding $\varphi_i^*$ in the initial tree, so that all the edges of $C_i$ are lighter than $\varphi_i^*$ (or else $\varphi_i^*$ would have been an element of the initial optimal tree). Then, if the edge $\varphi_i^*$ is attributed to the deleted component $E_j^2 \cup E_j^1$, the set of the partition corresponding to $\varphi_i^*$ will be $C_i \cap (E_j^2 \cup E_j^1)$.

We must now prove that all the edges of $E_j^2 \cup E_j^1$ will be part of at least one set of the partition (redundant elements can be deleted later to define a real partition).

Let us first prove that the assignment method attributes all the edges of $\varphi_i^*$ to a deleted component. To understand this, we must first remark that it attributes $|E_j^1|-1$ edges to each deleted component $E_j^2 \cup E_j^1$. Let $|CC^2|$ be the number of deleted components. Summing the number of attributed edges for each deleted component we get:

$$\sum_{j=1}^{CC^2} \left(|E_j^1|-1\right) = \sum_{j=1}^{CC^2} \left(|E_j^1|\right) - |CC^2| = |E^1| - |CC^2|$$

As it can be seen in Figure 8, the edges of $E^1$ form an arborescence spanning connected components of both $E^2$ and $E^0$. Since there exist $|CC^2|$ and $|CC^0|$ connected components in $E^2$ and $E^0$, respectively, we can derive that $E^1 = |CC^2|+|CC^0|-1$. So, at the end of the procedure, the total number of attributed edges is:

$$|E^1| - |CC^2| = |CC^2| + |CC^0| - 1 - |CC^2| = |CC^0| - 1$$

We now count the number of edges of $\varphi_i^*$. There exist $|CC^0|$ non-deleted components that need to be reconnected. As before, we can thus affirm that there exist $|CC^0|-1$ edges $\varphi_i^*$. So, the number of attributed edges is the same as the number of edges $\varphi_i^*$. Taking into account that no edge can be attributed to more than one deleted component, leads us to the following fact.

**Fact 1.** All edges $\varphi_i^*$ are attributed by the assignment method. ∎

Furthermore, this method will never attribute two edges inducing the same path to a given component. It is very likely that two edges $\varphi_i^*$ and $\varphi_l^*$ induce two cycles $C_i$ and $C_l$, so that for a given deleted component $E_j^2 \cup E_j^1$, the induced sets for the partition are exactly the same, i.e., $C_i \cap (E_j^2 \cup E_j^1) = C_l \cap (E_j^2 \cup E_j^1)$. For instance, in the example of Figure 8, $\varphi_1^*$, $\varphi_2^*$, $\varphi_3^*$ and $\varphi_4^*$ induce the same path in $E_a^2 \cup E_a^1$.

But really, if two or more edges induce the same path in a given component, this means that in the first step of the assignment method, the initial candidate set is bigger than the set which will be finally defined by the method.

Revisit once more the example of Figure 8. The final set attributed to $E_a^2 \cup E_a^1$ will contain only one edge, for $|E_a^1| = 2$ (recall that the cardinality of a set finally attributed to a given component $E_j^2 \cup E_j^1$ is $|E_j^1| - 1$), and there exist 4 candidate edges, which all induce the same path in $E_a^2 \cup E_a^1$. Thus, 3 out of these 4 candidate edges will be deleted from the list.

Assume that two edges $\varphi_i^*$ and $\varphi_l^*$ induce the same path for a given component $E_j^2 \cup E_j^1$. Then:

- the candidate list of $E_j^2 \cup E_j^1$ will have at least one element deleted, before the end of the assignment procedure;
- either $\varphi_i^*$, or $\varphi_l^*$ appears in at least one other list.

This is shown in the gadget of Figure 9.

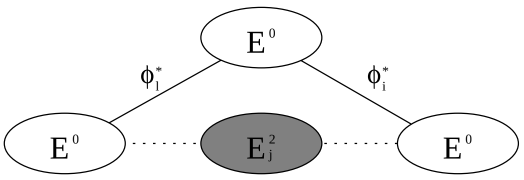

Figure 9: Edges $\varphi_i^*$ and $\varphi_l^*$ inducing the same path in $E_j^2 \cup E_j^1$.

Since the initial tree is connected, there exists at least one other deleted component for which either $\varphi_i^*$, or $\varphi_l^*$ is a candidate edge at the beginning of the assignment procedure. So, any deleted component has at least a redundant edge in its candidate set and in order that it has a list with exactly $|E_j^1| - 1$ elements, at least one edge must be removed from its initial list. In the assignment procedure, edges are progressively removed from the lists in two ways: either randomly, or when a set attains its appropriate number of edges. So, according to our hypothesis, either $\varphi_i^*$, or $\varphi_l^*$ will be removed. Discussion just above induces the following fact.

**Fact 2.** Two edges inducing the same path in a given component will never be both attributed to this component. ▮

Following the discussion above, the set of edges assigned to a given deleted component, can be seen as a "pseudo"-spanning tree on the non deleted nodes incident to this component. Indeed, a given deleted component $E_j^2 \cup E_j^1$ has exactly $|E_j^1|$ incident non deleted nodes. Each edge assigned to this component induces a cycle crossing two non-deleted nodes spanned by $E_j^1$. According to Fact 1, exactly $|E_j^1| - 1$ edges are assigned to this component, and according to Fact 2, each edge induces a distinct pair of non deleted nodes. Since the possibility of a cycle is clearly excluded, the set of edges assigned to a given deleted component can be seen as a pseudo spanning tree on the non deleted nodes incident this component. But, in fact, edges of $\varphi_i^*$ are not incident to the nodes that they span in the pseudo-spanning tree (this is why we call it "pseudo"-spanning). This idea is illustrated in Figure 10, where $\varphi_1^*$ "spans" $v_1$ and $v_2$, $\varphi_2^*$ "spans" $v_1$ and $v_3$, $\varphi_3^*$ "spans" $v_1$ and $v_5$ and, finally, $\varphi_4^*$ spans $v_4$ and $v_5$. This enables us to claim that each edge of $E_j^2 \cup E_j^1$ lies on a cycle induced by one $\varphi_i^*$ assigned to this component. Thus, the partition $P$ so-obtained, that already verifies Properties 3 and 4, will cover all edges of $E^2 \cup E^1$, verifying so Property 1 also.

Indeed, let $e$ be an edge of $E^2 \cup E^1$. If it is deleted from the component, then this component is obviously split into two parts. Let $V_1$ and $V_2$ be the set of leaves of the two trees resulting from this splitting. If $e$ does not lie on any cycle induced by any edge $\varphi_i^*$ attributed to the component, then there exists no edge $\varphi_i^*$ of the pseudo spanning tree between the set $V_1$ and $V_2$, impossible since this tree is a (pseudo) spanning tree.

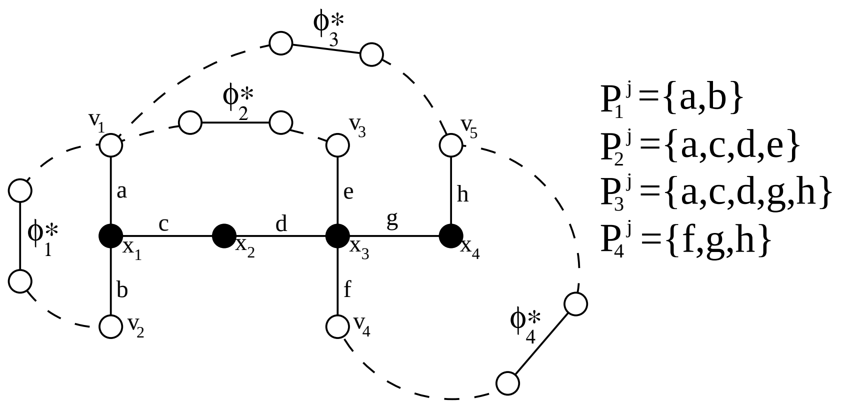

Figure 10: A covering $P^j$.

In all, we have proved that a partition $P$ verifying Properties 1 to 4 always exists, since it can be generated by the assignment method on any instance.

Using Property 3 and taking into account that each of the edges $\varphi_i^*$ was not part of the initial optimum, thus each of them is necessarily a heaviest edge on the cycles they induce in $T^*$, we derive that, for any $i$, $w(\varphi_i^*) \geqslant w(P_i)/|P_i|$.

Using Properties 1 and 2, we can derive that, for a given deleted component $E_j^2 \cup E_j^1$, the set of edges $\varphi_i^*$ attributed to it, verifies:

$$\sum w\left(\varphi_i^*\right) \geqslant \frac{w\left(E_j^2 \cup E_j^1\right)}{|P_{\max}|} \tag{6}$$

where $P_{\max}$ is the set with the biggest cardinality in partition $P$.

Finally, using Property 4, we can affirm that $|P_{\max}|$ is bounded by the length of the longest path $L_{\max}$ inside a deleted component (for instance, $|L_{\max}| = 2$ if all deleted nodes are not adjacent in $T^*$).

With Properties 1, 3, 4, $P$ is not a partition but rather a covering. But, as we have already mentioned, such a covering can be easily transformed into a partition by simply removing multiple occurrences of elements within all but one set of $P$. But there exists, in fact, a better way to perform this transformation in order to get the upper bound claimed in proposition's statement.

For a given deleted component $E_j^1 \cup E_j^2$, denote by $P^j$ the projection of $P$ on $E_j^1 \cup E_j^2$ (obviously, $P = \cup_j P^j$); $P^j$ is of the form: $P^j = (P_1^j, P_2^j, \ldots, P_i^j, \ldots, P_{|E_j^1|-1}^j)$. Recall that each set $P_i^j$ is a path of deleted edges between two non-deleted nodes. Observe also that $\cup_i P_i^j = E_j^1 \cup E_j^2$ is a (non-rooted) tree. For instance, consider Figure 10. There, $E_j^2 = \{a, b\}$, $E_j^1 = \{c, d, e, f, g\}$ and $|E_j^1| - 1 = 4$ edges $\varphi_i^*$ have been assigned to the component $E_j^2 \cup E_j^1$, each of them inducing a cycle going through the component and encounters some path composed of removed edges between two non-removed vertices (i.e., the non-endpoints of the path are all removed vertices). These paths correspond to the sets $P_i^j$ of $P^j$.

Transformation of $P^j$ into a partition consists of turning the deleted component into a rooted tree, by considering as root the median node of a longest path in the tree (one of the two "medians" if the length of the longest path is odd). For instance, $x_2$, or $x_3$ in Figure 10, that are the two medians on the longest path $P(v_1, v_5)$. Note that in this rooted tree, the longest oriented path encounters at most $\lceil |L_{\max}|/2 \rceil$ edges. Indeed, if it encounters more edges, then there exists a path between two nodes with more than $|L_{\max}|$ edges, impossible by definition of $L_{\max}$.

This rooted tree forms a new covering $R^j = (R^j_1, R^j_2, \ldots, R^j_{|E^1_j|})$ of the deleted component where the set $R^j_m$ corresponds to the oriented path between the root and the leaf $v^j_m$, the length of which is at most $\lceil |L_{\max}|/2 \rceil$. Each leaf $v^j_m$ is an endpoint of at least one path $P^j_i$ of $P^j$. The set of paths that have $v^j_m$ as endpoint will be referred to as the "candidate set" of $v^j_m$. Then, we can match $R^j$ and $P^j$ as follows:

1. pick a leaf $v^j_m$ whose candidate set has one only path, say, $P^j_i$;
2. replace the set $R^j_m$ by $P^j_i \cap R^j_m$;
3. remove all edges of $R^j_m$ from the other sets of $R^j$;
4. remove $P^j_i$ from $P^j$, and update the candidate sets;
5. if $P^j \neq \emptyset$, then go back to step 1, else exit.

Note that what procedure above gets is a partition $R^j$ with $|E^1_j|$ sets, corresponding each to at least one of the $|E^1_j| - 1$ edges $\varphi^*_i$ attributed to the partition. This partition preserves all properties of the former partition $P^j$, since all of its elements are subsets of sets in $P^j$. For the partition $R^j$, let $R^j_{\min}$ be the lightest set (in terms of weight). Then:

$$w\left(R^j_{\min}\right) \leqslant \frac{w\left(R^j\right)}{\left|E^1_j\right|}$$

Consequently, and taking into account that $R^j$ is a partition on the deleted tree $E^1_j \cup E^2_j$:

$$w\left(R^j \setminus R^j_{\min}\right) \geqslant \left(1 - \frac{1}{\left|E^1_j\right|}\right) w\left(E^1_j \cup E^2_j\right)$$

Consider partition $R^j \setminus R^j_{\min}$; it contains now $|E^1_j| - 1$ sets. Each set there is a subset of the cycle induced by an edge $\varphi^*_i$ in the initial optimal solution. Using Property 3 and taking into account that each of the edges $\varphi^*_i$ was not part of the initial optimum (thus are necessarily heaviest edges on the cycles created when added to this optimum), we get that, for any $i$, $w(\varphi^*_i) \geqslant w(R_i)/|R_i|$.

Since the cardinality of any set $R_i$ is bounded above by $\lceil |L_{\max}|/2 \rceil$, summing for all $\varphi^*_i$ attributed to a given component, we get:

$$\sum_i w\left(\varphi^*_i\right) \geqslant \frac{w\left(R^j \setminus R^j_{\min}\right)}{\left\lceil \frac{|L_{\max}|}{2} \right\rceil} = \left(1 - \frac{1}{\left|E^1_j\right|}\right) \frac{w\left(E^1_j \cup E^2_j\right)}{\left\lceil \frac{|L_{\max}|}{2} \right\rceil} \tag{7}$$

Combining (5) and (7), we are able to formulate a bound between the optimal reconnecting set, and that generated by `REOPT1-` for each deleted component $E^2_j \cup E^1_j$:

$$\frac{w\left(\mathrm{reopt}_j\right)}{\sum\limits_i w\left(\varphi^*_i\right)} \leqslant 2\left\lceil \frac{|L_{\max}|}{2} \right\rceil \tag{8}$$

where in (8) we only take into account edges $\varphi^*_i$ assigned to $E^2_j \cup E^1_j$.

Finally, the approximation ratio of the algorithm REOPT1- is:

$$\frac{w\left(T_{reopt}\right)}{w\left(T^*_x\right)} = \frac{w\left(E^0\right) + \sum\limits_j w\left(\mathrm{reopt}_j\right)}{w\left(E^0\right) + \sum\limits_i w\left(\varphi^*_i\right)} \leqslant \frac{\sum\limits_j w\left(\mathrm{reopt}_j\right)}{\sum\limits_i w\left(\varphi^*_i\right)} \overset{(8)}{\leqslant} 2\left\lceil \frac{|L_{\max}|}{2} \right\rceil$$

as claimed. This bound is equal either to $|L_{\max}|$, or to $|L_{\max}| + 1$.

We now show tightness of the bound given in (8). Let $I$ be an instance of MIN SPANNING TREE. The $k$ nodes that are deleted form a unique connected component, that is a star of paths. The terminal node of each branch of the star is a non-deleted node. The star has $2(k-1)/\sqrt{k}-2$ branches, each one having $(\sqrt{k}-2)/2$ nodes. So, the total number of nodes on all branches is:

$$2\frac{k-1}{\sqrt{k}-2}\frac{\sqrt{k}-2}{2} = k-1$$

Adding the central node of the star, we finally find the $k$ deleted nodes in the connected component.

Since each branch has $(\sqrt{k}-2)/2$ nodes, it has $\sqrt{k}/2$ edges, and thus, the longest deleted path between two non-deleted nodes has $\sqrt{k}$ edges (in other words, $|L_{\max}| = \sqrt{k}$). All these edges have weight 1; thus, each branch of the star has weight $\sqrt{k}/2$.

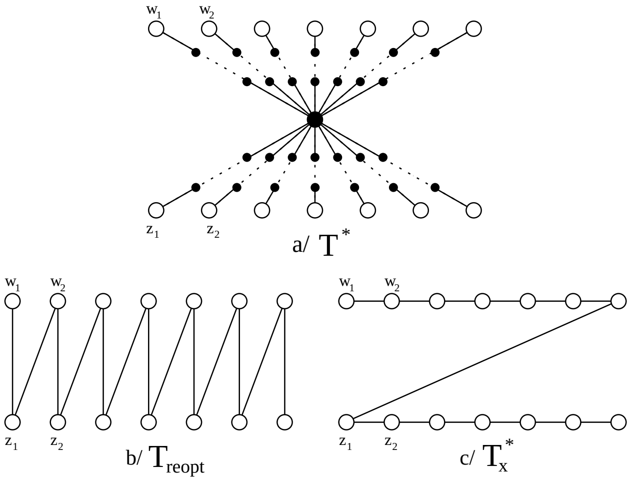

Figure 11: Lower Bound for the approximation ratio of REOPT1-.

Figure 11 shows a simplified version of the instance. In Figure 11(a), the deleted paths, each of them having $\sqrt{k}/2$ edges are represented (these are the paths between the central black node and the – non-deleted – white nodes $w_i$ and $x_i$). White nodes are non-deleted nodes incident to deleted ones. Figure 11(b) represents the solution computed by algorithm REOPT1-. Each edge there has weight exactly $\sqrt{k}$, so that the metric hypothesis is verified (with respect to the edges of Figure 11(a)). Recall that REOPT1- computes a path connecting the non deleted nodes incident to each deleted component, in the order of their occurrence in a bfs applied on the deleted component. So in this instance, any path connecting all the non-deleted white nodes is a possible solution for REOPT1-; thus, so is the path of Figure 11(b). Finally, Figure 11(c) represents the optimal solution on the modified instance. Any edge there has weight 1. Each of them is as heavy as any deleted edge, thus they do not belong to the initial optimum.

The metric hypothesis is also verified between the edges of Figures 11(b) and (c). Indeed, in the latter figure, there exist exactly $2(k-1)/\sqrt{k}-2$ non-deleted connected components (in white), half of them laying in the upper part of the figure and the other half in the lower part. Pick any edge $(w, z)$ of Figure 11(b); we have to show that this edge is at least as light as the unique path connecting $w$ to $z$ in Figure 11(c), so that the graph verifies the metric hypothesis. Given the structure of the optimal tree $T_x^*$, any path connecting two nodes $w$ and $z$ (these two nodes being neighbors in $T_{\text{reopt}}$) will encounter at least $(k-1)/(\sqrt{k}-2)-1$ edges. For instance, the path between $w_1$ and $z_1$ in Figure 11(c) includes all the edges in the upper part (namely, $(k-1)/(\sqrt{k}-2)-1$ edges), plus the transversal edge, in all $(k-1)/(\sqrt{k}-2)$ edges. The path between $w_2$ and $z_1$ has one less edge (namely, the edge $(w_1, w_2)$), thus exactly $(k-1)/(\sqrt{k}-2)-1$ edges. Then, between $w_2$ and $z_2$, the number of edges turns into $(k-1)/(\sqrt{k}-2)-1$, and so on. Since each edge has weight 1 in $T_x^*$, any path connecting two nodes $w$ and $z$ there will have weight at least $(k-1)/(\sqrt{k}-2)+1$. Finally, taking into account that $\sqrt{k} \leqslant (k-1)/(\sqrt{k}-2)-1$, one immediately derives that the instance handled is metric.

Now, let us assume that the edges within any non-deleted components have weight equal to 0. Then the tree $T_x^*$ encounters $2(k-1)/(\sqrt{k}-2)-1$ edges of weight 1, and the tree $T_{\text{reopt}}$ encounters $2(k-1)/(\sqrt{k}-2)-1$ edges of weight $\sqrt{k}$. So, the approximation ratio on this instance is $\sqrt{k} = |L_{\max}|$. The proof of Proposition 6 is now completed.

**Algorithm** `REOPT2-`

In this section we devise an optimal reoptimization strategy for MIN SPANNING TREE$_k-$. It is denoted by `REOPT2-` and works as follows:

- contract each non deleted connected component $E_i^0$ of $T^*$ into a single node $y_i$, link nodes $y_i$ and $y_j$ with one of the lightest edges linking a node of $E_i^0$ to a node of $E_j^0$ and run Kruskal's Algorithm on the resulting graph; let $\varphi_{\text{reopt}}$ the solution obtained;
- output $T_{\text{reopt}} = E^0 \cup \varphi_{\text{reopt}}$.

The complexity of this algorithm is $O((m-|E^{0|})\log(n-k-|E^0|))$. Obviously, when $|E^0| = 0$, `REOPT2-` is exactly the same as an application of Kruskal's Algorithm on the modified instance. However, as soon as $|E^0| \neq 0$, `REOPT2-` provides a better complexity than Kruskal's Algorithm.

**Proposition 7.** *`REOPT2+` computes an optimal solution for* MIN SPANNING TREE$_k-$.

**Proof.** According to Lemma 4, every edge of $E^0$ belongs to $T_x$. Now let us show that any edge of $\varphi_{\text{reopt}}$ also belongs to $T_x$.

Suppose that an edge $e$ of $T_{\text{reopt}}$ is heaviest on one cycle $C$ of $G_x$. This edge is necessarily an edge of $\varphi_{\text{reopt}}$ since all edges of $E^0$ belong to $T_x$ and therefore, they cannot be heaviest on any cycle of $G_x$. The existence of $C$ induces the existence of a cycle $C'$ in the graph resulting from the contractions of all connected components of $E^0$, and $e$ is also heaviest on $C'$. But, from Lemma 4, such an edge would not be included in $\varphi_{\text{reopt}}$. ∎

## 4 Conclusion

In this paper we have established approximation results for MIN SPANNING TREE under reoptimization setting. We have proposed simple and fast reoptimization strategies tackling light modifications of the input graph that is some vertex insertions and deletions. What is of interest here is that an asymmetry is observed between these two types of modifications, the latter seeming "harder" than the former. Proposing new more efficient algorithms at least for vertex deletions is a subject of ongoing research.

Study of MIN SPANNING TREE in metric and complete graphs exhibit, for both types of modifications, gaps between upper and lower approximation bounds. Another interest subject to be studied is bridging this gap either by reducing upper bounds, or by strengthening lower ones.

Finally, following our approach, an interesting generalization would be to consider MIN SPANNING TREE in a fully dynamic situation. Starting from a given optimal solution on an initial graph, the graph evolves in booth senses (nodes are added and deleted), and the goal is to maintain efficiently, along this process, an approximate solution as good as possible. This is also an interesting subject of forthcoming research.

## References

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

[2] G. Ausiello, B. Escoffier, J. Monnot, and V. Th. Paschos. Reoptimization of minimum and maximum traveling salesman's tours. In L. Arge and R. Freivalds, editors, *Proc. Scandinavian Workshop on Algorithm Theory, SWAT'06*, volume 4059 of *Lecture Notes in Computer Science*, pages 196–207. Springer-Verlag, 2006.

[3] M. Bartusch, R. H. Mohring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Ann. Oper. Res.*, 16:201–240, 1988.

[4] M. Bartusch, R. H. Möhring, and F. J. Radermacher. Design aspects of an advanced model-oriented DSS for scheduling problems in civil engineering. *Decision Support Systems*, 5:321–344, 1989.

[5] H. J. Böckenhauer, L. Forlizzi, J. Hromkovic, J. Kneis, J. Kupke, G. Proietti, and P. Widmayer. On the approximability of TSP on local modifications of optimally solved instances. *Algorithmics Oper. Res.*, 2(2):83–93, 2007.

[6] H. J. Böckenhauer, J. Hromkovic, T. Mömke, and P. Widmayer. On the hardness of reoptimization. In *Proc. SOFSEM 2008—Theory and Practice of Informatics*, volume 4910 of *Lecture Notes in Computer Science*, pages 50–65. Spinger-Verlag, 2008.

[7] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification - a technique for speeding up dynamic graph algorithms. *J. Assoc. Comput. Mach.*, 44(5):669–696, 1997.

[8] B. Escoffier, M. Milanič, and V. Th. Paschos. Simple and fast reoptimizations for the steiner tree problem. Cahier du LAMSADE 245, LAMSADE, Université Paris-Dauphine, 2007. Available at `http://www.lamsade.dauphine.fr/cahiers/PDF/cahierLamsade245.pdf`.

[9] M. R. Henzinger and V. King. Maintaining minimum spanning trees in dynamic graphs. In *Proc. ICALP'97*, Lecture Notes in Computer Science, pages 594–604. Springer-Verlag, 1997.

[10] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.*, 7(1):48–50, 1956.

[11] M. W. Schäffter. Scheduling with forbidden sets. *Discrete Appl. Math.*, 72(1-2):155–166, 1997.