Theoretical Computer Science 562 (2015) 643–657

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Specializations and generalizations of the Stackelberg minimum spanning tree game[☆]

Davide Bilò[a,*], Luciano Gualà[b,*], Stefano Leucci[c,*], Guido Proietti[c,d,*]

[a] *Dipartimento di Scienze Umanistiche e Sociali, University of Sassari, Italy*
[b] *Dipartimento di Ingegneria dell'Impresa, University of Rome "Tor Vergata", Italy*
[c] *Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, University of L'Aquila, Italy*
[d] *Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", CNR, Rome, Italy*

ARTICLE INFO

Article history:
Received 28 February 2014
Received in revised form 26 September 2014
Accepted 6 November 2014
Available online 13 November 2014
Communicated by G. Ausiello

Keywords:
Communication networks
Minimum spanning tree
Stackelberg games
Network pricing games

ABSTRACT

Let be given a graph $G = (V, E)$ whose edge set is partitioned into a set R of *red* edges and a set B of *blue* edges, and assume that red edges are weighted and form a spanning tree of G. Then, the *Stackelberg Minimum Spanning Tree* (StackMST) problem is that of pricing (i.e., weighting) the blue edges in such a way that the total weight of the blue edges selected in a minimum spanning tree of the resulting graph is maximized. StackMST is known to be APX-hard already when the number of distinct red edge weights is 2. In this paper we analyze some meaningful specializations and generalizations of StackMST, which shed some more light on the computational complexity of the problem. More precisely, we first show that if G is restricted to be *complete*, then the following holds: (i) if there are only 2 distinct red edge weights, then the problem can be solved optimally (this contrasts with the corresponding APX-hardness of the general problem); (ii) otherwise, the problem can be approximated within $7/4 + \epsilon$, for any $\epsilon > 0$. Afterwards, we define a natural extension of StackMST, namely that in which blue edges are associated with a non-negative *activation cost*, and it is given a global *activation budget* that can be used (and must not be exceeded) in order to activate a subset of blue edges to be priced. Here, after showing that the very same approximation ratio as that of the original problem can be achieved, we prove that if the spanning tree of red edges can be rooted so as that any root-leaf path contains at most h edges, then the problem admits a $(2h + \epsilon)$-approximation algorithm, for any $\epsilon > 0$.

1. Introduction

Leader–follower games, which were introduced by von Stackelberg in the far 1934 [15], have recently received a considerable attention from the computer science community. This is mainly due to the fact that the Internet is a vast electronic market composed of millions of independent end-users (i.e., the followers), whose actions are influenced by a limited number of owners of physical/logical portions of the network (i.e., the leaders), that can set the price for using their own

[☆] A preliminary version of this paper was published in the *Proceedings of the 6th International Workshop on Internet and Network Economics (WINE 2010)*, and appeared in vol. 6484 of Lecture Notes in Computer Science, Springer, pp. 75–86. This work is partially supported by the Research Grant PRIN 2010 "ARS TechnoMedia" (Algorithms for Techno-Mediated Social Networks), funded by the Italian Ministry of Education, University, and Research.
[*] Corresponding authors.
E-mail addresses: davidebilo@uniss.it (D. Bilò), guala@mat.uniroma2.it (L. Gualà), stefano.leucci@univaq.it (S. Leucci), guido.proietti@univaq.it (G. Proietti).

http://dx.doi.org/10.1016/j.tcs.2014.11.009
0304-3975/© 2014 Elsevier B.V. All rights reserved.

network links. In particular, in a scenario in which the leaders know in advance that the followers will allocate a communication subnetwork enjoying some criteria, a natural arising problem is that of analyzing how the leaders can optimize their pricing strategy. Games of this latter type are widely known as *Stackelberg Network Pricing Games* (SNPGs).

When only 2 players (i.e., a leader and a follower) are involved, an SNPG can be formalized as follows: We are given a connected graph $G = (V, E)$ with $|V| = n$ vertices, whose edge set E is partitioned into a set R of *red* edges and a set B of *blue* edges, and an edge cost function $c : R \to \mathbb{R}^+ \cup \{0\}$ for red edges only, while blue edges need instead to be priced by the leader. The leader moves first and chooses a pricing function $p : B \to \mathbb{R}^+ \cup \{0\}$ for her[1] edges, in an attempt to *maximize* her objective function $f_1(p, H(p))$, where $H(p)$ denotes the decision which will be taken by the follower, consisting in the choice of a subgraph of G. This denotation stresses the fact that the leader's problem is implicit in the follower's decision. Once observed the leader's choice, the follower reacts by selecting a subgraph $H(p) = (V', E')$ of G which *minimizes* his objective function $f_2(p, H)$, parameterized in p. Note that the leader's strategy affects both the follower's objective function and the set of feasible decisions, while the follower's choice only affects the leader's objective function. Quite naturally, we assume that f_1 is *price-additive*, i.e., $f_1(p, H(p)) = \sum_{e \in B \cap E'} p(e)$. This means, the leader decides edge prices having in mind that her revenue equals the overall price of her selected edges. Therefore, the 2-player game can be equivalently thought (as we will do in the rest of the paper) as a *bilevel* optimization problem in which an optimal value of f_1 has to be computed.

Previous work The most immediate SNPG is that in which we are given two specified nodes in G, say s, t, and the follower wants to travel along a *shortest path* in G between s and t (see [14] for a survey). This problem has been shown to be APX-hard [10], as well as not approximable within a factor of $2 - o(1)$ unless P = NP [6], while an $O(\log |B|)$-approximation algorithm is provided in [12]. For the case of multiple followers (each with a specific source-destination pair), Labbé et al. [11] derived a bilevel LP formulation of the problem (and proved NP-hardness), while Grigoriev et al. [9] presented algorithms for a restricted shortest path problem on parallel edges. Furthermore, when all the followers share the same source node, and each node in G is a destination of a single follower, then the problem is known as the *Stackelberg single-source shortest paths tree* game. In this game, the leader's revenue for each selected edge is given by its price multiplied by the number of paths – emanating from the source – it belongs to, and in [2] it was proved that finding an optimal pricing for the leader's edges is NP-hard, as soon as $|B| = \Theta(n)$.

Another basic SNPG, which is of interest for this paper, is that in which the follower wants to use a *minimum spanning tree* (MST) of G. For this game, known as *Stackelberg MST* (StackMST) game, in [7] the authors proved the APX-hardness already when the number of distinct red edge costs is 2, and gave a $\min\{k, 1 + \ln \beta, 1 + \ln \rho\}$-approximation algorithm, where k is the number of distinct red edge costs, β is the number of blue edges selected by the follower in an optimal pricing, and ρ is the maximum ratio among pairs of red edge costs. In a further paper [8], the authors proved that the problem remains NP-hard even if G is planar, while it can be solved in polynomial time once G has bounded treewidth. We point out that a structural property about StackMST, which will also hold for our generalized version we are going to present, is that the hardness in finding an optimal solution lies in the selection of the optimal set of blue edges that will be purchased by the follower, since once a set of blue edges is assumed to be part of the final MST, then their best possible pricing can be computed in polynomial time, as claimed in [7], and as we now explicitly show in the technical part of this paper. Notice this also implies that the problem is fixed-parameter tractable once that the parameter is assumed to be the number of blue edges.

All the above examples fall within the class of SNPGs handled by the general model proposed in [4], encompassing all the cases where each follower aims at optimizing a polynomial-time network optimization problem in which the cost of the network is given by the sum of prices and costs of contained edges. Nevertheless, SNPGs for models other than this one have been studied in [3,5].

Our results In this paper we analyze a generalization and some meaningful specializations of StackMST, thus shedding some more light on the computational complexity of the game. For the sake of presenting our results in a unifying framework, we start by defining the aforementioned generalized version of StackMST.

First of all, notice that the computational complexity of StackMST is fully characterized by the instances in which the set of red edges forms a spanning tree of G. Indeed, on the one hand, if the subgraph of G induced by R, say $G[R]$, is not connected, then blue edges connecting two red components can receive unbounded prices. On the other hand, if $G[R]$ is not a tree, let R' be a subset of edges of R inducing a red MST of G. Then, it is easily seen that the discarded red edges do not influence the strategy of the leader, since for any pricing of the blue edges, there exists a MST of G that does not contain any discarded red edge. Thus, in the rest of the paper, we will focus ourselves on the case in which edges in R form a spanning tree of G. In a practical network application, this red MST may reasonably model an existing communication infrastructure, while the blue edges can be seen as *potential* links that the leader is able to activate. Clearly, it may well be that the cost of activating these edges is heavily context-dependant, but unfortunately the classic definition of the StackMST problem is unable to model this scenario, and so we need to extend it in order to remove this shortage. Quite obviously, and still in a perspective of being as close to a real situation as possible, one has also to put a bound on the capability for the leader of activating links. Then, more formally, the 2-player *budgeted* StackMST game can be defined as follows. We

[1] Throughout the paper, we adopt the convention of referring to the leader and to the follower with female and male pronouns, respectively.

are given a tree $T = (V, E(T))$ of n nodes where each (red) edge $e \in E(T)$ has a fixed non-negative cost $c(e)$. Moreover, we are given a non-negative *activation cost* $\gamma(e)$ for each (blue) edge $e = (u, v) \notin E(T)$, and a budget Δ. The game, denoted by Budget-StackMST in the following, consists of two phases. In the first phase the leader selects a set F of edges to add to T such that the budget is not exceeded, i.e., $\sum_{e \in F} \gamma(e) \le \Delta$, and then prices them with a price function $p : F \to \mathbb{R}^+$ having in mind that, in the second phase, the follower will take the weighted graph $G = (V, E(T) \cup F)$ resulting from the first phase, and will compute an MST $M(F, p)$ of G. Then, the leader will collect a revenue of $r(M(F, p)) = \sum_{e \in F \cap M(F,p)} p(e)$. Our goal is to find a strategy for the leader which maximizes her revenue.[2] Notice that using this more general definition, the original StackMST game can be rephrased as a Budget-StackMST game in which T is any red spanning tree of G, the budget Δ is equal to 0, and the activation cost for an edge not in $E(T)$ is equal to 0 if it belongs to B, otherwise it is equal to any positive value. In this paper, we show that Budget-StackMST admits a $\min\{k, 1 + \ln \beta, 1 + \ln \rho, 2h + \epsilon\}$-approximation algorithm, for any $\epsilon > 0$, where k, β and ρ are as previously defined for StackMST, and h denotes the *radius* of T w.r.t. the number of edges, once T is rooted at its center. Thus, whenever the radius of the red tree is bounded, which might well happen in practice, our result also extends and complements the approximation ratio given in [7] for the classic StackMST.

On the other hand, notice also that a Budget-StackMST game in which the budget Δ is equal to 0 and the activation cost function γ is identically equal to 0, produces an interesting specialization of the original StackMST game, namely that in which the input graph is *complete* (i.e., the set of blue edges is exactly the complement of the set of red edges given in input). In this paper, for this specialized version of the game, that we call Complete-StackMST, we prove the following results:

1. when the red edge have only two distinct costs, the problem can be solved optimally, in contrast with the APX-hardness of the corresponding version of StackMST;
2. when the input red tree is actually a path, the problem can be approximated within $3/2 + \epsilon$, for any $\epsilon > 0$;
3. finally, the problem can be approximated within $7/4 + \epsilon$, for any $\epsilon > 0$, in general.

We point out that all the above problems have an application counterpart. Indeed, the Complete-StackMST problem models the case in which the leader is allowed to activate/price any non-red connection in the network, like one might expect to encounter in the planning phase of the network infrastructure. Notice also that although Complete-StackMST is quite a special case of StackMST, we were actually not able to prove whether the problem is in P or not (for more than 2 red edge costs). Actually, we guess that for an arbitrary number of red edge costs the problem is NP-hard, since the hardness of the classic StackMST resides solely in the selection of the subset of blue edges that will be part of the final solution (and not in their optimal pricing, that as we said before, can be found in polynomial time). This difficulty seems not to be mitigated, in general, when the set of blue edges complements the red tree. Nevertheless, this argument does not carry over into the construction of the hardness proof provided in [7], where the set of input blue edges is carefully picked on purpose (and even more puzzling, since that construction makes use of only 2 red edge costs, then if we enrich the set of blue edges so as to complement the red path, as we said we can prove the problem becomes polynomial!). Thus, a seemingly very different approach is needed, as one cannot leverage on the topology of the set of blue edges. Therefore, this remains a challenging open problem.

The rest of the paper is organized by providing each of the above results in a corresponding section. For the sake of exposition, we first present the results concerning Complete-StackMST, and then we analyze Budget-StackMST. Finally, we provide a concluding section listing some interesting problems left open.

2. An exact algorithm for Complete-StackMST with two red edge costs only

In this section we present an exact polynomial-time algorithm for Complete-StackMST when the cost of any red edge belongs to the set $\{a, b\}$, with $0 \le a < b$. Notice that this case is already APX-hard for StackMST [7]. For the sake of clarity, we will first present the algorithm and the analysis when the red tree is actually a path. The extension to the general case will be derived in the subsequent subsection.

2.1. Solving Complete-StackMST with two red edge costs when T is a path

Now, we present an exact algorithm for Complete-StackMST on a red path P with edge costs in $\{a, b\}$, with $0 \le a < b$. We call a subpath P' of P an *a-block* if P' has all edges of cost a, and P' is maximal (w.r.t. inclusion). We say that an a-block is *good* if its length is greater than or equal to 3, *bad* otherwise. The algorithm exploits the fact that the blue edges within each good block can be priced to obtain the same revenue as the total cost of the block itself. This is because a good block is sufficiently long, hence its (blue) complement is connected. In contrast, this is no longer true for bad blocks for which it is needed to use some blue edge exiting from the block. Intuitively, in order to maximize the overall revenue, the algorithm

[2] Throughout the paper, as usual we assume that when multiple optimal solutions are available for the follower, then he selects an optimal solution maximizing the leader's revenue.

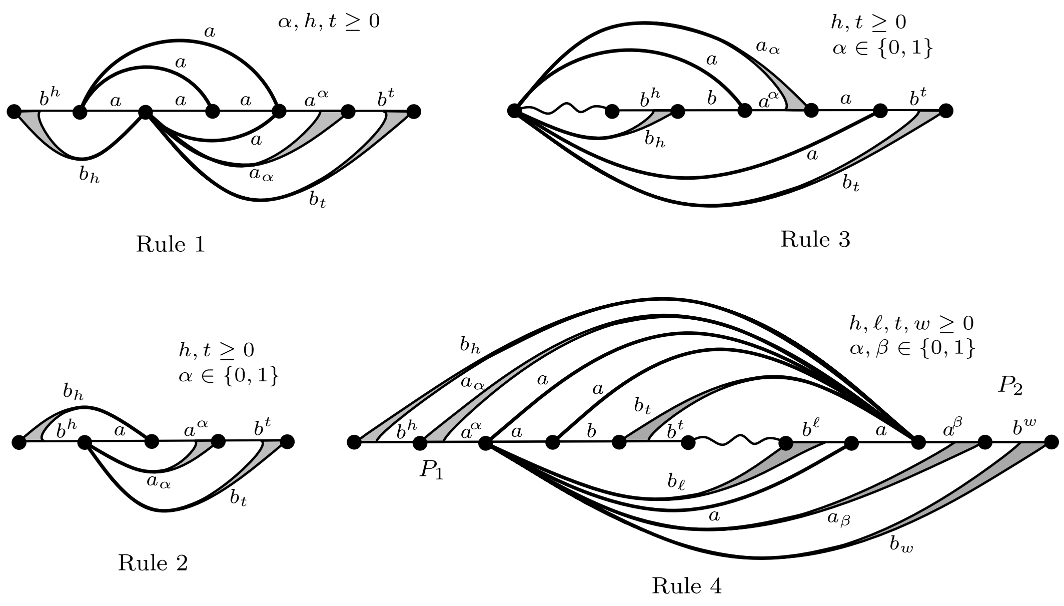


Fig. 1. Rules used by the algorithm to solve subpaths. For the sake of figure clarity, subpaths and set of edges are represented in the following way: with a η^δ along a (red) path edge, we denote a (sub)path of δ red edges each having a cost of η, while with a η_δ along a shadowed region, we denote a star of δ blue edges each of cost η leading to the set of vertices on the red path spanned by the region itself, i.e., the path vertex on its boundary plus $\delta - 1$ unrepresented vertices. Observe that, except for Rule 2, all red (path) edges with cost a will be discarded by the follower. Concerning Rule 2, the follower will select only a single red edge of cost a (of the shown subpath). Concerning Rule 3, notice that the picture illustrates the case in which the red edge of cost b precedes the last bad a-block on P (the other case mentioned in the rule is specular).

can either (i) lose some fraction of bad blocks to gain all the edges of cost b or (ii) pair the bad blocks together in order to collect all their revenue, losing some of the edges of cost b.

Let σ be the number of bad blocks of P. We first present an algorithm achieving a revenue of $c(P) - \min\{\sigma a, \lfloor \frac{\sigma}{2} \rfloor (b - a) + (\sigma \bmod 2) \min\{a, b-a\}\}$, where $c(P)$ denotes the sum of costs of edges of P, and then we show that such a revenue is actually an upper bound to the optimal revenue.

For technical convenience, we only consider instances where P has at least 2 a-blocks as the exact solutions for the remaining instances can be easily computed. The algorithm uses the following four rules. Each rule considers a subpath of P and specifies a feasible solution for the subpath, i.e., a set of blue edges incident to the vertices of the subpath with a corresponding pricing. The solutions corresponding to the rules are shown in Fig. 1.

Rule 1: Let P' be a subpath of P containing only one a-block, and this a-block is good. We can obtain revenue $c(P')$ from P' by adding blue edges only within P'.

Rule 2: Let P' be a subpath of P containing only one a-block and this a-block is bad. We can compute a solution with revenue $c(P') - a$ from P'.

Rule 3: Let P' be a subpath of P containing only one a-block, and this a-block is the last bad block of P (assume that P is traversed from one of its endvertices to the other one). Moreover P' has at least one more edge of cost b that either precedes or follows the a-block. We can obtain a revenue of $c(P') - (b - a)$ from P' by using a star of blue edges centered at the left or right endvertex of P, depending on the position of the edge of cost b. Notice that the endvertices of P' might be followed by other good blocks.

Rule 4: Let P_1, P_2 be two edge-disjoint subpaths of P each containing only one a-block. Assume that both a-blocks are bad and P_1 contains an edge of cost b whose removal separates the two a-blocks. We can obtain a revenue of $c(P_1) + c(P_2) - (b - a)$ from P_1 and P_2. Notice that P_1 and P_2 do not need to be adjacent.

Our algorithm decomposes the red path P into subpaths. The structure of each of these subpaths is the following. Each subpath starts with a (potentially empty) sequence of edges of cost b, which is followed by one or more edges of cost a, and ends with zero or more edges of cost b. Notice that for each of those subpaths we could use either Rule 1 or Rule 2 but this, in general, would lead to a suboptimal solution. For this reason, our algorithm also uses Rule 3 and Rule 4 which allows to collect more revenue. In order to apply such rules, however, we need a suitable decomposition, as described in the following.

If $b \geq 3a$ then we split P into subpaths each of which contains exactly one a-block. Then we apply Rule 1 or Rule 2 to each subpath, depending on whether the a-block in the subpath is good or bad. Hence, this solution yields a revenue of $c(P) - \sigma a$.

Now, consider the case $b < 3a$. Let $B_1, \dots, B_\sigma$ be the bad a-blocks contained in P from left to right w.r.t one of the end-vertices. We first consider the case where $\sigma \geq 2$, i.e., there are at least two bad blocks. The algorithm splits P into subpaths such that (i) each subpath contains exactly one a-block (either good or bad), (ii) for every $i = 0, \dots, \lfloor \sigma/2 \rfloor - 1$, subpath containing B_{2i+1} has an edge of cost b incident to its right endvertex, and (iii) if σ is odd, the subpath containing B_σ has an edge of cost b incident to its left endvertex.

Observe that such a decomposition always exists. Condition (ii) requires a bad block B_i of odd index $i \neq \sigma$ to be followed by an edge of cost b, and can always be satisfied since there must be at least one such edge between any two consecutive bad blocks. Condition (iii) can always be satisfied since σ is an odd number greater than 2 and there must be at least one edge of cost b between the last bad block of even index, and B_σ (having odd index).

Let P_i be the subpath containing B_i. The algorithm uses Rule 1 for every subpath containing a good a-block, and Rule 4 for every pair of subpaths P_{2i+1}, P_{2i+2}, $i = 0, \dots, \lfloor \sigma/2 \rfloor - 1$. Notice that we can apply Rule 4 since, by property (ii), P_{2i+1} contains a red edge of cost b on the immediate right of the corresponding bad block B_{2i+1}. Finally, if σ is odd, we apply Rule 3 for P_σ when $b \leq 2a$ (we can apply Rule 3 since property (iii) above holds), while we use Rule 2 when $b > 2a$. It is easy to see that the revenue of this solution coincides with $c(P) - \min\{\sigma a, \lfloor \frac{\sigma}{2} \rfloor (b-a) + (\sigma \bmod 2) \min\{a, b-a\}\}$. Indeed, as the set of blue edges is acyclic, every (blue) edge priced to a is selected by the follower. It only remains to show that (blue) edges priced to b are selected as well. This is true since every cycle containing any such blue edge must also contain a red edge of cost b (see Fig. 1).

Concerning the case $\sigma \leq 1$, then either P has no bad blocks, and then a revenue of $c(P)$ can be obtained, or there exists only one bad block B_1. In this latter case:

- if $b \geq 2a$, let P' be any subpath containing B_1 and no other a-block; then, solve P' using Rule 2.
- Otherwise, if $b < 2a$, let P' be a subpath containing B_1 and no other a-block plus a suitable additional edge of cost b; then, use Rule 3 on P'. Observe that Rule 3 can always be used since there exists at least one good block.

Finally, split $P \setminus P'$ into subpaths, each containing one good a-block, and solve them using Rule 1. By doing so we obtain a revenue of $c(P) - \min\{a, b-a\}$.

Now, we show that the revenue computed by the above algorithm is the optimal revenue r^*:

Lemma 1. $r^* \leq c(P) - \min\{\sigma a, \lfloor \frac{\sigma}{2} \rfloor (b-a) + (\sigma \bmod 2) \min\{a, b-a\}\}$.

Proof. Let n_a be the number of red edges of cost a. Let T^* be the tree computed by the follower w.r.t. an optimal solution. Moreover, let $B_1, \dots, B_\sigma$ and $\hat{B}_1, \dots, \hat{B}_{\sigma'}$ be the bad and the good blocks of P, respectively. We denote by m_i and $\hat{m}_j$ the number of edges of B_i and $\hat{B}_j$, respectively. Moreover, for an edge $e = (x, y)$, $T^*(e)$ will denote the unique path in T^* between x and y (observe that $T^*(e)$ may be the path containing only edge e). For each $i = 1, \dots, \sigma$ and $j = 1, \dots, \sigma'$, consider the *bad tree* $T_i = \bigcup_{e \in E(B_i)} T^*(e)$,[3] and the *good tree* $\hat{T}_j = \bigcup_{e \in E(\hat{B}_j)} T^*(e)$, respectively. Let $\mathcal{T} = \{T_1, \dots, T_\sigma\} \cup \{\hat{T}_1, \dots, \hat{T}_{\sigma'}\}$. Observe that for each i, j, we have: (i) T_i and $\hat{T}_j$ are both trees containing only edges of cost a, (ii) $V(B_i) \subseteq V(T_i)$ and $V(\hat{B}_j) \subseteq V(\hat{T}_j)$, and, by (i) and (ii), (iii) $E(T^*) \cap E(B_i) \neq \emptyset$, or T_i contains at least $m_i + 1$ edges.

Let us consider the following graph $H = (\bigcup_i V(T_i) \cup \bigcup_j V(\hat{T}_j), \bigcup_i E(T_i) \cup \bigcup_j E(\hat{T}_j))$, and let N be the number of nodes of H. Clearly, H is a forest. Moreover, each connected component of H is either a single tree of $\mathcal{T}$ or it consists of the union of at least two trees in $\mathcal{T}$. Let us consider the set X of "unmerged" bad trees, i.e., $X = \{T_i \mid i = 1, \dots, \sigma, V(T_i) \cap V(T) = \emptyset, \forall T \in \mathcal{T} \setminus \{T_i\}\}$. We define $\ell = |X|$. Observe that each tree in X is in the set $\mathcal{C}$ of connected components of H. Let t be the number of the remaining connected components of H, i.e., $|\mathcal{C}| = t + \ell$. As each bad tree not in X has been merged with some other tree, we have $t \leq \sigma' + \lfloor \frac{\sigma - \ell}{2} \rfloor$.

In order to relate t to the number N of nodes of H, we define $\ell_1 = |\{T_i \mid T_i \in X, E(T^*) \cap E(B_i) \neq \emptyset\}|$. Notice that ℓ_1 is a lower bound to the number of red edges in H. We now give a lower bound to N. Since H spans the vertices of all a-blocks (which are pairwise vertex disjoint), and since property (iii) holds, we have that $N \geq n_a + \sigma + \sigma' + \ell - \ell_1$. Therefore, as H has $N - \ell - t$ edges of cost a, and using $c(P) = n_a(a-b) + (n-1)b$, we have:

$$\begin{aligned}
r^* &\leq (N - \ell - t)a - \ell_1 a + \big(n - 1 - (N - \ell - t)\big)b \\
&= (N - \ell - t)(a - b) - \ell_1 a + (n-1)b \\
&\leq \big(n_a + \sigma + \sigma' - \ell_1 - t\big)(a-b) + (n-1)b - \ell_1 a \\
&= c(P) - \big(\sigma + \sigma' - \ell_1 - t\big)(b-a) - \ell_1 a \\
&\leq c(P) - \left((\sigma - \ell_1) - \left\lfloor \frac{\sigma - \ell}{2} \right\rfloor\right)(b-a) - \ell_1 a
\end{aligned}$$

[3] Here the union symbol denotes the union of graphs.

$$\leq c(P) - \min\left\{\sigma a, \left\lfloor \frac{\sigma}{2} \right\rfloor (b-a) + (\sigma \bmod 2)\min\{a, b-a\}\right\}.$$

To see why the latter inequality holds, one can consider the different parity of σ and ℓ for each of the following three cases: $b \geq 3a$, $2a \leq b < 3a$, and $b < 2a$. □

Hence, from the above lemma, we have:

Theorem 1. *Complete-StackMST can be solved in polynomial time when the red edges form a path and their costs are in $\{a, b\}$.*

2.2. *Solving Complete-StackMST with two red edge costs: the general case*

We now extend the previous result by providing an exact polynomial-time algorithm for Complete-StackMST when the red edge costs belong to the set $\{a, b\}$, and T is a tree.

Let $0 \leq a < b$, and let T be a red tree with costs in $\{a, b\}$. In a similar fashion as before, we call a subtree T' of T an *a-block* if T' has all edges of cost a, and T' is maximal (w.r.t. inclusion). We say that an a-block is *bad* if it is a star, *good* otherwise. Once again we only consider instances that have at least 2 a-blocks for technical convenience.

The intuition behind the algorithm is the same we discussed in the previous section for the case of a red path: for good blocks we are able to obtain a revenue equal to the cost of its red edges, while bad blocks need to be suitably paired.

Let σ be the number of bad blocks of T. As the upper bound to the maximum revenue r^* shown in Lemma 1 still holds,[4] we now present a general algorithm achieving a revenue equal to the given upper bound.

The four rules used by the algorithm are similar to the ones used in the algorithm for the path and they are shown in Fig. 2 and 3, along with the corresponding revenues.

Our algorithm is as follows. If $b \geq 3a$, then we split T into subtrees, each of them containing exactly one a-block. Then we apply Rule 1 or Rule 2 to each subtree, depending on whether the a-block in the subtree is good or bad. Clearly, this solution yields a revenue of $c(T) - \sigma a$.

Now, consider the case $b < 3a$. Let $B_1, \dots, B_\sigma$ be the bad a-blocks contained in T. The algorithm splits T into subtrees such that (i) each subtree contains exactly one a-block (either good or bad), (ii) there exists a permutation $B'_1, \dots, B'_\sigma$ of the bad a-blocks such that for every $i = 0, \dots, \lfloor \sigma/2 \rfloor - 1$, subtree containing B'_{2i+1} has an edge of cost b along the (unique) path joining B'_{2i+1} with B'_{2i+2}, and (iii) if σ is odd, the subtree containing B'_σ has an edge of cost b.

Let T_i be the subtree containing B'_i. The algorithm uses Rule 4 for every pair of subtrees T_{2i+1}, T_{2i+2}, $i = 0, \dots, \lfloor \sigma/2 \rfloor$, Rule 1 for every subtree containing a good a-block. Finally, if σ is odd, we apply Rule 3 for T_σ when $b \leq 2a$, while we use Rule 2 when $b > 2a$. From this, we have:

Theorem 2. *Complete-StackMST can be solved in polynomial time when red edge costs are in $\{a, b\}$.*

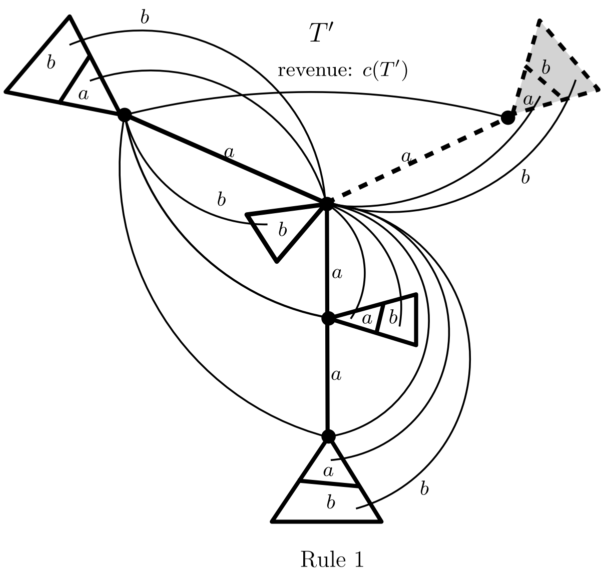
3. Complete-StackMST can be approximated within $3/2 + \epsilon$ when the red edges form a path

In this section we design a $(\frac{3}{2} + \epsilon)$-approximation algorithm for Complete-StackMST when the tree T is actually a path, say P.

Let then P be the path of red edges. The idea of the algorithm is to consider three possible solutions and to pick the best one. The first two solutions are similar: each one of them is made up by covering pairs of two consecutive red edges with a blue edge, and they are essentially obtained one from the other by an edge shift on the red path. Roughly speaking, these two solutions will guarantee the sought approximation factor when the costs of adjacent red edges in the path are apart each other, since in this way one of the two solutions will gain the costlier red edge. On the other hand, when the costs of all the red edges are comparable, these two solutions can only obtain about a half of the optimal revenue, hence we also consider a third solution consisting of a blue star which, in this case, collects an almost optimal revenue. In general, we will argue that the revenue of the best solution is at least a fraction $\frac{2}{3}$ of the cost of almost the entire path. More precisely, we select a cheap subpath $\bar{P}$ of P with 2 or 3 edges, and we then compute a solution achieving a revenue of at least $\frac{2}{3}(c(P) - c(\bar{P}))$.

Let $m = n - 1$ be the length of P, and let $e_1, \dots, e_m$ be the red edges of P in the order of traversing P from an endpoint to the other one. Moreover, let us set ℓ to 2, if m is even, and to 3 otherwise. Let P_i be the subpath of P of length ℓ starting from e_i, i.e., P_i consists of the edges $e_i, \dots, e_{i+\ell-1}$. Let $\bar{P}$ be the subpath with minimum cost among P_{2j-1}, $j = 1, \dots, \lfloor m/2 \rfloor$. If we remove $\bar{P}$ from P, we obtain two paths of even length, say Q_1 and Q_2. At most one of Q_1 and Q_2 may be empty. Let us assume for the ease of presentation that both paths are non-empty (similar arguments hold when this does not happen), and let $2h$ and $2k$ be the length of Q_1 and Q_2, respectively. Moreover, let $u_0, u_1, \dots, u_{2h}$ and $v_0, v_1, \dots, v_{2k}$ be the nodes of Q_1 and Q_2, respectively. The two endvertices of $\bar{P}$ are u_{2h} and v_0. Let $x_i = c(u_{i-1}, u_i)$ and $y_j = c(v_{j-1}, v_j)$. Finally, let z be an internal node of $\bar{P}$ (see Fig. 4).

[4] The proof is identical to the one previously shown.



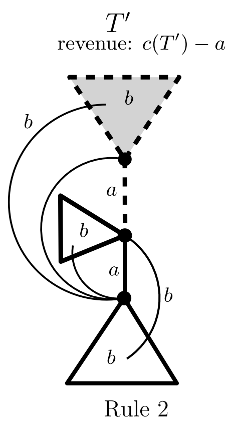


Fig. 2. Rules 1 and 2 for the general case. Edges without label are priced to a. A single edge of price η leading to a subtree (depicted as a triangle) represents a star whose edges span all the vertices of the subtree and have price η. Each dashed subtree can appear 0 or more times.

Let us define the two quantities $A = \sum_{i=1}^{h} \max\{x_{2i-1}, x_{2i}\} + \sum_{i=1}^{k} \max\{y_{2i-1}, y_{2i}\}$, and $B = \sum_{i=1}^{h} \min\{x_{2i-1}, x_{2i}\} + \sum_{i=1}^{k} \min\{y_{2i-1}, y_{2i}\}$. Notice that $c(Q_1) + c(Q_2) = A + B$. The first solution we consider is

$$F_1 = \{(u_{2i-2}, u_{2i}) \mid i = 1, \dots, h\} \cup \{(v_{2i-2}, v_{2i}) \mid i = 1, \dots, k\},$$

and the price function is defined as $p(u_{2i-2}, u_{2i}) = \max\{x_{2i-1}, x_{2i}\}$, and $p(v_{2i-2}, v_{2i}) = \max\{y_{2i-1}, y_{2i}\}$. Notice that this solution obtains a revenue $r_1 = A$.

The second solution is a star centered in the node z; more precisely:

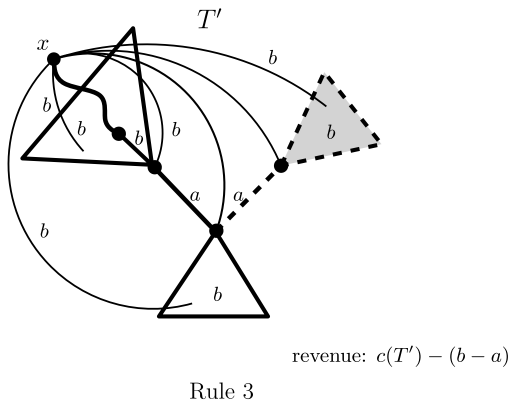
$$F_2 = \{(z, u_i) \mid i = 0, \dots, 2h-1\} \cup \{(z, v_i) \mid i = 1, \dots, 2k\},$$

and the prices are defined as $p(z, u_0) = x_1$, $p(z, v_{2k}) = y_{2k}$, $p(z, u_i) = \min\{x_i, x_{i+1}\}$, and $p(z, v_i) = \min\{y_i, y_{i+1}\}$. Notice that this solution obtains a revenue of

$$r_2 = B + x_1 + y_{2k} + \sum_{i=0}^{h-2} \min\{x_{2i+2}, x_{2i+3}\} + \sum_{i=0}^{k-2} \min\{y_{2i+2}, y_{2i+3}\}.$$

Finally, the third solution is the following:

$$F_3 = \{(u_{2i+1}, u_{2i+3}) \mid i = 0, \dots, h-2\} \cup \{(v_{2i+1}, v_{2i+3}) \mid i = 0, \dots, k-2\} \cup \{(u_{2h-1}, z), (z, v_1)\},$$



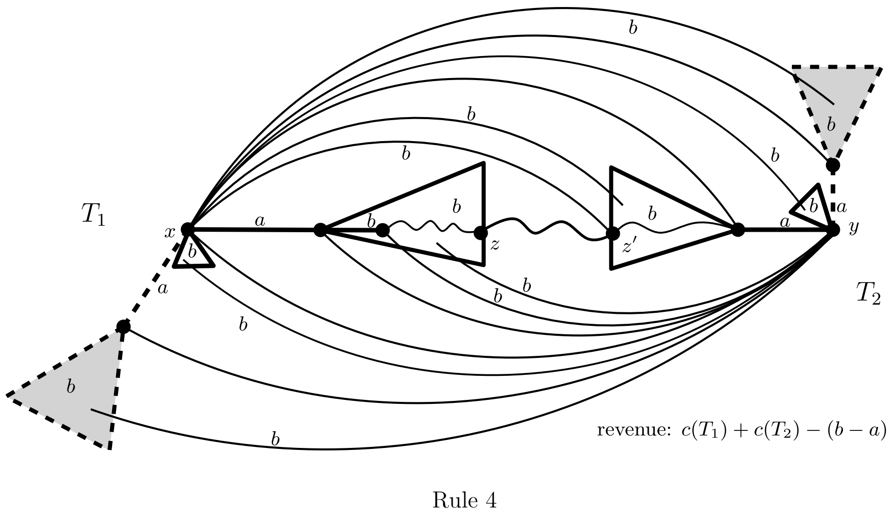


Fig. 3. Rules 3 and 4 used by the algorithm to solve subtrees. A single edge of price η leading to a subtree (depicted as a triangle) represents a star whose edges span all the vertices of the subtree and have price η. Each dashed subtree can appear 0 or more times. Vertex x in Rule 3 is any leaf of T that is not contained in subtree T' (this vertex always exists since there are at least 2 a-blocks). Notice that in Rule 4, while there is a blue edge between x and z', there is no edge between y and z.

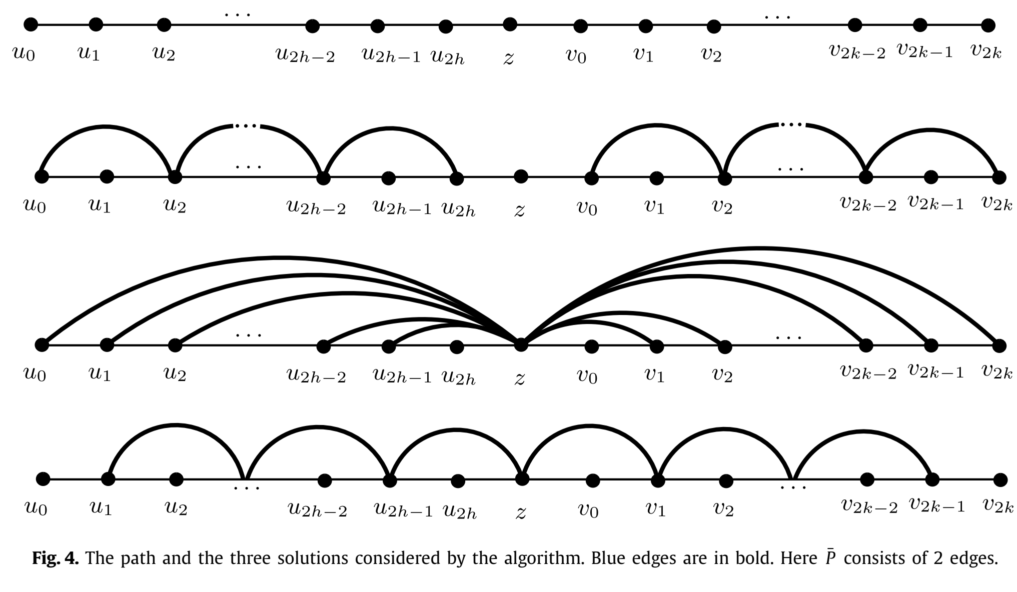


Fig. 4. The path and the three solutions considered by the algorithm. Blue edges are in bold. Here $\bar{P}$ consists of 2 edges.

and the pricing is as follows: $p(u_{2h-1}, z) = x_{2h}$, $p(z, v_1) = y_1$, $p(u_{2i+1}, u_{2i+3}) = \max\{x_{2i+2}, x_{2i+3}\}$, and $p(v_{2i+1}, v_{2i+3}) = \max\{y_{2i+2}, y_{2i+3}\}$. Hence, the corresponding revenue is:

$$r_3 = x_{2h} + y_1 + \sum_{i=0}^{h-2} \max\{x_{2i+2}, x_{2i+3}\} + \sum_{i=0}^{k-2} \max\{y_{2i+2}, y_{2i+3}\}.$$

Hence, we have:

$$\begin{aligned} r_1 + r_2 + r_3 = {} & A + B + x_1 + x_{2h} + y_1 + y_{2k} \\ & + \sum_{i=0}^{h-2} \big(\min\{x_{2i+2}, x_{2i+3}\} + \max\{x_{2i+2}, x_{2i+3}\}\big) \\ & + \sum_{i=0}^{k-2} \big(\min\{y_{2i+2}, y_{2i+3}\} + \max\{y_{2i+2}, y_{2i+3}\}\big) \\ = {} & 2\big(c(Q_1) + c(Q_2)\big), \end{aligned}$$

from which it follows that the revenue $r = \max\{r_1, r_2, r_3\}$ is at least $\frac{2}{3}(c(Q_1) + c(Q_2))$. Now, observe that by construction we have

$$c(\bar{P}) \le \frac{c(P)}{\lfloor \frac{n}{\ell} \rfloor} \le \frac{3}{n-2}\big(c(\bar{P}) + c(Q_1) + c(Q_2)\big),$$

and hence $c(\bar{P}) \le \frac{3}{n-5}(c(Q_1) + c(Q_2))$. Denoting by r^* the optimal revenue, and observing that the cost of the red tree is always an upper bound to r^*, we then have

$$\frac{r^*}{r} \le \frac{c(P)}{r} = \frac{c(Q_1) + c(Q_2)}{r} + \frac{c(\bar{P})}{r} \le \frac{3}{2} + \frac{\frac{3}{n-5}(c(Q_1) + c(Q_2))}{\frac{2}{3}(c(Q_1) + c(Q_2))} = \frac{3}{2} + \frac{9}{2n-10}.$$

Finally, observe that for any constant $\epsilon > 0$, all the small instances of the problem (i.e., those for which $\frac{9}{2n-10} > \epsilon$), can be solved in $O(1)$ time. Indeed, the prices of the blue edges appearing in the MST of any optimal solution are restricted to belong to the set of red edge costs (see Lemma 1 in [7]), and so an optimal pricing can simply be found exhaustively. Therefore, we have proved the following:

Theorem 3. *If the input red tree is a path, then Complete-StackMST can be approximated in polynomial time within a factor of $3/2 + \epsilon$, for any constant $\epsilon > 0$.*

We point out that our algorithm is asymptotically tight with respect to the adopted upper-bound scheme. An example is the path in which $c(e_1) = 1, c(e_2) = 2$, and $c(e_i) = 0$, for every $i > 2$. It is easy to see that for this path the revenue obtained by an optimal solution is 2, while the total cost of the path is 3.

4. Complete-StackMST can be approximated within $7/4 + \epsilon$

In this section we design an algorithm that achieves an approximation ratio of $7/4+\epsilon$ for the general Complete-StackMST game.

The idea of the algorithm is to partition the red tree $T = (V, E(T))$ into suitable edge-disjoint subtrees for which we can guarantee a revenue of at least 4/7 of the cost of each one of them. It is easy to see that we can solve locally a Complete-StackMST game for each red subtree of the partition, and then solve the original problem by joining together all the local solutions (by maintaining the corresponding pricing). Indeed, the union of all the trees associated with the local solutions is clearly a spanning tree of the entire graph. Hence, we can claim the following

Lemma 2. *Let $T_1, \dots, T_\ell$ be a partition of T into ℓ edge-disjoint subtrees. For each i, let r_i be the revenue returned by a local solution of T_i. Then, the revenue which can be obtained for T is at least $\sum_{i=1}^{\ell} r_i$.*

Proof. Let p_i be a pricing yielding revenue r_i and let F_i be the set of the blue edges in the corresponding MST of a local solution for T_i. Let $F = \bigcup_{i=1}^{r} F_i$ and let p be a function that prices each edge e belonging to a set F_i to $p_i(e)$. We will argue that the revenue obtained by activating the edges in F with pricing p will be at least $\sum_{i=1}^{\ell} r_i$ as each edge e must belong to $M(F, p)$.

Suppose, by contradiction, that there exists a blue edge $e \in F_i$ which does not belong to $M(F, p)$. Then, it must exist a simple cycle C in the graph $(V, E(T) \cup F)$ such that e is the heaviest edge in C. Since $E(T)$ is acyclic and since every blue

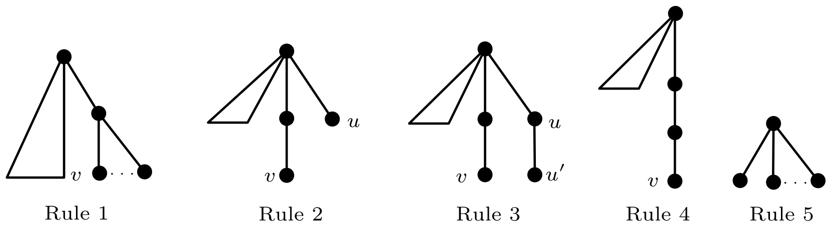


Fig. 5. The five rules of the decomposition algorithm.

edge has its end-points in the same subtree, in order to be simple, C must also be a cycle of the graph $(V_i, E_i \cup F_i)$. Then, e cannot belong to $M(F_i, p_i)$. □

Moreover, we can prove the following:

Lemma 3. *Let T be a tree rooted at a node s. There always exists a partition of T into ℓ edge-disjoint subtrees $T_1, \dots, T_\ell$ such that*

- *T_ℓ has at most 2 edges and at least one of them is incident to s;*
- *for every $1 \le j \le \ell - 1$, T_j is either (i) a path of 3 or 4 edges, or (ii) a star with at least 3 edges.*

Moreover, this partition can be found in polynomial time.

Proof. We provide a polynomial-time algorithm that finds the partition of the lemma. We will use an auxiliary tree T' that is initially equal to T and that will be updated during the execution of the algorithm. In the following, given a vertex $v \in V(T')$ we will denote by $d(v)$ the depth of v in T', i.e., the number of edges of the root-to-v path. Moreover, we will denote by $S(v)$ the set of the children of v in T'. Finally, we will use $\bar{v}$ to denote the parent of v.

The algorithm works in phases. During phase j, we find a subtree T_j of T' according to one of the rules below. We consider these rules in order, i.e., we always use the first of the rules that can be applied to T'. Then, we update T' by removing T_j and we move to the next phase. We stop when no rule can be applied. Let L be the set of leaves of T' with depth equal to the height of T'. The rules are the following (see Fig. 5):

Rule 1: if there exists a node $v \in L$ with $d(v) \ge 2$ and such that v has at least one sibling, then T_j is the star with edge set $\{(\bar{v}, \bar{\bar{v}})\} \cup \{(\bar{v}, u) \mid u \in S(\bar{v})\}$;

Rule 2 if there exists a node $v \in L$ with $d(v) \ge 2$ such that $\bar{v}$ has a sibling u and u is a leaf, then T_j is the path with edge set $\{(v, \bar{v}), (\bar{v}, \bar{\bar{v}}), (\bar{\bar{v}}, u)\}$;

Rule 3: if there exists a node $v \in L$ with $d(v) \ge 2$ such that $\bar{v}$ has a sibling u and u is not a leaf, then let u' be the unique child of u (u' must be unique otherwise Rule 1 would apply). Then, T_j is the path with edge set $\{(v, \bar{v}), (\bar{v}, \bar{\bar{v}}), (\bar{\bar{v}}, u), (u, u')\}$;

Rule 4: if there exists a node $v \in L$ with $d(v) \ge 3$, then T_j is the path with edge set $\{(v, \bar{v}), (\bar{v}, \bar{\bar{v}}), (\bar{\bar{v}}, \bar{\bar{\bar{v}}})\}$;

Rule 5: if T' is a star with at least 3 edges, then $T_j = T'$.

Now, assume that the last phase is phase $\ell - 1$, then we set T_ℓ equal to the remaining tree T'. If there is no edge left, we set T_ℓ equal to the empty subtree. It is easy to see that if T_ℓ is non-empty, it must have at most 2 edges, and one of them must be incident to s. Moreover, since each phase takes polynomial time and each T_j with $j < \ell$ contains at least one edge, $\ell \le n$ and the claim follows. □

The following lemmas allow us to obtain a revenue of at least $\frac{4}{7} c(T_i)$ for each subtree $T_i, i = 1, \dots, \ell - 1$, of the partition.

Lemma 4. *Let S be a star with at least 3 edges, then we can obtain a revenue of at least $\frac{2}{3} c(S)$.*

Proof. Let s be the center of the star, and let $u_1, \dots, u_t$ be the leaves ordered such that $c(s, u_1) \le c(s, u_2) \le \dots \le c(s, u_t)$. The set of blue edges $F = \{(u_1, u_j) \mid j = 2, \dots, t\}$ yields a revenue of $\sum_{j=2}^{t} c(s, u_j) \ge \frac{2}{3} c(S)$, since $t \ge 3$. □

Lemma 5. *Let P be a path of 3 or 4 edges, then we can obtain a revenue of at least $\frac{4}{7} c(P)$.*

Proof. Let us consider the path of 3 edges first. Let $0 \le c_1 \le c_2 \le c_3$ be the edge costs. If the cost of the middle edge is c_1, we can easily obtain a revenue of $c_2 + c_3 \ge \frac{2}{3} c(P)$. Assume that the cost of the middle edge is not c_1. In Fig. 6 three solutions are shown. The corresponding revenues are: $c_3, 2c_2, 2c_1 + c_2$. A trivial calculation shows that the maximum of the three revenues is at least $\frac{4}{7} c(P)$.

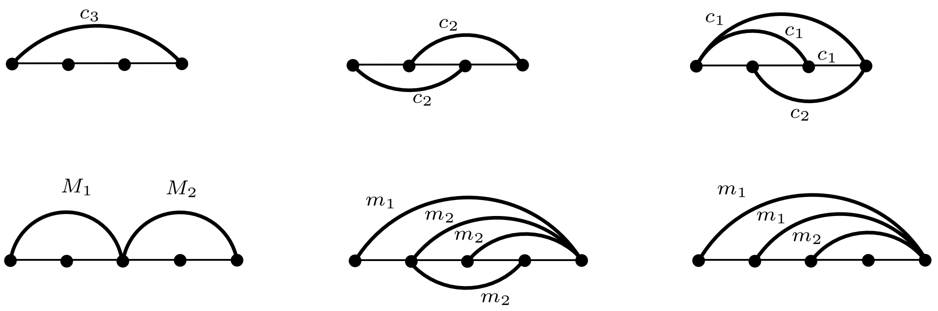


Fig. 6. Three possible solutions for paths of 3 or 4 edges.

Now, we consider a path of 4 edges. Let $x_1, \ldots, x_4$ be the costs of the edges from left to right. We set $M_1 = \max\{x_1, x_2\}, m_1 = \min\{x_1, x_2\}, M_2 = \max\{x_3, x_4\}, m_2 = \min\{x_3, x_4\}$. Assume w.l.o.g. that $m_1 \geq m_2$. Three solutions are shown in Fig. 6. The corresponding revenues are: $M_1 + M_2, m_1 + 3m_2, 2m_1 + m_2$, the maximum of which is easy to see to be at least $\frac{4}{7}c(P)$. $\square$

We are now ready to prove the following:

Theorem 4. *Complete-StackMST can be approximated in polynomial time within a factor of* $7/4 + \epsilon$, *for any constant* $\epsilon > 0$.

Proof. W.l.o.g., we can restrict ourselves to the case $n \geq \frac{7}{2\epsilon} + 1$, since otherwise we can find an optimal solution by an exhaustive search, as described in the previous section for the path case. Then, for each v, let $\mu(v) = \max_{u|(u,v)\in E(T)} c(u, v)$. We root T at a node s minimizing μ. Then we decompose T using the algorithm given in Lemma 3, and we solve locally each T_j with $j \leq \ell$. Let r_j be the corresponding obtained revenue, and observe that $r_\ell \geq c(T_\ell) - \min_{e\in E(T_\ell)} c(e) \geq c(T_\ell) - \mu(s)$.

As Lemma 2 together with Lemmas 4 and 5 implies that the total revenue r is at least $\sum_{j=1}^{\ell} r_j \geq \frac{4}{7}(c(T) - \mu(s))$, and since $c(T) \geq \frac{1}{2}\sum_{v\in V} \mu(v) \geq \frac{n}{2}\mu(s)$, we obtain

$$\frac{r^*}{r} \leq \frac{c(T)}{r} \leq 7/4 + \frac{7}{2n-4} \leq 7/4 + \epsilon. \qquad \square$$

5. Approximating Budget-StackMST

In this section, we study the general Budget-StackMST, and we show that this problem (and thus, *a fortiori*, StackMST) can be approximated within a factor of $\min\{k, 1 + \ln\beta, 1 + \ln\rho, 2h + \epsilon\}$, for any $\epsilon > 0$, where k is the number of distinct red edge costs, β is the number of blue edges selected by the follower in an optimal solution, ρ is the maximum ratio among pairs of red edge costs, and finally h is the height of T (measured w.r.t. the number of edges) once rooted at its center. In other words, we add to the argument of the minimum approximation-ratio function provided in [7] a further value depending on the radius of the red tree. Since we can prove at the same time that StackMST (and thus Budget-StackMST) remains APX-hard also when the radius of T is equal to 1 (and the number of red edge costs is at least 2), i.e., T is a star, then we have the relevant consequence that our approximation algorithm is asymptotically tight (for both problems) when $h = O(1)$ (and in particular, red stars admit a $(2 + \epsilon)$-approximation).

Let us start by proving the following hardness result:

Theorem 5. *StackMST is APX-hard even if* T *is a star with only two red edge costs.*

Proof. In [7], the authors proved that StackMST for the case in which T is a path is APX-hard, even if there are only two different red edge costs, say 1 and 2. More precisely, they showed that on a special class of red path instances of StackMST, the existence for any $\epsilon > 0$ of a $(1 - \epsilon)$-approximation algorithm,[5] would imply the existence of $(1 + 8\epsilon)$-approximation algorithm for VERTEX COVER on graphs of maximum degree at most 3, which is well-known to be APX-hard [1]. Each of these hard instances of StackMST consists of a red path $T = \langle x_1, \ldots, x_p, y_1, \ldots, y_q\rangle$, where all the red edges up to x_p cost 1, while the remaining edges cost 2, and finally a set of blue edges B each going from an x-vertex to a y-vertex (see Fig. 7(a)).

Now, we show an approximation-preserving reduction from this class of instances of StackMST to the class of instances of StackMST in which the input red tree is a star with only two costs. The reduction works as follows: Given any of the

[5] Notice that in this case we are adopting the strictly formal definition of the *approximation* guarantee for a maximization problem, as StackMST actually is, while throughout the paper we have used the *performance* guarantee notation, according to the previous literature.

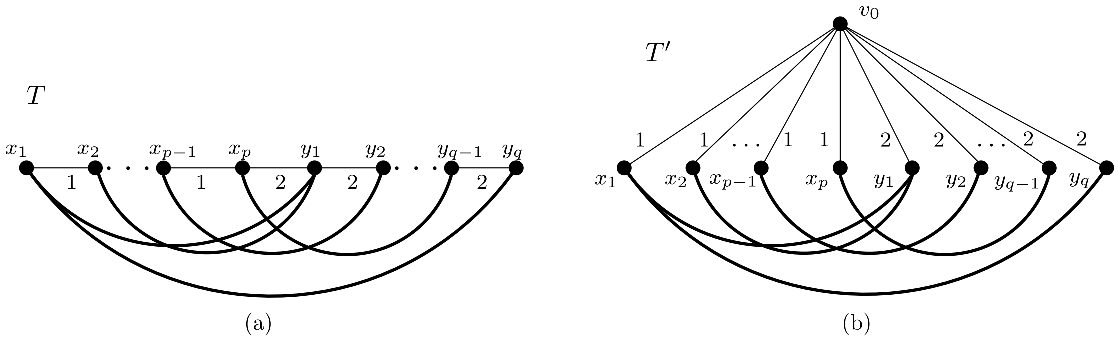


Fig. 7. The APX-reduction: in (a), the hard path instance (blue edges are bold), while in (b), the corresponding star. Notice that, by construction, each pair of blue edges incident to a same y-vertex forms a cycle with red edges all of cost 1.

aforementioned hard instances for StackMST on red paths, we add to it a vertex v_0, and then we replace the red path by a star of red edges centered at v_0. Let T' denote the obtained star of red edges. The cost of a red edge e of T' is set to 1 if e is incident to a x-vertex, and to 2 otherwise (see Fig. 7(b)). Then, it is easy to see that for any subset $F \subseteq B$ of blue edges selected in a final MST, the maximum yieldable revenue in both instances of StackMST is the same, as in both instances, a blue edge (x_i, y_j) can be priced to 2 iff it is the only blue edge of F incident to y_j (indeed, two or more blue edges of F incident to a same y-vertex always forms a cycle with red edges all of cost 1, and moreover each blue edge in B always forms a cycle w.r.t. the red path/star containing at least one red edge of cost 2). □

We now study the approximability of Budget-StackMST. First, we will argue that for this problem the very same approximation ratio as that of StackMST can be achieved, since the *Best-out-of-k algorithm* defined in [7] (recall that k is the number of distinct red edge costs) can be easily adapted to provide an approximation of $\min\{k, 1 + \ln\beta, 1 + \ln\rho\}$ for Budget-StackMST as well. Indeed, let $c_1 < c_2 < \cdots < c_k$ denote the red edge costs. To extend the algorithm, we proceed as follows. We consider the complete graph consisting of the union of the red tree and all the potential blue edges. For each $j = 1, \ldots, k$, we set the price of every potential blue edge to c_j, and we compute a spanning tree by a slight modification of Kruskal's algorithm as follows: when (red and blue) edges of cost c_j are considered, we first check blue edges in non-decreasing order of activation cost, and if the activation cost of the current blue edge does not exceed the remaining budget, and no cycle is created, we add it to the solution. The solution for the j-th iteration will be then the set of all the added blue edges, each with price c_j. Then we select the iteration such that the corresponding revenue is maximum, and we return the corresponding solution. In fact, the only difference w.r.t. the algorithm given in [7] is that we discard at each iteration the blue edges that cannot be inserted in the solution just because the budget has been exhausted. It turns out that this variation does not affect the fact that each computed MST, say T_j when cost c_j is considered, will contain at least a number of blue edges equal to the number of blue edges of cost at least c_j in an optimal solution. This is exactly the crucial structural property which is used in [7] to lower bound the obtained revenue. As a result, the analysis provided in [7] for StackMST can be applied to our modified algorithm as well, and so we have the following:

Theorem 6. *The above algorithm achieves an approximation ratio of* $\min\{k, 1 + \ln\beta, 1 + \ln\rho\}$ *for Budget-StackMST.*

In the rest of the section, we will finally show the existence of a $(2h + \epsilon)$-approximation algorithm for Budget-StackMST, where h denotes the radius of T once rooted at its center, say v_0. Before starting, recall that once a set F of activated edges is part of the final MST, then the optimal pricing for each $e \in F$ can be computed in polynomial time, as observed in [7]. More precisely, this can be done by computing efficiently

$$p_F(e) := \min_{H \in \mathtt{cycle}(F,e)} \max_{e' \in E(H) \cap E(T)} c(e') \tag{1}$$

where $\mathtt{cycle}(F, e)$ is the set of (simple) cycles containing edge e in the graph $G' = (V(T), E(T) \cup F)$.

To show that p_F is indeed optimal w.r.t F, observe that when we use the above pricing, there always exists an MST of G' containing all the blue edges in F since F is acyclic and no blue edge can be the heaviest edge of any cycle. On the other hand, in order for an MST of G' to contain all the blue edges in F, no edge $e \in F$ can have a price strictly greater than $p_F(e)$, since otherwise there would be a cycle in G' containing a blue edge as its heaviest edge.

Notice that the above pricing can also be computed efficiently (in almost linear time) as follows: set the prices of all the blue edges in F to 0 and compute an MST T' of the resulting graph. Then, solve the *sensitivity analysis* problem [13] on T' which, for every edge $e \in F$ of the MST, returns the maximum weight $p'_F(e)$ that can be assigned to e without affecting the optimality of T'. It is well-known that $p'_F(e)$ coincides with the weight of a lightest (red) edge $f \in E \setminus E(T')$ cycling with e in $T' \cup \{f\}$. As a consequence, T' is still an MST w.r.t. the pricing p'_F, hence all the blue edges in F are selected by

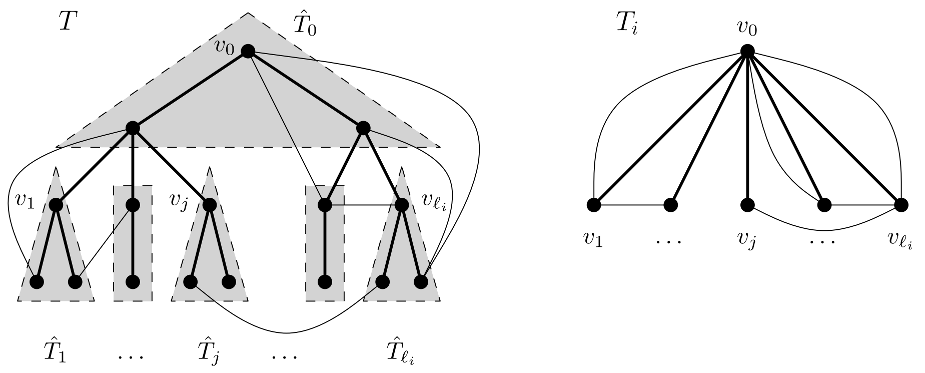


Fig. 8. Auxiliary instance for Budget-StackMST corresponding to level i. Subtrees $\hat{T}_j$, $j = 1, \dots, q$ are contracted to their roots v_j, $j = 1, \dots, q$. When two or more edges between the same two trees are present, only the one with smallest activation cost is preserved.

the follower. To prove that p'_F is indeed optimal it suffices to show that $p'_F(e) \geq p_F(e)$. This is clearly true since f is the heaviest edge of the unique cycle in $T' \cup \{f\}$ which belongs to $\mathtt{cycle}(F, e)$.

With a little abuse of notation, in the following we will denote by $r(F)$ the revenue yielded by the above optimal pricing p_F. The main idea of the algorithm is to reduce the problem instance to h instances in which the red trees are stars. For technical convenience, in each of the h auxiliary instances, we sometimes allow the leader to activate edges which are parallel to red edges. We denote by $V_i = \{v_1, \dots, v_{\ell_i}\}$ the set of vertices at level i in T, and by E_i the set of edges in T going from vertices in V_i to their parents. Let T_i be a red star obtained by contracting all red edges in T but those in E_i. When edge (u, v) is contracted, and w.l.o.g. u is the parent of v in T, we assume that the corresponding vertex is labeled with u. Thus, according to this assumption, we have that T_i is a star centered at v_0 with $v_1, \dots, v_{\ell_i}$ as leaves. The cost of a red edge (v_0, v) in T_i is $c_i(v_0, v) = c(u, v)$, where u is the parent of v in T, and let us set $c_i(v_0, v_0) = 0$. Let T_i be fixed. Let $\hat{T}_0, \hat{T}_1, \dots, \hat{T}_{\ell_i}$ be the connected components in $T - E_i$. W.l.o.g., assume $v_i \in V(\hat{T}_i)$. Let $e_{j,q}$ be a blue edge connecting $\hat{T}_j$ and $\hat{T}_q$ with cheapest activation cost. Let $\mathtt{blue}_i := \{e_{j,q} \mid j, q = 0, \dots, \ell_i, j \neq q\}$. Notice that this set contains edges that the leader can activate in the original instance of the problem. We now map them to their counterpart in T_i, namely let $B_i := \{\bar{e}_{j,q} := (v_j, v_q) \mid e_{j,q} \in \mathtt{blue}_i\}$ be the set of blue edges the leader is allowed to activate in T_i. The activation cost of an edge $\bar{e}_{j,q} \in B_i$ is $\gamma_i(\bar{e}_{j,q}) := \gamma(e_{j,q})$. The auxiliary instance corresponding to level i is shown in Fig. 8.

Let F^* be an optimal solution for the leader on input instance T and let $F_i^* := \{(v_j, v_q) \in B_i \mid (u, v) \in F^*, u \in V(\hat{T}_j), v \in V(\hat{T}_q), j \neq q\}$ be the corresponding edges in T_i. Let $G_i^* := (\{v_0, \dots, v_{\ell_i}\}, F_i^*)$, and denote by $\mathtt{comp}(G_i^*)$ the set of connected components of G_i^*. We start by proving an upper bound on the revenue yielded by F^*. Then, we have:

Lemma 6. $r(F^*) \leq c(T) - \sum_{i=1}^{h} \sum_{H \in \mathtt{comp}(G_i^*)} \min_{v \in V(H)} c_i(v_0, v)$.

Proof. Observe that for every $H \in \mathtt{comp}(G_i^*)$ not containing vertex v_0, at least one red edge (v_0, v), for some $v \in V(H)$, has to be contained in any MST of $(V(T_i), E(T_i) \cup F_i^*)$. Thus, for some $v \in V(H)$, at least one edge (u, v) where u is the parent of v in T has to be contained in any MST of the graph $(V(T), E(T) \cup F^*)$. As $c_i(v_0, v) = c(u, v)$, the claim follows by summing over all components $H \in \mathtt{comp}(G_i^*)$ for all i's. □

The key idea of our algorithm is to find a set F of blue edges whose overall activation cost does not exceed the budget, and such that (V, F) is a forest of stars. More precisely, for every $i = 1, \dots, h$, the algorithm first finds a set $\hat{F}_i \subseteq B_i$ such that $\sum_{e \in \hat{F}_i} \gamma_i(e) \leq \Delta$ and $\hat{G}_i := (V(T_i), \hat{F}_i)$ is a forest of stars; then, it considers the set $F_i := \{e_{j,q} \mid \bar{e}_{j,q} \in \hat{F}_i\}$ of the corresponding blue edges for the original instance. Observe that (i) $G_i := (V(T), F_i)$ is still a forest of stars and (ii) the overall activation cost of the edges in F_i equals that of the edges in $\hat{F}_i$. Furthermore, using Eq. (1), we can derive the following lemma, which claims that when we map $\hat{F}_i$ back to F_i the obtained revenue cannot decrease:

Lemma 7. $r(F_i) \geq r(\hat{F}_i)$.

We now give a lower bound of the revenue that can be obtained from $\hat{F}_i$ by considering the leaves of the stars in $\hat{G}_i$.[6] The bound trivially follows from (1):

[6] If a star contains only one edge, then let any of its vertices to be a leaf.

Algorithm 1

```
1: for $i = 1$ to $h$ do
2:     compute a $(1+\epsilon/(2h))$-approximate solution $S_i$ for the knapsack instance $K_i$
3:     $B' = \{e \in B_i \mid o^i_j \in S_i, e$ is associated with $o^i_j\}$
4:     compute $F^1$ and $F^2$ w.r.t. $B'$ as explained in Lemma 9
5:     if $r(F^1) \geq r(F^2)$ then $\hat{F}_i := F^1$ else $\hat{F}_i := F^2$ end if
6:     $F_i := \{e_{j,q} \mid \bar{e}_{j,q} \in \hat{F}_i\}$
7: end for
8: return the best of the $F_i$'s
```

Lemma 8. *Let $L_i := \{v \mid v \in V(T_i), v$ is a leaf of some star in $\hat{G}_i\}$. Then, $r(\hat{F}_i) \geq \sum_{v \in L_i} c_i(v_0, v)$.*

Next lemma essentially shows that there exists a solution for T_i which is a forest of stars yielding a revenue of at lest a half of the optimal revenue for T_i.

Lemma 9. *Let $B' \subseteq B_i$ and let $U = \{v \mid v$ is an endvertex of some edge in $B'\}$. There exists a polynomial time algorithm that finds two sets F^1 and F^2 such that (i) $F^1, F^2 \subseteq B'$, (ii) both $(V(T_i), F^1)$ and $(V(T_i), F^2)$ are forests of stars, and (iii) $r(F^1) + r(F^2) \geq \sum_{v \in U} c_i(v_0, v)$.*

Proof. Let D be the graph induced by edge set B'. Let D^j be any of the t connected components in D, and let T^j be any spanning tree in D^j. As T^j is a bipartite graph, it is possible to partition the set of its vertices into two sets V^j_1 and V^j_2 in polynomial time. Moreover, by the connectivity of T^j, every vertex $v \in V^j_\ell$ ($\ell \in \{1, 2\}$) is adjacent to some vertex in $V^j_{3-\ell}$, and thus it is easy to find a set E^j_ℓ of edges in T^j such that $(V(D^j), E^j_\ell)$ is a forest of stars with centers in V^j_ℓ and leaves in $V^j_{3-\ell}$. Therefore, for $\ell = 1, 2$, $F^\ell = \bigcup_{j=1}^t E^j_\ell$ are two sets of edges satisfying (i) and (ii). Furthermore, $\bigcup_{j=1}^t (V^j_1 \cup V^j_2) = \bigcup_{j=1}^t V(D^j) = V(D)$. As a consequence, from Lemma 8, (iii) is also satisfied. □

To compute $\hat{F}_i$, the algorithm does the following. Our algorithm uses the well-known FPTAS for the *Knapsack Problem* to compute a $(1+\epsilon/(2h))$-approximate solution S_i for the following instance of knapsack. For each v_j, consider the blue edge $e \in B_i$ incident to v_j with cheapest activation cost. We create an object o^i_j of profit $c_i(v_0, v_j)$ and volume $\gamma_i(e)$; we say that e is associated with the object o^i_j. Finally, the volume of the knapsack is Δ. Denote by K_i the input instance of knapsack. The solution S_i for K_i identifies a set of blue edges, namely $B' = \{e \in B_i \mid \exists o^i_j \in S_i$ s.t. e is associated with $o^i_j\}$. The algorithm then uses the decomposition algorithm described in Lemma 9 to find two subsets of edges F^1 and F^2, and then it sets $\hat{F}_i$ to F^1 if $r(F^1) \geq r(F^2)$, and to F^2 otherwise. The pseudocode of the algorithm is given in Algorithm 1.

Theorem 7. *Algorithm 1 computes a $(2h+\epsilon)$-approximate solution in polynomial time for Budget-StackMST, for any constant $\epsilon > 0$.*

Proof. Remind that $G^*_i = (\{v_0, \dots, v_{\ell_i}\}, F^*_i)$, where F^*_i is obtained by mapping some of the edges of an optimal solution F^* to the blue edges of the auxiliary instance T_i. In order to show a lower bound for the profit of the optimal solution of K_i, we define a feasible solution S^*_i as follows: for each connected component H of G^*_i, let $v \in V(H)$ be the vertex that minimizes $c(v_0, v)$, and consider any spanning tree $\mathcal{T}_H$ of H rooted at v. Notice that for each $v_j \in V(H) \setminus \{v\}$ we have an object o^i_j whose volume is at most $\gamma(\bar{v}_j, v_j)$, where $\bar{v}_j$ denotes the parent of v_j in $\mathcal{T}_H$. Add such objects to the solution S^*_i. Once we have considered all the connected components of G^*_i, the solution S^*_i contains a set of objects of total volume at most Δ (since the solution F^* is feasible). Moreover, by construction, the profit of S^*_i must be at least

$$\mathtt{profit}\big(S^*_i\big) \geq c(T_i) - \sum_{H \in \mathrm{comp}(G^*_i)} \min_{v \in V(H)} c_i(v_0, v).$$

We now bound the revenue $r(\hat{F}_i)$. Since, from Lemma 7 and Lemma 9, $r(F_i) \geq r(\hat{F}_i) \geq \frac{1}{2}\mathtt{profit}(S_i)$, we have that

$$c(T_i) - \sum_{H \in \mathrm{comp}(G^*_i)} \min_{v \in V(H)} c_i(v_0, v) \leq \mathtt{profit}\big(S^*_i\big) \leq \left(1 + \frac{\epsilon}{2h}\right)\mathtt{profit}(S_i) \leq \left(2 + \frac{\epsilon}{h}\right) r(\hat{F}_i) \leq \left(2 + \frac{\epsilon}{h}\right) r(F_i).$$

By summing over all levels and using Lemma 6, we obtain

$$r\big(F^*\big) \leq c(T) - \sum_{i=1}^h \sum_{H \in \mathrm{comp}(G^*_i)} \min_{v \in V(H)} c_i(v_0, v) \leq \left(2 + \frac{\epsilon}{h}\right) \sum_{i=1}^h r(F_i) \leq (2h + \epsilon) \max_{i=1,\dots,h} r(F_i).$$

This completes the proof. □

6. Conclusions and open problems

In this paper we have presented a generalized budgeted version of the StackMST problem, and we have studied few variants of a specialized version of StackMST in which the set of priceable blue edges coincides with the complement of the given red edges, thus maximizing the leader's strategy space.

Concerning the latter problem, we showed that when there are only 2 red edge costs it can be solved optimally in polynomial time (against the APX-hardness and the 2-approximability of the corresponding StackMST problem), while in the general case it admits a $(7/4+\epsilon)$-approximation, for any $\epsilon > 0$, against a general $O(\log n)$-approximation ratio holding for StackMST. On the other hand, quite surprisingly, the former problem is approximable within the same bound as StackMST, and moreover we were able to provide a refinement of the corresponding approximation algorithm in which the radius of the input red tree does enter in the approximation factor.

Many intriguing problems are left open. Among the others, we first of all clearly mention that of fixing the computational complexity of Complete-StackMST. Establishing its NP-hardness, as we conjecture, would obviously provide more relevance to our given approximation algorithm. Along this direction, a possible intermediate step would be that of studying the fixed-parameter tractability of the problem, once the parameter is assumed to be the number of distinct red edge costs in the input. Moreover, in an effort of designing a better approximation algorithm for Complete-StackMST, it would be nice to study whether other notable topologies of the red tree (e.g., bounded-degree, full k-ary, etc.) allow for a better approximation (or even exact) algorithm, as for the path case.

Concerning the Budget-StackMST, it would be nice to prove a stronger inapproximability result for it as compared to StackMST. On the other side, we plan to study an intermediate variant between StackMST and Budget-StackMST, namely that in which the instance has *uniform* activation costs.

On a more general side, we mention that there is still a large gap to be filled between the lower (a constant close to 1) and upper (a logarithmic factor) bound on the approximability of StackMST. Finally, it would be interesting to investigate the possibility of integrating our knapsack-based approach to other budgeted versions falling in the general framework of Stackelberg network pricing games.

Acknowledgements

We wish to thank the anonymous reviewers for their insightful comments and for their helpful suggestions for improving the paper.

References

[1] P. Alimonti, V. Kann, Some APX-completeness results for cubic graphs, Theoret. Comput. Sci. 237 (1–2) (2000) 123–134.
[2] D. Bilò, L. Gualà, G. Proietti, P. Widmayer, Computational aspects of a 2-player Stackelberg shortest paths tree game, in: Proc. of the 4th Int. Workshop on Internet & Network Economics (WINE), in: LNCS, vol. 5385, Springer, 2008, pp. 251–262.
[3] D. Bilò, L. Gualà, G. Proietti, Hardness of an asymmetric 2-player Stackelberg network pricing game, in: Electronic Colloquium on Computational Complexity (ECCC), 2009, No. TR09-112.
[4] P. Briest, M. Hoefer, P. Krysta, Stackelberg network pricing games, Algorithmica 62 (3–4) (2012) 733–753.
[5] P. Briest, M. Hoefer, L. Gualà, C. Ventre, On Stackelberg pricing with computational bounded consumers, Networks 60 (1) (2012) 31–44.
[6] P. Briest, S. Khanna, Improved hardness of approximation for Stackelberg shortest-path pricing, in: Proc. of the 6th Int. Workshop on Internet & Network Economics (WINE), in: LNCS, vol. 6484, Springer, 2010, pp. 444–454.
[7] J. Cardinal, E.D. Demaine, S. Fiorini, G. Joret, S. Langerman, I. Newman, O. Weimann, The Stackelberg minimum spanning tree game, Algorithmica 59 (2) (2011) 129–144.
[8] J. Cardinal, E.D. Demaine, S. Fiorini, G. Joret, I. Newman, O. Weimann, The Stackelberg minimum spanning tree game on planar and bounded-treewidth graphs, J. Comb. Optim. 25 (1) (2013) 19–46.
[9] A. Grigoriev, S. van Hoesel, A. van der Kraaij, M. Uetz, M. Bouhtou, Pricing network edges to cross a river, in: Proc. of the 3rd Workshop on Approximation and Online Algorithms (WAOA), in: LNCS, vol. 3351, Springer, 2005, pp. 140–153.
[10] G. Joret, Stackelberg network pricing is hard to approximate, Networks 57 (2) (2011) 117–120.
[11] M. Labbé, P. Marcotte, G. Savard, A bilevel model of taxation and its application to optimal highway pricing, Manag. Sci. 44 (12) (1998) 608–622.
[12] S. Roch, G. Savard, P. Marcotte, An approximation algorithm for Stackelberg network pricing, Networks 46 (1) (2005) 57–67.
[13] R.E. Tarjan, Sensitivity analysis of minimum spanning trees and shortest path trees, Inform. Process. Lett. 14 (1) (1982) 30–33.
[14] S. van Hoesel, An overview of Stackelberg pricing in networks, European J. Oper. Res. 189 (3) (2008) 1393–1402.
[15] H. von Stackelberg, Marktform und Gleichgewicht (Market and Equilibrium), Verlag von Julius Springer, Vienna, Austria, 1934.