The Stackelberg Minimum Spanning Tree Game*

Jean Cardinal[1], Erik D. Demaine[2], Samuel Fiorini[3],
Gwenaël Joret[1**], Stefan Langerman[1***], Ilan Newman[4], and Oren Weimann[2]

[1] Computer Science Department, Université Libre de Bruxelles,
B-1050 Brussels, Belgium, {jcardin,gjoret,slanger}@ulb.ac.be
[2] MIT Computer Science and Artificial Intelligence Laboratory,
Cambridge, MA 02139, USA, {edemaine,oweimann}@mit.edu
[3] Department of Mathematics, Université Libre de Bruxelles,
B-1050 Brussels, Belgium, sfiorini@ulb.ac.be
[4] Department of Computer Science, University of Haifa,
Haifa 31905, Israel, ilan@cs.haifa.ac.il

Abstract. We consider a one-round two-player network pricing game, the *Stackelberg Minimum Spanning Tree* game or StackMST. The game is played on a graph (representing a network), whose edges are colored either red or blue, and where the red edges have a given fixed cost (representing the competitor's prices). The first player chooses an assignment of prices to the blue edges, and the second player then buys the cheapest possible minimum spanning tree, using any combination of red and blue edges. The goal of the first player is to maximize the total price of purchased blue edges. This game is the minimum spanning tree analog of the well-studied Stackelberg shortest-path game.

We analyze the complexity and approximability of the first player's best strategy in StackMST. In particular, we prove that the problem is APX-hard even if there are only two different red costs, and give an approximation algorithm whose approximation ratio is at most $\min\{k, 3 + 2 \ln b, 1 + \ln W\}$, where k is the number of distinct red costs, b is the number of blue edges, and W is the maximum ratio between red costs. We also give a natural integer linear programming formulation of the problem, and show that the integrality gap of the fractional relaxation asymptotically matches the approximation guarantee of our algorithm.

1 Introduction

Suppose that you work for a networking company that owns many point-to-point connections between several locations, and your job is to sell these connections. A customer wants to construct a network connecting all pairs of locations in the form of a spanning tree. The customer can buy connections that you are selling, but can also buy connections offered by your competitors. The customer will always buy the cheapest possible spanning tree. Your company has researched

* This work was partially supported by the *Actions de Recherche Concertées (ARC)* fund of the *Communauté française de Belgique*.

** Aspirant F.R.S. – FNRS.

*** Chercheur qualifié F.R.S. – FNRS.

the price of each connection offered by the competitors. The problem considered in this paper is how to set the price of each of your connections in order to maximize your revenue, that is, the sum of the prices of the connections that the customer buys from you.

This problem can be cast as a *Stackelberg game*, a type of two-player game named introduced by the German economist Heinrich Freiherr von Stackelberg [18]. In a Stackelberg game, there are two players: the *leader* moves first, then the *follower* moves, and then the game is over. The follower thus optimizes its own objective function, knowing the leader's move. The leader has to optimize its own objective function by anticipating the optimal response of the follower. In the scenario depicted in the preceding paragraph, you were the leader and the customer was the follower: you decided how to set the prices for the connections that you own, and then the customer selected a minimum spanning tree. In this situation, there is an obvious tradeoff: the leader should not put too a high price on the connections—otherwise the customer will not buy them—but on the other hand the leader needs to put sufficiently high prices to optimize revenue.

Formally, the problem we consider is defined as follows. We are given an undirected graph $G = (V, E)$ whose edge set is partitioned into a *red edge set* R and a *blue edge set* B. Each red edge $e \in R$ has a nonnegative fixed *cost* $c(e)$ (the best competitor's price). The leader owns every blue edge $e \in B$ and has to set a *price* $p(e)$ for each of these edges. The cost function c and price function p together define a *weight* function w on the whole edge set. By "weight of edge e" we mean either "cost of edge e" if e is red or "price of edge e" if e is blue. A spanning tree T is a *minimum spanning tree* (MST) if its *total weight*

$$\sum_{e \in E(T)} w(e) = \sum_{e \in E(T) \cap R} c(e) + \sum_{e \in E(T) \cap B} p(e) \tag{1}$$

is minimum. The *revenue* of T is then

$$\sum_{e \in E(T) \cap B} p(e). \tag{2}$$

The Stackelberg Minimum Spanning Tree problem, StackMST, asks for a price function p that maximizes the revenue of an MST. Throughout, we assume that the graph contains a spanning tree whose edges are all red; otherwise, there is a cut consisting only of blue edges and the optimum value is unbounded. Moreover, to avoid being distracted by epsilons, we assume that among all edges of the same weight, blue edges are always preferred to red edges; this is a standard assumption. As a consequence, all minimum spanning trees for a given price function p have the same revenue; see Section 2 for details.

Related work. A similar pricing problem, where the customer wants to construct a shortest path between two vertices instead of a spanning tree, has been studied in the literature; see van Hoesel [17] for a survey. Complexity and approximability results have recently been obtained by Roch, Savard and Marcotte [14], and by Grigoriev, van Hoesel, Kraaij, Uetz, and Bouhtou [9]: the problem is strongly NP-hard and $O(\log |B|)$-approximable. A generalization of the problem to more than one customer has been tackled using mathematical programming tools, in particular bilevel programming; see Labbé, Marcotte, and Savard [12]. This generalization was motivated by the problem of setting tolls on highway networks. Cardinal, Labbé, Langerman, and Palop [3] give a geometric version of the problem.

Sometimes the goal of the leader is not to invite the followers to use his/her part of the network and maximize his/her own revenue but to encourage socially acceptable or optimal behaviors among the followers (the users of the network) so as to maximize some global objective. These kinds of Stackelberg games have been studied recently, e.g., by Cole, Dodis and Roughgarden [4] and Swamy [16]. An extensive bibliography on similar networking games has been compiled recently by Altman et al. [2]. In another Stackelberg game studied by Roughgarden [15], the leader is a job scheduler whose goal is to compute a scheduling strategy for the jobs he/she controls such that total latency in the system is minimized after the followers have selfishly scheduled their jobs.

Hartline and Koltun [10] propose approximation algorithms for several APX-hard pricing problems, where the goal is to find the best prices for a set of items, given knowledge of the consumer's behavior in the form of a combinatorial preference structure.

Finally, our problem should not be confused with other spanning tree games found in cooperative games and mechanism design theory [8,11], with parametric spanning tree problems [7,6], or with two-stage stochastic minimum spanning tree problems [5].

Our results. We analyze the complexity and approximability of the StackMST problem. Specifically, we prove the following:

1. StackMST is APX-hard, even if there are only two red costs, 1 and 2 (Section 3). This result is also the first NP-hardness proof for this problem. The reduction is from SetCover.
2. StackMST is $O(\log n)$-approximable, and is $O(1)$-approximable when the red costs either fall in a constant-size range or have a constant number of distinct values (Section 4). More precisely, we analyze the following simple approximation algorithm, called *Best-out-of-k*: for all i between 1 and k, consider the price function for which all blue edges have price c_i, and output the best of these k price functions. Here, and throughout the paper, c_i denotes the ith smallest cost of a red edge and k the number of distinct red costs. We prove that the approximation ratio of this algorithm is bounded above by $\min\{k, 3 + 2\ln b, 1 + \ln(c_k/c_1)\}$, where b is the number of blue edges.
3. The integrality gap of a natural integer linear programming formulation asymptotically matches the approximation guarantee of Best-out-of-k (Section 5). Thus, effectively, any approximation algorithm based on the linear programming relaxation of our integer program (or any weaker relaxation) cannot do better than Best-out-of-k. Of course, this result does not imply that Best-out-of-k is optimal. In fact, a central open question about StackMST is to determine if it admits a constant factor approximation algorithm.

For brevity, some of the proofs are omitted and will appear in the full version of this paper.

2 Basic Results

Before we proceed to our main results, we prove a few basic lemmas about StackMST.

We claimed in the introduction that the revenue of the leader depends on the price function p only, and not on the particular MST picked by the follower. To see this, let $w_1 < w_2 < \cdots < w_\ell$ denote the different edge weights. The greedy algorithm (a.k.a. Kruskal's algorithm) will work in ℓ phases: in its ith phase, it will consider all blue edges of price w_i (if any) and then all red edges of cost w_i (if any). The number of blue edges selected in the ith phase will not depend on the order in which blue or red edges of weight w_i are considered. This shows the claim. Moreover, if there is no red edge of cost w_i then p is not an optimal price function because the leader can raise the price of every blue edge of price w_i to the next weight w_{i+1} and thus increase his/her revenue. This implies the following lemma.

Lemma 1. *In every optimal price function, the prices assigned to the blue edges appearing in some MST belong to the set* $\{c(e) : e \in R\}$. □

Notice that the prices given to the blue edges that are in no MST do not really matter (as long as they are high enough). We find it convenient to see them as equaling ∞. This has the same effect as deleting those blue edges. A direct consequence of Lemma 1 is that the decision version of StackMST belongs to NP, using some price function p with $p(e) \in \{c(e) : e \in R\} \cup \{\infty\}$ for all $e \in B$ as a certificate. Another possibility for a certificate is an acyclic set of blue edges F, interpreted as the set of blue edges in any MST. Given F, we can easily compute an optimal price function such that F is the set of blue edges in any MST, with the help of Lemma 2 below. In the lemma, we denote by $\mathcal{C}_e$ the set of cycles of $G = (V, E)$ that include some edge e. (Notice that $\mathcal{C}_e$ is nonempty whenever e is blue because $G_r = (V, R)$ is connected.)

Lemma 2. *Consider a price function p, a corresponding minimum spanning tree T, and let $F = E(T) \cap B$. Then for every $e \in F$, we have*

$$p(e) \leq \min_{C \in \mathcal{C}_e} \max_{e' \in E(C) \cap R} c(e'). \tag{3}$$

Moreover, whenever F is any acyclic set of blue edges and we set $p(e)$ equal to the right hand side of (3) for $e \in F$ and $p(e) = \infty$ for $e \in B - F$, we have $E(T') \cap B = F$ for any corresponding MST T'.

It follows from the above lemma that StackMST is fixed parameter tractable with respect to the number of blue edges. Indeed, to solve the problem, one could try all acyclic subsets F of B, and for each of them put the prices as above (this can easily be done in polynomial time), and finally take the solution yielding the highest revenue. We conclude this section by stating a useful property satisfied by all optimal solutions of StackMST.

Lemma 3. *Let p be an optimal price function and T be a corresponding MST. Suppose that there exists a red edge e in T and a blue edge f not in T such that e belongs to the unique cycle C in $T + f$. Then there exists a blue edge f' distinct from f in C such that $c(e) < p(f') \leq p(f)$.*

3 Complexity and Inapproximability

By Lemma 1, StackMST is trivially solved when the cost of every red edge is exactly 1, i.e., when $c(e) = 1$ for all $e \in R$. In this section, we show that the problem is APX-hard even when the costs of the red edges are only 1 and 2, i.e., when $c(e) \in \{1, 2\}$ for all $e \in R$. We start with NP-hardness:

Theorem 1. StackMST *is NP-hard even when* $c(e) \in \{1, 2\}$ *for all* $e \in R$*.*

Proof. We present a reduction from SetCover (in its decision version). Let $(\mathcal{U}, \mathcal{S})$ and the integer t be an instance of SetCover, where $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$, and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$. Without loss of generality, we assume that $u_n \in S_i$ for every $i = 1, 2, \ldots, m$ (we can always add one element to $\mathcal{U}$ and to every S_i to make sure this holds).

We construct a graph $G = (V, E)$ with edge set $E = R \cup B$ and a cost function $c : R \to \{1, 2\}$ as follows. The vertex set of G is $\mathcal{U} \cup \mathcal{S} = \{u_1, u_2, \ldots, u_n\} \cup \{S_1, S_2, \ldots, S_m\}$. The edge set of G and cost function c are defined as follows:

- there is a red edge of cost 1 linking u_i and u_{i+1} for every $1 \leq i < n$;
- there is a red edge of cost 2 linking u_n and S_1, and linking S_j and S_{j+1} for every $1 \leq j < m$;
- whenever $u_i \in S_j$ we link u_i and S_j by a blue edge.

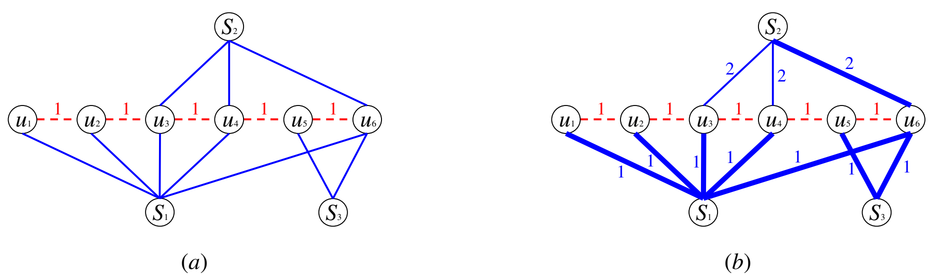

Fig. 1. (a) The graph G constructed for $n = 6$, $m = 3$ with $S_1 = \{u_1, u_2, u_3, u_4, u_6\}$, $S_2 = \{u_3, u_4, u_6\}$ and $S_3 = \{u_5, u_6\}$. The red edges of cost 2 are omitted for clarity. The red edges of cost 1 are dashed, and the blue edges are solid. (b) An optimal price function p on the blue edges that yields a revenue of 9, an example MST is depicted in bold.

We illustrate such a construction in Fig. 1. We claim that $(\mathcal{U}, \mathcal{S})$ has a set cover of size t if and only if there exists a price function $p : B \to \{1, 2, \infty\}$ for the blue edges of G whose revenue is $n + 2m - t - 1$.

$(\Rightarrow)$ Suppose $(\mathcal{U}, \mathcal{S})$ has a set cover of size t. We construct p as follows: for every blue edge $e = u_i S_j$, we set $p(e)$ to be 1 if S_j is in the set cover, and 2 otherwise. We show that the revenue of p equals $n + 2m - t - 1$ by running Kruskal's MST algorithm starting with an empty tree, T. Because the blue edges of weight 1 are the lightest, we start with adding them one by one to T such that we add an edge only if it doesn't close a cycle in T. After going over all blue edges of weight 1, we are guaranteed that T is a tree that spans all the vertices u_i for every $i = 1, \dots, n$, and every vertex S_j such that S_j is in the set cover. This is because these vertices are connected through u_n with only blue edges of weight 1. So the current weight of T is $|T| - 1 = n + t - 1$. We next try to add the red edges of weight 1, but every such edge connects two vertices, u_i and u_{i+1}, already spanned by T and therefore closes a cycle, so we add none of them. Next we add the blue edges of weight 2. For every S_j not in the set cover, we connect S_j to T with one blue edge of weight 2 (the second one will close a cycle). Therefore, after going over all the blue edges of weight 2, we added a weight of $2(m - t)$ to T. Furthermore, T spans the entire graph so there is no need to add any red edges of weight 2. All the edges in T are blue and the revenue of T is $(n + t - 1) + 2(m - t) = n + 2m - t - 1$.

($\Leftarrow$) Suppose that there exists a price function $p : B \to \{1, 2, \infty\}$ for the blue edges of G whose revenue is $n + 2m - t - 1$ for some t. By Lemma 1, there exists such a function p that is optimal. Choose then $p : B \to \{1, 2, \infty\}$ as an optimal price function that minimizes the number of red edges in an MST T.

Assume first that T contains only blue edges. Then every vertex u_i is incident to some blue edge in T with price 1. Thus the set $\mathcal{S}'$ of those S_j's that are linked to some blue edge in T with price 1 is a set cover of $(\mathcal{U}, \mathcal{S})$. On the other hand, notice that any $S_j \in \mathcal{S} \setminus \mathcal{S}'$ is a leaf of T, because if there were two blue edges $u_i S_j, u_{i+\ell} S_j$ in T then none of them could have a price of 2 because of the cycle $S_j u_i u_{i+1} \dots u_{i+\ell} S_j$. Therefore, the revenue of p equals $(n+|\mathcal{S}'|-1)+2(m-|\mathcal{S}'|) = n + 2m - |\mathcal{S}'| - 1$. As by hypothesis this is at least $n + 2m - t - 1$, we deduce that the set cover $\mathcal{S}'$ has size at most t.

Suppose now that T contains some red edge e and denote by X_1 and X_2 the two components of $T - e$. There exists some blue edge $f = u_i S_j$ in G that connects X_1 and X_2 because the graph (V, B) induced by the blue edges is connected (because u_n is linked with blue edges to every S_j). By Lemma 3, there exists a blue edge $f' = u_{i'} S_{j'}$ distinct from f in the unique cycle C in $T + f$ such that $c(e) < p(f') \leq p(f)$. In particular, we have $c(e) = 1$ and $p(f') = 2$. By an argument given in the preceding paragraph, $S_{j'}$ is a leaf of T, hence we have $j' = j$. Also, every blue edge distinct from f and f' in C has price 1. But then the price function p' obtained from p by setting the price of both f and f' to 1 is also optimal and has a corresponding MST that uses less red edges than T, namely $T - e + f$, a contradiction. This completes the proof. $\square$

The reduction used in Theorem 1 implies a stronger hardness result.

Theorem 2. StackMST *is APX-hard even when $c(e) \in \{1, 2\}$ for all $e \in R$.*

The above theorem is proved by showing that, for any $\varepsilon > 0$, a $(1 - \varepsilon)$-approximation for StackMST implies a $(1 + 8\varepsilon)$-approximation for Vertex-Cover in graphs of maximum degree at most 3. The theorem then follows from the APX-hardness of the latter problem [1,13]. The formal details will appear in the full version of this paper.

4 The Best-Out-Of-k Algorithm

As before, let k denote the number of distinct red costs, and let $c_1 < c_2 < \cdots < c_k$ denote those costs. Without loss of generality, we assume that all weights are positive (otherwise we contract all red edges of cost 0). Recall that the Best-out-of-k algorithm is as follows. For each j between 1 and k, set $p(e) = c_j$ for all blue edges $e \in B$ and compute an MST T_j. Then pick j such that the revenue of T_j is maximum and output the corresponding feasible solution. We analyze the approximation ratio ensured by this algorithm.

Theorem 3. *Best-out-of-k is a ρ-approximation, where*

$$\rho = 1 + \sum_{i=2}^{k} \frac{c_i - c_{i-1}}{c_i} = 1 + \frac{c_2 - c_1}{c_2} + \cdots + \frac{c_k - c_{k-1}}{c_k}.$$

Proof. Consider a minimum cost red spanning tree, that is, an MST obtained after setting the prices on all blue edges to ∞. For $j = 1, \ldots, k$, let m_j denote the number of red edges of cost c_j in that tree. Similarly, denote by m'_j the number of red edges of cost c_j in any MST the follower can select when all blue edges are for free (i.e., have a price of 0). Then we can bound the optimal revenue as follows:

$$\mathrm{OPT} \leq \sum_{i=1}^{k} c_i m_i - \sum_{i=1}^{k} c_i m'_i = \sum_{i=1}^{k} c_i (m_i - m'_i). \tag{4}$$

Let b_j be the number of blue edges in T_j, the MST computed by Best-out-of-k at step j, and set $b_{k+1} = 0$. Let also R_j denote the red edges of cost at most c_j. For $E' \subseteq E$, we denote by G/E' the graph obtained from G after contraction of every edge in E'. Because the number of edges in a set $F \subseteq E$ inducing a forest in G equals $|G| - |G/F|$ (where $|H|$ denotes the number of vertices of graph H), we obtain

$$\begin{aligned} m_j &= |G/R_{j-1}| - |G/R_j|, \\ m'_j &= |G/(R_{j-1} \cup B)| - |G/(R_j \cup B)|, \\ b_j &= |G/R_{j-1}| - |G/(R_{j-1} \cup B)|, \end{aligned}$$

and so we deduce

$$m_j - m'_j = b_j - b_{j+1}. \tag{5}$$

Thus the revenue given by T_j, which we denote q_j, is exactly

$$q_j = c_j b_j = \sum_{i=j}^{k} c_j (b_i - b_{i+1}) = \sum_{i=j}^{k} c_j (m_i - m'_i).$$

Then (5) implies:

$$m_j - m'_j = \frac{q_j}{c_j} - \frac{q_{j+1}}{c_{j+1}} \quad \text{for } 1 \leq j \leq k-1 \quad \text{and} \quad m_k - m'_k = \frac{q_k}{c_k}.$$

Therefore we can rewrite our upper bound on OPT given in (4) in terms of the q_j's:

$$\begin{aligned}\sum_{i=1}^{k} c_i(m_i - m_i') &= c_1\left(\frac{q_1}{c_1} - \frac{q_2}{c_2}\right) + c_2\left(\frac{q_2}{c_2} - \frac{q_3}{c_3}\right) + \cdots + c_k\frac{q_k}{c_k} \\ &= q_1 + \frac{c_2 - c_1}{c_2}q_2 + \cdots + \frac{c_k - c_{k-1}}{c_k}q_k.\end{aligned}$$

Now consider the index j such that q_j is maximum. The above equation and (4) together imply:

$$\begin{aligned}\frac{\mathrm{OPT}}{q_j} \le \frac{\sum_{i=1}^{k} c_i(m_i - m_i')}{q_j} &= \frac{q_1}{q_j} + \frac{c_2 - c_1}{c_2}\cdot\frac{q_2}{q_j} + \cdots + \frac{c_k - c_{k-1}}{c_k}\cdot\frac{q_k}{q_j} \qquad (6)\\ &\le 1 + \frac{c_2 - c_1}{c_2} + \cdots + \frac{c_k - c_{k-1}}{c_k} = \rho.\end{aligned}$$

So Best-out-of-k is a ρ-approximation algorithm. (The last inequality follows from the maximality of q_j.) □

Observe that the above proof shows that Best-out-of-k can be implemented to run within the same time complexity as an MST algorithm. Indeed, when the price of all blue edges is set to c_j, the resulting revenue is $\sum_{i=j}^{k} c_j(m_i - m_i')$. Thus we can find which c_j would maximize the revenue simply by computing the m_i's and m_i''s, which can be done by computing an MST of (V, R) and $(V, R \cup B)$, respectively (where the edges in B have price 0).

Note also that if the costs are exactly 1, 2, ..., k, then ρ equals the kth harmonic number $H_k = 1 + 1/2 + \cdots + 1/k$. In general, $\rho - 1$ can be regarded as a Riemann (under-)approximation of the integral $\int_{c_1}^{c_k} \frac{1}{t}\,\mathrm{d}t$. So we have

$$\rho \le 1 + \int_{c_1}^{c_k} \frac{1}{t}\,\mathrm{d}t = 1 + \ln c_k - \ln c_1 = 1 + \ln W,$$

where $W = c_k/c_1$. We also have $\rho \le k$, because ρ is the sum of k terms, all not exceeding 1. Moreover, as we now prove, Equation (6) implies the following result leading to the conclusion that Best-out-of-k is, in particular, a $\min\{k, 3 + 2\ln|B|), 1 + \ln(\frac{c_k}{c_1})\}$ - approximation.

Proposition 1. *Best-out-of-k is a $(3 + 2\ln b)$-approximation, where b is the number of blue edges.*

Proof. Recall q_i is the revenue of the ith tree computed by Best-out-of-k, and q_j is their maximum. Let $c_0 = 0$ and ℓ be the index where $c_{\ell-1} < \frac{c_k}{b^2}$ and $c_\ell \ge \frac{c_k}{b^2}$. Without loss of generality, we assume $q_k \ne 0$ (otherwise we focus on the last q_i that is non-zero) so we have $c_k \le q_k \le q_j$. We then deduce

$$q_i \le c_i \cdot m < \frac{c_k}{b^2}\cdot b \le \frac{q_j}{b} \text{ for every } 1 \le i \le \ell - 1. \qquad (7)$$

Equation (6) in the proof of Theorem 3 and Equation (7) together imply:

$$
\begin{aligned}
\frac{\mathrm{OPT}}{q_j} &\le \sum_{i=1}^{\ell-1} \frac{c_i - c_{i-1}}{c_i}\frac{q_i}{q_j} + \sum_{i=\ell}^{k} \frac{c_i - c_{i-1}}{c_i}\frac{q_i}{q_j} \\
&\le \sum_{i=1}^{\ell-1} \frac{q_i}{q_j} + \sum_{i=\ell}^{k} \frac{c_i - c_{i-1}}{c_i} \\
&< \sum_{i=1}^{\ell-1} \frac{1}{b} + 1 + \int_{c_\ell}^{c_k} \frac{1}{t}\,\mathrm{d}t \\
&< \ell/b + 1 + \ln\frac{c_k}{c_\ell} \le \ell/b + 1 + \ln(b^2) = \ell/b + 1 + 2\ln b.
\end{aligned}
$$

To complete the proof we describe a procedure to simplify the StackMST instance in order to ensure $\ell/b \le 2$. First, as long as some vertex v of the graph has no blue edge incident to it, we contract the cheapest edge in $\delta(v) = \{e \in E : v \in e\}$. Next, we remove the most expensive edge in every red cycle in the graph, until the red edges form a spanning tree. As is easily verified, the resulting StackMST instance is equivalent to the original one. That is, the set of blue edges does not change and the revenue of every price function is the same for both instances. So for the analysis we can assume that every vertex has some blue edge incident to it and the red edges form a spanning tree. Therefore, we have $b \ge n/2 \ge (k+1)/2 \ge \ell/2$ and Best-out-of-k is a $(3+2\ln b)$-approximation. □

A natural generalization of StackMST to matroids is as follows. Given a matroid $(S, \mathcal{I})$ with $\mathcal{I}$ partitioned into two sets $\mathcal{R}$ and $\mathcal{B}$, and nonnegative costs on the elements of $\mathcal{R}$, assign prices on the elements of $\mathcal{B}$ in such a way that the revenue given by a minimum weight basis of $(S, \mathcal{I})$ is maximized. We mention that the analysis of Best-out-of-k given in the proof of Theorem 3 extends swiftly to the case of matroids, yielding a ρ-approximation algorithm in this more general setting.

5 Linear Programming Relaxation

In this section, we give an integer programming formulation for the problem and study its linear programming relaxation. In this section, all costs c_i are assumed to be positive. For each $j = 1, \ldots, k$, and each blue edge $e \in B$ we define a variable $x_{j,e}$. The interpretation of these variables is as follows: think of a feasible solution $p : B \to \{c_1, c_2, \ldots, c_k\}$ and a minimum spanning tree T with respect to p. Then $x_{j,e} = 1$ means that the blue edge e appears in T, with a price $p(e)$ of at least c_j.

We let $c_0 = 0$ and denote by R_j the set of red edges of cost at most c_j. For t pairwise disjoint sets of vertices $C_1, \ldots, C_t$, we denote by $\delta_B(C_1 : C_2 : \cdots : C_t)$ the set of blue edges that are in the cut defined by these sets. The integer programming formulation then reads:

$$\text{(IP)}\quad \max \sum_{\substack{e\in B\\ 1\le j\le k}} (c_j - c_{j-1})x_{j,e}$$

$$\begin{aligned}
\text{s.t.}\quad & \sum_{e\in\delta_B(C_1:C_2:\cdots:C_t)} x_{j,e} \le t-1 && \forall j\ge 1, && (8)\\
& && \forall C_1,...,C_t \text{ components of } (V,R_{j-1});\\
& \sum_{e\in P\cap B} x_{1,e} + x_{j,f} \le |P\cap B| && \forall f = ab\in B, \forall j\ge 2, && (9)\\
& && \forall P\ ab\text{-path in } (B\cup R_{j-1}) - f;\\
& x_{1,e}\ge x_{2,e}\ge\cdots\ge x_{k,e}\ge 0 && \forall e\in B; && (10)\\
& x_{j,e}\in\{0,1\} && \forall j, \forall e\in B. && (11)
\end{aligned}$$

Proposition 2. *The integer program above is a formulation of* StackMST.

As already noted, $\sum_{j=1}^k c_j(m_j - m'_j)$ is an upper bound on OPT (see Section 4). The rest of this section is devoted to the LP relaxation of the above IP, obtained by dropping constraint (11). We will show that the LP is tractable and that it provides an upper bound on OPT at least as good as $\sum_{j=1}^k c_j(m_j - m'_j)$. On the other hand, its integrality gap turns out to be k on instances with k distinct costs, thus matching the guarantee given by the Best-out-of-k algorithm. (Let us recall that the integrality gap of the LP on a specified set of instances is defined as the supremum of the ratio (LP)/(IP) over these instances.)

Proposition 3. *The LP can be separated in polynomial time.*

Proposition 4. *We have* (IP) $\le$ (LP) $\le \sum_{j=1}^k c_j(m_j - m'_j)$.

Proposition 5. *The integrality gap of the LP is k on instances with k distinct costs.*

Proof. We already know from Proposition 4 that the integrality gap is at most k on instances with k distinct costs. In order to show that it is also at least k, we define an instance of StackMST as follows:

- choose an integer $a\ge 2$;
- the graph has $a^{k-1}+1$ vertices, the set of whose is denoted $V = \{v_0, v_1, \ldots, v_{a^{k-1}}\}$;
- the set of blue edges is a spanning star with v_0 as center, i.e. $B = \{v_0v_i | 1 \le i \le a^{k-1}\}$;
- the ith red cost is $c_i = a^{i-1}$, for $1\le i\le k$;
- the components of the graph (V, R_i), where R_i is the set of red edges of cost at most c_i (and $R_0 = \emptyset$), are $\{v_1,\ldots,v_{a^i}\}, \{v_{a^i+1},\ldots,v_{2a^i}\},\ldots,\{v_{(a^{k-i}-1)a^i+1},\ldots,v_{a^{k-i}a^i}\}$, for $1\le i\le k-1$;
- the unique component of (V, R_k) is V.

We didn't define explicitely the set of red edges in the above description. This is because, as shown by the IP formulation, it is sufficient to give the components of the graph (V, R_i) for $i = 1, 2, \ldots, k$. (Notice for instance that we may always 'realize' these components with a set of red edges inducing a path.)

Consider an optimal solution of the StackMST problem for the instance defined above, and let T be a corresponding MST. Look at any blue edge e in T, of price c_i, and let C_e be the unique component of $(V - v_0, R_{i-1})$ that contains an endpoint of e. No other blue edge of T has an endpoint in C_e, because otherwise T has not minimum weight. Thus, if e and f are two distinct blue edges of T, then $C_e \cap C_f = \emptyset$. Noticing that the price given to e is $c_i = a^{i-1} = |C_e|$, we deduce that the revenue given by T is

$$\sum_{e \in B \cap T} |C_e| \leq a^{k-1}.$$

Moreover, a revenue of a^{k-1} is easily achieved, set for instance all blue edges to the same price c_i for some $i \in \{1, \ldots, k\}$. Hence, (IP) $= a^{k-1}$.

We now define a feasible solution x^* for the LP. The point x^* will have the property that $x^*_{i,e} = x^*_{i,f}$ for $1 \leq i \leq k$ and all $e, f \in B$. We thus let $y_i = x^*_{i,e}$ for $e \in B$. The constraints on the y_i's imposed by the LP are then:

$$\begin{aligned}
&a^{i-1} y_i \leq 1 && \text{for } 1 \leq i \leq k;\\
&y_1 + y_i \leq 1 && \text{for } 2 \leq i \leq k;\\
&y_1 \geq y_2 \geq \cdots \geq y_k \geq 0.
\end{aligned}$$

Set $y_1 = (a-1)/a$ and $y_i = 1/a^{i-1}$ for $2 \leq i \leq k$, which satisfies the above constraints. The value of the objective function of the LP for the point x^* is

$$\begin{aligned}
\mathrm{LP}(x^*) &= \sum_{\substack{e \in B\\ 1 \leq i \leq k}} (c_i - c_{i-1}) x^*_{i,e}\\
&= a^{k-1} \left(\frac{a-1}{a} + \sum_{2 \leq i \leq k} (a_{i-1} - a_{i-2}) \frac{1}{a^{i-1}} \right) = k a^{k-1} - k a^{k-2}.
\end{aligned}$$

Therefore, the ratio $\mathrm{LP}(x^*)/(\mathrm{IP})$ tends to k as $a \to \infty$, which implies the claim. $\square$

To conclude this section, let us mention that we know of additional families of valid inequalities that cut the fractional point used in the above proof. We leave the study of those for future research.

Acknowledgments

We thank Martine Labbé and Gilles Savard for preliminary discussions concerning the Stackelberg minimum spanning tree problem. We also thank Martin Hoefer for his comments which led us to prove Proposition 1.

References

1. P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoret. Comput. Sci.*, 237(1-2):123–134, 2000.
2. E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter. A survey on networking games in telecommunications. *Computers and Operations Research*, 33(2):286–311, 2006.
3. J. Cardinal, M. Labbé, S. Langerman, and B. Palop. Pricing of geometric transportation networks. In *Proc. Canadian Conference on Computational Geometry (CCCG)*, pages 92–96, 2005.
4. R. Cole, Y. Dodis, and T. Roughgarden. Pricing network edges for heterogeneous selfish users. In *Proc. Symposium on Theory of Computing (STOC)*, pages 521–530, 2003.
5. K. Dhamdhere, R. Ravi, and M. Singh. On two-stage stochastic minimum spanning trees. In *Proc. Integer Programming and Combinatorial Optimization (IPCO)*, volume 3509 of *Lecture Notes in Computer Science*, pages 321–334. Springer, 2005.
6. D. Eppstein. Setting parameters by example. *SIAM Journal on Computing*, 32(3):643–653, 2003.
7. D. Fernández-Baca, G. Slutzki, and D. Eppstein. Using sparsification for parametric minimum spanning tree problems. *Nordic J. Computing*, 3(4):352–366, 1996.
8. D. Granot and G. Huberman. Minimum cost spanning tree games. *Mathematical Programming*, 21(1):1–18, 1981.
9. A. Grigoriev, S. van Hoesel, A. van der Kraaij, M. Uetz, and M. Bouhtou. Pricing network edges to cross a river. In *Proc. Workshop on Approximation and Online Algorithms (WAOA)*, number 3351 in Lecture Notes in Computer Science, pages 140–153, 2005.
10. J. D. Hartline and V. Koltun. Near-optimal pricing in near-linear time. In *Proc. Workshop on Algorithms and Data Structures (WADS)*, pages 422–431, 2005.
11. A. Karlin, D. Kempe, and T. Tamir. Beyond VCG: Frugality of truthful mechanisms. In *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 615–626, 2005.
12. M. Labbé, P. Marcotte, and G. Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management Science*, 44(12):1608–1622, 1998.
13. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. System Sci.*, 43(3):425–440, 1991.
14. S. Roch, G. Savard, and P. Marcotte. An approximation algorithm for Stackelberg network pricing. *Networks*, 46(1):57–67, 2005.
15. T. Roughgarden. Stackelberg scheduling strategies. *SIAM Journal on Computing*, 33(2):332–350, 2004.
16. C. Swamy. The effectiveness of Stackelberg strategies and tolls for network congestion games. In *Proc. Symposium on Discrete Algorithms (SODA)*, 2007. to appear.
17. S. van Hoesel. An overview of Stackelberg pricing in networks. Research Memoranda 042, Maastricht : METEOR, Maastricht Research School of Economics of Technology and Organization, 2006.
18. H. von Stackelberg. *Marktform und Gleichgewicht (Market and Equilibrium)*. Verlag von Julius Springer, Vienna, 1934.