The Stackelberg Minimum Spanning Tree Game*

Jean Cardinal[†] Erik D. Demaine[‡] Samuel Fiorini[§] Gwenaël Joret[¶]
Stefan Langerman[‖] Ilan Newman[**] Oren Weimann[††]

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 $\min\{k, 1+\ln b, 1+\ln W\}$, 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.

*A preliminary version of this article appeared in the Proceedings of the 10th Workshop on Algorithms and Data Structures (WADS 2007), see [7]. This work was partially supported by the *Actions de Recherche Concertées (ARC)* fund of the *Communauté française de Belgique*.

†Université Libre de Bruxelles, Département d'Informatique, c.p. 212, B-1050 Brussels, Belgium, jcardin@ulb.ac.be.

‡MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA, edemaine@mit.edu.

§Université Libre de Bruxelles, Département de Mathématique, c.p. 216, B-1050 Brussels, Belgium, sfiorini@ulb.ac.be.

¶Université Libre de Bruxelles, Département d'Informatique, c.p. 212, B-1050 Brussels, Belgium, gjoret@ulb.ac.be. G. Joret is a Postdoctoral Researcher of the Fonds National de la Recherche Scientifique (F.R.S.–FNRS).

‖Université Libre de Bruxelles, Département d'Informatique, c.p. 212, B-1050 Brussels, Belgium, slanger@ulb.ac.be. S. Langerman is a Research Associate of the Fonds National de la Recherche Scientifique (F.R.S.–FNRS).

**Department of Computer Science, University of Haifa, Haifa 31905, Israel, ilan@cs.haifa.ac.il.

††MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA, oweimann@mit.edu.

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 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 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 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[1] $G = (V, E)$ 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

[1] All graphs in this paper are finite and may have loops and multiple edges.

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 one wants to price the edges in B and 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 [15], and by Bouhtou, Grigoriev, van Hoesel, van der Kraaij, Spieksma, and Uetz [4]: 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 [13]. This generalization was motivated by the problem of setting tolls on highway networks. Note that the StackMST problem is only interesting in the single-customer case, since otherwise all customers purchase the same tree. Cardinal, Labbé, Langerman, and Palop [8] give a geometric version of the shortest path problem.

Recently, part of the results of the current paper have been generalized to other problems by Briest, Hoefer and Krysta [5]. They also exhibit a polynomial-time algorithm for a special case of a Stackelberg vertex cover problem, in which the follower's problem is to find a minimum vertex cover in a bipartite graph.

Other pricing problems have been studied, in which the goal is to find the best prices for a set of items, after bidders have announced their preferences in the form of subset valuations. *Envy-free* pricing, in particular, can be viewed as a simple Stackelberg game. APX-hardness and approximability of such problems have been established by Hartline and Koltum [12], and by Guruswami, Hartline, Karlin, Kempe, Kenyon, and McSherry [11]. Balcan and Blum [2] gave improved approximation results. Approximability within a logarithmic factor has also been recently established for more general cases by Balcan, Blum and Mansour [3]. The case in which items are edges of a graph has been studied by Grigoriev, van Loon, Sitters and Uetz [10], and Briest and Krysta [6]. A semi-logarithmic inapproximability result for a special case of the unlimited supply pricing problem has been given by Demaine, Feige, Hajiaghayi, and Salavatipour [9].

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, and, to our knowledge, the first APX-hardness proof for a Stackelberg pricing game with a single customer. 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 $\min\{k, 1 + \ln b, 1 + \ln(c_k/c_1)\}$, 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.

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 for optimal price functions, 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 use the notation $\mathcal{C}(B', e)$ for the set of cycles of $G = (V, R \cup B')$ that include the edge e, where B' is an acyclic subset of blue edges and $e \in B'$. (Notice that $\mathcal{C}(B', e)$ is nonempty because (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*

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

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'.

Proof. The first part of the lemma is straightforward. Indeed, if (3) fails for some edge $e \in F$, then there exists a red edge e' with $c(e') < p(e)$ that links the two components of $T - e$, and so T cannot be an MST. We now turn to the second part of the lemma. First note that $E(T') \cap B$ is clearly contained in F because no MST can use any edge with an infinite price. By contradiction, suppose there is some edge e in F that is not used by T' and let e' be a red edge with maximum cost on the unique cycle of $T' + e$. Because the price function p we have chosen satisfies (3) (with equality), the weight of edge e is at most the weight of e', and thus T' is not an MST because of our assumption that blue edges have priority over the red edges of the same weight. □

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 $c(e) < p(f') \leq p(f)$.*

Proof. The inequality $c(e) < p(f)$ follows from the optimality of T and from our assumption on the priority of blue edges versus red edges of the same weight. If all blue edges f' distinct from f in C satisfied $p(f') \leq c(e)$ or $p(f) < p(f')$ then by decreasing the price of f by some amount we would be able to find a new price function p' such that $T' = T - e' + f$ is an MST with respect to p', where e' is some red edge on C. This contradicts the optimality of p because the revenue of T' is bigger than that of T. □

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

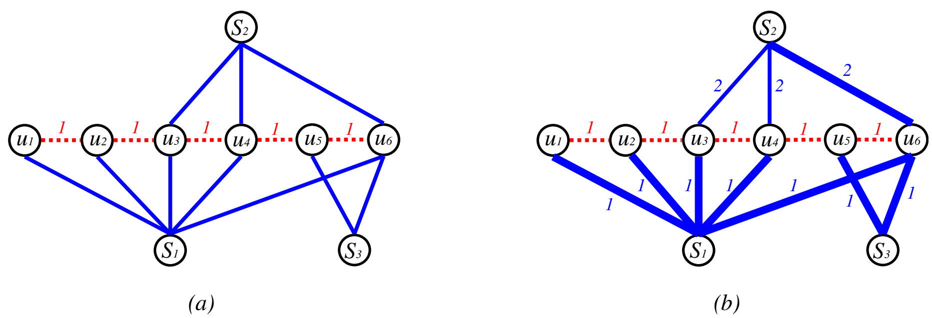

Figure 1: (a) The graph G constructed for $n = 6$, $m = 3$ with $S_1 = \{u_1, u_2, u_3, u_4, u_6\}$, $S_2 = \{u_3, u_4, u_6\}$ and $S_3 = \{u_5, u_6\}$. The red edges of cost 2 are omitted for clarity. The red edges of cost 1 are dashed, and the blue edges are solid. (b) An optimal price function p on 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 $i = 1, \ldots, n$, 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. To see this, observe that u_i is adjacent to a vertex S_j that is not a leaf, thus S_j has a neighbor u_k, and the red edges in the cycle $S_j, u_1, \ldots, u_k, S_j$ all have cost 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 $u_iS_j, u_{i+\ell}S_j$ in T then none of them could have a price of 2 because of the cycle $S_ju_iu_{i+1} \ldots u_{i+\ell}S_j$. 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 $f = u_iS_j$ 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 $c(e) < p(f') \le p(f)$. 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. □

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 $e \in R$.*

Proof. We will show that, for any $\varepsilon > 0$, a $(1-\varepsilon)$-approximation for StackMST implies a $(1 + 8\varepsilon)$-approximation for VertexCover in graphs of maximum degree at most 3. The claim will then follows from the APX-hardness of the latter problem [1, 14].

Let H denote any given graph with maximum degree at most 3. We can assume that H is connected because otherwise we process each connected component separately. Moreover, we can assume that H has at least as many edges as vertices because VertexCover can be solved exactly in polynomial time if H is a tree.

Clearly, the VertexCover instance we consider is equivalent to a SetCover instance with $|V(H)|$ sets and $|E(H)|$ elements in the ground set. Let $(\mathcal{U}, \mathcal{S})$ be the SetCover instance obtained from the latter one by adding a new dummy element d in the ground set, and adding d to every subset of the instance. Hence, we have $n = |\mathcal{U}| = |E(H)| + 1$ and $m = |\mathcal{S}| = |V(H)|$. Any vertex cover of H yields a set cover of $(\mathcal{U}, \mathcal{S})$ with the same size, and vice-versa. Thus the reduction used in the proof of Theorem 1 provides a way to convert in polynomial time a vertex cover of size s into a feasible solution of the StackMST instance corresponding to $(\mathcal{U}, \mathcal{S})$ with revenue $n + 2m - s - 1$, and vice-versa. In particular, we have $\mathrm{OPT} = n + 2m - \mathrm{OPT}_{\mathrm{VC}} - 1$, where OPT and $\mathrm{OPT}_{\mathrm{VC}}$ denote the value of the optimum for the StackMST and VertexCover instances, respectively.

Now consider the vertex cover found by running the $(1-\varepsilon)$-approximation algorithm on the StackMST instance and then converting the result into a vertex cover of H. Denoting by s its size and letting $r = n + 2m - s - 1$, we obtain:

$$\begin{aligned} s = n + 2m - r - 1 &\leq n + 2m - (1-\varepsilon)\,\mathrm{OPT} - 1 \\ &= n + 2m - (1-\varepsilon)\,(n + 2m - \mathrm{OPT}_{\mathrm{VC}} - 1) - 1 \\ &= \varepsilon\,(n - 1 + 2m) + (1-\varepsilon)\,\mathrm{OPT}_{\mathrm{VC}} \\ &\leq \varepsilon\,(3\,\mathrm{OPT}_{\mathrm{VC}} + 6\,\mathrm{OPT}_{\mathrm{VC}}) + (1-\varepsilon)\,\mathrm{OPT}_{\mathrm{VC}} \\ &= (1 + 8\varepsilon)\,\mathrm{OPT}_{\mathrm{VC}}. \end{aligned}$$

Above we have used the fact that $n - 1 = |E(H)| \geq |V(H)| = m$ and that $\mathrm{OPT}_{\mathrm{VC}} \geq |E(H)|/3 = (n-1)/3$ because H has maximum degree at most 3. ☐

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 red costs are positive (otherwise we contract all red edges of cost 0). Recall that the Best-out-of-k algorithm is as follows. For each i between 1 and k, set $p(e) = c_i$ for all blue edges $e \in B$ and compute an MST T_i. Then pick i such that the revenue of T_i is maximum and output the corresponding feasible solution. In this section, we analyze the approximation ratio ensured by this algorithm.

Theorem 3. *Best-out-of-k is a* $\min\{k, 1 + \ln b, 1 + \ln W\}$*-approximation algorithm, where* b *denotes the number of blue edges, and* $W = c_k/c_1$ *is the maximum ratio between red costs.*

Proof. We let T^* be an MST of the graph with an optimal price function, and let n_i denote the number of blue edges of price c_i in T^*. We also define N_i as the set of blue edges of price at least i in T^*, that is, $N_i = \sum_{j=i}^{k} n_j$.

Let G_i be the graph in which all blue edges have price c_i. We first prove the following claim: any MST T_i of G_i contains at least N_i blue edges. For $S \subseteq E$, let $r(S)$ denote the maximum cardinality of an acyclic subset of S (that is, the rank function of the cycle matroid of G). We also let R_i be the set of red edges with cost at most c_i, and B_i be the set of blue edges with price at most c_i in G^*. We have

$$N_i \leq r(R_{i-1} \cup B) - r(R_{i-1} \cup B_{i-1}) \leq r(R_{i-1} \cup B) - r(R_{i-1}).$$

If we consider an execution of Kruskal's algorithm on G_i, the latter expression is exactly the number of blue edges that are added to T_i. This proves the claim.

Using this claim, we can bound the revenue q:

$$q \geq \max_{i=1}^{k} N_i \cdot c_i.$$

We also know that $OPT = \sum_{i=1}^{k} n_i \cdot c_i$.

Since $n_i \leq N_i$, we have

$$OPT = \sum_{i=1}^{k} n_i \cdot c_i \leq \sum_{i=1}^{k} N_i \cdot c_i \leq k \cdot q,$$

proving the first approximation factor.

Also, we have (letting $N_{k+1} = 0$):

$$\begin{aligned}
OPT &= \sum_{i=1}^{k} n_i \cdot c_i \\
&= \sum_{i=1}^{k} N_i \cdot c_i \cdot \frac{n_i}{N_i} \\
&= \sum_{i=1}^{k} N_i \cdot c_i \cdot \frac{N_i - N_{i+1}}{N_i} \\
&\leq (\max_{i=1}^{k} N_i \cdot c_i) \cdot \sum_{i=1}^{k} \frac{N_i - N_{i+1}}{N_i} \\
&\leq q \cdot \sum_{i=1}^{k} \frac{N_i - N_{i+1}}{N_i},
\end{aligned}$$

and

$$\sum_{i=1}^{k} \frac{N_i - N_{i+1}}{N_i} \leq 1 + \int_{N_k}^{N_1} \frac{dt}{t} \leq 1 + \ln \frac{N_1}{N_k} \leq 1 + \ln b,$$

which proves the second approximation factor.

Finally, we also have the following (letting $c_0 = 0$):

$$\begin{aligned} OPT &= \sum_{i=1}^{k} n_i \cdot c_i \\ &= \sum_{i=1}^{k} n_i \sum_{j=1}^{i} (c_j - c_{j-1}) \\ &= \sum_{j=1}^{k} N_j \cdot (c_j - c_{j-1}) \\ &\leq q \cdot \sum_{j=1}^{k} \frac{c_j - c_{j-1}}{c_j}, \end{aligned}$$

and

$$\sum_{j=1}^{k} \frac{c_j - c_{j-1}}{c_j} \leq 1 + \ln W,$$

establishing the third approximation factor. □

The three approximation factors are tight for the following examples. Consider a graph with $k+1$ vertices $v_1, v_2, \ldots, v_{k+1}$, in which the red edges are of the form $v_i v_{i+1}$, and there is a blue edge parallel to every red edge. The cost of the red edge $v_i v_{i+1}$ is $1/i$. The optimal solution involves setting a price of $1/i$ for every blue edge $v_i v_{i+1}$, yielding a revenue of $\sum_{i=1}^{k} 1/i$. On the other hand, the Best-out-of-k algorithm sets the price of every blue edge to $1/i$ for some i, always yielding a revenue of 1. This proves that the ratios $1 + \ln b$ and $1 + \ln W$ are tight.

The factor k can be proven tight as well by considering a similar example. The graph is composed of $1 + \sum_{i=1}^{k} a^{i-1}$ vertices for some large integer a. The red edges form a path connecting these vertices using a^{k-i} edges of cost $c_i = a^{i-1}$ for every i between 1 and k. Every red edge is doubled by a blue edge. The optimal solution again involves setting the prices of the blue edges equal to that of the parallel red edge, yielding a revenue of $k \cdot a^{k-1}$. The Best-out-of-k algorithm setting the prices to c_i yields an MST containing $\sum_{j=i}^{k} a^{k-j}$ blue edges, with a revenue of

$$a^{i-1} \cdot \sum_{j=i}^{k} a^{k-j} = a^{i-1} \cdot \frac{a^{k-i+1} - 1}{a - 1} < a^{k-1} \cdot \frac{a}{a-1}.$$

The ratio between the two revenues tend to k as a tends to infinity.

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 the same approximation for this more general case.

5 Linear Programming Relaxation

In this section, we give an integer programming formulation for the problem and study its linear programming relaxation. All red costs c_i are assumed to be positive throughout the section. 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:

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

$$\text{s.t.} \quad \sum_{e \in \delta_B(C_1:C_2:\cdots:C_t)} x_{j,e} \le t - 1 \qquad \forall j \in \{1, 2, \ldots, k\}, \; \forall C_1, ..., C_t \text{ components of } (V, R_{j-1}); \tag{4}$$

$$\sum_{e \in P \cap B} x_{1,e} + x_{j,f} \le |P \cap B| \qquad \forall f = ab \in B, \forall j \in \{2, 3, \ldots, k\}, \; \forall P \; ab\text{-path in } (B \cup R_{j-1}) - f; \tag{5}$$

$$x_{1,e} \ge x_{2,e} \ge \cdots \ge x_{k,e} \ge 0 \qquad \forall e \in B; \tag{6}$$

$$x_{j,e} \in \{0, 1\} \qquad \forall j \in \{1, 2, \ldots, k\}, \forall e \in B. \tag{7}$$

Let us first give some intuition on this integer program. Consider a minimum spanning tree T with respect to a feasible solution p, let F be the set of blue edges appearing in T, and let $F_j = \{e \in F : p(e) \ge c_j\}$. Then F $(= F_1)$ must obviously be a forest. Also, F_j ($j \in \{2, \ldots, k\}$) must be a forest in the graph where every component of (V, R_{j-1}) has been contracted, since otherwise we could swap in T some edge of F_j with an edge in R_{j-1}. This is encoded by constraints (4). Similarly, if a cycle C of the graph is such that every red edge in C has cost at most c_{j-1} and some blue edge f of C appears in T with a price at least c_j, then there must be another blue edge of C which is not included in T. This is ensured by constraints (5).

Proposition 1. *The integer program above is a formulation of* StackMST.

Proof. Consider a feasible solution x of the integer program (IP) and let $F = \{e \in B : x_{1,e} = 1\}$. Inequality (4) ensures that F is a forest. For $e \in F$, let $p(e) = c_j$ if j is the last index for which $x_{j,e} = 1$ and, for $e \in B - F$, let $p(e) = \infty$. Now consider a minimum spanning tree T with respect to p. We claim $E(T) \cap B = F$ and that the revenue of T is exactly the objective value for x.

It suffices to prove that all edges of F belong to T. All edges $e \in F$ of price c_1 are necessarily in T. Assume that all edges $e \in F$ of price less than c_j are in T, for some $j \geq 2$. We show that this holds too for edges of price c_j. Consider some edge f with $p(f) = c_j$. Suppose that f is not in T. This means that there exists a cycle in G consisting of blue edges of price at most c_j and of red edges of price at most c_{j-1}. But then (5) is violated, a contradiction. So the claim holds.

Conversely, consider any optimal solution to the StackMST problem with price function $p(\cdot)$ and a corresponding MST T. Let $F = E(T) \cap B$. We define a vector x as follows: for $e \in B$, $x_{i,e} = 1$ if $e \in F$ and $p(e) \geq c_i$, otherwise $x_{i,e} = 0$. It is easily checked that the revenue given by p equals the objective function of the IP for x. Moreover, constraints (4), (6) and (7) are clearly satisfied by x. Finally, note that if x violates (5) for some $e \in F$, then e also violates the min-max formula given in Lemma 2. This completes the proof. □

The rest of this section is devoted to the LP relaxation of the above IP, obtained by dropping constraint (7). We show that the LP is tractable and that its integrality gap matches essentially 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 $\mathcal{I}$ is defined as the supremum of the ratio (LP)/(IP) over all instances in $\mathcal{I}$.)

Proposition 2. *The LP can be separated in polynomial time.*

Proof. For fixed j, (4) can be separated in polynomial time using standard techniques for the forest polytope, as described e.g. in Schrijver [16, pp. 880–881]. Inequality (5) can be rewritten as

$$\sum_{e \in P \cap B} (1 - x_{1,e}) \geq x_{j,f}.$$

Thus, for each fixed j and $f = ab$, (5) can be separated by finding a shortest ab-path in the graph $(V, (B \cup R_{j-1}) - f)$ where every red edge has weight 0 and every blue edge e has weight $1 - x_{1,e}$. Finally, (6) can obviously be separated in polynomial time. □

We first bound the integrality gap from above:

Proposition 3. *We have* (LP) $\leq \min\{k, 1 + \ln b, 1 + \ln W\} \cdot$ (IP)*, where b denotes the number of blue edges, and $W = c_k/c_1$ is the maximum ratio between red costs.*

Proof. Let x be any feasible vector for the LP. The value of the objective function for x is thus

$$\sum_{\substack{e \in B \\ 1 \leq i \leq k}} (c_i - c_{i-1}) x_{i,e}.$$

Let $i \in \{1, \dots, k\}$, let $C^1, \dots, C^\ell$ be components of the graph $(V, R_{i-1} \cup B)$, and denote by $C^j_1, \dots, C^j_{\ell_j}$ the components of the subgraph of (V, R_{i-1}) induced by C^j. For every $j \in \{1, \dots, \ell\}$, we have

$$\sum_{e \in B[C^j_1 \cup \dots \cup C^j_{\ell_j}]} x_{i,e} = \sum_{e \in \delta_B(C^j_1 : C^j_2 : \dots : C^j_{\ell_j})} x_{i,e}.$$

(Here, for $S \subseteq V$, the notation $B[S]$ means the set of blue edges with both endpoints in S.) Indeed, this holds trivially if $i = 1$, since then each C^j_p is a vertex of C^j. For $i \geq 2$, for any blue edge $f = ab$ that is internal to a component C^j_p of C^j (that is, $f \in B[C^j_p]$), there exists an ab-path consisting of edges of R_{i-1}, and so (5) enforces that $x_{i,f} \leq 0$.

Also, constraints (4) imply

$$\sum_{e \in \delta_B(C^j_1 : C^j_2 : \dots : C^j_{\ell_j})} x_{i,e} \leq \ell_j - 1,$$

for every $j \in \{1, \dots, \ell\}$. We thus obtain

$$\sum_{e \in B} x_{i,e} = \sum_{j=1}^{\ell} \sum_{e \in \delta_B(C^j_1 : C^j_2 : \dots : C^j_{\ell_j})} x_{i,e} \leq \sum_{j=1}^{\ell} (\ell_j - 1) = r(R_{i-1} \cup B) - r(R_{i-1}).$$

The number of blue edges in the i-th MST computed by Best-out-of-k being exactly $r(R_{i-1} \cup B) - r(R_{i-1}) =: A_i$, it then follows

$$\sum_{\substack{e \in B \\ 1 \leq i \leq k}} (c_i - c_{i-1}) x_{i,e} \leq \sum_{i=1}^{k} (c_i - c_{i-1}) A_i.$$

Letting $q = \max_{i=1}^{k} A_i \cdot c_i$ denote the revenue given by the Best-out-of-k algorithm, we deduce

$$\sum_{i=1}^{k} (c_i - c_{i-1}) A_i = \sum_{i=1}^{k} \frac{c_i - c_{i-1}}{c_i} A_i \cdot c_i \leq q \cdot \sum_{i=1}^{k} \frac{c_i - c_{i-1}}{c_i},$$

and, letting $A_{k+1} = 0$,

$$\sum_{i=1}^{k} (c_i - c_{i-1}) A_i = \sum_{i=1}^{k} c_i (A_i - A_{i+1}) = \sum_{i=1}^{k} A_i \cdot c_i \frac{A_i - A_{i+1}}{A_i} \leq q \cdot \sum_{i=1}^{k} \frac{A_i - A_{i+1}}{A_i}.$$

As in the proof of Theorem 3, we have

$$\sum_{i=1}^{k} \frac{c_i - c_{i-1}}{c_i} \leq \min\{k, 1 + \ln W\}$$

and

$$\sum_{i=1}^{k} \frac{A_i - A_{i+1}}{A_i} \leq 1 + \ln b.$$

Therefore,

$$\begin{aligned}\sum_{\substack{e \in B \\ 1 \leq i \leq k}} (c_i - c_{i-1}) x_{i,e} &\leq \min\{k, 1 + \ln b, 1 + \ln W\} \cdot q \\ &\leq \min\{k, 1 + \ln b, 1 + \ln W\} \cdot (\text{IP}),\end{aligned}$$

as claimed. □

Proposition 4. *The integrality gap of the LP is*

- k *on instances with* k *distinct costs;*
- $\Theta(\ln W)$ *on instances with maximum ratio between red costs* W*, and*
- $\Theta(\ln b)$ *on instances with* b *blue edges.*

Proof. We already know from Proposition 3 that the integrality gap of the LP is at most $\min\{k, 1+\ln b, 1+\ln W\}$. We first by prove that the integrality gap is at least k on instances with k distinct costs. To this aim, we define an instance of StackMST as follows: Choose an integer $a \geq 2$ and let the vertex set of the graph be $V = \{0, 1, 2, \ldots, a^{k-1}\}$. The graph has a^{k-1} blue edges, linking vertex 0 to every other vertex. The ith red cost is $c_i = a^{i-1}$. For $i \in \{1, 2, \ldots, k-1\}$, the subgraph spanned by the red edges with cost c_i is a disjoint union of a^{k-i-1} cliques, each of cardinality a^i; the vertex sets of these cliques are $\{1, \ldots, a^i\}, \{a^i + 1, \ldots, 2a^i\}, \ldots, \{a^{k-1} - a^i + 1, \ldots, a^{k-1}\}$. Finally, there is a unique red edge with cost c_k, linking vertex 0 to vertex 1.

Consider an optimal solution of the StackMST problem for the instance defined above, and let T be a corresponding MST. Consider any blue edge e in T, of price c_i, and let C_e be the unique component of $(V - \{0\}, R_{i-1})$ that contains an endpoint of e. No other blue edge of T has an endpoint in C_e, because otherwise one could replace the edge e in T with an appropriate red edge of R_{i-1} and obtain a new spanning tree with weight strictly less than that of T, a contradiction. 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

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

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

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 $a \to \infty$.

Now, the same construction can be used to show that the integrality gap is $\Omega(\ln W)$ and $\Omega(\ln b)$ on instances with $c_k/c_1 = W$ and b blue edges, respectively. We explain it in the case where the number of blue edges is fixed to some value b, the case where the ratio c_k/c_1 is fixed is done similarly.

Take an instance as above, with $a = 2$ and k being the greatest integer such that $2^{k-1} \leq b$. Choose an arbitrary blue edge and add $b - 2^{k-1}$ parallel blue edges to it (so that the number of blue edges is exactly b). These extra blue edges have clearly no influence on the value of (IP) and $\mathrm{LP}(x^*)$ (where x^* is defined as before). Using $b < 2^k$, we deduce

$$\frac{\mathrm{LP}(x^*)}{(\mathrm{IP})} = \frac{k 2^{k-1} - k 2^{k-2}}{2^{k-1}} = \frac{k}{2} > \frac{\log_2 b}{2},$$

and thus that the integrality gap is $\Omega(\ln b)$, as claimed. □

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 this problem, Martin Hoefer for his comments which led us to refine our approximability result. We are also most grateful to the second anonymous referee for providing us with a much shorter proof of Theorem 3, and for her or his many insightful remarks which led to an improved version of the paper.

References

[1] P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoret. Comput. Sci.*, 237(1-2):123–134, 2000.

[2] M.-F. Balcan and A. Blum. Approximation algorithms and online mechanisms for item pricing. In *Proc. ACM Conference on Electronic Commerce (EC)*, 2006.

[3] M.-F. Balcan, A. Blum, and Y. Manshour. Item pricing for revenue maximization. In *Proc. ACM Conference on Electronic Commerce (EC)*, 2008.

[4] M. Bouhtou, A. Grigoriev, S. van Hoesel, A. F. van der Kraaij, F. C. R. Spieksma, and M. Uetz. Pricing bridges to cross a river. *Naval Res. Logist.*, 54(4):411–420, 2007.

[5] P. Briest, M. Hoefer, and P. Krysta. Stackelberg network pricing games. In *Proc. 25th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 133–142, 2008.

[6] P. Briest and P. Krysta. Single-minded unlimited supply pricing on sparse instances. In *Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1093–1102, 2006.

[7] J. Cardinal, E. D. Demaine, S. Fiorini, G. Joret, S. Langerman, I. Newman, and O. Weimann. The stackelberg minimum spanning tree game. In *Proc. 10th international Workshop on Algorithms and Data Structures (WADS)*, volume 4619 of *Lecture Notes in Computer Science*, pages 64–76. Springer-Verlag, 2007.

[8] 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.

[9] E. D. Demaine, U. Feige, M. Hajiaghayi, and M. R. Salavatipour. Combination can be hard: Approximability of the unique coverage problem. *SIAM Journal on Computing*, to appear.

[10] A. Grigoriev, J. van Loon, R. Sitters, and M. Uetz. How to sell a graph: Guidelines for graph retailers. In *Proc. Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 4271 of *Lecture Notes in Computer Science*, pages 125–136. Springer-Verlag, 2006.

[11] V. Guruswami, J. Hartline, A. Karlin, D. Kempe, C. Kenyon, and F. McSherry. On profit-maximizing envy-free pricing. In *Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1164–1173, 2005.

[12] J. D. Hartline and V. Koltun. Near-optimal pricing in near-linear time. In *Proc. Workshop on Algorithms and Data Structures (WADS)*, volume 3608 of *Lecture Notes in Computer Science*, pages 422–431. Springer-Verlag, 2005.

[13] 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.

[14] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. System Sci.*, 43(3):425–440, 1991.

[15] S. Roch, G. Savard, and P. Marcotte. An approximation algorithm for Stackelberg network pricing. *Networks*, 46(1):57–67, 2005.

[16] A. Schrijver. *Combinatorial optimization. Polyhedra and efficiency. Vol. B*, volume 24 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2003. Matroids, trees, stable sets, Chapters 39–69.

[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.