The Stackelberg Minimum Spanning Tree Game on Planar and Bounded-Treewidth Graphs

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

Abstract

The Stackelberg Minimum Spanning Tree Game is a two-level combinatorial pricing problem introduced at WADS'07. 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 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.

We study this problem in the cases of planar and bounded-treewidth graphs. We show that the problem is NP-hard on planar graphs but can be solved in polynomial time on graphs of bounded treewidth.

1 Introduction

A young startup company has just acquired a collection of point-to-point tubes between various sites on the Interweb. The company's goal is to sell the use of these tubes to a particularly stingy client, who will buy a minimum-cost spanning tree of the network. Unfortunately, the company has a direct competitor: the government sells the use of a different collection of point-to-point tubes at publicly known prices. Our goal is to set the company's tube prices to maximize the company's income, given the government's prices and the knowledge that the client will buy a minimum spanning tree made from any combination of company and government tubes. Naturally, if we set the prices too high, the client will rather buy the government's tubes, while if we set the prices too low, we unnecessarily reduce the company's income.

*Université Libre de Bruxelles (ULB), Département d'Informatique, CP 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 (ULB), Département de Mathématique, CP 216, B-1050 Brussels, Belgium, sfiorini@ulb.ac.be.

§Université Libre de Bruxelles (ULB), Département d'Informatique, CP 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).

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

‖Weizmann Institute of Science, Faculty of Mathematics and Computer Science, POB 26, Rehovot 76100, Israel, oren.weimann@weizmann.ac.il.

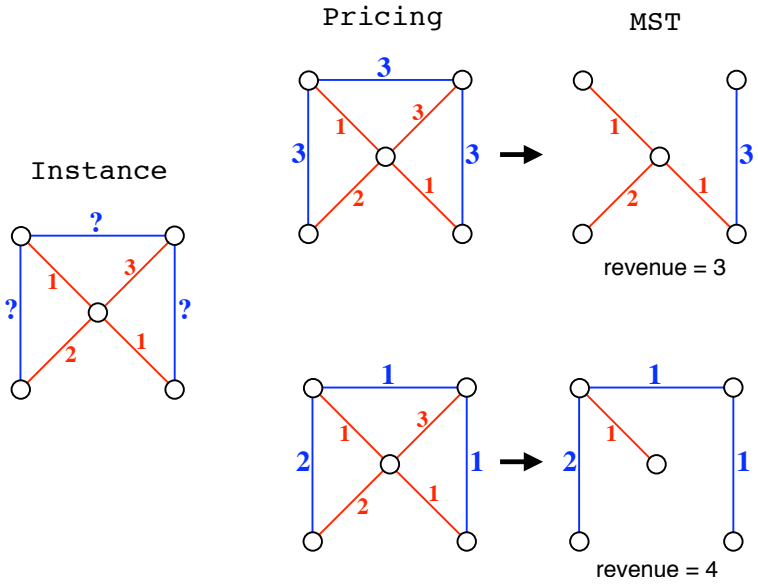


Figure 1: A sample instance of the StackMST problem. The goal is to assign prices to the blue edges to maximize the total price of the blue edges purchased in a minimum spanning tree.

This problem is called the *Stackelberg Minimum Spanning Tree Game* [CDF+07], and is an example in the growing family of algorithmic game-theoretic problems about combinatorial optimization in graphs [BHK08, GvLSU09, LMS98, RSM05, vH06, BGPW08]. More formally, we are given an undirected graph G (possibly with parallel edges, but no loops), whose edge set $E(G)$ is partitioned into a *red edge set* $R(G)$ and a *blue edge set* $B(G)$. We are also given a cost function $c : R(G) \to \mathbb{R}^+$ assigning a positive cost to each red edge. The StackMST problem is to assign a price $p(e)$ to each blue edge e, resulting in a weighted graph $(G, c \cup p)$, to maximize the total price of blue edges in a minimum spanning tree. We assume that, if there is more than one minimum spanning tree, we obtain the maximum possible income. (Otherwise, we could decrease the prices slightly and get arbitrarily close to the same income.) Figure 1 shows an example.

This problem is thus a two-player two-level optimization problem, in which the leader (the company) chooses a strategy (a price assignment), taking into account the strategy of the follower (the client), which is determined by a second-level optimization problem (the minimum spanning tree problem). Such a game is known as a *Stackelberg game* in economics [vS34].

The complexity and approximability of the StackMST problem has been studied in a previous paper [CDF+07], which shows the following results. The problem is APX-hard, but can be approximated within a logarithmic factor. Constant-factor approximation exist for the special cases in which the given costs are bounded or take a bounded number of distinct values. Finally, an integer programming formulation has an integrality gap corresponding to the best known approximation factors.

Instead of restricting the edge weights, we can restrict the class of allowed graphs, with the hope of obtaining better approximation algorithms. One natural class of graphs is *planar graphs*, on which many important problems admit polynomial-time approximation schemes. Many of these results use Baker's technique [Bak94] or more modern variations [Kle05, Kle06, BKMK07, DHM07, DHK09],

which ultimately rely on the ability to efficiently solve the problem in graphs of bounded treewidth in polynomial time. Such algorithms generally use dynamic programming, using a textbook technique for well-behaved problems. In particular, the problem of checking a graph-theoretic property expressible in monadic second-order logic is fixed-parameter tractable with respect to the treewidth of the graph; see [Cou08] for a survey. However, few if any such dynamic programs have been developed for a two-level optimization problem such as STACKMST, and standard techniques do not seem to apply.

In this paper, we consider the STACKMST problem in these two graph classes: planar graphs and bounded-treewidth graphs. We prove in Section 2 that STACKMST remains NP-hard when restricted to planar graphs. We develop in Section 4 a polynomial-time dynamic programming algorithm for STACKMST in graphs of bounded treewidth. Along the way, we develop in Section 3 a dynamic programing algorithm for series-parallel graphs, or equivalently, biconnected graphs of treewidth at most 2, which are also planar.

To our knowledge, our algorithms are the first examples of a two-level pricing problem solved by dynamic programming on a graph decomposition tree. We believe that this result provides insight into the structure of the problem, and could be a stepping stone toward a polynomial-time approximation scheme for planar graphs. More generally, we believe that our techniques may be useful in the design of dynamic programming algorithms for other pricing problems in graphs, including pricing problems with many followers [BHK08, GvLSU09], and Stackelberg problems involving shortest paths [RSM05] or shortest path trees [BGPW08].

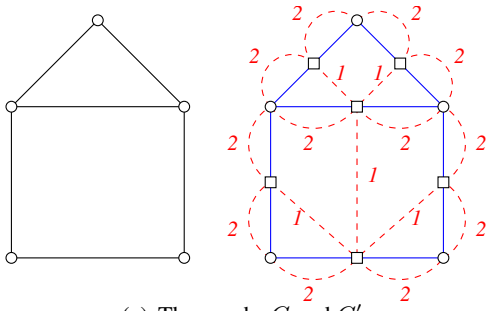
2 Planar Graphs

We consider the STACKMST problem on planar graphs. We strengthen the hardness result given in [CDF$^+$07] by showing that the problem remains NP-hard in this special case. The reduction is from the *minimum connected vertex cover problem*, which is known to be NP-hard, even when restricted to planar graphs of maximum degree 4 (see Garey and Johnson [GJ79]). The minimum connected vertex cover problem consists of finding a minimum-size subset C of the vertices of a graph, such that every edge has at least one endpoint in C, and C induces a connected graph.

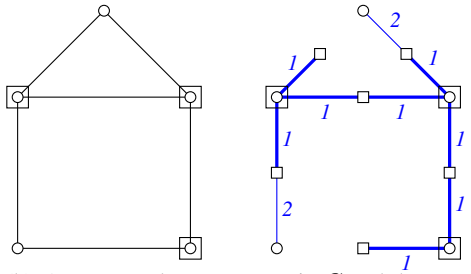
Theorem 1. *The* STACKMST *problem is NP-hard, even when restricted to planar graphs.*

Proof. Given a planar graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, we construct an instance of STACKMST with red costs in $\{1, 2\}$. Let $G' = (V', R \cup B)$ be the graph for this instance, with (R, B) a bipartition of the edge set. We first let $V' = V \cup E$. The set of blue edges B is the set $\{ve : e \in E, v \in e\}$. Thus the blue subgraph is the vertex-edge incidence graph of G, which is clearly planar. Given a planar embedding of the blue subgraph, we connect all vertices $e \in E$ of G' by a tree, all edges of which are red and have cost 1. The graph can be kept planar by letting those red edges be nonintersecting chords of the faces of the embedding. Finally, we double all blue edges by red edges of cost 2. The whole construction is illustrated in figure 2(a).

Let t be a positive integer. We show that the revenue for an optimal price function for G' is at least $m + 2n - t - 1$ if and only if there exists a connected vertex cover of G of size at most t.



(a) The graphs G and G'.



(b) A connected vertex cover in G and the corresponding price function in G'.

Figure 2: Illustration of the proof of Theorem 1.

($\Leftarrow$) We first suppose that there exists such a connected vertex cover $C \subseteq V$, and show how to construct a price function yielding the given revenue.

From the set C, we can construct a tree made of blue edges that spans all vertices $e \in E$ of G'. The set of vertices of this tree is $C \cup E$, and its edges are of the form $ue \in E'$, with $u \in C$ and $e \in E$ (see figure 2(b)). This tree has $t + m - 1$ blue edges, to which we assign price 1. Now we have to connect the remaining $n - t$ vertices belonging to V. Since the only red edges incident to these vertices have cost 2, we can use $n - t$ blue edges of price 2 to include these vertices in the minimum spanning tree. The price of the other blue edges is set to ∞. The revenue for this price function is exactly $(t + m - 1) + 2(n - t) = m + 2n - t - 1$.

($\Rightarrow$) Now suppose that we have a price function yielding revenue at least $m + 2n - t - 1$. We can assume (see [CDF+07]) that all the prices belong to the set $\{1, 2, \infty\}$. We also assume that the price function is optimal and minimizes the number of red edges in the resulting spanning tree T.

First, we observe that T does not contain any red edge. By contradiction, if T contains a red edge of cost 2, then this edge can be replaced by the parallel blue edge. On the other hand, if T contains a red edge f of cost 1, we consider the cut defined by removing f from T. In the face used to define f, there exists a blue edge having its endpoints across the cut and does not belong to T. So we can use this blue edge, with a price equal to 1, to reconnect the tree.

Now let us consider the blue edges of price 1 in T. We claim that the graph H induced by these edges contains all vertices $e \in E$ of G' and is connected.

Clearly, all vertices $e \in E$ of G' are incident to a blue edge of price 1, otherwise it can be reconnected to T with a red edge of cost 1, and T is not minimum. Thus $E \subseteq V(H)$, where $V(H)$ is the vertex set of H. Letting $C := V(H) \cap V$, we conclude that C is a vertex cover of the original graph G.

Now we show that H is connected. Suppose otherwise; then there exist two vertices of G' in E that are connected by a red edge of cost 1, and belonging to two different connected components H_1 and H_2 of H. Consider the (blue) edge that connects H_1 and H_2 in T. This edge cannot have price 2 in T, since H_1 and H_2 are connected by a red edge of cost 1. Hence the blue edge has price 1 and belongs to H. Therefore H is connected and C is a connected vertex cover of G.

Finally the remaining vertices $V - C$ of G' must be leaves of T, since otherwise they belong to a cycle containing a red edge of cost 1. The total cost of T is therefore $(m + |C| - 1) + 2(n - |C|) =$

$m + 2n - |C| - 1$. Since we know this is at least $m + 2n - t - 1$, we conclude that $|C| \leq t$. □

3 Series-Parallel Graphs

We now describe a polynomial-time dynamic programming algorithm for solving the StackMST problem on series-parallel graphs. These graphs are the biconnected graphs excluding K_4 as a minor, or equivalently, biconnected graphs with treewidth at most 2.

We use the following inductive definition of (connected) series-parallel graphs. Consider a connected graph G with two distinguished vertices s and t. The graph (G, s, t) is a *series-parallel* graph if either G is a single edge (s, t), or G is a *series* or *parallel* composition of two series-parallel graphs (G_1, s_1, t_1) and (G_2, s_2, t_2). The series composition of G_1 and G_2 is formed by setting $s = s_1, t = t_2$ and identifying $t_1 = s_2$; the parallel composition is formed by identifying $s = s_1 = s_2$ and $t = t_1 = t_2$.

Theorem 2. *The* StackMST *problem can be solved in $O(m^4)$ time on series-parallel graphs.*

3.1 Definitions

Let us fix an instance of StackMST, that is, a graph G with $E(G) = R(G) \cup B(G)$ endowed with a cost function $c : R(G) \to \mathbb{R}_+$. Denote by $c_1, c_2, \ldots, c_k$ the different values taken by c, in increasing order. Let also $c_0 := 0$.

For two distinct vertices $s, t \in V(G)$ of G and a subset $F \subseteq B(G)$ of blue edges, define $\mathcal{P}(G, F, s, t)$ as the set of st-paths in the graph $(V(G), R(G) \cup F)$. Let also $\widetilde{\mathcal{P}}(G, F, s, t)$ denote the subset of paths in $\mathcal{P}(G, F, s, t)$ that contain at least one red edge. A lemma of Cardinal *et al.* [CDF+07] can be restated as follows.

Lemma 1 ([CDF+07])**.** *Suppose that G contains a red spanning tree, and let $F \subseteq B(G)$ be an acyclic subset of blue edges. Then, the maximum revenue achievable by the leader, over solutions where the set of blue edges bought by the follower is exactly F, is obtained by setting the price of each edge $st \notin F$ to $+\infty$, and the price of each edge $st \in F$ to*

$$\min \left\{ \max_{e \in P \cap R(G)} c(e) \mid P \in \widetilde{\mathcal{P}}(G, F, s, t) \right\}.$$

This lemma states that if we know the set of blue edges that will eventually be bought, the price of a selected blue edge st is given by the minimum, over the paths from s to t, of the largest red cost on this path.

Motivated by this result, we introduce some more notations. For a subset $Z \subseteq E(G)$ of edges, we define $\mathrm{mc}(Z)$ as the maximum cost of a red edge in Z if $Z \cap R(G) \neq \varnothing$, as $c_0 = 0$ otherwise. (The two letters mc stand for "max cost".) We define $w(G, F, s, t)$ as

$$w(G, F, s, t) := \begin{cases} \min \{\mathrm{mc}(P) \mid P \in \mathcal{P}(G, F, s, t)\} & \text{if } \mathcal{P}(G, F, s, t) \neq \varnothing; \\ c_k & \text{otherwise.} \end{cases}$$

Similarly,

$$\widetilde{w}(G,F,s,t) := \begin{cases} \min\left\{\mathrm{mc}(P) \mid P \in \widetilde{\mathcal{P}}(G,F,s,t)\right\} & \text{if } \widetilde{\mathcal{P}}(G,F,s,t) \neq \varnothing; \\ c_k & \text{otherwise.} \end{cases}$$

Thus, the price assigned to the edge $st \in F$ in Lemma 1 is $\widetilde{w}(G,F,s,t)$. Also, we will consider graphs that do not necessarily contain a red spanning tree; this is why we need to treat the case where $\mathcal{P}(G,F,s,t)$ or $\widetilde{\mathcal{P}}(G,F,s,t)$ is empty in the above definitions.

In what follows, we let $[k] := \{0,1,\ldots,k\}$. Our dynamic programming solution for series-parallel graphs associates a value to each pair (H,q), where $q \in [k]^2$, and H is a graph appearing in the series-parallel decomposition of G.

A subset $F \subseteq B(G)$ of blue edges *realizes* $q = (i,j) \in [k]^2$ in (G,s,t) if F is acyclic and $w(G,F,s,t) = c_i$. Although this property does not depend on j, the formulation will appear to be convenient. Similarly, we say that q is *realizable* in (G,s,t) if there exists such a subset F.

For $j \in [k]$ and distinct vertices $s,t \in V(G)$, let G^+ denote the graph G with an additional red edge between s and t of cost c_j. We define

$$\mathrm{OPT}_{(i,j)}(G,s,t) := \max\left\{ \sum_{uv \in F} \widetilde{w}(G^+,F,u,v) \;\middle|\; F \subseteq B(G), F \text{ realizes } (i,j) \text{ in } (G,s,t) \right\},$$

if such a subset F exists, and set $\mathrm{OPT}_{(i,j)}(G,s,t) := -\infty$ otherwise.

Intuitively, we want to keep track of optimal acyclic subsets of blue edges for every graph G obtained during the construction of a series-parallel graph. The problem is, that the weights of the blue edges in the optimal solution might change as we compose graphs in the series-parallel decomposition. However, the weights of edges depend only on the maximum red costs, or *bottlenecks*, of the new st-paths that will be added to G. We can thus prepare $\mathrm{OPT}(G,s,t)$ for every possible set of bottlenecks. These bottlenecks are the values j in what precedes.

Note that by Lemma 1, if G has a red spanning tree, then the maximum revenue achievable by the leader on instance G equals

$$\max_{i \in [k]} \mathrm{OPT}_{(i,k)}(G,s,t).$$

This will be the result returned by the dynamic programming solution.

3.2 Series Compositions

Let $q = (i,j)$, $q_1 = (i_1,j_1)$, and $q_2 = (i_2,j_2)$, with $q, q_1, q_2 \in [k]^2$. We say that the pair (q_1,q_2) is *series-compatible* with q if

(S1) $\max\{i_1,i_2\} = i$;

(S2) $\max\{j,i_2\} = j_1$, and

(S3) $\max\{j,i_1\} = j_2$,

Notice that (q_1, q_2) is series-compatible with q if and only if (q_2, q_1) is.

This condition allows us to use the following recursion in our dynamic programming algorithm.

Lemma 2. *Suppose that (G, s, t) is a series composition of (G_1, s_1, t_1) and (G_2, s_2, t_2), and that $q \in [k]^2$ is realizable in (G, s, t). Then*

$$\mathrm{OPT}_q(G, s, t) = \max\left\{\mathrm{OPT}_{q_1}(G_1, s_1, t_1) + \mathrm{OPT}_{q_2}(G_2, s_2, t_2) \mid (q_1, q_2) \text{ is series-compatible with } q\right\}.$$

We now prove that the recursion is valid. We need the following lemmas. In what follows, (G, s, t) is a series composition of (G_1, s_1, t_1) and (G_2, s_2, t_2); $q, q_1, q_2 \in [k]^2$ with $q = (i, j)$, $q_1 = (i_1, j_1)$, and $q_2 = (i_2, j_2)$ are such that (q_1, q_2) is series-compatible with q; and $F_\ell \subseteq B(G_\ell)$ realizes q_ℓ in (G_ℓ, s, t), for $\ell = 1, 2$.

We first observe that $F := F_1 \cup F_2$ realizes q.

Lemma 3. *F realizes q in (G, s, t).*

Proof. Since $V(G_1) \cap V(G_2) = \{t_1\}$ $(= \{s_2\})$, the set F is clearly acyclic. It remains to show $w(G, F, s, t) = c_i$. Every st-path in $\mathcal{P}(G, F, s, t)$ is the combination of an s_1t_1-path of $\mathcal{P}(G_1, F_1, s_1, t_1)$ with an s_2t_2-path of $\mathcal{P}(G_2, F_2, s_2, t_2)$. It follows

$$w(G, F, s, t) = \max\left\{w(G_1, F_1, s_1, t_1), w(G_2, F_2, s_2, t_2)\right\} = \max\{c_{i_1}, c_{i_2}\} = c_i,$$

where the last equality is from (S1). □

The proof of the next lemma is illustrated on Figure 3. It motivates the definition of series-compatibility.

Lemma 4. *Let G^+ be the graph G augmented with a red edge st of cost c_j, and G^+_ℓ (for $\ell = 1, 2$) the graph G_ℓ augmented with a red edge $s_\ell t_\ell$ of cost c_{j_ℓ}. Then for $\ell = 1, 2$ and every edge $uv \in F_\ell$,*

$$\widetilde{w}(G^+, F, u, v) = \widetilde{w}(G^+_\ell, F_\ell, u, v).$$

Proof. We prove the statement for $\ell = 1$, the case $\ell = 2$ follows by symmetry. Let $uv \in F_1$, and let $e = st$ and $e_1 = s_1t_1$ be the additional red edges in G^+ and G^+_1, respectively.

We first show:

Claim 1. $\widetilde{w}(G^+, F, u, v) \geq \widetilde{w}(G^+_1, F_1, u, v)$.

Proof. The claim is true if $\widetilde{\mathcal{P}}(G^+, F, u, v) = \varnothing$, since then $\widetilde{w}(G^+, F, u, v) = c_k \geq \widetilde{w}(G^+_1, F_1, u, v)$. Suppose thus $\widetilde{\mathcal{P}}(G^+, F, u, v) \neq \varnothing$, and let $P \in \widetilde{\mathcal{P}}(G^+, F, u, v)$. It is enough to show that $\mathrm{mc}(P) \geq \widetilde{w}(G^+_1, F_1, u, v)$. This clearly holds if $e \notin E(P)$, as P belongs then also to $\widetilde{\mathcal{P}}(G^+_1, F_1, u, v)$ (recall that $|V(G_1) \cap V(G_2)| = 1$). Hence, we may assume $e \in E(P)$. It follows $s_1, t_1 \in V(P)$.

Let P_1 denote the path of $\widetilde{\mathcal{P}}(G^+_1, F_1, u, v)$ obtained by replacing the subpath s_1Pt_1 of P with the edge e_1. Using (S2), we obtain

$$\mathrm{mc}(s_1Pt_1) = \max\{c_j, \mathrm{mc}(t_2Pt_1)\} \geq \max\{c_j, c_{i_2}\} = c_{j_1},$$

implying $\mathrm{mc}(P) \geq \mathrm{mc}(P_1) \geq \widetilde{w}(G^+_1, F_1, u, v)$. □

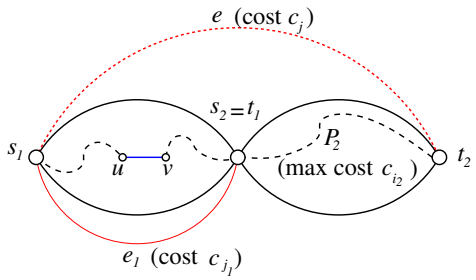


Figure 3: Series composition: illustration of the proof of Lemma 4.

Conversely, we prove:

Claim 2. $\widetilde{w}(G^+, F, u, v) \leq \widetilde{w}(G_1^+, F_1, u, v)$.

Proof. Again, this trivially holds if $\widetilde{\mathcal{P}}(G_1^+, F_1, u, v)$ is empty. Suppose thus $\widetilde{\mathcal{P}}(G_1^+, F_1, u, v) \neq \varnothing$, and let $P_1 \in \widetilde{\mathcal{P}}(G_1^+, F_1, u, v)$. Similarly as before, it is enough to show that $\widetilde{w}(G^+, F, u, v) \leq \mathrm{mc}(P_1)$. This is true if $e_1 \notin E(P_1)$, since then $P_1 \in \widetilde{\mathcal{P}}(G^+, F, u, v)$. Assume thus $e_1 \in E(P_1)$.

If $\mathcal{P}(G_2, F_2, s_2, t_2) = \varnothing$, then $i_2 = k$ and $\mathrm{mc}(P_1) \geq c_{j_1} = \max\{c_j, c_{i_2}\} = c_k \geq \widetilde{w}(G^+, F, u, v)$ by (S2). We may thus assume that $\mathcal{P}(G_2, F_2, s_2, t_2)$ contains a path P_2; we choose P_2 such that $\mathrm{mc}(P_2) = c_{i_2}$.

Denote by P the path obtained from P_1 by replacing the edge e_1 with the combination of edge e and path P_2. Since $P \in \widetilde{\mathcal{P}}(G^+, F, u, v)$, (S2) yields

$$\begin{aligned}
\mathrm{mc}(P_1) &= \max\{c_{j_1}, \mathrm{mc}(P_1 - e_1)\} \\
&= \max\{c_j, c_{i_2}, \mathrm{mc}(P_1 - e_1)\} \\
&= \max\{c_j, \mathrm{mc}(P_2), \mathrm{mc}(P_1 - e_1)\} \\
&= \mathrm{mc}(P) \\
&\geq \widetilde{w}(G^+, F, u, v).
\end{aligned}$$

□

The lemma follows from Claims 1 and 2. □

We are now ready to prove the correctness of the recursion step in Lemma 2.

Proof of Lemma 2. Let q and G^+ be defined as before. We first show:

Claim 3. *There exist* $q_1, q_2 \in [k]^2$ *such that* (q_1, q_2) *is series-compatible with* q *and* $\mathrm{OPT}_q(G, s, t) \leq \mathrm{OPT}_{q_1}(G_1, s, t) + \mathrm{OPT}_{q_2}(G_2, s, t)$.

Proof. Let $F \subseteq B(G)$ be a subset of blue edges realizing q in (G, s, t) such that

$$\mathrm{OPT}_q(G, s, t) = \sum_{uv \in F} \widetilde{w}(G^+, F, u, v).$$

For $\ell = 1, 2$, let also $F_\ell := F \cap E(G_\ell)$ and $q_\ell := (i_\ell, j_\ell)$, with i_ℓ the index such that $c_{i_\ell} = w(G_\ell, F_\ell, s_\ell, t_\ell)$, and $j_\ell := \max\{j, i_{\ell+1}\}$ (indices are taken modulo 2). F_ℓ ($\ell = 1, 2$) clearly realizes q_ℓ in (G_ℓ, s_ℓ, t_ℓ). It is also easily verified that (q_1, q_2) is series-compatible with q. Hence we can apply Lemma 4:

$$\begin{aligned}\mathrm{OPT}_q(G, s, t) &= \sum_{uv \in F} \widetilde{w}(G^+, F, u, v) \\ &= \sum_{uv \in F_1} \widetilde{w}(G_1^+, F_1, u, v) + \sum_{uv \in F_2} \widetilde{w}(G_2^+, F_2, u, v) \\ &\leq \mathrm{OPT}_{q_1}(G_1, s_1, t_1) + \mathrm{OPT}_{q_2}(G_2, s_2, t_2),\end{aligned}$$

as claimed. □

We now prove:

Claim 4. $\mathrm{OPT}_q(G, s, t) \geq \mathrm{OPT}_{q_1}(G_1, s_1, t_1) + \mathrm{OPT}_{q_2}(G_2, s_2, t_2)$ *holds for every* $q_1, q_2 \in [k]^2$ *such that* (q_1, q_2) *is series-compatible with* q*.*

Proof. Suppose that (q_1, q_2) is series-compatible with q. Let $F_\ell \subseteq B(G_\ell)$ ($\ell = 1, 2$) be a subset of blue edges of G_ℓ such that

$$\mathrm{OPT}_{q_\ell}(G_\ell, s_\ell, t_\ell) = \sum_{uv \in F_\ell} \widetilde{w}(G_\ell^+, F_\ell, u, v).$$

By Lemma 3, $F := F_1 \cup F_2$ realizes q in (G, s, t). Using again Lemma 4, we have:

$$\begin{aligned}\mathrm{OPT}_q(G, s, t) &\geq \sum_{uv \in F} \widetilde{w}(G^+, F, u, v) \\ &= \sum_{uv \in F_1} \widetilde{w}(G_1^+, F_1, u, v) + \sum_{uv \in F_2} \widetilde{w}(G_2^+, F_2, u, v) \\ &= \mathrm{OPT}_{q_1}(G_1, s_1, t_1) + \mathrm{OPT}_{q_2}(G_2, s_2, t_2),\end{aligned}$$

and the claim follows. □

The lemma follows from Claims 3 and 4. □

3.3 Parallel Compositions

The recursion step for parallel compositions follows a similar scheme. Let $q, q_1, q_2 \in [k]^2$ with $q = (i, j)$, $q_1 = (i_1, j_1)$, and $q_2 = (i_2, j_2)$. We say that the pair (q_1, q_2) is *parallel-compatible* with q if

(P1) at least one of i_1, i_2 is non-zero;

(P2) $\min\{i_1, i_2\} = i$;

(P3) $\min\{j, i_2\} = j_1$, and

(P4) $\min\{j, i_1\} = j_2$,

The recursion step for parallel composition is as follows.

Lemma 5. *Suppose that (G, s, t) is a parallel composition of (G_1, s, t) and (G_2, s, t), and that $q \in [k]^2$ is realizable in (G, s, t). Then*

$$\mathrm{OPT}_q(G, s, t) = \max\{\mathrm{OPT}_{q_1}(G_1, s, t) + \mathrm{OPT}_{q_2}(G_2, s, t) \mid (q_1, q_2) \text{ is parallel-compatible with } q\}.$$

In what follows, (G, s, t) is a parallel composition of (G_1, s, t) and (G_2, s, t); (q_1, q_2) is parallel-compatible with q; and $F_\ell \subseteq B(G_\ell)$ realizes q_ℓ in (G_ℓ, s, t), for $\ell = 1, 2$. Also, $F := F_1 \cup F_2$.

Similarly to Lemma 3, the definition of parallel-compatibility implies the following lemma.

Lemma 6. *F realizes q in (G, s, t).*

Proof. We have to prove that F is acyclic and that $w(G, F, s, t) = c_i$.

First, suppose that $(V(G), F)$ contains a cycle C. Since F_1 and F_2 are both acyclic, C includes the vertices s and t, and moreover $E(G_1) \cap E(C)$, $E(G_2) \cap E(C)$ are both non-empty. But then, there is an st-path in $(V(G), F_\ell)$ for $\ell = 1, 2$, implying $i_1 = i_2 = 0$, which contradicts (P1). Hence, F is acyclic.

Now, since each path of $\mathcal{P}(G, F, s, t)$ is included in either $\mathcal{P}(G_1, F_1, s, t)$ or $\mathcal{P}(G_2, F_2, s, t)$, it follows $w(G, F, s, t) = \min\{w(G_1, F_1, s, t), w(G_2, F_2, s, t)\} = \min\{c_{i_1}, c_{i_2}\}$, which equals c_i by (P2). □

The next lemma is the analogue of Lemma 4 for parallel compositions.

Lemma 7. *Let G^+ be the graph G augmented with a red edge st of cost c_j, and let G_ℓ^+ (for $\ell = 1, 2$) be the graph G_ℓ augmented with a red edge $s_\ell t_\ell$ of cost c_{j_ℓ}. Then for $\ell = 1, 2$ and every edge $uv \in F_\ell$,*

$$\widetilde{w}(G^+, F, u, v) = \widetilde{w}(G_\ell^+, F_\ell, u, v).$$

Proof. We prove the statement for $\ell = 1$, the case $\ell = 2$ follows by symmetry. Let $e = st$ and $e_1 = s_1 t_1$ be the additional red edges in G^+ and G_1^+, respectively.

Let $uv \in F_1$. Observe that $\widetilde{\mathcal{P}}(G^+, F, u, v)$ is empty if and only if $\widetilde{\mathcal{P}}(G_1^+, F_1, u, v)$ is. If both are empty, then $\widetilde{w}(G^+, F, x, y) = \widetilde{w}(G_1^+, F_1, u, v) = c_k$, and the claim holds. Hence, we may assume $\widetilde{\mathcal{P}}(G^+, F, u, v) \neq \varnothing$ and $\widetilde{\mathcal{P}}(G_1^+, F_1, u, v) \neq \varnothing$.

We first show:

Claim 5. $\widetilde{w}(G^+, F, u, v) \leq \widetilde{w}(G_1^+, F_1, u, v)$.

Proof. Let $P_1 \in \widetilde{\mathcal{P}}(G_1^+, F_1, u, v)$. It is enough to show $\widetilde{w}(G^+, F, u, v) \leq \mathrm{mc}(P_1)$. If $e_1 \notin E(P_1)$, then $P_1 \in \widetilde{\mathcal{P}}(G^+, F, u, v)$, and $\widetilde{w}(G^+, F, u, v) \leq \mathrm{mc}(P_1)$ holds by definition. Hence we may assume $e_1 \in E(P_1)$.

By (P3), we have $j_1 = \min\{j, i_2\}$. If $j_1 = j$, then replacing the edge e_1 of P_1 by e yields a path $P \in \widetilde{\mathcal{P}}(G^+, F, u, v)$ with $\mathrm{mc}(P) = \mathrm{mc}(P_1)$, implying $\widetilde{w}(G^+, F, u, v) \leq \mathrm{mc}(P_1)$. Similarly, if $j_1 = i_2 < j$, then $i_2 < k$, implying that $\mathcal{P}(G_2, F_2, s, t)$ is not empty. Replacing in P_1 the edge e with any path $P_2 \in \mathcal{P}(G_2, F_2, s, t)$ with $\mathrm{mc}(P_2) = c_{i_2}$ gives again a path P with $\mathrm{mc}(P) = \mathrm{mc}(P_1)$. While the path P_2 does not necessarily contain a red edge, the path P, on the other hand, cannot be completely

blue. This is because otherwise F contains the cycle $P \cup \{uv\}$, contradicting the fact that F is acyclic (as follows from Lemma 6). Hence, $P \in \widetilde{\mathcal{P}}(G^+, F, u, v)$, and $\widetilde{w}(G^+, F, u, v) \leq \mathrm{mc}(P) = \mathrm{mc}(P_1)$. Claim 5 follows. □

Conversely, we prove:

Claim 6. $\widetilde{w}(G^+, F, u, v) \geq \widetilde{w}(G_1^+, F_1, u, v)$.

Proof. Let $P \in \widetilde{\mathcal{P}}(G^+, F, u, v)$. Again, it is enough to show $\mathrm{mc}(P) \geq \widetilde{w}(G_1^+, F_1, u, v)$. This clearly holds if $P \in \widetilde{\mathcal{P}}(G_1^+, F_1, u, v)$. Hence, we may assume $s, t \in V(P)$, and that the subpath sPt of P either belongs to $\mathcal{P}(G_2, F_2, s, t)$, or corresponds to the edge e.

In the first case, $c_{i_2} \leq \mathrm{mc}(sPt)$ holds by definition. Moreover, $j_1 \leq i_2$ follows from (P3). Therefore, replacing the subpath sPt of P with the edge e_1 yields a path $P_1 \in \widetilde{\mathcal{P}}(G_1^+, F_1, u, v)$ with $\mathrm{mc}(P_1) \leq \mathrm{mc}(P)$, implying $\widetilde{w}(G_1^+, F_1, u, v) \leq \mathrm{mc}(P)$.

Similarly, (P3) implies $j_1 \leq j$ in the second case. Hence, replacing the edge e of P with e_1 results in a path $P_1 \in \widetilde{\mathcal{P}}(G_1^+, F_1, u, v)$ with $\mathrm{mc}(P_1) \leq \mathrm{mc}(P)$, showing $\widetilde{w}(G_1^+, F_1, u, v) \leq \mathrm{mc}(P)$. This completes the proof of Claim 6. □

Lemma 7 follows from Claims 5 and 6. □

Using the two previous lemmas, the proof of Lemma 5 is the same as that of Lemma 2 for series composition. We omit it.

3.4 The Algorithm

A series-parallel decomposition of a connected series-parallel graph can be computed in linear time [VTL82]. Given such a decomposition, Lemmas 2 and 5 yield the following algorithm: consider each graph (H, s, t) in the decomposition tree in a bottom-up fashion. If H is a single edge, compute $\mathrm{OPT}_q(H, s, t)$ for every $q \in [k]^2$. If (H, s, t) is a series or parallel composition of (H_1, s_1, t_1) and (H_2, s_2, t_2), compute $\mathrm{OPT}_q(H, s, t)$ for every $q \in [k]^2$ based on the previously computed values for (H_1, s_1, t_1) and (H_2, s_2, t_2), relying on Lemmas 2 and 5.

For every $q = (i, j) \in [k]^2$, there are $O(k)$ possible values for either series-compatible or parallel-compatible pairs (q_1, q_2). Hence every step costs $O(k)$ times. Since there are $O(k^2)$ possible values for q, and $O(m)$ graphs in the decomposition of G, the overall complexity is $O(k^3 m) = O(m^4)$.

This results in a polynomial-time algorithm computing the maximum revenue achievable by the leader. Moreover, using Lemmas 3 and 6, it is not difficult to keep track at each step of a witness $F \subseteq B(H)$ for $\mathrm{OPT}_q(H, s, t)$, whenever $\mathrm{OPT}_q(H, s, t) > -\infty$. This proves Theorem 2.

4 Bounded-Treewidth Graphs

In the previous section, we gave a polynomial-time algorithm for solving the StackMST problem on series-parallel graphs, which are biconnected graphs of treewidth 2. In this section, we show how to extend the algorithm to any graph of bounded treewidth, as indicated by the following theorem.

Theorem 3. *The* StackMST *problem can be solved in $m^{O(\omega^2)}$ time on graphs of treewidth ω.*

The *treewidth* of a graph can be defined in several ways. We follow Abrahamson and Fellows [AF93] and characterize a graph of treewidth ω as an *ω-boundaried graph*. An ω-boundaried graph is a graph with ω distinguished vertices (called *boundary vertices*), each uniquely labeled by a label in $\{1, \ldots, \omega\}$. ω-boundaried graphs are formed recursively by the following composition operators:

1. The null operator $\varnothing$ creates a boundaried graph which has only boundary vertices, and they are all isolated.

2. The binary operator $\oplus$ takes the disjoint union of two ω-boundaried graphs by identifying the ith boundary vertex of the first graph with the ith boundary vertex of the second graph. If there are only two boundary vertices s and t then this is exactly a parallel-composition.

3. The unary operator η introduces a new isolated vertex and makes this the new vertex with label 1 in the boundary. The previous vertex that was labeled 1 is removed from the boundary but not from the graph.

4. The unary operator ϵ adds an edge between the vertices labeled 1 and 2 in the boundary.

5. Unary operators that permute the labels of the boundary vertices.

Any ω-boundaried graph (and hence any graph of treewidth ω) can be constructed by applying $O(\omega n)$ compositions according to the above five operators. This construction as well as the boundary vertices can be found in linear time [Bod96].

4.1 Definitions

Given an ω-boundaried graph $G = (V, E)$ and two distinct boundary vertices $a, b \in \{1, 2, \ldots, \omega\}$, we call an ab-path *internal* if the only boundary vertices it passes through are a and b. We slightly modify the definition of $\mathcal{P}(G, F, a, b)$, $\widetilde{\mathcal{P}}(G, F, a, b), w(G, F, a, b)$, and $\widetilde{w}(G, F, a, b)$ from the previous section to only include internal ab-paths.

As in the series-parallel case, we want to keep track of optimal acyclic subsets of blue edges for every graph G obtained during the construction of a bounded-treewidth graph. Notice that the weights of edges in an optimal solution depend only on the bottlenecks of the new internal paths that will be added to G. We thus prepare $OPT(G)$s for every possible set of bottlenecks (the j_{ab}s in what follows) between any two boundary vertices a and b.

Let $I_{\omega \times \omega}$ be a matrix of pairs where $I[a, b] = I[b, a] = (i_{ab}, j_{ab})$ for some $i_{ab}, j_{ab} \in \{0, 1, \ldots, k\}$. Let $OPT_I(G)$ be the optimal solution to StackMST (that is, an acyclic subset of blue edges F) on the graph G^+ obtained from G by adding a red edge connecting a and b of cost $c_{j_{ab}}$ for every pair of distinct boundary vertices $a, b \in \{1, 2, \ldots, \omega\}$, subject to the conditions that for every distinct $a, b \in \{1, 2, \ldots, \omega\}$ we have $w(G, F, a, b) = c_{i_{ab}}$.

During the construction, we store for every graph G the partial solutions $OPT_I(G)$ for every possible I. In cases where $OPT_I(G)$ is undefined (no proper F exists), we set $OPT_I(G) = -\infty$. Also, we abuse the notation $OPT_I(G)$ for denoting both the acyclic subset F and its revenue.

4.2 The Algorithm.

We now describe our bounded-treewidth algorithm by showing how to maintain the OPT_I information as G is constructed by the five operators. We present the algorithm along with a proof sketch of its correction. We use (i_{ab}, j_{ab}) to denote I's pairs, (i^1_{ab}, j^1_{ab}) for I_1's pairs, (i^2_{ab}, j^2_{ab}) for I_2's pairs, and (i'_{ab}, j'_{ab}) for I''s pairs.

We begin with the null operator $\varnothing$ that creates a new graph G with isolated vertices labeled $1, \ldots, \omega$. Therefore, we set $OPT_I(G) = 0$ (associated with $F = \varnothing$) for every I whose entries are all of the form (k, j_{ab}). The value i_{ab} is required to be k as there are no internal paths at all; the j_{ab}s can be arbitrary values in $\{0, 1, \ldots, k\}$. For all other Is we set $OPT_I(G) = -\infty$.

If $G = G_1 \oplus G_2$, then $OPT_I(G) = \max\{OPT_{I_1}(G) \cup OPT_{I_2}(G)\}$. This operator is a lot like a parallel-composition of series-parallel graphs. Indeed, in the following conditions on the compatibility of I, I_1, I_2, the first four conditions are exactly the same as in a parallel-composition, only they must hold for every pair of boundary vertices (whereas in the series-parallel case there was only one pair). The fifth condition makes sure that the blue edges that will be purchased do not form a cycle. Therefore, if $G = G_1 \oplus G_2$, we require that

1. at least one of i^1_{ab}, i^2_{ab} is non-zero,
2. $i_{ab} = \min\{i^1_{ab}, i^2_{ab}\}$,
3. $j^1_{ab} = \min\{j_{ab}, i^2_{ab}\}$,
4. $j^2_{ab} = \min\{j_{ab}, i^1_{ab}\}$,

for every distinct $a, b \in \{1, \ldots, \omega\}$, and moreover that

5. the graph H is acyclic, where $V(H) = \{1, \ldots, \omega\}$ and distinct $a, b \in V(H)$ are adjacent in H if $i_{ab} = 0$.

If $G = \eta(G')$, then a new isolated boundary vertex v with label 1 is created, and the old 1-labeled vertex u is now no longer a boundary vertex. Since no edges are modified, the optimal solutions for G' and G are the same and we set $OPT_I(G) = OPT_{I'}(G')$. Notice that an ab-path between two distinct boundary vertices $a, b \neq u$ that go through u is not an *internal* path in G', but could be in G (if the path does not contain any other boundary vertex). This fact is captured by the first two of the following four conditions. If $G = \eta(G')$ we require that, for every distinct $a, b \in \{2, \ldots, \omega\}$,

1. $i_{ab} = \min\left\{i'_{ab}, \max\{i'_{a1}, i'_{1b}\}\right\}$,
2. $j'_{ab} = \min\left\{j_{ab}, \max\{j_{a1}, j_{1b}\}\right\}$,
3. $i_{a1} = k$, (there is no internal path incident to v as it is isolated)
4. $j'_{a1} = k$. (there will be no new internal paths originating from u)

If $G = \epsilon(G')$, then G is obtained from G' by adding an edge e between vertices labeled 1 and 2. Notice that $\epsilon(G') = G' \oplus G''$ where G'' is the boundaried graph which has only boundary vertices, and its only edge is e. Thus, instead of dealing with the ϵ operator we can introduce two new null-like operators that create a graph G isomorphic to G'' with the edge e being red and blue, respectively.

- If e is red with cost $c(e)$ then we set $OPT_I(G) = 0$ (associated with $F = \varnothing$) for every I whose entries are all of the form (k, j_{ab}) (the j_{ab}s can be anything in $\{0, 1, \ldots, k\}$) except that $i_{12} = c(e)$. For all other Is we set $OPT_I(G) = -\infty$.

- If e is blue then we set $OPT_I(G) = 0$ (associated with $F = \varnothing$) for every I whose entries are all of the form (k, j_{ab}) (the j_{ab}s can be anything in $\{0, 1, \ldots, k\}$).

  In addition, we set $OPT_I(G) = p(e)$ for every I whose entries are all of the form (k, j_{ab}) except that $i_{12} = 0$. This corresponds to setting $F = \{e\}$. For such a I, the price $p(e)$ assigned to e is determined as follows. Let P_{12} be the set of all ordered sequences of the form a_1-a_2-...-a_t where every $a_i \in \{1, 2, \ldots, \omega\}$ is a unique boundary vertex, $a_1 = 1$, $a_t = 2$, and $2 \leq t \leq \omega$ (if $t = 2$ then the path is simply 1-2). Every such sequence, together with e could close a cycle when new internal paths will be added. We therefore set

  $$p(e) = \min_{a_1\text{-}\ldots\text{-}a_t \in P_{12}} \max_{2 \leq i \leq t} j_{a_{i-1}a_i}$$

  For all other Is we set $OPT_I(G) = -\infty$.

Unary operators that permute the labels of the boundary vertices are trivial to handle. They merely represent a permutation of I.

Time Complexity. The composition operators require us to check every combination of at most three different Is for compatibility. There are k^{ω^2} possible Is, so we need to check $O(k^{3\omega^2})$ combinations. Each check requires at least $O(\omega^2)$ time to read the Is.

The most time-consuming check is the one of the ϵ operator when it adds a blue edge e. This might require figuring out $p(e)$. Notice that $|P_{12}| < \omega!$, so we can perform the check in $O(\omega!)$ time. The total complexity of the above algorithm is therefore bounded by $O(k^{3\omega^2} \cdot \omega!) = m^{O(\omega^2)}$.

Although the problem is polynomial for every constant value of ω, it is unclear whether there exists a fixed-parameter algorithm of complexity $O(f(\omega)n^c)$ for an arbitrary (possibly large) function f of ω and a constant c. In fact, we conjecture that under reasonable complexity-theoretic assumptions, such an algorithm does not exist.

References

[AF93] K. R. Abrahamson and M. R. Fellows. Finite automata, bounded treewidth, and well-quasiordering. In *Graph Structure Theory (ed. N. Robertson and P. Seymour)*, pages 539–564, 1993.

[Bak94] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.

[BGPW08] D. Bilò, L. Gualà, G. Proietti, and P. Widmayer. Computational aspects of a 2-player Stackelberg shortest paths tree game. In *Proc. 4th Workshop on Internet and Network Economics (WINE)*, volume 5385 of *Lecture Notes in Computer Science*, pages 251–262. Springer-Verlag, 2008.

[BHK08] 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.

[BKMK07] Glencora Borradaile, Claire Kenyon-Mathieu, and Philip N. Klein. A polynomial-time approximation scheme for Steiner tree in planar graphs. In *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[Bod96] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25:1305–1317, 1996.

[CDF+07] 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. To appear in Algorithmica.

[Cou08] B. Courcelle. Graph structure and monadic second-order logic: Language theoretical aspects. In *Proc. International Conference on Automata, Languages, and Programming (ICALP)*, volume 5125 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 2008.

[DHK09] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Ken-ichi Kawarabayashi. Approximation algorithms via structural results for apex-minor-free graphs. In *Proc. 36th International Colloquium on Automata, Languages and Programming (ICALP)*, 2009.

[DHM07] E. D. Demaine, M. Hajiaghayi, and B. Mohar. Approximation algorithms via contraction decomposition. In *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 278–287, 2007.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[GvLSU09] A. Grigoriev, J. van Loon, R. Sitters, and M. Uetz. Optimal pricing of capacitated networks. *Networks*, 53(1):79–87, 2009.

[Kle05] Philip N. Klein. A linear-time approximation scheme for TSP for planar weighted graphs. In *Proc. 46th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 146–155, 2005.

[Kle06] Philip N. Klein. A subset spanner for planar graphs, with application to subset TSP. In *Proc. 38th ACM Symposium on Theory of Computing (STOC)*, pages 749–756, 2006.

[LMS98] 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.

[RSM05] S. Roch, G. Savard, and P. Marcotte. An approximation algorithm for Stackelberg network pricing. *Networks*, 46(1):57–67, 2005.

[vH06] 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.

[vS34] H. von Stackelberg. *Marktform und Gleichgewicht (Market and Equilibrium)*. Verlag von Julius Springer, Vienna, 1934.

[VTL82] J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series parallel digraphs. *SIAM J. Comput.*, 11(2):298–313, 1982.