On the Stackelberg fuel pricing problem[*]

Cosimo Vinci[1] and Vittorio Bilò[2]

[1] Gran Sasso Science Institute, L'Aquila - Italy
cosimo.vinci.gssi@gmail.it
[2] Department of Mathematics and Physics "Ennio De Giorgi", University of Salento,
Provinciale Lecce-Arnesano, P.O. Box 193, 73100 Lecce - Italy
vittorio.bilo@unisalento.it

Abstract. We consider the Stackelberg fuel pricing problem in which a company has to decide the fuel selling price at each of its gas stations in order to maximize its revenue, assuming that the selling prices of the competitors and the customers' preferences are known in advance. We show that, even in the basic case in which the road network is modeled by an undirected planar graph and the competitors discriminate on two different selling prices only, the problem is APX-hard. On the positive side, we design a polynomial time algorithm for instances in which the number of gas stations owned by the company is constant, while, in the general case, we show that the single-price algorithm (which provides the best-known solutions for essentially all the Stackelberg pricing problems studied in the literature up to date) achieves an approximation ratio which is logarithmic in some parameters of the input instance. This result, in particular, is tight and holds for a much more general class of Stackelberg network pricing problems.

1 Introduction

A fundamental decisional process in many business activities concerns how to set the selling prices so as to maximize its own revenue, once knowing the selling prices of the competitors and the customers' preferences. The scenario in which the latter are implicitly defined in terms of some optimization problem are usually referred to as *Stackelberg pricing problems* [15]. These problems can be modeled as multi-player one-round games in which there is a special player, the *leader*, while all the others are *followers*. The first action is undertaken by the leader who decides on some parameters (e.g., the selling prices) and then the followers respond by deciding their actions. Each follower adopts, as her action, the optimal solution of a certain optimization problem (e.g., satisfying her own demand at the minimum cost) which depends on some of the parameters fixed by the leader and on some other values on which the leader has no control (e.g., the selling prices of the competitors). Since the followers' choices influence, in

[*] This work was partially supported by the PRIN 2010–2011 research project ARS TechnoMedia: "Algorithmics for Social Technological Networks" funded by the Italian Ministry of University.

turn, the leader's revenue, the determination of the best possible action for the leader often results in a challenging algorithmic problem.

A considerable research attention, see [2–8, 11, 12, 14], has been devoted in the last years to the study of Stackelberg network pricing problems based on some fundamental (polynomial time solvable) optimization problems, such as shortest paths, shortest path trees and minimum spanning trees. In this paper, we study the *Stackelberg fuel pricing problem* (SFPP) which is a Stackelberg network pricing problem based on the *gas station problem* (GSP): an optimization problem introduced in [9] to model situations in which drivers have to go from one location to another and have to decide where to fill their cars with fuel so as to minimize the travel cost, once knowing the fuel prices at the various gas stations along the road network.

Model and Notation. For a positive integer k, let $[k]$ denote the set $\{1, \dots, k\}$. A *road network* is an edge-weighted directed graph $G = (V, E, w)$, with $|V| = n$, $|E| = m$ and $w : E \to \mathbb{R}_{\geq 0}$ such that, for each $e = (u, v) \in E$, $w(e)$ specifies the amount of fuel (expressed in gallons) needed to go from u to v.

An instance (G, s, t, S, p) of the GSP is defined by a road network G, a pair of nodes $s, t \in V$, a set of nodes $S \subseteq V$ and a function $p : S \to \mathbb{R}_{\geq 0}$. The set of nodes S represents the locations of gas stations and the function p models the selling prices (per gallon) at each of the gas stations. There is a driver who needs to go from s to t. For the sake of simplicity, we assume that the driver's car is equipped with a tank of unlimited capacity and that $s \in S$, so that the driver can fill with as much fuel as she wants at price $p(s)$ when starting her trip. The driver wants to determine the best possible *itinerary*, that is, which (s, t)-path to drive through and which gas stations to stop at for fueling so as to minimize the total fuel cost[3].

An instance $(G, (s_i, t_i, \lambda_i)_{i \in [k]}, L, C, p_C, e)$ of the SFPP is defined by a road network G, k source-destination pairs (s_i, t_i) with an associated integer weight $\lambda_i \geq 1$ for each $i \in [k]$, two sets $L, C \subseteq V$, a function $p_C : C \to \mathbb{R}_{\geq 0}$ and a value $e \geq 0$. The sets of nodes L and C represent the locations of gas stations: the gas stations located at nodes in L are owned by the leader, while those located at nodes in C are owned by her competitors, so that the function p_C models the selling prices established by the competitors at each of their gas stations[4]. Note that we do not require L and C to be disjoint, i.e., either the leader and one of her competitors may own a gas station at the same location. The leader buys (or produces) the fuel at price e per gallon. There are k types of drivers (the followers) such that, for each $i \in [k]$, the driver of type i wants to go from s_i to t_i along the cheapest path in G, where λ_i denotes the number of drivers of type i, that is, how many drivers want to go from s_i to t_i. Once the leader has established a pricing function $p_L : L \to \mathbb{R}_{\geq 0}$ defining the fuel prices at her own gas stations, each follower of type $i \in [k]$ determines her action by

[3] The amount of fuel bought at each gas station s is implicitly defined by the minimum distance between s and the successive station chosen for fueling

[4] In the case in which more than one competitor owns a gas station in a given location v, $p_C(v)$ will denote the cheapest fuel price among them.

solving the instance $(G, s_i, t_i, L \cup C, p)$ of the GSP, in which, for each $v \in L \cup C$, $p(v) := \min\{p_L(v), p_C(v)\}$. The leader has to determine the pricing function $p_L : L \to \mathbb{R}_{\geq 0}$ providing her with the highest possible revenue. To this aim, we make the following simplifying assumption which is common in the setting of Stackelberg pricing problems: when the gas station problem has more than one optimal solution, the follower always chooses the one providing the leader with the highest revenue[5]. Furthermore, we assume that $s_i \in C\ \forall i \in [k]$, otherwise, either the problem is not feasible or the revenue is unbounded.

More formally, given a node $v \in V$ and a pricing function p_L for the leader, let $q(p_L, v) := 1_{p_L(v) \leq p_C(v)}$. For a pair of nodes $u, v \in V$, let $d(u, v)$ be the distance from u to v in G and $\mathcal{B}(u, v)$ be the set of itineraries connecting u to v, that is, the set of sequences of nodes $[u = v_{i_1}, v_{i_2}, \ldots, v_{i_r} = v]$ such that $v_{i_s} \in L \cup C$ for each $s \in [r-1]$. Set $p_s := p(v_{i_s})$, $d_s := d(v_{i_s}, v_{i_{s+1}})$ and $q_s := q(p_L, v_{i_s})$ for each $s \in [r-1]$. Given p_L and $B \in \mathcal{B}(u, v)$, a driver choosing itinerary B experiences a cost $c(p_L, B)$ and yields a contribution $g(p_L, B)$ to the leader's revenue which are defined as follows:

$$c(p_L, B) := \sum_{s=1}^{r-1} p_s \cdot d_s, \quad g(p_L, B) := \sum_{s=1}^{r-1} (p_s - e) \cdot d_s \cdot q_s.$$

Let $cg(p_L, B) = (c(p_L, B), g(p_L, B))$ and define a total ordering relation $\prec$ on $\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ such that $[x_1, y_1] \prec [x_2, y_2] \Leftrightarrow x_1 < x_2 \vee \{x_1 = x_2 \wedge y_1 > y_2\}$. Let $\mathcal{B}^*(u, v) \subseteq \mathcal{B}(u, v)$ be the set of itineraries $B \subseteq \mathcal{B}(u, v)$ minimizing $cg(p_L, B)$ according to the ordering relation $\prec$. Let $cg(p_L, u, v) := [c(p_L, u, v), g(p_L, u, v)] := \min_{B \in \mathcal{B}(u,v)} cg(p_L, B)$. Given a driver of type i, we have that $c(p_L, s_i, t_i)$ is the cost of her cheapest path, and $g(p_L, s_i, t_i)$ is her contribution to the leader's revenue, so that $g(p_L) := \sum_{i=1}^{k} \lambda_i \cdot g(p_L, s_i, t_i)$ is the leader's total revenue that has to be maximized. Let $p_M := \max_{i \in [k]} p_C(s_i)$. It is easy to prove that, if we set $p_L(v) > \min\{p_M, p_C(v)\}$ for some $v \in L$, no driver will stop at station v for fueling. Similarly, whenever $p_C(v) > p_M$ for some $v \in C$, no driver will stop at station v for fueling. Therefore, we can suppose without loss of generality that $p_C(v) \leq p_M\ \forall v \in C$, the leader's revenue $g(p_L)$ is bounded and, in order to be maximized, p_L can be chosen in such a way that $e \leq p_L(v) \leq p_C(v)\ \forall v \in L$.

Related Work. The GSP has been introduced by Khuller, Malekian and Mestre in [9]. They give polynomial time algorithms for the basic problem and for some of its variations.

Briest, Hoefer and Krysta author an influential work [6] on Stackelberg network pricing problems which widely generalizes previous results in the field. In particular, they consider the case in which the edges of the network are partitioned into two sets: the set of *fixed-price edges* and that of *priceable edges*, with the latter owned by the leader. Each follower buys a subnetwork of minimum cost and so the leader wants to assign suitable prices to the priceable edges so

[5] In fact, if this is not the case, the leader can decrease some selling price of a negligible amount so as to achieve almost the same revenue as in the case in which the assumption holds.

as to maximize her revenue. They study the approximation guarantee of the *single-price algorithm* in this general class of problems. This algorithm, which assigns the same (suitably computed) price to all priceable edges, has been first analyzed in [7] for the case of a single follower buying a minimum spanning tree. Briest, Hoefer and Krysta show that, for the case of a single follower, the approximation guarantee is $(1+\epsilon)H_h$, where $\epsilon > 0$ is an arbitrary value, h is the number of priceable edges and H_i is the ith harmonic number, while, for the case of k followers, it becomes $(1+\epsilon)(H_h+H_k)$. Finally, when the followers may have different weights, they show that the single-price algorithm achieves an approximation guarantee of $(1+\epsilon)h^2$ and also provide a lower bound of $O(h^\epsilon)$ on the approximability of the problem.

Determining whether there are approximation algorithms better than the single-price one is, perhaps, the most important open problem in this field of research. In fact, while the performance of this algorithm remains essentially the same even when instantiated to specific optimization problems such as shortest paths, shortest path trees and minimum spanning trees, the impossibility results known so far in these cases only refer to APX-hardness, see [3, 5, 7].

Our Contribution. We show that the SFPP is APX-hard even in the basic case in which the road network is modeled by an undirected planar graph and the competitors discriminate on two different selling prices only, by means of a reduction from the maximum independent set problem on cubic graphs. This reduction, however, requires that $|L|$ is a non-constant value. This assumption is essential, anyway, since, for the case in which $|L| = O(1)$, we show that the problem can be solved in polynomial time.

We stress that the SFPP does not fall within the scope of the Stackelberg network pricing problems defined by Briest, Hoefer and Krysta in [6] and that the presence of additional parameters in the definition of the problem (in particular, the edge-weights) makes the performance of the single-price algorithm unlikely to be uninfluenced by the characteristics of the road network given in input. To this aim, we define a general class of Stackelberg network pricing problems which extends the one given by Briest, Hoefer and Krysta and includes the SFPP. For this class of problems, we show that the single-price algorithm provides an approximation guarantee which is logarithmic in some parameters of the input instance (see the claim of Theorem 4 for the exact characterization) and that this bound is tight.

Due to space limitations, some details and proofs have been removed.

2 Complexity Results

We first show that the SFPP is APX-hard even under some restrictions.

Theorem 1 *The* SFPP *is APX-hard even when p_C assigns only two different prices and G is an undirected planar graph as long as $|L| \in \Theta(n)$.*

Proof. We consider a polynomial reduction from the Maximum Indipendent Set Problem on cubic graphs (MISP). Given a cubic graph (U, F), with $|U| = n$

and $|F| = m$, and an arbitrary value $\theta \in]0, 1/2]$ polynomially representable with respect to n, consider an instance $(G, (s_i, t_i, \lambda_i)_{i \in [k]}, L, C, p_C, e)$ of the SFFP that we define incrementally as follows. Suppose that the sets V, E, L and C are initially empty. Fix an arbitrary orientation on the edges in U, insert a node v^* in V with $p_C(v^*) = 1 + 3\theta$, then, $\forall i \in [n]$, insert in V three nodes $\overline{v}_i^1$, $\overline{v}_i^2$ and $\overline{v}_i^3$ with $p_C(\overline{v}_i^1) = p_C(\overline{v}_i^3) = 1 + 3\theta$ and $p_C(\overline{v}_i^2) = 4\theta$; $\overline{v}_i^1$ belongs to L. Then, $\forall i \in [n]$, insert in E three edges $\overline{e}_i^1 = \{\overline{v}_i^1, \overline{v}_i^2\}$, $\overline{e}_i^2 = \{\overline{v}_i^2, \overline{v}^*\}$ and $\overline{e}_i^3 = \{\overline{v}_i^3, \overline{v}_i^1\}$ with $\omega(\overline{e}_i^1) = \theta$, $\omega(\overline{e}_i^2) = 1/2 - \theta$ and $\omega(\overline{e}_i^3) = 1 - \theta$. $\forall e_j = (u_i, u_h) \in F$, denote $\overline{v}_i^1$ with v_j^1, $\overline{v}_i^2$ with v_j^2, $\overline{v}_h^2$ with v_j^3, $\overline{v}_h^1$ with v_j^4 and $\overline{v}_h^3$ with v_j^5. There is a driver $(\overline{v}_i^1, \overline{v}_i^2)$ $\forall i \in [n]$, and a driver (v_j^1, v_j^5) $\forall j \in [m]$. Finally, set $e = 3\theta$.

We stress that the constructed road network G is a planar graph.

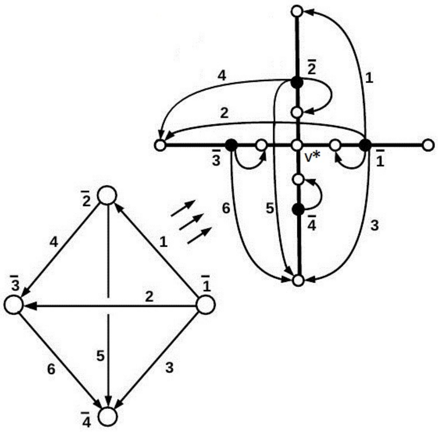

Fig. 1. We consider, for example, the polynomial reduction applied to a complete graph (U, F) with $|U| = 4$, depicted on the left. In (U, F) the edges are arbitrary oriented and there is an enumeration of vertices and edges such that the vertex numbers are overlined, whereas the edge numbers are not. On the right, we have the road network G obtained from U. The index $\overline{i}$ indicates the driver $(\overline{v}_i^1, \overline{v}_i^2)$ $\forall i \in [4]$, and the index j indicates the driver (v_j^1, v_j^5) $\forall j \in [6]$. The starting and the arriving nodes of the source-destination pair associated with every driver are depicted trough an arrow.

The following lemmas hold.

Lemma 1 *Consider the instance obtained by the reduction from (U, F). Given a pricing function p_L, let p'_L be the pricing function such that $p'_L(\overline{v}_i^1) = 4\theta$ if $p_L(\overline{v}_i^1) \leq 4\theta$ and $p'_L(\overline{v}_i^1) = 1 + 3\theta$ otherwise. It holds that $g(p'_L) \geq g(p_L)$.*

Proof (Sketch). We first prove that, independently from p_L, the cheapest path for driver $(\overline{v}_i^1, \overline{v}_i^2)$ is $[\overline{v}_i^1, \overline{v}_i^2]$ $\forall i \in [n]$ and that the cheapest path for driver (v_j^1, v_j^5) is $[v_j^1, v_j^2, v^*, v_j^3, v_j^4, v_j^5]$ $\forall j \in [m]$. At this point, we prove that, by setting to 4θ all the prices not bigger than 4θ and to $1+3\theta$ the remaining ones (thus obtaining p'_L), the leader gets a revenue of $g(p'_L) \geq g(p_L)$. □

Lemma 2 *Given the pricing function p'_L of the previous lemma, there exists a pricing function p^*_L which can be obtained in polynomial time from p'_L such that $p^*_L(\overline{v}_i^1) = 4\theta$ if exists $f_j \in \delta^+_{(U,F)}(u_i)$ such that $p'_L(v_j^1) = p'_L(v_j^4) = 1 + 3\theta$, and $p^*_L(\overline{v}_i^1) = p'_L(\overline{v}_i^1)$ otherwise. Moreover, it holds that $g(p^*_L) \geq g(p'_L)$ and that $g(p^*_L) = |\{i \in [n] : p^*_L(\overline{v}_i) = 1 + 3\theta\}|(\theta - \theta^2) + \theta^2 n + (2\theta - \theta^2)m$.*

Proof (Sketch). First we prove that, given $j \in [m]$ such that $p'_L(v_j^1) = p'_L(v_j^4) = 1 + 3\theta$, the leader's revenue does not decrease when decreasing the value $p'_L(v_j^1)$ to 4θ. By repeating this operation $\forall j \in [m]$ such that $p'_L(v_j^1) = p'_L(v_j^4) = 1 + 3\theta$, we obtain p_L^* which, by induction, satisfies $g(p_L^*) \geq g(p_L)$. Hence, we get $g(p_L^*, \overline{v}_i^1, \overline{v}_i^2) = (p_L^*(\overline{v}_i^1) - 3\theta)\theta \ \forall i \in [n]$ and $g(p_L^*, v_j^1, v_j^5) = 2\theta - \theta^2$. By summing these values for all drivers, we obtain the claimed value for $g(p_L^*)$. □

Now, we prove the theorem. Let $1 + \epsilon$ be a lower-bound on the approximability of MISP (see [1] for a proof of the APX-hardness of MISP). We prove, by contradiction, that SFPP cannot be $(1 + \delta)$-approximable $\forall \delta < \frac{\epsilon}{13}$. Suppose that there exists a $(1 + \delta)$-approximation algorithm for the SFPP with $\delta < \frac{\epsilon}{13}$. Let p_L^* be an optimal pricing function and p_L be the pricing function returned by the approximation algorithm. We have that $g(p_L^*)/g(p_L) \leq 1 + \delta$. By Lemma 2, we can suppose without loss of generality that p_L verifies $p_L(\overline{v}_i^1) \in \{4\theta, 1 + 3\theta\}$ $\forall i \in [n]$ and $p_L(v_j^1) = 4\theta$ or $p_L(v_j^4) = 4\theta \ \forall j \in [m]$. In fact, if this is not the case, we can apply the polynomial time transformation outlined in the proof of Lemma 2 so as to obtain from p_L a pricing function p'_L verifying the assumption and such that $p_L^*/p'_L \leq p_L^*/p_L$. Clearly, again by Lemma 2 and the optimality of p_L^*, the same assumption can also be made for p_L^*.

Let $M(p_L) := \{u_i \in U : p_L(\overline{v}_i^1) = 1 + 3\theta\}$. From the assumptions on p_L^* and p_L, it follows that $M^* := M(p_L^*)$ and $M := M(p_L)$ are independent sets of (U, F). By the optimality of p_L^* and because of the characterization of $g(p_L^*)$ given in Lemma 2, M^* has to be a maximum independent set of (U, F). Let $\theta := n^{-2}$. Again by the characterization of the leader's revenue given in Lemma 2, for a sufficiently big n, we have

$$\frac{g(p_L^*)}{g(p_L)} = \frac{(|M^*| + 2m + (n - |M^*| - m)n^{-2})n^{-2}}{(|M| + 2m + (n - |M| - m)n^{-2})n^{-2}} \sim \frac{|M^*| + 2m}{|M| + 2m} \leq 1 + \delta. \quad (1)$$

By manipulating (1), we have that $|M^*|/|M| \leq 1 + \delta + 2m\delta/|M|$. In a graph of degree 3 and m edges, we can find in polynomial time an independent set with at least $\lceil m/6 \rceil$ nodes. So, we can suppose that $|M| \geq m/6$. Therefore, $|M^*|/|M| \leq 1 + \delta + 2m\delta/|M| \leq 1 + 13\delta < 1 + \epsilon \Rightarrow |M^*|/|M| < 1 + \epsilon$, thus contradicting the $1 + \epsilon$ lower bound on the approximability of the MISP. □

Note that, in the claim of Theorem 1, we required that $|L| = \Theta(n)$. However, if we consider instances of the SFPP such that $|L| = O(1)$, then the problem can be efficiently solved. We prove this fact by designing an algorithm, called SFPP-CL, which solves a polynomial number of linear systems in order to determine the best possible pricing function for the leader.

Let $K = \{(s_i, t_i) : i \in [k]\}$. For a pair of nodes (u, v), let $c_\infty(u, v)$ be the minimum cost paid by a driver to go from u to v without stopping at gas stations in L. Given a driver going from u to v, consider an optimal itinerary $B \in \mathcal{B}^*(p_L, u, v)$. There exists an itinerary D extracted from B, that we call *strong itinerary*, of the form $D = (u = v_1^1, v_2^1, \ldots v_{t(1)}^1 = v_1^C, v_1^2, v_2^2, \ldots v_{t(2)}^2 = v_2^C, \ldots \ldots v_{r-1}^C, v_1^r, v_2^r, \ldots v_{t(r)}^r = v)$, with $v_s^C \in C \setminus L \ \forall s \in [r-1]$ while all the other nodes belong to L, such that the cost c and the revenue g with respect to B can be computed as follows (set $p_z^s := p_L(v_z^s)$, $d_z^s := d(v_z^s, v_{z+1}^s)$ and $c_\infty(v_r^C, v_1^{r+1}) := 0$):

$$c(p_L, D) := \sum_{s=1}^{r} \sum_{z=1}^{t(s)-1} p_z^s \cdot d_z^s + c_\infty^s, \quad g(p_L, D) := \sum_{s=1}^{r} \sum_{z=1}^{t(s)-1} (p_z^s - e) \cdot d_z^s.$$

Set $cg(p_L, D) := [c(p_L, D), g(p_L, D)]$ and let $\mathcal{D}(u, v)$ be the set of strong itineraries starting at u and ending at v. It holds that $cg(p_L, u, v) = \min_{D \in \mathcal{D}(u,v)} cg(p_L, D)$. From this fact, it follows that an optimal pricing function p_L^* is an optimal solution to an LP problem $LP(\{D_{uv}\}_{(u,v)\in K})$, yielded by a set of optimal strong itineraries $\{D_{uv}\}_{(u,v)\in K}$, whose objective function to be maximized is $g(p_L)$ and the constraints are: *(i)* $c(p_L, D_{uv}) \leq c(p_L, D)$ for each $D \in \mathcal{D}(u, v)$, and *(ii)* $e \leq p_L(v) \leq p_C(v)$ for each $v \in L$. From the LP Theory, we know that there exist $|L|$ constraints in $LP(\{D_{uv}\}_{(u,v)\in K})$ defining a linear system of $|L|$ equations in $|L|$ variables whose unique solution is p_L^*. By evaluating the revenue yielded by all the pricing functions that solve all the possible linear systems and returning the one giving the highest leader's revenue, we obtain an optimal pricing function p_L^*.

We now describe algorithm SFPP-CL, which is based on a refinement of the ideas outlined above. SFPP-CL is divided into two phases: an *initializing phase*, in which several data structures are created and a *running phase*, in which SFPP-CL evaluates the leader's revenue yielded by several pricing functions obtained trough the data structures defined during the initializing phase. Let $K_U = \{u \in V : (u, v) \in K\}$ and $K_V = \{v \in V : (u, v) \in K\}$.

SFPP-CL: **initializing phase**

Step 1 Compute $c_\infty(u, v)$ $\forall u, v \in V$.

Step 2 $\forall u \in L$, $v \in L \cup K_V$, let $X_{uv} := [e = x_{uv}(0), x_{uv}(1), \ldots, x_{uv}(r_{uv}) = p_C(u)]$ be the vector given by the abscissa of all the vertices of the polyhedron $\mathcal{P}_{uv} := \bigcap_{z\in C\cup\{v\}} \{[x, y] : e \leq x \leq p_C(u), y \leq f_{uv}^z(x) := d(u, z) \cdot x + c_\infty(z, v)\}$ listed in increasing order (in order to compute X_{uv}, see [13]). Let $Z_{uv} := [z_{uv}(s)]_{s\in[r_{uv}]}$ be the vector such that $z_{uv}(s) = \arg\max_{z\in C\cup\{v\}}\{d(u, z) : [x_{uv}(s), f_{uv}^z(x_{uv}(s))]$ is a vertex of $\mathcal{P}_{uv}\}$ $\forall s \in [r_{uv}]$.

The usefulness of the vectors X_{uv} and Z_{uv} is that, if an optimal pricing function p_L^* verifies that $p_L^*(u) \in]x_{uv}(s-1), x_{uv}(s)]$ (set $x_{uv}(-1) = -\infty$), then the itinerary $D_{uv}^* := [u, z_{uv}(s), v]$ minimizes $cg(p_L^*, D)$ among all the itineraries of the form $D = [u, z, v]$ with $z \in C \cup \{v\}$. This observation will imply the correctness of the running phase of SFPP-CL. Before describing this phase, we present an algorithm that, given p_L, computes $g(p_L)$ and which will be used as a subroutine in the running phase (note that this algorithm can also be used to certify that the SFPP belongs to NP).

SFPP-CL-C **algorithm (**SFPP-CL**-Certificate algorithm)**

Step 1 Given $a \in L$ and $b \in L \cup K_V$, let s_{ab} be the smallest strictly positive integer such that $p_L(a) \leq x_{ab}(s_{ab})$ and, given $(u,v) \in K$, let $G_{uv} = (V_{uv}, E_{uv}, w_{uv})$ be a connected weighted graph that has an edge (a,b) with $w(a,b) = [c_\infty(a,b), 0]$ if $a \in \{u\} \setminus L$ and $b \in L \cup \{v\}$, $w(a,b) = [p_L(a) \cdot d(a, z_{ab}(s_{ab})) + c_\infty(z_{ab}(s_{ab}), b), (p_L(a) - e) \cdot d(a, z_{ab}(s_{ab}))]$ if $a \in L$ and $b \in L \cup \{v\}$.

Step 2 Evaluate the length of the shortest path from u to v in the graph G_{uv} $\forall (u,v) \in K$, which is equal to $g(p_L, u, v)$ (because of the observation we did when discussing the initializing phase). Finally, compute and return $g(p_L)$.

SFPP-CL**: running phase**

Step 1 Fix a set $K_L \subseteq K$ of $|L|$ elements. Let $K_U^L := K_U \cap K_L$ and $K_V^L := K_V \cap K_L$. Given $u \in L$, let $X_u = [e = x_u(0), x_u(1), \ldots, x_u(r_u) = p_C(u)]$ be the vector obtained by sorting the elements of the vectors X_{uv} with $v \in L \cup K_V^L$. Let $Z_u = (z_u(t,v))_{t \in [r_u], v \in L \cup K_V}$ be a matrix of nodes in V, in which the generic $z_u(t,v)$ is equal to the node $z_{uv}(s)$ if $x_{uv}(s-1) < x_u(t) \leq x_{uv}(s)$, where $s \in [r_{uv}]$.

Step 2 Let $T = [t(u)]_{u \in L}$ be a vector of integers indexed in L, such that $1 \leq t(u) \leq r_u$ $\forall u \in L$. Let $G_T = (V_T, E_T)$ be a connected graph such that E_T has the edges $(u, z_u(t(u), v)), (z_u(t(u), v), v), (z, v)$ with $z \in K_U^L$, $u \in L$, $v \in L \cup K_V^L$.

Step 3 Fix $L_-, L_+ \subseteq L$ such that $L_- \cap L_+ = \emptyset$ and set $r := |L| - |L_-| - |L_+|$. Let $\mathcal{D}_T^2 := \{(D_s, D'_s)\}_{s \in [r]}$ be a set of r pairs of strong itineraries such that, $\forall s \in [r]$, D_s and D'_s both start at u and end at v, with $(u,v) \in K_L$ and are simple paths in G_T.

Step 4 Solve a linear system in the unknown variables yielded by p_L, with the following $|L|$ equations: $c(p_L, D_s) = c(p_L, D'_s)$ $\forall s \in [r]$, $p_L(u) = x_u(t(u)-1)$ $\forall u \in L_-$ and $p_L(u) = x_u(t(u))$ $\forall u \in L_+$. If the system has only one solution p_L with $p_L(u) \in]x_u(t(u)-1), x_u(t(u))[$ $\forall u \in L \setminus \{L_- \cup L_+\}$, compute $g(p_L)$ by using SFPP-CL-C.

Step 5 Run Steps 1, 2, 3 and 4 for all possible choices of K_L, T, L_-, L_+ and $\mathcal{D}_T^2$ and return the best pricing function among the considered ones.

The following theorem holds.

Theorem 2 SFPP-CL *solves the* SFPP *in polynomial time when* $|L| = O(1)$.

3 An Approximation Algorithm: the Class SGPS

Given a generic pricing problem, a *uniform pricing function*, or UPF, is an assignment of the same price to all the priceable elements. An *optimal uniform pricing function*, or UPF*, maximizes the leader's revenue among all the UPFs. We introduce the class of *Stackelberg Games with Priceable Sets*, or SGPS, extending that of *Stackelberg Network Pricing Games* described in [6], and characterize the approximation factor provided by an UPF* in these games. Furthermore, we define an algorithm that finds an UPF^*, prove that the SFPP belongs to class SGPS and adapt such an algorithm to the SFPP.

A game in SGPS is a tuple $(E, C, L, p_C, w, \mathcal{E}, K, \{\lambda_i\}_{i\in K}, \{\mathcal{F}_i\}_{i\in K},)$ such that E is a set, $\{C, L\}$ is a partition of E, $p_C : C \to \mathbb{R}_{\geq 0}$ is a pricing function, $w : L \to \mathbb{R}_{\geq 0}$ is a weight function, $\mathcal{E} := \{E_1, E_2, \dots E_r\}$ is a partition of L, K is a set of followers and, $\forall i \in K$, λ_i is the weight of follower i and $\mathcal{F}_i \subseteq \mathcal{P}(E)$ is the family of sets that can be purchased by follower i. We call *competitor's elements* the elements of C and *leader's elements* the elements of L. Given $p_L : \mathcal{E} \to \mathbb{R}_{\geq 0}$ and $F \subseteq E$, we define a cost c and a revenue g as follows:

$$g(p_L, F) := \sum_{E'\in\mathcal{E}} \sum_{e\in E'\cap F} w(e)\cdot p_L(E'), \quad c(p_L, F) := \sum_{e\in F\cap C} p_C(e) + g(p_L, F).$$

Let $cg(p_L, F) := [c(p_L, F), g(p_L, F)]$. Given $i \in K$, let $[c(p_L, i), g(p_L, i)] := cg(p_L, i) := \min_{F\in\mathcal{F}_i} cg(p_L, F)$, where the minimum is defined according to the ordering relation $\prec$. Informally, $c(p_L, i)$ is the cost of follower i when choosing a subset $F \in \mathcal{F}_i$ of minimum cost and $g(p_L, i)$ is the contribution of follower i to the leader's revenue recalling that, when a follower has different optimal choices, she adopts the one providing the highest revenue to the leader. Let $F^*(p_L, i) \in \mathcal{F}_i$ be a set minimizing the function $cg(p_L, i)$, that is, an optimal choice for follower i. The leader's revenue $g(p_L)$ is equal to $\sum_{i\in K} \lambda_i \cdot g(p_L, i)$. In order for g to be bounded, we suppose that, $\forall i \in K$, $\exists F \in \mathcal{F}_i$ such that $F \subseteq C$. Our objective is to find a pricing function p_L^* maximizing g. Different families of games in SGPS can be defined by different choices of $\{\mathcal{F}_i\}_{i\in K}$. The main differences with the Stackelberg Network Pricing Games are that single elements cannot be priced independently in general, since the leader has to assign the same price to every set in $\mathcal{E}$, and that the prices are weighted by the function w.

Given $\theta \geq 0$, p_θ is an UPF such that $p_\theta(E') = \theta\ \forall E' \in \mathcal{E}$. Let SG_1 be a game in SGPS with one follower only (it is then possible to avoid considering the follower's weight and to omit the index 1 in several functions) and let SG_2 be the game obtained from SG_1 by setting $\mathcal{E} = \{\{e\}\}_{e\in L}$. Note that the maximum leader's revenue g_1^* in SG_1 is lower than the maximum leader's revenue g_2^* in SG_2 and that a same UPF gives the same leader's revenues in both games. Hence, given the leader's revenues $g_{1,\theta}^*$ and $g_{2,\theta}^*$ yielded by an UPF^* for SG_1 and SG_2, respectively, we have that $g_1^*/g_{1,\theta}^* \leq g_2^*/g_{2,\theta}^*$. From this observation, in order to find un upper bound on the approximation factor yielded by an UPF^* for both games SG_1 and SG_2, we can restrict our analysis only to SG_2. Let $W, C : \mathcal{P}(E) \to \mathbb{R}_{\geq 0}$ be two functions such that $W(F) = \sum_{e\in F\cap L} w(e)\ \forall F \subseteq E$ and $C(F) = \sum_{e\in F\cap C} p_C(e)$. Let $X := \{W(F^*(p_\theta)) > 0 : \theta \geq 0\}$ be the *optimal weights* vector of follower 1 and consider it as a vector $X := [x(j) := x_j]_{j\in[n]}$ sorted in increasing order.

We define three functions $\theta, c, \varDelta : X \to \mathbb{R}_{\geq 0}$ such that, given $x_j \in X$, it holds that $\theta(x_j) := \theta_j := \max\{\theta \in \overline{\mathbb{R}}_{\geq 0} : W(F^*(p_\theta)) = x_j\}$, $c(x_j) := c_j := \min\{C(F) : W(F) = x_j, F \in \mathcal{F}\}$ and $\varDelta(x_j) := \varDelta_j := c(x_0) - c(x_j)$. Note that the function θ is decreasing in its argument. We call functions θ and c, respectively, the *optimal prices* function and the *optimal costs* function of follower 1. We observe that, $\forall j \in [n]$, assuming that $\theta_0 \cdot x_0 = \infty \cdot 0 = 0$, it holds that

$c(p_{\theta_j}) = c_j + \theta_j \cdot x_j = c_{j-1} + \theta_j \cdot x_{j-1}$. This implies the following equation:

$$\theta_j = \frac{c_{j-1} - c_j}{x_j - x_{j-1}} = \frac{\Delta_j - \Delta_{j-1}}{x_j - x_{j-1}}. \tag{2}$$

Let p_L^* be an optimal pricing function, the following relationship holds: $c(p_L^*) - g(p_L^*) = C(F^*(p_L^*)) \geq \min\{C(F) : F \in \mathcal{F}\} = C(F^*(p_0)) = c_n$. Moreover, we have $c(p_L^*) \leq c_0$. By using the previous inequalities, we get

$$g(p_L^*) = c(p_L^*) - (c(p_L^*) - g(p_L^*)) \leq c_0 - c_n = \Delta_n. \tag{3}$$

Let $W_m := x_1$, $W_M := x_n$, $\theta_m := \theta_n$, $\theta_M := \theta_1$. By exploiting inequalities (2) and (3), it is possible to prove the following fundamental theorem in analogy with the results in [6].

Theorem 3 *Any* UPF^* p_{θ^*} *provides an approximation guarantee of* $1 + \ln(\min\{W_M/W_m, \theta_M/\theta_m\})$ *to both* SG_1 *and* SG_2.

Now we can generalize the previous theorem to the case of more followers. We use the subscript i to denote the quantities previously defined for the case of a single follower when instantiated to follower i. Let SG_0 be a generic game in SGPS and define $\theta_m \leq \min_{i\in K} \theta_{m,i}$, $\theta_M \geq \max_{i\in K} \theta_{M,i}$, $W_m \leq \min_{i\in K} W_{m,i}$, $W_M \geq \max_{i\in K} W_{M,i}$. Let $\lambda_m := \min_{i\in K} \lambda_i$ and $\lambda_M := \sum_{i\in K} \lambda_i$.

The following result holds.

Theorem 4 *Any* UPF^* p_{θ^*} *provides an approximation guarantee of* $1 + \ln(\min\{(W_M/W_m) \cdot (\lambda_M/\lambda_m), \theta_M/\theta_m\})$ *to* SG_0.

Now, we design an algorithm to find an UPF^*. Given $i \in K$, let $X^i := [x^i(j) := x^i_j]_{j\in[n_i]}$ be the optimal weights vector of follower i and let $X := [x(h) := x_h]_{h\in[n]}$ be a vector sorted in increasing order containing all the values in X^i and such that $\exists F \in \mathcal{F}_i : x(h) = W(F)$ $\forall h \in [n]$, $\forall i \in K$. Moreover, let $C^i := [c^i(h) := c^i_h]_{h\in[n]}$ be the vector such that, $\forall h \in [n]$, $c^i(h) = c(x_h)$, where c denotes the optimal costs function of follower i. We suppose that the vectors X and $(C^i)_{i\in K}$ are known. Let $\Theta^i := [\theta^i(j) := \theta^i_j]_{j\in[n_i]}$ be the vector such that, $\forall j \in [n_i]$, $\theta^i(j) = \theta(x^i_j)$, where θ denotes the optimal prices function of follower i. Observe that Θ^i is the vector given by the positive abscissa of all the vertices of the polyhedron $\mathcal{P}_i := \bigcap_{h\in[n]} \left\{[\theta, y] : \theta \in \mathbb{R}, y \leq f^i_h(\theta) := c^i_h + \theta \cdot x_h\right\}$, and X^i is the vector such that $x^i_j = \max\{x_h : [\theta^i_j, f^i_h(\theta^i_j)]$ is a vertex of $\mathcal{P}_i\}$. By these characterizations we can compute $(X^i)_{i\in K}$ and $(\Theta^i)_{i\in K}$ in $O(|K|n\log(n))$ time (see [13] for further details). Let $\Theta = [\theta(h) := \theta_h]_{h\in[m]}$ be the sorted fusion of the vectors $(\Theta^i)_{i\in K}$ without the eventual null element. Θ can be computed in $O(|K|n\log(|K|))$ time. Given $i \in K$ and $h \in [m]$, let $j(h,i) \in [n_i]$ be such that $\theta^i_{j(h,i)} \geq \theta_h > \theta^i_{j(h,i)+1}$ (set $\theta^i_{n_1+1} := 0$). Observe that $W(F^*(p_{\theta_h}, i)) = x^i_{j(h,i)}$ and then $g(p_{\theta_h}, i) = \theta_h \cdot x^i_{j(h,i)}$. Since there exists $h^* \in [m]$ such that $p_{\theta_{h^*}}$ is a UPF^*, by the above arguments, we have that

$$g(p_{\theta_{h^*}}) = \max_{h\in[m]} g(p_{\theta_h}) = \max_{h\in[m]} \sum_{i\in K} \lambda_i \cdot g(p_{\theta_h}, i) = \max_{h\in[m]} \sum_{i\in K} \lambda_i \cdot \theta_h \cdot x^i_{j(h,i)}. \tag{4}$$

Observe that $g(p_{\theta_{h^*}})$ can be computed in $O(|K|n)$ time, by using the characterization provided in (4). So, we can define an algorithm that returns a UPF^* starting from X and $(C^i)_{i\in K}$, and that runs in $O(|K|n(\log(|K|)+\log(n)))$ time. We call this algorithm *Perfect-Single-Price* algorithm (PSP); it differs from the *Single-Price* algorithm given in [6] since the approximation ratio of the latter (which is worse than the one of PSP algorithm) and its complexity depend on an arbitrary ϵ given in input. An approximation ratio provided by this algorithm can be obtain by setting $W_m := x(1)$, $W_M := x(m)$, $\theta_m := \theta(m)$, $\theta_M := \theta(1)$.

To apply the PSP algorithm to the SFPP, we reformulate the SFPP as a game in SGPS. Given an instance $I := (G, K, \{\lambda_{uv}\}_{(u,v)\in K}, L, C, p_C, e)$ of the SFPP, define an instance $I' := (E', C', L', p'_C, w', \mathcal{E}, K, \{\lambda_{uv}\}_{(u,v)\in K}, \{\mathcal{F}_{uv}\}_{(u,v)\in K})$ in the class SGPS as follows. Let $E' = C' \cup L'$ be a set of edges in a graph (V', E') defined as follows: given $u \in L$, $v \in C$ and $z \in V$, create a node z'_{uv}, insert in L' an edge (u, z'_{uz}) with $w'((u, z'_{uz})) = d(u, z)$, insert in C' an edge (z'_{uz}, z) with $p'_C((z'_{uz}, z)) = e \cdot d(u, z)$, insert in C' an edge (v, z) with $p'_C((v, z)) = p_C(v) \cdot d(v, z)$. Given $u \in L$, set $E_u = \{(u, z'_{uz}) \in L'\}$ and $\mathcal{E} := \{E_u\}_{u\in L}$. Set, $\forall (u, v) \in K$, $\mathcal{F}_{uv}$ equal to the set of all the paths connecting u to v in (V', E'). Given two pricing functions p_L and p'_L for the problems I and I', respectively, such that $p_L(u) = p'_L(E_u) + e\ \forall u \in L$, we have that $g(p_L) = g(p'_L)$. Moreover, by assigning uniformly the prices θ and $\theta' := \theta - e$ to I and I', respectively, we obtain that the length of the sub-path covered by follower $(u, v) \in K$ by using fuel bought from the leader in I, is equal to $W(F^*_{uv}(p_{\theta'}))$. From these observations, if one transforms I into I' and applies the PSP algorithm to I', I can be approximated according to the performance guarantee stated in Theorem 4.

To apply the PSP algorithm, we need a preliminary procedure to find the vectors X and $(C^{uv})_{(u,v)\in K}$. Observe that we can set $X := \{d(u, v) > 0 : u \in L, v \in C \cup K_V\} := [x(h)]_{h\in[l]}$ to apply the PSP algorithm, since every follower, given an UPF, can always choose a path such that only a subpath is covered using fuel bought from the leader. Because of this fact, we also have that $C^{uv}(h) = \min\{c_\infty(u, z) + e \cdot d(z, z') + c_\infty(z', v) : d(z, z') = x(h), z \in L, z' \in C \cup \{v\}\}$ $\forall h \in [l]$ and $\forall (u, v) \in K$. Observe that, given $(u, v) \in K$, C_{uv} can be computed in $O(|L| \cdot |C|) \subseteq O(n^2)$ time. To approximate the SFPP, compute the vectors X and $(C_{uv})_{(u,v)\in K}$ and apply the PSP algorithm to these vectors. The resultant algorithm, that we call SFPP-PSP, requires $O(n^4)$ time in the worst-case.

We conclude by showing that the approximation ratio provided by the SFPP-PSP algorithm is tight and also significantly high. Consider an instance with $V = \{v_i\}_{i\in[n+1]}, E = \{(v_i, v_{i+1})\}_{i\in[n]}$, $C = L = V \setminus \{v_{n+1}\}$, $w((v_i, v_{i+1})) = 2^{i-n}$ and $p_C(v_i) = 2^{n-i}\ \forall i \in [n]$, $e = 0$, $K = \{(v_1, v_{n+1})\}$. Observe that the optimal assignment p^*_L verifies that $p^*_L(v_i) = p_C(v_i)\ \forall i \in [n-1]$ and so $g(p^*_L) = \sum_{i=1}^{n} 2^{i-n} \cdot 2^{n-i} = n$. Observe that an UPF^* is of the form $p_{(2^i)}$, with $i \in [n-1] \cup \{0\}$, and we have that $g(p_{(2^i)}) = \sum_{j=0}^{n-1-i} 2^{-j}$. Therefore $p_{(2^0)} = p_{(1)}$ is an UPF^* and $g(p_{(1)}) = \sum_{j=0}^{n-1} 2^{-j} \overset{n\to\infty}{\longrightarrow} 2$. Hence, the approximation ratio provided by the SFPP-PSP algorithm is asymptotically equal to $g(p^*)/g(p_{(1)}) = n/2$, and so it is very high. Since this instance verifies $1 + \ln(W_M/W_m) = 1 + \ln(\sum_{i=1}^{n} w((v_i, v_{i+1}))/w((v_1, v_2))) = 1 + \ln(\sum_{i=0}^{n-1} 2^i) = 1 + \ln(2^n - 1) \sim \ln(2^n) =$

$\ln(4) \cdot n/2 \simeq 1.386 \cdot n/2$ and $1 + \ln(\theta_M/\theta_m) = 1 + \ln(2^{n-1}) \sim \ln(4) \cdot n/2$, we have that the approximation ratio claimed in Theorem 3 for the SFPP may not be asymptotically improved. By using an instance of SFPP similar to the previous one, it is possible to prove that also the approximation ratio $1 + \ln(\lambda_M/\lambda_m)$ claimed in Theorem 4 is asymptotically tight.

References

1. P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. In *Proc. of the 3rd Italian Conference on Algorithms and Complexity (CIAC)*, LNCS 1203, Springer, pp 288–298, 1997.
2. D. Bilò, L. Gualà, S. Leucci, and G. Proietti. Specializations and generalizations of the Stackelberg minimum spanning tree game. In *Proc. of the 6th Workshop on Internet and Network Economics (WINE)*, LNCS 6484, Springer, pp. 75–86, 2010.
3. D. Bilò, L. Gualà, G. Proietti, and P. Widmayer. Computational aspects of a 2-player Stackelberg shortest paths tree game. In *Proc. of the 4th Workshop on Internet and Network Economics (WINE)*, LNCS 4858, Springer, pp. 251–262, 2008.
4. M. Bouhtou, A. Grigoriev, S. van Hoesel, A. van der Kraaij, F. Spieksma, and M. Uetz. Pricing bridges to cross a river. *Naval Research Logistics*, 54(4):411–420, 2007.
5. P. Briest, P. Chalermsook, S. Khanna, B. Laekhanukit, and D. Nanongkai. Improved hardness of approximation for Stackelberg shortest-path pricing. In *Proc. of the 6th Workshop on Internet and Network Economics (WINE)*, LNCS 6484, Springer, pp. 444–454, 2010.
6. P. Briest, M. Hoefer, and P. Krysta. Stackelberg network pricing games. *Algorithmica*, 62(3-4):733–753, 2012.
7. J. Cardinal, E. D. Demaine, S. Fiorini, G. Joret, S. Langerman, I. Newman, and O. Weimann. The Stackelberg minimum spanning tree game. *Algorithmica*, 59(2):129–144, 2011.
8. J. Cardinal, E. D. Demaine, S. Fiorini, G. Joret, I. Newman, and O. Weimann. The Stackelberg minimum spanning tree game on planar and bounded-treewidth graphs. *Journal of Combinatorial Optimization*, 25(1):19–46, 2013.
9. S. Khuller, A. Malekian, and J. Mestre. To fill or not to fill: The gas station problem. *ACM Transactions on Algorithms*, 7(3):36, 2011.
10. D. G. Kirkpatrick, and R. Seidel. The ultimate planar convex hull algorithm. *SIAM Journal on Computing*, 15(1):287–299, 1986.
11. 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.
12. S. Roch, G. Savard, and P. Marcotte. An approximation algorithm for Stackelberg network pricing. *Networks*, 46(1):57-67, 2005.
13. M. I. Shamos, and D. Hoey. Geometric intersection problems. In *Proc. of the 17th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE Computer Society, pp. 208–215, 1976.
14. 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.
15. H. von Stackelberg. *Marktform und Gleichgewicht (Market and Equilibrium)*. Verlag von Julius Springer, Vienna, 1934.