On Stackelberg Pricing with Computationally Bounded Customers*

Patrick Briest † Luciano Gualà ‡ Martin Hoefer § Carmine Ventre ¶

Abstract

In Stackelberg pricing a *leader* sets prices for items in order to maximize revenue from a *follower* purchasing a feasible subset of items. We consider computationally bounded followers who cannot optimize exactly over the range of all feasible subsets, but who apply publicly known algorithms to determine the items to purchase. This corresponds to general multi-dimensional pricing when customers cannot optimize their valuation functions efficiently but still aim to act rationally to the best of their ability.

We consider two versions of this novel type of pricing problem. In the Min-Knapsack variant items are weighted objects and the follower seeks to purchase a min-cost selection of objects of some bounded weight. When he uses a greedy 2-approximation algorithm, we provide a polynomial-time $(2+\varepsilon)$-approximation algorithm for the leader's revenue maximization problem based on so-called near-uniform price assignments. We also prove the problem to be strongly NP-hard.

In the Set-Cover variant items are subsets of some ground set which the follower seeks to cover. When he uses a standard primal-dual approach, we prove that exact revenue maximization is possible in polynomial time when elements have frequency 2 (Vertex-Cover variant). This stands in sharp contrast to APX-hardness for the problem with elements of frequency 3.

1 Introduction

The problem of *multi-dimensional pricing* consists of assigning revenue maximizing prices to a set of distinct items given information about the preferences of potential customers. A natural way to describe customer preferences is via *valuation functions* that map subsets of items to non-negative real numbers describing how much a set is valued by a certain customer. Given fixed item prices a rational customer acting according to *quasi-linear utilities* chooses to purchase a subset of items maximizing her *utility*, i.e., the difference between her value for the set and the sum of prices of items contained in it. It is known that general multi-dimensional pricing with unlimited supply of each item allows polynomial-time approximation algorithms achieving ratios that are logarithmic in the number of customers or linear in the number of distinct items via considering only *single-price solutions* [4, 10]. Lower bounds of the same order of magnitude have also been proven for approximation guarantees in both parameterizations [8, 18].

*This work appeared as an extended abstract in WINE 2009 [9].

†Department of Computer Science, University of Paderborn, Germany. E-mail: patrick.briest@upb.de.

‡Dipartimento di Matematica, Universitá di Roma "Tor Vergata", Italy. E-mail: guala@mat.uniroma2.it.

§Corresponding Author. Department of Computer Science, RWTH Aachen University, Germany. E-mail: mhoefer@cs.rwth-aachen.de.

¶Department of Computer Science, University of Liverpool, UK. E-mail: Carmine.Ventre@liverpool.ac.uk.

For the known algorithmic results it is sufficient that valuation functions can be accessed via *demand oracles*, i.e., customers are treated as black boxes that can answer the question: "Given some vector of prices, which set of items would you choose to buy?" But how much can it help us to have more detailed information about the structure of customer preferences? If the number of customers is large, unfortunately, the answer is "not at all" as follows readily from the known lower bounds, which hold for special cases of the problem in which the exact preferences can be elicited via demand queries. However, the situation is different when the number of distinct customers is small.

In a 2-player *Stackelberg Pricing Game*, named after the underlying market model due to Heinrich Freiherr von Stackelberg [32], we are given a collection of items, some of which have fixed-costs. A so-called *leader* may assign prices to the remaining items. A *follower* then purchases a *feasible* set of items of minimum cost and pays the leader for the priceable items in the set. This problem is equivalent to multi-dimensional pricing with a single follower if the follower's feasible sets are unrestricted. However, a standard assumption when considering Stackelberg games is that the follower has to be able to optimize in polynomial time over her feasible sets. As an example, think of items as edges in a graph and the follower's feasible sets as all possible paths connecting some vertices s and t. For some kinds of followers, e.g., buying a min-cost vertex cover in a bipartite graph, it has been shown that improved approximation guarantees are possible [10].

In this paper we initiate the study of a closely related question: What if the follower is unable to exactly optimize over her feasible sets, because the problem is computationally hard, but is still guaranteed to act rationally to the best of her ability? More formally, we will assume that when prices have been fixed the follower applies a publicly known approximation algorithm to find a near-optimal feasible set of items to purchase. This assumption is quite reasonable when followers are actually software agents with known implementation details. To the best of our knowledge, this is the first analysis of multi-dimensional pricing with follower preferences that are neither *single-minded* or *unit-demand*, nor expressible as exact optimization over the full domain of the valuation function. Before describing our results in detail, we review some important related work and introduce the notation used throughout the paper.

1.1 Related Work

Algorithmic aspects of multi-dimensional pricing problems, which are important in the context of pure optimization as well as the design of revenue-maximizing auction mechanisms [3], were first studied by Aggarwal et al. [1] and Guruswami et al. [24]. Subsequently, quite a number of improved algorithmic results for special cases of the problem [2, 12, 16, 17, 19, 20, 25] and complexity theoretic lower bounds [8, 13, 18] have been derived.

Our introductory example of shortest-path Stackelberg pricing was first introduced in the operations research literature. In particular, Labbé et al. [28] derived a bi-level LP formulation of the problem and proved NP-hardness. Subsequently, Roch et al. [29] presented a first polynomial-time approximation algorithm with a provable approximation guarantee of $O(\log m)$, where m is the number of priceable edges. The current best lower bound is inapproximability within $2 - o(1)$ shown by Briest and Khanna [11]. Initial results for a Stackelberg pricing game, in which the follower purchases a single-source shortest path tree, are presented in [7].

Cardinal et al. [15] investigate the corresponding minimum spanning tree game, proving that this version of the problem is APX-hard and that the very simple *single-price algorithm* achieves an approximation guarantee of $O(\log m)$ for m priceable edges. More recently, improved algorithms

for special cases like bounded-treewidth graphs [14] and other variants [6] have appeared. Finally, Briest et al. [10] extend the analysis of the single-price algorithm to Stackelberg pricing in general and prove that it yields a $O(\log m)$-approximation for m priceable items. This analysis is tight for the shortest-path and minimum spanning tree games.

Stackelberg pricing can also be considered with objectives other than revenue maximization. When prices are tolls on network arcs, the problem of congestion minimization has received considerable attention. Karakostas and Kolliopoulos [27], Fleischer, Jain and Mahdian [22], Fleischer [21] and Swamy [30] show how to efficiently compute a set of optimal tolls for all arcs, which enforce optimal routings as Nash equilibria for selfish network routing games. More closely related to Stackelberg pricing is the case when only a subset of arcs are taxable, which is considered by Hoefer et al. [26]. They show that finding optimal tolls for subnetworks is NP-hard.

1.2 Preliminaries

In this paper we consider games falling in the following general class. There are two players in the game, one *leader* and one *follower*. There is also a set of items $\mathcal{I}$ that is partitioned into fixed-cost items $\mathcal{F}$ and priceable items $\mathcal{P}$. Each fixed-cost item $i \in \mathcal{F}$ has a fixed-cost $c(i) \geq 0$. For each priceable item $i \in \mathcal{P}$ the leader can specify a price $p(i) \geq 0$. The follower has a set $\mathcal{S} \subset 2^{\mathcal{I}}$ of *feasible subsets* and is interested in buying some subset in $\mathcal{S}$. The cost of a subset $S \in \mathcal{S}$ is given by the cost of fixed-cost items and the price of priceable items:

$$cost(S) = \sum_{i \in S \cap \mathcal{F}} c(i) + \sum_{i \in S \cap \mathcal{P}} p(i) \ .$$

The *revenue* of the leader from subset S is given by the prices of the priceable items that are included in S, that is,

$$r(S) = \sum_{i \in S \cap \mathcal{P}} p(i) \ .$$

We let $S_A(p)$ be the feasible subset in $\mathcal{S}$ chosen by the follower when she uses polynomial-time algorithm A given prices p. Naturally, the follower would like A to return the minimum-cost subset in $\mathcal{S}$, but this could be an NP-hard task in general. We capture this intuition by making no assumption on optimality of the algorithm: A can return a suboptimal subset in $\mathcal{S}$. Our interest is to find a pricing function p^* for the leader that generates maximum revenue when the follower uses algorithm A, i.e.,

$$p^* \in \arg\max_p r(S_A(p)) \ .$$

We denote this maximum revenue by r^*. To guarantee that the revenue is bounded and the optimization problem is non-trivial, we assume that there is at least one feasible subset that is composed only of fixed-cost items and that the follower algorithm outputs it under certain circumstances. Towards this aim, we further assume that for each priceable item there is a threshold price above which no subset including it will be output by the follower algorithm. This last assumption holds for every follower algorithm with bounded approximation ratio.

In the above class of games, we will consider the Min-Knapsack pricing problem and the Set-Cover pricing problem. In the knapsack pricing problem, the set of items is a set of weighted objects $\mathcal{O}$. A subset of $\mathcal{O}$ is feasible if the total weight of the objects comprising it is at least a given bound W. We will refer to the revenue optimization problem for the knapsack pricing problem by

StackKP. In the set cover pricing problem, following our terminology, given some ground set to be covered, every item corresponds to a given subset of the ground set. A subset of items is feasible for the follower whenever it covers all the elements of the ground set. We will refer to the revenue optimization problem for the set cover pricing problem by StackSC and denote the special case of vertex cover pricing by StackVC.

1.3 Contributions and Outline

The focus of this paper are followers applying approximation algorithms that are (i) structurally simple and (ii) sufficiently suboptimal to ensure that revenue maximization has to take into account the algorithm's exact structure. We first consider StackKP and assume that the follower uses the well known greedy algorithm (see Section 2) to compute a 2-approximate solution to the minimization version of the knapsack problem she needs to solve. Even though structurally quite simple, this problem seems to capture many of the fundamental problems of Stackelberg pricing with computationally limited followers.

We show that in this case a careful adaptation of the known single-price strategy termed *near-uniform pricing*, which essentially assigns a single price to a subset of items and removes the remaining ones from the market by assigning a sufficiently high price (see below for a formal definition), can be used to approximate optimal revenue in most of the solution space. Adding some fairly standard enumeration techniques we are able to derive a polynomial-time $(2 + \varepsilon)$-approximation for the revenue maximization problem. The main technical difficulty lies in the fact that our analysis needs to argue about the exact computation done by the follower for a given price vector rather than using some global optimality condition. We point out that our algorithm is best possible among all algorithms based on near-uniform price assignments. Finally, we show that the revenue maximization problem in this setting is strongly NP-hard.

We then turn our attention to StackSC and assume that the follower is using the primal-dual schema based approximation algorithm (see Section 3) to find a selection of sets to purchase. We view the problem in its equivalent formulation of Vertex-Cover in hypergraphs and start by investigating the special case of regular Vertex-Cover in standard graphs. We prove that while near-uniform price assignments cannot achieve better than logarithmic approximation guarantees in this case, exploiting the algorithm's structure nevertheless allows for exact revenue maximization in polynomial time. Previously, it was shown that games with a follower purchasing a min-cost vertex cover in a bipartite graph in the special case that all priceable vertices are located on one side of the bipartition allows for polynomial-time revenue maximization [10]. It would be very interesting to see whether there is a deeper connection between these two problems.

Turning to general hypergraphs it turns out that revenue maximization (StackSC) becomes hard already with edges of cardinality 3 (or elements of frequency 3 in the Set-Cover view) and is APX-hard in general. This is quite surprising given that the follower's primal-dual algorithm achieves approximation guarantee f for any frequency f, i.e., the approximation complexity of the underlying problem scales quite smoothly. We also argue that in this general case neither near-uniform price vectors nor our algorithm from the Vertex-Cover case can guarantee any sub-exponential approximation ratio.

Finally, we conclude with interesting open problems for further research in Section 4.

1.4 Knowing the Follower's Algorithm

A central assumption in our analysis is that the leader has full knowledge of the follower's algorithm. This assumption might appear quite strong. It turns out, however, that this non-black-box attitude on the leader's side is necessary to achieve any reasonable approximation guarantees. Suppose the leader only knows the approximation guarantee of the follower's algorithm A and is given black-box-access to it, but no specific details are revealed. In this case it is easy to argue that for both StackKP and StackVC no algorithm can achieve a finite approximation guarantee.

Proposition 1. *For any constant $\rho > 1$, there are instances of* StackKP *and* StackVC *in which no leader's algorithm can achieve a finite approximation guarantee given only information about ρ and black-box-access to the follower's ρ-approximation algorithm.*

Proof. We present the argument only for StackVC, since it can be adjusted easily to hold for StackKP as well. Let $\rho = 1 + \varepsilon$, and consider a star of $n + 1$ vertices with center node c. Now assume that there is just one single priceable node $v \neq c$, and that any other node has cost 1. We consider a special ρ-approximation algorithm A to be used by the follower: pick c, and only if $p(v) = x$, also buy v, where $0 < x \leq \varepsilon$ is a fixed value. Clearly, A returns a ρ-approximation. If the leader does not know the value of x, the best thing she can do is to choose a value randomly. However, the probability of guessing the right value x is 0. Moreover, the optimal solution provides a revenue of x by pricing v at x and hence any pricing algorithm has an unbounded approximation ratio. □

Similarly, it is necessary to assume that the follower decides on the algorithm to be used in advance. If the follower is allowed to choose the algorithm (from a known set of alternatives) once the leader has set the prices, then an impossibility result similar to the one above applies.

2 Knapsack Pricing

In the Min-Knapsack problem we are given a set $\mathcal{O}$ of n *objects*, some of them with fixed cost and some priceable. Each object $o \in \mathcal{O}$ has weight $w(o) \in \mathbb{N}$ and we are given an integer weight bound W. Following the general framework given above, each subset X of $\mathcal{O}$ has a cost which is defined as the sum of the cost of the fixed-cost objects in X and the prices of the priceable objects in X. The follower wants to purchase a set of objects of minimum cost whose weight is at least W. We assume that the follower uses the standard greedy algorithm outlined below to find an approximation of such a minimum-cost set.

2.1 The Follower's Algorithm

An object's cost-efficiency (below referred to as efficiency for brevity) is defined as $\phi(o) = c(o)/w(o)$ or $\phi(o) = p(o)/w(o)$, depending on whether it is fixed-cost or priceable. Algorithm 1 below proceeds as follows. First, order all objects by non-decreasing efficiency (breaking ties by decreasing weight). Then add objects to the knapsack in this order. If an object makes the weight of the solution it completes at least W, remember this (feasible) solution and discard the object. Finally, return the best solution found. Note that we assume that ties are broken according to decreasing weight, i.e., larger objects are considered first given identical efficiency. This is a natural tie breaking rule, as it aims at minimizing the *overlap* of objects that exceed the knapsack's remaining capacity when they are considered.

Algorithm 1: The greedy approximation algorithm for MIN-KNAPSACK

```
Let o_1, o_2, ..., o_n be the objects ordered by non-decreasing efficiency, i.e.,
φ(o_1) ≤ ··· ≤ φ(o_n).
X ← Y ← ∅.
c_Y ← +∞.
for i = 1, ..., n do
    X ← X ∪ {o_i}.
    if w(X) ≥ W then
        if cost(X) < c_Y then
            Y ← X.
            c_Y ← cost(X).
        X ← X \ {o_i}.
Return Y.
```

2.2 Transforming the Optimal Solution

Let p^* be the optimal price assignment and let $\mathcal{P}^*$ be the set of priceable objects that are selected by Algorithm 1 given these prices.

The key ingredient to our approximation algorithm for knapsack pricing is the observation that price assignments that result in a large number of priceable objects being bought by the follower can be approximated by almost uniform price assignments at the expense of reducing overall revenue by no more than a constant factor.

Definition 1. *We call a price assignment p* near-uniform*, if there exists a single efficiency $\phi > 0$, such that $p(o) = w(o) \cdot \phi$ or $p(o) = +\infty$ for every $o \in \mathcal{P}$.*

We call an object b *blocking*, if the weight of the current solution exceeds W when it is added by the greedy algorithm. Let $\mathcal{B} = \{b_1, \ldots, b_l\}$ denote the blocking objects with $\phi(b_1) \leq \cdots \leq \phi(b_l)$. Since every blocking object corresponds to a unique solution checked by the greedy algorithm, our approach to relating the algorithm's behavior on two different price vectors is to relate the sets of blocking objects in both cases.

Thus, in order to prove Theorem 1 below we need to argue that the set of blocking objects does not change in completely uncontrolled ways when we transform optimal pricing p^* into a near-uniform pricing $\tilde{p}$. We will do so in two steps. Let $\tilde{\phi}$ denote the single efficiency in the near-uniform pricing $\tilde{p}$. We will first argue that blocking objects b_j with $\phi(b_j) < \tilde{\phi}$ are the same with respect to both p^* and $\tilde{p}$. We then show that even though blocking objects b_j with $\phi(b_j) \geq \tilde{\phi}$ might change, we can guarantee that the cheapest solution found contains most of our priceable objects.

Theorem 1. *Let p^* be the optimal price assignment for a given* STACKKP *instance and let r^* be the resulting revenue. Furthermore, assume that given prices p^* the follower purchases at least $k \in \mathbb{N}$ priceable objects, i.e., $|\mathcal{P}^*| \geq k$. Then there exists a near-uniform price assignment $\tilde{p}$ with revenue at least $r^*(k-1)/(2k)$.*

Proof. Define $w^* = \sum_{o \in \mathcal{P}^*} w(o)$, $r^* = \sum_{o \in \mathcal{P}^*} p^*(o)$ and let $\phi^*_{ave} = r^*/w^*$. Let c^* denote the total cost of the solution bought by the follower given prices p^*. We define a near-uniform price

assignment $\tilde{p}$ by

$$\tilde{p}(o) = \begin{cases} (1/2) \cdot w(o) \cdot \phi^*_{ave} & \text{for all } o \in \mathcal{P}^*, \\ +\infty & \text{otherwise.} \end{cases}$$

There is a one-to-one correspondence between the blocking objects $\mathcal{B} = \{b_1, \ldots, b_l\}$ given prices p^* and solutions checked by the greedy algorithm. Because of the fact that blocking objects do not influence the set of solutions checked by the greedy algorithm (beyond the one they belong to themselves), it is w.l.o.g. to assume that there is at most a single priceable blocking object given prices p^*, and if it exists it belongs to $\mathcal{P}^*$.

Proving the claimed revenue guarantee for $\tilde{p}$ consists of two parts. First, we show that with prices $\tilde{p}$ the blocking objects of efficiency less than $\phi^*_{ave}/2$ are still blocking with prices $\tilde{p}$ and their corresponding solutions have cost greater than $c^* - (r^*/2)$. We then show that among the solutions with blocking objects of efficiency greater than or equal to $\phi^*_{ave}/2$ there exist some with cost at most $c^* - (r^*/2)$ and the cheapest among these contains priceable objects of value at least $r^*(k-1)/(2k)$.

We first deal with blocking objects with efficiency $\phi(b_j) < \phi^*_{ave}/2$ and argue that they remain blocking. Consider b_j with $\phi(b_j) = c(b_j)/w(b_j) < \phi^*_{ave}/2$. Since we have at most one blocking priceable object (of efficiency at least ϕ^*_{ave}), b_j must be fixed-cost. Let $w_{<j}$ and $r_{<j}$ denote the weight and total price of priceable objects in the knapsack before object b_j is considered. Similarly, let $w_{>j}$ and $r_{>j}$ denote the weight and price of priceable objects from $\mathcal{P}^*$ that are considered after b_j. Let $\phi_{<j} = r_{<j}/w_{<j}$ and $\phi_{>j} = r_{>j}/w_{>j}$ be these objects' average efficiencies. This situation is depicted in Fig. 1.

Using $w_{<j} < w^*$ and $\phi_{<j} < \phi^*_{ave}/2$, we obtain

$$r^* = w_{<j}\phi_{<j} + w_{>j}\phi_{>j} < w^*(\phi^*_{ave}/2) + w_{>j}\phi_{>j} \ ,$$

and rearranging yields

$$\begin{aligned} r_{>j} = w_{>j}\phi_{>j} &> r^* - w^*(\phi^*_{ave}/2) \\ &= w^*(\phi^*_{ave}/2) > w^*\phi(b_j) \ , \end{aligned}$$

since $\phi(b_j) < \phi^*_{ave}/2$ by assumption. Finally, as the best solution found by the follower's greedy algorithm does not contain b_j but additional priceable objects of total price $r_{>j}$, we may write

$$w(b_j)\phi(b_j) = c(b_j) \geq r_{>j} > w^*\phi(b_j) \ ,$$

and so we conclude that $w(b_j) > w^*$. In other words, even if we removed *all* priceable objects from the solution, b_j would still remain blocking.

The fact that blocking objects of efficiency less than $\phi^*_{ave}/2$ remain blocking also implies that no non-blocking objects with efficiency below $\phi^*_{ave}/2$ can become blocking, since under prices $\tilde{p}$ the knapsack contains less total weight when such an object is considered. Also, for given prices $\tilde{p}$, the fact that $r_{<j} < r^*/2$ implies that every solution found by the greedy algorithm with a blocking object b_j of efficiency less than $\phi^*_{ave}/2$ has total cost greater than $c^* - (r^*/2)$.

We continue by considering blocking objects with efficiency $\phi(b_j) \geq \phi^*_{ave}/2$. If $\mathcal{P}^*$ did not contain a blocking object given prices p^*, then changing prices to $\tilde{p}$ does not change the set of blocking objects of efficiency at least $\phi^*_{ave}/2$. To see this, note that with prices $\tilde{p}$ the knapsack's remaining capacity is smaller than before at any point, so no previously blocking object can become non-blocking. On the other hand, all the non-blocking objects left enough room to pack the objects

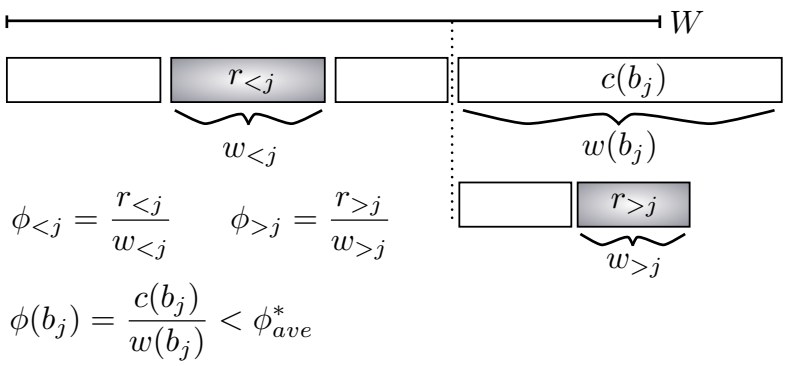


Figure 1: Situation from the first part of the proof of Theorem 1. Shaded blocks depict priceable objects that are selected by Algorithm 1 as part of the eventually cheapest solution before and after blocking object b_j is considered, respectively.

from $\mathcal{P}^*$, so changing the order in which they are considered cannot create new blocking objects either. It follows that all solutions found by the greedy algorithm with blocking objects of efficiency at least $\phi^*_{ave}/2$ contain priceable objects with a total price of $r^*/2$. The previously cheapest solution is still found and has cost $c^* - (r^*/2)$.

If $\mathcal{P}^*$ did contain a blocking element with prices p^*, but does not given prices $\tilde{p}$, we consider two cases. If the previously optimal solution is still found by the greedy algorithm, the same argumentation as above guarantees revenue $r^*/2$. If it is not, it must be the case that some previously non-blocking element has now become blocking. Consider the first such element b_j. Since all objects in the knapsack at the time b_j is considered and b_j itself were part of the cheapest solution given prices p^*, we have again found a solution of total cost at most $c^* - (r^*/2)$. Since all priceable objects are in the knapsack at this point and remain in it, a lower bound of $r^*/2$ on the revenue follows immediately.

Finally, assume that $\mathcal{P}^*$ contains a blocking element with both prices p^* and $\tilde{p}$. In this case, the solution with a priceable blocking object is guaranteed to have cost at most $c^* - (r^*/2)$. By the algorithm's tie-breaking rule every solution with a blocking object of efficiency larger than $\phi^*_{ave}/2$ found at a later time contains all but the smallest object from $\mathcal{P}^*$. By the fact that $|\mathcal{P}^*| \geq k$, it contains priceable objects of total value at least $(1 - 1/k) \cdot w^* \cdot \phi^*_{ave}/2 = r^*(k-1)/(2k)$. □

2.3 Approximation Results for Revenue Maximization

In this section we present our $(2+\varepsilon)$-approximate algorithm for StackKP. Algorithm 2 proceeds in two stages. First it checks for some given constant $k \in \mathbb{N}$ all possible price assignments to sets of at most $k-1$ priceable objects. Then for each possible weight w, it finds a set with k or more priceable objects with total weight (roughly) w (if such a set exists), and considers all near-uniform price assignments to that set.

Note that in both stages, checking all possible efficiencies cannot be done in polynomial time. Instead we restrict our attention to the efficiencies of the fixed-cost objects plus all powers of $(1+\delta)$ for some $\delta > 0$ to guarantee that we efficiently find a near-optimal price assignment in which all objects are considered by the greedy algorithm in the right order. Similarly, in the second stage we

Algorithm 2: A $(2+\varepsilon)$-approximation algorithm for StackKP

Let 1 be the minimum, Φ the maximum efficiency of fixed-cost objects, W the knapsack capacity, $\sqrt{1+\varepsilon/2}-1 \geq \delta > 0$, $r_{max} \leftarrow 0$.
Choose $k \geq (2(2+\varepsilon))/(2+\varepsilon-2(1+\delta)^2)$ and let
$\Lambda = \{\phi(o) \mid o \in \mathcal{F}\} \cup \{(1+\delta)^j \mid j = 0, \ldots, \lceil \log_{1+\delta} \Phi \rceil\}$.
foreach *set* $S \subseteq \mathcal{P}$ *of priceable objects with* $|S| \leq k-1$ **do**
  foreach *price assignment* p *with* $p(o)/w(o) \in \Lambda$ *for all* $o \in S$ *and* $p(o) = +\infty$ *else* **do**
    Let r be the resulting revenue.
    $r_{max} \leftarrow \max\{r_{max}, r\}$.
foreach $0 \leq i \leq \lceil \log_{1+\delta} \Phi \rceil$ **do**
  Let $\phi_i = (1+\delta)^i$.
  foreach $0 \leq j \leq \lceil \log_{1+\delta} W \rceil$ **do**
    Let S be a set of at least k priceable objects with total weight between $(1+\delta)^j$ and $(1+\delta)^{j+1}$, if such set exists, else $S = \emptyset$.
    Set $p(o) = w(o)\phi_i$ for all $o \in S$ and $p(o) = +\infty$ for all other priceable objects. Let r be the resulting revenue.
    $r_{max} \leftarrow \max\{r_{max}, r\}$.
Return r_{max}.

cannot enumerate all possible weights for the objects in our near-uniform price assignments, but again have to settle for powers of $(1+\delta)$.

Theorem 2. *Algorithm 2 computes in polynomial time a* $(2+\varepsilon)$*-approximation for* StackKP.

Proof. As outlined above, the algorithm proceeds in two stages. In the first stage it checks the revenue from all possible price assignments of the following form: A total number of at most $k-1$ objects are assigned prices, such that their respective efficiency is in the set Λ as defined in the algorithm. All other objects are assigned price $+\infty$. Note, that Λ consists of all powers of $1+\delta$ between 1 and Φ and the efficiencies of the fixed price objects $\mathcal{F}$.

Assume now that the optimal price assignment p^* results in priceable objects $\mathcal{P}^*$ being bought by the follower and $|\mathcal{P}^*| \leq k-1$. W.l.o.g. we may assume that $p^*(o) = +\infty$ for all $o \notin \mathcal{P}^*$. Consider then the price assignment p' defined as

$$p'(o) = w(o) \cdot \max\left([0, p^*(o)/w(o)] \cap \Lambda\right)$$

for $o \in \mathcal{P}^*$ and $p'(o) = +\infty$ else, which is considered by our algorithm. It is straightforward to argue that the solutions checked by the follower's greedy algorithm are identical under prices p^* and p', since all objects are considered in the exact same order. The cheapest solution under prices p' is the same as under p^* and the total revenue is reduced by a factor of at most $(1+\delta)^{-1}$. Thus, we obtain a $(1+\delta)$-approximation whenever the optimal solution consists of selling at most k priceable objects.

The second stage of the algorithm consists of approximating optimal solutions with more than k priceable objects by good near-uniform price assignments, which are guaranteed to exist by Theorem 1. Consider again p^* and $\mathcal{P}^*$ and assume that $|\mathcal{P}^*| \geq k$. Then by Theorem 1 there exists

a near-uniform price assignment $\tilde{p}$ with $\tilde{p}(o) = \phi w(o)$ for all $o \in \tilde{S}$ and $\tilde{p}(o) = +\infty$ for $o \notin \tilde{S}$ for some appropriately chosen $\tilde{S}$ and ϕ. The resulting revenue is at least $r^*(k-1)/(2k)$. We may w.l.o.g. assume that all priceable objects with a finite price are actually bought by the follower.

Let now $S \subset \mathcal{P}$ be a set of priceable objects with $|S| \geq k$ and

$$\sum_{o \in \tilde{S}} w(o) \leq \sum_{o \in S} w(o) \leq (1+\delta) \cdot \sum_{o \in \tilde{S}} w(o).$$

Such a set is computed in line 11 by Algorithm 2. Consider the price assignment p' defined as

$$p'(o) = w(o) \cdot (1+\delta)^{\lfloor \log_{1+\delta} \phi \rfloor - 1}$$

for $o \in S$ and $p'(o) = +\infty$ else, which is considered by the algorithm. Given prices p' the follower's greedy algorithm considers all priceable objects at an earlier time as with prices $\tilde{p}$. Let $\tilde{c}$ be the total cost of fixed-price objects bought by the follower given prices $\tilde{p}$. Using the same arguments as in the proof of Theorem 1 it follows that looking only at objects with efficiency at most ϕ the greedy algorithm finds a solution with total cost

$$\begin{aligned}
& \tilde{c} + \left(\sum_{o \in S} w(o) \right) \cdot (1+\delta)^{\lfloor \log_{1+\delta} \phi \rfloor - 1} \\
\leq \; & \tilde{c} + \left((1+\delta) \cdot \sum_{o \in \tilde{S}} w(o) \right) \cdot ((1+\delta)^{-1} \cdot \phi) \\
\leq \; & \tilde{c} + \frac{1+\delta}{1+\delta} \cdot \left(\sum_{o \in \tilde{S}} w(o) \right) \cdot \phi \\
= \; & \tilde{c} + \left(\sum_{o \in \tilde{S}} w(o) \right) \cdot \phi \;\; ,
\end{aligned}$$

which is exactly the cost of the cheapest solution found by the follower under price assignment $\tilde{p}$. Any solution found before the priceable objects are considered was not optimal under prices p^* and, thus, is still not optimal. Every solution found at a later time contains the $k-1$ largest objects from S and, consequently, carries a $(1-1/k)$-fraction of the revenue. Hence, the overall revenue r from price assignment p' is at least

$$r \geq \left(1 - \frac{1}{k}\right) \cdot (1+\delta)^{-2} \cdot \frac{k-1}{2k} \cdot r^* = r^* \cdot \frac{(k-1)^2}{2(1+\delta)^2 k^2} \;\; .$$

For $\delta \leq \sqrt{1+\varepsilon/2} - 1$ and $k \geq (2(2+\varepsilon))/(2+\varepsilon-2(1+\delta)^2)$ the approximation guarantee of $2+\varepsilon$ follows.

It remains to argue about the algorithm's running time. In the first stage the algorithm tests a total number of at most $|\mathcal{P}|^k |\Lambda|^k = |\mathcal{P}|^k (|\mathcal{F}| + \log \Phi)^k$ price assignments, which is polynomially bounded for any constant value of k. In the second stage a total of $(\log W)(\log \Phi)$ price assignments are tested. We only need to argue that we can compute a set S of at least k priceable objects and total weight between $(1+\delta)^j$ and $(1+\delta)^{j+1}$ in polynomial time, if such a set exists.

This can be done by a straightforward extension of the dynamic programming approach for the knapsack problem. Let n objects with weights $w_1, \ldots, w_n$ be given. By $A(j, \ell, w)$ we denote the "cost" of any selection of at least ℓ of the first j objects with total weight exactly w. We set $A(0, 0, 0) = 0$, $A(0, \ell, w) = +\infty$ for all $\ell, w \neq 0$ and use the recurrence

$$A(j, \ell, w) = \min\{A(j-1, \ell-1, w-w_j), A(j-1, \ell, w)\} \ .$$

A selection of at least k objects and total weight w exists if and only if $A(n, k, w) = 0$. Applying standard rounding techniques to the weights of the objects in the pricing instance we obtain the desired result. □

Note that the above analysis is essentially tight. Consider the following StackKP instance with knapsack capacity W. There is a single fixed-price object with size W and price W. There are $W-1$ priceable objects of size 1 and one priceable object of size W. By setting the prices of all small priceable objects to $1 - \varepsilon/W$ and the price of the large one to $W - \varepsilon/W$ we obtain revenue of $2W - \varepsilon$ for arbitrary $\varepsilon > 0$, since in this case the greedy algorithm starts by fitting all the small priceable objects into the knapsack and then adds the large priceable object to obtain a solution.

Consider then any pricing that assigns prices other than $+\infty$ to at most some constant number $k \ll W$ of objects. If the priceable object of size W is among these, total revenue is at most $W + k - 1$, otherwise it is at most k. Finally, consider any near-uniform pricing. To obtain any revenue at all, all priceable objects with price other than $+\infty$ must have efficiency at most 1. If the large priceable object is among these objects, it is considered first due to the tie breaking rule and, thus, revenue is at most W. If it is not, revenue is clearly at most $W-1$. Hence, the approximation guarantee achieved by Algorithm 2 is bounded below by $(2W - \varepsilon)/(W + k - 1) \to 2$ for $W \to +\infty$.

Observe that by choosing $k = 2$ in Algorithm 2, we obtain a constant factor approximation by considering only near-uniform price assignments, since in this case the first step of the algorithm considers price assignments to sets of size 1, which must trivially be uniform.

Finally, it turns out that revenue maximization against a follower using Algorithm 1 is strongly NP-hard.

Theorem 3. StackKP *is strongly* NP*-hard.*

Proof. We prove the result by a reduction from the 3-Partition problem, which is known to be strongly NP-complete [23]. Given a collection of integer weights $w_1, \ldots, w_n \in \mathbb{N}$ with $n = 3m$, $\sum_{i \in [n]} w_i = W$ and $W/(4m) < w_i < W/(2m)$, we want to decide whether the weights can be partitioned into subsets $S_1 \cup \cdots \cup S_m = [n]$, such that $\sum_{i \in S_j} w_i = W/m$ for $1 \leq j \leq m$ where, of course, W is a multiple of m.

We construct a knapsack pricing instance as follows. The knapsack has capacity W. For each weight w_i in the 3-Partition instance we have a priceable object with size w_i. There are two types of fixed-cost objects. First, we have an object with size $(1-j/m)W+1$ and cost $(1-j/m)W+1+1/W$ for $1 \leq j \leq m-1$. We denote these fixed-cost objects of *type A* as $A_1, \ldots, A_{m-1}$ and their respective efficiencies as $\phi_1, \ldots, \phi_{m-1}$. Then there are W/m additional fixed-cost objects, each with size 1 and cost $1 + 1/W$. We refer to these as fixed-cost objects of *type B* and denote their efficiency as ϕ_m. Observe that even by multiplying all weights and costs above by mW, all numbers in the instance are integers of size $O(mW^2)$. The 3-Partition problem remains hard even for instances in which m and W^2 have polynomial size.

We start with a simple observation about our pricing instance. First, note that the greedy algorithm considers fixed-cost objects $A_1, \ldots, A_{m-1}$ in this order before objects of type B. By a

straightforward inductive argument it follows that after the greedy algorithm has considered object A_j, the knapsack contains objects of total size at least jW/m and maximum efficiency ϕ_j. In particular, this implies that after A_{m-1} has been considered the remaining capacity is at most $W-(m-1)W/m = W/m$ and, thus, when fixed-cost objects of type B are considered the greedy algorithm finds a solution with total weight exactly W, at which point it stops. This solution has total cost at most $W(1+1/W) = W+1$ by the fact that fixed-cost objects of type B have efficiency $\phi_m = 1+1/W$. Note that such a solution is found independently of the prices assigned to the priceable objects.

We first argue about soundness. Assume that a partitioning $S_1 \cup \cdots \cup S_m = [n]$ exists. We define a corresponding pricing by assigning efficiency $\phi_j - \varepsilon$, for some carefully chosen $\varepsilon > 0$ (specified below), to the objects in S_j. Thus, because of the way the follower's greedy algorithm works we have that priceable objects are considered by the greedy algorithm before fixed-cost objects of efficiency ϕ_j. It is easy to check that given this price assignment none of the fixed-cost objects will fit into the knapsack and the algorithm finds a solution with total size exactly W that consists of selecting all the priceable objects and has total price less than $W+1$. Every other solution found by the greedy algorithm has size at least $W+1$ and, since all fixed-cost objects have efficiency at least ϕ_1, costs at least $(W+1)\phi_1 > W+1$. It follows that the priceable objects form the cheapest solution. Define $\phi_0 = 0$. The total revenue from the above price assignment is

$$\begin{aligned} r^* &= \sum_{k=1}^{m} \frac{W}{m} \cdot (\phi_k - \varepsilon) \\ &= \frac{W}{m} \cdot \sum_{k=1}^{m} \sum_{j=0}^{k-1} (\phi_{j+1} - \phi_j) - W\varepsilon \\ &= \sum_{j=0}^{m-1} \left(W - \frac{jW}{m} \right) \cdot (\phi_{j+1} - \phi_j) - W\varepsilon \ . \end{aligned}$$

Assume then that a partitioning of the weights does not exist and consider an arbitrary price assignment p. We are going to bound its revenue r. If the cheapest solution found by the greedy algorithm contains a fixed-cost object, total revenue is bounded above by $W+1-(1+1/W) < W < r^*$. This follows by our initial observation that the greedy algorithm always finds a solution with total cost at most $W+1$ and the cheapest fixed-price object has cost $1+1/W$. Thus, any price assignment with high revenue must ensure that no fixed-cost object becomes part of the cheapest solution found by the greedy algorithm.

For the given price assignment p let $\pi(1), \ldots, \pi(n)$ be an ordering of the priceable objects according to increasing efficiency, i.e., $p_{\pi(1)}/w_{\pi(1)} \leq \cdots \leq p_{\pi(n)}/w_{\pi(n)}$. Let $\ell(j)$ be the smallest index, such that $p_{\pi(\ell(j))}/w_{\pi(\ell(j))} > \phi_j$. Denote by

$$\xi_j = \sum_{i=1}^{\ell(j)-1} w_{\pi(i)}$$

the total weight of priceable objects with efficiency at most ϕ_j and define $\xi_0 = 0$. By the fact that no fixed-cost object can fit into the knapsack it follows that $\xi_j \geq jW/m$ for all j. We define

$\Delta_j = \xi_j - jW/m$. The revenue r of price assignment p can now be bounded above by

$$\begin{aligned} r &\leq \sum_{j=1}^{m} (\xi_j - \xi_{j-1}) \cdot \phi_j \\ &= \sum_{j=0}^{m-1} (W - \xi_j) \cdot (\phi_{j+1} - \phi_j) \\ &= \sum_{j=0}^{m-1} \left(W - \frac{jW}{m} - \Delta_j \right) \cdot (\phi_{j+1} - \phi_j) \\ &= r^* + W\varepsilon - \sum_{j=0}^{m-1} \Delta_j \cdot (\phi_{j+1} - \phi_j) \ . \end{aligned}$$

Now, the fact that no partitioning of the priceable objects into m subsets of identical size exists implies that for any price assignment there must exist at least one index j, such that $\Delta_j > 0$. But as such a Δ_j is defined as the difference of two integers, we have $\Delta_j \geq 1$. On the other hand, it easy to see that for all j we have $\phi_{j+1} - \phi_j \geq 1/(m+W)$. Therefore, we set ε such that $W\varepsilon < 1/(m+W)$. This completes the proof of completeness. □

3 Set Cover Pricing

In this section we consider the vertex and set cover pricing problems. In the simplest case the follower wants to purchase a minimum vertex cover of a graph $G = (V, E)$. Let $V = \mathcal{P} \cup \mathcal{F}$, where as above $\mathcal{F}$ and $\mathcal{P}$ denote the set of fixed-cost and priceable vertices respectively. More generally, for set cover we consider an equivalent formulation of vertex cover in hypergraphs. Namely, given the universe and its subsets of the set cover instance, we define a hypergraph where the vertices are the universe subsets and hyperedges are the elements of the universe. In this case there can be hyperedges in E connecting more than two vertices. We assume that the follower uses the standard primal-dual algorithm to find an approximation to the minimum vertex cover.

3.1 The Follower's Algorithm

The algorithm iteratively considers uncovered edges and raises budgets at the incident vertices until (at least) one of the vertices becomes tight. The algorithm is given for the case of regular vertex cover as Algorithm 3; for more details see [31, Chapter 15].

3.2 Revenue Maximization for Vertex Cover

At first, let us consider the approximation ratio of single-price and near-uniform price assignments. For the latter, as from Definition 1, we have a $\phi \in \mathbb{R}_0^+$ and $p(v) \in \{\phi, \infty\}$ for all $v \in \mathcal{P}$. It turns out that for these pricings there is a simple logarithmic lower bound.

Theorem 4. *Single-price and near-uniform price assignments yield an approximation factor of* $\Omega(\log n)$ *for* StackVC*, where* n *is the number of priceable vertices.*

Algorithm 3: Primal-dual algorithm of the follower

$\gamma_e \leftarrow 0$ for all edges $e \in E$
For all $v \in V$ let $c'(v) \leftarrow c(v)$ if $v \in \mathcal{F}$ and $c'(v) \leftarrow p(v)$ otherwise
Fix an order of edges $e_1, \ldots, e_m$
for $i = 1, \ldots, m$ **do**
  Let $e_i = (u, v)$ and $\sigma_u \leftarrow c'(u) - \sum_{e \,:\, e=(u,v')} \gamma_e$
  $\sigma_v \leftarrow c'(v) - \sum_{e \,:\, e=(u',v)} \gamma_e$
  $\gamma_e \leftarrow \min(\sigma_u, \sigma_v)$
Add every vertex v with $c'(v) = \sum_{e \,:\, e=(u,v)} \gamma_e$ to the cover.

Proof. The instance establishing the lower bound consists of n components of single edges. Each edge is connected to exactly one fixed-cost and one priceable vertex. The fixed costs resemble the harmonic sequence, i.e., 1, 1/2, 1/3,... It is obvious that any single price can only achieve a revenue of 1, while the optimum revenue is $H_n = \Theta(\log n)$. This argument holds also for near-uniform pricings. □

Instead, we present a natural greedy algorithm to compute optimal prices for the seller. It simulates a run of the primal-dual algorithm and raises prices of the priceable vertices in the same manner as the dual budgets γ are raised by the follower. In this way the algorithm greedily tries to sell a vertex to the follower as soon as she is willing to pay for it.

Algorithm 4: Greedy pricing for StackVC

$\gamma_e \leftarrow 0$ for all edges $e \in E$
for *each edge* $e = e_1, e_2, \ldots, e_m$ *in the order of the follower* **do**
  if $e = (u, v)$ *is incident to a priceable vertex* v **then**
    $\gamma_e \leftarrow c(u) - \sum_{e' : e'=(u,v')} \gamma_{e'}$
    $p(v) \leftarrow \sum_{e' : e'=(u',v)} \gamma_{e'}$
  else
    $\sigma_u \leftarrow c(u) - \sum_{e' : e'=(u,v')} \gamma_{e'}$
    $\sigma_v \leftarrow c(v) - \sum_{e' : e'=(u',v)} \gamma_{e'}$
    $\gamma_e \leftarrow \min(\sigma_u, \sigma_v)$

Theorem 5. *Algorithm 4 solves* StackVC *in polynomial time.*

Proof. Consider an optimum pricing p^*, which yields a strictly larger revenue than the *greedy pricing* p_g computed with our algorithm. For such a given set of prices p^*, we denote by γ_e^* the dual contribution of edge e, which we call the *budget* of edge e resulting from p^*. This contribution is the result of applying the follower's algorithm to the instance using p^*.

The heart of our proof is an inductive argument that one can transform any pricing (including p^*) by iteratively adjusting the prices without decreasing revenue, such that the budget of every edge is equal to the one resulting from the greedy pricing p_g. In particular, this implies that p_g is an optimal pricing.

We begin by restricting our attention to an optimal pricing p^* for which the follower purchases all priceable vertices. The existence of such a pricing follows from the next lemma.

Lemma 1. *Given any pricing p, there is a pricing p_l with $p_l(v) \leq p(v)$ for all $v \in \mathcal{P}$, for which the leader obtains at least as much revenue as for p and the follower purchases every vertex $v \in \mathcal{P}$.*

Proof. Consider any pricing p for which at least one vertex v is not bought by the follower. Note that this is equivalent to $p(v) > \sum_{e=(u,v)} \gamma_e$, where the γ_e are the respective budgets computed by the follower. If we lower $p_l(v) = \sum_{e=(u,v)} \gamma_e$, the execution of the follower algorithm is not affected and all the budgets remain the same. This means that the follower purchases exactly the same priceable vertices as before and in addition vertex v. This can only increase the revenue. Thus, by reducing prices to the sum of incident budgets, it is possible to construct a pricing p_l with at least as much revenue, for which all priceable vertices are bought by the follower. □

Now consider the pricings p^* and p_g and the resulting budgets γ^* and γ. Consider the smallest i', for which edge $e' = e_{i'} = (u, v')$ has $\gamma^*_{e'} \neq \gamma_{e'}$. It is easy to note that this edge must be incident to a priceable vertex v', and the difference in budgets is a result of setting different prices. By our algorithm and Lemma 1 we know that in both p^* and p_g all vertices are bought by the follower. This implies that $p^*(v) < p_g(v)$, and hence $\gamma^*_{e'} < \gamma_{e'}$. We now compare the revenue of pricing p^* to a pricing p' with $p'(v) = p^*(v)$ for every vertex $v \neq v'$, and for which $p'(v') = p^*(v') + \gamma_{e'} - \gamma^*_{e'}$. In p' the budgets γ'_e are equivalent to γ_e from p_g for every edge $e_1, \ldots, e_{i'-1}$ and also $e_{i'} = e'$.

We call $\delta^j(v) = \sum_{e_i : e_i = (v,u), i \leq j} \gamma'_{e_i} - \gamma^*_{e_i}$ the *reservation* that is created by p^* at vertex v at the end of processing edge e_j. The budget of e' is raised to a smaller amount in p^* than in p', so after processing e' there is positive reservation at the other endpoint u of e', i.e., $\delta^{i'}(u) = \delta^{i'}(v')$ at vertex u. No other vertex except u and v' has reservation at this point, so $\sum_{v \neq v'} |\delta^{i'}(v)| \leq \delta^{i'}(v')$. This will be our invariant, and in the following we prove it holds for the remaining edges $j > i'$ and the remaining iterations of the algorithm with pricing p'.

Lemma 2. *For any iteration $j \geq i'$ we have $\sum_{v \neq v'} |\delta^j(v)| \leq \delta^{i'}(v')$.*

Proof. At first, notice that the reservation of vertex v' satisfies $\delta^j(v') = \delta^{i'}(v')$ for $j \geq i'$. It will remain unaltered during the remaining iterations of the algorithm, because the vertex becomes tight in both runs at the time when considering edge e'. In both cases $\gamma'_{e_j} = \gamma^*_{e_j} = 0$ for any later edge e_j with $j > i'$, which is incident to v'. For the rest of the proof we consider several cases, depending on the existing reservation at the incident vertices.

Case 1: Consider an edge $e_j = (u, v)$ with $j > i'$ such that no incident vertex has reservation. In this case, the budget $\gamma'_{e_j} = \gamma^*_{e_j}$, thus the invariant continues to hold.

Case 2: Consider an edge $e_j = (u, v)$ with $j > i'$, for which vertex u has positive reservation $\delta^{j-1}(u) > 0$ and vertex v has 0 reservation. If v is the first vertex in both iterations to become tight, then $\gamma'_{e_j} = \gamma^*_{e_j}$ and we do not change the reservation at all. If u is the first vertex to become tight for p^*, then u is still the first vertex to become tight for p', because of positive reservation. In this case the budget γ'_{e_j} can be raised to a smaller amount, and we have $\gamma'_{e_j} = \gamma^*_{e_j} - \delta^{j-1}(u)$, $\delta^j(u) = 0$, and $\delta^j(v) = -\delta^{j-1}(u)$. The total absolute value of all reservations remains the same. Now consider the case if v is the first vertex to become tight for p^*, and u becomes tight for p'. Without loss of generality we assume u and v are

fixed-price vertices. Then for the budget

$$\begin{aligned}\gamma'_{e_j} &= c(u) - \sum_{e\in E, e=(u,\cdot)\neq e_j} \gamma'_e \\ &= \left(c(u) - \sum_{e\in E, e=(u,\cdot)\neq e_j} \gamma^*_e\right) - \delta^{j-1}(u) \\ &\geq \gamma^*_{e_j} - \delta^{j-1}(u)\end{aligned}$$

holds. For the $\gamma^*_{e_j}$ we observe

$$\gamma^*_{e_j} = c(v) - \sum_{e\in E, e=(v,\cdot)\neq e_j} \gamma^*_e \;\geq\; \gamma'_{e_j}.$$

The reservation removed at u is $\varepsilon = \gamma^*_{e_j} - \gamma'_{e_j} \leq \delta^{j-1}(u)$, while the reservation created at v is $-\varepsilon$. The total absolute reservation remains the same. The same argument holds with reversed sign if one vertex has negative reservation while the other has no reservation. The invariant is always preserved.

Case 3: Consider an edge $e_j = (u, v)$ with $j > i'$, for which vertex u has positive reservation $\delta^{j-1}(u) > 0$ and vertex v has negative reservation $\delta^{j-1}(v) < 0$. In this case, we can combine the arguments made above. If in both cases v becomes tight first, we add reservation on both vertices, however at most $|\delta^{j-1}(v)|$. If in both cases u becomes tight first, we subtract reservation from both vertices, but at most $\delta^{j-1}(u)$. Similarly, for any other combination, we add or subtract at most the absolute values of the corresponding reservations, which keeps the invariant.

Case 4: Finally, consider an edge $e_j = (u, w)$ with $j > i'$, for which the vertices both have reservation of the same parity. In this case we add or subtract to both reservations, which reduces the absolute values of *both* reservations. Hence we reduce the total absolute reservation, which keeps the invariant.

□

The lemma above shows that the sum of absolute values of reservation at all vertices except v' at any point during the remaining iterations in the follower's algorithm is at most the initial reservation $\delta^{i'}(v')$. Note that v' is bought in both cases. In p^* all priceable vertices are bought by the follower, but this might not be true for p' and vertices $v \neq v'$: p' might lose revenue there. By Lemma 1, this can be fixed by reducing the price in p' of every priceable vertex to the sum of the budgets of incident edges. Note that in p^* every vertex is bought, which implies every price $p^*(v) = \sum_{e:e=(u,v)} \gamma^*_e$ is also the sum of budgets of incident edges. As the total absolute reservation at the end of the algorithm is at most $\delta^{i'}(v')$, the total decrease in revenue that is lost on vertices $v \neq v'$ in this step is at most $\delta^{i'}(v')$. This is exactly the revenue surplus that p' generates over p^* at vertex v'. Thus, pricing p' yields at least as much revenue as p^*. This implies our claim, i.e., one can transform any pricing by iteratively adjusting the prices without decreasing revenue, such that all budgets of the edges are equal to those resulting from p_g. In particular, this implies that the greedy pricing p_g is optimal. □

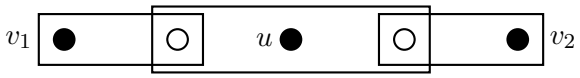


Figure 2: Instance with element frequency 3, for which the greedy pricing is suboptimal. Priceable sets are displayed as filled vertices, fixed-cost sets are empty vertices. All fixed-cost sets have cost 1. Elements are displayed as boxed hyperedges. Follower considers at first the hyperedge incident to u.

The next proposition shows that knowing the order of the edges when pricing for the primal-dual algorithm in StackVC is essential.

Proposition 2. *For every constant $\varepsilon > 0$, there exists an instance of* StackVC *such that if the order of the edges in the follower's primal-dual 2-approximation algorithm is unknown to the leader, then every pricing p yields an approximation ratio of $\Omega(1/\varepsilon)$.*

Proof. The instance is a path with four vertices, $V = \{u_1, ..., u_4\}$ and $E = \{e_i = (u_i, u_{i+1}) \mid i = 1, 2, 3\}$. Vertex u_3 is the only priceable vertex. Moreover, $c(u_4) = \varepsilon$, $c(u_1) = c(u_2) = 1$. For the order (e_3, e_1, e_2) the leader obtains revenue at most ε, while for the order (e_3, e_2, e_1) she obtains revenue $1 + \varepsilon$. Without knowing the order she could either decide to price $p(u_3) > \varepsilon$, which gives revenue 0 with the first order. Otherwise, she could secure revenue ε using $p(u_3) = \varepsilon$, but then she might be able to obtain $1 + \varepsilon$ with the right order. Thus, any pricing strategy gives a ratio of at least $(1 + \varepsilon)/\varepsilon \in \Omega(1/\varepsilon)$. □

3.3 Hardness Results

The main argument in the previous section works only for the case of regular vertex cover. Let us turn to the case of set cover with elements contained in at least three sets, i.e., elements with *frequency* at least three. We understand them as hyperedges incident to more than two vertices. In this case it might be profitable to reduce the price for a vertex from the value in the greedy pricing. Consider the example in Fig. 2. If we set $p(u) = 1$, then there is no revenue at v_1 and v_2. If instead we set $p(u) = 0$, then the optimum revenue is 2, as both v_1 and v_2 will be bought for a price of 1. This property can be used to show that the set cover pricing problem is much harder to solve.

Theorem 6. StackSC *is* APX*-hard even if all elements have maximum frequency 3 and all fixed-cost sets have cost 1.*

We prove the APX-hardness in two steps. We begin by proving NP-hardness and introduce the main technicalities. This allows for a simple proof of APX-hardness below.

Theorem 7. StackSC *is* NP*-hard even if all elements have maximum frequency 3 and all fixed-cost sets have cost 1.*

Proof. We describe a reduction from MAX-2-SAT. An instance of MAX-2-SAT is given by a set of $\{x_1, \ldots, x_n\}$ of boolean variables and a set $\{c_1, \ldots, c_m\}$ of clauses. Each clause has exactly 2

variables, and the problem is to find an assignment of variables such that the maximum number of clauses is satisfied.

We construct an instance of vertex cover pricing with hyperedges. Our construction is layered in three stages, in *variable gadgets*, *distribution gadgets*, and *clause gadgets*. There is a variable gadget for each variable x with two priceable vertices u_x and $u_{\overline{x}}$. The optimal pricing sets the price of one vertex to 1 and the other price to 0. This corresponds to a decision about x, i.e., if $p(u_x) < p(u_{\overline{x}})$, then x is true and false otherwise. In the distribution gadgets the decision about x is distributed to the gadgets for clauses that include x. In each clause gadget we can obtain a maximum revenue of 1. It is obtained if and only if at least one of the variables evaluates the clause to true. The reduction is complete by observing that it is possible to obtain a revenue of $n + k$ if and only if k clauses are satisfied.

In the first stage we introduce for each variable x a *variable gadget* as shown in Fig. 3(a). There are two priceable vertices u_x and $u_{\overline{x}}$, two fixed-cost vertices v_x and $v_{\overline{x}}$ of cost 1, and one coordination vertex of cost 1. The two hyperedges incident to priceable vertices are considered first, then the edge incident to v_x and $v_{\overline{x}}$. Note that we can extract a total revenue of at most 1 from u_x and $u_{\overline{x}}$. Independently of the prices we set, after execution of the algorithm at least one of the vertices v_x or $v_{\overline{x}}$ is tight and bought. This encodes the decision about setting the variables as follows. Namely, x is set to true (false) if and only if the corresponding set v_x ($v_{\overline{x}}$) is not part of the cover after running the algorithm on the variable gadget.

Based on the variable gadgets we construct *distribution gadgets* to distribute the information about the setting of variables to the clause gadgets. For each non-negated occurrence of a variable x in any of the clauses, we introduce a vertex v with fixed cost 1. Then, we introduce one hyperedge incident to all these vertices and v_x. The construction for negated occurrences of x and $v_{\overline{x}}$ is similar. The maximum frequency depends on the maximum number of occurrences of a variable in the sets. This, however, can be reduced to 3 by using a layered construction that we describe in the end of the proof.

Finally, for each clause there is a *clause gadget* starting with the two fixed-cost sets (denoted v_1 and v_2) that were introduced for the corresponding clause in the distribution gadget. The construction is depicted in Fig. 3(b). One edge, denoted e_c, is incident to v_1, v_2, and an additional fixed-cost vertex v_3. The second edge is (v_3, u_c) for a priceable vertex u_c. The follower considers e_c first.

The global order of the elements is given by first considering all the variable gadgets, then all the distribution gadgets, and finally all the clause gadgets, while using for each gadget the respective order of edges outlined before.

We now show that in the given instance one can obtain a revenue of at least $n + k$ if and only if the corresponding instance of MAX-2-SAT has an assignment that satisfies at least k clauses. It is obvious that we can always obtain at most a revenue of n from the variable gadgets. We already outlined above that after the follower has considered all variable gadgets, at least one vertex v_x or $v_{\overline{x}}$ from each gadget is bought and tight. For exposition consider a clause $c = (\overline{x}, y)$, such that both $v_{\overline{x}}$ and v_y are tight. Then the budgets of the corresponding distribution edges are not raised, and hence the budget of e_c in the clause gadget is raised to 1. This means that v_1, v_2 and v_3 are bought, while there is no profit from u_c. In contrast, if at least one set $v_{\overline{x}}$ or v_y is not tight after processing the variable gadgets, the budget of the distribution elements is raised to the amount of existing slack. Hence, the budget of e_c decreases. In this case a profit corresponding to the slack can be obtained at u_c. Note that e_c is designed such that the budget is raised until the v_1 or v_2

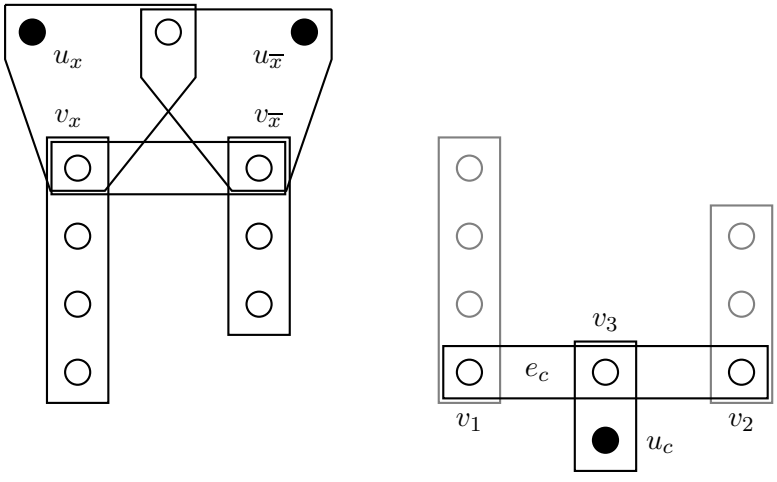


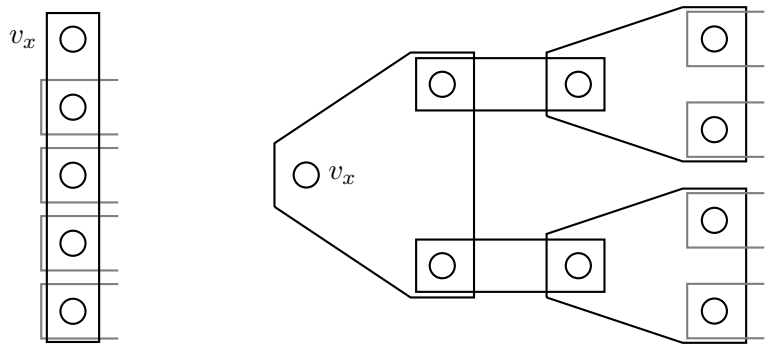
Figure 3: Gadgets used in the reduction. Filled vertices are priceable, fixed-cost vertices have cost 1. Hyperedges are displayed as boxes. (a) Variable gadget with distribution gadgets for a variable occurring in three clauses non-negated and in two clauses negated. (b) Clause gadget connecting vertices from the distribution gadgets of the occurring variables.

with smaller budget becomes tight. Hence, in order to obtain a revenue of 1, there has to be at least one vertex which is tight and bought by the distribution element. Then, v_3 can be bought together with u_c at a cost of 1.

We use this observation in our argument. Consider any pricing of the vertices. If for a variable gadget $p(u_x) = p(u_{\overline{x}})$, then after processing the variable gadgets, both v_x and $v_{\overline{x}}$ are bought and unnecessarily restrict the profit that can be obtained at the clause gadgets. Hence, we can w.l.o.g. assume that $p(u_x) \neq p(u_{\overline{x}})$. Now the set with larger price is the one, for which the corresponding set v_x or $v_{\overline{x}}$ will be bought. Also, the total revenue extracted from u_x and $u_{\overline{x}}$ is at most 1. So we can restrict attention to pricings such that in each variable gadget exactly one vertex has price 1 and one vertex has price 0. This encodes a decision about the variable as mentioned before. In addition, we can obtain a revenue of at most 1 from each priceable vertex u_c. The revenue is obtained for clause $c = (x, y)$ if and only if the budget of e_c is restricted to 0. This requires that at least one of the corresponding distribution vertices v_1 or v_2 is bought after processing the edges. These vertices get bought if and only if one of the corresponding sets v_x or v_y remains unpurchased after processing the variable gadgets. Hence, u_c gets bought at a price of 1 if and only if at least one of the corresponding priceable sets u_x or u_y has been restricted to price 0. In this way a revenue of $n + k$ directly implies an assignment of variables such that k clauses are satisfied.

Finally, in order to satisfy maximum frequency 3, we can modify the construction with one hyperedge in the distribution gadget to a layered tree construction with more vertices and edges as shown in Fig. 4. The construction preserves the argument and generates only a constant overhead in the size of the reduction. □

We are now ready to show the APX-hardness proof for StackSC.



(a) (b)

Figure 4: Gadget to avoid hyperedges with high frequency. Hyperedges are displayed as boxes, all vertices have fixed-cost 1. (a) Distribution gadget for non-negated occurrences of a variable x. (b) Equivalent tree gadget with maximum frequency 3. Edges are considered in order from left to right.

Proof of Theorem 6: We use the same reduction as for showing NP-hardness, however, we restrict attention to the special case E3-OCC-MAX-2-SAT, in which each variable appears in exactly three clauses. Note that by a simple majority vote, we can satisfy at least two literals per variable. As there are at most two literals in each clause, we can always get at least n satisfied clauses, and the optimum solution to an instance of E3-OCC-MAX-2-SAT has value between n and $2n^2$. Now suppose it is possible to receive a revenue of at least $n+k$ such that $(n+k)(1+\varepsilon) \geq n+k^*$. This means that $k+\varepsilon(n+k) \geq k^*$. We observed that it is trivial to guarantee $k \geq n$, hence we can bound $k+2\varepsilon k \geq k^*$. Thus, the existence of a $(1+\varepsilon)$-approximation for set cover pricing implies a $(1+2\varepsilon)$-approximation for E3-OCC-MAX-2-SAT. It is, however, known that for any $\delta < 1/459$ this problem is hard to approximate to within $\frac{460}{459} - \delta$ unless P=NP [5]. This implies that our pricing problem is NP-hard to approximate to within $\frac{919}{918} - \varepsilon$, for any $\varepsilon < 1/918$, which proves the theorem. □

Finally, we derive a devastating lower bound on greedy, single-price, and near-uniform price assignments in StackSC. In particular, we outline an example, with $n = k^2+k+2$ hyperedges and $n+1$ vertices, in which both the pricings extract only a constant revenue, while the optimal pricing gets a revenue of k^k. This number must be set as a fixed cost, which dominates the representation size. This means we get a lower bound on the approximation ratio of $2^{\Omega(|I|)}$, where $|I|$ is the representation size of the instance.

Theorem 8. *The greedy, single-price, and near-uniform price assignments yield an approximation factor of $2^{\Omega(|I|)}$ for* StackSC*, where $|I|$ is the size of the representation.*

Proof. Our family of instances essentially consists of k layers with $k+1$ vertices each, leading to an exponential increase in fixed costs. An instance for $k=3$ is depicted in Fig. 5. The follower considers hyperedges from left to right. If we set the price of the left priceable vertex to 0, the follower raises the budgets of the hyperedges with frequency k, and we can obtain an optimal

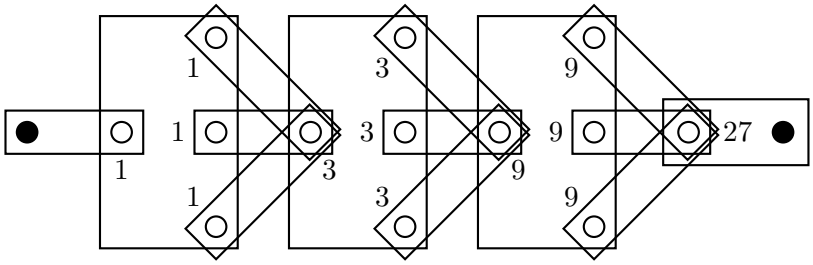


Figure 5: Structure of instances of set cover pricing establishing an exponential lower bound for greedy, single-price, and near-uniform price assignments. Follower considers hyperedges in order from left to right.

revenue of k^k at the right vertex. If, however, we increase the price at the left vertex to 1 or greater, this results in a revenue of 0 at the right vertex. Thus, none of the mentioned price assignments can obtain a reasonable approximation guarantee. □

Using the distribution construction outlined in the proof of Theorem 6 (Fig. 4 depicts the main idea of the construction), it is possible to reduce the maximum frequency of any element to 3 with only a constant overhead in the size of the instance.

4 Conclusions and Open Problems

In this paper we have initiated the study of algorithmic pricing with complex valuation functions but computationally limited customers in the setting of Stackelberg pricing. We have presented a number of results on the special cases of knapsack and vertex- or set-cover pricing. There are a number of obvious open questions directly addressing the results presented here. In particular, is the knapsack pricing problem APX-hard? Or is it possible to extend our enumeration-based algorithm to achieve a PTAS? Note, however, that a PTAS cannot be based on near-uniform price assignments, but necessarily needs to assign a super-constant number of different efficiencies to a super-constant number of different objects. In the case of set-cover pricing, what is the best approximation guarantee we can achieve for revenue maximization when elements have frequency greater than 2? So far, we only know that the problem is APX-hard but do not know of any non-trivial upper bounds. Finally, are there other interesting problems and general classes of follower algorithms that allow for exact or near-optimal revenue maximization in polynomial time?

Acknowledgments

We wish to thank Guido Proietti for the helpful discussions in the starting phase of this study. We are very grateful to Piotr Krysta for several insightful discussions relating to a number of results presented in this paper during a productive week of research in Liverpool.

Financial support is acknowledged as follows. The work has been supported by DFG grant Kr 2332/1-2 within Emmy Noether Program. Patrick Briest was supported by a scholarship of the German Academic Exchange Service (DAAD) for a stay at Cornell University, where part of

this work was done. Martin Hoefer is supported by UMIC Research Center at RWTH Aachen University, by DFG grant Ho 3831/3-1, and by a scholarship of the German Academic Exchange Service (DAAD) for a stay at Stanford University, where part of this work was done. Carmine Ventre acknowledges support by EPSRC grant EP/F069502/1 and EU through IP AEOLUS.

References

[1] Gagan Aggarwal, Tomás Feder, Rajeev Motwani, and An Zhu. Algorithms for multi-product pricing. In *Proc. 31st Intl. Coll. Automata, Languages and Programming (ICALP)*, pages 72–83, Turku, Finland, 2004. Springer Verlag.

[2] Maria-Florina Balcan and Avrim Blum. Approximation algorithms and online mechanisms for item pricing. *Theory of Computing*, 3(1):179–195, 2007.

[3] Maria-Florina Balcan, Avrim Blum, Jason Hartline, and Yishay Mansour. Reducing mechanism design to algorithm design via machine learning. *J. Comput. Syst. Sci.*, 74(8):1245–1270, 2008.

[4] Maria-Florina Balcan, Avrim Blum, and Yishay Mansour. Item pricing for revenue maximization. In *Proc. 9th Conf. Electronic Commerce (EC)*, pages 50–59, Chicago, IL, USA, 2008. ACM Press.

[5] Piotr Berman and Marek Karpinski. Improved approximation lower bounds on small occurrence optimization. Technical Report ECCC-TR03-008, available at eccc.hpi-web.de, 2003.

[6] Davide Biló, Luciano Gualá, Stefano Leucci, and Guido Proietti. Specializations and generalizations of the Stackelberg minimum spanning tree game. In *Proc. 6th Intl. Workshop Internet & Network Economics (WINE)*, pages 75–86, Stanford, CA, USA, 2010. Springer Verlag.

[7] Davide Bilò, Luciano Gualà, Guido Proietti, and Peter Widmayer. Computational aspects of a 2-player Stackelberg shortest paths tree game. In *Proc. 4th Intl. Workshop Internet & Network Economics (WINE)*, pages 251–262, Shanghai, China, 2008. Springer Verlag.

[8] Patrick Briest. Uniform budgets and the envy-free pricing problem. In *Proc. 35th Intl. Coll. Automata, Languages and Programming (ICALP)*, volume 1, pages 808–819, Reykjavik, Iceland, 2008. Springer Verlag.

[9] Patrick Briest, Martin Hoefer, Luciano Gualà, and Carmine Ventre. On Stackelberg pricing with computationally bounded consumers. In *Proc. 5th Intl. Workshop Internet & Network Economics (WINE)*, pages 42–54, Rome, Italy, 2009. Springer Verlag.

[10] Patrick Briest, Martin Hoefer, and Piotr Krysta. Stackelberg network pricing games. In *Proc. 25th Symp. Theoretical Aspects of Computer Science (STACS)*, pages 133–142, Bordeaux, France, 2008. Online at stacs-conf.org.

[11] Patrick Briest and Sanjeev Khanna. Improved hardness of approximation for Stackelberg shortest-path pricing. CoRR abs/0910.0110, 2009.

[12] Patrick Briest and Piotr Krysta. Single-minded unlimited-supply pricing on sparse instances. In *Proc. 17th Symp. Discrete Algorithms (SODA)*, pages 1093–1102, Miami, FL, USA, 2006. ACM Press.

[13] Patrick Briest and Piotr Krysta. Buying cheap is expensive: Hardness of non-parametric multi-product pricing. In *Proc. 18th Symp. Discrete Algorithms (SODA)*, pages 716–725, New Orleans, LA, USA, 2007. ACM Press.

[14] Jean Cardinal, Erik Demaine, Samuel Fiorini, Gwenael Joret, Stefan Langerman, Ilan Newman, and Oren Weimann. The Stackelberg minimum spanning tree game on planar and bounded-treewidth graphs. In *Proc. 5th Intl. Workshop Internet & Network Economics (WINE)*, pages 125–136, Rome, Italy, 2009. Springer Verlag.

[15] Jean Cardinal, Erik Demaine, Samuel Fiorini, Gwenael Joret, Stefan Langerman, Ilan Newman, and Oren Weimann. The Stackelberg minimum spanning tree game. *Algorithmica*, 59(2):129–144, 2011.

[16] Shuchi Chawla, Jason Hartline, and Robert Kleinberg. Algorithmic pricing via virtual valuations. In *Proc. 8th Conf. Electronic Commerce (EC)*, pages 243–251, San Diego, CA, USA, 2007. ACM Press.

[17] Ning Chen, Arpita Ghosh, and Sergei Vassilvitskii. Optimal envy-free pricing with metric substitutability. *SIAM J. Comput.*, 40(3):623–645, 2011.

[18] Erik Demaine, Uriel Feige, MohammadTaghi Hajiaghayi, and Mohammad Salavatipour. Combination can be hard: Approximability of the unique coverage problem. *SIAM J. Comput.*, 38(4):1464–1483, 2008.

[19] Khaled Elbassioni, Rajiv Raman, Saurabh Ray, and René Sitters. On profit-maximizing pricing for the highway and tollbooth problems. In *Proc. 2nd Intl. Symp. Algorithmic Game Theory (SAGT)*, pages 275–286, Paphos, Cyprus, 2009. Springer Verlag.

[20] Khaled Elbassioni, René Sitters, and Yan Zhang. A Quasi-PTAS for profit-maximizing pricing on line graphs. In *Proc. 15th European Symposium on Algorithms (ESA)*, pages 451–462, Eilat, Israel, 2007. Springer Verlag.

[21] Lisa Fleischer. Linear taxes suffice: New bounds and algorithms for tolls in single source networks. *Theoret. Comput. Sci.*, 348(2-3):217–225, 2005.

[22] Lisa Fleischer, Kamal Jain, and Mohammad Mahdian. Tolls for heterogeneous selfish users in multicommmodity networks and generalized congestion games. In *Proc. 45th Symp. Foundations of Computer Science (FOCS)*, pages 277–285, Rome, Italy, 2004. IEEE Press.

[23] Michael Garey and David Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.

[24] Venkatesan Guruswami, Jason Hartline, Anna Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *Proc. 16th Symp. Discrete Algorithms (SODA)*, pages 1164–1173, Vancouver, BC, Canada, 2005. ACM Press.

[25] Jason Hartline and Vladlen Koltun. Near-optimal pricing in near-linear time. In *Proc. 8th Workshop Algorithms and Data Structures (WADS)*, pages 422–431, Waterloo, Canada, 2005. Springer Verlag.

[26] Martin Hoefer, Lars Olbrich, and Alexander Skopalik. Taxing subnetworks. In *Proc. 4th Intl. Workshop Internet & Network Economics (WINE)*, pages 286–294, Shanghai, China, 2008. Springer Verlag.

[27] Geroge Karakostas and Stavros Kolliopoulos. Edge pricing of multicommodity networks for heterogeneous users. In *Proc. 45th Symp. Foundations of Computer Science (FOCS)*, pages 268–276, Rome, Italy, 2004. IEEE Press.

[28] Martine Labbé, Patrice Marcotte, and Gilles Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management Sci.*, 44:1608–1622, 1998.

[29] Sébastien Roch, Gilles Savard, and Patrice Marcotte. Design and analysis of an approximation algorithm for Stackelberg network pricing. *Networks*, 46(1):57–67, 2005.

[30] Chaitanya Swamy. The effectiveness of Stackelberg strategies and tolls for network congestion games. In *Proc. 18th Symp. Discrete Algorithms (SODA)*, pages 1133–1142, New Orleans, LA, USA, 2007. ACM Press.

[31] Vijay Vazirani. *Approximation Algorithms.* Springer Verlag, Heidelberg, Germany, 2000.

[32] Heinrich von Stackelberg. *Marktform und Gleichgewicht (Market and Equilibrium).* Verlag von Julius Springer, 1934.