Single-Minded Unlimited Supply Pricing on Sparse Instances

Patrick Briest* Piotr Krysta†

Abstract

We deal with the problem of finding profit-maximizing prices for a finite number of distinct goods, assuming that of each good an unlimited number of copies is available, or that goods can be reproduced at no cost (e.g., digital goods). Consumers specify subsets of the goods and the maximum prices they are willing to pay. In the considered single-minded case every consumer is interested in precisely one such subset. If the goods are the edges of a graph and consumers are requesting to purchase paths in this graph, then we can think of the problem as pricing computer network connections or transportation links.

We start by showing weak NP-hardness of the very restricted case in which the requested subsets are nested, i.e., contained inside each other or non-intersecting, thereby resolving the previously open question whether the problem remains NP-hard when the underlying graph is simply a line. Using a reduction inspired by this result we present an approximation preserving reduction that proves APX-hardness even for very sparse instances defined on general graphs, where the number of requests per edge is bounded by a constant B and no path is longer than some constant ℓ. On the algorithmic side we first present an $O(\log \ell + \log B)$-approximation algorithm that (almost) matches the previously best known approximation guarantee in the general case, but is especially well suited for sparse problem instances. Using a new upper bounding technique we then give an $O(\ell^2)$-approximation, which is the first algorithm for the general problem with an approximation ratio that does not depend on B.

1 Introduction

The question of how to optimally price goods in a given market has always been central in economics. Only recently, however, companies are faced with a variety of new possibilities emerging from the wide spread use of the Internet to gather large amounts of data about consumer preferences in their market segments. A lot of work has been done on the side of computer science to answer the question of how this data can be used in the computation of optimal, i.e., revenue maximizing prices. Inspired by General Motor's Auto Choice Advisor (a web site designed to gather such data), Rusmevichientong, Van Roy and Glynn [15] defined the non-parametric multi-product pricing problem, in which consumers are characterized by a ranking of all available products and some budget constraint. The first algorithms with provable approximation guarantee for this problem were given by Aggarwal, Feder, Motwani and Zhu [1]. Very recently, another approach to pricing that requires more precise knowledge about consumers' preferences was introduced by Guruswami, Hartline, Karlin, Kempe, Kenyon and McSherry [11]. In the envy-free pricing problem the goal is to find profit-maximizing prices that allow an allocation of the goods that maximizes the personal utility of all participating consumers. This problem, which is particularly relevant in the case that goods are only available in limited supply, is inspired by the notion of truthfulness in mechanism design [14], which has in turn received a lot of attention from the side of computer science in the context of efficiently computable auctions during the last years [2, 5, 7, 13]. In fact, envy-free pricing and a corresponding allocation define a fair pricing equilibrium and can be used as a comparison to measure the quality of the revenue generated by auctions. Guruswami et al. present hardness results and approximation algorithms for the case of unit-demand bidders (i.e., consumers that are interested in purchasing a single good out of their set of preferences) and for single-minded bidders (i.e., bidders that are interested in a single subset of all goods) assuming unlimited supply. The later nicely models the sale of digital goods or, in general, goods that can be reproduced at no or almost no additional cost, and it is the base problem for our paper. If we assume that the goods we are selling are the edges of a graph and that consumers are purchasing paths in this graph we can interpret this as the problem of pricing network connections, street segments (therefore termed *toll-booth problem* in [11]) or other types of transportation links (e.g., railway or flight connections). If the underlying graph is just a line, then this problem is called *highway problem* in [11]. Interestingly, even this very restricted variant turns out to be intriguingly complex. Hartline and Koltun [12] have presented a near-linear time FPTAS for the practically relevant case that the number of goods for sale is a fixed constant.

*Department of Computer Science, University of Dortmund, Baroper Str. 301, 44221 Dortmund, Germany. E-mail: patrick.briest@cs.uni-dortmund.de. The author is supported by DFG grant Kr 2332/1-2 within Emmy Noether program.

†Department of Computer Science, University of Dortmund, Baroper Str. 301, 44221 Dortmund, Germany. E-mail: piotr.krysta@cs.uni-dortmund.de. The author is supported by DFG grant Kr 2332/1-2 within Emmy Noether program.

Independently of this work, Balcan and Blum [4] also give approximation algorithms for the unlimited supply case.

Another related line of research is mechanism design for profit maximization with goods available in unlimited or limited supply [8, 10]. The main difference between this approach to profit maximization and the one studied in this paper is that they define a specific benchmark (optimal fixed price profit) to which they compare the attained profit, whereas we compare to the overall best possible profit. Especially, incentive issues do not need to be taken into account in the unlimited supply case, as clearly every bidder who can afford to buy her requested goods at the computed prices can be allocated what she requests.

Bouhtou et al. [6] study a related pricing problem with unlimited supply on a directed graph, which has interesting applications in telecommunication networks. In fact they prove that their version of the problem is APX-hard by using a reduction from the same problem as we do, but their reduction does not imply APX-hardness for the problem we consider in this paper.

1.1 Preliminaries Our definition of the *single-minded unlimited supply pricing* problem is the same as the one given in [11]. We will associate a single-minded bidder with the set she is interested in and mostly talk of a *set* and its *value* rather than a bidder and her bid.

DEFINITION 1. *The* single-minded unlimited supply pricing *problem (SUSP) is defined by a universe U of goods and a collection $\mathcal{S}$ of subsets of U (allowing multiple copies of a subset), where each set $S \in \mathcal{S}$ has value $w(S)$. We want to assign prices $c^* : U \to \mathbb{R}_0^+$ to the goods, such that*

$$c^* \in \operatorname{argmax}_c \sum_{S \in \mathcal{S}: w(S) \geq \sum_{e \in S} c(e)} \sum_{e \in S} c(e),$$

i.e., c^* maximizes the sum of prices of goods in those sets for which the sum of prices is smaller than the set's value. By $prof(c)$ we refer to the total profit resulting from price assignment c.

Intuitively, a bidder specifies the maximum price she is willing to pay for a certain set of goods. After a price has been assigned to each good, bidders decide to buy their sets depending on whether the sum of prices of goods contained in their sets exceeds their specified budget. If this is not the case, a bidder is charged exactly the sum of prices of the goods she receives. Whenever the problem is defined on an underlying graph, we slightly adjust our notation to reflect this difference.

DEFINITION 2. *In the* SUSP problem on graphs *(G-SUSP) we are given a graph $G = (V, E)$ rather than a universe of goods and a collection $\mathcal{P}$ of paths in G. A feasible price assignment is a mapping $c : E \to \mathbb{R}_0^+$.*

Note, that if sets are nested, i.e., for any $S_1, S_2 \in \mathcal{S}$ we have that $S_1 \subseteq S_2$, $S_2 \subseteq S_1$ or $S_1 \cap S_2 = \emptyset$, it does not make a difference if the problem is defined on a graph or not. In fact, each such instance can easily be defined on a graph that consists of a single line by simply ordering the goods appropriately.

1.2 Our results We will present two types of results in this paper. Guruswami et al. [11] proved that SUSP is APX-hard. However, their reduction creates a problem instance in which certain goods are requested by a constant fraction of all participating bidders. From a technical standpoint, this appears quite unavoidable, since an approximation preserving reduction to SUSP always brings up the problem that we need to force optimal (or approximately optimal) solutions to be in a sense integral in order to be able to reconstruct solutions to the (integral) problem that was our reduction's starting point. On the other hand, it is certainly desirable to have hardness results also for sparse instances, especially because it turns out that the number of requests per good is the most crucial parameter when it comes to finding good approximations using upper bounding techniques known so far. The main result of the first part of this paper is a proof of APX-hardness even if the parameters mentioned above (and some more) are bounded by a small constant. The proof is done by an approximation preserving reduction from a variant of MAX-SAT. Technically, this requires the design of individual components (a small graph with some paths defined on it) to represent the formula's clauses, which ensure integrality of the optimal solution and give different profit depending on whether the corresponding clause is satisfied. The design used here is inspired by much simpler components that are used to prove NP-completeness of the highway problem. Summarizing, we present the following negative results:

(1) Guruswami et al. [11] have already pointed out that G-SUSP is surprisingly complex even when G is a line, but left as an open problem whether it remains NP-hard or is polynomially solvable. We answer this question and show that this version of the problem is NP-hard even with the further limitation that paths are nested. Additionally, we give an FPTAS for the case of nested paths, showing that NP-hardness here is only weak.

(2) APX-hardness of G-SUSP has been shown in [11]. We prove that G-SUSP remains APX-hard even under a variety of strong restrictions, e.g., considering only constant degree graphs, a constant number of requests per edge, paths of constant length, etc.

In the second part of the paper we present approximation algorithms for general SUSP with approximation ratios that depend on the maximum size ℓ of the sets and the maximum number B of requests per good. The ratio of our first algorithm depends on ℓ and B and matches the ratios of previously known algorithms even in the worst case, i.e., if none of the above parameters can be bounded better than trivially ($\ell \leq |U|$, $B \leq |\mathcal{S}|$). If these parameters turn out to be small, however, our algorithms are capable of exploiting these instances' sparse structure in order to obtain better approximation ratios. The approximation ratio of our second algorithm depends only on ℓ. More precisely, we present the following:

(1) Guruswami et al. [11] gave an $O(\log |U| + \log |\mathcal{S}|)$-approximation for SUSP. Our first algorithm achieves approximation ratio $O(\log B + \log \ell)$, which gives an advantage on sparse instances but matches the previous result in the general case. It is interesting to see that our result in the general situation can be achieved by completely different approaches than those of [11]. Our algorithm is based on an elementary partitioning argument. We use the sum of the values of all sets as an upper bound on the value of an optimal solution (the same upper bound is used in [11]), and our approximation ratio is almost tight with respect to this upper bound.

(2) The second algorithm gives an $O(\ell^2)$-approximation. This cannot be achieved by using the sum of the values of all sets as an upper bound on the optimal profit, but requires a novel upper bounding technique, which is obtained by first defining optimal prices for each single good without taking care of other prices. The main step towards a solution is then combining these prices into a common price assignment, which is done by using a specially designed hypergraph structure to model dependencies that we have to take into account. Independently of our results Balcan and Blum [4] have achieved a randomized $O(\ell)$-approximation, which relies on a basically identical upper bounding technique.

The rest of the paper is organized as follows. Section 2 deals with the highway problem in which the underlying graph is simply a line. We first give a proof of NP-completeness and then show how to derive an FPTAS for the restricted case used for the preceeding hardness result. Section 3 gives an APX-hardness result for a severely restricted class of instances modeling realistic networking conditions. After this, Sections 4 and 5 present the two approximation algorithms for the general pricing problem.

2 The Highway Problem

We now focus on the special case of G-SUSP where the underlying graph structure is simply a line. We can think of this as the problem of pricing single segments of a privately owned highway. Guruswami et al. give a pseudopolynomial algorithm (which can be turned into an FPTAS by appropriately scaling the input) for the case in which the length of all paths is bounded by a constant. We introduce another special case of the problem in which we allow paths to have arbitrary length but require that they are *nested*, i.e., given any two paths $P_1, P_2 \in \mathcal{P}$ we have that $P_1 \subseteq P_2$, $P_2 \subseteq P_1$ or $P_1 \cap P_2 = \emptyset$. As mentioned before any instance of SUSP in which the requested sets are nested can be viewed as defined on a line. We will prove NP-hardness of this problem and show how to derive an FPTAS by a simple dynamic programming approach.

2.1 NP-hardness We prove NP-hardness by a reduction from the PARTITION problem. Given weights $w_1, \ldots, w_n$ we want to find a set $S \subset \{1, \ldots, n\}$, such that $\sum_{i \in S} w_i = \sum_{i \notin S} w_i$, i.e., find a partitioning into two sets of identical total weight. PARTITION is known to be NP-hard [9].

THEOREM 2.1. *G-SUSP with nested paths is NP-hard.*

Proof. We prove a polynomial time reduction from PARTITION to the considered problem. For each weight w_i we construct a *weight component* $\mathcal{W}_i$ as depicted in Figure 1. On a graph consisting of 3 vertices v_1^i, v_2^i, v_3^i and 2 edges e_1^i, e_2^i we define the 3 possible different paths $P_1^i = \{e_1^i\}$, $P_2^i = \{e_2^i\}$, $P_3^i = \{e_1^i, e_2^i\}$ and let $w(P_1^i) = w(P_2^i) = w(P_3^i) = w_i$. We start with a simple observation about weight components which turn out to have the nice property of in a sense discretizing the problem.

More precisely, the maximum profit obtainable from weight component $\mathcal{W}_i$ is $2w_i$. If profit $2w_i$ is obtained under price assignment c then $c(e_1^i) = c(e_2^i) = w_i$ or $c(e_1^i) + c(e_2^i) = w_i$. To see this, note that if path P_3^i contributes to the profit, it must be the case that $c(e_1^i) + c(e_2^i) \leq w_i$. P_1^i and P_2^i can never give more profit than $c(e_1^i)$ and $c(e_2^i)$, respectively. It immediately follows that total profit is at most $2w_i$. On the other hand, the profit obtained from each P_1^i and P_2^i is also bounded by w_i and so profit $2w_i$ can also not be exceeded if P_3^i does not contribute. This gives the first part of the claim. From the above argumentation it follows that profit $2w_i$ cannot be reached if $c(e_1^i) + c(e_2^i) < w_i$, while, e.g., $c(e_1^i) = c(e_2^i) = w_i/2$ results in full profit. Now assume that c results in profit $2w_i$ but $c(e_1^i) + c(e_2^i) > w_i$. Then P_3^i obviously contributes 0. It follows that P_1^i and P_2^i must give profit w_i each and, thus, $c(e_1^i) = c(e_2^i) = w_i$. We note that we can make use of the symmetry of a weight component by slightly changing the above claim and requiring w.l.o.g. that $c(e_1^i) = c(e_2^i) = w_i/2$ instead of $c(e_1^i) + c(e_2^i) = w_i$. In fact, we will from now on only consider the price $c(\mathcal{W}_i) = c(e_1^i) + c(e_2^i)$ that is assigned to weight component $\mathcal{W}_i$ and implicitly assume that this price is split evenly among edges e_1^i and e_2^i.

We now define the final G-SUSP instance. The weight components $\mathcal{W}_1, \ldots, \mathcal{W}_n$ are assembled into a single line by identifying vertices v_3^i and v_1^{i+1} for $i = 1, \ldots, n-1$. We define one more path P running all the way from v_1^1 to v_3^n and set $w(P) = (3/2)\sum_{i=1}^n w_i$, as shown in Figure 1. It is then straightforward to argue that total profit $(7/2)\sum_{i=1}^n w_i$ can be reached on this instance if and only if a partitioning S with $\sum_{i\in S} w_i = \sum_{i\notin S} w_i$ exists. For the one direction we define prices c by $c(\mathcal{W}_i) = 2w_i$ if $i \in S$ and $c(\mathcal{W}_i) = w_i$ else. For the other direction we argue that the optimal pricing is of just this form and let $S = \{i \mid c(\mathcal{W}_i) = 2w_i\}$. □

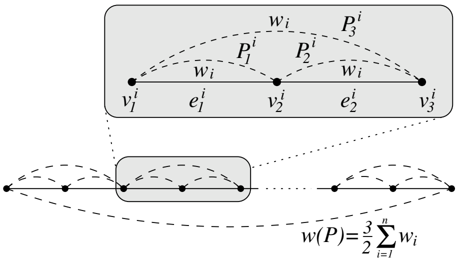

Figure 1: Single weight component $\mathcal{W}_i$. Path P connecting all n components ensures that maximum profit can be reached only if the weights can be 2-partitioned.

2.2 An FPTAS We present a pseudopolynomial algorithm which can be turned into an FPTAS by scaling and rounding the input appropriately. For the description of the dynamic programming approach we assume that all declared prices are integral. By an observation of Guruswami et al. concerning the unimodularity of the problem's constraint matrix this implies the existence of an integral optimal solution. Given a G-SUSP instance with edges E, $|E| = m$, and nested paths $\mathcal{P}$, $|\mathcal{P}| = n$, we define the set of *intervals* $\mathcal{I}$ as follows. First, each path P defines an interval $I = P$. Then we add the interval $E = \{e_1, \ldots, e_m\}$ containing all edges. Now consider any interval I and let J be the maximum size interval contained in I. We then also add $K = I \backslash J$ to $\mathcal{I}$ if it is not already contained. Note, that in general K need not be an interval in the classical sense, since it might contain edges that are situated on the left or right of J, respectively. It is, however, w.l.o.g. to assume that this is not the case, because we can always reorder edges to ensure that I and J have the same left border. Conceptually, at this point we view paths as simply collections of edges and observe that edges can be arranged in a single line, such that paths are mapped onto intervals. If interval I is defined by path P we let $w(I) = w(P)$ be the interval's value. If I is defined in a later step (i.e., not defined by a path), we set $w(I) = 0$. Note, that intervals can naturally be arranged as a binary tree with root corresponding to the complete line of edges. For any interval I we let A_b^I refer to the maximum profit that can be obtained from paths fully contained in I under the condition that the prices assigned to edges in I sum up to exactly b, i.e., $\sum_{e\in I} c(e) = b$. Consider interval I containing maximum length subintervals J and K. We have

$$A_b^I = \begin{cases} \max_{b'}\{A_{b'}^J + A_{b-b'}^K\} + b, & \text{if } b \leq w(I) \\ \max_{b'}\{A_{b'}^J + A_{b-b'}^K\}, & \text{else} \end{cases}$$

by the observation that any path which is contained in I is also fully contained in either J or K. We now only need to compute $\max_b A_b^E$ to find the optimal price assignment by simple backtracking.

LEMMA 2.1. *The above algorithm finds an optimal solution for any instance of G-SUSP with nested paths and integral valuations in time $O(n^3W^2)$, where $W = \max_{P\in\mathcal{P}} w(P)$.*

The pseudopolynomial time algorithm can easily be turned into an FPTAS for the problem. To this end, let $\alpha = nm/(\varepsilon W)$ and define scaled maximum prices $w'(P) = \lfloor \alpha w(P) \rfloor$ for all paths. Also, define $w''(P) = \alpha^{-1}w'(P)$ to undo the scaling step without respect to the applied rounding. Since it immediately follows that $w(P) - w''(P) \leq \alpha^{-1}$ for any P we can compare total profit of the original optimal solution *Opt* and an optimal solution *Opt''* under the rounded valuations and get that $prof(Opt) - prof(Opt'') \leq nm\alpha^{-1} = \varepsilon \cdot W \leq \varepsilon \cdot prof(Opt)$. This is due to the fact that we can obtain a solution under the rounded valuations by taking the original optimal solution and reducing the price of each edge by α^{-1}. In this solution, all paths that give any profit in the optimal solution will still do so. On the other hand, each of the n paths contains at most m edges, bounding our loss on a single path by $m \cdot \alpha^{-1}$. For polynomial running time observe that after scaling no declaration with value higher than nm/ε exists.

THEOREM 2.2. *For any instance of G-SUSP with nested paths the FPTAS described above achieves approximation ratio $(1-\varepsilon)$ in time $O(n^5m^2\varepsilon^{-2})$.*

3 The Tollbooth Problem

We now turn to the general G-SUSP, which was termed *tollbooth problem* in [11], since it can be thought of as setting up tollbooths on a highway system. In general it models the problem of defining the prices for direct connections in any public transportation system or network. Our focus here will be laid on rather sparse instances, i.e., we will especially be interested in graphs with constant maximum degree, paths of bounded length and a bounded number of requests per edge. We show APX-hardness by a reduction from MAX-2SAT(3). Remember that the MAX-2SAT problem is defined by a set of variables $Var = \{x_1, \ldots, x_n\}$ and a collection of disjunctive clauses of at most 2 literals, where each literal is a variable or negated variable from Var. We want to find a truth assignment $t : Var \rightarrow \{0, 1\}$ that maximizes the number of satisfied clauses. MAX-2SAT(3) is the special case in which the number of occurrences of each literal is bounded by 3. MAX-2SAT(3) is APX-hard [3].

For each appearance of a literal in the SAT instance we define a literal component as found in Figure 2. Literal components are completely similar to the weight components that were used in the previous section. We now simply set the value of all paths to 1. As before we will w.l.o.g. not assign prices to the individual edges of a literal component but only to the component itself, assuming that the price is split evenly among the edges. We note that a literal component $\mathcal{L}$ gives maximum profit 2 under price assignment c if and only if $c(\mathcal{L}) = 1$ or $c(\mathcal{L}) = 2$. If two literals appear in the same clause, their corresponding literal components will be combined into a clause component as depicted in Figure 2. We assume w.l.o.g. that all clauses have length 2. Clauses that consist of only a single literal l are transformed into clauses of type $(l \vee y) \wedge (\overline{y} \vee \overline{y})$, where y is a newly added variable. The following lemma states that clause components indeed model the behavior of clauses in the SAT-instance.

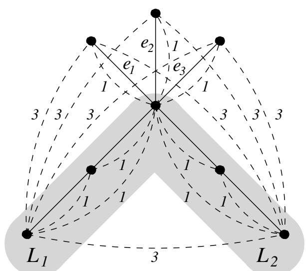

Figure 2: Two literal components $\mathcal{L}_1$ and $\mathcal{L}_2$ are combined into a clause component.

LEMMA 3.1. *Let $\mathcal{C}$ be a clause component with literal components $\mathcal{L}_1$ and $\mathcal{L}_2$. The maximum profit obtainable from $\mathcal{C}$ is 25. Profit 25 is obtained under price assignment c if and only if $c(\mathcal{L}_i) = 2$ and $c(\mathcal{L}_j) = 1$, $\{i, j\} = \{1, 2\}$, or $c(\mathcal{L}_1) = c(\mathcal{L}_2) = 2$. $\mathcal{C}$ gives profit 24 if $c(\mathcal{L}_1) = c(\mathcal{L}_2) = 1$.*

The main ideas for the proof of Lemma 3.1 are not much different from those needed to treat the weight components of the previous section. As the length of the proof, however, tends to grow with the complexity of the component in question, details are omitted in this extended abstract. Figure 3 shows the maximum profit obtainable from a clause component as a function of the prices assigned to its literal components. Depending on the prices assigned to both literal components profit from a clause component is maximized by finding the optimal price for edges e_1, e_2 and e_3 (see Fig. 2). A precise analysis is omitted due to space reasons.

We will now define the complete G-SUSP instance for our reduction. For each clause $c_i = (x_j \vee x_k)$ we construct a clause component $\mathcal{C}_i$ on literal components $\mathcal{L}_h(x_j)$

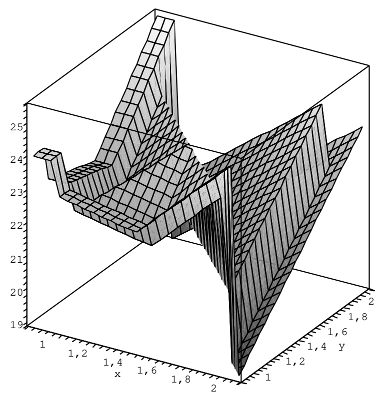

Figure 3: Maximum profit from a clause component as a function of $c(\mathcal{L}_1), c(\mathcal{L}_2) \in [1, 2]$.

and $\mathcal{L}_l(x_k)$, $h, l \in \{0, 1, 2\}$, as described above. We then add *dummy literal components* until for each variable x_i exactly 6 literal components $\mathcal{L}_0(x_i), \mathcal{L}_1(x_i), \mathcal{L}_2(x_i)$ and $\mathcal{L}_0(\overline{x_i}), \mathcal{L}_1(\overline{x_i}), \mathcal{L}_2(\overline{x_i})$ exist. A connected graph is obtained by identifying the end vertex of a literal component $\mathcal{L}_h(x_k)$ with the central vertex of a clause component $\mathcal{C}_l$ containing literal component $\mathcal{L}_h(\overline{x_k})$ in a cyclic fashion, i.e., we continue by connecting $\mathcal{L}_h(\overline{x_k})$ with $\mathcal{L}_{h+1 \mod 3}(x_k)$, $\mathcal{L}_{h+1 \mod 3}(x_k)$ with $\mathcal{L}_{h+1 \mod 3}(\overline{x_k})$ and so on. Additionally 6 *literal exclusion paths* $P_1^i, \dots, P_6^i$ for each variable x_i are defined. Every literal exclusion path P contains the 4 edges belonging to two of the literal components that were just joined together and has value $w(P) = 3$.

A price assignment c on the resulting graph is said to be *integral* if $c(\mathcal{L}) \in \{1, 2\}$ for all literal components $\mathcal{L}$. We say that an integral price assignment is *SAT-feasible* if $c(\mathcal{L}_0(x_i)) = c(\mathcal{L}_1(x_i)) = c(\mathcal{L}_2(x_i))$, $c(\mathcal{L}_0(\overline{x_i})) = c(\mathcal{L}_1(\overline{x_i})) = c(\mathcal{L}_2(\overline{x_i}))$ and $c(\mathcal{L}_0(x_i)) \neq c(\mathcal{L}_0(\overline{x_i}))$ for all i. The intuition behind *SAT-feasibility* is quite obvious. We can associate a SAT-feasible price assignment c with a truth assignment t for the original SAT instance by setting $t(x_i) = 1$ if $c(\mathcal{L}_0(x_i)) = 2$ and $t(x_i) = 0$ else. On the constructed graph SAT-feasible prices result in maximum profit from all dummy literal components and literal exclusion paths. Profit from each clause component is 24 or 25 depending on whether at least one of the contained literal components has price 2. Obviously, total profit of price assignment c is then directly related to the number of clauses satisfied by t.

The main step towards our hardness result lies in proving that we can transform in polynomial time an arbitrary price assignment c on the constructed graph into a SAT-feasible price assignment c^* of no smaller profit. This also yields that the optimal price assignment can be assumed to be SAT-feasible and, thus, gives us an easy way of upper bounding the optimal profit obtainable on the constructed G-SUSP instance. Given any price assignment c, the following transformation returns a SAT-feasible price assignment c^*.

1. Define c' by $c'(\mathcal{L}) = 1$ if $c(\mathcal{L}) < 1$, $c'(\mathcal{L}) = 2$ if $c(\mathcal{L}) > 2$ and $c'(\mathcal{L}) = c(\mathcal{L})$ else.
2. Let $c'' = c'$ and iterate over all literal exclusion paths P. Let P connect literal components $\mathcal{L}_1$, $\mathcal{L}_2$ and $c''(\mathcal{L}_1) \leq c''(\mathcal{L}_2)$. If $c''(\mathcal{L}_1) + c''(\mathcal{L}_2) > 3$ set $c''(\mathcal{L}_1) = 1$.
3. Let c^* be the SAT-feasible price assignment that minimizes
$$|\{\mathcal{L} \mid c''(\mathcal{L}) > 1.75 \wedge c^*(\mathcal{L}) = 1\}|.$$

LEMMA 3.2. *For price assignments c and c^* as defined in the above transformation it holds that $prof(c^*) \geq prof(c)$.*

Sketch of Proof: The full proof of this fact is omitted here. We just briefly describe the main ideas needed to show that profit does not decrease in any of the individual steps.

Step 1: It can be shown that profit does not decrease on any of the components of the graph. For a dummy literal component $\mathcal{L}$ this is straightforward, since profit becomes maximal if $c'(\mathcal{L}) \neq c(\mathcal{L})$. Then consider a literal exclusion path P connecting literal components $\mathcal{L}_1$ and $\mathcal{L}_2$. If $c(\mathcal{L}_1) + c(\mathcal{L}_2) \leq 3$, we observe that $c(\mathcal{L}_1) + c(\mathcal{L}_2) \leq c'(\mathcal{L}_1) + c'(\mathcal{L}_2) \leq 3$, which gives the claim. If $c(\mathcal{L}_1)+c(\mathcal{L}_2) > 3$ then profit from P under c is 0 and the claim follows trivially. For clause components we can apply a similar, but more lengthy, argument.

Step 2: Price assignment c'' is constructed by iterating over all literal exclusion paths and modifying c' locally. We consider a single iteration and prove that profit can only be increased by the modifications. Let P be literal exclusion path connecting literal components $\mathcal{L}_1$ and $\mathcal{L}_2$, where w.l.o.g. $c'(\mathcal{L}_1) \leq c'(\mathcal{L}_2)$. If $c'(\mathcal{L}_1) + c'(\mathcal{L}_2) \leq 3$ nothing is changed. Assume then that $c'(\mathcal{L}_1) + c'(\mathcal{L}_2) > 3$. Thus, we have $c''(\mathcal{L}_1) = 1, c''(\mathcal{L}_2) = c'(\mathcal{L}_2)$. The profit under c' and c'' may differ in the following 3 places: the clause component $\mathcal{C}$ in which literal component $\mathcal{L}_1$ is contained, literal exclusion path P and the second literal exclusion path Q in which $\mathcal{L}_1$ is contained. We refer to the change of profit in these places as $\Delta_{\mathcal{C}}$, Δ_P, Δ_Q and note that the total change of profit caused by changing $c'(\mathcal{L}_1)$ can be written as $\Delta = \Delta_{\mathcal{C}} + \Delta_P + \Delta_Q$. We will bound each summand individually.

Consider literal exclusion path P. Since $c'(\mathcal{L}_1) + c'(\mathcal{L}_2) > 3$ it follows that P gives profit 0 under price assignment c'. From $c'(\mathcal{L}_1) \leq c'(\mathcal{L}_2)$ we conclude that $c'(\mathcal{L}_2) \geq 3/2$. We also observe that $c'(\mathcal{L}_2) \leq 2$ by construction. With $c''(\mathcal{L}_1) = 1$ path P then gives profit at least $5/2$ under c'' and we have that $\Delta_P \geq 5/2$. For literal component $\mathcal{L}_1$ we have that $c'(\mathcal{L}_1) - c''(\mathcal{L}_1) \leq 1$ and, thus, $\Delta_Q \geq -1$. For clause component $\mathcal{C}$ we can argue that $\Delta_{\mathcal{C}} \geq -3/2$. Details of this part of the proof are omitted. We conclude that $\Delta \geq -3/2 + 5/2 - 1 = 0$.

Step 3: We first note that the SAT-feasible assignment c^* can be constructed locally, i.e., considering only the literal exclusion paths belonging to one variable at a time. We will also follow this local approach to show that total profit does not decrease. For variable x_i we define $\mathcal{X}_i = \{\mathcal{L}_0(x_i), \mathcal{L}_1(x_i), \mathcal{L}_2(x_i)\}$, $\overline{\mathcal{X}}_i = \{\mathcal{L}_0(\overline{x_i}), \mathcal{L}_1(\overline{x_i}), \mathcal{L}_2(\overline{x_i})\}$ and let

$$\alpha_i = |\{\mathcal{L} \mid \mathcal{L} \in (\mathcal{X}_i \cup \overline{\mathcal{X}}_i) \wedge c''(\mathcal{L}) > 1.75 \wedge c^*(\mathcal{L}) = 1\}|$$

denote the number of *problematic* literal components belonging to variable x_i. We start by arguing that $\alpha_i \in \{0, 1\}$. To this end, let $B = \{\mathcal{L} \mid \mathcal{L} \in (\mathcal{X}_i \cup \overline{\mathcal{X}}_i) \wedge c''(\mathcal{L}) > 1.75\}$. If $B \subseteq \mathcal{X}_i$ or $B \subseteq \overline{\mathcal{X}}_i$ then we can obviously define a SAT-feasible c^* such that $\alpha_i = 0$. Assume then that $B \cap \mathcal{X}_i \neq \emptyset$ and $B \cap \overline{\mathcal{X}}_i \neq \emptyset$. From the construction of c'' we know that B cannot contain 2 literal components that are connected by a literal exclusion path. Using this we can argue that $|B \cap \mathcal{X}_i| = |B \cap \overline{\mathcal{X}}_i| = 1$. This gives the claim.

Let now Δ_i denote the change in profit in all literal exclusion paths belonging to variable x_i going from price assignment c'' to c^*. We will show that $\Delta_i \geq \alpha_i$. If $\alpha_i = 0$ this is clear, because profit from all literal exclusion paths becomes maximal under c^*. If $\alpha_i = 1$ we can use a similar argument as above to isolate two literal exclusion paths that give profit at most $5/2$ under c''. This implies that profit from all literal exclusion paths of variable x_i is bounded by 17 under c'' and becomes 18 under c^*.

For a clause component $\mathcal{C}$ let $\Delta_{\mathcal{C}}$ denote the change in profit. We can argue that $\Delta_{\mathcal{C}} \geq 0$ if $\mathcal{C}$ does not contain a problematic literal component and $\Delta_{\mathcal{C}} \geq -1$ if it does. We omit the rather technical details of this argument. It follows, however, that at most $\sum_i \alpha_i$ clause components can have negative $\Delta_{\mathcal{C}}$-values. This finishes the proof. $\square$

For a given MAX-2SAT(3) instance I_{SAT} with m clauses over n variables the optimal truth assignment can immediately be turned into an optimal solution for our G-SUSP instance with profit $18n + 24m + 2d + 43v + opt(I_{SAT})$, where $opt(I_{SAT})$ refers to the maximum number of satisfiable clauses in the original SAT formula, v is the number of variables added due to clauses of length 1 and d denotes the number of dummy literal components in the graph. We note that $opt(I_{SAT}) \geq m/2$ and, thus, the expression $18n + 24m + 2d + 43v$ can be upper bounded by $133 \cdot opt(I_{SAT})$. Hence, for any ε a $(1 - \varepsilon/134)$-approximation algorithm for G-SUSP combined with the above transformation into SAT-feasible solutions yields a corresponding $(1 - \varepsilon)$-approximation for MAX-2SAT(3).

THEOREM 3.1. *G-SUSP is APX-hard even if the length of each path is bounded by a constant $\ell \geq 4$, the valuations* *for all paths are of constant size, there are only* 3 *distinct price levels, there is at most a single offer for each possible path, the number of paths in which each edge is contained is bounded by a constant* $B \geq 8$ *and the underlying graph's maximum degree is bounded by a constant* $c \geq 7$.

The above APX-hardness result is in a sense best possible. Guruswami et al. present a pseudopolynomial time algorithm for the highway problem with paths of constant length. This approach also yields an FPTAS and can in fact easily be generalized to the case of constant degree trees instead of a line. Hence, APX-hardness is lost if we simultaneously require a tree and a maximum degree bounded by a constant.

4 An $O(\log \ell + \log B)$-approximation

We now deal with the design of approximation algorithms for SUSP. Our algorithms are especially suitable for sparse problem instances. We therefore assume that the size of each set is bounded by ℓ and that none of the goods appear in more than B sets. We let $\delta(S) = w(S)/|S|$ refer to the relative *price per good* offered by set S. The following algorithm is based on a partitioning approach.

1. Round all $\delta(S)$ to powers of 2 and let $2^k = \max_S \delta(S)$. Then partition $\mathcal{S}$ into $\mathcal{S}_0, \dots, \mathcal{S}_t$, where
$$\mathcal{S}_i = \left\{ S \in \mathcal{S} \mid \delta(S) \in \left\{ 2^{i+j\cdot\lceil \log(2\ell^2 B)\rceil} \mid j \in \mathbb{N}_0 \right\} \right\}$$
and $t = \lceil \log(2\ell^2 B)\rceil - 1$.
2. For all i let $\mathcal{S}_i^* = \mathcal{S}_i$. Then remove set $S \in \mathcal{S}_i^*$ from $\mathcal{S}_i^*$ if it intersects with a set $T \in \mathcal{S}_i^*$ and $\delta(S) < \delta(T)$. Define prices c_i as $c_i(e) = \delta(S)$ if e is still contained in a set $S \in \mathcal{S}_i^*$ and $c_i(e) = 0$ else.
3. Return $c = \operatorname{argmax}\{prof(c_i) \mid c_i, i = 0, \dots, t\}$.

The crucial step in the analysis of the above algorithm consists of showing that we do not lose too much potential profit by removing sets in Step 2. This claim is formalized in the following lemma.

LEMMA 4.1. *Let* $\mathcal{S}_i$ *and* $\mathcal{S}_i^*$ *be defined as in the above algorithm. Then* $\sum_{S\in\mathcal{S}_i^*} |S|\delta(S) \geq \sum_{S\in\mathcal{S}_i\setminus\mathcal{S}_i^*} |S|\delta(S)$.

Proof. We will construct the following collection of trees. We start with a single vertex $v(S)$ for each set S. If set S_1 is removed from $\mathcal{S}_i^*$ because of its intersection with S_2, we assign vertex $v(S_1)$ as a child to vertex $v(S_2)$. Since we require δ-values to be strictly increasing in this procedure, it follows that we do not construct cycles. In the resulting collection of trees $\mathcal{S}_i^*$ corresponds to the set of all root vertices, while $\mathcal{S}_i\setminus\mathcal{S}_i^*$ contains the sets corresponding to inner vertices. We assign to each vertex $v(S)$ a label $\alpha(v(S)) = |S|\delta(S)$.

For a single tree with root r and inner vertices W it holds that $\alpha(r) \geq \sum_{v\in W}\alpha(v)$. To see this, consider some vertex v with children $u_1, \dots, u_j$ and let v correspond to set S. Then $\alpha(v) \geq \delta(S)$. From the fact that S contains at most ℓ goods and each of these goods is itself contained in at most $B-1$ further sets, we can upper bound the number of vertices assigned as children to v, thus, $j \leq \ell B$. From our partitioning step we know that for any set S_i corresponding to vertex u_i it must be true that $\delta(S_i) \leq (2\ell^2 B)^{-1} \cdot \delta(S)$. Consequently $|S_i|\delta(S_i) \leq (2\ell B)^{-1}\delta(S)$. Thus,
$$\sum_{i=1}^{j} \alpha(u_i) \leq \ell B \cdot (2\ell B)^{-1} \cdot \delta(S) \leq \frac{1}{2}\alpha(v).$$
Let $L(j)$ be the set of all vertices of the tree with distance j from the root. By induction it readily follows that $\sum_{v\in L(j)} \alpha(v) \leq 2^{-j}\alpha(r)$. Summing up immediately gives the claim. Since the above holds for every tree in the collection this finishes the proof of the lemma. □

Applying this immediately yields the desired approximation guarantee.

THEOREM 4.1. *The above algorithm computes an* $O(\log \ell + \log B)$*-approximation to SUSP.*

Proof. For any set S let $\delta(S)$ refer to the rounded value after Step 1 of the algorithm. We observe that $\delta(S) = \delta(T)$ for any $S, T \in \mathcal{S}_i^*$ with $S \cap T \neq \emptyset$. Therefore, it holds that $prof(c_i) = \sum_{S\in\mathcal{S}_i^*} |S|\delta(S)$ for $0 \leq i \leq t$. Choosing $c = \operatorname{argmax}\{prof(c_i) \mid c_i, i = 0, \dots, t\}$ and applying Lemma 4.1 yields
$$\begin{aligned} prof(c) &\geq \frac{1}{t+1}\sum_{i=0}^{t}\sum_{S\in\mathcal{S}_i^*} |S|\delta(S) \\ &\geq \frac{1}{2(t+1)}\sum_{i=0}^{t}\sum_{S\in\mathcal{S}_i} |S|\delta(S) \\ &= \frac{1}{2(t+1)}\sum_{S\in\mathcal{S}} |S|\delta(S) \geq \frac{1}{4(t+1)} opt, \end{aligned}$$
where the last inequality takes into account the rounding applied to δ-values in Step 1. Since $t = O(\log(\ell B))$, the claim follows. □

A lower bound. It has already been observed in [11] that using the sum of values of all sets as an upper bound on the optimal solution cannot result in an approximation guarantee better than $\Omega(\log B)$. As a tight example consider a single good e and sets $S_1, \dots, S_B$ with $w(S_i) = 1/i$.

5 An $O(\ell^2)$-approximation

Given that each set is of size at most ℓ, we will derive an $O(\ell^2)$-approximation. It was already observed in [11] that bounding the optimal solution's value by the sum of values of all sets cannot give an approximation ratio better than $\log B$. Thus, we will need to introduce a new upper bounding technique. We proceed in two steps. First, we consider a restriction of SUSP in which we count profit only from those sets for which the total price is split evenly among the contained goods. This is formalized in the following definition.

DEFINITION 3. *The* smoothed *single-minded unlimited supply pricing problem (s-SUSP) is defined on the same input $(U, \mathcal{S}, w)$ as the general SUSP. We want to find the price assignment c^* that maximizes $\sum_{S\in\Lambda(c)} \sum_{e\in S} c(e)$, where $\Lambda(c) = \{S \in \mathcal{S} \mid c(e) \leq w(S)/|S| \; \forall\; e \in S\}$.*

Given a SUSP-instance $\mathcal{I}$ we can compare the optimal profit $opt(\mathcal{I})$, $opt_s(\mathcal{I})$ under the objective functions of SUSP and s-SUSP, respectively. It is straightforward to note that $opt_s(\mathcal{I}) \geq (1/\ell)opt(\mathcal{I})$. On the other hand, consider an arbitrary price assignment c and let $prof(c)$ and $prof_s(c)$ denote its profit under these two objectives. Then clearly $prof(c) \geq prof_s(c)$, since $S \in \Lambda(c)$ implies that $\sum_{e\in S} c(e) \leq w(S)$. It follows that in order to obtain the desired $O(\ell^2)$-approximation to SUSP it is sufficient to compute an $O(\ell)$-approximation to s-SUSP on the same input.

To find such a solution to s-SUSP we start by computing the optimal price separately for each good, i.e., assuming all other prices are 0 and, thus, no bidder's budget is exceeded by the price of any of the remaining goods. Clearly, setting prices independently from each other will lead to *conflicts* in the resulting price assignment c, where we consider any set S with $e_1, e_2 \in S$ and $c(e_1) \leq w(S)/|S| < c(e_2)$ a conflict. The main task lies in resolving these conflicts and incorporating the individual prices into a consistent s-SUSP solution.

Consider a single good e. For a given price assignment c let $\mathcal{C}_c(e) = \{S \mid \exists\, e' : e, e' \in S \wedge c(e) \leq w(S)/|S| < c(e')\}$ refer to the sets that contain e and are not violated by its price, but become conflicting because of some other good e'. Analogously, the non-conflicting sets are denoted by $\mathcal{N}_c(e) = \{S \mid e \in S \wedge c(e') \leq w(S)/|S| \,\forall\, e' \in S\}$. Finally, we let $\mathcal{C}_c = \bigcup_{e\in U} \mathcal{C}_c(e)$ refer to all conflicting sets under c. The following hypergraph structure models conflicts between different goods.

DEFINITION 4. *Let c be an arbitrary price assignment. The corresponding* conflict graph *is a labeled directed multi-hypergraph $H = (V_H, E_H)$ with a vertex v_e for each $e \in U$ and a directed hyperedge $f_S = (P, Q)$ for every conflicting set $S \in \mathcal{C}_c$, where $P = \{v_e \mid S \in \mathcal{C}_c(e)\}$ and $Q = S \backslash P$. For each $f_S = (P, Q)$ we let $f_S \in Out(v_e)$ for all $v_e \in P$, $f_S \in In(v_e)$ for all $v_e \in Q$. We define labels $\alpha(v_e) = c(e) \cdot |\mathcal{N}_c(e)|$ and $\alpha(v_e, f_S) = c(e)$ for all $v_e \in V_H$ s.t. $f_S \in Out(v_e)$, and let $\alpha(f_S) = \sum_{v_e\in P} \alpha(v_e, f_S)$ for all $f_S = (P, Q) \in E_H$. Finally, we set $c(v_e) = c(e)$ for all $v_e \in V_H$.*

The intuition behind a price assignment's conflict graph is quite simple. Each vertex corresponds to a good in the s-SUSP instance, each hyperedge represents a conflicting set. Label $\alpha(v_e)$ of a vertex $v_e \in V_H$ describes the profit made by the corresponding good e under price assignment c on non-conflicting sets only. Hyperedge f_S is labeled with the profit $\alpha(f_S)$ that could be obtained by the corresponding set S assuming there were no violations. Label $\alpha(v_e, f_S)$ denotes the contribution of good e to the profit that is lost because S is conflicting. For ease of notation, let $H = (V_H, E_H)$, $V' \subseteq V_H$ and $E' \subseteq E_H$ and define $\alpha(V') = \sum_{v\in V'} \alpha(v)$, $\alpha(E') = \sum_{f\in E'} \alpha(f)$ and $\alpha(H) = \alpha(V_H) + \alpha(E_H)$. The following algorithm is based on the idea of transforming the conflict graph of an initial price assignment obtained by assigning optimal prices to each good separately.

1. For all $e \in E$ find the optimal price $c^*(e)$ under the s-SUSP objective assuming all other prices are set to 0.
2. Construct conflict graph $H_1 = (V_1, E_1)$ for price assignment c^*.
3. Let $V_1 = \{v_1, \ldots, v_m\}$ where $c(v_1) \leq \cdots \leq c(v_m)$. For $i = 1, \ldots, m$ check if
$$\sum_{f\in In(v_i)} \alpha(f) > \frac{1}{2} \sum_{f\in Out(v_i)} \alpha(v_i, f).$$
In this case replace each $f = (P, Q) \in Out(v_i)$ by $f' = (P\backslash\{v_i\}, Q)$ and update $\alpha(f')$. Refer to the resulting graph as $H_2 = (V_2, E_2)$, where $V_2 = V_1$.
4. Let $R = \{v \in V_2 \mid Out(v) = \emptyset\}$ and $\mathcal{E} = \{(P, Q) \in E_2 \mid Q \subseteq R\}$. If $\alpha(\mathcal{E}) > \alpha(R)$ then for all $(P, Q) \in \mathcal{E}$ and all $v \in P$ set $\alpha(v) = \alpha(v) + \sum_{f\in Out(v)\cap\mathcal{E}} \alpha(v, f)$, for all $v \in R$ set $\alpha(v) = c(v) = 0$. Finally, remove all edges from graph H_2 and let $H_3 = (V_3, E_3)$ with $V_3 = V_2$, $E_3 = \emptyset$ denote the resulting graph.
5. For good $e \in U$ with corresponding vertex v_e in H_3 set $c(e) = c(v_e)$.

After the conflict graph of the initial price assignment c^* is constructed, the algorithm proceeds roughly as follows. In Step 3 we transform the conflict graph in order to ensure some purely technical requirement that will be of importance later on. Especially, note that the resulting hypergraph might no longer be a legal conflict graph of some price assignment on our s-SUSP instance. Step 4 of the algorithm takes care of this problem. Here, we first identify a set of hyperedges that carry a large portion of the overall profit, namely $\mathcal{E}$. We then decide whether to make these hyperedges non-conflicting by reducing the labels of vertices in R. Finally, we remove all edges that are still present at this point of time. The following lemma states that this again results in a legal conflict graph, if we remove those sets from the s-SUSP instance whose corresponding hyperedges have been removed at the very end of the algorithm.

LEMMA 5.1. *Let $(U, \mathcal{S}, w)$ be the original s-SUSP instance. Then graph H_3 is conflict graph for price assignment c as defined by the algorithm on s-SUSP instance $(U, \mathcal{S}', w)$, where $\mathcal{S}' \subseteq \mathcal{S}$.*

Proof. If $\alpha(\mathcal{E}) \leq \alpha(R)$ in Step 4, no vertex label is modified and the algorithm simply removes all hyperedges from the conflict graph. In this situation the observation is rather trivial, since we can set $\mathcal{S}' = \mathcal{S} \backslash \mathcal{C}$, where $\mathcal{C}$ contains all conflicting sets.

Else let R and $\mathcal{E}$ be defined as in the algorithm and $R' = \{v \in V_2 \,|\, Out(v) \cap \mathcal{E} \neq \emptyset\}$. Sets corresponding to hyperedges $f \in \mathcal{E}$ are by definition violated only by the prices of goods corresponding to vertices $v \in R$. Hence, as all these prices are set to 0, the sets corresponding to hyperedges in $\mathcal{E}$ become non-conflicting. By definition of the conflict graph labels of vertices $v \in R'$ must then be updated by adding $\alpha(v, f)$ for each $f \in Out(v) \cap \mathcal{E}$, as done by the algorithm. As before, the remaining hyperedges are removed from the conflict graph and we get $\mathcal{S}'$ by removing their corresponding sets from $\mathcal{S}$. $\square$

Conflict graph H_3 does not contain hyperedges and, thus, by Lemma 5.1 price assignment c does not cause any conflicts on the remaining sets $\mathcal{S}'$. Hence, c realizes total profit $prof_s(c) = \alpha(H_3)$ under the s-SUSP objective. We proceed by comparing $\alpha(H_3)$ to the value $\alpha(H_1)$ of the initial conflict graph.

LEMMA 5.2. *It holds that $\alpha(H_2) \geq \frac{1}{2\ell - 1}\alpha(H_1)$.*

Proof. As in the algorithm let $c(v_1) \leq \cdots \leq c(v_m)$. By $In^i(v_i)$ and $Out^i(v_i)$ we refer to the sets of incoming and outgoing hyperedges of vertex v_i at the beginning of the i-th iteration, i.e., just before vertex v_i is treated in Step 3 of the algorithm. By $\alpha^i(f)$ we denote the α-value of hyperedge f at this time. The important observation is that by the construction of H_1 we have that

$$(P, Q) \in E_1 \;\Rightarrow\; c(v) < c(w) \quad \forall\, v \in P, w \in Q,$$

since otherwise price $c(w)$ could not violate a set's value which $c(v)$ does not. This immediately implies that $In^i(v_i) \cap Out^j(v_j) = \emptyset$ if $j > i$. Thus, $In^i(v_i) \subseteq E_2$ and $\alpha^i(f) = \cdots = \alpha^{m+1}(f)$ for any $f \in In^i(v_i)$, because no hyperedge from $In^i(v_i)$ can be removed or assigned a new label in a later step. Now let W be the set of vertices v_i that are removed from their outgoing hyperedges $Out^i(v_i)$ by the algorithm. Then

$$\begin{aligned}
\sum_{f \in E_2} \alpha(f) &\geq \alpha^{m+1}\Big(\bigcup_{v_i \in W} In^i(v_i)\Big) \\
&\geq \frac{1}{\ell - 1} \sum_{v_i \in W} \sum_{f \in In^i(v_i)} \alpha^{m+1}(f) \\
&= \frac{1}{\ell - 1} \sum_{v_i \in W} \sum_{f \in In^i(v_i)} \alpha^{i}(f) \\
&> \frac{1}{2(\ell - 1)} \sum_{v_i \in W} \sum_{f \in Out^i(v_i)} \alpha(v, f) \\
&= \frac{1}{2(\ell - 1)} \left(\alpha(E_1) - \alpha(E_2)\right),
\end{aligned}$$

where the second inequality is due to the fact that $|Q| \leq \ell - 1$ for all $(P, Q) \in E_1$ and the fourth inequality holds by the construction of H_2. It follows that

$$\alpha(E_2) > \frac{1}{2\ell - 1}\alpha(E_1)$$

and, since $\alpha(V_1) = \alpha(V_2)$, this finishes the proof. $\square$

LEMMA 5.3. *It holds that $\alpha(H_3) \geq \frac{1}{3}\alpha(H_2)$.*

Proof. As in the algorithm, let $R = \{v \in V_2 \,|\, Out(v) = \emptyset\}$ and $\mathcal{E} = \{(P, Q) \in E_2 \,|\, Q \subseteq R\}$. It suffices to show that

$$\alpha(\mathcal{E}) \geq \frac{1}{2}\alpha(E_2).$$

Having this, it readily follows that

$$\begin{aligned}
\max\{\alpha(\mathcal{E}), \alpha(R)\} &\geq \frac{1}{3}(2\alpha(\mathcal{E}) + \alpha(R)) \\
&\geq \frac{1}{3}(\alpha(E_2) + \alpha(R))
\end{aligned}$$

and, since the algorithm removes either $\mathcal{E}$ or R from the graph,

$$\begin{aligned}
\alpha(H_3) &= \alpha(H_2) - \alpha(E_2) - \alpha(R) \\
&\quad + \max\{\alpha(\mathcal{E}), \alpha(R)\} \\
&\geq \alpha(H_2) - \frac{2}{3}\left(\alpha(E_2) + \alpha(R)\right) \geq \frac{1}{3}\alpha(H_2).
\end{aligned}$$

We will now show the above claim. Let $A = \{v \in V_2 \,|\, In(v) = \emptyset\}$ and $B = V_2 \backslash (A \cup R)$. W.l.o.g. assume that $B = \{v_1, \ldots, v_k\}$, where $c(v_1) \leq \cdots \leq c(v_k)$. Initialize $X^0 = \bigcup_{v \in A} Out(v)$ and $Y^0 = \emptyset$. Define label $\gamma(f) = \alpha(f)$ for $f \in X^0$ and $\gamma(f) = 0$ else. Again let $\gamma(F) = \sum_{f \in F} \gamma(f)$ for any $F \subseteq E_2$. Obviously, $\gamma(X^0) \geq \gamma(Y^0)$. For $j = 1, \ldots, k$ we repeat the following procedure.

1. Set $X^j = \big(X^{j-1} \backslash In(v_j)\big) \cup Out(v_j)$.
2. Set $Y^j = Y^{j-1} \cup In(v_j)$
3. For each $f \in Out(v_j)$ set $\gamma(f) = \gamma(f) + \alpha(v_j, f)$.

By the construction of H_2 we know that

$$\sum_{f \in Out(v_j)} \alpha(v_j, f) \geq 2 \cdot \sum_{f \in In(v_j)} \alpha(f)$$

for all $v_e \in B$. Since $\gamma(f) \leq \alpha(f)$ at all times, we have

$$\begin{aligned} \gamma(X^j) - \gamma(X^{j-1}) \quad &\geq \quad \sum_{f \in Out(v_j)} \alpha(v_j, f) \\ &- \sum_{f \in In(v_j)} \alpha(f) \geq \sum_{f \in In(v_j)} \alpha(f) \end{aligned}$$

and

$$\gamma(Y^j) - \gamma(Y^{j-1}) \leq \sum_{f \in In(v_j)} \gamma(f) \leq \sum_{f \in In(v_j)} \alpha(f).$$

Thus, the inequality $\gamma(X^j) \geq \gamma(Y^j)$ must hold for any j. Finally, we observe that $X^k \cup Y^k = E_2$, $\gamma(X^k) = \alpha(X^k)$, $\gamma(Y^k) = \alpha(Y^k)$ and $X^k = \mathcal{E}$. It follows that

$$\alpha(\mathcal{E}) = \alpha(X^k) \geq \frac{1}{2}\left(\alpha(X^k) + \alpha(Y^k)\right) = \frac{1}{2}\alpha(E_2),$$

finishing the proof. □

Thus, we lose at most a factor of ℓ by incorporating the prices initially assigned to individual goods into a solution to s-SUSP. This yields the following result.

THEOREM 5.1. *The above algorithm computes an $O(\ell^2)$-approximation to SUSP with sets of size at most ℓ.*

Proof. Let $\mathcal{I}$ be a SUSP instance and let $opt_s(\mathcal{I})$ be the value of an optimal solution under the s-SUSP objective. We argue that the algorithm computes an $O(\ell)$-approximation for s-SUSP on this instance. In the initial price assignment c^* let $prof^*(e) = c^*(e) \cdot |\mathcal{N}_{c^*}(e) \cup \mathcal{C}_{c^*}(e)|$ denote the profit made by good e assuming there were no conflicts. It holds that $\alpha(H_1) = \sum_{e \in U} prof^*(e) \geq opt_s(\mathcal{I})$. Let c denote the prices returned by the algorithm. By Lemmas 5.2 and 5.3,

$$prof_s(c) = \alpha(H_3) \geq \frac{1}{6\ell - 3}\alpha(H_1) \geq \frac{1}{6\ell - 3} opt_s(\mathcal{I}).$$

As we have argued about the relation between SUSP and s-SUSP before, this gives the claim. □

A lower bound. We can view our upper bound as the sum of optimal profits obtainable from single goods. Clearly, using this upper bounding technique we cannot expect to achieve approximation ratios better than $\Omega(\ell)$. Independently of our results, Balcan and Blum [4] recently presented a randomized algorithm with approximation ratio $O(\ell)$ in expectation. This algorithm relies on the same upper bounding technique as ours, but uses randomization to determine a subset of all goods and sets, such that pricing goods separately does not lead to conflicts. Though this algorithm can be derandomized in some cases, it is an interesting open problem to obtain a deterministic $O(\ell)$-approximation in general.

6 Acknowledgments

We would like to thank Berthold Vöcking for proposing this problem and Thorsten Bernholt for an insightful discussion on a closely related topic.

References

[1] G. Aggarwal, T. Feder, R. Motwani and A. Zhu. Algorithms for multi-product pricing. In *Proc. of 31st ICALP*, 2004.

[2] A. Archer, C.H. Papadimitriou, K. Talwar, and É. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proc. of 14th SODA*, 2003.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi. *Complexity and Approximation.* Springer, 1999.

[4] N. Balcan and A. Blum. Approximation Algorithms for Item Pricing. *Technical Report CMU-CS-05-176*, 2005.

[5] Y. Bartal, R. Gonen and N. Nisan. Incentive Compatible Multi-Unit Combinatorial Auctions. In *Proc. of 9th TARK*, 2003.

[6] M. Bouhtou, A. Grigoriev, S. van Hoesel, A. van der Kraaij, M. Uetz and F. Spieksma. Pricing Bridges to Cross a River. Submitted. An extended abstract appeared as Pricing Network Edges to Cross a River in *Proc. of 2nd WAOA*, 2004.

[7] P. Briest, P. Krysta and B. Vöcking. Approximation Techniques for Utilitarian Mechanism Design. In *Proc. of 37th STOC*, 2005.

[8] A. Fiat, A.V. Goldberg, J.D. Hartline and A. Karlin. Competitive Generalized Auctions. In *Proc. of 34th STOC*, 2002.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, 1979.

[10] A.V. Goldberg, J.D. Hartline and A. Wright. *Competitive Auctions and Digital Goods*, In *Proc. of 12th SODA*, 2001.

[11] V. Guruswami, J.D. Hartline, A.R. Karlin, D. Kempe, C. Kenyon and F. McSherry. On Profit-Maximizing Envy-free Pricing. In *Proc. of 16th SODA*, 2005.

[12] J. Hartline and V. Koltun. Near-Optimal Pricing in Near-Linear Time. In *Proc. of WADS*, 2005.

[13] D. Lehmann, L. O'Callaghan and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. In *Proc. of 1st EC*, 1999.

[14] N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *Proc. of 31st STOC*, 1999.

[15] P. Rusmevichientong, B. Van Roy and P. Glynn. A Non-Parametric Approach to Multi-Product Pricing. To appear in *Operations Research.*

A Proofs from Section 3

We present the technical details from the proof of APX-hardness of G-SUSP. Lemma A.1 shows how the profit obtained from a clause component can be expressed as a piecewise linear function that depends solely on the prices assigned to the contained literal components $\mathcal{L}_1$ and $\mathcal{L}_2$. This is due to the fact that the additional edges e_1, e_2 and e_3 (see Fig. 2) are contained only in paths that are part of this component and, thus, their optimal prices are determined by $c(\mathcal{L}_1)$ and $c(\mathcal{L}_2)$. Lemma 3.1 then follows as a simple corollary.

Lemma A.1 *For $c(\mathcal{L}_1), c(\mathcal{L}_2) \in [1,2]$ we can describe $prof(\mathcal{C})$ as a function of $c(\mathcal{L}_1)$ and $c(\mathcal{L}_2)$ as follows:*

$prof(\mathcal{C}) =$
(1) 24*, if* $c(\mathcal{L}_1) = c(\mathcal{L}_2) = 1$
(2) $24 - c(\mathcal{L}_2)$*, if* $1 = c(\mathcal{L}_1) < c(\mathcal{L}_2) \leq \frac{3}{2}$
(3) $15 + 5c(\mathcal{L}_2)$*, if* $1 = c(\mathcal{L}_1), \frac{3}{2} < c(\mathcal{L}_2)$
(4) $18 + 5c(\mathcal{L}_1) - c(\mathcal{L}_2)$*, if* $1 < c(\mathcal{L}_1) \leq c(\mathcal{L}_2) \leq \frac{3}{2}$
(5) $9 + 5c(\mathcal{L}_1) + 5c(\mathcal{L}_2)$*, if* $1 < c(\mathcal{L}_1) < \frac{3}{2} < c(\mathcal{L}_2), c(\mathcal{L}_1) + c(\mathcal{L}_2) \leq 3$
(6) - (9) symmetric to (2) - (5) with $c(\mathcal{L}_1) > c(\mathcal{L}_2)$
(10) $9 + 4c(\mathcal{L}_1) + 4c(\mathcal{L}_2)$*, if* $c(\mathcal{L}_1) + c(\mathcal{L}_2) > 3$

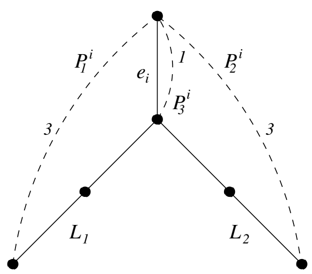

Figure 4: Substructure of a clause component.

Proof. A clause component consists of its literal components and three copies of the substructure found in Figure 4. In order to prove the claim we just need to describe how $c(e_i)$ has to be chosen to maximize total profit. Consider the substructure in Figure 4 consisting of edge e_i and the paths defined on e_i, $\mathcal{L}_1$ and $\mathcal{L}_2$. For any given $c(\mathcal{L}_1) \leq c(\mathcal{L}_2)$ in $[1,2]$ there are only two possible prices that can potentially give maximum profit. These are $c(e_i) = 1$ or $c(e_i) = 3 - c(\mathcal{L}_2)$, respectively. To see this, note, that for any $c(e_i) \leq 1$ edge e_i will contribute $3c(e_i)$ to the substructure's total profit ($c(e_i)$ for each path in which it appears). It follows that any price assignment c with $c(e_i) < 1$ can be improved by setting $c(e_i) = 1$. Now consider c with $c(e_i) > 1$. For $c(e_i) \leq 3 - c(\mathcal{L}_2)$ edge e_i contributes profit $2c(e_i)$ (being counted on two paths). Thus, any price assignment c with $1 < c(e_i) < 3 - c(\mathcal{L}_2)$ can be improved by setting $c(e_i) = 3 - c(\mathcal{L}_2)$. Especially, since $c(\mathcal{L}_2) \leq 2$, it follows that $2c(e_i) \geq 2$. For $c(e_i) > 3 - c(\mathcal{L}_2)$ the contribution becomes $c(e_i)$, since the price for the path containing $\mathcal{L}_2$ exceeds its threshold and edge e_i is counted only once on the path containing $\mathcal{L}_1$. From $c(\mathcal{L}_1) \geq 1$ it follows that e_i can contribute at most 2 on this path and, thus, profit does not decrease by setting $c(e_i) = 3 - c(\mathcal{L}_2)$.

Using this observation it is clear how $c(e_i)$ must be chosen in each case with $c(\mathcal{L}_1) \leq c(\mathcal{L}_2)$. We let $c(e_i) = 3 - c(\mathcal{L}_2)$ whenever $2(3 - c(\mathcal{L}_2)) > 3$ ($\Leftrightarrow c(\mathcal{L}_2) < 3/2$) and $c(e_i) = 1$ otherwise. Cases with $c(\mathcal{L}_1) > c(\mathcal{L}_2)$ are symmetric, thus, $c(e_i) = 1$ or $c(e_i) = 3 - c(\mathcal{L}_1)$. Total profit from clause component $\mathcal{C}$ is then obtained by summing up over 3 copies of the substructure, 2 literal components and path P (see Fig. 2). □

In Section 3 we present a procedure that transforms in polynomial time an arbitrary price assignment c into a SAT-feasible price assignment c^* of no smaller profit. Lemmas A.2, A.3 and A.4 show that none of the three steps of this transformation decreases total profit of the solution. Formally, we have that $prof(c) \leq prof(c') \leq prof(c'') \leq prof(c^*)$.

Lemma A.2 *It holds that* $prof(c') \geq prof(c)$*.*

Proof. We show that profit does not decrease on any literal exclusion path, in any clause or dummy literal component. For dummy literal components this is trivial, since profit from these components becomes maximal if the assigned price is changed.

We now fix some clause component $\mathcal{C}$ with literal components $\mathcal{L}_1$ and $\mathcal{L}_2$. We assume w.l.o.g. that $c(\mathcal{L}_1) \leq c(\mathcal{L}_2)$ and consider all possible cases. By $prof(\mathcal{C})$ and $prof'(\mathcal{C})$ we refer to the total profit under c and c', respectively.

Case (1) $c(\mathcal{L}_1), c(\mathcal{L}_2) < 1 \Rightarrow c'(\mathcal{L}_1) = c'(\mathcal{L}_2) = 1$. The profit from each substructure (Fig. 4) is bounded by 6, thus,

$$prof(\mathcal{C}) \leq 3 \cdot 6 + 3(c(\mathcal{L}_1) + c(\mathcal{L}_2)) \leq 24 = prof'(\mathcal{C}),$$

by summing up over 3 substructures, 2 literal components and the connecting path P.

Case (2) $c(\mathcal{L}_1) < 1 \leq c(\mathcal{L}_2) \leq 2 \Rightarrow c'(\mathcal{L}_1) = 1, c'(\mathcal{L}_2) = c(\mathcal{L}_2)$. From the proof of Lemma A.1 we know that maximum profit from each substructure is obtained by setting $c(e_i) = 1$ or $c(e_i) = 3 - c(\mathcal{L}_2)$ depending on $c(\mathcal{L}_2)$.

In both cases increasing $c(\mathcal{L}_1)$ to 1 gives an increase in profit of $1 - c(\mathcal{L}_1)$ in each substructure. Identical observations are easily made for the literal components and path P. It follows that

$$prof(\mathcal{C}) \leq prof(\mathcal{C}) + 6(1 - c(\mathcal{L}_1)) = prof'(\mathcal{C}).$$

Case (3) $c(\mathcal{L}_1) < 1, c(\mathcal{L}_2) > 2 \Rightarrow c'(\mathcal{L}_1) = 1, c'(\mathcal{L}_2) = 2$. By Lemma A.1 profit from $\mathcal{C}$ is maximal under c' and the claim follows.

Case (4) $1 \leq c(\mathcal{L}_1) \leq 2 < c(\mathcal{L}_2) \Rightarrow c'(\mathcal{L}_1) = c(\mathcal{L}_1), c'(\mathcal{L}_2) = 2$. From observations analogous to those in the proof of Lemma A.1 it follows that under price assignment c it must be $c(e_i) = 3 - c(\mathcal{L}_2)$ in order to obtain maximum profit $3 + 2(3 - c(\mathcal{L}_2)) + c(\mathcal{L}_1)$ from each substructure. (If $c(\mathcal{L}_2) > 3$ we let $c(e_i) = 0$ and the former is an upper bound on the profit from each substructure.) Under c' setting $c'(e_i) = 1$ leads to profit $5 + c(\mathcal{L}_1)$. With $3 - c(\mathcal{L}_2) < 1$ we can conclude that profit increases in each substructure. Literal component $\mathcal{L}_2$ and path P give profit 0 under c and, thus, cannot contribute less under c'.

Case (5) $c(\mathcal{L}_1), c(\mathcal{L}_2) > 2 \Rightarrow c'(\mathcal{L}_1) = c'(\mathcal{L}_2) = 2$. By Lemma A.1 profit from $\mathcal{C}$ is maximal under c' and the claim follows.

We then look at a single literal exclusion path P connecting literal components $\mathcal{L}_1$ and $\mathcal{L}_2$. We distinguish the following 2 cases.

Case (1) $c(\mathcal{L}_1) + c(\mathcal{L}_2) \leq 3$. We observe that $c(\mathcal{L}_1) + c(\mathcal{L}_2) \leq c'(\mathcal{L}_1) + c'(\mathcal{L}_2) \leq 3$ and, since in this case profit from P is just the sum of these prices, it follows that profit under c' is no smaller than under c.

Case (2) $c(\mathcal{L}_1) + c(\mathcal{L}_2) > 3$. Then the profit from P under price assignment c is 0 and can obviously only increase when going to c'.

Hence, $prof(c') \geq prof(c)$. This finishes the proof. □

Lemma A.3 *It holds that $prof(c'') \geq prof(c')$.*

Proof. Price assignment c'' is constructed by iterating over all literal exclusion paths and modifying c' locally. We will consider a single iteration and prove that profit can only be increased by the modifications.

Let P be literal exclusion path connecting literal components $\mathcal{L}_1$ and $\mathcal{L}_2$ and look at the iteration in which P is considered. If $c'(\mathcal{L}_1) + c'(\mathcal{L}_2) \leq 3$ then nothing is changed and obviously profit remains the same. Assume then that $c'(\mathcal{L}_1) + c'(\mathcal{L}_2) > 3$. We let w.l.o.g $c'(\mathcal{L}_1) \leq c'(\mathcal{L}_2)$ and, thus, will have $c''(\mathcal{L}_1) = 1, c''(\mathcal{L}_2) = c'(\mathcal{L}_2)$. The profit under c' and c'' may differ in the following 3 places: the clause component $\mathcal{C}$ in which literal component $\mathcal{L}_1$ is contained, literal exclusion path P and the second literal exclusion path Q in which $\mathcal{L}_1$ is contained. We refer to the change of profit in these places as $\Delta_\mathcal{C}$, Δ_P, Δ_Q and note that the total change of profit caused by changing $c'(\mathcal{L}_1)$ can be written as $\Delta = \Delta_\mathcal{C} + \Delta_P + \Delta_Q$. We will bound each summand individually.

Consider clause component $\mathcal{C}$ that consists of $\mathcal{L}_1$ and some other literal component $\mathcal{L}_3$. We go through all possible cases and apply Lemma A.1. Note, that $c'(\mathcal{L}_1) > 1$.

Case (1) $1 = c'(\mathcal{L}_3) < c'(\mathcal{L}_1) \leq 3/2$. Due to the changed price we jump from Case (2) to Case (1) in Lemma A.1. Hence, $\Delta_\mathcal{C} \geq 24 - (24 - c'(\mathcal{L}_1)) \geq 0$.

Case (2) $c'(\mathcal{L}_3) = 1, c'(\mathcal{L}_1) > 3/2$. We jump from (3) to (1), thus, $\Delta_\mathcal{C} \geq 24 - (15 + 5c'(\mathcal{L}_1)) \geq -1$.

Case (3) $1 < c'(\mathcal{L}_3) \leq c'(\mathcal{L}_1) \leq 3/2$. We jump from Case (4) to Case (6), thus,

$$\begin{aligned}\Delta_\mathcal{C} &= (24 - c'(\mathcal{L}_3)) - (18 + 5c'(\mathcal{L}_3) - c'(\mathcal{L}_1)) \\ &= 6 - 5c'(\mathcal{L}_3) \geq -3/2.\end{aligned}$$

Case (4) $1 < c'(\mathcal{L}_3) < 3/2 < c'(\mathcal{L}_1), c'(\mathcal{L}_1) + c'(\mathcal{L}_3) \leq 3$. We jump from Case (5) to Case (6), thus,

$$\begin{aligned}\Delta_\mathcal{C} &= (24 - c'(\mathcal{L}_3)) - (9 + 5c'(\mathcal{L}_3) + 5c'(\mathcal{L}_1)) \\ &= 15 - 5(c'(\mathcal{L}_1) + c'(\mathcal{L}_3)) - c'(\mathcal{L}_3) \geq -3/2.\end{aligned}$$

Case (5) $1 < c'(\mathcal{L}_1) \leq c'(\mathcal{L}_3) \leq 3/2$. We jump from Case (8) to case (6), thus,

$$\begin{aligned}\Delta_\mathcal{C} &= (24 - c'(\mathcal{L}_3)) - (18 + 5c'(\mathcal{L}_1) - c'(\mathcal{L}_3)) \\ &= 6 - 5c'(\mathcal{L}_1) \geq -3/2.\end{aligned}$$

Case (6) $1 < c'(\mathcal{L}_1) < 3/2 < c'(\mathcal{L}_3), c'(\mathcal{L}_1) + c'(\mathcal{L}_3) \leq 3$. We jump from Case (9) to Case (7), thus,

$$\begin{aligned}\Delta_\mathcal{C} &= (15 + 5c'(\mathcal{L}_3)) - (9 + 5c'(\mathcal{L}_1) + 5c'(\mathcal{L}_3)) \\ &= 6 - 5c'(\mathcal{L}_1) \geq -3/2.\end{aligned}$$

Case (7) $c'(\mathcal{L}_1) + c'(\mathcal{L}_3) > 3$. If $c'(\mathcal{L}_3) \leq 3/2$ then we jump from Case (10) to Case (6) and have

$$\begin{aligned}\Delta_\mathcal{C} &= (24 - c'(\mathcal{L}_3)) - (9 + 4c'(\mathcal{L}_1) + 4c'(\mathcal{L}_3)) \\ &= 15 - 4(c'(\mathcal{L}_1) + c'(\mathcal{L}_3)) - c'(\mathcal{L}_3) \\ &\geq 15 - 4(2 + 3/2) - 3/2 \geq -1/2.\end{aligned}$$

If $c'(\mathcal{L}_3) > 3/2$ then we jump from Case (10) to Case (7) and have

$$\begin{aligned}\Delta_\mathcal{C} &= (15 + 5c'(\mathcal{L}_3)) - (9 + 4c'(\mathcal{L}_1) + 4c'(\mathcal{L}_3)) \\ &= 6 - 4c'(\mathcal{L}_1) + c'(\mathcal{L}_3) \geq 6 - 8 + 3/2 \geq -1/2.\end{aligned}$$

We conclude that $\Delta_{\mathcal{C}} \geq -3/2$.

Consider literal exclusion path P. Since $c'(\mathcal{L}_1) + c'(\mathcal{L}_2) > 3$ it follows that P gives profit 0 under price assignment c'. From $c'(\mathcal{L}_1) \leq c'(\mathcal{L}_2)$ we conclude that $c'(\mathcal{L}_2) \geq 3/2$. With $c''(\mathcal{L}_1) = 1$ path P then gives profit at least $5/2$ under c'' and we have that $\Delta_P \geq 5/2$. It is $c'(\mathcal{L}_1) - c''(\mathcal{L}_1) \leq 1$. Since the profit on literal exclusion path Q can decrease by no more than this difference we observe $\Delta_Q \geq -1$. We can then bound the total difference in profit by

$$\begin{aligned}\Delta &= \Delta_{\mathcal{C}} + \Delta_P + \Delta_Q \\ &\geq -3/2 + 5/2 - 1 = 0.\end{aligned}$$

This gives the claim. □

Lemma A.4 *It holds that* $prof(c^*) \geq prof(c'')$.

Proof. For each variable of the SAT-instance 6 corresponding literal exclusion paths induce some cyclic structure on our graph. Two of theses cycles can be connected by the paths of a clause component containing literal components from the respective cycles.

We first note that the SAT-feasible assignment c^* can be constructed locally, i.e., considering only the literal exclusion paths belonging to one variable at a time. We will also follow this local approach to show that total profit does not decrease. For variable x_i we define $\mathcal{X}_i = \{\mathcal{L}_0(x_i), \mathcal{L}_1(x_i), \mathcal{L}_2(x_i)\}$, $\overline{\mathcal{X}}_i = \{\mathcal{L}_0(\overline{x_i}), \mathcal{L}_1(\overline{x_i}), \mathcal{L}_2(\overline{x_i})\}$ and let

$$\alpha_i = |\{\mathcal{L} \,|\, \mathcal{L} \in (\mathcal{X}_i \cup \overline{\mathcal{X}}_i) \wedge c''(\mathcal{L}) > 1.75 \wedge c^*(\mathcal{L}) = 1\}|$$

denote the number of *problematic* literal components belonging to variable x_i. We start by observing that $\alpha_i \in \{0, 1\}$. Let $B = \{\mathcal{L} \,|\, \mathcal{L} \in (\mathcal{X}_i \cup \overline{\mathcal{X}}_i) \wedge c''(\mathcal{L}) > 1.75\}$. If $B \subseteq \mathcal{X}_i$ or $B \subseteq \overline{\mathcal{X}}_i$ then we can obviously define a SAT-feasible c^* such that $\alpha_i = 0$. So let us assume that $B \cap \mathcal{X}_i \neq \emptyset$ and $B \cap \overline{\mathcal{X}}_i \neq \emptyset$. From the construction of c'' we know that B cannot contain 2 literal components that are connected by a literal exclusion path, since one of the prices would have been set to 1 in Step 2 of the transformation. From the fact that we are looking at a cyclic structure of length 6 it then follows that $|B| = 2$ and we can assume w.l.o.g that $B = \{\mathcal{L}_0(x_i), \mathcal{L}_1(\overline{x_i})\}$. It is then clear that any SAT-feasible price assignment results in $\alpha_i = 1$.

Let Δ_i denote the change in profit in all literal exclusion paths belonging to variable x_i going from price assignment c'' to c^*. We will now show that $\Delta_i \geq \alpha_i$. To see this, first note that $\Delta_i \geq 0$ is a trivial observation, since profit from the literal exclusion paths becomes maximal under c^*. It then only remains to be shown that $\Delta_i \geq 1$ if $\alpha_i = 1$. We have already argued that we can assume w.l.o.g that $B = \{\mathcal{L}_0(x_i), \mathcal{L}_1(\overline{x_i})\}$. Again from the construction of c'' it follows that $c''(\mathcal{L}_1(x_i)), c''(\mathcal{L}_2(x_i)), c''(\mathcal{L}_0(\overline{x_i})), c''(\mathcal{L}_2(\overline{x_i})) < 1.25$. Profit from paths P_2^i and P_5^i under c'' is then bounded by 2.5 each. Hence, total profit from the cycle is at most $4 \cdot 3 + 2 \cdot 2.5 = 17$ under c'' and will increase to its maximum of 18 under c^*, thus, $\Delta_i \geq 1$.

For the second part of the proof we now consider an arbitrary clause component $\mathcal{C}$ consisting of some literal components $\mathcal{L}_1$, $\mathcal{L}_2$ and let $\Delta_{\mathcal{C}}$ refer to the relative change of profit. Under price assignment c^* each clause component gives at least profit 24. From Lemma A.1 it follows that profit from $\mathcal{C}$ under c'' is at most 24 if $c''(\mathcal{L}_1), c''(\mathcal{L}_2) \leq 1.75$. We note that $\Delta_{\mathcal{C}} \geq 0$ in this case. But how can $\Delta_{\mathcal{C}}$ become negative? This can happen only if there is a $j \in \{1, 2\}$ such that $c''(\mathcal{L}_j) > 1.75$ and $c^*(\mathcal{L}_1) = c^*(\mathcal{L}_2) = 1$. It is clear that $\Delta_{\mathcal{C}} \geq -1$ in this case.

The important observation now is that each variable i with $\alpha_i = 1$ can cause at most one clause component to have decreasing profit. It immediately follows that $\sum_{\mathcal{C}} \Delta_{\mathcal{C}} \geq -\sum_i \alpha_i$. Finally, we note that profit from any dummy literal component is maximal under c^* and, thus, can only be higher than under c''. Now let Δ be the change of profit in the complete graph. From the above argumentation it is immediately clear that

$$\begin{aligned}\Delta &\geq \sum_{i=1}^{n} \Delta_i + \sum_{\mathcal{C}} \Delta_{\mathcal{C}} \\ &\geq \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i = 0.\end{aligned}$$

This finishes the proof. □