Product Pricing with Additive Influences

Algorithms and Complexity Results for Pricing in Social Networks

Vom Fachbereich Mathematik der
Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften
(Doctor rerum naturalium, Dr. rer. nat.)

genehmigte Dissertation

von
Florian David Schwahn

1. Gutachter: Prof. Dr. Sven Oliver Krumke
2. Gutachter: Prof. Dr. Rainer Schrader

Datum der Disputation
16. Dezember 2016

D386

To my parents and sisters

Contents

1 Introduction

Never before was it that easy to spread the word, communicate with people, and discuss with strangers. Eventually these contacts are affecting a person. In particular opinions and valuations about products and services are changing. People may use recommendations to find help in making a difficult choice. Those recommendations might either be from good friends, whose opinion we value highly. Or they might be from strangers we do not trust as much.

Content publishers and retailers promote these review systems. A "good" reviewer is more influential. Popular *YouTube* users are earning money by simple product placement, so called affiliate marketing. Yet in contrast to commercials, the advertisement of a familiar "internet person" is valued as a friendly recommendation. Considering these realities, we propose a model for the underlying influences. We study a product pricing model in social networks where the value a possible buyer assigns to a product is influenced by the current customers.

Imagine Tony has just developed his new smartshoe and wonders what would be a good sales price, i.e., a price that yields the most revenue. A quick poll among his internet friends Ada, Butch, and Cathy reveals that they are willing to pay different prices of \$500, \$100, and \$300, respectively. Using a linear time algorithm, Tony first computes that \$300 is the best price since then Ada and Cathy will buy the shoes, guaranteeing him a revenue of \$600. A chat with Butch and Cathy, however, makes Tony realize that they are big admirers of Ada and if Ada has acquired the smartshoes, the next day (round) Butch would pay additional \$400 and Cathy \$100 for the smartshoe. Thus, under these circumstances, Tony could generate a total revenue of \$1200 by offering the product for \$400 and then letting things evolve. This situation describes the "easy" case of our model: problems such as finding a price for a product when people within a social network influence each other "in a positive way" feature monotonicity and can therefore be solved efficiently in polynomial time.

Though it is not that easy with Djustin. He has heard about the existence of Tony's revolutionary product and is willing to pay \$400, no matter who else owns the product. Butch and Cathy dislike Djustin and they would be less interested in Tony's smartshoe if Djustin was a customer. We model this as Djustin *decreasing* Butch's and Cathy's valuation for the smartshoe by \$400 and \$200, respectively. Hence, the small social network comprised of Ada, Butch, Cathy, and Djustin looks as in Figure 1.1 and the highest possible revenue is \$1000 obtained for the price \$500 (and *not* \$800, which would be obtained for the price \$400). This second situation, where some people may exert negative influences within the social network, is the "NP-hard" case of our basic model.

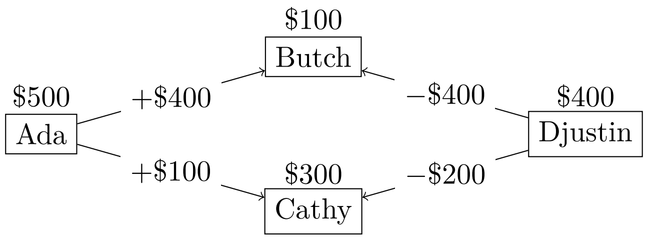

Figure 1.1: Tony's friends, their initial values for his smartshoes, and influences within the social network

The simple additive influence model described for Tony's product is the foundation for every model we examine in this thesis. The *Product Pricing with Additive Influence* model is based on graph dynamical systems [KKM+11, MR08, BHM+07]. We assume a graph consisting of all prospective customers, their relations and valuations, is given to the product seller. The questions we ask are: What price is "best possible" and who exactly is interested to purchase for a particular price?

Models for the propagation of ideas and influences through a social network have been studied in a number of domains. Examples include the diffusion of medical and technological innovations [CKM66, Val95], the sudden and widespread adoption of various strategies in game-theoretic settings [Blu93, Ell93, Mor00, You06], and the effects of "word of mouth" in the promotion of new products [DR01, GLM01].

The linear threshold model [KKT03] provides the foundation for the decision making of the potential buyers in our model. Based on the linear threshold model [FKLL13, BFO10, CLS+10, HMS08, CBO12] are investigating game theoretic or stochastic variations. Our model differs from others in the facts that it is completely deterministic and not strategic.

The model in [KKM+10, KKM+15] is deterministic and the goal is to find a critical set of nodes such that the contagion is minimal. The influence maximization starting from some seeding set is studied in [Swa14]. In our model the single choice lies in the price of the product. The price is affecting the set of early adopters and the thresholds. While it is usually the set of "early adopters" that is chosen.

Outline

This section provides an outline of the major results in the thesis.

In Chapter 3 we define the fundamental problem *Product Pricing with Additive Influences* PPAI. We show that a for a given price the revenue can be computed in linear time.

Thereby, checking all possible prices yields the price generating the highest revenue. This leads to a pseudo-polynomial time algorithm. We improve upon this pseudo-polynomial method with Algorithm Frag. A key concept in this respect is the notion of fragments — intervals of prices sharing the exact same customer behavior. For a polynomial amount of fragments the algorithm is also polynomial. Thereafter we provide a reduction from 3SAT and show NP-completeness of PPAI. The graph constructed in the reduction shows that the amount of fragments can be exponential in the amount of nodes.

In the following we aim to identify restrictions to the graph and its weights such that the amount of fragments is polynomial. For graphs with exclusively negative or positive influences this is indeed the case. In particular, Algorithm FixHighest solves the problem for positive graphs in polynomial time. We show that the algorithm can be implemented to run in almost linear time. We also prove monotonicity for buyers decisions in positive graphs with respect to prices. In the next step we examine the fragments for special graph classes. Amongst others, a polynomial bound on the fragments is given for graphs with bounded in-degree and graphs with bounded length of positive paths. However, the problem remains hard even for graphs with highest out-degree 2. Assuming symmetric influences, i.e., undirected graphs, we modify the original reduction graph and show NP-completeness also in the undirected setting.

We study some gadgets to transform instances and conclude computational complexity equivalence of our natural PPAI variants. In the penultimate section we tackle the complexity of PPAI on trees. Even though we can not provide a proof, we have strong indication for a polynomially bounded amount of fragments. The last section of Chapter 3 contains a formulation as graph dynamical system and as integer program.

In Chapter 4 on Delayed Product Pricing we extend the model by potential release times and influence delays instead of the immediate propagation we had so far. This leads to problem DelPPAI. We provide a polynomial algorithm to compute the revenue for a given price and show that the problem is closely connected to PPAI.

Next, in Chapter 5, we extend the basic model by allowing price changes. We call this Dynamic Product Pricing and denote the problem by DynPPAI. Given one particular trend of prices, we can apply DynSell to compute the revenue. Similar to the fragments in Chapter 3, we introduce b-fragments which capture the behavior of customers in the dynamic setting. To solve DynPPAI we provide an algorithm iterating over all possible b-fragments.

We provide computational complexity results for decreasing and increasing trends of prices. Even more, we can show that the problem is NP-complete for positive and negative graphs. The problem remains hard even if we restrict the number of different prices in a trend. Yet, if we have only a constant amount of price changes, the problem is equivalent to PPAI and we can solve it in polynomial time given an algorithm for static prices. We conclude the chapter with the formulation of DynPPAI as an integer program.

In Chapter 6 the buyers do not remain a customer forever. Instead they have to rebuy the perishable product or subscription if they are still interested. The problem of this chapter is Perishable Product Pricing — PerPPAI. The durations of the product are either individual for the nodes or fixed to a common value. Whereas the computation of the revenue for a given price was polynomial in the previous cases, this is now a harder problem. We define the notion of breaks, i.e., selling rounds in which someone either decides not to rebuy anymore and rounds in which some previous non-customer is purchasing the product. Given a polynomial amount of breaks, the Algorithm Break can compute the revenue in polynomial time.

Positive graphs are once again benevolent and feature the same monotonicity as in Chapter 3. In the case of exclusively negative influences and individual durations, we provide a #P-hardness proof for the computation of the revenue for a fixed price. On arbitrary graphs with fixed duration we give another #P-hardness reduction. However, for individual durations the problem is even hard on acyclic graphs.

Built on these complexity results we identify that PerPPAI with fixed durations is NP-hard. For positive graphs we rehash previous FixHighest to compute the optimum revenue. The problem PerPPAI on negative, acyclic graphs with unit duration is shown to be NP-complete. Instead of an algorithm for PerPPAI we state a formulation of the problem as integer program.

We conclude the chapter with return options for the product in our basic PPAI model. The return option results in a similar buying behavior as for perishable goods.

In Chapter 7 we study and review diverse aspects of our product pricing problem. In the section on Cooperative Pricing we assume that the potential buyers are communicating with each other and can therefore plan their purchase. Algorithm FixLowest provides an optimal solution in case the influences are positive and runs in polynomial time. The next section is groundwork for mechanism design on our PPAI model. We use one initial value as parameter and determine the revenue for positive graphs subject to this value. More game theoretic aspects are featured in the section on Two Product Pricing. Introducing a competitor we show a primary result for positive graphs. Finally, we provide another influence model — the Bounded Additive Influence.

Credits and Acknowledgements

Some parts of this thesis were developed in collaboration with fellow members of the University of Kaiserslautern:

Most results of this thesis are joint work with my supervisor Sven Krumke and co-supervisor Clemens Thielen. Section 6.4 was worked out with Sebastian Johann under supervision of Sven Krumke and me. The foundation for parts of Chapter 7 was worked out with and extended by students Thanh To and Patrick Gerhards under supervision of Clemens Thielen.

Parts of Chapter 3 were already published in the proceedings of the *7th International Conference on Fun with Algorithms* [KST14].

First and foremost, I want to thank Sven Krumke. I could not imagine a wiser or kinder supervisor and I am deeply honored for all the inspiration he provided me with.

My second thanks is to Clemens Thielen. His confidence in my work and our collaboration in the last few years can never be underestimated or forgotten.

I am grateful to the Optimization Group for providing a pleasant environment. And I am even more grateful for all the friends I met during my studies in Kaiserslautern. Their support in all these years has made my professional and private life a wonderful time. I look back with great joy and ahead with solid confidence that even more memories will be made with them.

In particular, I want to thank André, Andrea, Annika, Cornelia, Lena, and Nico for technical assistance and proofreading.

Finally, I want to express my gratitude to my family. Their encouragement and support is limitless — which I cherish as much as I want to reciprocate it. In particular, my sister Anne deserves greatest praise for all her guidance on the path to this thesis.

2 Preliminaries

In this chapter, we summarize basic definitions used throughout this thesis. While we assume that the reader has basic knowledge of combinatorics, graph theory, and computational complexity, the notations in literature may vary immensely. With the following definitions, we make sure that the content is unambigous.

Sets and Basics: [BS79]

The natural numbers $\{1, 2, \dots\}$ are denoted by $\mathbb{N}$. The first n natural numbers are denoted by $\mathbb{N}_n \coloneqq \{1, \dots, n\}$. By $(x, y] \coloneqq \{\, z \in \mathbb{Z} : x < z \leq y \,\}$ we denote the integer intervals. For $x, y \in \mathbb{Z}$ this is $\{x + 1, \dots, y\} = (x, y]$.

By $|A|$ we denote the cardinality of a set A. Throughout the thesis we only use finite sets, i.e., $|A| \in \mathbb{N}$. The power set of A is denoted by 2^A. By $A \dot\cup B$ we denote the disjoint union of two sets.

We use the binary logarithm $\operatorname{ld} = \log_2$. Numbers n in binary representation are indicated as $(n)_2$.

The binomial coefficient $\binom{n}{k} \coloneqq \frac{n \cdots (n-k+1)}{1 \cdots k}$ for $0 \leq k \leq n$, is the amount of different choices to choose k elements from a set of size n, disregarding the order. For $k < 0$ or $k > n$ we set $\binom{n}{k} = 0$. With $\left(\!\binom{n}{k}\!\right) \coloneqq \binom{n+k-1}{k}$ we denote the respective amount of multisets.

Growth of Functions: [PS82]

We use the Bachmann-Landau notations to classify the growth of functions.

For functions $f, g : \mathbb{N} \to \mathbb{N}$, we write

- $f(n) = \mathcal{O}(g(n))$ if there exists some constant $c > 0$ such that for large enough n, $f(n) \leq c \cdot g(n)$,
- $f(n) = \Omega(g(n))$ if there exists some constant $c > 0$ such that for large enough n, $f(n) \geq c \cdot g(n)$, and
- $f(n) = \Theta(g(n))$ if there exist constants $c, c' > 0$ such that for large enough n, $c \cdot g(n) \leq f(n) \leq c' \cdot g(n)$.

Furthermore, poly(n) is the class of polynomial functions, i.e., if there is some polynomial function g such that $f(n) = \mathcal{O}(g(n))$, we have $f(n) = \text{poly}(n)$.

Graph Theory: [Die05]

A graph $G = (V, A)$ is specified by the set of nodes V and the set of arcs A. Throughout the thesis we assume that the graphs are finite, i.e., both $n := |V|$ and $m := |A|$ are finite. Furthermore, the graphs are defined as directed graphs. The graphs do not contain loops or parallels. Thus we can identify the arcs $a = (u, v) \in A \subseteq V \times V$ without any conflict.

By $\mathcal{N}^-(v) := \{\, u \in V : (u, v) \in A \,\}$ we denote the (in-)neighborhood of node v, also called predecessors. The in-degree of node v is $g^-(v) := |\mathcal{N}^-(v)|$. We define $\mathcal{N}^+(v) := \{\, u \in V : (v, u) \in A \,\}$, the out-neighborhood of node v, also called successors. Analogously we have out-degree $g^+(v)$.

A directed path of length k is a node sequence $(v_0, \dots, v_k)$ with $(v_{i-1}, v_i) \in A$ for all $i = 1, \dots, k$, and $v_1, \dots, v_k$ distinct nodes. If we relax $(v_{i-1}, v_i) \in A$ and also allow $(v_i, v_{i-1}) \in A$ we have an undirected path. For $v_0 = v_k$ the path is a cycle. If a graph G does not contain a directed cycle, we say it is acyclic. Acyclic graphs have a topological sorting, i.e., for sorting $v_1, \dots, v_n$ it holds for all arcs $a \in A$ that $a = (v_i, v_j)$ for some $i < j$.

A graph is weakly connected if for any two nodes there is an undirected path connecting them. A directed graph is a tree if it is weakly connected and contains $m = n - 1$ arcs.

A tree is a rooted in-tree with root v if every other node in the graph has a directed path to root v. In the in-tree we consider nodes with $g^-(v) = 0$ as leaves.

Computational Complexity: [GJ79, Pap94, CSRL01]

We distinguish decision and function problems. Throughout the thesis we use the following classes.

P	Decision problems solvable in polynomial time on a deterministic Turing machine
FP	Function problems solvable in polynomial time on a deterministic Turing machine
NP	Decision problems solvable in polynomial time on a non-deterministic Turing machine
NPO	Optimization problems solvable in polynomial time on a non-deterministic Turing machine
APX	Problems of NPO that have polynomial time constant factor approximation algorithms [APMS+99]

#P Counting problems: Counting the number of accepting paths of a non-deterministic Turing machine [Val79]

As computational-machine model, we use the unit cost random-access machine. The reductions provided in the thesis are Karp reductions. We denote $A \propto B$ if problem A is reducible to problem B.

The satisfiability problem 3SAT is used for the reductions. We further assume that a clause does not contain a complementary pair of literals.

3 Product Pricing with Additive Influences

In this chapter we examine the basic additive influence model discussed in the introduction. We prove that in general it is NP-complete to find a price yielding a certain revenue. Furthermore, we state an algorithm that solves the problem in pseudo-polynomial time. We further investigate the problem for special instances and adapt the model to other theories.

3.1 Definitions and Problems

Before we give the proper definitions for the *Product Pricing with Additive Influences* model, we introduce the decision problem constituting this chapter. The exact way how the selling process happens will be given shortly.

Problem 1: PPAI *(Product Pricing with Additive Influences)*

Instance: *A directed graph* $G = (V, A)$*, initial values* $p(v) \in \mathbb{Z}$ *for the nodes* $v \in V$*, influences* $w(u, v) \in \mathbb{Z} \setminus \{0\}$ *for the arcs* $(u, v) \in A$*, and some revenue* $\hat{R} \in \mathbb{N}$*.*
Question: *Is there a price* $\pi \in \mathbb{N}$ *such that its revenue* $R(\pi) \geq \hat{R}$*?*

Throughout this dissertation we assume that the graph $G = (V, A)$ is *weakly connected* and *simple*, i.e., it does not contain loops or parallel arcs. Unless stated otherwise, the number of nodes and arcs are denoted by $n = |V|$ and $m = |A|$, respectively. A node $v \in V$ is designed to model the potential *customer* of the product we consider.

In this chapter we do not offer a return option for the product, a single customer only needs one edition of it, and it will be available as well as functional for all eternity. This mainly implies that we do never lose a customer. Therein lies a fundamental difference to the basic Discrete Dynamical Systems [MR08] and to the perishable goods, temporary services, and returnable products featured in Section 6.4.

Buying Decision: Prior to the existence of influences, every potential customer $v \in V$ has an initial value $p(v) \in \mathbb{Z}$ assigned to the product. If $p(v) = 0$ we loosely state that node v has no initial value.

The arcs express the influence exerted by the nodes. Every node $v \in V$ is influenced by its *predecessors* $\mathcal{N}^-(v)$ and influences its *successors* $\mathcal{N}^+(v)$. Given a set C of current customers, i.e., the nodes that have bought the product already, the *influenced value* for the particular *influences* $w : A \to \mathbb{Z} \setminus \{ 0 \}$ is computed as follows:

$$p_C(v) = p(v) + \sum_{u \in C \cap \mathcal{N}^-(v)} w(u, v)$$

We assume that a node *buys* deterministically for a *price* π once its influenced value is at least as large as π, i.e., $p_C(v) \geq \pi$ for some customer set C, provided it did not already buy.

Selling Process: The whole selling process in an instance of PPAI proceeds in discrete, synchronous *selling rounds*. Given a fixed price π, in the first round $t = 1$, all nodes in the set

$$C_1(\pi) \coloneqq B_1(\pi) \coloneqq \{ v \in V : p_\emptyset(v) \geq \pi \} = \{ v \in V : p(v) \geq \pi \}$$

acquire the product. For round $t > 1$, we inductively let $C_{t-1}(\pi)$ denote the previous customers. Then, the *set of buyers* in round t is

$$B_t(\pi) \coloneqq \{ v \notin C_{t-1}(\pi) : p_{C_{t-1}(\pi)}(v) \geq \pi \}$$

and $C_t = C_{t-1} \,\dot\cup\, B_t$. The overall *revenue* obtained is $R(\pi) = \pi \, |C(\pi)|$, where $C(\pi) \coloneqq \bigcup\limits_{i \geq 1} B_i(\pi)$ is the set of nodes who buy the product in some round.

The prices, values, and influences are restricted to integers just to ensure a simple input length. One could use rational and real numbers in actual computations without any changes to the algorithms developed here.

One could imagine and interpret negative initial and influenced values, e.g., for a person being paid for acquiring a product or contributing to the service himself. However, in order to determine the maximum revenue in PPAI, we can restrict ourselves to non-negative values of the sales price π. In every instance the price $\pi = 0$, i.e., giving away the product for free, is superior to the non-positive revenue generated for negative prices.

Apart from the fundamental PPAI problem we also consider some modifications from the basic *Instance* or *Question* posed. Note that these notations are also applied to problems in the following chapters for the corresponding problems therein.

Definition 3.1 (Variants of PPAI)**.** For all of these problems the bound $\hat{R}$ is obsolete and therefore removed.

PPAI^{π}: Compute the revenue for a given price.

$\mathsf{PPAI}^{\mathrm{opt}}$: The corresponding revenue maximization problem.

Note that the aforementioned problems are function problems.

Instead of the revenue bound we introduce a lower bound $\hat{\pi}$ on a buying price.

PPAI-v: Is there a price $\pi \geq \hat{\pi}$ such that for a given node $v \in V$ we have $v \in C(\pi)$

PPAI-S: Is there a price $\pi \geq \hat{\pi}$ such that for a given subset $S \subseteq V$ we have $S \subseteq C(\pi)$?

PPAI-i: Is there a price $\pi \geq \hat{\pi}$ such that for a given $i \in \{1, \dots, n\}$ we have $|C(\pi)| \geq i$?

Further outline of this chapter: The first problem posed is PPAI$^{\pi}$, namely how to compute the revenue $R(\pi)$ for a given price π (Section 3.2). We continue with pseudo-polynomial algorithms (Section 3.3) and complexity results for the other problems (Section 3.4).

Thereafter, we discuss the impact of different graph classes, in particular sign constrained influences (Section 3.5), trees (Section 3.8) or other special graph classes (Section 3.6).

We conclude with relations inside our problems (Section 3.7) and the transition to established models (Section 3.9).

3.2 Computing the Revenue

Before we start to maximize the revenue, we need a way to compute it for a fixed price.

Given the sequential nature of the selling rounds, the obvious approach is to evaluate all selling rounds one after another.

Remark 3.2. If there is no buyer in some selling round $t \in \mathbb{N}$, i.e., $B_t(\pi) = \emptyset$ and $C_t(\pi) = C_{t-1}(\pi)$, then the influenced values will not change for round $t+1$. Hence, there will be no new buyers in any round $i > t$. Since the sets $B_t(\pi)$ form a partition of $C(\pi) \subseteq V$, there can be at most n rounds with $B_t(\pi) \neq \emptyset$.

Also note that in case of $\pi > p_{\max} := \max\limits_{v \in V} p(v)$, there will be no buyer in the first round and hence, no customer at all.

Since we already know the revenue $R(0) = 0$ — our fail-safe revenue — for every instance, the actual selling process occurs for prices π in the integer interval $(0, p_{\max}]$ and lasts at most n rounds.

In Subsection 3.6.1 we show further bounds on the number of selling rounds.

Algorithm Sell implements the basic definitions of the selling process straightforward. Since in the actual use of Sell we need the revenue as well as the customers, we output both and assume, with some abuse of notation, we get the desired value.

Lemma 3.1. *The algorithm Sell(G, π) computes the buyers $C(\pi)$ and revenue $R(\pi)$ of a graph G and a fixed price π in $\mathcal{O}(m)$ time.*

Algorithm 1: $\mathsf{Sell}(G, \pi)$

```
Data: A graph $G = (V, A)$ with weights $p$ and $w$, and a price $\pi$.
Result: The set $C(\pi)$ of customers and the revenue $R(\pi)$.
Initialize: $p'(v) = p(v)$ for $v \in V$, $C = \emptyset$, and $B = \{ v \in V : p(v) \geq \pi \}$.
while $B \neq \emptyset$ do
                                                    // Update influenced values
    $C = C \cup B$
    $X = \emptyset$                                 // nodes whose valuations change
    for $u \in B$ do
        for $v \in \mathcal{N}^+(u) \setminus C$ do
            Update $p'(v) = p'(v) + w(u, v)$.
        $X = X \cup \left(\mathcal{N}^+(u) \setminus C\right)$
                                                    // Compute $B$
    $B = \emptyset$
    for $v \in X$ do
        if $p'(v) \geq \pi$ then  $B = B \cup \{v\}$
return $R(\pi) = \pi\,|C|$ and $C(\pi) = C$
```

Proof. The correctness follows immediately by the definition of the selling process.

The initialization takes $\mathcal{O}(n)$ time. To update the influenced values, we iterate over the outgoing arcs of every customer exactly once, which takes $\mathcal{O}(m)$ time. And since we only consider nodes in X for new buyers, we check every node at most $g^-(v)$ times, again totalling to $\mathcal{O}(m)$ time.

Hence, and since the graphs are assumed to be weakly connected, $\mathcal{O}(m)$ time is needed in total. □

We end this section with the first results on complexity classes.

Corollary 3.2. *$\mathsf{PPAI}^\pi \in \mathsf{FP}$ and $\mathsf{PPAI} \in \mathsf{NP}$.*

Likewise the problems PPAI-v, PPAI-S, and PPAI-i are in NP, the optimization variant $\mathsf{PPAI}^{\text{opt}}$ is in NPO.

Proof. As shown in Lemma 3.1 we can compute the revenue in polynomial time. Hence, given a price π for an instance of PPAI, we can verify whether the required revenue $R(\pi) \geq \hat{R}$ is met by a single call to Sell — in polynomial time.

Using the output $C(\pi)$ we can solve the variations, i.e., check whether $v \in C(\pi)$, $S \subseteq C(\pi)$, or $i \leq |C(\pi)|$. □

3.3 Pseudo-polynomial Algorithms

After we established that PPAI is in NP, we consider two increasingly apt algorithms solving the optimization problem $\mathsf{PPAI}^{\mathrm{opt}}$. Note that we always formulate algorithms solving the optimization problem, whereas the complexity proofs are focused on the decision problems. For this reason we will interchangeably claim that an algorithm solves both and they are of the same complexity.

Algorithm 2: BruteSell(G)

```
Data: A graph $G = (V, A)$ with weights $p$ and $w$, and the maximum value $p_{\max}$.
Result: An optimal (actually the lowest optimal) price $\pi$ and revenue $R(\pi)$.
Initialize: $R^* = 0$, $\pi^* = 0$                         // temporary results
for $\pi = 1, \dots, p_{\max}$ do
    if Sell($G, \pi$) $> R^*$ then
        $R^* = \mathsf{Sell}(G, \pi)$, $\pi^* = \pi$
return $R^*$ and $\pi^*$
```

Proposition 3.3. *The brute-force method in Algorithm* BruteSell *is pseudo-polynomial and solves* $\mathsf{PPAI}^{\mathrm{opt}}$ *in time* $\mathcal{O}(mp_{\max})$.

Algorithm BruteSell is already running in pseudo-polynomial time, but we can see that we have potential room for improvement — which in this case means time.

Consider some instance with integer euro values and weights. If instead of running Algorithm Sell for every euro value, we compute the revenue for all cents in between, the running time increases by a factor of 100. The optimal price though can only be found at integer euro values, since in between the values of the potential buyers do not change.

While in this case we can obviously get a speed-up, the question remains how to detect "useless" computations all along. To answer this, we introduce the notion of fragments.

3.3.1 Fragments and the Frag Algorithm

Definition 3.3. For a round $t \in \{1, \dots, n\}$, a *t-fragment* is an inclusion-wise maximal integer interval $(a, b] := \{ x \in \mathbb{N} : a < x \leq b \} \subseteq (0, p_{\max}]$ of prices such that, for all $\pi, \pi' \in (a, b]$ and all rounds $i \leq t$, we have $B_i(\pi) = B_i(\pi')$.

We call $C_t((a, b]) := C_t(b)$ the *customers associated with t-fragment* $(a, b]$. Depending on the scarcity of either computation space or time we may assume that this set is well-known.

By a *temporary fragment* we mean a t-fragment for some $t < n$. The n-fragments are just called *fragments*. We denote by $\operatorname{frag}_t(G)$ and $\operatorname{frag}(G)$ the collection of t-fragments and fragments, respectively.

Remark 3.4. The sets $\operatorname{frag}_t(G)$ and $\operatorname{frag}(G)$ are partitions of $(0, p_{\max}]$ and therefore $|\operatorname{frag} G| \leq p_{\max}$. Even more, the fragments in $\operatorname{frag}_{t+1}(G)$ partition those of $\operatorname{frag}_t(G)$.

If all fragments of a graph G are known, the optimal price can be computed by running Sell once for the highest value in each fragment (in $\mathcal{O}(m \cdot |\operatorname{frag}(G)|)$ time). Thus, as long as the complete set $\operatorname{frag}(G)$ is adressed, we can also identify it by a sorted list of upper fragment endpoints.

In the remainder of this section we develop an algorithm that computes the fragments, whilst keeping note of the corresponding revenue.

Hitters: Let $(x, y]$ be a t-fragment and C_t the set of customers associated with it. We partition $V \setminus C_t$ into three sets:

$$L := \{\, v \in V \setminus C_t : p_{C_t}(v) \leq x \,\},$$

$$H := \{\, v \in V \setminus C_t : p_{C_t}(v) \in (x, y) \,\}, \text{ and}$$

$$O := \{\, v \in V \setminus C_t : y \leq p_{C_t}(v) \,\}.$$

By the fact that the $(t+1)$-fragments are subsets of t-fragments, the t-fragment $(x, y]$ is a disjoint union of $(t+1)$-fragments. No node in L will buy in round $t+1$ for any price $\pi \in (x, y]$. Similarly, any node in O buys in round $t+1$ for all prices $\pi \in (x, y]$.

Finally, each *hitter* $v \in H$ induces the endpoint of a $(t+1)$-fragment within $(x, y]$ at $p_{C_t}(v)$, where the decision of v whether to buy in round $t+1$ switches.

Hence, if $s_1 < \cdots < s_q$ is the sorted sequence of different influenced values from hitters, then

$$\{\, (x = s_0, s_1], (s_1, s_2], \ldots, (s_{q-1}, s_q], (s_q, s_{q+1} = y] \,\}$$

is the collection of $(t+1)$ -fragments that partition $(x, y]$. The maximum revenue $R^*((x, y]) := \max_{\pi \in (x, y]} R(\pi)$ that can be obtained by any price from the t-fragment $(x, y]$ then satisfies the recursion to $(t+1)$-fragments

$$R^*((x, y]) = \max_{i=1,\ldots,q+1} R^*\left((s_{i-1}, s_i]\right). \tag{3.1}$$

Algorithm Frag uses this recursion and Equation (3.3.1) to compute the optimum revenue for a given temporary fragment $(x, y]$. In order to save space, the algorithm uses a depth-first technique to iterate over the temporary fragments into which $(x, y]$ is ultimately split.

```
Algorithm 3: Frag($G, (x,y], C$)
Data: A graph $G = (V, A)$ with weights $p$ and $w$, a temporary fragment $(x, y]$ with its
      previous customers $C \subseteq V$.
Result: The optimal overall revenue $R^*$ and its corresponding price $\pi^* \in (x, y]$.
Initialize: $H = \emptyset$, $O = \emptyset$, and $P = \emptyset$      // the next fragment endpoints in $(x, y]$
    Set $\pi^* = y$, $R^* = y\,|C|$
                                                                  // Compute $H$ and $P$
if $C \neq V$ then
    Calculate influenced values $p_C(v)$ for all $v \in V \setminus C$.
    for $v \in V \setminus C$ do
        if $p_C(v) \in (x, y)$ then  $H = H \cup \{v\}$, $P = P \cup \{p_C(v)\}$
        if $p_C(v) \geq y$ then  $O = O \cup \{v\}$, $P = P \cup \{y\}$
                                                          // Initiate recursive calls
if $P \neq \emptyset$ then
    Let $s_1 < \cdots < s_{|P|}$ be the sequence of prices in $P$ and $s_0 = x$.
    for $i = |P|, \ldots, 1$ do
        Initialize $C^i = \emptyset$.
        for $v \in H$ do
            if $p_C(v) \geq s_i$ then  $C^i = C^i \cup \{v\}$
        $C^i = C^i \cup C \cup O$
        $(R_i, \pi_i) = \mathsf{Frag}(G, (s_{i-1}, s_i], C^i)$
        if $R_i > R^*$ then  $R^* = R_i$ and $\pi^* = \pi_i$
return $(R^*, \pi^*)$
```

Theorem 3.4. *Algorithm Frag correctly computes the optimum solution for PPAI$^{\text{opt}}$ in $\mathcal{O}\left(|\text{frag}(G)|\left(nm + n^2 \log n\right)\right)$ time — with a call on the 0-fragment $(0, p_{\max}]$ and customers $C = \emptyset$.*

Proof. Correctness follows immediately from the validity of Equation (3.3.1).

To compute H and P, we need to add each influence at most once, thus $\mathcal{O}(m)$ time. The sorting of P can be done in $\mathcal{O}(n \log n)$ time. It suffices to sum the required time for the computation of a single customer set, which takes $\mathcal{O}(n)$ time, since we can attribute these costs to the succeeding calls on Frag. As the final comparison of revenues takes $\mathcal{O}(n)$ time, we obtain the time-complexity of $\mathcal{O}(m + n \log n)$ for every call of Frag.

Since, for any t, the number of t-fragments is bounded by $|\text{frag}_t(G)| \leq |\text{frag}(G)|$, the total number of recursive calls is in $\mathcal{O}(n\,|\text{frag}(G)|)$. This yields the claimed running time. □

However, as we see in Section 3.4, $|\mathrm{frag}(G)|$ can be of exponential size, making this algorithm exponential in the "worst case". The "good cases" identified in this thesis, see Sections 3.5 and 3.6, all rely on $|\mathrm{frag}(G)| = \mathrm{poly}(n)$. For any of these, the algorithm Frag is therefore already a polynomial time algorithm.

3.4 NP-completeness

In this section the proof of NP-completeness is developed. Since the main idea will be reused in the course of this thesis, we portion the several aspects required to utilize them later on.

In order to provide a reduction from 3SAT, which is known to be NP-complete [GJ79, Pap94], we have to map the concept of variables and clauses to our PPAI problem. We start by building a representation of variables in our *additive influence* setting.

3.4.1 Variable Gadget

An instance $\mathcal{I}$ of PPAI is given by a graph $G = (V, A)$ with initial values p and influences w. The only choice in $\mathcal{I}$, corresponding to the choice of variable assignment, lies in the various prices $\pi \in (0, p_{\max}]$. In this subsection we establish a one-to-one correspondence between *price* and *variable assignment* in an instance of 3SAT.

Now, given an instance $\mathcal{J}$ of 3SAT with $\tilde{n}$ variables $x_1, \ldots, x_{\tilde{n}}$ and $\tilde{m}$ clauses $y_1, \ldots, x_{\tilde{m}}$ we have $2^{\tilde{n}}$ different variable assignments, therefore we need at least $2^{\tilde{n}}$ fragments in $\mathcal{I}$ to cover them. Note that this implies $p_{\max} \neq \mathrm{poly}(\tilde{n})$.

Remark 3.5. For the sake of simplicity, we assume for $\mathcal{J}$ that every variable is contained in at least one clause, as otherwise it can be deleted, so in particular$\tilde{n} \leq 3\tilde{m}$. Furthermore, we have at most

$$\tilde{m} \leq \binom{2\tilde{n}}{3} < \frac{4}{3} \cdot \tilde{n}^3$$

potentially different clauses. Altogether this polynomially limits $\tilde{n}$ and $\tilde{m}$ together to

$$\tilde{n} \leq 3\tilde{m} < 4\tilde{n}^3.$$

Gadget 3.6 (Variable Graph)**.**

In the gadget graph $G^x = (V^x, A^x)$, we set the nodes $V^x = \{t_1, \dots, t_{2\tilde{n}}, x_1, \dots, x_{\tilde{n}}, \overline{x}_1, \dots, x_{\tilde{n}}\}$ with $p(t_1) = 2^{\tilde{n}+1} - 1$ and $p(v) = 0$ for all other $v \in V \setminus \{t_1\}$.
We define the arc set A^x implicitly by setting the influences as follows:

$$\begin{aligned}
w(t_i, t_{i+1}) &= 2^{\tilde{n}+1} - 1 & i &= 1, \dots, 2\tilde{n} - 1\\
w(t_{2j-1}, \overline{x}_j) &= 2^{\tilde{n}} - 1 + 2^{\tilde{n}-j} & j &= 1, \dots, \tilde{n}\\
w(x_i, \overline{x}_j) &= 2^{\tilde{n}-i} & 1 &\le i < j \le \tilde{n}\\
w(t_{2i}, x_i) &= 2^{\tilde{n}+1} - 1 & i &= 1, \dots, \tilde{n}\\
w(\overline{x}_i, x_i) &= -(2^{\tilde{n}+1} - 1) & i &= 1, \dots, \tilde{n}
\end{aligned}$$

The variable graph $G^x = (V^x, A^x)$ contains $n = 4\tilde{n}$ nodes and $m = 5\tilde{n} - 1 + \frac{\tilde{n}\cdot(\tilde{n}-1)}{2}$ arcs. Also including the weights, the encoding length of graph G^x is $\mathcal{O}\left(\tilde{n}^2\right)$— polynomial in the encoding length of its original 3SAT instance.

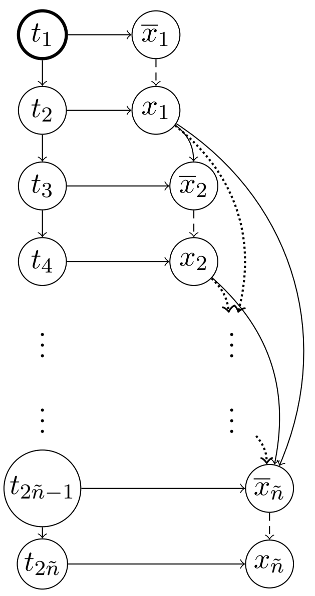

Figure 3.1: Variable Graph G^x

Lemma 3.5. *The graph G^x constructed in Gadget 3.6 has at least $2^{n/4} = 2^{\tilde{n}}$ fragments. In particular, for the set of prices $\Pi := \left\{ 2^{\tilde{n}}, \dots, 2^{\tilde{n}+1} - 1 \right\}$ every customer subset $C(\pi) \cap \{ x_1, \dots, x_{\tilde{n}} \}$ occurs and the one-to-one correspondence between assignment and price is given by a simple bijective function.*

Proof. We restrict this proof, and the further use of the given graph, to the set of prices Π. For PPAI we give the reasoning in the proof of Theorem 3.9, for the other problems we just set $\hat{\pi}$ such that $\Pi = \{ \hat{\pi}, \dots, p_{\max} \}$.

For any price $\pi \in \Pi$ the only purchase in the first round is by t_1. Note that thereafter the other *time nodes* $t_2, \dots, t_{2\tilde{n}}$ buy one after another. Every variable node (positive literal) $x_i \in \{ x_1, \dots, x_{\tilde{n}} \}$ has exactly one positive influence, namely $w(t_{2i}, x_i) = 2^{\tilde{n}+1} - 1$. Hence, if at all, x_i can only buy one round after t_{2i} did — in round $2i + 1$.

A variable node (negative literals) $\overline{x}_j \in \{ \overline{x}_2, \dots, \overline{x}_{\tilde{n}} \}$ has potentially more positive influences, yet without the influence of t_{2j-1} the influenced value does not exceed

$$2^{\tilde{n}-1} + \dots + 2^{\tilde{n}-(j-1)} < 2^{\tilde{n}}$$

and, hence, does not buy for any price in Π. The influence from t_{2j-1} itself is the latest, thus $\overline{x}_j$ can only buy one round after it — in round $2j$.

Now, after we established the exact (potential) buying round of each node, we determine their buying prices by induction.

Before we state the actual hypothesis, we start with the induction basis.

The node $\overline{x}_1$ buys for all prices in $\left\{2^{\tilde{n}}, \ldots, 2^{\tilde{n}} + 2^{\tilde{n}-1} - 1\right\}$. In binary representation this is the set $\left\{1\underbrace{00\cdots0}_{\tilde{n}\text{ bits}}, \ldots, 1\underbrace{01\cdots1}_{\tilde{n}\text{ bits}}\right\}$, namely all prices that start with a 1, followed by a 0 in the next digit. Node x_1 either has influenced value 0, in case $\overline{x}_1$ bought before, or $2^{\tilde{n}+1} - 1$ otherwise. Hence, x_1 buys exactly for the other half of prices in Π, those with 1 at the second digit instead — $\{110\cdots0_2, \ldots, 111\cdots1_2\}$.

The influence of x_1 on the following nodes $\overline{x}_2, \ldots, \overline{x}_{\tilde{n}}$ is $2^{\tilde{n}-1} = 010\cdots0_2$ and is exerted exactly if the price features 1 on the same digit.

We show the following property by induction on k.

Property 3.4.1. Node $\overline{x}_k$ buys if and only if the $(k+1)$-th digit from the left of the price $\pi \in \Pi$ in binary representation is 0, otherwise x_k buys.

Proof. Assume that, for a $k < n$, the nodes $\overline{x}_1, \ldots, \overline{x}_k$ and their counterparts $x_1, \ldots, x_k$ bought as claimed and consider node $\overline{x}_{k+1}$ for which whether to buy or not is decided in round $2k+2$. The node is influenced from t_{2k+1} by

$$1\underbrace{0\cdots001\cdots1}_{k+2\text{-th digit}}.$$

Adding the potential influences from nodes $x_1, \ldots, x_k$, i.e., for the digits to the left adding 1 if and only if the price has the digit 1 as well, we get an influenced value $\pi^{(1)} \cdots \pi^{(k+1)}01\cdots1_2$ of $\overline{x}_{k+1}$ that shares the first $k+1$ digits with π. Thus $\overline{x}_{k+1}$ buys precisely as claimed, if and only if the $(k+2)$-th digit of π is actually 0. □

All in all, we produced a one-to-one correspondence of prices $\pi \in \Pi$ and customer sets $\{x_1, \ldots, x_{\tilde{n}}\} \cap C(\pi)$, whereof we have $2^{\tilde{n}}$ many. □

Corollary 3.6. *Graph G^x has a number of fragments exponential in the number of nodes.*

Even though we constructed a plethora of fragments, their regularity gives us an easy solution to the optimization problem — price $p_{\max}$ with $3\tilde{n}$ many customers is optimal.

To add some complexity, we extend the graph with a clause gadget corresponding to $\mathcal{J}$.

3.4.2 Clause Gadget

Gadget 3.7 (Clause extension)**.** Let $V^y = \{x_1, \ldots, x_{\tilde{n}}, \overline{x}_1, \ldots, x_{\tilde{n}}, y_1, \ldots, y_{\tilde{m}}\}$, in which the variable nodes are buying according to the construction in Gadget 3.6. The initial values of the newly introduced *clause nodes* $y_1, \ldots, y_{\tilde{m}}$ are set to 0.

We again define A^y implicitly by setting the weights as follows according to instance $\mathcal{J}$ of 3SAT:

$$\begin{aligned} w(x_i, y_j) &= 2^{\tilde{n}+1} - 1 && \text{if } x_i \in y_j \\ w(\overline{x}_i, y_j) &= 2^{\tilde{n}+1} - 1 && \text{if } \overline{x}_i \in y_j \end{aligned}$$

Additionally to the recycled variable nodes and influences of G^x, we set $G^y = (V^x \cup V^y, A^x \cup A^y)$, the extended graph with $n = 4\tilde{n} + \tilde{m}$ nodes and $m = 3\tilde{m} + 5\tilde{n} - 1 + \frac{\tilde{n}\cdot(\tilde{n}-1)}{2}$ arcs. Again, the encoding length of G^y is polynomial in the size of $\mathcal{J}$.

Property 3.4.2. A clause node y_i buys for some price $\pi \in \Pi$ if and only if at least one of its corresponding variable nodes bought.

Proof. By definition the initial value is 0 and the node y_i is influenced exactly by its literals $\{ l_1, l_2, l_3 \} =: y_i$. As every single influence is already raising the influenced value to $p_{\max}$, the clause node buys as claimed. □

With these two gadgets combined, we can already prove our first NP-completeness result.

Lemma 3.7. *The problem PPAI-S is NP-complete.*

Proof. In Corollary 3.2 we already showed that PPAI-S is in NP.

Given an instance $\mathcal{J}$ of 3SAT with variables $x_1, \ldots, x_{\tilde{n}}$ and clauses $y_1, \ldots, y_{\tilde{m}}$, we construct our instance $\mathcal{I}$ according to Gadgets 3.6 and 3.7. In total, the PPAI-S instance $\mathcal{I}$ is defined by G^y, the price bound $\hat{\pi} := 2^{\tilde{n}}$, and the subset $S := \{ y_1, \ldots, y_{\tilde{m}} \}$. The transformation is indeed of polynomial size and therefore we are left to show that the output of $\mathcal{I}$ equals that of $\mathcal{J}$.

We assume $\mathcal{J}$ is `true`, i.e., there is a valid variable assignment, which we can transform to a price $\pi \in \Pi$. By Properties 3.4.1 and 3.4.2 we know that the respective variable nodes buy for this price, leading to every clause node doing it as well and $\mathcal{I}$ is accordingly `true`.

Assume the other way, there is a price $\pi \geq \hat{\pi}$, here actually also $\pi \in \Pi$, satisfying $\mathcal{I}$. Again, by the same properties, we can transform the price to a variable assignment. As in $\mathcal{I}$ every clause node bought, meaning it was influenced, we know that in $\mathcal{J}$ every clause contains a literal which is set to `true`. □

We extend this result and the graph G^y by one extra node in the next subsection.

3.4.3 Collector node

Gadget 3.8 (Clause collection extension)**.** We add a single *collector node* z_+ with initial value $p(z_+) = (1 - \tilde{m}) \cdot (2^{\tilde{n}+1} - 1)$ to the graph G^y of Gadget 3.7. The new node is influenced by all variable nodes as follows:

$$w(y_i, z_+) = 2^{\tilde{n}+1} - 1 \qquad\qquad \text{for } i = 1, \ldots, \tilde{m}$$

This new extended gadget is named G^{z_+}.

Lemma 3.8. *The problem PPAI-v is NP-complete.*

Proof. The newly introduced node z_+ in G^{z_+} has an influenced value of $2^{\tilde{n}+1} - 1 = p_{\max}$ if and only if it is influenced by every single clause node.

If one mere clause node does not buy, corresponding to the clause being not fulfilled, the influenced value is already down to $0 < \hat{\pi} := 2^{\tilde{n}}$. This establishes the equivalence of both problems. □

The next gadget completes this section and finally provides the polynomial time transformation to PPAI.

3.4.4 Revenue Gadget

Gadget 3.9 (Revenue extension)**.** To the graph G^{z_+} of Gadget 3.8, we add a large amount of *revenue nodes* $z_1, \ldots, z_M$ with initial value 0 and a supplementary *negative node* z_- with initial value $p(z_-) = 2^{\tilde{n}} - 1$.

The revenue nodes are influenced by z_+,

$$w(z_+, z_i) = 2^{\tilde{n}+1} - 1 \qquad\qquad \text{for } i = 1, \ldots, M,$$

whereas the negative node itself may influence z_+ by $w(z_-, z_+) = -(2^{\tilde{n}+1} - 1)$.

In total, this is graph G illustrated in Figure 3.2 (bold nodes have non-zero initial value, unlabeled arcs have influence $\pm p_{\max}$, dashed arcs have negative value), consisting of $n = 4\tilde{n} + \tilde{m} + 2 + M$ nodes and $m = 5\tilde{n} + 4\tilde{m} + \frac{\tilde{n}\cdot(\tilde{n}-1)}{2} + M$ arcs. The encoding length of G depends on the size of M. To ensure a polynomial size we require $M = \text{poly}(\tilde{n})$.

Note that we have not yet specified the actual value of M, as we need to devise two individual values for the remaining proofs of this section.

Theorem 3.9. *The problem PPAI is NP-complete.*

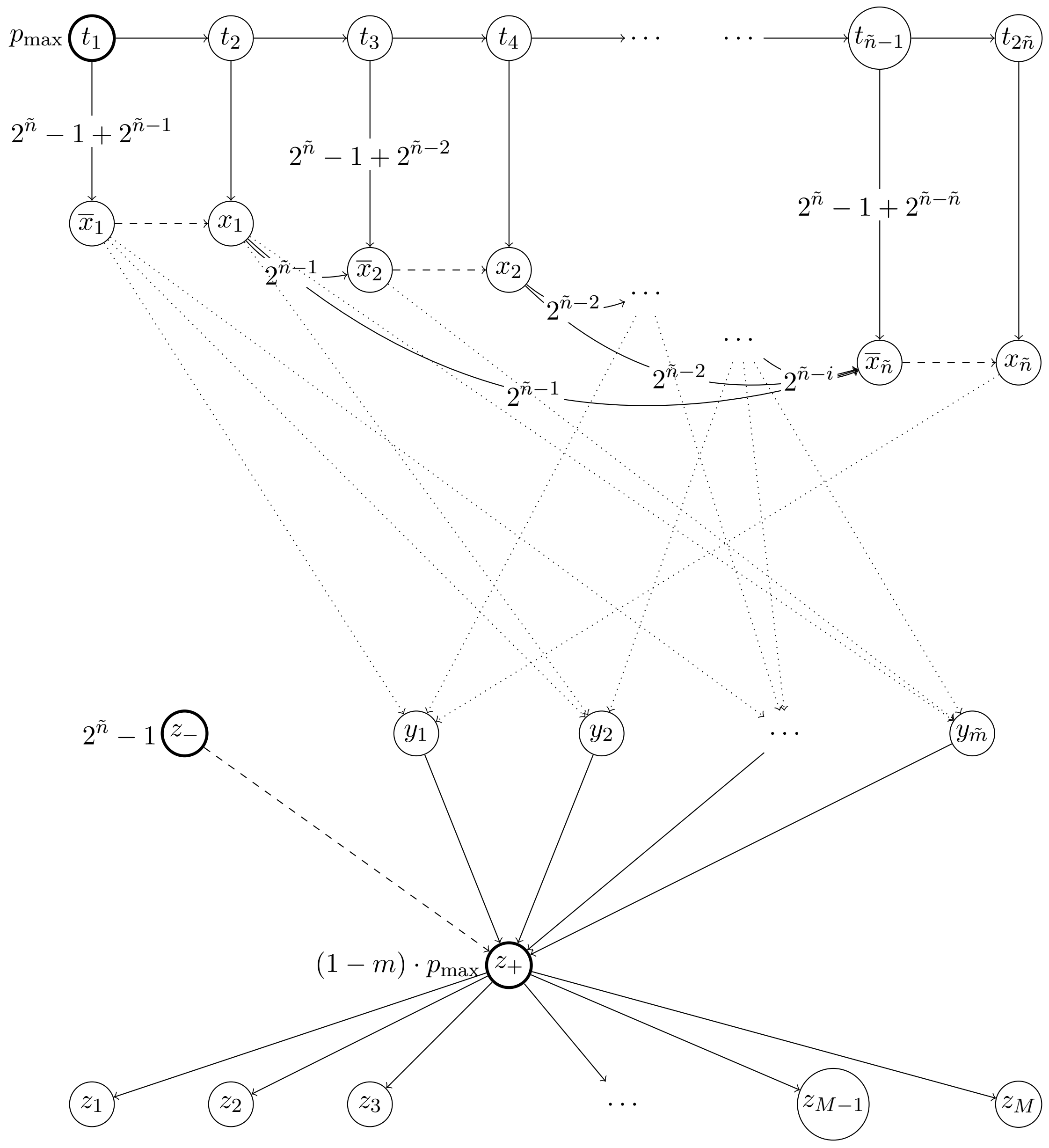

Figure 3.2: Reduction Graph G for 3SAT to PPAI

Proof. The trick of this proof is already specified by the revenue extension — “increase” the amount of purchases of a single node with the introduction of loyal followers.

We want our instance to reach a certain revenue if and only if node z_+ buys. To get that right, we need to set $\hat{R}$ and M correctly.

At first, assume that z_+ is not buying. Therefore, the nodes $z_1, \dots, z_M$ are not influenced and neither buys. The maximum possible revenue R' in such an instance would be for the maximum price and a customer set consisting of all remaining nodes:

$$R' \leq (2^{\tilde{n}+1} - 1) \cdot (4\tilde{n} + \tilde{m} + 1) < 2 \cdot 2^{\tilde{n}} \cdot (4\tilde{n} + \tilde{m} + 1) =: \hat{R}$$

Through the choice of $\hat{R}$, we already require a purchase of z_+ for a positive instance.

The negative node z_- ensures what was previously part of the problem instance, namely that only prices $\pi \geq 2^{\tilde{n}} = \hat{\pi}$ are eligible. If the price is set lower, there is no way the collector node z_+ can have a positive influenced value, which implies we can not sell to the revenue nodes. Remark that the modifications from G^{z+} to G did not change anything about the instance of PPAI-v: Does z_+ buy for any price $\pi \geq 2^{\tilde{n}}$?

To find a lower bound on the revenue of an instance in which z_+ is buying, we need the lowest possible price and the fewest buyers any eligible price $\pi \in \Pi$ can have.

Considering any price $\pi \in \Pi$ fulfilling the instance of PPAI-v, we know that,

- the time nodes buy anyway ($2\tilde{n}$ many),
- either x_i or $\overline{x}_i$ buy ($\tilde{n}$ many),
- the clause nodes buy ($\tilde{m}$ many),
- the negative node does not buy,
- the collector node obviously buys, and
- the revenue nodes follow it (M many).

This totals to $3\tilde{n}+\tilde{m}+1+M$ buyers, while the minimum price is $2^{\tilde{n}}$. For $M := 5\tilde{n}+\tilde{m}+1$, the product of these is exactly $\hat{R}$. Therefore, every price $\pi \geq \hat{\pi}$, for which the collector node z_+ buys, yields the desired revenue. This means that the positive instances of PPAI-v and PPAI coincide on this graph.

As $M = \text{poly}(n)$ we also did not overly inflate the size of the instance — it is still polynomial in the size of the 3SAT instance $\mathcal{I}$ — and thus the proof concludes. □

Corollary 3.10. *Unless P = NP, there is no polynomial time r-approximate algorithm for PPAI$^{\text{opt}}$ for any $r = \mathcal{O}(1)$.*

Proof. In the proof of Theorem 3.9, we tried to get a small revenue gap between the positive and negative instances of the underlying PPAI-v. Thereby we did not need a large amount of revenue nodes.

Following the same technique, we now set $M = \tilde{n}(8r-3) + \tilde{m}(2r-1) + (2r-1)$ instead. Note that this is still polynomial in n for our choice of $r = \mathcal{O}(1)$. The buying behavior of the nodes in G did not change, but the minimum revenue for an instance with z_+ purchasing did increase to

$$2^{\tilde{n}} \cdot (3\tilde{n} + \tilde{m} + 1 + M) = 2^{\tilde{n}} \cdot 2r(4\tilde{n} + \tilde{m} + 1) = r\hat{R}.$$

Assume we have a polynomial time approximation algorithm ALG with approximation ratio r. If there is a price π^* generating a revenue of $r \cdot \hat{R}$, which would need to be a price with $z_+ \in C(\pi^*)$, the algorithm ALG finds a price that generates a revenue of at least $\hat{R}$, which is still only possible if $z_+ \in C(\pi^*)$ buys.

Thus it would actually solve the underlying instance $\mathcal{J}$ of 3SAT in polynomial time. □

Remark 3.10 (Summary). Until now we concluded the following complexities:

- $\mathsf{PPAI}^\pi \in \mathsf{FP}$.
- PPAI-S, PPAI-v, and PPAI are NP-complete. To be more precise, they are weakly NP-complete, since Algorithm Frag runs in pseudo-polynomial time.
- Unless $\mathsf{P} = \mathsf{NP}$, $\mathsf{PPAI}^{\text{opt}} \notin \mathsf{APX}$.

The reductions in this section are all based on the construction of G and its preliminary stages. In Section 3.6 the remaining complexity proof for PPAI-i is presented, when the interconnections of our different problems for restricted graph classes are formally investigated.

3.5 Positive or Negative Influences

In our general model the influences can be a mix of positive and negative (and nonexistent) relations. In possible applications we may just ignore one type of influences.

If some product can easily be shared with friends and neighbors (*Public Goods* [SB75, Led97]) and there is (little to) no benefit in having a multitude of the same product in ones neighborhood, the value of a potential buyer is only decreasing with every lender around, e.g. a lawn-mower you are only rarely using. Thus, in our model, the influences are negative respectively.

But if some product or service is needed to participate in a specific social activity which is increasingly more fun the more join in, e.g. football gear, the value of a potential buyer is only increasing and, hence, the influences are modeled positive.

Definition 3.11. For a graph $G = (V, A)$ and influences w, we set $G^+ = (V, A^+)$ with the positive influences $A^+ = \{\, a \in A : w(a) > 0 \,\}$. We say $G = (V, A)$ with influences w is *positive* if $G^+ = G$.

Analogously we set G^- for the *negative* influences. We set $G^0 = (V, \emptyset)$ and name it the *unaffected* graph. Note that we do not restrict the initial values herein.

We start with the easier cases — the negative graphs.

Remark 3.12. For a negative graph G, the selling stops after the first round, since $p_C(v) \leq p(v)$ for every $v \in V$ and those $v \notin B_1(\pi)$, have initial value $p(v) < \pi$. Thus they are not persuaded by any influence to buy. In fact, we can just assume there are no influences at all and use G^0 instead of G.

Proposition 3.11. *A negative or unaffected graph G has at most n fragments. We can solve PPAI in $\mathcal{O}(n \log n)$ time for such a graph.*

Proof. As stated in the previous remark we have no purchases after the first round. Thus the fragments are bounded by the initial values and it suffices to run Sell for every initial value. This already gives us an algorithm that runs in $\mathcal{O}\left(n^2\right)$ time.

Instead we can sort the values $p(v_1) \leq \cdots \leq p(v_n)$ in $\mathcal{O}(n \log n)$ time. We know that for $\pi = p(v_i)$ the nodes $v_i, \ldots, v_n$, exactly $n - i + 1$ many, buy. Hence, we can compute the revenues $p(v_1) \cdot n, \ldots, p(v_n) \cdot 1$ and find the maximum in $\mathcal{O}(n)$ time. In total this yields the claimed running time of $\mathcal{O}(n \log n)$. $\square$

We can mix positive and negative influences with caution.

Remark 3.13. If some node $v \in V$ has only negative influences, we can still apply Remark 3.12 and remove its neighbors $\mathcal{N}^-(v)$. So, if every node in graph G has either only positive or only negative influences, we can compute the revenue for G^+ instead of G itself.

This already completes the results on negative graphs in this chapter. In the upcoming chapters the results on negative graphs are more elaborate and complex since we can not apply Remark 3.12 anymore.

So in the remainder of this section, we consider only positive influences.

Definition 3.14. For a node $v \in V$, we define $P(v) \coloneqq \{\, \pi \in (0, p_{\max}] : v \in C(\pi) \,\}$ to be the set of *buying prices* of v.

More detailed, for every $v \in V$ and rounds $t = 1, \ldots, n$, we set

$$P_t(v) \coloneqq \{\, \pi : v \in B_t(\pi) \,\} \text{ and } P_{\leq t}(v) \coloneqq \{\, \pi : v \in C_t(\pi) \,\}.$$

The following lemma states that, in contrast to our results for arbitrary influences, higher prices generate fewer buyers.

Lemma 3.12. *In a positive graph $G = (V, A)$ with initial values p and influences w, it holds that for every node $v \in V$ and all rounds $t = 1, \dots, n$, the set $P_{\leq t}(v)$ is an integer interval.*

To be more precise, there is a value $p_t^*(v) := \max_{\pi \in P_{\leq t}(v)} \pi$ such that $P_{\leq t}(v) = (0, p_t^*(v)]$.

Proof. We show the statement by induction on t. Starting with $t = 1$, we immediately have $P_{\leq 1}(v) = (0, p(v)]$ for all nodes $v \in V$.

Now assume that the claim holds for all buying sets in some round t. Thereby it follows that $C(\pi) \supseteq C(\pi')$ for all prices $\pi \leq \pi'$, since every $v \in C(\pi')$ buys for the lower price π.

For any particular node $v \in V$ and the current round t we define the map

$$f : (p_t^*(v), p_{\max}] \to \mathbb{Z}, \quad \pi \mapsto p_{C_t(\pi)}(v).$$

The map f denotes the influenced value of v on the prices at which it did not buy yet. Note that by definition node v buys in round $t+1$ for price π if and only if $f(\pi) - \pi \geq 0$.

The function f is non-increasing, since the potential positive influences from $\mathcal{N}^-(v)$ are only fading with increasing prices. The composed function $f(\pi) - \pi$ is actually decreasing, so there is at most one root x of the function. It follows that $P_{t+1}(v) = (p_t^*(v), x]$ and in case there is no root either $P_{t+1}(v) = \emptyset$ or $P_{t+1}(v) = (p_t^*(v), p_{\max}]$. In every case, it holds that $P_{\leq t+1}(v)$ is again an integer interval. □

Corollary 3.13. *For all positive graphs G the amount of fragments is bounded by $|\text{frag}(G)| \leq n^2$. Hence, Algorithm Frag solves $\mathsf{PPAI}^{\text{opt}}$ in time $\mathcal{O}\left(n^3 \cdot (m + n \log n)\right)$.*

Proof. Consider an arbitrary t-fragment $(x, y]$. Then, by definition, there can be no $p_t^*(v)$ with $x < p_t^*(v) < y$ as v would otherwise be a hitter in round t. Hence, fragments can start and end only at values $p_{t'}^*(v)$ for $t' \leq t$, of which there are at most $tn \leq n^2$ many. The running time follows from Theorem 3.4. □

By the above corollary, we already have a polynomial time algorithm for $\mathsf{PPAI}^{\text{opt}}$ on positive graphs. We continue for a faster algorithm.

Remark 3.15. For the sake of a shorter notation, let $p^*(v) := p_n^*(v)$ denote the maximum price at which node v would buy in any round. The maximum revenue R^* then satisfies $R^* = \max_{v \in V} p^*(v) \cdot |C(p^*(v))|$.

Note that this equals the revenue of a graph G^0 with initial values $p(v) := p^*(v)$ for all nodes $v \in V$. So, as previously shown in Property 3.11, given these values we can compute the revenue in $\mathcal{O}(n \log n)$ time.

```
Algorithm 4: FixHighest($G$)
Data: A graph $G = (V, A)$ with weights $p$ and positive $w$.
Result: The optimal revenue $R^*$ with corresponding price $\pi^*$.
Initialize: Revenue $R^* = 0$, values $p'(v) = p(v)$ for all $v \in V$, and $F = \emptyset$.
// $F$ for fixed nodes
while $V \setminus F \neq \emptyset$ do
    Compute $\pi' = \max_{v \in V \setminus F} p'(v)$ and $N = \{ v \in V \setminus F : p'(v) = \pi' \}$.
    $F = F \cup N$
    if $\pi' \cdot |F| > R^*$ then $\pi^* = \pi'$ and $R^* = \pi' \cdot |F|$
    Update $p'(v) = \min \{ p_F(v), \pi' \}$ for all $v \in V \setminus F$.
return $R^*$ and $\pi^*$
```

Algorithm FixHighest computes the values $p^*(v)$ in its run and outputs the optimum revenue price π^* and revenue R^*. The idea is to fix the highest values $p^*(v)$ first, starting with some $p(v) = p_{\max} = p^*(v)$.

Theorem 3.14. *For positive graphs Algorithm FixHighest correctly solve PPAI$^{\text{opt}}$ in $\mathcal{O}(m + n \log n)$ time.*

Proof. Denote by F_i, N_i, π'_i, and $p'_i(v)$ the values of algorithm FixHighest at the end of iteration i of the **while**-loop.

Correctness: By definition of N, the current nodes with highest influenced value, it holds that $N_i \neq \emptyset$ for each round until every node is fixed. Hence, apart from the actual revenue computed, we need to show that in every round i, the newly fixed nodes $v \in N_i$ satisfy $p^*(v) = p'_{i-1}(v)$, i.e. $p^*(v) = \pi'_i(v)$.

The highest price π' is non-increasing over the rounds, as

$$\pi'_{i+1} = \max_{v \in V \setminus F_i} \min \{ p_{F_i}(v), \pi'_i \} \geq \pi'_i.$$

In contrast, the influenced values $p'(v)$ are non-decreasing, since as long as they are updated it holds

$$p'_i(v) \leq p_{F_i} \leq p_{F_{i+1}} \text{ and } p'_i(v) \leq \pi'_{i+1},$$

therefore it is also less or equal to $\min \{ p_{F_{i+1}}(v), \pi'_{i+1} \} = p'_{i+1}$. From Lemma 3.12 we conclude that $C(\pi'_i) \subseteq C(\pi'_{i+1})$.

Property 3.5.1. For all iterations i, the following conditions hold:

1. $F_i \subseteq C(\pi'_i)$ and
2. $p'_i(v) \leq p^*(v)$ for all $v \in V \setminus F_i$, i.e., $v \in C(p'_i(v))$.

Proof. Note that the value $p'_{i-1}(v)$ is not updated anymore if $v \in N_i$, thus it suffices to observe the nodes up to the iteration before they get fixed.

In iteration $i = 1$, we have $\pi'_1 = p_{\max}$. Thus $F_1 = N_1 = \{ v \in V : p(v) = p_{\max} \} \subseteq C(p_{\max})$, (a subset of) the nodes that buy for any price $\pi \in (0, p_{\max}]$. Thereupon their influence is always present and the other nodes $v \in V \setminus F_1$ definitely buy for $p'_1(v) = \min \{ p_{\max}, p_{F_1}(v) \}$. Thus the induction basis is concluded.

We now assume that the two conditions hold at the end of iteration i and show they still hold after iteration $i+1$. As $F_i \subseteq C(\pi'_i)$ by hypothesis, we know that $F_i \subseteq C(\pi'_i) \subseteq C(\pi'_{i+1})$.

Since $F_{i+1} = F_i \cup N_{i+1}$, we yet need $N_{i+1} \subseteq C(\pi'_{i+1})$. For nodes $v \in N_{i+1}$ it holds that $p'_i(v) = \pi'_{i+1}$ and $p'_i(v) \leq p^*(v)$ by induction hypothesis. By Lemma 3.12 it follows that v indeed buys for π'_{i+1}.

In round $i + 1$ we update the values $p'_{i+1}(v) = \min \left\{ p_{F_{i+1}(v)}, \pi'_{i+1} \right\}$. Since we already showed that all nodes in F_{i+1} buy for this value and the influenced value of v is at least $p'_{i+1}(v)$, it also follows that $v \in C(\pi'_i(v))$. □

Property 3.5.2. When the price π' drops in round $i + 1$, i.e., $\pi'_{i+1} < \pi'_i$, we have $F_i = C(\pi'_i)$

Proof. Assume there is some $v \in C(\pi'_i) \setminus F_i$. Then, for the plain selling of the product for price π'_i there is some round t such that $v \in B_t(\pi'_i)$. Either v is influenced by some other node $u \in C(\pi'_i) \setminus F_i$ from an earlier round — which we can subsequently examine — or it contradicts the assumption. □

So after all the nodes are fixed to their value $p^*(v)$. The computation of R^* is then just keeping tabs of the best possible revenue.

Runtime: We can implement the algorithm using Fibonacci heaps [FT87]. Since our values p' are increasing, we use a max-heap to store the values $p'(v)$.

Starting with the initialization we do n `insert` operations. Computing π' and N is a `delete-max` operation. And updating the values p', i.e., applying `increase-key` is triggered once by each arc — m times.

In total this yields us the claimed complexity, the same as for Dijkstra's algorithm with Fibonacci heaps □

3.6 PPAI on Special Graph Classes

With the restriction on influences, we got our first “benevolent” instances. In this chapter we devise bounds on $|\mathrm{frag}(G)|$ for more special graph classes.

Recall the notion of fragments. Given a graph G — and implicitly its initial values and weights — we denote by $\mathrm{frag}(G)$ the collection of fragments on this graph. For clarity and a shorter notation we now define $\mathrm{frag}(n)$.

Definition 3.16. Given any class of graphs

$$\mathcal{G}_n = \{\, G = (V, A), w, p : |V| = n \text{ and any common property} \,\},$$

we set, in context of $\mathcal{G}_n$, the value $\mathrm{frag}(n) = \max_{G \in \mathcal{G}_n} |\mathrm{frag}(G)|$.

Likewise, we denote the amount of additional fragment endpoints from round $t-1$ to t with $\mathrm{frag}_t(n)$.

Remark 3.17. In the previous sections we already showed

- the arbitrary graph class contains Gadget 3.6 and therefore $\mathrm{frag}(n) \geq 2^{n/4}$,
- negative or unaffected graphs have $\mathrm{frag}(n) = n$, and
- positive graphs have at most $\mathrm{frag}(n) \leq n^2$.

3.6.1 Round Bound

Definition 3.18. For some t-fragment $(x, y]$ with customers C_t, the complement $V \setminus C_t$ is the set of *active nodes.*

The *activity sequence* denotes the amount of active nodes for the entire collection $\mathrm{frag}_t(G)$ of t-fragments.

Theorem 3.15. *For the class of arbitrary graphs, we have in any round $t = 1, \ldots, n$ at most* $\mathrm{frag}_t(n) \leq \binom{n}{t}$ *new endpoints for our temporary fragments. Altogether it follows that* $\mathrm{frag}(n) \leq 2^n - 1$.

Proof. Assume for now that we have a graph G with the maximum amount of fragments.

In the first round we normally start with the 0-fragment $(0, p_{\max}]$. For the sake of simplicity and without negative consequences, we instead use $(0, \infty]$ in this proof. Note that this inactual “proto”-fragment is consequently not summarized in the claim of this Lemma.

Previous to round 1 no node bought, which means we still have $n = |V \setminus \emptyset|$ potential buyers left — the *active nodes.* We denote this by the one-element-sequence $a^1 = (n)$

We now follow the initial call of $\mathsf{Frag}(G, (0, \infty], \emptyset)$. At the beginning we have the (un)influenced values $p_\emptyset(v)$ for all $v \in V$. The 1-fragments are created by splitting $(0, \infty]$ through the various hitters $H \subseteq V \setminus C_0 = V$.

We assume now, and in the following, that for every subsequent call of Frag, we always have the maximum amount of hitters ($H = V \setminus C$) and furthermore, their values are distinct ($|H| = |P|$).

Thus, as claimed, we can create $n \leq \binom{n}{1}$ new fragment endpoints in the first round. For the initial values this implies the increasing sequence $0 < p(v_1) < \cdots < p(v_n)$.

In contrast to the original algorithm Frag that used a depth-first technique, in this proof we inspect all calls for the subsequent rounds at the same time. For the second round these are namely

$$\begin{aligned}
&\mathsf{Frag}(G, (0, p(v_1)], V)\\
&\mathsf{Frag}(G, (p(v_1), p(v_2)], V \setminus \{v_1\}),\\
&\mathsf{Frag}(G, (p(v_2), p(v_3)], V \setminus \{v_1, v_2\}),\\
&\vdots\\
&\mathsf{Frag}(G, (p(v_{n-1}, p(v_n)], \{v_n\})
\end{aligned}$$

Note that we did not call $\mathsf{Frag}(G, (p(v_n), \infty], \emptyset)$ as the algorithm already excludes any fragment without a buyer in the current round. Altogether we set $a^2 = (0, 1, 2, \ldots, n-1)$ as our new activity sequence. The first entry, as well as the corresponding call $\mathsf{Frag}(G, (0, p(v_1)], V)$, is actually superfluous. The following elementary Property 3.6.1 justifies along the way, that we can delete all 0 entries in the activity sequences.

Property 3.6.1. Given an activity sequence $a^t = (a_1^t, \ldots, a_\alpha^t)$ with $a_i^t > 0$ for all $i = 1, \ldots, \alpha$, we can have at most $\sum_{i=1}^{\alpha} a_i^t$ hits in round t. In the maximum case, the activity sequence of the next round is as follows:

$$a^{t+1} = \Big(0, 1, \ldots, a_1^t - 1, \quad 0, 1, \ldots, a_2^t - 1, \quad \ldots, \quad 0, 1, \ldots, a_\alpha^t - 1\Big)$$

An illustration of the first two rounds is given in Figure 3.3.

By the way, as already illustrated in the Figure, using Property 3.6.1 on a^2 yields us the bound of $1 + \cdots + n - 1 = \binom{n}{2} \leq \mathrm{frag}_2(n)$.

Proof. The total amount of hits is again bounded by the total amount of active nodes.

Regarding the next activity sequence a^{t+1}, we conclude Algorithm Frag again. For a single fragment with a' active nodes we generate the canonical a' hits. In preparation of the subsequent calls we compute the customer sets C^i of decreasing sizes $n, \dots, n-(a'-1)$, whereas the amount of active nodes increases as $0, \dots, a'-1$.

Hence, for the entirety of current fragments we just sum the amount of hits and concatenate the single activity sequences. ☐

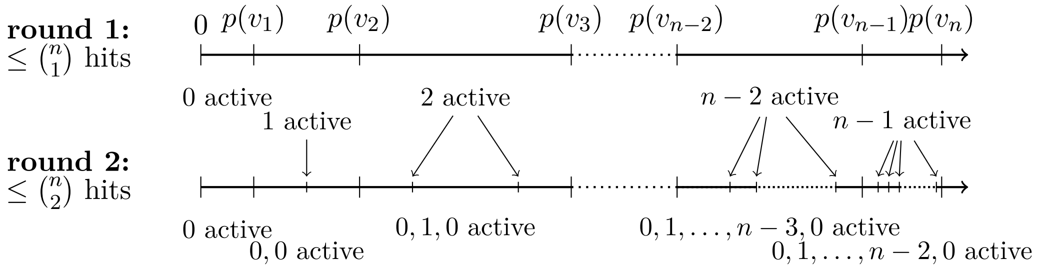

Figure 3.3: Number of potential hits

For the analysis of the activity sequences, the correct element order is not needed, but rather the quantity a certain amount of active nodes is there. Let the *amount sequence* be $r(a) = (r_1(a), \dots, r_n(a))$, with $r_j(a)$ denoting how many elements $a_i = j$ the sequence contains. In this notation we start for $a^1 = (n)$ with $r(a^1) = (0, \dots, 0, 1)$ and $r(a^2) = (1, \dots, 1, 0)$

Property 3.6.2. For a round t and sequence $r(a^t) =: r^t = \left(r_1^t, \dots, r_n^t\right)$, the sequence $r\left(a^{t+1}\right) =: r^{t+1} = \left(r_1^{t+1}, \dots, r_n^{t+1}\right)$ satisfies:

$$r_i^{t+1} = r_{i+1}^t + \dots + r_n^t \qquad \text{for } i = 1, \dots, n-1$$

The maximum amount sequences are as follows:

$$r^t = \left(\binom{n-2}{t-2}, \binom{n-3}{t-2}, \dots, \binom{t-2}{t-2}, 0, \dots, 0\right) \qquad \text{for } t = 3, \dots, n$$

Proof. As previously discussed, a fragment with a_i active nodes may spawn one fragment for each $1, \dots, a_i - 1$. So to compute the amount of fragments r_i^{t+1}, we need to sum the amount of bigger fragments of the previous round.

We show again by induction, given that round k is properly described by r^k as stated, the values of r^k are also correct.

For this property it remains to show for all $j = 1, \dots, n-t$ that

$$r_j^{k+1} = \binom{n-j-1}{k-2} = \binom{n-j-2}{k-2} + \dots + \binom{k-2}{k-2}.$$

With the series (cf. [BS79])

$$\sum_{c=0}^{a} \binom{c}{d} \binom{a-c}{b-d} = \binom{a+1}{b+1}, \tag{3.2}$$

which holds for values $0 \leq d \leq b \leq a$, and with $d = 0$ we know

$$\binom{n-j-1}{k-2} = \sum_{c=0}^{n-j-2} \binom{n-j-2-c}{k-2} = \binom{n-j-2}{k-2} + \cdots + \binom{k-2}{k-2}.$$

This concludes the proof of Property 3.6.2. □

Finally we can show the claim by summing the potential hits $\sum_{i=1}^{n} i \cdot r_i^t$. Following by the last property this is:

$$\sum_{i=1}^{n} i \cdot r_i^t = \sum_{i=1}^{n} i \cdot \binom{n-1-i}{t-2}$$

Again, with the Formula (3.6.1), now for $d = 1$, we conclude

$$\sum_{i=1}^{n-1} i \cdot \binom{n-1-i}{t-2} = \binom{n}{t}.$$

□

With Theorem 3.15 we established a first upper bound on the amount of fragments for arbitrary graphs. Yet, the assumption that on every fragment every active node hits as claimed is quite generous. And it requires additional characteristics, as longevity.

Definition 3.19. Given a graph G, we define the *decay* as the last round in which v buys, i.e., $\text{decay}(v) := \max\{\, t : v \in B_t(\pi) \text{ for some } \pi \in (0, p_{\max}] \,\}$.

With $\text{decay}(G) := \max_{v \in V} \text{decay}(v)$ we denote the last round in which any purchase occurs.

Note that in Remark 3.2 we already established $\text{decay}(G) \leq n$.

By Theorem 3.15 we conclude:

Corollary 3.16. *For a graph G there can be at most*

$$|\mathrm{frag}(G)| \leq \sum_{i=1}^{\mathrm{decay}(G)} \binom{n}{i} \leq \mathrm{decay}(G) \cdot n^{\mathrm{decay}(G)}$$

fragments.

So given a constant decay, we would have at most a polynomial amount of fragments and thus a polynomial algorithm with Frag.

Proposition 3.17. *A node $v \in V$ can only buy in round $t > 2$ for a price π if some $u \in \mathcal{N}^-(v) \cap B_{t-1}(\pi)$ has positive influence $w(u,v) > 0$.*

Proof. For v to buy in round t, we require that it did not do so before, in particular, in round $t-1$ we have $p_{C_{t-2}(\pi)}(v) < \pi$. Yet $\pi \leq p_{C_{t-1}(\pi)}(v)$, therefore

$$\sum_{u \in B_{t-1}(\pi) \cap \mathcal{N}^-(v)} w(u,v) > 0$$

and at least one of those influences was positive. □

From this property we devise a bound on decay(G).

Corollary 3.18. *Let $L(v)$ be the longest path — meaning path with most arcs — to v with exclusively positive weighted arcs and $|L(G)| := \max_{v \in V} |L(v)|$. We have* $\mathrm{decay}(v) \leq |L(v)| + 1$ *and equally* $\mathrm{decay}(G) \leq |L(G)| + 1$*.*

With Corollary 3.16 we have the bound $\mathrm{frag}(G) \leq n^{L(G)+2}$*.*

The problem of finding a longest path in a graph, in this case G^+, is NP-hard [GJ79].

3.6.2 Greedy Hitting

Given the round bound from the last subsection, the question remains whether getting maximum hits in every round is achievable at all.

In this subsection we show that the fully greedy approach of getting as much hits as possible reaches its decline already after three rounds.

Lemma 3.19. *Given a graph G with $n \geq 4$, we can not achieve*

$$|\mathrm{frag}_4(G)| = n + \binom{n}{2} + \binom{n}{3} + \binom{n}{4}.$$

Proof. At first, we need distinct initial values $p(v)$, as otherwise we can not even achieve the maximum of n hits in the first round. Let $p(v_1) < \cdots < p(v_n)$ be the sorted sequence of those. In the second round we have amongst others the three 1-fragments $(p(v_1), p(v_2)], (p(v_2), p(v_3)], (p(v_3), p(v_4)]$. On all these fragments we already have $V \backslash C \subseteq \{v_1, v_2, v_3\}$ as the last possible active nodes. Without loss of generality we can say that the nodes are not influenced by $v_5, \ldots, v_n$ and subsume all influences to $w(v_4, v)$, since on these fragments, these nodes buy collectively.

We show for $v \in \{v_1, \ldots, v_3\}$ that the full amount of hits in the first four rounds can not even be achieved amongst those fragments. Assume for the sake of contradiction that it is possible.

The node v_1 is active on all 1-fragments. For v_1 to hit everywhere, the influenced value has to increase for higher fragments. This implies that

$$\begin{aligned} p(v_1) < p(v_1) + \sum_{i=2}^{n} w(v_i, v_1) < p(v_2) < p(v_1) + \sum_{i=3}^{n} w(v_i, v_1) \\ < p(v_3) < p(v_1) + \sum_{i=4}^{n} w(v_i, v_1) < p(v_4). \end{aligned}$$

Thus, it directly follows $w(v_4, v_1) > 0$ and $w(v_3, v_1), w(v_2, v_1) < 0$. After the second round the influenced value of v_1 can not increase. So we avoid v_1 being an active node by having v_1 providing the highest hit in all three fragments.

We can conclude this incomplete order of values, for "?" as representation of v_2 *and* v_3.

$$\begin{aligned}
&p(v_1) \\
&< p(v_1) + w(v_4, v_1) + w(v_3, v_1) + w(v_2, v_1) = p_V(v_1) && (3.3) \\
&< p(v_2) \\
&< p(v_2) + w(v_4, v_2) + w(v_3, v_2) \\
&\quad < p(v_2) + w(v_4, v_2) + w(v_3, v_2) + w(v_1, v_2) = p_V(v_2) && (3.4) \\
&< p(v_1) + w(v_4, v_1) + w(v_3, v_1) \\
&< p(v_3) \\
&< p(?) + w(v_4, x) \\
&\quad < p(?) + w(v_4, ?) + w(v_1, ?) + w(?, ?) = p_V(?) && (3.5) \\
&< p(?) + w(v_4, ?) \\
&\quad < p(?) + w(v_4, ?) + w(v_1, ?) \\
&\qquad < p(?) + w(v_4, ?) + w(v_1, ?) + w(?, ?) = p_V(?) && (3.6) \\
&\quad < p(?) + w(v_4, ?) + w(v_1, ?) \\
&< p(v_1) + w(v_4, v_1) \\
&< p(v_4)
\end{aligned}$$

Which Note that we have hits at $p_V(v_1)$ and $p_V(v_2)$ in Equations (3.6.2) and (3.6.2). Additionally we have the fully influenced values in Equations (3.6.2) and (3.6.2). Since only $p_V(v_3)$ is left, these four lines contain a duplicate. And as we assumed that they are distinct, there is no choice of initial values and influences such that we get the full amount of hits in the first four rounds. □

Observation 3.20. The scheme specified in the following generates a graph with

$$\operatorname{frag}(G) = \operatorname{frag}_3(G) = n + \binom{n}{2} + \binom{n}{3}.$$

Set n distinct initial values $p(v_1) < \cdots < p(v_n)$.

For the second round, set the influences $w(v_i, v_j)$ for $i > j$ such that in every 1-fragment the subsequent hits of round 2 are inverse sorted. In particular $w(v_n, v_i) > 0$ are the only positive influences here and v_1 is the highest hitter as we assumed in the proof of Lemma 3.19.

The remaining unspecified influences $w(v_i, v_j)$ for $j > i$ can finally be set accordingly. Now $w(v_1, v_i) > 0$ are the only positive influences and the hits are again ordinary sorted.

We illustrate the sign of the influences in the matrix W, where $w_{i,j} = w(v_i, v_j)$

$$W = \begin{pmatrix} 0 & + & \cdots & + & 0 \\ - & \ddots & - & - & \vdots \\ \vdots & \ddots & \ddots & - & \vdots \\ - & \cdots & - & \ddots & \vdots \\ + & \cdots & \cdots & + & 0 \end{pmatrix}$$

For example, the actual values for a minimum $p_{\max}$ and 5 nodes are specified as follows:

$$p(v_1) = 1, p(v_2) = 3, p(v_3) = 7, p(v_4) = 15, p(v_5) = 26 \text{ and}$$

$$W = \begin{pmatrix} 0 & 1 & 5 & 8 & 0 \\ -4 & 0 & -4 & -4 & 0 \\ -8 & -7 & 0 & -3 & 0 \\ -11 & -10 & -10 & 0 & 0 \\ 24 & 18 & 11 & 1 & 0 \end{pmatrix}$$

Corollary 3.20. *By Lemma 3.19 and Observation 3.20 we conclude* $\operatorname{frag}(4) = 14$.

3.6.3 Neighborhood

A social network is typically rather large and full of complex relations. Yet one can probably classify its own relations to a few groups of equally influential people.

Definition 3.21. Given the graph G and a node v, we set

$$\gamma(v) \coloneqq \left|\{\, w(u,v) : u \in \mathcal{N}^-(v) \,\}\right|,$$

the amount of different influences on v. For the graph itself we define $\gamma(G) = \max_{v \in V} \gamma(v)$.

Remark 3.22. Any endpoint of a fragment corresponds to at least one particular hit. More formally, for any fragment $(x, y]$ there is a node v and a set of customers such that $p_C(v) = x$.

This enables us to provide another bound.

Proposition 3.21. *A node $v \in V$ can have at most $n^{\gamma(v)+2}$ different influenced values. Furthermore, it follows that $|\mathrm{frag}(G)| \leq n^{\gamma(G)+3}$.*

Proof. We want to determine the possible amount of distinct influenced values for every node $v \in V$.

For v we may choose for every neighbor, i.e., exactly $g^-(v)$ times, if influence is exerted and (in case it is) which influence is exerted on v. Thus we have $\gamma(v) + 1$ choices for a single arc.

As we are not interested in the particular order of influences, the number of possibilities is $\left(\!\binom{\gamma(v)+1}{g^-(v)}\!\right)$, the amount of multisets. With $\left(\!\binom{n}{k}\!\right) = \binom{n+k-1}{k}$ and $\gamma(v) \leq g^-(v) \leq n - 1$ we have

$$\begin{aligned}
\left(\!\binom{\gamma(v)+1}{g^-(v)}\!\right) &= \binom{\gamma(v)+g^-(v)+2}{g^-(v)} \\
&= \binom{\gamma(v)+g^-(v)+2}{\gamma(v)+2} \\
&= \frac{(\gamma(v)+g^-(v)+2)\cdots(g^-(v)+1)}{(\gamma(v)+2)\cdots 1} \\
&\leq (g^-(v)+1)^{\gamma(v)+2} \leq n^{\gamma(v)+2}.
\end{aligned}$$

□

This enables us to compute the optimum price and revenue in polynomial time, as long as every person has classified all acquaintances in a small (constant) amount of groups.

By setting a different value for every arc, we get $g^-(v) = \gamma(v)$ and thus can already deduce that PPAI is solvable in polynomial time if the in-degree is bounded. But we can provide a better bound for it.

Lemma 3.22. *A single node $v \in V$ can hit at most $2^{g^-(v)}$ times.*

Given $\Delta^-(G) := \max_{v \in V} g^-(v)$, the maximum in-degree in G, it follows

$$|\mathrm{frag}(G)| \leq n \cdot 2^{\Delta^-(G)}.$$

Proof. The influenced value $p_C(v)$ depends by definition only on $C \cap \mathcal{N}^-(v)$. These are sets of size up to $g^-(v)$ and we have at most $2^{g^-(v)}$ different influenced values as claimed.

Since by Remark 3.22 the total amount of fragments is bounded by the potential influenced values, we have $|\mathrm{frag}(G)| \leq \sum_{v \in V} 2^{g^-(v)} \leq n \cdot 2^{\Delta^-(G)}$. □

For graphs with $\gamma(G) = 1$ we return to the results of Section 3.5.

Lemma 3.23. *If $\gamma(G) = 1$, i.e., the potential buyers are indifferent as of who exactly is buying and only the amount of (acquainted) buyers counts, we can determine $P(v)$ by running FixHighest on G^+.*

Proof. All $v \in V$ with $w(u, v) \leq 0$ will buy at exactly $[0, p_v]$ in the first round and never in any other round. This is the same as for no influence at all and we can switch to G^+ which can be solved in $\mathcal{O}\left(n^2\right)$. □

So far we only had results for the in-degree. The following Proposition 3.24 shows why we can not develop a polynomial bound for a limit on the out-degree.

Gadget 3.23. Given a node $u \in V$ of graph $G = (V, A)$, we substitute the arcs to $\mathcal{N}^+(u)$ by $G_{u\,\mathrm{out}}$ as follows (see Figure 3.4):

Add a full binary out-tree of height $h := \lceil \mathrm{ld}(n-1) \rceil - 1$ to u, where the new nodes have no initial value and get an influence of $p_{\max}$ from their predecessor. Expand the binary tree with the nodes $v \in \mathcal{N}^+(u)$, which are then influenced with the original weight $w(u, v)$.

Applying this to all original nodes $u \in V$ we get the new graph G_{out}.

Proposition 3.24. *The gadget graph G_{out} has $n + n \cdot \left(2^{\lceil \mathrm{ld}(n-1) \rceil} - 2\right) \leq 2n^2 - 3n = \mathcal{O}\left(n^2\right)$ nodes and the original nodes from V still buy for the same prices.*

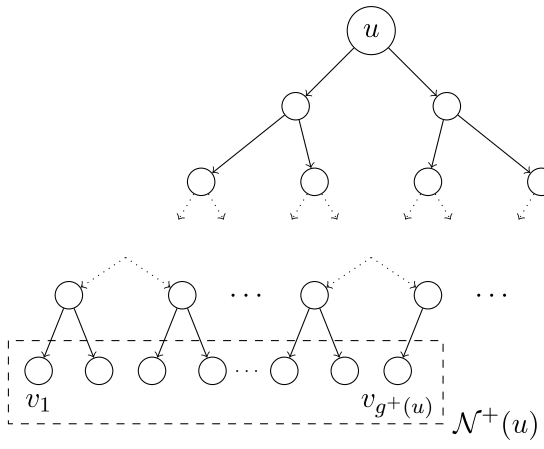

Figure 3.4: Out-degree Gadget

Proof. For every gadget $G_{u\,\mathrm{out}}$ we add $2^{\lceil \mathrm{ld}(n-1) \rceil} - 2$ new nodes. During the selling, they have either value 0 (uninfluenced) or $p_{\max}$ (influenced), which implies they buy if and only if their predecessor buys. This results in $2^{\lceil \mathrm{ld}(n-1) \rceil} - 2$ additional purchases for every single original node $u \in V$. The leaves buy exactly h rounds later, thus the influences to $\mathcal{N}^+(u)$ are exerted with a delay of $h + 1$ rounds instead of one.

Since this holds for every influence on the original $v \in V$ it follows that the buying satisfies $B_t^G(\pi) = B_{h\cdot(t-1)+1}^{G_{\mathrm{out}}}(\pi)$. $\square$

We can even transform the graph G from Gadget 3.9, thereby reproducing the proof of Theorem 3.9.

Corollary 3.25. *The problem PPAI is still NP-complete for graphs with a maximum out-degree of* 2.

Thereby this also applies to graphs with average degree of at most 2.

This raises the question: Is PPAI still hard if the out-degree is at most 1? If we assume that the graph is still weakly connected, this is a tree with possibly one additional arc. We deal with the problem on trees in Section 3.8.

We already assumed that a person is indifferent about who exactly is a customer as they are equally influenced by everyone. Now we assume the contrary, i.e., every node exerts the same influence on the rest of the network.

Lemma 3.26. *For a complete graph G with $w_v := w(v,u)$ for all $v \in V$ and all $u \in V \setminus \{v\}$ it holds that* $\mathrm{frag}(G) \leq \binom{n+1}{2}$.

Proof. Given a set of customers C, we know that $p_C(v) - p(v) = p_C(u) - p(u)$ for nodes $u, v \in V \setminus C$. This implies that for sorted $p(v_1) \leq \cdots \leq p(v_n)$ this order is maintained for all possible influenced values.

Hence, for any price π and round t we know that there is some node v_i such that $C_t(\pi) = \{ v_i, \dots, v_n \}$ or $C_t(\pi) = \emptyset$. We deduce that any node $v_i \in \{ v_1, \dots, v_n \}$ can be influenced by at most $n - i + 1$ customer sets, therefore can hit only as often. In summation we have at most $\binom{n+1}{2}$ fragments as claimed. □

3.6.4 Undirected Setting

So far we investigated directed graphs and simply by adding the inverse arcs with equal influence, we can create symmetric, i.e., undirected, influences. Hence, we can use all algorithms as usual.

The other way round, given any directed graph, such as the reduction graph G of Gadget 3.8, is there a way to "undirect" it? We show in Gadget 3.24 that it is possible indeed and, hence, PPAI is still NP-complete for undirected graphs.

Gadget 3.24 (Undirect Gadget)**.** Given a directed graph $G = (V, A)$ we construct undirected $G_{\text{sym}} = (V \cup V_A, E)$ as follows.

Every arc $(u, v) = a \in A$ is substituted by gadget $G_a = (\{u, v\} \cup V_a, E_a)$ (see Figure 3.5). The nodes $V_a = \{ a_1, a_2, a_3 \}$ have no initial value. The influence $w(a_2, v) = w(u, v)$ is set to the actual influence from u to v.

In order to ensure only the original direction we furthermore set $w(u, a_1) = w(a_1, a_2) = w(a_2, a_3) = p_{\max}$ and $w(u, a_3) = -p_{\max}$.

Proof. Given a price π, we assume that u buys in round t before v does. In round $t + 1$ node a_1 buys. After that in round $t + 2$ node a_2 buys. In round $t + 3$ the influence $w(u, v)$ arrives at v, two rounds later than usual.

Assume on the contrary that v buys first. Depending on $\pi \leq w(u, v)$ node a_2 buys in round $t + 1$. If it doesn't, nothing arrives at u as desired. If it does indeed, we have in round $t + 2$ nodes a_1 and a_3 buying. Both exert influence on u, but they cancel out each other, so nothing happens.

Now what if they both buy in the same round or before the influences arrive. Given that we change every single arc in the graph to this G_a, there are normal purchases in round 1. In round 2 we then have only new gadget nodes influenced, i.e., no single node $v \in V$ changes the influenced value, so no v buys. This still holds for round 3.

In round 3 the influences finally pushed through to the original V, whereas no new gadget node buys. Following this we get $C(\pi) \cap V = \bigcup_{t=1,\dots,|V|} B_{3t-2}(\pi)$, exactly the same buyers as in the original directed G.

Note that we do not always have the same amount of buyers from the arc gadgets G_a. Starting from v, we have possibly three purchases or even none at all. Otherwise we have exactly 2 in any case.

In total we added $3m$ nodes and transformed the m arcs to $5m$ edges. Thus G_{sym} is a polynomial transformation of G. □

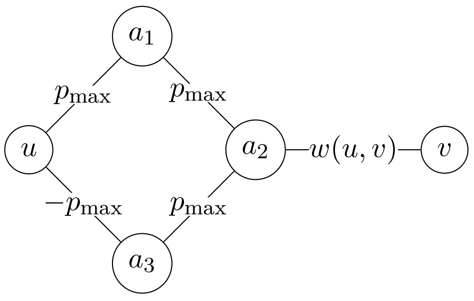

Figure 3.5: Undirect Gadget $G_{(u,v)}$

Considering problems PPAI-v and PPAI-S this does not change a bit for the actual instances.

In the next section we develop the equivalence of all decision problems, including for undirected graphs. We already know for now:

Corollary 3.27. *The problem PPAI-v is NP-complete for undirected graphs.*

Proof. We polynomially transform the graph G^{z+} of Gadget 3.8 to G^{z+}_{sym}.

Actually we only need to transform the edges from variable to clause nodes. By relaxing to symmetric influences even the behavior in G^x does not change. □

3.7 Modifications and Equivalence

Given any graph $G = (V, A)$ the important part for our problems is the specific behavior of the nodes for prices $\pi \in (0, p_{\max}]$. In the following we define three gadgets that conserve "the original behavior".

Gadget 3.25 (Restrict Gadget)**.** For graph $G = (V, A)$ and price range $(0, p_{\max}]$ we construct $G_{|(x,y]} = (V \cup \{\alpha_+, \alpha_-\}, A \cup A_{|(x,y]})$ such that on $(x, y] \subseteq (0, p_{\max}]$ we still have the same customers $C(\pi) \cap V$ as before and $C(\pi) \cap V = \emptyset$ otherwise.

Set the initial values from all $v \in V$ to 0 so they do not buy in round 1 for any positive price. The influences for $a \in A$ stay the same.

Now to start the selling on V we set $p(\alpha_+) = y$ and $w(\alpha_+, v) = p(v)$ for all $v \in V$. With one round delay and only for prices in $(0, y]$ we set them to their initial values.

Again for α_- we set $p(\alpha_-) = x$ and $w(\alpha_-, v) = -p(v)$ for all $v \in V$. So for prices in $(0, x]$ the value cancels out and only on $(x, y]$ the selling progresses as desired.

Gadget 3.26 (Shift Gadget)**.** For a graph $G = (V, A)$ and the price range $(0, p_{\max}]$ we construct G_{+x} such that we have the customers shifted from price $\pi \in (0, p_{\max}]$ to $\pi + x \in (x, p_{\max} + x]$.

We just add x to all initial values and then apply Gadget 3.25 and restrict the graph to $(x, p_{\max} + x]$.

For negative x we do not need the restriction, yet lose some parts since negative prices are always excluded.

Gadget 3.27 (Multiply Gadget)**.** For a graph $G = (V, A)$ we multiply all initial values and influences by some integer $x \in \mathbb{N}$ and call this graph $G_{\cdot x}$.

Note that we somehow already Gadgets 3.25 and 3.26 in the construction of Gadgets leading to Theorem 3.9. We constrained the prices used to $\left\{ 2^{\tilde{n}}, \ldots, 2^{\tilde{n}+1} - 1 \right\}$ by adding node z_-. The shift from $\left\{ 1, \ldots, 2^{\tilde{n}} \right\}$ was already implied in the construction of G^x. We required this to decrease the ratio from $\dfrac{2^{\tilde{n}}}{1}$ to $\dfrac{2^{\tilde{n}+1} - 1}{2^{\tilde{n}}} < 2$.

3.7.1 Polynomial equivalence

In the development of Theorem 3.9 we focused all effort on this specific instance. For further use we provide polynomial equivalence for problems PPAI, PPAI-v, PPAI-S, PPAI-i. The reductions provided are for arbitrary graphs. They can still function for other graph classes. In preparation of Section 3.8, we already discuss trees herein.

Lemma 3.28. *PPAI-v $\propto$ PPAI.*

Proof. Given an instance of PPAI-v, i.e., a graph $G = (V, A)$, a fixed node v and a bound $\hat{\pi}$, we construct $G' := G_{|(\hat{\pi}-1, p_{\max}]+p_{\max}}$ by restriction and shift. The node v actually functions as the collecting node, upon which we add the revenue nodes $z_1, \ldots, z_M$ with $M := |G'|$.

We set $\hat{R} \coloneqq M \cdot (\hat{\pi} + p_{\max})$ and analogously to the proof of Theorem 3.9 achieve the desired revenue if and only if node v buys. $\square$

For a tree G we can conserve this property by adding seperate α_+, α_- for every node. The next two reductions do not require adding any cycles and therefore uphold the tree property.

Lemma 3.29. *PPAI-v $\propto$ PPAI-i.*

Proof. Analogously to Lemma 3.28 we add revenue nodes $z_1, \dots, z_M$ to $G = (V, A)$ with $M = |V|$. So we can achieve $i \coloneqq M + 1$ purchases if and only if v buys. $\square$

Lemma 3.30. *PPAI-v $\propto$ PPAI-S.*

Proof. Simply set $S \coloneqq \{v\}$. $\square$

Now assuming on the other hand, that we have some algorithm ALG-v to solve PPAI-v and we want to solve the other problems.

Lemma 3.31. *PPAI-S $\propto$ PPAI-v.*

Proof. Given a graph $G = (V, A)$ and subset $S \subseteq V$, we add one node v with $p(v) = (1 - |S|) \cdot p_{\max}$ and influences $w(u, v) = p_{\max}$ for all $u \in S$. The values imply, as in Gadget 3.8, that v buys if and only if it is influenced by all nodes in S. $\square$

Lemma 3.32. *PPAI-i $\propto$ PPAI-v.*

Proof. Analogously to Lemma 3.31 we add node v with $p(v) = (1 - i) \cdot p_{\max}$ that is influenced by all $u \in V$. $\square$

To conserve the tree property in the last two proofs, we can copy the graph G either $|S|$ or n times. Every copy is assigned to one specific node in S or V respectively. Adding the arcs to v can thereby not induce a graph.

Lemma 3.33. *PPAI $\propto$ PPAI-i.*

Proof. To solve an instance $\mathcal{I}$ of PPAI with requested revenue $\hat{R}$, we can make n calls to any algorithm for PPAI-i. If an algorithm finds a price $\pi \geq \dfrac{\hat{R}}{i} =: \hat{\pi}$ with at least i buyers, we have also found a price for the original instance $\mathcal{I}$. And if no call for $i = 1, \dots, n$ finds an appropriate price, we can conclude that $\mathcal{I}$ is unsatisfied. $\square$

Corollary 3.34. *Altogether we conclude the polynomial equivalence of PPAI, PPAI-i, PPAI-v, and PPAI-S.*

3.8 Trees

In Section 3.6 we already derived some results on different graph classes. So far unmentioned, this already implies complexity results for acyclic graphs, cycles and paths.

Remark 3.28. The graph constructed for the proof of Theorem 3.9 is acyclic. Hence, PPAI is NP-complete in particular for acyclic graphs.

For cycle or path graphs we have $\Delta^-(G) \leq 2$ and therefore $\text{frag}(n) \leq 4n$ by Lemma 3.22.

Kind of in between these lies the class of trees, graphs that do not even contain an undirected cycle. In this section we review different approaches to solve the complexity of PPAI on trees and finally conjecture that the amount of fragments is polynomially bounded.

As previously examined in the last subsection, we can also establish an equivalence of PPAI variations on the class of trees.

With these arguments, we focus on PPAI-v in the remainder of this section. More precisely PPAI-v for in-trees, since we can require a directed path from the other nodes to v.

Definition 3.29. We define $T_v = (V, A)$ as in-tree of v, if the nodes $u \in V \setminus \{v\}$ have $g^+(u) = 1$, the root v has $g^+(v) = 0$ and the graph is weakly connected.

Remark 3.30. If we relax the definition such that the graph can contain some pair $\{(u, v), (v, u)\} \subseteq A$, whether the corresponding weights are symmetric or not, the problem PPAI-v remains the problem on the in-tree of v.

Proof. Given such a graph G and a node $u \in V \setminus \{v\}$, we know that there is a unique directed path from u to v, $P = (u, u_2, \ldots, u_k, v)$. If u where to be influenced by u_2, this implies that u_2 already bought and is therefore unaffected by any future purchase of u. We can basically eliminate the arc (u_2, u) when considering PPAI-v and thus drop to the in-tree T_v. □

3.8.1 Constructing a Variable Gadget

In the proof of Theorem 3.9 we used Gadget 3.6, which contains the topologically sorted $K_{\tilde{n}}$ as a minor.

If we could (polynomially) transform the variable gadget G^x to an in-tree with the same buying behavior of some variable node x_i or $\overline{x}_i$ in the original, we could use a separate copy of this transformed gadget for every literal ($3\tilde{m}$ many), influence the clause nodes $y_1, \ldots, y_{\tilde{m}}$ and collect their behavior in z_+.

Note that we can in either case introduce a separate chain of time nodes for every node and the size only increases by a factor n.

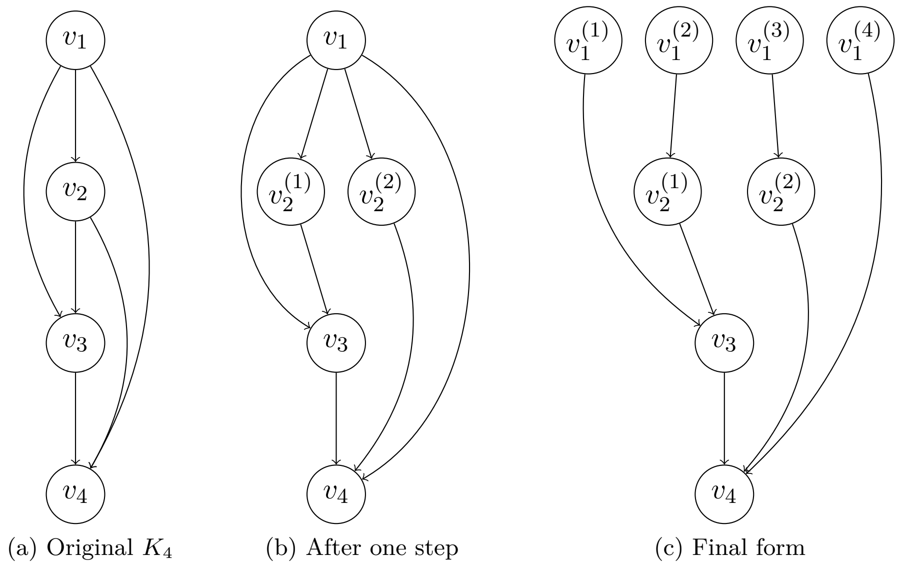

Figure 3.6: Split Gadget applied to K_4

We formally define the required nodes for a variable gadget.

Definition 3.31. Let $g(\tilde{n})$ be the minimum amount of nodes for some $x_{\tilde{n}}$ *generating in-tree* T_v, i.e., for price set $\Pi = \{\pi_{\min}, \dots, \pi_{\max}\}$ of size $|\Pi| = 2^{\tilde{n}}$ and any price $\pi \in \Pi \setminus \{\pi_{\max}\}$ either $v \in C(\pi)$ or $v \in C(\pi + 1)$ holds — v buys for all even or all odd prices.

Lemma 3.35. *To construct a graph for a reduction to instance $\mathcal{J}$ of 3SAT, we need at most $3\tilde{m} \cdot g(\tilde{n}) + \tilde{m} + 1$ nodes. So if $g(\tilde{n}) = \text{poly}(\tilde{n})$ we can follow PPAI-v is NP-complete on trees.*

Gadget 3.32 (Split-Copy Gadget)**.** Given an acyclic graph G and a topological sorting $v_1, \dots, v_n$, we transform the graph to an in-tree T_{v_n} as follows:

We proceed for $v_i = v_{n-2}, \dots, v_1$ and split the node to $g^+(v_i)$ nodes $v_i^{(1)}, \dots, v_i^{(g^+(v))}$. All nodes are influenced by the complete set $\mathcal{N}^-(v_i)$ and only influence one successor of $\mathcal{N}^+(v)$ each.

Since every copy gets the same influence as the original node before, we have the same buying behavior for any price π.

Proposition 3.36. *Applying Gadget 3.32 to the complete oriented graph K_n we get a tree of size 2^{n-1}.*

Proof. The out-degree for v_i in K_n is $g^+(v_i) = n - i$. We start with the split of node v_{n-2}. In this step we get one additional node and increase the out-degree of nodes $v_1, \dots, v_{n-3}$

by 1. We proof by induction that in the round of the splitting node v_i has out-degree $g^+(v_i) = 2^{n-i-1}$. This implies that we split the graph to $1+1+2+4+\dots+2^{n-2} = 2^{n-1}$ nodes in total.

For node v_i we know that previously node v_{i+1} was split. In this step v_{i+1} had an out-degree of 2^{n-i-2}. Since we have started with a complete graph, node v_i is influencing the exact same nodes, with the addition of v_{i+1} itself. By splitting v_{i+1} we increase $g^+(v_i)$ from $2^{n-i-2} + 1$ to $2^{n-i-2} + 2^{n-i-2} = 2^{n-i-1}$ as claimed. □

Corollary 3.37. *Applying Gadget 3.32 to the variable gadget G^x yields an exponential amount of nodes.*

Thus, unfortunately we can not use this gadget to show NP*-completeness for* PPAI *on trees.*

Remark 3.33. By Lemma 3.22 and Theorem 3.15 we already know that a $g(\tilde{n})$-generating tree requires both, a high degree and a high depth.

This raises two candidates. The full Δ-nary tree of height h (where $n = \frac{\Delta^{h+1} - 1}{\Delta - 1}$) and alternatively the path of length h with Δ neighbors attached to each node in the path (where $n = \Delta h + 1$). We call this latter graph *claw-path* graph.

We do a short analysis on both graphs by the *Tree-Pluck* method.

Proposition 3.38 (Tree-Pluck method)**.** *Given a in-tree T_v and the initial values $p(v_1) \leq \dots \leq p(v_n)$, we can split the problem into n parts.*

For any fragment $(p(v_i), p(v_{i+1})]$ we have a specific set of customers in the first round. In order to determine all fragment endpoints at which the root v changes its buying behavior, we can "pluck the tree" by removing nodes. In particular, we remove leaves until in the v-rooted tree T_v^i exactly the leaves are buyers in the first round.

Restricted to this fragment we can just add all the influences from the leaves, remove them altogether and continue for the next round.

Proof. Consider for some fragment any node $u \in B_1$. In tree T_v there is a unique path from u to v. If some other node on this path already bought in the first round, the further behavior of u does not change anything for the root v. So it can be neglected.

If on the other hand some $u \notin B_1$ spawns a subtree $T_u \cap B_1 = \emptyset$, it can not be influenced at all. In total we can remove the nodes as claimed. □

For the claw-path graph we get a path after one plucking. So restricted to a 1-fragment we know that the remaining nodes can hit at most twice. In total we have up to $n \cdot 2h < n^2$ fragments. The graph is not suitable to generate an exponential amount of fragments.

Whereas the claw-path graph is dropping to a path after one plucking, the Δ-nary tree is self-similar. Note that any constant amount of successive tree plucks has to leave a "complex" graph in order to be a potential $g(\tilde{n})$ generating tree.

The full Δ-nary tree has less than Δ^h non-leaves, each with Δ predecessors. In total we have at most $\Delta^h + \Delta^h \cdot 2^\Delta < n \cdot 2^\Delta$ hits, therefore fragments. By plucking the tree down to v in up to h iterations, we conclude that the graph does not get more than n^h fragments.

Balancing both bounds, the maximum is achieved for approximately $h \approx \sqrt{\dfrac{\Delta}{\log \Delta}}$. Further studies on the tree remained inconclusive.

We could not find an $x_{\tilde{n}}$ generating in-tree T_v of polynomial size. Whereas a non-existence thereof does not imply an upper bound for the possible amount of fragments, it is the natural alternative to study.

3.8.2 Upper Bound on Fragments

Now given an amount of nodes in in-tree T_v we want to bound the number of fragments it can generate.

Definition 3.34. Analogously to our previous definition of fragments we set $\mathrm{frag}_v(G)$ to be the partition of prices only considering changing buying behavior of v, i.e., the integer interval $(x, y]$ is inclusion-wise maximum for node v buying in any specific round t.

Remark 3.35. Note that $|\mathrm{frag}(G)| \leq \sum_{v \in V} |\mathrm{frag}_v(G)|$.

We define $f^*(n) := \max_{T_v : |T_v| = n} |\mathrm{frag}_v(T_v)|$ as the actual maximum achievable. For the first few values we can determine $f^*(n)$ by checking all possible in-trees T_v, e.g. $f^*(1) = 1$, $f^*(2) = 2$, and $f^*(3) = 4$. Instead of actually computing f^* we develop an upper bound for it.

Lemma 3.39. *We define the upper bound $f(n) \geq f^*(n)$ recursively as,*

$$f(n) := \max_{a \in \mathcal{A}_{n-1}} \left(\sum_{i=1}^{|a|} f(a_i) + \min_{i=1,\dots,|a|} \left\{ 2^{|a|}, 2^{|a|-i} \cdot \sum_{j=1}^{i} f(a_j) \right\} \right),$$

where

$$\mathcal{A}_{n-1} = \left\{ a = (a_1, \dots, a_{|a|}) : a_1 + \dots + a_{|a|} = n - 1, a \in \mathbb{N}^k, a_1 \leq \dots \leq a_{|a|} \right\}$$

is the set of sorted integer partitions and $f(1) := 1$ is set as starting value.

Proof. Note that, at least for $n > 2$, the growth satisfies $f(n) - f(n-1) \geq 2$ as we can just use partition $a = (n-1)$. Therefore, $f(n)$ is strictly increasing.

Given any in-tree T_v of size n, every predecessor of v forms a subtree of its own. Let $\left\{ u_1, \dots, u_{g^-(v)} \right\} = \mathcal{N}^-(v)$ and T_{u_i} be the corresponding subtrees. For the computation we only use the size of these subtrees, so for every tree T_v there is some $a \in \mathcal{A}_{n-1}$ with $|T_{u_i}| = a_i$. This implies that maximizing over the partitions actually covers all possible in-trees of size n.

Now given one specific tree T_v, as well as the neighbors $\mathcal{N}^-(v)$ and the corresponding partition $a = (a_1, \dots, a_{g^-(v)})$. Before v buys anywhere, we have already the fragments without it, $\text{frag}(G \setminus \{v\})$. Given the correctly computed lower function values $f(a_1), \dots, f(a_{g^-(v)})$ this amounts to $\sum_{i=1}^{g^-(v)} f(a_i)$.

Considering the fragments without v, we know that v itself can land at most one hit in each of these. Hence, the function f can at most double. However this does not take the in-degree into account, as we can only have $2^{g^-(v)}$ different influenced values. This limits the amount of hits as well.

In the case of trees we can go even further and mix both bounds. Given any subset $S \subseteq \mathcal{N}^-(v)$ we accumulate their fragments as before for $\mathcal{N}^-(v)$ itself. On these fragments the possible sets of customers $\mathcal{C}$ are guaranteed to satisfy $C \cap \mathcal{N}^-(v) = C' \cap \mathcal{N}^-(v)$ for all $C, C' \in \mathcal{C}$. So the sets can only vary in $\mathcal{N}^-(v) \setminus S$ and we have $|\mathcal{C}| \leq 2^{|\mathcal{N}^-(v) \setminus S|}$.

Translating to a partition $a = (a_1, \dots, a_{g^-(v)})$, we get the nonempty $S \subseteq \{1, \dots, g^-(v)\}$. This set has at most $\sum_{i \in S} f(a_i)$ accumulated fragments and we can generate at most $2^{g^-(v)-|S|}$ different influenced values on each of the fragments. In total we have

$$2^{g^-(v)-|S|} \cdot \sum_{i \in S} f(a_i)$$

as upper bound on hits for the partition a and subset S.

To find a subset S that achieves the minimum, we can consider only sets of type $S_i = \{i, \dots, g^-(v)\}$. Any other set would have some $i \in S$ with $i+1 \notin S$. Switching those would not increase the upper bound, since a is sorted and f is increasing, hence, we can exclude such S.

All in all we have derived the aforementioned definition of $f(n)$ as an upper bound on $\text{frag}_v(T_v)$. $\square$

Observation 3.36. We implemented the recursion of Lemma 3.39 in Python and computed the first hundred values by an exhaustive search of all partitions. The computed values are plotted in Figure 3.7.

For the first 11 Fibonacci numbers $1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89$ we noticed that the computed value doubled from one to the other.

To be more precise, for $F_k = F_{k-1} + F_{k-2} = \frac{\varphi^k - (-\frac{1}{\varphi})^k}{\sqrt{5}} \approx \frac{\varphi^k}{\sqrt{5}}$ [1], we observed $f(F_k) = 2^{k-2}$. Even more, a partition that achieved the maximum was $a = (F_1, \dots, F_{k-2})$ in every case. For these partitions the actual minimum satisfied

$$\min_{i=1,\dots,k-2} \left\{ 2^{k-2-i} \cdot \sum_{j=1}^{i} f(F_j) \right\} = \min_{i=1,\dots,k-2} \left\{ 2^{k-2-i} \cdot 2^{i-1} \right\} = 2^{k-3}.$$

The corresponding trees are generated as in Figure 3.8. Note that the trees coincide with *Trees of minimum possible size for a given rank in a Fibonacci heap* as seen in [FT87].

So, assuming this phenomenon continues, the growth of f is exponential in the Fibonacci numbers.

For the approximate value we set $\tilde{f}(n) = \tilde{f}\left(\frac{\varphi^x}{\sqrt{5}}\right) = 2^{x-2}$. So $x = \log_\varphi(\sqrt{5} \cdot n)$ and we solve

$$\tilde{f}(n) = \frac{\sqrt{5}^{1/\operatorname{ld}\varphi}}{4} \cdot n^{1/\operatorname{ld}\varphi} \approx 0.797 \cdot n^{1.440}.$$

In Figure 3.7 we can observe that the computed values in between the Fibonacci numbers are bounded by this function $\tilde{f}$.

Conjecture 3.40. *We conjecture that the amount of fragments in a tree of size n is* $\operatorname{frag}(n) = \mathcal{O}\left(n \cdot n^{1/\operatorname{ld}\varphi}\right)$. *Therefore,* $\mathsf{PPAI}^{\mathrm{opt}}$ *on trees would be solvable in polynomial time with* Frag.

In addition to this conjecture, it is still an open question what the impact of bounded treewidth is. Furthermore, in Remark 3.30, we already relaxed the tree to cactus graphs with elementary cycles of length up to 2.

3.9 Discrete Dynamical System and Integer Program

As an alternative formulation for our instances of PPAI, we present it as a Discrete/Graph Dynamical System[MR08, KKM⁺11]. Note that in contrast to the usual instances of Discrete Dynamical Systems and Cellular Automata, we do have a definite round limit, as we do not allow a node to change its state, i.e., return the product.

[1]The last identity is Binet's Fibonacci Number formula, $\phi = \frac{\sqrt{5}+1}{2}$ the Golden Ratio

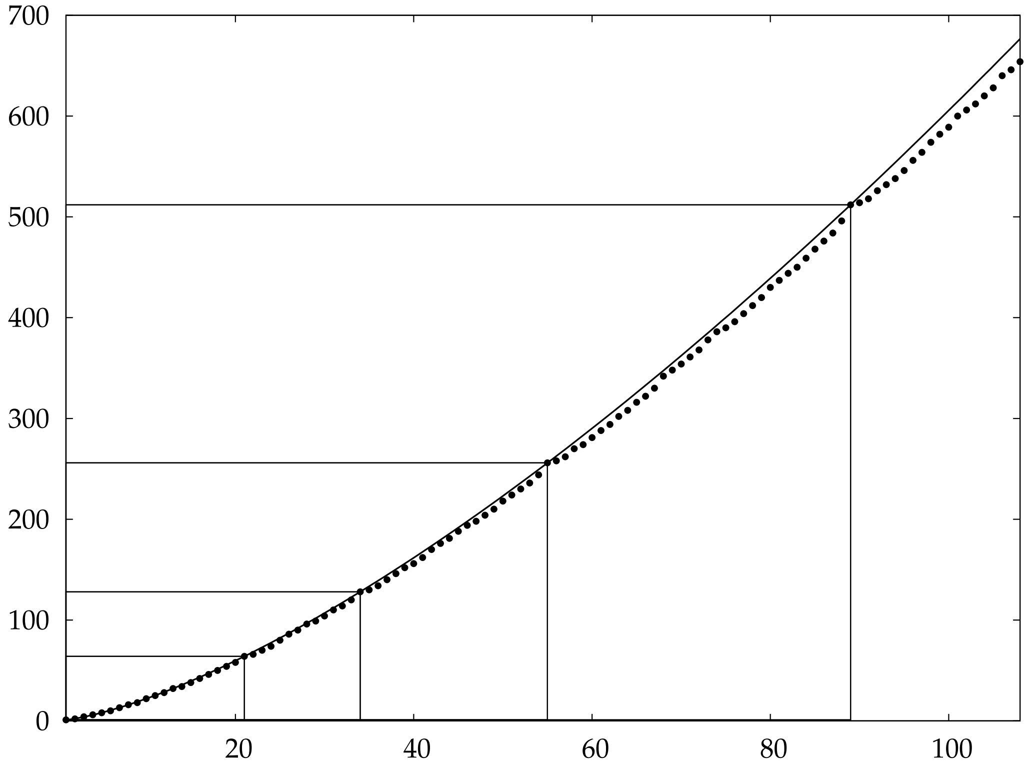

Figure 3.7: Fragment bound for trees: Computed values and the conjectured bound

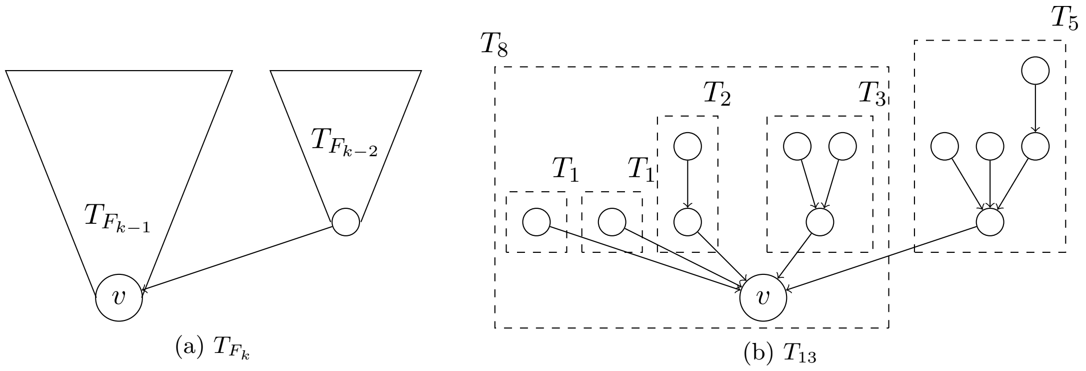

Figure 3.8: Tree with the highest possible amount of fragments

Definition 3.37. Let $(x_v)_{t\in\mathbb{N}} \in (\mathbb{F}_2)_{t\in\mathbb{N}}$ be the state sequence, that denotes whether node $v \in V$ has bought the product at some stage or not. The function

$$f_{v,p} : \mathbb{F}_2^{\mathcal{N}^-(v)\cup\{v\}} \longrightarrow \mathbb{F}_2$$

$$f_v\left(x^{\mathcal{N}^-(v)\cup\{v\}}\right) = \begin{cases} 1 & \text{if } x_v = 1 \\ 1 & \text{if } \sum\limits_{u:(u,v)\in A} w(u,v)\cdot x_u \geq p - p_v \\ 0 & \text{else} \end{cases}$$

determines whether v will buy or possess the product given the previous customers.

We set $(x_v)_1 = 1$ if $p_v \geq p$ and to 0 else. For $t > 1$ we set

$$(x_v)_t = f_v(x_{t-1}^{\mathcal{N}^-(v)\cup\{v\}}).$$

As an alternative to the usage of Algorithm Frag we formulate an integer program.

Integer Program 1: PPAI *(Formulation of PPAI)*

$$\max \pi \cdot \sum_{v \in V} \sum_{t=1}^{n} x_{v,t}$$

subject to

$$\begin{aligned}
x_{v,t} &\geq 0 & &\forall v \in V, t = 1, \ldots, n \\
x_{v,t} &\leq 1 & &\forall v \in V, t = 1, \ldots, n \\
\sum_{t=1}^{n} x_{v,t} &\leq 1 & &\forall v \in V \\
\sum_{j<t} x_{v,j} &= y_{v,t} & &\forall v \in V, t = 1, \ldots, n \\
p(v) + \sum_{u \in \mathcal{N}^-(v)} y_{u,t} \cdot w(u,v) &= p_{v,t} & &\forall v \in V, t = 1, \ldots, n \quad (3.7) \\
p_{v,t} - \pi &\geq (x_{v,t} - 1) \cdot M & &\forall v \in V, t = 1, \ldots, n \quad (3.8) \\
p_{v,t} - \pi &\leq (x_{v,t} + y_{v,t}) \cdot M - 1 & &\forall v \in V, t = 1, \ldots, n \quad (3.9)
\end{aligned}$$

The variable $x_{v,t}$ denotes whether $v \in B_t$, whereas $y_{v,t}$ denotes whether $v \in C_{t-1}$. Considering only integer values, the equations imply that every node can buy at most once.

In Equation (1) we compute the influenced value as usual.

If we want v to buy in round t, i.e., set $x_{v,t} = 1$, we obtain $p_{v,t} \geq \pi$ from Equation (1). And if $x_{v,t} = y_{v,t} = 0$, we obtain $p_{v,t} \leq \pi - 1 < \pi$ from Equation (1), i.e., an influenced value that is not high enough to buy.

Observation 3.38. As part of the Bachelor program at the University of Kaiserslautern, the students Arne Herzel and Sebastian Johann generated a series of random instances of PPAI, using the $G(n,p)$-model with uniformly distributed weights.

In their tests, the implementation of Frag provided a faster runtime than *Gurobi* on the integer program. The necessity of using the Big M method did presumably hamper the performance significantly.

Due to the lack of proper weighted social networks, we did not undertake further simulations. Thus the actual comparison of both alternatives is yet unanswered.

4 Delayed Product Pricing

In the standard PPAI model we assumed that every potential buyer is familiar with the product from day one. Typically, the marketing of a new product is the major issue, studied amongst others in [KKT03, BFO10].

For our model of Delayed Product Pricing we assume that we know the date when a node gets to know the product.

In a related fashion we put a delay on the influences, i.e., it may take some time to spread the word of a purchase.

Definition 4.1. For a graph $G = (V, A)$ we define the *delay* $\tau : V \cup A \to \mathbb{N}$. For a node $v \in V$ the value $\tau(v)$ denotes the *release time*, whereas for influences $a \in A$ the *influence delay* is denoted by $\tau(a)$. This means, a node $v \in V$ can not buy before its release time $\tau(v)$ and influences $w(u, v)$ are exerted after $\tau(u, v)$ rounds.

We further set $\tau_V := \max_{v \in V} \tau(v)$ and $\tau_A := \sum_{v \in V} \max_{u \in \mathcal{N}^-(v)} \tau(u, v)$, the maximum release time and total delay, respectively.

The influenced value of v is set to 0 — or any other negative value — before the node is released in round $\tau(v)$.

For a sequence $\emptyset = C_0 \subseteq C_1 \subseteq \cdots \subseteq C_{t-1}$ of customers we set the *relevant* customers for node v as

$$C^v_{t-1} := \left\{ u \in \mathcal{N}^-(v) : u \in C_{t-\tau(u,v)} \right\},$$

and the influenced value to

$$p_{C^v_{t-1}}(v) := \begin{cases} p(v) + \sum_{u \in C^v_{t-1}} w(u, v) & \text{if } \tau(v) \leq t \\ 0 & \text{else .} \end{cases}$$

Node $v \in V \setminus C_{t-1}$ buys in round t for price π if and only if $p_{C^v_{t-1}}(v) \geq \pi$.

The basic problem of this chapter is DelPPAI.

Problem 2: DelPPAI *(Delayed Product Pricing with Additive Influences)*

Instance: *A directed graph $G = (V, A)$, initial values $p(v) \in \mathbb{Z}$ for the nodes $v \in V$, influences $w(u, v) \in \mathbb{Z} \setminus \{0\}$ for the arcs $(u, v) \in A$, release times $\tau(v) \in \mathbb{N}$ for $v \in V$, influence delays $\tau(u, v) \in \mathbb{N}$ for $(u, v) \in A$, and some revenue $\hat{R} \in \mathbb{N}$.*

Question: *Is there a price $\pi \in \mathbb{N}$ such that $R(\pi) \geq \hat{R}$?*

As in Definition 3.1 we set the variants.

Definition 4.2 (Variants of PPAI)**.**

DelPPAI$^\pi$: Compute the revenue for a given price.

DelPPAI$^{\text{opt}}$: The corresponding revenue maximization problem.

DelPPAI-v: Is there a price $\pi \geq \hat{\pi}$ such that $v \in C(\pi)$?

DelPPAI-S: Is there a price $\pi \geq \hat{\pi}$ such that $S \subseteq C(\pi)$?

DelPPAI-i: Is there a price $\pi \geq \hat{\pi}$ such that $|C(\pi)| \geq i$?

The order of this chapter is based on Chapter 3, thus we start with the selling problem DelPPAI$^\pi$.

Lemma 4.1. *The selling stops after at most $T := \tau_V + \tau_A$ rounds.*

Proof. As in Corollary 3.18, we devise the decay of G by the longest path.

We consider graph $G^T := \Big(V \cup \{\alpha\}, A^+ \cup (\{\alpha\} \times V)\Big)$. For the new arcs $(\alpha, v) \in \{\alpha\} \times V$ with arc lengths c we set $c(\alpha, v) = \tau(v)$.

The length of the old arcs $a \in A^+$ with positive influence is set to their delay — $c(a) = \tau(a)$.

Where before node v needed a positive influence in round $t - 1$, since all delays were implicitly set to 1, node v can now be influenced in various earlier rounds. The set

$$\{\, t - \tau(u, v) : u \in \mathcal{N}^-(v), w(u, v) > 0 \,\}$$

contains all these rounds. The selling for node v does only start in round $\tau(v)$, hence, the paths from v are extended by $c(\alpha, v) = \tau(v)$.

Any path in G^T that starts at some node $v \in V$ can be extended by (α, v). In particular, a longest path starts in α just as the "chain" of influences starts after the release of the first node.

Since any path starting at α contains at most one arc to each node $v \in V$, we can assume that it is the longest of all arcs from $\mathcal{N}^-(v)$ to v. This concludes the bound of $\tau_A + \tau_V$. □

Remark 4.3. We can modify the basic Algorithm Sell for T rounds and compute the revenue $R(\pi)$ for a given price $\pi \in (0, p_{\max}]$ by progressing every round $t = 1, \ldots, T$.

Yet we know in advance that certain rounds are not interesting, i.e., no node changed its influenced value. In Algorithm DelSell we skip those rounds.

```
Algorithm 5: DelSell$(G, \pi)$
Data: A graph $G = (V, A)$ with weights $p$ and $w$, delay $\tau$ and a price $\pi$.
Result: The set $C(\pi)$ of customers and the revenue $R(\pi)$.
Initialize: $p'(v) = p(v)$, $C = \emptyset$ , $B = \{\, v \in V : p(v) \geq \pi, \tau(v) = 1 \,\}$, and
      $D = \Big\{ (1 + \tau(u, v), u, v) : u \in B, v \in \mathcal{N}^+(u) \setminus B \Big\}$.
// $D$ denotes the round in which influence is exerted
while $D \neq \emptyset$ do
    // Update influenced values
    Set $t' = \min_{(t,u,v) \in D} t$ and $N = \{\, (t, u, v) \in D : t = t' \,\}$.
    Set $D = D \setminus N$ and $X = \emptyset$.                    // nodes whose valuations change
    for $(t, u, v) \in N$ do
        $p'(v) = p'(v) + w(u, v)$
        $X = X \cup \{v\}$
    // Compute $B$ and update $D$
    $B = \emptyset$
    for $v \in X$ do
        if $p'(v) \geq \pi$ then
            $B = B \cup \{v\}$
            for $u \in \mathcal{N}^+(v) \setminus C$ do
                $D = D \cup \{(t' + \tau(v, u), v, u)\}$
    $C = C \cup B$
return $R(\pi) = \pi\,|C|$ and $C(\pi) = C$
```

Lemma 4.2. *Algorithm DelSell computes the revenue $R(\pi)$ in $\mathcal{O}(m \log m)$ time for a given price π.*

Proof. The set D denotes the influences that are currently "on their way". Round t' is computed as the next time at which influence "arrives". So in between two **while**-iterations the influenced value does not change for any node, hence, no buying occurs. All in all, the algorithm follows the procedure of Sell and the new definition for delayed pricing.

As for the runtime, the set D grows at most m times, since every new element corresponds to a unique arc and every arc is triggered at most once, i.e., by the purchase of the tail. Thus, using Fibonacci heaps again [FT87], we have at most m `insert` and `delete-min` operations to manage D, which needs $\mathcal{O}(m \log m)$ in amortized time. The rest of the algorithm is only simple operations, of which we have $\mathcal{O}(m+n)$. This yields the runtime as claimed. $\square$

Corollary 4.3. *The problem DelPPAI$^\pi$ is in FP. Furthermore, DelPPAI and the variations are in NP or NPO, respectively.*

The definitions of problems PPAI and DelPPAI share most of their details. In the following we show how exactly both are connected to each other.

Corollary 4.4. *By setting delays $\tau = 1$ for the domain of τ, we get a standard instance of PPAI.*

Therefore, it directly follows that DelPPAI is NP-complete. The same holds for the variants thereof.

The variable gadget does not even require time nodes, thus we can even achieve $2^{n/2}$ fragments with n nodes.

We already know that adding delays is as hard as before. In Gadget 4.4 we provide a way to polynomially transform any instance of DelPPAI-S to an instance of PPAI-S, thereby establishing equivalence.

Gadget 4.4. Given a graph $G = (V, A)$ with delay τ, we define G^τ as follows.

Every node $v \in V$ with $\tau(v) > 1$ is substituted by a chain $v_1, \ldots, v_{\tau(v)}$, where the last node is identified with v. We set for some sufficiently large $M := p_{\max} + \sum_{a \in A\,:\,w(a)>0} w(a)$ the initial values $p(v_1) = p_{\max}$, $p(v_i) = 0$ for $i = 2, \ldots, \tau(v) - 1$, and $p(v_{\tau(v)}) = p(v) - M$. The elements in the chain are consecutively influencing their successor with $w(v_i, v_{i+1}) = M$ for $i = 1, \ldots, \tau(v) - 1$.

We substitute the arcs similarly. For any arc $a = (u, v) \in A$ with $\tau(a) > 1$ we introduce nodes $a_1, \ldots, a_{\tau(v)-1}$ with no initial value. They are chained in between $u = a_0, a_1, \ldots, a_{\tau(v)-1}, a_{\tau(v)} = v$ where the influences are $w(a_i, a_{i+1}) = M$ for $i = 0, \ldots, \tau(v) - 2$. The last influence $w(a_{\tau(v)-1}, a_{\tau(v)}) = w(a)$ is of the original weight.

The total amount of nodes and arcs are

$$|V^\tau| = \sum_{v \in V} \tau(v) + \sum_{a \in A} (\tau(a) - 1) \text{ and}$$
$$|A^\tau| = \sum_{v \in V} (\tau(v) - 1) + \sum_{a \in A} \tau(a).$$

Note that we have already used the delay in the construction of the variable graph G^x in Gadget 3.6. Whereas we used only a single delay chain for G^x, we introduce a separate release time chain for every node here, thereby not introducing any directed or undirected cycles.

Proposition 4.5. *Given an instance $\mathcal{I}$ of* DelPPAI-S*, the Gadget 4.4 provides an instance $\mathcal{J}$ of* PPAI-S *with equal outcome.*

The reduction is polynomial if $\tau = \mathrm{poly}(n)$.

Proof. The nodes in the chains do exactly buy one round after their predecessor and the head of the chain always gets the influence as before. By the setting of M we make sure that the unreleased nodes are not coincidentally released by the influences of other nodes.

We assume that the graph G^τ is encoded as usual in an adjacency list. The delay values $\tau(v)$ and $\tau(a)$ themselves are of logarithmic size, whereas we now require a linear amount of nodes in G^τ. Thus we require $\tau = \mathrm{poly}(n)$ for a polynomial reduction. $\square$

Remark 4.5. By the extension from G to G^τ we messed with the actual revenue. In order to compute the optimum price we still can reuse the previous algorithms.

The fragments for DelPPAI can be computed by Frag on G^τ. Instead of the revenue computation in Frag we run DelSell on G for every endpoint to find the maximum revenue.

Proposition 4.6. *Given the sorted release times $\tau(v_1) \leq \cdots \leq \tau(v_n)$ we can iteratively compress it to $\tau'(v_1) \leq \cdots \leq \tau'(v_n) \leq n\tau_A$.*

Proof. Without loss of generality $\tau'(v_1) = 1$. So after round $1 + \tau(v_1, v_2)$ the influenced value of v_2 remains unchanged until release. Hence, we can set

$$\tau'(v_2) = \min\left\{ 1 + \tau(v_1, v_2), \tau(v_2) \right\}.$$

Assuming we compressed the release times up to $\tau'(v_{i-1})$ already, the longest path containing only released nodes is bounded by

$$\sum_{i=1,\dots,i-1} \max_{u \in \mathcal{N}^+(v_i)} \tau(v_i, u).$$

Therefore, the difference between $\tau'(v_{i-1})$ and $\tau'(v_i)$ can be set to at most this value. In sum it follows that we can construct τ' such that $\tau'(v_n) \leq n\tau_A$.

Thereby, it suffices to have $\tau_A = \mathrm{poly}(n)$ for the polynomial reduction. $\square$

In the upcoming chapters we will not consider combinations of the respective problems and delays. However, the idea of delays is used frequently by adding time nodes.

5 Dynamic Product Pricing

In the previous chapters we had one fixed price for our product. Thus we could not take into account that the influenced value the potential buyers hold for the product may be changing over time. In this chapter we are adjusting the price to increase the revenue even more.

The dynamic pricing follows two core properties:

- The potential buyers do not act strategically, i.e., they buy as soon as their influenced value exceeds the current price.
- The product is still everlasting and unreturnable.

Definition 5.1. For a given time horizon $\mathbb{N}_T := \{1, \ldots, T\}$ (or just T for short), a sequence $(\pi_i)_{i \in \mathbb{N}_T} \in \mathbb{N}^T$ of prices is a *trend of prices* for our product. If the time horizon or the explicit sequence is clear from the context we will use $(\pi_i)_i$ or π instead.

For a given trend of prices $(\pi_i)_{i \in \mathbb{N}_T}$ and some round t a node $v \in V \setminus C_{t-1}(\pi)$ buys, if $p_{C_{t-1}(\pi)}(v) \geq \pi_t$. By $R_t(\pi) := \pi_t \, |B_t(\pi)|$, $R_{\leq t}(\pi) := \sum_{i \leq t} R_i(\pi)$ and $R(\pi) := R_{\leq T}(\pi)$ we denote the revenue generated in some round, up to some round, and the total revenue for the trend of prices $(\pi_i)_{i \in \mathbb{N}_T}$, respectively.

With these definitions we define the main problem of the chapter:

Problem 3: DynPPAI *(Dynamic Product Pricing with Additive Influences)*

Instance: *A directed graph $G = (V, A)$, initial values $p(v) \in \mathbb{Z}$ for the nodes $v \in V$, influences $w(u, v) \in \mathbb{Z} \setminus \{0\}$ for the arcs $(u, v) \in A$, and some revenue $\hat{R} \in \mathbb{N}$.*
Question: *Is there a trend of prices $\pi \in \mathbb{N}^T$ such that $R(\pi) \geq \hat{R}$?*

In the case of dynamic pricing, the former restriction to find a price $\pi \geq \hat{\pi}$ is not useful anymore. We substitute this with $\pi \in \hat{\Pi}$ for some set $\hat{\Pi}$ of trends.

The variants of DynPPAI are defined analogously to Definition 3.1

Definition 5.2.

DynPPAI$^{\pi}$: Compute the revenue for a given price.

DynPPAI$^{\text{opt}}$: The corresponding revenue maximization problem.

DynPPAI-v: Is there a price $\pi \in \hat{\Pi}$ such that $v \in C(\pi)$?

DynPPAI-S: Is there a price $\pi \in \hat{\Pi}$ such that $S \subseteq C(\pi)$?

DynPPAI-i: Is there a price $\pi \in \hat{\Pi}$ such that $|C(\pi)| \geq i$?

Notice that, in general, prices may change as fast as the influences are exerted — from one round to another. Therefore, we speak of *fast pricing*, our standard model in this chapter.

We can alternatively wait in-between price changes to allow the influences to be spread.

Definition 5.3. Under the *slow pricing* paradigm, the trend of prices states the sequence of prices, but not the actual rounds in which they are set. That means we sell for the same price π_1 as long as $B_t(\pi_1) \neq \emptyset$, hence, until exactly the static customers $C(\pi_1)$ acquired the product. The selling process is continued recursively with π_2 on $V \setminus C(\pi_1)$ with preset $p(v) \coloneqq p_{C(\pi_1)}(v)$.

Remark 5.4. In the other chapters we could restrict $\pi \in (0, p_{\max}]$. Since we can change the prices in any round this does not apply here for the prices $\pi_1, \ldots, \pi_T$. After the first selling for some price in $(0, p_{\max}]$ the influenced values can exceed $p_{\max}$, hence, we can raise the price and still get buyers.

In particular we can have $p(v) = 1 = p_{\max}$ for $v \in V$ and $p(u) = 0$ for the other nodes $u \in V \setminus \{v\}$. The revenue for PPAI$^{\text{opt}}$ on this graph is at most n.

In the dynamic setting we can profit from the influences. Let $w(v, u) = M$ for all $u \in V \setminus \{v\}$ be the influences from v. After $\pi_1 = 1$ we can sell the product for $\pi_2 = M$ in the second round. Hence, the maximum revenue for DynPPAI$^{\text{opt}}$ is $R^* = 1 + (n-1) \cdot M$.

5.1 Computing the Revenue

As Algorithm Sell computed the revenue for PPAI$^{\pi}$, we define Algorithm DynSell to solve DynPPAI$^{\pi}$.

Lemma 5.1. *Algorithm DynSell computes the revenue for a given trend of prices* $(\pi_i)_{i \in \mathbb{N}_T}$ *in* $\mathcal{O}(T + m \log n)$ *time.*

Proof. Computing B and updating the influenced values $p'(v)$ is executed exactly by Definition 5.1.

In q we store the current highest influenced value. If $q < \pi_i$ we instantly know that this round does not have a buyer and skip it. Otherwise we know that there is at least one buyer, i.e., the outermost **if**-condition is `true` at most n times. As long as the product did not sell to all nodes in V, we need to compare q with π_i. This takes $\mathcal{O}(T)$ time.

We store the nodes $v \in V \setminus C$ in a max-heap. The initialization with values $p'(v) = p(v)$ takes n `insert`-operations. Computing B requires up to $\mathcal{O}(n)$ `delete-max`-operations, whereas computing q requires up to n `find-max`-operations. Since our influences are not sign constrained, the update for some arc $a \in A$ can be either `decrease-key` or `increase-key`.

In total these actions take $\mathcal{O}(T + m \log n)$ time. □

Algorithm 6: DynSell$(G, (\pi))$

```
Data: A graph $G = (V, A)$ with weights $p$ and $w$, and a price trend $(\pi_i)_{i \in \mathbb{N}_T}$.
Result: The set $C(\pi)$ of customers and the revenue $R(\pi)$.
Initialize: $p'(v) = p(v)$ for $v \in V$, $C = \emptyset$, and $q = p_{\max}$.
                                        // $q$ is the current highest influenced value
for $i = 1, \ldots, T$ do
    if $q \geq \pi_i$ then
                                                                        // Compute $B$
        for $v \in V \setminus C$ do
            if $p'(v) \geq \pi_i$ then $B = B \cup \{v\}$
        $R = R + \pi_i |B|$
        $C = C \cup B$
                                                        // Update influenced values
        for $u \in B$ do
            for $v \in \mathcal{N}^+(u) \setminus C$ do
                $p'(v) = p'(v) + w(u, v)$
        $B = \emptyset$
                                                                        // Compute $q$
        if $V \neq C$ then $q = \max_{v \in V \setminus C} p'(v)$
        else break
return $R(\pi) = R$ and $C(\pi) = C$
```

Note that the input includes T prices and not only the value T, i.e., the encoding size is at least linear in T, whereas it is only logarithmic in the time horizon T in Chapters 4 and 6.

Corollary 5.2. *Algorithm DynSell runs in polynomial time, thus DynPPAI$^\pi$ is in FP and the variations of DynPPAI are in NP or NPO, respectively.*

The complexity of the selling problem is already established for trends of any size. Yet, we can potentially skip lots of those prices.

Proposition 5.3. *Given an instance of DynPPAI, every trend of prices $(\pi_i)_{i \in \mathbb{N}_T}$ can be contracted to an equivalent trend $(\pi'_i)_{i \in \mathbb{N}_{T'}}$ with $T' \leq \min\{T, n\}$, in the sense that $C(\pi) = C(\pi')$, and $R(\pi) = R(\pi')$.*

Proof. As we have at most $|C(\pi)| \leq n$ rounds with a buyer, we can skip every round and its price without a buyer. $\square$

For the sake of simplicity we assume that the given trends in the remainder of this chapter are already *contracted* to n prices. Without loss of generality, the prices in rounds $i = T' + 1, \dots, n$ can either be set to 0 or some unaffordable price M. This guarantees in both cases that there is no additional revenue and checking these $n - T'$ rounds does take at most $\mathcal{O}(n)$ time.

Remark 5.5. We can solve slow pricing, by changing Algorithm DynSell such that every price π_i is considered again in the next iteration as long as $q \geq \pi_i$.

5.2 b-Fragments

In Subsection 3.3.1 we concluded that the optimal price is the endpoint of some fragment, since the prices in the interior of fragments are inferior. In a similar fashion we can improve the prices π_i for dynamic pricing.

Proposition 5.4 (Improved trends)**.** *Given a trend of prices π, the customers $C_{t-1}(\pi)$, and the current buyers $B_t(\pi)$, we can potentially increase the revenue by setting*

$$\pi'_t \coloneqq \min_{v \in B_t(\pi)} p_{C_{t-1}(\pi)}(v) \geq \pi_t.$$

Proof. The price guarantees that $B_t(\pi_t) = B_t(\pi'_t)$ and it is the highest with this property, since at least one buyer would not buy for $\pi'_t + 1$. $\square$

Hence, the essential information of the trend is the amount of buyers in any given round.

Definition 5.6. For a graph G and trend of prices $(\pi_i)_{i \in \mathbb{N}_n}$ we define the *trend of sales* by $(b_i)_{i \in \mathbb{N}_n}$ such that $b_i = |B_i(\pi)|$.

The *improved* trend of prices $(\pi^b_i)_{i \in \mathbb{N}_n}$ is constructed according to Proposition 5.4

$$\pi^b_t = \max \{ \pi'_t : |B_t(\pi'_t)| = b_t \} .$$

The overall revenue is $R(b) \coloneqq R(\pi^b) \coloneqq \sum_{i=1}^{n} b_i \cdot \pi_i$.

Remark 5.7. With the contraction in Proposition 5.3 we can conclude that there is some round $1 \leq t \leq n$, such that $b_i > 0$ for rounds $i = 1, \dots, t$ and $b_i = 0$ for the final rounds $i = t + 1, \dots, n$. Furthermore, $\sum_{i=1}^{n} b_i \leq n$.

There may be choices of b for which there is no trend of prices inducing exactly b. For example, if there are negative or coinciding influenced values. In this case, a specific amount b_i might be unattainable.

In the style of fragments, we define b-fragments and optimize over those with Algorithm bFrag.

Definition 5.8. The set $\mathrm{bfrag}(G)$ denotes the set of all possible trend of sales b of G. By $\mathrm{bfrag}_t(G)$ we denote the trends with exactly t non-zero entries. A trend of sales b' is an *extension* of $b \in \mathrm{bfrag}_t(G)$ if the first t values of both trends coincide.

Assuming both — trend of prices and trend of sales — are known, we adress them interchangeably simply as trend.

Note that in contrast to the temporary fragments in PPAI the temporary trends of sales are also included in $\mathrm{bfrag}(G) = \bigcup_{i=1,\dots,n} \mathrm{bfrag}_i(G)$.

Remark 5.9. Given a trend of sales b, we can run a modified DynSell in $\mathcal{O}(m \log n)$ time. It is even easier, since the trend of sales does not require a value q and instead of computing B we `delete-max` exactly $b_1 + \cdots + b_n$ times.

Corollary 5.5. *Given an instance of DynPPAI*$^{\mathrm{opt}}$ *and its b-fragments* $\mathrm{bfrag}(G)$ *we can compute the optimum solution in* $\mathcal{O}(|\mathrm{bfrag}(G)| \cdot m \log n)$ *time.*

As in Section 3.3, in particular Algorithm Frag, we define recursive Algorithm bFrag that proceeds on $\mathrm{bfrag}(G)$ by depth-first technique.

```
Algorithm 7: bFrag($G, b, C, R$)
Data: A graph $G = (V, A)$ with weights $p$ and $w$, some $b \in \mathrm{bfrag}_{t-1}(G)$, previous
      customers $C$, and revenue $R$.
Result: The optimal overall revenue $R^*$ of any extension of $b$.
if $C = V$ then return $R^* = R$
                                                        // Compute $b_t$ candidates
Compute $p_C(v)$ for all $v \in V \setminus C$.
Determine the trend triples $(b_t^{(1)}, \pi_t^{(1)}, B_t^{(1)}), \dots, (b_t^{(x)}, \pi_t^{(x)}, B_t^{(x)})$.
                                                        // Initiate recursive calls
for $i = 1, \dots, x$ do
    if $R < R^{(i)} := bFrag(G, (b_1, \dots, b_{t-1}, b_t^{(i)}), C \cup B_t^{(i)}, R + b_t^{(i)} \cdot \pi_t^{(i)})$ then
        $R = R^{(i)}$
return $R^* = R$
```

Theorem 5.6. *Algorithm bFrag*$(G, (), \emptyset, 0)$ *correctly computes the optimum revenue in* $\mathcal{O}(|\mathrm{bfrag}(G)| \cdot (m + n \log n))$ *time.*

Proof. Correctness is clear, since we iterate over all possible b-fragments.

A single call without the recursive sub-calls, takes $\mathcal{O}(m + n \log n)$ time to compute the influenced values and determine the possible choices. $\square$

So if the amount of b-fragments is polynomial we can find the optimum revenue in polynomial time.

We will discuss the amount of b-fragments in Section 5.6.

Definition 5.10. An *increasing* trend of prices π satisfies $\pi_1 \leq \cdots \leq \pi_n$. Analogously, *decreasing* trends have $\pi_1 \geq \cdots \geq \pi_n$.

We examine the computational complexity of DynPPAI separately for both increasing and decreasing trends in the next two sections.

5.3 Decreasing Trends

5.3.1 Positive Graphs

In Section 3.5 we studied PPAI for positive graphs. Since π is fixed to a single value in PPAI, we could show a monotonicity of $p_C(v) - \pi$ in Lemma 3.12.

In our general dynamic pricing model this property is lost, because we can change π_i to any value. However, we show that it is still upheld in the case of decreasing trends.

Lemma 5.7. *Given a positive graph G and two decreasing trends of prices (π_i), (π'_i) with $\pi_i \geq \pi'_i$ for all $i = 1, \ldots, n$, it follows that*

$$C_i(\pi) \subseteq C_i(\pi').$$

Proof. The claim obviously holds for the first round. We show the claim by induction

Given $C_k(\pi) \subseteq C_k(\pi')$ we can compute the influenced values for each. We have to assure that no node from $V \setminus C_k(\pi')$ buys for the trend π without doing the same for π' in round $k + 1$.

This holds, as $p_{C_k(\pi)} \leq p_{C_k(\pi')}$ and $-\pi_{k+1} \leq -\pi'_{k+1}$, therefore $p_{C_k(\pi)} - \pi_{k+1} \leq p_{C_k(\pi')} - \pi'_{k+1}$. $\square$

Given this monotonicity for dynamic pricing we can redevelop Algorithm FixHighest into Fewest.

Lemma 5.8. *Algorithm Fewest computes the maximum revenue for decreasing trends on positive graphs in $\mathcal{O}(m + n \log n)$ time.*

Proof. By the construction of $p'(v)$ we guarantee that the prices are decreasing.

We know that v buys for the static price $p^*(v)$ by Lemma 3.12 and Theorem 3.14, but it does not for $p^*(v)+1$.

By Lemma 5.7 this implies that it can not buy for any decreasing trend π' with $\pi'_i \geq p^*(v)+1$. Recall that $\pi = (p^*(v)+1, \dots, p^*(v)+1)$ is also a decreasing trend. Thus the overall revenue $\sum_{v \in V} p^*(v)$ is optimal.

The runtime is obviously the same as of Algorithm FixHighest. □

Algorithm 8: Fewest(G)

```
Data: A positive graph $G = (V, A)$ with weights $p$ and $w$.
Result: The optimum trend of prices $\pi^*$ and its revenue $R^*$.
Initialize: Revenue $R^* = 0$, $\pi^* = ()$, values $p'(v) = p(v)$ for all $v \in V$, and $C = \emptyset$.
for $i = 1, \dots, n$ do
    if $C = V$ then
        return $R^*$ and $\pi^*$.
    Compute next price $\pi^*_i = \max_{v \in V \setminus C} p'(v)$.
    if $\pi^*_i \leq 0$ then
        return $R^*$ and $\pi^* = (\pi^*_1, \dots, \pi^*_{i-1}, 0, \dots, 0)$.
    Compute $B_i(\pi^*)$.
    $R = R + \pi^*_i \cdot |B_i(\pi^*)|$
    $C = C \cup B_i(\pi^*)$
    Update $p'(v) = \min\{p_C(v), \pi^*_i\}$ for all $v \in V \setminus F$.
```

Proposition 5.9. *Given a positive graph G the optimal solution for DynPPAI$^{\text{opt}}$ with decreasing prices may be up to H_n times as high as for the static setting in PPAI$^{\text{opt}}$, where H_n is the n-th partial sum of the harmonic series.*

Proof. We have a graph G with the nodes $v_1, \cdots, v_n$ sorted decreasingly, i.e., $p^*(v_1) \geq \cdots \geq p^*(v_n)$. The ratio between the optimal solutions is

$$\frac{\sum_{i=1}^n p^*(v_i)}{\max_{i=1}^n p^*(v_i) \cdot i}.$$

Assume for now that all $p^*(v_i)$ are optimal for PPAI$^{\text{opt}}$ on G, i.e., they yield the same revenue $R^* = p^*(v_i) \cdot i$ for $i = 1, \dots, n$. So in particular $p^*(v_i) = \dfrac{p^*(v_1)}{i}$ for all $i = 1, \dots, n$ and

$$\frac{\sum_{i=1}^n p^*(v_i)}{R^*} = \frac{p^*(v_1) \cdot H_n}{R^*} = H_n.$$

This gives us the ratio H_n, which we now show to be maximal.

Given an optimal $p^*(v_j) \cdot j = \max_{i=1}^{n} p^*(v_i) \cdot i = R^*$ we have $p^*(v_i) \leq j \cdot p^*(v_j) \cdot \frac{1}{i}$. Hence, for the ratio we have

$$\frac{\sum_{i=1}^{n} p^*(v_i)}{\max_{i=1}^{n} p^*(v_i) \cdot i} \leq \frac{j \cdot p^*(v_j) \cdot \sum_{i=1}^{n} \frac{1}{i}}{R^*} = H_n.$$

□

Remark 5.11. Applying Algorithm Fewest, without requiring the decreasing trend of prices, is not necessarily optimal or even r-approximate.

On the graph G of Figure 5.1 the algorithm would sell to v_3 in the first round. After that the influenced values of the other nodes are the same and we altogether achieve a revenue of 15.

The optimum revenue is achieved by selling to two nodes for price $\pi_1 = 2$ in the first round. The influenced value of v_1 rises to $15r - 3$, resulting in a revenue of $15r + 1$ over the 15 we got from Fewest.

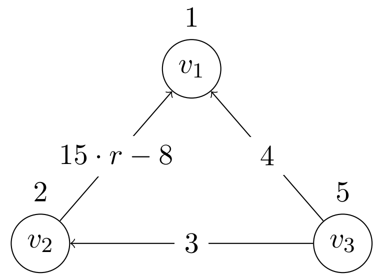

Figure 5.1: Graph with bad performance of Fewest

5.3.2 Negative Graphs

On negative graphs the selling for some static price π stops after the first round (see Remark 3.12).

For our dynamic pricing problem this implies that we can assume decreasing (contracted) trends of prices. This enables us to offer the product to nodes that are negatively influenced.

In the remainder of this subsection we show that it is particularly hard to find an adequate decreasing trend of prices. During the construction of gadgets we show that a satisfying variable assignment translates to a particular behavior and revenue. The other way, showing that this is the only possibility to achieve revenue $\hat{R}$, is covered in the proof of Theorem 5.14.

Gadget 5.12 (Variable Gadget)**.** Given an instance $\mathcal{J}$ of 3SAT with variables $x_1, \dots, x_{\tilde{n}}$ we construct the variable gadget.

The gadget $G^x = (V^x, A^x)$ consists of nodes $V^x := \{ t_1, \dots, t_{2\tilde{n}}, x_1, \dots, x_{\tilde{n}}, \overline{x}_1, \dots, \overline{x}_{\tilde{n}} \}$. Value X used in the initial values is determined later on in the proof of Theorem 5.14. The initial values and influences are as follows:

$$\begin{aligned}
p(t_{2i-1}) &= X + 4\tilde{n} - 4i + 3 && \text{for } i = 1, \dots, \tilde{n}\\
p(\overline{x}_i) &= X + 4\tilde{n} - 4i + 2 && \text{for } i = 1, \dots, \tilde{n}\\
w(t_{2i-1}, \overline{x}_i) &= -p(\overline{x}_i) && \text{for } i = 1, \dots, \tilde{n}\\
p(t_{2i}) &= X + 4\tilde{n} - 4i + 1 && \text{for } i = 1, \dots, \tilde{n}\\
p(x_i) &= X + 4\tilde{n} - 4i + 0 && \text{for } i = 1, \dots, \tilde{n}\\
w(t_{2i}, x_i) &= -p(x_i) && \text{for } i = 1, \dots, \tilde{n}
\end{aligned}$$

Proposition 5.10. *For every variable assignment of $\mathcal{J}$ we can choose a trend of sales such that the corresponding variable nodes buy and their negated counterparts do not. We call these* assignment trends*.*

Proof. It takes $2\tilde{n}$ rounds to achieve this. Note that we have the distinct initial values $X, \dots, X + 4\tilde{n} - 1$, i.e., we can choose $b_1 \in \{1, \dots, 4\tilde{n}\}$. For our trends corresponding to variable assignments we choose $b \in \{1, 2\}^{2\tilde{n}}$. For any $b_i > 2$ we can have the same buyers with $b_i = 2, b_{i+1} = 2, \dots, b_{i'} = 2$ with $i' := i + \left\lceil \frac{i-4}{2} \right\rceil$. For odd b_i we have $b_{i'+1} = 1$, whereas $b_{i'+1} = 2$ for even b_i.

In the first round we can sell to t_1 alone or t_1 and $\overline{x}_1$. If we do only sell to t_1 the influence of it deletes the initial value of $\overline{x}_1$. Hence, we have no chance to sell to $\overline{x}_1$ ever again. Both cases exclude t_1 and $\overline{x}_1$ from further purchases and leave the two current highest values to t_2 and x_1. As we want to have either x_1 or $\overline{x}_1$ buying, we need to set $b_1 \neq b_2$.

We say that $x_1 = \texttt{true}$ for $b = (1, 2, b_3, \dots, b_{2\tilde{n}})$ and $x_1 = \texttt{false}$ for $b = (2, 1, b_3, \dots, b_{2\tilde{n}})$.

Since the variable nodes x_1 and $\overline{x}_i$ do not further interact with the subsequent variable nodes, we can continue analogously. □

Note that we did not yet force the restriction to assignment trends. The restriction will be done through an inevitable loss of revenue any other trend suffers.

Remark 5.13. The revenue for a trend corresponding to a variable assignment is

$$R^x \in \left\{ 3\tilde{n}(X + 2\tilde{n} - 1), \dots, 3\tilde{n}(X + 2\tilde{n} - {}^1\!/_3) \right\}.$$

Gadget 5.14 (Clause Gadget)**.** For the clause gadget G^y we add nodes $y_1, \ldots, y_{\tilde{m}}$ corresponding to the clauses in $\mathcal{J}$ to G^x.

We set the initial values $p(y_i) = 3Y < X$ for all clause nodes. The influences between the variable nodes and the clause nodes have weight $-Y$.

However, this time we set the influences from the counterpart, i.e., $w(\overline{x}_i, y_j) = -Y$ for $x_i \in y_j$ and analogously $w(x_i, y_j) = -Y$ for $\overline{x}_i \in y_j$.

Proposition 5.11. *For every assignment trend $b = (b_1, \ldots, b_{2\tilde{n}})$, a clause node can buy after the first $2\tilde{n}$ rounds for some value $\pi_{2\tilde{n}+1}$ if and only if it is satisfied by the underlying variable assignment.*

Proof. We set $X > 3Y$ such that the clause nodes can not buy earlier during the variable assignment phase, i.e., rounds $1, \ldots, 2\tilde{n}$. If a clause node y_j is influenced by all its predecessors, the value is thereby decreased to 0 and no further purchase is possible. But this also implies that not a single actual variable of y_j bought and the clause is not `true`.

On the other hand, y_j clearly buys for Y if it is only influenced by up to two predecessors, implying one actual variable of y_j is `true`. ☐

Gadget 5.15 (Collector and Destructive Nodes)**.** We add the *collector node* z_- and the *destructive nodes* $d_1, \ldots, d_{\tilde{n}}$ to G^y.

The initial values of a destructive node d_i is $p(d_i) = 2Y$. Node d_i is influenced by the variable nodes x_i and $\overline{x}_i$ by $-Y$. The destructive nodes are introduced to ensure a proper assignment trend.

Here, the collector node z_- is chosen such that $p(z_-) = (\tilde{m}+\tilde{n}) \cdot (Z+1) < Y$. All clause and destructive nodes exert influence $-(Z+1)$ on z_-. In order to achieve the requested revenue, we want to avert a purchase of z_-.

Values Y and Z, like X in Gadget 5.14, are determined later on in the proof of Theorem 5.14.

For simplicity of notation, we denote a trend of sales b and a trend of prices π as pair (b, π).

Proposition 5.12. *For every extended assignment trend $(b_1, \ldots, b_{2\tilde{n}}, \pi_{2\tilde{n}+1} = Y)$ we decrease the influenced value $p_C(z_-)$ to 0 if and only if the underlying assignment is satisfying.*

Proof. Every assignment trend implies that the destructive nodes d_i are influenced by exactly one variable node. Thus their influenced value is exactly Y. This already decreases the influenced value of z_- to $m \cdot (Z+1)$.

The value drops to 0 if and only if it is also influenced by every clause node. By Proposition 5.11 we conclude the claim. ☐

Note that the revenue of such an instance is at least

$$R \geq 3X\tilde{n} + 12\binom{\tilde{n}}{2} + 3\tilde{n} + Y \cdot (\tilde{m} + \tilde{n}).$$

We finalize the construction of the reduction graph G (illustrated in Figure 5.2) with the Revenue Gadget 5.16.

Gadget 5.16 (Revenue Gadget)**.** We add the nodes $z_1, \ldots, z_M$ and $z'_1, \ldots, z'_{M'}$ with the following initial values and influences:

$$\begin{aligned} p(z_i) &= Z && \text{for } i = 1, \ldots, M \\ w(z_-, z_i) &= -Z && \text{for } i = 1, \ldots, M \\ p(z'_i) &= Z' && \text{for } i = 1, \ldots, M' \\ w(z_-, z'_i) &= -Z' && \text{for } i = 1, \ldots, M' \end{aligned}$$

We furthermore have $Z > Z'$.

The constructed reduction graph is illustrated in Figure 5.2.

Proposition 5.13. *For every extended satisfying assignment trend*

$$(b_1, \ldots, b_{2\tilde{n}}, \pi_{2\tilde{n}+1} = Y, \pi_{2\tilde{n}+2} = Z, \pi_{2\tilde{n}+3} = Z')$$

we generate a revenue of at least

$$\hat{R} \coloneqq 3X\tilde{n} + 12\binom{\tilde{n}}{2} + 3\tilde{n} + Y \cdot (\tilde{m} + \tilde{n}) + M \cdot Z + M' \cdot Z'.$$

We set $\hat{R}$ for the instance $\mathcal{I}$ such that every trend corresponding to a satisfying variable assignment gives $R(\pi) \geq \hat{R}$.

In the remainder of this subsection we show that there is no trend of prices π with $R(\pi) \geq \hat{R}$ that does not correlate to a satisfying assignment.

Theorem 5.14. *The problem DynPPAI is NP-complete for decreasing trends of prices.*

Proof. We have already determined the reduction graph up to minor details.

We now set the remaining values:

$$\begin{aligned} X &= (6\tilde{m})^4 & Y &= (6\tilde{m})^3 \\ Z &= (6\tilde{m})^2 & M &= (6\tilde{m})^4 \\ Z' &= (6\tilde{m})^1 & M' &= (6\tilde{m})^5 \end{aligned}$$

Note that with these values the requested order $X > 3Y > Y > (\tilde{n}+\tilde{m})\cdot(Z+1) > Z > Z'$ is uphold.

Thus

$$\hat{R} := 3\tilde{n}((6\tilde{m})^4 + 2\tilde{n} - 1) + (6\tilde{m})^3 \cdot (\tilde{m} + \tilde{n}) + 2 \cdot (6\tilde{m})^6$$

is the minimum revenue of a satisfying instance. We assume again that $\tilde{n} < 3\tilde{m}$ since otherwise we have unused variables.

Now consider the situation where we have any trend of prices π.

Property 5.3.1. If $z_- \in C(\pi)$, we have $R(\pi) < \hat{R}$.

Proof. Assuming the nodes prior to z_- have all bought for their initial values, we measure the difference between this revenue R' and $\hat{R}$.

$$R' - \hat{R} < -2(6\tilde{m})^6 + \frac{1}{2}(6\tilde{m})^5 + \frac{5}{6}(6\tilde{m})^4 + \frac{1}{2}(6\tilde{m})^2 + \frac{1}{2}(6\tilde{m})^1$$

We have to distinguish three cases for z_- buying. In all three cases the value $-2(6\tilde{m})^6$ dominates the achievable revenue of the revenue nodes.

1. Node z_- buys before the revenue nodes do. Thus we can only add up to $(\tilde{m} + \tilde{n}) \cdot (6\tilde{m})^2$ revenue from z_- itself to $R' - \hat{R}$. This is not enough revenue to reach $\hat{R}$ altogether.
2. Node z_- buys with the nodes $z_1, \dots, z_M$ and before $z'_1, \dots, z'_{M'}$. The maximum price for this is $(6\tilde{m})^2$ and we generate an additional revenue of $(6\tilde{m})^6 + (6\tilde{m})^2$, which is still not enough to reach $\hat{R}$.
3. Node z_- buys together with all revenue nodes. The price is at most $(6\tilde{m})^1$. The revenue $(6\tilde{m})^6 + (6\tilde{m})^5 + (6\tilde{m})^1$ of it does also not reach $\hat{R}$.

$\square$

Since we have to sell to the revenue nodes for the requested revenue, we know that the price has to drop below $Z+1$. In this case we need to have $\{y_1, \dots, y_{\tilde{m}}, d_1, \dots, d_{\tilde{n}}\} \subseteq C$.

Property 5.3.2. If $z_- \notin C(\pi)$ and we do not have an extended assignment trend π, it holds that $R(\pi) < \hat{R}$.

Proof. The easiest way to achieve $\{y_1, \dots, y_{\tilde{m}}, d_1, \dots, d_{\tilde{n}}\} \subseteq C$ is by directly selling to the clause and destructive nodes. That means we start with $\pi_1 = 2Y$. Then the revenue is exactly

$$2Y \cdot (5\tilde{n} + \tilde{m}) \leq \frac{16}{3} \cdot (6\tilde{m})^4.$$

The revenue we actually need, given that the purchases of the revenue nodes are guaranteed, is

$$3\tilde{n}((6\tilde{m})^4 + 2\tilde{n} - 1) + (\tilde{m} + \tilde{n})(6\tilde{m})^3.$$

Thus, in order to achieve the requested revenue, we can not sell to clause and destructive nodes in the first round. Instead we start with prices like in an assignment trend.

If we do sell to a pair x_i, $\overline{x}_i$ of variable nodes, this must be at a price such that d_i can also buy in the same round as x_i. This implies that we are also selling to all nodes x_j, $\overline{x}_j$ for $j > i$ at the same time.

We distinguish two cases:

1. The first $2t - 2$ rounds we sold to exactly one corresponding variable node. In round $2t - 1$ we sell to $\overline{x}_t$, x_t, and d_t for $2Y$. The revenue of the variable gadget, clause and destructive nodes is at most:

$$\begin{aligned}&\sum_{i=1}^{t-1}(3X + 12\tilde{n} - 12i + 5 + Y) + \sum_{i=t}^{\tilde{n}}(5 \cdot 2Y) + \tilde{m} \cdot 2Y \\ =&(t-1)\cdot(3X + 6(2\tilde{n} - t) + 5 + Y) + (\tilde{n} - t + 1)\cdot(10Y) + \tilde{m}2Y \qquad (5.1)\end{aligned}$$

2. We sold the first $2t - 1$ rounds according to a variable assignment. In round $2t$ we sell to x_t and d_t for Y. The accumulated revenue is at most:

$$\begin{aligned}&\sum_{i=1}^{t-1}(3X + 12\tilde{n} - 12i + 5 + Y) + 2X + 8(\tilde{n} - t) + 4 + 2 \cdot Y + \tilde{m} \cdot Y \\ &\qquad + \sum_{i=t}^{\tilde{n}}(5 \cdot Y) \\ &= (t-1)\cdot(3X + 6(2\tilde{n} - t) + 5 + Y) + 2X + 8(\tilde{n} - t) + 4 \\ &\qquad + Y(2 + 5(\tilde{n} - t + 1) + \tilde{m}) \qquad (3.2)\end{aligned}$$

Both Equations (1) and (3.2) are less than the revenue an assignment trend achieves on the same nodes, namely $3n(X + 2n - 1) + (m + n)Y$.

Hence, we showed that we need an assignment trend to achieve the requested revenue. This concludes the proof of Property 5.3.2. □

The remaining case was already done with the construction of $\hat{R}$ and requires a satisfying variable assignment for the underlying $\mathcal{J}$ instance of 3SAT.

Since the constructed graph, such as its initial values and influences, is of polynomial size, the construction is complete and the claim follows. □

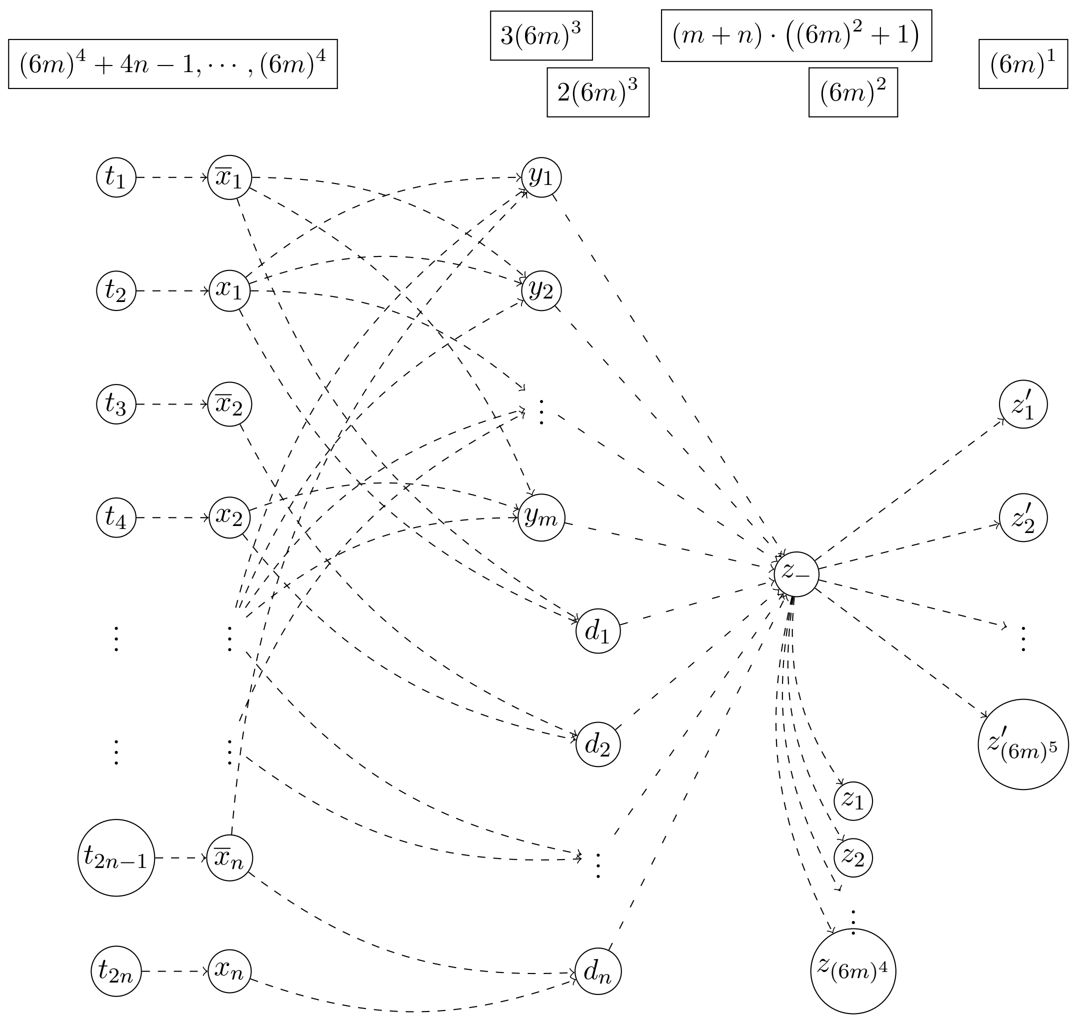

Figure 5.2: Reduction Graph for DynPPAI on negative graphs and decreasing trend of prices

Since the influences are negative any contracted trend of prices is decreasing. This already gives us two results for a more general setting.

Corollary 5.15. *The problem* DynPPAI *is also* NP*-complete regarding*

- *negative graphs with arbitrary trends,*
- *arbitrary graphs with negative trends, and*
- *arbitray graphs with arbitrary trends.*

5.4 Increasing Trends

In this section we show that DynPPAI is also NP-complete for increasing trends of prices and positive graphs.

We construct the reduction graph for an instance $\mathcal{J}$ of 3SAT as usually.

Gadget 5.17 (Reduction Graph)**.** The variable gadget G^x consists of nodes

$$V^x = t_1, \dots, t_{2\tilde{n}+2}, x_1, \dots, x_{\tilde{n}}, \overline{x}_1, \dots, \overline{x}_{\tilde{n}}, d_1, \dots, d_{\tilde{n}}.$$

The initial value $p(t_1) = 1$ is the only one not set to 0.

We define the arcs indirectly by their influences as follows:

$$\begin{aligned}
w(t_{2i-1}, \overline{x}_i) &= 2i - 1 && \text{for } i = 1, \dots, \tilde{n} \\
w(t_{2i-1}, t_{2i}) &= 2i && \text{for } i = 1, \dots, \tilde{n} \\
w(t_{2i}, x_i) &= 2i && \text{for } i = 1, \dots, \tilde{n} \\
w(t_{2i}, t_{2i+1}) &= 2i + 1 && \text{for } i = 1, \dots, \tilde{n} \\
w(x_i, d_i) &= i + 1 && \text{for } i = 1, \dots, \tilde{n} \\
w(\overline{x}_i, d_i) &= i + 1 && \text{for } i = 1, \dots, \tilde{n}
\end{aligned}$$

We directly add the remaining nodes, since their interaction is needed to "steer" the variable nodes. Those are the variable nodes $y_1, \dots, y_{\tilde{m}}$, the collector node z_+ with $p(z_+) = -\tilde{m}$, and the revenue node z_M with $p(z_M) = -1$.

Their influences are as follows:

$$\begin{aligned}
w(x_i, y_j) &= 1 && \text{if } x_i \in y_j \\
w(\overline{x}_i, y_j) &= 1 && \text{if } \overline{x}_i \in y_j \\
w(t_{2\tilde{n}+1}, y_j) &= 2\tilde{n} + 1 && \text{for } j = 1, \dots, \tilde{m} \\
w(t_{2\tilde{n}+2}, z_+) &= 2\tilde{n} + 2 && \\
w(y_i, z_+) &= 1 && \text{for } i = 1 \dots, \tilde{m} \\
w(d_i, z_M) &= 2 && \text{for } i = 1, \dots, \tilde{n}
\end{aligned}$$

$$\begin{aligned} w(t_i, z_M) &= 1 && \text{for } i = 1, \dots, 2\tilde{n} \\ w(t_{2\tilde{n}+1}, z_M) &= 2 \\ w(z_+, z_M) &= M \end{aligned}$$

The reduction graph is illustrated in Figure 5.3.

Theorem 5.16. *DynPPAI with increasing trends of prices is NP-complete on positive graphs.*

Proof. We show that instance $\mathcal{I}$ on Gadget 5.17 with requested revenue

$$\hat{R} := \tilde{m}(2\tilde{n} + 4) + 5\tilde{n}^2 + 13\tilde{n} + 5$$

is equivalent to our 3SAT instance $\mathcal{J}$.

The maximum revenue can surely be bounded by the sum of all initial values and influences. By choice, this sum is exactly $\hat{R} - 1 + M$. To achieve the revenue of $\hat{R}$, we require the exerted influence $w(z_+, z_M)$, i.e., the node z_+ has to buy before z_M, as

$$\sum_{v \in V} p(v) + \sum_{a \in A \setminus \{(z_+, z_M)\}} w(a) = \tilde{m} \cdot (2\tilde{n} + 4) + 5\tilde{n}^2 + 11\tilde{n} + 2 < \hat{R}$$

bounds the maximum revenue without said influence. Apart of node z_M, nodes t_i have the highest influenced value in rounds $i = 1, \dots, 2\tilde{n} + 2$. Hence, assuming every round has at least one buyer, nodes t_i buy in rounds i. Furthermore, the influenced value of z_M in round $i = 1, \dots, 2\tilde{n}$ is at least $i - 2$, since the influences from the time nodes are secured. So, in order to achieve the required revenue $\hat{R}$, for rounds $i = 1, \dots, 2\tilde{n} + 2$ the prices have to satisfy $\pi_i \in \{i - 1, i\}$. Remark that thereby the trend of prices is guaranteed to be increasing.

Variable Nodes: In rounds $i = 2, \dots, 2\tilde{n} + 1$, if $\pi_i = i$ the time node buys, but the "current" variable node (either $\overline{x}_{\frac{i}{2}}$ for even i or $x_{\frac{i-1}{2}}$ otherwise) does not. As its influenced value will not increase anymore and the trend of prices is ascending, this is a final state. Otherwise, with $\pi_i = i - 1$, the time node and variable node buy.

A variable node in G buying corresponds to the variable set to be `true` in the underlying instance of 3SAT. Hence, we have to make sure that (illegitimately) setting $x = \texttt{true}$ and $\overline{x} = \texttt{true}$ does not benefit the potential to reach revenue $\hat{R}$. The destructive nodes d_i are influenced by the variable nodes $\overline{x}_i$ and x_i, such that a single influence is not enough to persuade d_i to buy. If both variable nodes bought, in round $2i + 2$ the influenced value $p_B(d_i) = 2i + 2 = p_B(t_{2i+2})$ and d_i buys.

But for the next round $2i + 3$ this means that $p_B(t_{2i+3}) = 2i + 3 = p_B(z_M)$ and z_M buys before z_+. Thus for an optimal trend of prices we can rule out $x = \texttt{true} = \overline{x}$. It is still allowed that neither the variable node nor its counterpart buys, but changing the trend of prices such that exactly one of them is buying, does only improve the overall revenue.

Clause Nodes: The clause nodes y_i for $i = 1, \dots, m$ are influenced by their variable nodes contained and time node $t_{2\tilde{n}+1}$. Until round $2\tilde{n}+2$ only influences of the variable nodes are exerted, with a maximum of $p_B(y_i) = 3$. Hence, even for the first few rounds, this does not suffice for y_i to buy beforehand.

In round $2\tilde{n}+2$ the influence $w(t_{2\tilde{n}+1}, y_i) = 2\tilde{n}+1$ is added. By then, as long as no destructive node bought, node z_M has reached $p_B(z_M) = 2\tilde{n}+1$. Thus y_i needs the influence of at least one variable node to surpass z_M and buy before it. As in the underlying instance of 3SAT the clause node buys (is `true`) if and only if at least one of the contained variable nodes buys (are `true`).

Collecting Node and the Big Contributor: The collecting node z_+ summarizes whether all clause nodes have bought. If so, the influenced value in round $2n+3$ is $p_B(z_+) = 2n+2 \geq \pi_{2n+2} = 2n+2$ and by keeping the old price z_+ finally buys before z_M. With $\pi_{2\tilde{n}+4} = 2\tilde{n}+1+R$ alone we can surpass the requested revenue.

All in all, the instance $\mathcal{J}$ of 3SAT and the constructed instance $\mathcal{I}$ of DynPPAI with positive influences and increasing trends of prices are equivalent. Since the construction given in this proof can be done in polynomial time and DynPPAI $\in$ NP by Corollary 5.2, the claim holds.

$\square$

Since the influence $w(z_+, z_M)$ in itself is more than the rest, we can conclude this corollary on approximability with $w(z_+, z_M) = r \cdot M$.

Corollary 5.17. *Unless P = NP, there is no polynomial time r-approximate algorithm for DynPPAI$^{\text{opt}}$ for any $r = \mathcal{O}(1)$.*

For the remaining result on graphs with positive influences and decreasing trends of prices, we will prove some helpful results first.

We already noted that the selling stops after one round for increasing trends and negative graphs. The solution is to choose the price giving the most revenue in this first round. The following Algorithm Most stems from this idea.

Lemma 5.18. *Algorithm Most runs in time $\mathcal{O}\left(n^2 + m \log n\right)$.*

For negative graphs and increasing trends Most computes the optimum revenue after one round in $\mathcal{O}(\tilde{n} \log \tilde{n})$ time.

Proof. The algorithm computes one possible b_i for each round and follows through on this particular trend of sales, in contrast to bFrag which searches over all.

Correctness in the standard case is clear from construction.

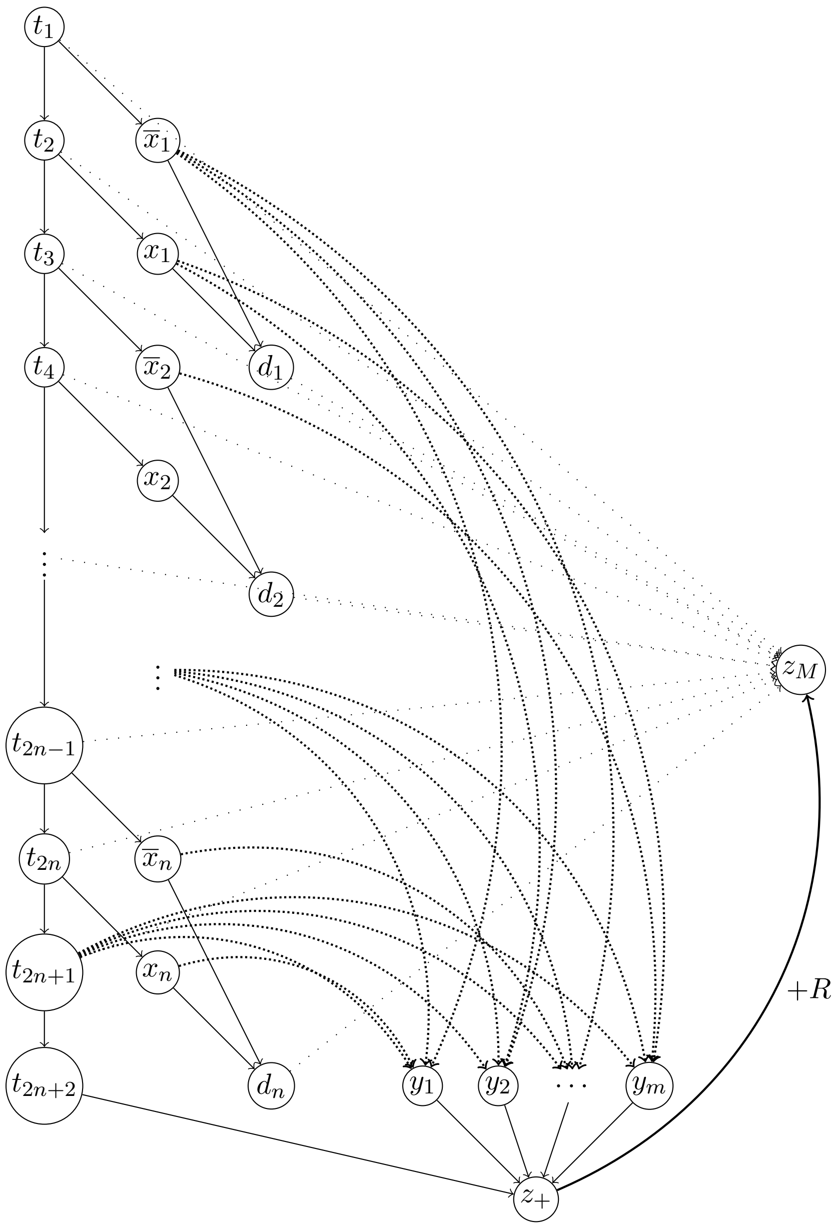

Figure 5.3: Reduction Graph for DynPPAI on positive graphs and increasing trend of prices

Algorithm 9: Most

```
Data: A graph $G = (V, A)$ with weights $p$ and $w$.
Result: A trend of prices $\pi$ and its revenue $R(\pi)$.
Initialize: Revenue $R = 0$, $\pi = ()$, values $p'(v) = p(v)$ for all $v \in V$, and $C = \emptyset$.
for $i = 1, \dots, n$ do
    if $C = V$ then
        return $R$ and $\pi$.
    Compute next price $\pi_i = \text{argmax}_{\{p'(v):v \in V \setminus C\}} |B_i(p'(v))| \cdot p'(v)$.
    Set $R = R + \pi_i \cdot |B_i(\pi_i)|$ and $C = C \cup B_i(\pi_i)$.
    if $\pi_i \leq 0$ then
        return $R$ and $\pi = (\pi_1, \dots, \pi_{i-1}, 0, \dots, 0)$.
    Update $p'(v) = \min\{p_C(v), \pi_i\}$ for all $v \in V \setminus F$.
```

In every **for**-iteration of Most we have to determine the price for the highest revenue and update the influenced values. We use a height-balanced binary search tree to store the nodes sorted by values $p'(v)$.

This search tree is updated m times, i.e., once for every arc. Changing the influenced value of a single node is accomplished by deleting and inserting the node once. This takes $\mathcal{O}(m \log n)$ time in total.

Using the search tree to find the maximum revenue in a single **for**-iteration is therefore only taking $\mathcal{O}(n)$ time. Since we have at most n of those iterations, we conclude the runtime $\mathcal{O}\left(n^2 + m \log n\right)$.

For negative graphs we have already noted that prices have to decrease. So with the restriction to increasing prices, we can only stay static. Hence, this was already covered in Proposition 3.11 □

Remark 5.18. Algorithm Most provides the same result as Fewest on the graph in Figure 5.1.

We conclude the sections on increasing and decreasing trends with an overview of the complexity results.

Corollary 5.19. *We have shown the complexity of DynPPAI for all combinations of influences and trend of prices.*

influences / *trend of prices*	*arbitrary*	*positive*	*negative*
arbitrary	*NP-complete*	*NP-complete*	*NP-complete*
decreasing	*NP-complete*	*Fewest in $\mathcal{O}(m + n \log n)$*	*NP-complete*
increasing	*NP-complete*	*NP-complete*	*Most in $\mathcal{O}(n)$*

5.5 Bounded Amount of Prices

In the reductions for increasing and decreasing trends of prices, we limited the feasible prices to a narrow strip — $\pi_i \in \{i-1, i\}$ and $\pi_i \in \{X + 4\tilde{n} - 2i, X + 4\tilde{n} - 2i + 1\}$, respectively. This was managed by the introduction of a negative node z_-. Even though we only had a few viable prices in each round, the total amount of different prices remained large.

With the possibility of arbitrary prices and trends we can furthermore limit the possible prices to a solid set Π and still construct an instance $\mathcal{I}$ of DynPPAI corresponding to some 3SAT instance $\mathcal{J}$.

Definition 5.19. Let $\Pi \subset \mathbb{N}$ be a *set of prices.* If $|\Pi| = \mathcal{O}(1)$ we speak of a *bounded amount of prices.*

Remark 5.20. Both reduction graphs in Figures 5.3 and 5.2 contained no directed cycle. Using the Split-Copy Gadget 3.32 both can be transformed to trees which are still of polynomial size.

For the reduction in this section we will directly construct a tree. The reduction also includes a more direct take on the variants PPAI-S and PPAI-v with the restriction $\pi \in \{1,2\}^n =: \hat{\Pi}$.

We start again with the variable gadget. Since we need to have a separate in-tree for every literal influencing the clause nodes, the constructed gadget is actually ambiguous whether x_i or $\overline{x}_i$ is used.

Gadget 5.21 (Literal Gadget)**.** For some literal $l \in y_j$, where $l \in \{x_i, \overline{x}_i\}$, we construct its gadget G^l.

The tree G^l is illustrated in Figure 5.4, where the initial values are 0 except for $p(t_1^{d_i}) = p(t_1^{\tilde{x}_i}) = p(t_1^{x_i}) = 2$. Note that the time nodes on the left are nonexistent for $i = 1$ and instead their influences are given to d_1, $\tilde{x}_1$, and $t_1^{x_1}$ directly.

The same holds analogously for the right side and $i = n$.

If $l = x_i$ node x_i is the sole influence to t_{i+1}^l with $w(x_i, t_{i+1}^l) = 2$, otherwise it is $w(\overline{x}_i, t_{i+1}^l) = 2$.

Proposition 5.20. *Any contracted trend of prices π in Gadget 5.21 satisfies $\pi \in \{1,2\}^{\tilde{n}+1}$.*

If $\pi_i = 1$ node $\overline{x}_i$ buys in round $i+1$. Otherwise x_i buys in round $i+1$.

Proof. Up to round $i-1$ the time nodes $t_1, \dots, t_{i-1}$ buy consecutively and delay the process. All these nodes have influenced value 2, so the prices are either 1 or 2.

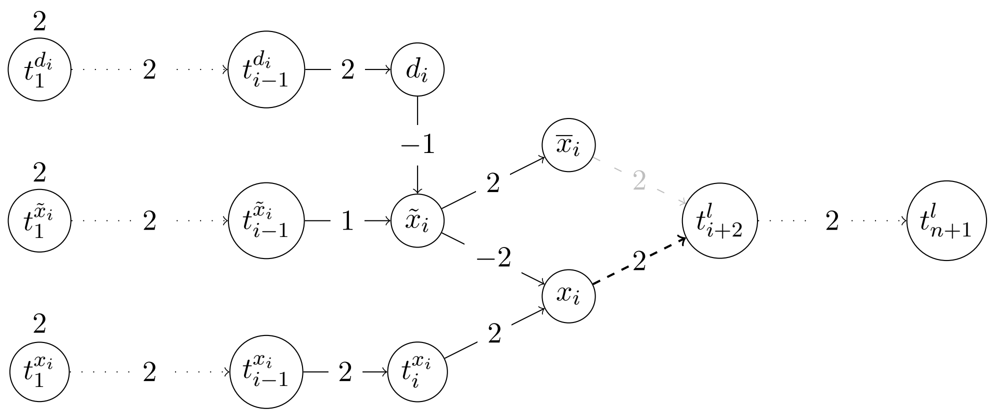

Figure 5.4: Tree Literal Gadget (in this case G^{x_i})

In round i the influenced values are as follows:

$$p_{C_{t-1}(\pi)}(d_i) = 2, \quad p_{C_{t-1}(\pi)}(\tilde{x}_i) = 1, \text{ and } \quad p_{C_{t-1}(\pi)}(t_i^{x_i}) = 2.$$

We have the choice $\pi_i = 1$ or $\pi_i = 2$. For $\pi_i = 1$ all of these nodes buy. Therefore, the influenced values in the next round are

$$p_{C_t(\pi)}(x_i) = 0 \text{ and } \quad p_{C_t(\pi)}(\overline{x}_i) = 2.$$

If the gadget is actually for $\overline{x}_i = l$ it follows that node t_{n+1}^l buys in round $n+1$.

With the other choice $\pi_i = 2$ we only sell to d_i and $t_i^{x_i}$. This erases the influenced value of $\tilde{x}_i$, hence, this node can not buy anymore. In contrast to the previous choice, the influenced values of x_i and $\overline{x}_i$ are reversed, such as their buying behavior. □

Theorem 5.21. *The problem DynPPAI is NP-complete for a bounded amount of prices even on trees.*

Proof. Given $\mathcal{J}$, we construct a literal gadget copy for all $3\tilde{m}$ literals. For any trend of prices π we restrict $\pi_i \in \{1,2\} =: \Pi$. A clause node $y_j := \{l_1, l_2, l_3\}$ with initial value $p(y_j) = 0$ is then influenced by 2 from nodes $t_{\tilde{n}+1}^{l_1}$, $t_{\tilde{n}+1}^{l_2}$, $t_{\tilde{n}+1}^{l_3}$.

The influenced value is in $\{0, 2, 4, 6\}$, so even without a restriction to $\Pi = \{1, 2\}$ we would have at most 6 different possible prices with buyers.

Thus, in round $n+2$ the clause nodes buy for $\pi_{\tilde{n}+2}$ if and only if they are influenced by at least one of their literal nodes. Note that this already implies a reduction for DynPPAI-S.

Thus far we have introduced $3\tilde{m}$ literal gadgets and $\tilde{m}$ more nodes. A literal gadget contains $\tilde{n} + 2(i+1) \leq 3\tilde{n} + 2$ nodes. In total we have up to $\tilde{m}(9\tilde{n} + 7)$ nodes and at most twice that revenue.

In the next step we connect those nodes to the collector node z_+ with $p(z_+) = 2 - 2\tilde{m}$ and $w(y_j, z_+) = 2$ for all $j = 1, \dots, \tilde{m}$. Node z_+ buys if and only if all clause nodes do. Hence, this implies a reduction for DynPPAI-v.

At the end, we connect z_+ to revenue nodes z_M for $M = \tilde{m}(9\tilde{n}+7)$ and set $\hat{R} = 2(M+1)$. Since $n = 2M + 1$ the collector node together with the revenue nodes are the majority in the reduction graph and their purchase for $\pi_{\tilde{n}+3} = \pi_{\tilde{n}+4} = 2$ is required. □

Corollary 5.22. *The problems DynPPAI-v, DynPPAI-S, DynPPAI-i are also NP-complete.*

Unless P = NP, there is no polynomial time r-approximate algorithm for DynPPAI$^{\text{opt}}$ with $r = \mathcal{O}(1)$.

Proof. This follows by raising the amount of revenue nodes to rM. □

In the proof of Theorem 5.21 we allowed only two different prices. This sufficed to make the problem NP-complete even on trees, in contrast to Conjecture 3.40 and PPAI where exactly one price is allowed.

In the next section we restrict the trends of prices even further and allow only a bounded amount of price changes.

5.6 Bounded Amount of Price Changes

The results in this chapter are most closely connected to Chapter 3.

Definition 5.22. Given a trend of prices $(\pi_i)_{i \in \mathbb{N}_n}$, we have a price change in round $i = 2, \dots, n$ if $\pi_{i-1} \neq \pi_i$. We say that π is c-bounded if at most c price changes occur in this trend of prices.

Remark 5.23. Recall that frag(n) denotes the maximum amount of fragments possible for graphs G with n nodes and contained in some graph class $\mathcal{G}$. For the classes we discussed so far (bounded degree, bounded path length, sign constrained, trees, etc.) it holds that the subgraphs, created by deletion of nodes, are still part of this class.

Proposition 5.23. *For any graph G and trend of prices π with the first price change in round $t + 1$, we can assume that π_1 is chosen from the endpoints of* $\text{frag}_t(G)$.

Proof. Considering any price π_i, it lies in a temporary fragment. Since we change the price anyways in the next round, the upper endpoint of said fragment yields a better revenue with the same set of customers. □

Note that in the context of a bounded amount of price changes we can not assume that the prices are set to a current influenced value. Thus the notion of b-fragments and trends of sales is not applicable in this case, since they miss the one-to-one correspondence with trends of prices. However, we can use the original fragments again.

Lemma 5.24. *For any graph class $\mathcal{G}$ with* $\operatorname{frag}(n)$ *fragments, we have at most*

$$\binom{n-1}{c} \cdot \operatorname{frag}(n)^{c+1}$$

dominating trends of prices with up to c price changes. Algorithm bFrag *therefore solves the problem in $\mathcal{O}\left(n^c \cdot \operatorname{frag}(n)^{c+1} \cdot (m + n\log n)\right)$ time.*

Proof. We have $\binom{n-1}{c}$ choices when to place the price changes. Since $\operatorname{frag}(n)$ is increasing, we can assume that for each time interval around the price changes we have up to $\operatorname{frag}(n)$ fragment endpoints to choose from. □

Corollary 5.25. *As in Chapter 3, we can polynomially bound the amount of choices for the same graph classes with* $\operatorname{frag}(n) = \operatorname{poly}(n)$ *(see Remark 3.17 and Section 3.6 in general), if we restrict the amount of price changes to $c = \mathcal{O}(1)$.*

5.7 Integer Program

Just like the Integer Program 1, we can formulate an integer program in the dynamic setting.

Integer Program 2: DynPPAI *(Formulation of DynPPAI)*

$$\max \sum_{v \in V} \sum_{t=1}^{n} \pi_t \cdot x_{v,t}$$

subject to

$$\begin{aligned}
x_{v,t} &\geq 0 && \forall v \in V, t = 1, \ldots, n \\
x_{v,t} &\leq 1 && \forall v \in V, t = 1, \ldots, n \\
\sum_{t=1}^{n} x_{v,t} &\leq 1 && \forall v \in V \\
\sum_{j<t} x_{v,j} &= y_{v,t} && \forall v \in V, t = 1, \ldots, n \\
p(v) + \sum_{u \in \mathcal{N}^-(v)} y_{u,t} \cdot w(u,v) &= p_{v,t} && \forall v \in V, t = 1, \ldots, n \\
p_{v,t} - \pi_t &\geq (x_{v,t} - 1) \cdot M && \forall v \in V, t = 1, \ldots, n
\end{aligned}$$

$$p_{v,t} - \pi_t \leq (x_{v,t} + y_{v,t}) \cdot M - 1 \qquad \forall v \in V, t = 1, \dots, n$$

If we want to restrict the amount of price changes to c, we can add the following conditions:

$$\begin{aligned}
c_t &\leq 1 && t = 1, \dots, n \\
c_t &\geq 0 && t = 1, \dots, n \\
c_1 &= 0 \\
\sum_{t=1}^{n} c_t &\leq k \\
c_t \cdot M &\geq \pi_t - \pi_{t-1} && t = 1, \dots, n \\
c_t \cdot M &\geq \pi_{t-1} - \pi_t && t = 1, \dots, n
\end{aligned}$$

Observation 5.24. In Observation 3.38 we already stated for PPAI — the formulation of a PPAI instance — that using the combinatorial algorithm Frag had a better performance on the $G(n, p)$ graphs.

We did not implement Program 2, though we can assume that it is even harder, since we only added more variables and equations.

6 Perishable Product Pricing

In this chapter we break with the paradigm "once a buyer, always a customer".

We provide a model for perishable products and subscriptions alike. The customers may or may not rebuy after the product perishes or the subscription ends. In Section 6.4 we connect the model to the basic PPAI problem with returnable products.

Definition 6.1. Given a graph $G = (V, A)$ with initial values p and influences w, we introduce the *duration* $d(v) \in \mathbb{N}$ for all $v \in V$, after which node v may — or may not — rebuy the product for price π.

The influenced value is computed as usual, but the set of customers is not increasing anymore. Instead v stays a customer for $d(v)$ rounds, i.e.,

$$v \in C_t(\pi) \Leftrightarrow v \in B_t \vee \cdots \vee v \in B_{t-d(v)+1}.$$

After that, the product is consumed and the node may buy again.

Node v with $d(v) > 1$ buys in round t if $v \notin B_{t-1} \cup \cdots \cup B_{t-d(v)+1}$ and $p_{C_t}(v) \geq \pi$. For node $v \in V$ with $d(v) = 1$ we just require the influenced value, i.e., $p_{C_t}(v) \geq \pi$.

The selling ends after T rounds, the given time horizon as in Definition 5.1. Every single purchase counts towards the revenue, i.e.,

$$R(\pi, T) = \pi \cdot \sum_{t=1}^{T} |B_t(\pi)|.$$

The special case that $d(v) = d$ for all $v \in V$ for some constant $d \in \mathbb{N}$ will be of particular interest. We call this the case of *fixed durations.* Otherwise we say that durations are *individual.* The special case of a fixed duration $d = 1$ for all nodes will be referred to as *unit durations.*

Problem 4: PerPPAI *(Perishable Product Pricing with Additive Influences)*

Instance: *A directed graph $G = (V, A)$, initial values $p(v) \in \mathbb{Z}$ for the nodes $v \in V$, influences $w(u, v) \in \mathbb{Z} \setminus \{0\}$ for the arcs $(u, v) \in A$, individual durations $d(v) \in \mathbb{N}$ for the nodes $v \in V$, time horizon T, and some revenue $\hat{R} \in \mathbb{N}$.*

Question: *Is there a price $\pi \in \mathbb{N}$ such that its revenue $R(\pi) \geq \hat{R}$?*

As in Definition 3.1 we define the variants of PerPPAI:

Definition 6.2 (Variants of PerPPAI)**.**

PerPPAI$^\pi$: Compute the revenue for a given price.

PerPPAI$^{\text{opt}}$: The corresponding revenue maximization problem.

A major focus of this chapter is the problem PerPPAI$^\pi$. The problems PerPPAI and PerPPAI$^{\text{opt}}$, i.e., finding a price with high revenue, are studied in Section 6.2 after we determined the complexity of computing the revenue for a given price.

6.1 Computing the Revenue

Observe that it is trivial to compute the revenue $R(\pi, T)$ in time $\mathcal{O}(mT)$ by going through each round $1, \ldots, T$ iteratively, updating influenced values and consumers. However, this running time is only pseudo-polynomial, since the encoding size is logarithmic in T. Intuitively, the moments in time when nodes change their decisions about whether to buy the product or not, capture the "important" rounds.

Remark 6.3. Without loss of generality, we can assume $\pi = 1$, since otherwise we can shift the initial values as in Gadget 3.26. Thereby $R(1, T)$ sums the number of purchases itself. We consider PerPPAI$^\pi$ consequently as a counting problem from now on.

Definition 6.4. We say that a node v *breaks* in round t if

$$v \in (C_{t-1} \cup C_t) \setminus (C_{t-1} \cap C_t) =: C_{t-1} \bigtriangleup C_t,$$

where $C_0 = \emptyset$ as usual. A round $t \in \{1, \ldots, T\}$ in which at least one node breaks is called a *break*.

The total number of breaks will be denoted by β.

By definition, the set of consumers remains unchanged between two breaks. This idea is exploited in Algorithm Break, which computes the revenue $R(\pi, T)$ in a more efficient way than the naive approach outlined above. For every node $v \in V$, we keep a number $z(v) \in \mathbb{N}$, which is the earliest time after the current round when v must be rechecked for its buying decision. Initially, $z(v) = 1$ for all $v \in V$.

Lemma 6.1. *Algorithm Break correctly computes the revenue $R(\pi, T)$ in $\mathcal{O}(m\beta)$ time.*

Proof. Obviously, each round where t increases by 1 is handled correctly by the algorithm. Moreover, between t and the next value of t considered in the **while**-loop, no break occurs and all current consumers repeat buying, yielding additional revenue of $k \cdot \pi$, but not changing any influenced values or decisions.

Algorithm 10: Break(G, π, T)

```
Data: A graph $G = (V, A)$ with initial values $p$, influences $w$, durations $d$, price $\pi$ and
      time horizon $1, \dots, T$.
Result: The revenue $R(\pi, T)$ generated in $G$.
Initialize $R = 0$, $C = \emptyset$, $t = 1$, $p'(v) = p(v)$ and $z(v) = 1$ for all $v \in V$.
                                    // $z(v)$ is the next round to check whether $z$ buys
while $t \leq T$ do
    for $v \in V$ with $z(v) = t$ do
        if $p'(v) \geq \pi$ then
            $C = C \cup \{v\}$, $R = R + \pi$, $z(v) = t + d(v)$
        else
            $C = C \setminus \{v\}$, $z(v) = t + 1$
    $t = T + 1$
    Update $p'(v)$ for all $v \in V$.
                                    // Find the next round where a node breaks
    $D = \{ v \in V : (v \in C \text{ and } p'(v) < \pi) \text{ or } (v \notin C \text{ and } p'(v) \geq \pi) \}$
    Find $v \in D$ with minimum $z(v)$.
    $t = z(v)$.
                                    // Update revenue
    for $v \in V$ with $z(v) < t$ do
        if $v \in C$ then
            $k = \left\lceil \frac{t - z(v)}{d(v)} \right\rceil$
            $R = R + k \cdot \pi$
            $z(v) = z(v) + k \cdot d(v)$
        else
            $z(v) = t$
return $R(\pi, T) = R$
```

Furthermore, a break occurs in all — but possibly the last — **while**-iterations. So, in total, we have β or $\beta + 1$ rounds to check.

Each **while**-iteration itself has to update the influenced values $p'(v)$. In the previous chapters the computation of the influenced values took $\mathcal{O}(m)$ in total, since we had no customers dropping out. In Break the same time is potentially needed in every iteration.

In $\mathcal{O}(n)$ time we determine the set D and its minimum entry $z(v)$. The same holds for the amounts of purchases k of each node $v \in C$.

Hence, the algorithm runs in time $\mathcal{O}(m \cdot \beta)$ as claimed. □

Remark 6.5. Note that Algorithm Break does not detect a "periodical" buying behavior. In Algorithm Stint, introduced in Section 6.1.2 on negative influences, we detect such behavior and avoid checking every break. Therefore, it can run in polynomial time even for $\beta = T$.

Yet, we have not provided a polynomial time algorithm to compute the revenue. Therefore, it remains unclear thus far whether PerPPAI is in NP. We examine this in detail, before we start with PerPPAI itself.

6.1.1 Positive Influences

Restricting the influences to be positive, we observe a "good" behavior of the consumers. This is similar to the results in Section 3.5, where non-perishable products were considered.

Lemma 6.2. *On positive graphs, consumers will always renew the purchase of the product without breaking. Hence, $\beta \leq n$.*

Proof. Assume that t is the first round in which some consumer v does not renew their purchase. Then, by definition, the set of consumers in rounds $1, \dots, t-1$ has never shrunk, i.e., $C_1 \subseteq C_2 \subseteq \cdots \subseteq C_{t-1}$. In particular, $C_{t-1} \supseteq C_{t-d(v)-1}$, and all the influencing consumers from the previous purchase of v in round $t - d(v)$ still own the product in round $t-1$. As all influences are positive, this implies that $\pi \leq p_{C_{t-d(v)-1}}(v) \leq p_{C_{t-1}}(v)$, which contradicts the assumption that v does not renew in round t. □

As a direct consequence of Lemmas 6.1 and 6.2, we obtain that, for positive graphs, Break runs in time $\mathcal{O}(mn)$. This is already polynomial, but we are still initiating superfluous computations.

So, for positive graphs, it is straightforward to see that all breaks must happen in consecutive rounds. Thus, determining the set D and finding the node $v \in D$ with minimum value $z(v)$ is, in fact, not necessary. Finally, since the set of consumers only grows, it suffices to update the values of all nodes u such that $(v, u) \in A$ at the moment in time when v becomes a consumer. Basically we can apply Algorithm Sell and compute the revenue contributed by $v \in B_t(\pi)$ as $\pi \cdot \left\lceil \frac{t+1-t}{d(v)} \right\rceil$.

Corollary 6.3. *For positive graphs the revenue $R(\pi, T)$ can be computed in time $\mathcal{O}(m)$. Thus PerPPAI$^\pi$ $\in$ FP and PerPPAI $\in$ NP.*

6.1.2 Negative Influences

For some products (e.g., a sports pay TV subscription), it appears less likely that one wants to buy it if a good friend already has it and shares it with others. We model this phenomenon by negative influences of consumers on their friends. Recall that for PPAI$^\pi$ on negative graphs we had no purchases after the first round.

The problem PerPPAI$^\pi$, however, is more interesting on negative graphs, since influences are fading. We first derive some important structural properties:

Lemma 6.4. *Suppose that all influences are negative and durations are fixed to $d = d(v)$ for all $v \in V$. Then the following statements hold:*

(*i*) *If $v \notin C_{i\cdot(d+1)-d}$ for some $i \in \mathbb{N}$, then $v \notin C_j$ for all $j \geq i \cdot (d+1) - d$.*

(*ii*) *If $v \in C_{i\cdot(d+1)}$ for some $i \in \mathbb{N}$, then $v \in C_j$ for all $j \geq i \cdot (d+1)$.*

Proof. We prove the claim by induction on $k := \left\lceil \frac{T}{d+1} \right\rceil$. For $k = 1$, we have at most $d+1$ rounds, so statement (*ii*) trivially holds. For statement (*i*), we have to show that nodes in $V \setminus C_1 = V \setminus B_1$ never buy in any round. If $v \in V \setminus B_1$, we have $p_\emptyset(v) = p(v) < \pi$. But since all influences are negative, this implies that $p_C(v) \leq p_\emptyset(v) = p(v) < \pi$ for all $C \subseteq V$, so v will indeed never buy.

For the induction step, note that, by the above argument, nodes in $V \setminus C_1 = V \setminus B_1$ never buy in any round, so statement (*i*) holds for $i = 1$. Moreover, we can remove all nodes in $V \setminus B_1$ from G without changing the values or the buying behavior of the other nodes. The nodes in $C_1 = B_1$ first decide about whether to rebuy or not in round $d+1$.

In round $d+1$, all negative influences from the nodes in B_1 are still present, so as the nodes in $V \setminus B_1$ were removed as they never buy, the current influenced value $p_{B_1}(v)$ of any node v can never be decreased further in later rounds. Thus, all nodes in $B_{d+1} = C_{d+1}$ will always continue renewing their purchase without breaking until the end, so statement (*ii*) holds for $i = 1$.

Moreover, the influences exerted by the nodes in B_{d+1} can be permanently added to the values of all nodes from round $d+1$ onwards. In total, this shows that the situation in rounds $d+2, \ldots, T$ is equivalent to the situation in rounds $1, \ldots, T' := T - (d+1)$ in the graph $G' := G - ((V \setminus B_1) \cup B_{d+1})$, where the initial value of each node v is given as

$p'(v) := p(v) + \sum_{u \in B_{d+1}} w(u,v)$. Hence, as $\left\lceil \frac{T'}{d+1} \right\rceil = \left\lceil \frac{T}{d+1} \right\rceil - 1 = k-1$, statements (i) and (ii) for $i \geq 2$ now follow by applying the induction hypothesis to G'. □

Lemma 6.4 and its proof show that, for negative influences and fixed durations, the selling process proceeds in *periods* consisting of $d+1$ rounds each. Within each period, purchases are only made in the first and the last round of the period. The nodes that do not buy in the first round of a period never buy again in any later rounds, while the nodes that buy in the last round of a period will always continue renewing their purchase without breaking until the end from this point in time.

In particular, this implies that the selling process "stabilizes" with the same set of nodes buying in the first round of each future period and no nodes buying in the last round of each period once the set of nodes buying in the last round of a period is empty for the first time. These observations are exploited in Algorithm Stint.

```
Algorithm 11: Stint($G, \pi, T$)
Data: A graph $G = (V, A)$ with initial values $p$, negative influences $w$, fixed durations
      $d(v) = d$, price $\pi$ and time horizon $1, \dots, T$.
Result: The revenue $R(\pi, T)$ generated in $G$.
Initialize $\underline{p}(v) = p_{V \setminus \{v\}}(v)$ and $\overline{p}(v) = p(v)$, $V' = V$, $R = 0$.
// $\underline{p}(v)$ and $\overline{p}(v)$ are the minimum and maximum influenced values,
   respectively
for $i = 1, \dots, \left\lceil \frac{T}{d+1} \right\rceil$ do
    $\overline{S}_i = \{v \in V' : \overline{p}(v) \geq \pi\}$                // Nodes that can still buy
    for $(u,v) \in (V' \setminus \overline{S}_i, \overline{S}_i)$ do
        $\underline{p}(v) = \underline{p}(v) - w(u,v)$
    $V' = \overline{S}_i$
    $\underline{S}_i = \{v \in V' : \underline{p}(v) \geq \pi\}$               // Nodes that will always buy
    for $(u,v) \in (\underline{S}_i, V' \setminus \underline{S}_i)$ do
        $\overline{p}(v) = \overline{p}(v) + w(u,v)$
    $V' = V' \setminus \underline{S}_i$
    $R = R + \pi \cdot |\overline{S}_i| + \pi \cdot \left\lceil \frac{T - i \cdot (d+1) + 1}{d} \right\rceil \cdot |\underline{S}_i|$
    if $\overline{S}_i = \overline{S}_{i-1}$ and $\underline{S}_i = \emptyset$ then
        $R = R + \pi \cdot \left\lceil \frac{T}{d+1} - i \right\rceil \cdot |\overline{S}_i|$
        return $R(\pi, T) = R$
```

Theorem 6.5. *For negative graphs and fixed durations, Algorithm Stint computes the revenue $R(\pi, T)$ in time $\mathcal{O}(m)$.*

Proof. Correctness follows immediately from Lemma 6.4 and the discussion afterwards.

The algorithm has at most n iterations of the **for**-loop and, except for the initial computation of $\underline{p}(v)$, iterates at most once over each arc. When we keep book of $\underline{S}_i$ and $\overline{S}_i$ while updating $\underline{p}(v)$ and $\overline{p}(v)$, all other operations can be done in at most $\mathcal{O}(n)$ time. □

For negative influences and *individual* durations, Lemma 6.4 does not hold. We show that it is in fact hard to compute $R(\pi, T)$ in this case.

Theorem 6.6. *PerPPAI$^\pi$ for negative graphs with individual durations is #P-hard.*

Proof. Let $\mathcal{J}$ be an instance of 3SAT, where $x_1, \ldots, x_{\tilde{n}}$ and $y_1, \ldots, y_{\tilde{m}}$ are the variables and clauses, respectively. From $\mathcal{J}$ we construct the graph G for instance $\mathcal{I}$ of PerPPAI$^\pi$ shown in Figure 6.3. Our instance $\mathcal{J}$ can be assumed to be restricted such that each clause contains literals corresponding to three different variables.

We can assume $\pi = 1$ as seen in Remark 6.3. After $T := 3 \cdot 2^n + 3$ rounds, the revenue $R(1, T)$ has summed up all purchases. Apart from the number of purchases of z_+, which we show to coincide with the number of satisfying truth assignments in $\mathcal{J}$, the revenue can be computed in polynomial time.

The initialization phase, rounds $1, \ldots, 2^{\tilde{n}}$, is needed in order to calibrate a regular buying behavior of the variable nodes x_i and $\overline{x}_i$ for $i = 1, \ldots, \tilde{n}$. Then, in the evaluation phase in rounds $2^{\tilde{n}} + 4, \ldots, T$, node z_+ starts to base its buying decision on whether the clause nodes $y_1, \ldots, y_{\tilde{m}}$ have purchased or not.

We set $p(v) = 1$ for all $v \in V$ and $w(a) = -1$ for all $a \in A$. Thus, all nodes buy in the first round and, afterwards, a node renews its purchase if and only if it is not influenced by any other node.

Variable Gadgets: The variable gadget G_i in Figure 6.1 generates two nodes "almost" alternately being consumers.

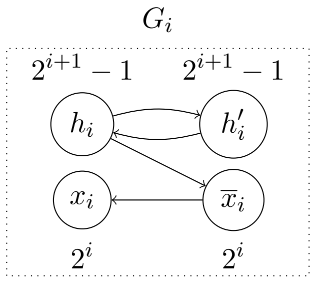

Figure 6.1: Variable Gadget G_i with durations of the nodes.

The nodes h_i and h'_i have the same duration $d = 2^{i+1} - 1$ and pause one round before renewing (cf. Lemma 6.4). Hence, they do not exert influence in rounds $k \cdot 2^{i+1} + 1$ for $k = 0, \ldots, \left\lfloor \frac{T}{2^{i+1}} \right\rfloor$.

As the node $\overline{x}_i$ is influenced exactly as h'_i, it buys at the same time with a duration of $d(\overline{x}_i) = 2^i$ rounds. Thus, $\overline{x}_i$ is a consumer in rounds $k \cdot 2^{i+1} + 1, \ldots, k \cdot 2^{i+1} + 2^i$ for $k = 0, \ldots, \left\lfloor \frac{T}{2^{i+1}} \right\rfloor$ (see Figure 6.2 for a complete illustration of $C_1, \ldots, C_T$ for $\tilde{n} = 3$).

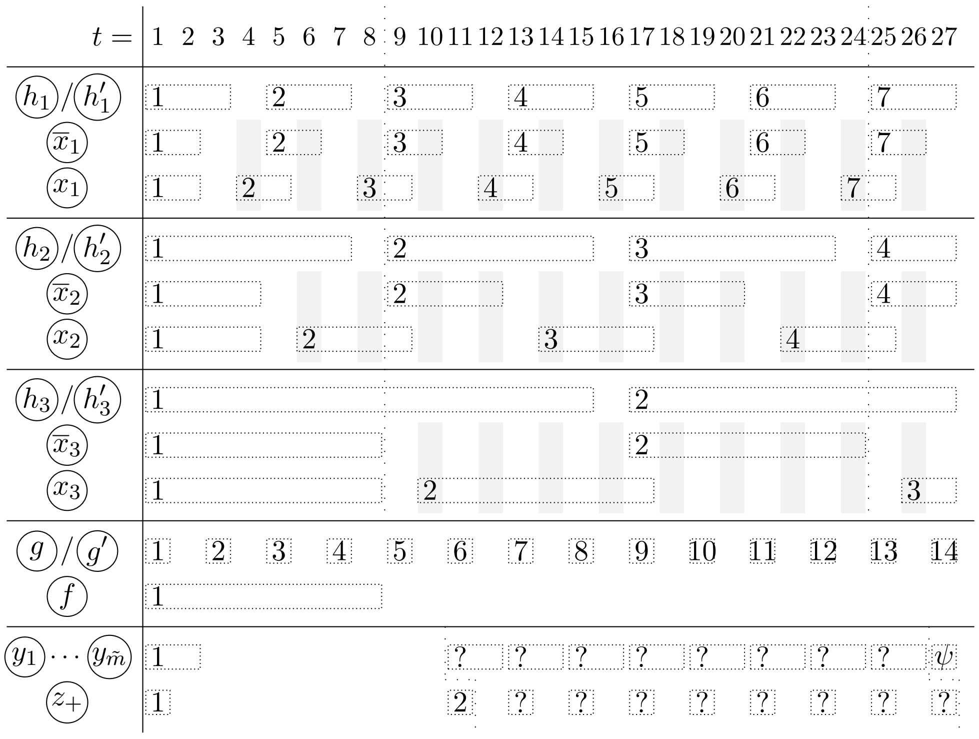

Figure 6.2: Consumer sets C_t in G with three variables

Due to the negative influence of $\overline{x}_i$, node x_i with duration $d(x_i) = 2^i$ buys once in the first round and, afterwards, always two rounds after $\overline{x}_i$ ceased to be a consumer. Hence, the consumer behavior of rounds $2^i + 1, \ldots, 2^i + 2^{i+1}$, the first period, is repeated every 2^{i+1} rounds. In particular, exactly $2^{\tilde{n}-i}$ periods occur in the rounds $2^{\tilde{n}}+1, \ldots, 2^{\tilde{n}}+2^{\tilde{n}+1}$.

In total, the revenue of gadget G_i for $1 \leq i < \tilde{n}$ is

$$R_{G_i}(1,T) = 4 \cdot \left\lceil \frac{T}{2^{i+1}} \right\rceil = 3 \cdot 2^{\tilde{n}-i+1} + 4.$$

The gadget $G_{\tilde{n}}$, while defined just as the the other gadgets, has a different behavior on rounds $1, \ldots, T$ and is therefore investigated individually. It generates a revenue of $R_{G_{\tilde{n}}}(1,T) = 9$ (see Figure 6.2 for x_3 and $\overline{x}_3$) and the roles of $\overline{x}_{\tilde{n}}$ and $x_{\tilde{n}}$ are switched.

Supplementary and Clause Nodes: For a gadget G_i, we say that the *k-th period* consists of rounds $k \cdot 2^{i+1} - 2^i + 1, \ldots, (k+1) \cdot 2^{i+1} - 2^i$. Each period starts with an odd round in which neither variable node is a consumer. The odd round $k \cdot 2^{i+1} + 1$ in the middle of the period has both variable nodes as consumers. Apart from those two rounds, always exactly one of x_i and $\overline{x}_i$ owns the product. In particular, after 2^i rounds, exactly one variable node is a consumer in each even round (marked gray in Figure 6.2).

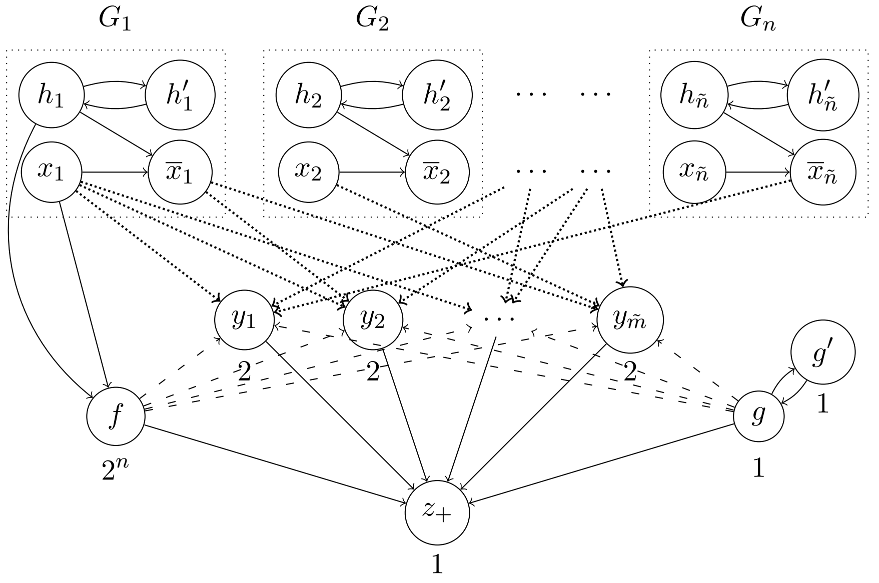

Figure 6.3: Reduction Graph for $\mathsf{PerPPAI}^{\pi}$ obtained from the $\mathsf{3SAT}$ instance

Altogether, the gadgets $G_1, \ldots, G_{\tilde{n}}$ run through all truth assignments exactly once in the even rounds $2^{\tilde{n}} + 2, 2^{\tilde{n}} + 4, \ldots, 3 \cdot 2^{\tilde{n}} = T - 3$.

The variable nodes influence their corresponding clause nodes $y_1, \ldots, y_{\tilde{m}}$. An edge (x_i, y_j), respectively $(\overline{x}_i, y_j)$, exists if and only if variable x_i is contained unnegated (respectively negated) in clause y_j. Thus, in the odd rounds after round $2^{\tilde{n}}$, node y_j buys if and only if the corresponding clause is *not* satisfied by the variable node consumers of the previous round.

We use node f to discourage the clause nodes from buying in rounds $2, \ldots, 2^{\tilde{n}} + 1$ and node g to do the same in all even rounds. Together, the three nodes f, g, and g' contribute $T + 2$ purchases.

The rounds in question (marked with "?" in Figure 6.2) yield exactly $\tilde{m} \cdot 1/8 \cdot 2^{\tilde{n}}$ purchases of clause nodes since, by the assumed structure of the $\mathsf{3SAT}$ instance (each clause contains literals corresponding to three different variables), a clause is *not* satisfied in one of eight assignments. In the last round, some clause nodes buy again, say ψ times in total. We can compute ψ easily by checking how many clauses are satisfied for one specific truth assignment.

So far, we have a well-known amount of purchases:

$$\begin{aligned} S &:= \underbrace{\sum_{i=1}^{\tilde{n}-1}(3\cdot 2^{\tilde{n}-i+1}+4)+9}_{\text{gadgets}} + \underbrace{3\cdot 2^{\tilde{n}}+5}_{f,g,g'} + \underbrace{\tilde{m}\cdot(2^{\tilde{n}-3}+1)+\epsilon}_{y_1,\ldots,y_{\tilde{m}}} \\ &= 9\cdot 2^{\tilde{n}} + \tilde{m}\cdot 2^{\tilde{n}-3} + m + \psi + 4\tilde{n} - 2 \end{aligned}$$

The collecting node z_+ buys in the first round and once in round $2^{\tilde{n}}+3$ when f stopped influencing and the clause nodes did not start again. In the following odd rounds, z_+ buys if and only if no clause node bought (is `false`). Thus, $R(1,T)-S-2$ is exactly the amount of satisfying truth assignments of the given 3SAT instance. □

6.1.3 Arbitrary Influences

The hardness result of Theorem 6.6 uses only negative influences but individual durations at the nodes. We now prove that the problem is even hard for unit durations if one allows arbitrary influences.

Theorem 6.7. *PerPPAI$^\pi$ with arbitrary influences and unit durations is #P-hard.*

Proof. We assume without loss of generality that $\pi = 1$ as noted in Remark 6.3. Given an instance $\mathcal{J}$ of 3SAT, i.e., variables $x_1,\ldots,x_{\tilde{n}}$ and clauses $y_1,\ldots,y_{\tilde{m}}$, we construct a graph for instance $\mathcal{I}$ such that the obtained revenue $R(\pi,T)$ — which equals the number of purchases as $\pi = 1$ — counts the satisfying truth assignments.

We construct variable nodes $x_1,\overline{x}_1,\ldots,x_{\tilde{n}},\overline{x}_{\tilde{n}}$ such that all configurations occur in rounds $\tilde{n},\ldots,\tilde{n}+2^{\tilde{n}}-1 = T-2$. Lacking longer (native) durations, we need to generate well-synchronized periods of lengths 2^i with only poly($\tilde{n}$) nodes.

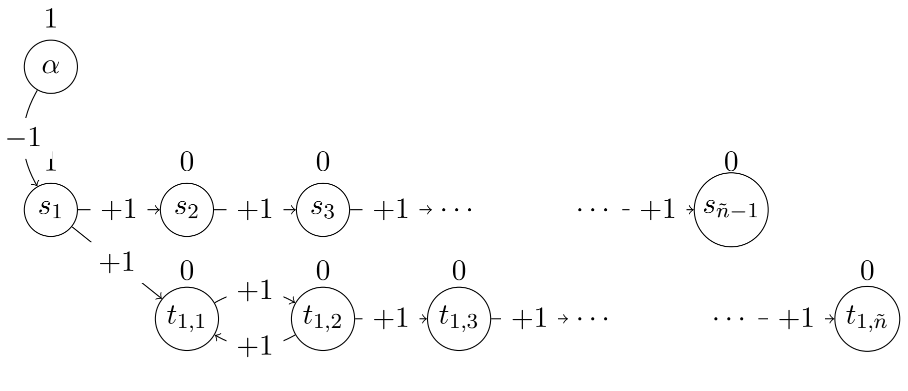

Figure 6.4: Construction of $s_{\tilde{n}-1}$, $g_1 := t_{1,1}$, and $t_1 := t_{1,\tilde{n}}$

Property 6.1.1. The node α in Figure 6.4 (labels above nodes denote initial values) buys each round and influences s_1 not to buy after the first round. Each node s_i for $i = 2, \dots, \tilde{n} - 1$ buys exactly in round i (and in no other round). While the s-chain is only traversed once, the nodes $t_{1,1}, \dots, t_{1,\tilde{n}}$ repeat buying every two rounds because $t_{1,1}$ and $t_{1,2}$ "keep them alive". Node $t_{1,1}$ buys in rounds $2, 4, \dots, n + 2^{\tilde{n}}$, and node $t_{1,\tilde{n}}$ buys in rounds $\tilde{n} + 1, \tilde{n} + 3, \dots, n + 2^{\tilde{n}} + 1$.

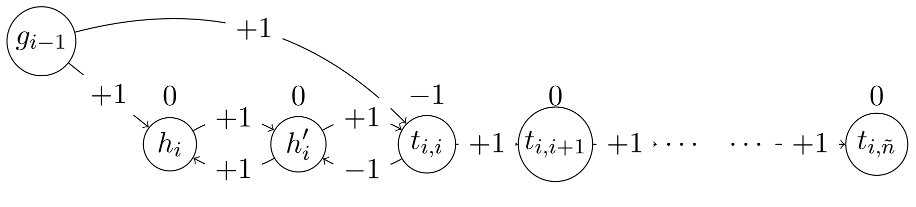

Figure 6.5: Construction of further g_i and t_i

Property 6.1.2. Assuming g_{i-1} buys in rounds $(i-2)+2^{i-1}, (i-2)+2\cdot 2^{i-1}, (i-2)+3\cdot 2^{i-1}, \dots,$ node $t_{i,i}$ buys in rounds $(i-1)+2^i, (i-1)+2\cdot 2^i, (i-1)+3\cdot 2^i, \dots$ and node $t_{i,n}$ always $n - i$ rounds later.

Proof. Node h_i buys for the first time in round $(i-2)+2^{i-1}+1$. Like the nodes $t_{1,1}$ and $t_{1,2}$, the nodes h_i and h'_i buy alternatingly as long as they are not influenced by some other node. When the second activation of g_{i-1} happens in round $(i-2)+2\cdot 2^{i-1}$, the node h'_i buys as well. The influenced values in the following round are as follows:

$$p_B(h_i) = 2, \quad p_B(h'_i) = 0, \quad p_B(t_{i,i}) = 1.$$

Hence, as claimed, h_i and $t_{i,i}$ buy in round $(i-1)+2^i$. In the next round, the influences of h_i and $t_{i,i}$ on h'_i cancel each other and none of these nodes buy. Thus, we are in the same situation as before and g_{i-1} starts the whole process anew in round $(i-2)+3\cdot 2^{i-1}$, namely 2^i rounds after the initial start.

The node $t_{i,\tilde{n}}$ repeats the behavior of $t_{i,i}$ exactly $\tilde{n} - i$ rounds later, as seen for the nodes in Figure 6.4. □

Property 6.1.3. For $i = 2, \dots, \tilde{n}$ and given the proper buying behavior shown in the previous properties, the "outer" influences on the variable nodes x_i and $\overline{x}_i$ are summarized in Table 6.1.

	$\tilde{n}+0\cdot 2^{i-1}$	$\tilde{n}+1\cdot 2^{i-1}$	$\tilde{n}+2\cdot 2^{i-1}$	$\tilde{n}+3\cdot 2^{i-1}$	$\dots$
$\overline{x}_i$	+1	−1	+1	−1	± 1
x_i	0	+1	−1	+1	∓ 1

Table 6.1: Summary of the influences $w(s_{\tilde{n}-1}, \cdot) + w(t_{i-1}, \cdot) + w(t_i, \cdot)$ exerted on the variable nodes

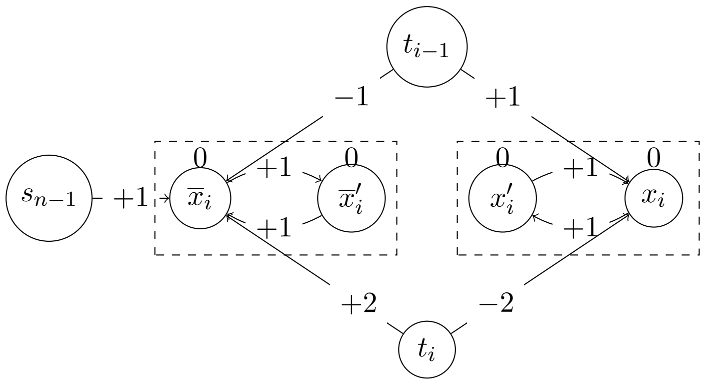

Figure 6.6: Variable Gadget

The corresponding nodes start buying (+1) or stop buying (−1) in those rounds. With their respective duplicate nodes x'_i and $\overline{x}'_i$, the buying is kept alive in between those specific rounds, i.e., in the following rounds either x_i or x'_i are consumers:

$$\tilde{n}, \dots, \tilde{n} + 2^{i-1} - 1, \tilde{n} + 2^i, \dots, \tilde{n} + 2^i + 2^{i-1} - 1, n + 2 \cdot 2^i, \dots$$

All (future) outgoing arcs of the variable nodes are copied to stem from both the original and duplicate nodes.

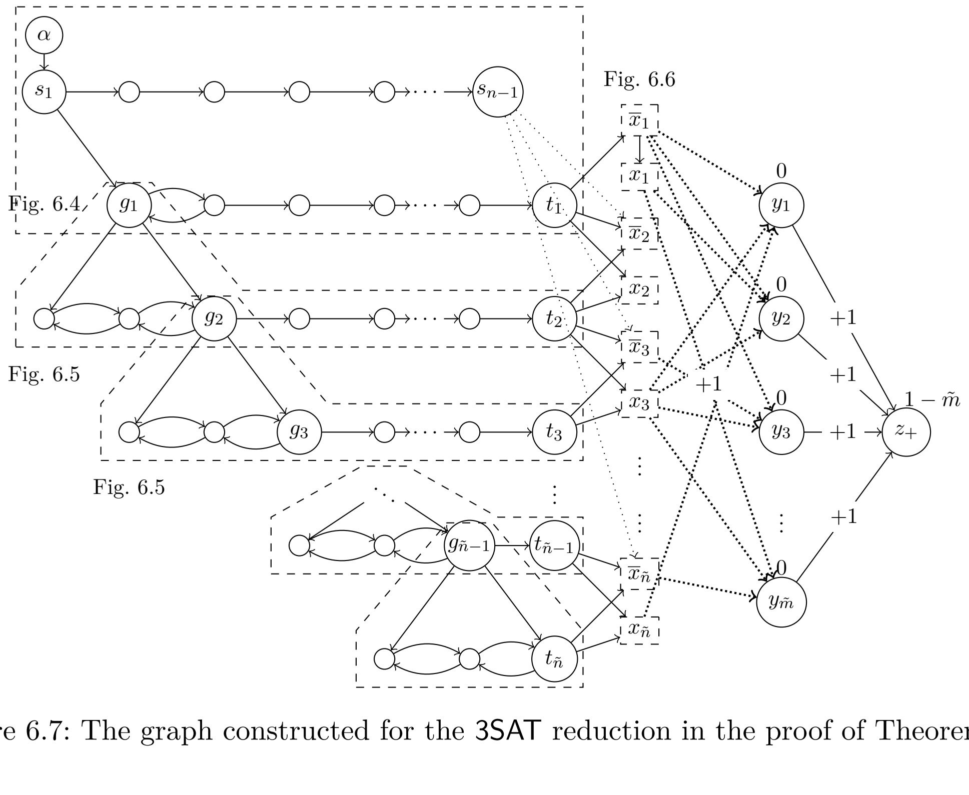

Figure 6.7: The graph constructed for the 3SAT reduction in the proof of Theorem 6.7

Property 6.1.4. The clause nodes $y_1, \dots, y_{\tilde{m}}$ are influenced by their respective variable nodes contained (see Figure 6.7). They buy if and only if at least one of these variable nodes did so in the previous round. The collecting node z_+ buys if and only if all clause nodes are consumers at the same time. This happens exactly when the configuration of the variable nodes two rounds before corresponds to a satisfying assignment.

This concludes the #P-hardness proof. □

In Figures 6.8 and 6.9 the status in round $\tilde{n}$ and $\tilde{n} + 2^{\tilde{n}} - 1$, respectively, is illustrated. In these figures the current buyers are marked gray, their influenced value is at top left and the amount of purchases at bottom right.

6.1.4 Acyclic Graphs

In this section, we study the problem PerPPAI$^{\pi}$ on acyclic graphs and, in addition to more structural insights, derive polynomial time algorithms for two special cases (see Lemma 6.9 and Corollary 6.10).

Lemma 6.8. *If $G = (V, A)$ is acyclic, no node breaks after round $\sum\limits_{v \in V} d(v)$.*

Proof. We call a node $v \in V$ a *source* if it has in-degree 0, i.e., if there are no arcs entering v. We define the graph

$$G^d = (V \cup \{s\}, A \cup \{(s, v) : \text{ } v \text{ is a source in } G\})$$

with arc lengths $c(u, v) := d(v)$ for $(u, v) \in A$ and $c(s, v) := 1$ for all sources v in G. We show that no node $v \in V$ breaks after round $L(v)$, where $L(v)$ is the length of the longest path from s to v in G^d with respect to the arc lengths c. Note that, in graphs with cycles, the longest non-simple path may be of infinite length. This is where we make use of the fact that G is acyclic.

Let $v_1, \dots, v_n$ be a topological sorting of G. The values $L(v_i)$ satisfy the recursion $L(v_i) = 1$ for all sources v_i in G and

$$\begin{aligned} L(v_i) &= \max\{\, L(u) + c(u, v_i) : u \in N^-(v_i) \,\} \\ &= d(v_i) + \max\{\, L(u) : u \in N^-(v_i) \,\}, \end{aligned} \tag{6.1}$$

for $N^-(v_i) \neq \emptyset$, where we used the fact that all arcs entering v_i have the same length $d(v_i)$ in order to obtain the last equality.

All sources v_i are never influenced by any other node and, consequently, will not break after round $1 = L(v_i)$. Now consider a node v_i such that $N^-(v_i) \neq \emptyset$. This node is influenced by the nodes in $N^-(v_i) \subseteq \{v_1, \dots, v_{i-1}\}$. By induction, each node $u \in N^-(v_i)$ does not break after round $L(u)$. By (6.1.4), the longest path to v_i goes through node

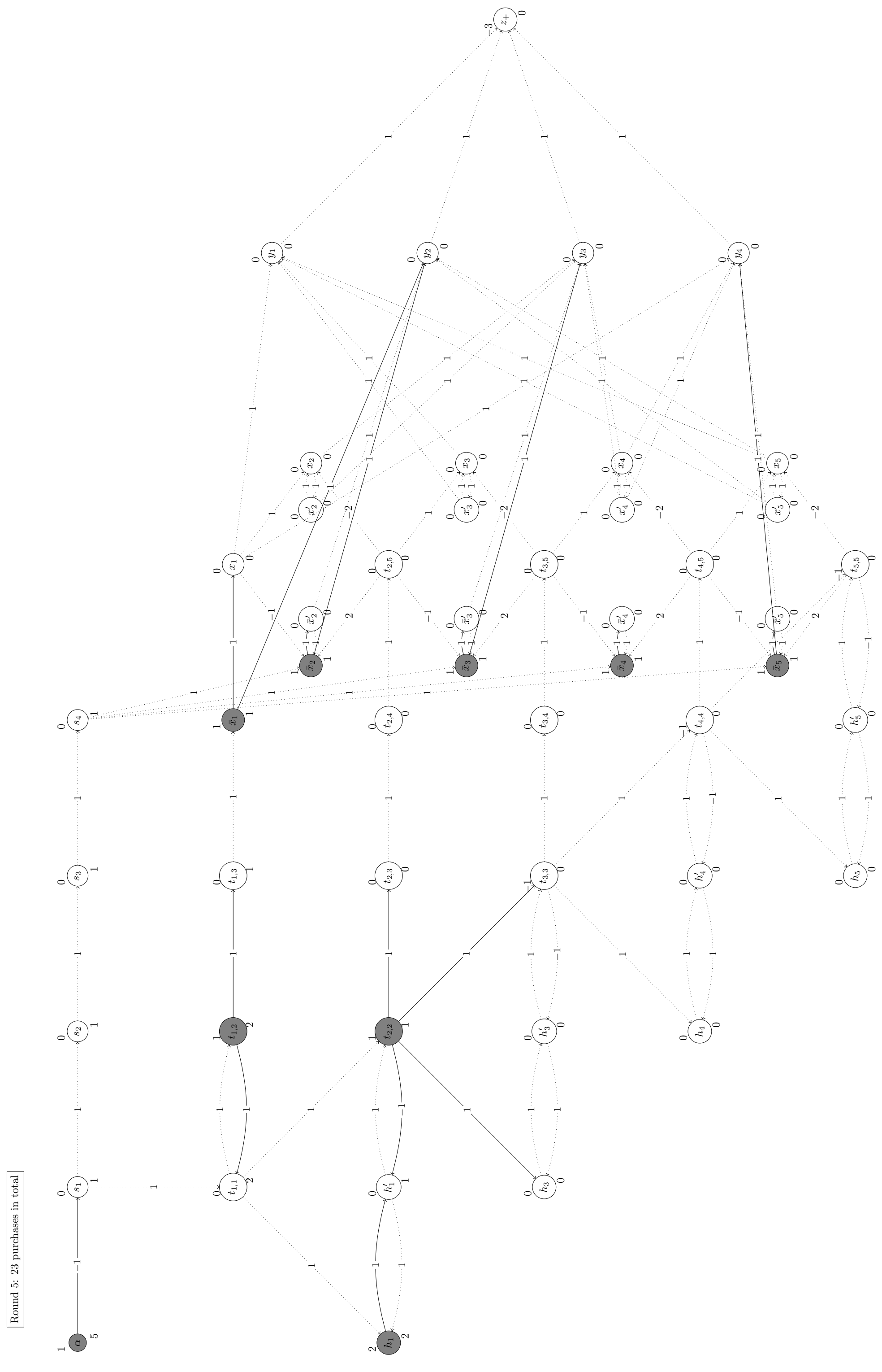

Figure 6.8: Snapshot of the selling: first round

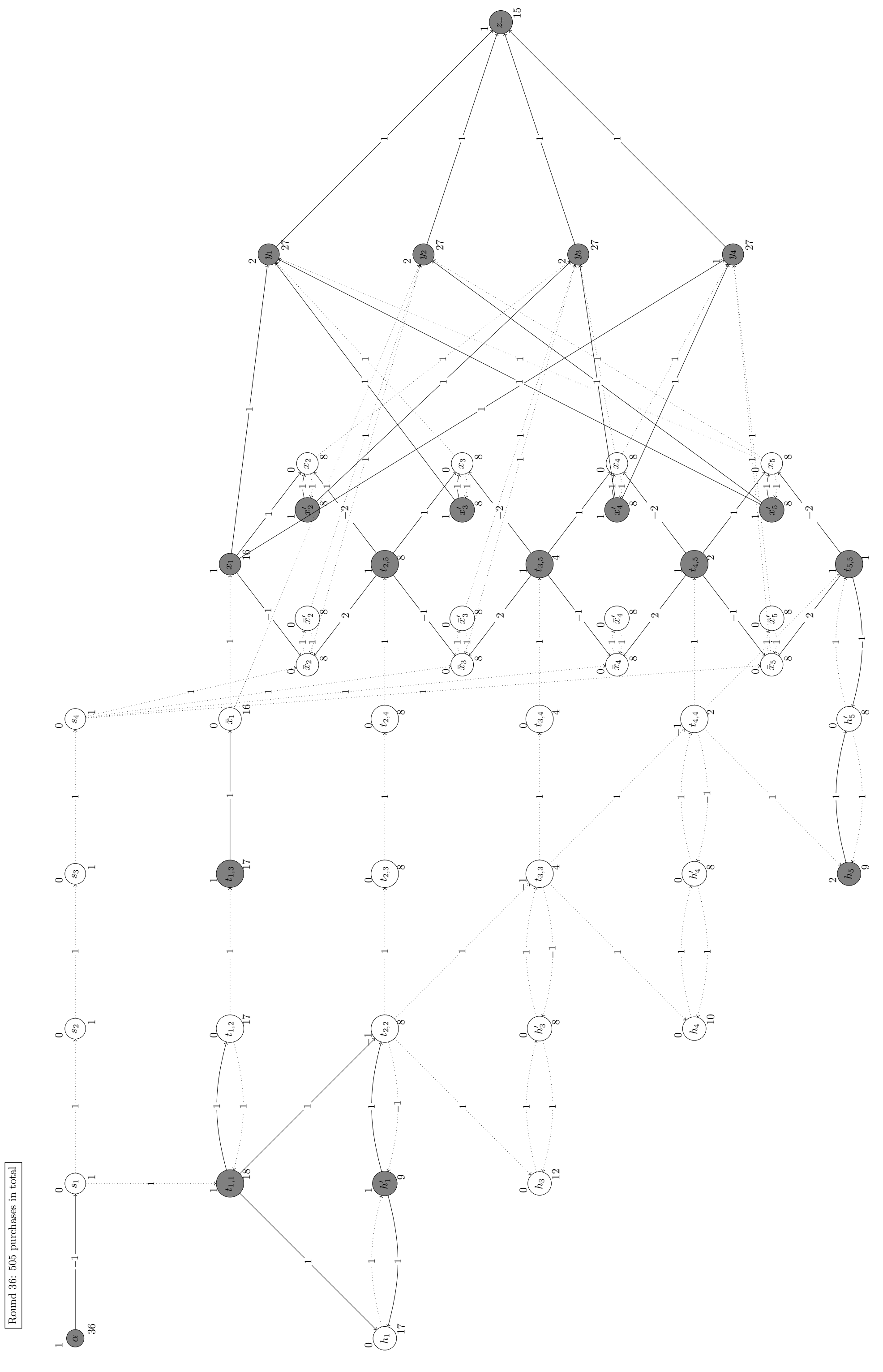

Figure 6.9: Snapshot of the selling: last round

$v^* = \text{argmax}_{v_j \in N^-(v_i)} L(v_j)$. The influenced value of v_i can change for the last time between rounds $L(v^*)$ and $L(v^*) + 1$. Hence, if v_i is no consumer in round $L(v^*)$, it can break (to being a consumer) no later than in round $L(v^*) + 1$. Otherwise, the longest delay is obtained when v_i decides to buy in round $L(v^*)$ and breaks (to being no consumer) in round $L(v^*) + d(v_i) = L(v_i)$. The claim of the lemma now follows from (6.1.4) and the fact that $d(v) \geq 1$ for all $v \in V$. □

Lemma 6.9. *For an acyclic graph $G = (V, A)$, let $d_{\min} := \min\limits_{v \in V} d(v)$ and $d_{\max} := \max\limits_{v \in V} d(v)$ denote the shortest and longest duration of a node, respectively. Then, if $d_{\max}/d_{\min} \leq \text{poly}(n)$ is polynomially bounded in n, Algorithm* Break *runs in polynomial time.*

Proof. By Lemma 6.8, after no more than $\sum\limits_{v \in V} d(v) \leq n \cdot d_{\max}$ rounds, no breaks will occur. As a node v may only break twice in a period of $d(v) + 1$ rounds, no node breaks more than $\mathcal{O}(n \cdot d_{\max}/d_{\min})$ times. Thus, we have $\beta \in \mathcal{O}\left(n^2 \cdot d_{\max}/d_{\min}\right)$ and the claim follows from the assumption and Lemma 6.1. □

For fixed durations, we have $d_{\max}/d_{\min} = 1$, so we obtain the following corollary:

Corollary 6.10. PerPPAI$^\pi$ *on acyclic graphs with fixed durations can be solved by* Break *in $\mathcal{O}\left(n^2 \cdot m\right)$ time.*

The next theorem shows that, if the assumption on the fixed durations is dropped, the problem becomes hard again:

Theorem 6.11. PerPPAI$^\pi$ *on acyclic graphs with individual durations and arbitrary influences is #P-hard.*

Proof. Let $\mathcal{J}$ be an instance of 3SAT where $x_1, \ldots, x_{\tilde{n}}$ and $y_1, \ldots, y_{\tilde{m}}$ are the variables and clauses respectively. As in the proof of Theorem 6.6, we construct a graph G for instance $\mathcal{I}$ of PerPPAI such that the revenue obtained for price $\pi = 1$ is

$$R(1, T = 2^{\tilde{n}+1} + 2) = 2^{\tilde{n}} \cdot (3 + \frac{7\tilde{m}}{8}) - 1 + X,$$

where X is the number of satisfying truth assignments. The graph consists of the following set of nodes:

$$V = \{\alpha, \alpha', x_{\tilde{n}}, x'_{\tilde{n}}, \overline{x}_{\tilde{n}}, \ldots, x_1, x'_1, \overline{x}_1, y_1, \ldots, y_{\tilde{m}}, z_+\}.$$

The initial values, durations, and influences are specified as follows:

$$p(\alpha) = 1, \quad d(\alpha) = T$$

$$\begin{aligned}
p(\alpha') &= 1, & d(\alpha) &= 2^{\tilde{n}+1} - 1 & & \\
p(x_i) &= 1, & d(x_i) &= 2^i & & \text{for } i = 1, \dots, \tilde{n} \\
p(x'_i) &= 1, & d(x_i) &= 2^i - 1 & & \text{for } i = 1, \dots, \tilde{n} \\
p(\overline{x}_i) &= 0, & d(\overline{x}_i) &= 2^i & & \text{for } i = 1, \dots, \tilde{n} \\
p(y_i) &= 0, & d(y_i) &= 2 & & \text{for } i = 1, \dots, \tilde{m} \\
p(z_+) &= 1 - \tilde{m}, & d(z_+) &= 2 & &
\end{aligned}$$

$$\begin{aligned}
w(\alpha, v) &= -1 & & \text{for } v \in \{\alpha', x_{\tilde{n}}, x'_{\tilde{n}}, \overline{x}_{\tilde{n}}, \dots, x_1, x'_1, \overline{x}_1\} \\
w(\alpha', v) &= +1 & & \text{for } v \in \{x_{\tilde{n}}, x'_{\tilde{n}}, \overline{x}_{\tilde{n}}, \dots, x_1, x'_1, \overline{x}_1\} \\
w(x_i, \overline{x}_i) &= +1 & & \text{for } i = \tilde{n}, \dots, 1 \\
w(x'_i, \overline{x}_i) &= -1 & & \text{for } i = \tilde{n}, \dots, 1 \\
w(x'_{i+1}, x_i) = w(x'_{i+1}, x'_i) &= -1 & & \text{for } i = \tilde{n} - 1, \dots, 1 \\
w(\overline{x}_{i+1}, x_i) = w(\overline{x}_{i+1}, x'_i) &= -1 & & \text{for } i = \tilde{n} - 1, \dots, 1 \\
w(x_i, y_j) &= +1 & & \text{if } x_i \in y_j \\
w(\overline{x}_i, y_j) &= +1 & & \text{if } \overline{x}_i \in y_j \\
w(y_i, z_+) &= +1 & & \text{for } i = 1, \dots, \tilde{m}
\end{aligned}$$

Note that the given order of nodes is a topological sorting, so the graph is acyclic. The buying behavior of the nodes in G is illustrated in Figure 6.10. As in our previous #P-hardness proofs, all truth assignments are traversed once (or twice as in this case). We sum up the amount of purchases in detail. The claim then follows by the same arguments.

Number of purchases: Nodes α and α' buy exactly once. The variable nodes stop buying after $2^{\tilde{n}+1}$ rounds, because the influences from α and α' are negative from then on. In rounds $1, \dots, 2^{\tilde{n}}$ the variable nodes x_i, x'_i and $\overline{x}_i$ are buying exactly $2^{\tilde{n}-i}$ times each. This amounts to a total of $3 \cdot (2^{\tilde{n}} - 1)$ purchases.

The clause nodes need to be influenced by one of their variable nodes. Assuming the clauses contain three different variables, they are satisfied for all but $\frac{1}{8}$ the time. This amounts to a total of $\frac{7}{8} \cdot \tilde{m} \cdot 2^{\tilde{n}}$ purchases for the clause nodes.

□

By Theorems 6.6, 6.7, and 6.11, we obtain that the general (arbitrary influences, individual durations) problem $\mathsf{PerPPAI}^{\pi}$ is #P-hard. We summarize the complexity results of this section in Table 6.2 (where "+" and "-" denote positive and negative graphs, respectively).

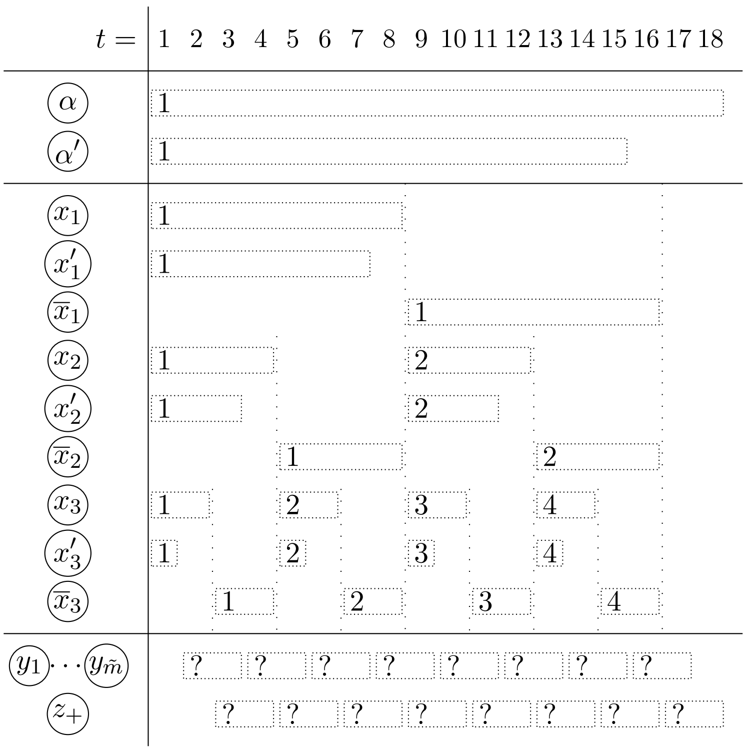

Figure 6.10: Consumer sets C_t in G with three variables

	unit/fixed duration	individual duration
$+$	P	P
$-$	P	#P-hard
acyclic	P	#P-hard
$\pm$	#P-hard	#P-hard

Table 6.2: Complexity results for different versions of PerPPAI$^{\pi}$

6.2 Complexity of PerPPAI

We only investigate the settings of PerPPAI for which PerPPAI$^{\pi}$ is solvable in polynomial time.

However, we can recycle parts of the results of Section 3.4.

Lemma 6.12. *PerPPAI with individual or fixed durations is NP-hard.*

Proof. For individual and fixed durations we can set $d(v) \geq T$. Thus we deny any breaks with longer durations and the reduction of Theorem 3.9 works again. $\square$

Note that this does not work for unit durations.

Graphs with positive influences have a monotonicity in the rounds by Lemma 6.2. We now show that there is also a monotonicity with respect to the prices:

Lemma 6.13. *For positive influences, it holds that $C_t(\pi') \supseteq C_t(\pi)$ for all prices $\pi' \leq \pi$ and all rounds $t = 1, \ldots, T$.*

Proof. The claim is obviously true for $t = 1$. Assume that round t is the first round in which the claim is false. Then, for some node $v \in V$, we have $p_{C_{t-1}(\pi')}(v) < \pi \leq p_{C_{t-1}(\pi)}(v)$. This is a contradiction, since by definition of t, we have $C_{t-1}(\pi') \supseteq C_{t-1}(\pi)$ and all influences are positive. □

In positive graphs, all nodes always renew without breaking (Lemma 6.2). Thus, after round n, there will be no new buyers/consumers. For $t = 1, \ldots, n$, let $p^*_t(v) \coloneqq \max\{\, \pi : v \in C_t(\pi) \,\}$ denote the maximum price for which node v is a consumer in round t. By Lemma 6.13, we know that it suffices to check all prices in $\Pi = \{p^*_t(v) : t = 1, \ldots, n \text{ and } v \in V\}$. We have $|\Pi| \in \mathcal{O}\left(n^2\right)$ and can compute all the values in Π in time $\mathcal{O}\left(nm\right)$.

This immediately leads to a polynomial time algorithm for PerPPAI on positive graphs: Compute the set Π and, for each $\pi \in \Pi$, compute the revenue $R(\pi, T)$ in time $\mathcal{O}\left(m\right)$ by the algorithm from Corollary 6.3. Then, pick the price from Π generating the largest revenue. This yields:

Theorem 6.14. *On positive graphs, PerPPAI can be solved in polynomial time.*

In contrast, the problem is hard if there are only negative influences, even for acyclic graphs and even in the case of unit durations.

Theorem 6.15. *PerPPAI is NP-complete for negative acyclic graphs with unit durations.*

Proof. Given an instance $\mathcal{J}$ of 3SAT consisting of the variables $x_1, \ldots, x_{\tilde{n}}$ and the clauses $y_1, \ldots, y_{\tilde{m}}$, we construct an instance of PerPPAI on the graph G shown in Figure 6.11 with time horizon $T \coloneqq 6 \cdot (\tilde{n} + 5)$ and define a requested revenue of $R \coloneqq 40 \cdot 2^{\tilde{n}} \cdot (\tilde{n} + \tilde{m} + 1) \cdot (\tilde{n} + 5)$.

For each variable x_i, the graph G contains two nodes x_i and $\overline{x}_i$ corresponding to the positive and negated literal associated with the variable, which are connected by an arc $(\overline{x}_i, x_i)$. Additionally, there is an arc $(\overline{x}_i, \overline{x}_j)$ whenever $i < j$. Moreover, for each clause y_j, there are two nodes y_j and $\overline{y}_j$ connected by an arc $(\overline{y}_j, y_j)$. If clause y_j contains the literals l_1, l_2, l_3, then there are arcs from each of the literal nodes l_i to $\overline{y}_j$. Moreover, there is a node z_- that influences all other nodes and a "collecting node" z_+ that is influenced by all nodes y_i and influences a large collection $z_1, \ldots, z_M$ of nodes that have out-degree zero.

We define the price range $\Pi := \{\pi_{\min} := 2^{\tilde{n}}, \dots, \pi_{\max} := 2^{\tilde{n}+1} - 1\}$ of meaningful prices in our instance of PerPPAI. The initial values of the nodes in G are defined as follows:

$$\begin{aligned}
p(z_-) &= \pi_{\min} - 1 && \\
p(\overline{x}_i) &= \pi_{\max} - 2^{\tilde{n}-i} && \text{for } i = 1, \dots, \tilde{n} \\
p(z_+) &= \tilde{m} \cdot \pi_{\max} && \\
p(v) &= \pi_{\max} && \text{for } v \in V \setminus \{z_-, \overline{x}_1, \dots, \overline{x}_{\tilde{n}}, z_+\}
\end{aligned}$$

Except for the influences $w(\overline{x}_i, \overline{x}_j) := -2^{\tilde{n}-i}$ for all $1 \le i < j \le \tilde{n}$ and the influences $w(z_-, z_+) := -\tilde{m} \cdot \pi_{\max}$, all influences are set to $-\pi_{\max}$.

We transform each truth assignment to a price in the range Π as follows: The binary representation $(\pi)_2$ of a price $\pi \in \Pi$ has $\tilde{n} + 1$ digits and the leading digit is always 1. We now show that, starting from round $i+1$, node x_i ($\overline{x}_i$) buys permanently if and only if the $(i+1)$-th digit from the left in the binary representation of π is 1 (0). Thus, the prices in Π are in one-to-one correspondence with the truth assignments of the variables. Afterwards, we show that a price $\pi \in \Pi$ generates a revenue of at least R if and only if the corresponding truth assignment satisfies all clauses.

Variable Gadget: As the graph is acyclic, we obtain from argumentation used in the proof of Lemma 6.8 that node $\overline{x}_i$ does not break after round i and node x_i after round $i+1$, respectively. Since node $\overline{x}_i$ nullifies the influenced value of its partner x_i, for each price in Π, exactly one of the variable nodes x_i and $\overline{x}_i$ has to buy in all rounds $t \ge i + 1$.

For prices $\pi \in \Pi$ with $\pi \le p(\overline{x}_1) = 2^{n+1} - 2^{n-1} - 1 = 101 \cdots 1_2$, i.e., for all the prices in Π for which the second digit is 0, node $\overline{x}_1$ buys the product in every round. Starting with round 2, it thereby discourages x_1 from buying. Otherwise, node $\overline{x}_1$ never buys and the initial value $p(x_1) = \pi_{\max}$ of x_1 remains unchanged and x_1 buys in each round.

The initial values of all nodes $\overline{x}_i$ with $i > 1$ have a 1 as the second digit. For prices $\pi \in \Pi$ with $\pi > 2^{\tilde{n}+1} - 2^{\tilde{n}-1} - 1$, this remains unchanged and, hence, the second digit of π coincides with those of the influenced values of the variable nodes $\overline{x}_i$. Otherwise, the influences $w(\overline{x}_1, \overline{x}_i) = -2^{\tilde{n}-1}$ for $i = 2, \dots, \tilde{n}$ change all these 1s in the binary representation of the initial values to 0s. Thus, once again the same digit is present in the price π and the influenced values.

Inductively, it follows that the variable nodes buy exactly as specified by the binary representation of the price π.

Clause Gadget: As we have no positive influences to change a node's decision from not buying to buying, we have to substitute this by stopping the exertion of "default" negative influences from additional nodes (see Figure 6.11). The negation $\overline{y}_j$ of a clause

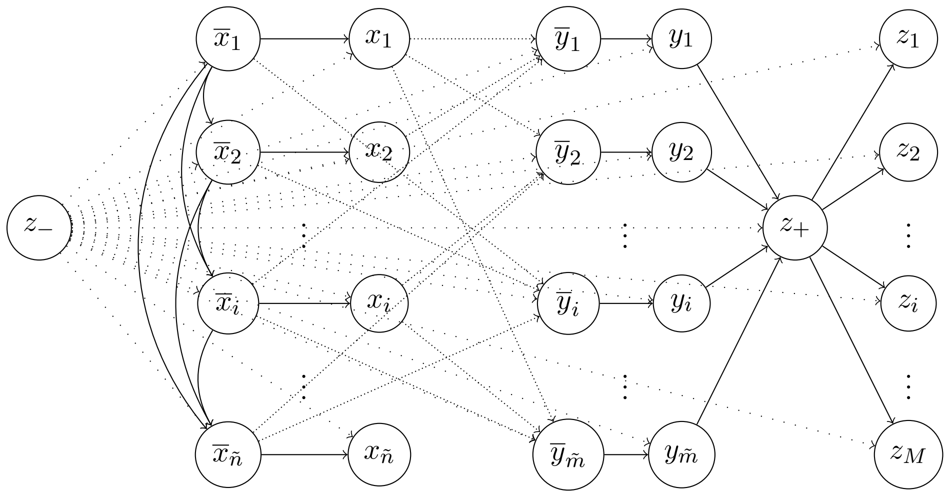

Figure 6.11: Graph used in the proof of Theorem 6.15

node y_j buys permanently from round $\tilde{n}+2$ onwards if and only if none of the corresponding variable nodes buy. The clause node y_j itself buys permanently from round $\tilde{n}+3$ onwards if and only if $\overline{y}_j$ does not buy. Exactly if all of the clause nodes buy permanently, the collecting node z_+ never buys from round $n+4$ onwards and its negative influence on the nodes $z_1,\dots,z_M$ (for $M := 8\cdot(n+m+1)$) ceases. All in all, the nodes $z_1,\dots,z_M$ buy permanently from round $\tilde{n}+5$ onwards if and only if the truth assignment corresponding to the price satisfies all clauses.

Revenue: We have a total of $|V| = 10\cdot(\tilde{n}+\tilde{m}+1)$ nodes that may buy at most $T = 6\cdot(\tilde{n}+5)$ times each.

- If $\pi > \pi_{\max}$, only node z_+ may buy and the revenue is at most $T\cdot\tilde{m}\cdot\pi_{\max} < 12\cdot 2^{\tilde{n}}\cdot(\tilde{n}+5)\cdot(\tilde{n}+\tilde{m}+1) < R$.
- If $\pi < \pi_{\min}$, node z_- buys in each round and prevents all other nodes from buying in all rounds $t \geq 2$. Thus, the revenue is at most $(|V|+T-1)\cdot(\pi_{\min}-1) < R$.
- If $\pi \in \Pi$ corresponds to a non-satisfying truth assignment, the nodes $z_1,\dots,z_M$ do not buy after round $\tilde{n}+5$ since z_+ always buys from round $n+4$ onwards. Thus, the revenue is at most $\pi_{\max}\cdot(|V|\cdot(\tilde{n}+5)+(2\tilde{n}+2\tilde{m}+2)\cdot 5\cdot(\tilde{n}+5)) < 2\cdot 2^{\tilde{n}}\cdot(10+10)\cdot(\tilde{n}+\tilde{m}+1)\cdot(\tilde{n}+5) = R$
- However, if $\pi \in \Pi$ corresponds to a satisfying truth assignment, the revenue is at least $\pi_{\min}\cdot M\cdot(T-\tilde{n}-5) = 2^{\tilde{n}}\cdot 8\cdot(\tilde{n}+\tilde{m}+1)\cdot 5\cdot(n+5) = R$

Thus, there exists a price π that generates at least the requested revenue of R if and only if there is a satisfying truth assignment for the instance of 3SAT. □

Corollary 6.16. *Unless* $\mathsf{P} = \mathsf{NP}$*, the above proof also shows that there is no* r*-approximate polynomial time algorithm for* $\mathsf{PerPPAI}^{\pi}$ *with* $r = \mathcal{O}(1)$ *and the restrictions as in Theorem 6.15. This can be established by increasing* M *and* T *in the above proof.*

For example, choosing $M \coloneqq 16 \cdot (\tilde{n} + \tilde{m} + 1)$ *and* $T \coloneqq 10 \cdot (\tilde{n} + 5)$ *shows that there is no* 2*-approximate algorithm in polynomial time unless* $\mathsf{P} = \mathsf{NP}$*.*

6.3 Integer Program

We formulate an integer program in the perishable setting, along the lines of Integer Program 1.

Integer Program 3: PerPPAI *(Formulation of* $\mathsf{PerPPAI}$*)*

$$\max \sum_{v \in V} \sum_{t=1}^{n} \pi \cdot x_{v,t}$$

subject to

$$\begin{aligned}
x_{v,t} &\geq 0 && \forall v \in V, t = 1, \dots, n \\
x_{v,t} &\leq 1 && \forall v \in V, t = 1, \dots, n \\
\sum_{i=t-\tau(v)+1}^{t} x_{v,t} &= y_{v,t} && \forall v \in V, t = 1, \dots, n \\
y_{v,t} &\leq 1 && \forall v \in V, t = 1, \dots, n \\
p(v) + \sum_{u \in \mathcal{N}^-(v)} y_{u,t} \cdot w(u,v) &= p_{v,t} && \forall v \in V, t = 1, \dots, n \\
p_{v,t} - \pi &\geq (x_{v,t} - 1) \cdot M && \forall v \in V, t = 1, \dots, n \\
p_{v,t} - \pi &\leq (x_{v,t} + y_{v,t}) \cdot M - 1 && \forall v \in V, t = 1, \dots, n
\end{aligned}$$

Remark 6.6. Whereas we provided the combinatorial algorithms Frag and bFrag for $\mathsf{PPAI}^{\text{opt}}$ and $\mathsf{DynPPAI}^{\text{opt}}$, respectively, we did not state one for $\mathsf{PerPPAI}^{\text{opt}}$ on arbitrary graphs.

Instead, the formulation PerPPAI is considered as our method to solve $\mathsf{PerPPAI}^{\text{opt}}$.

6.4 Return Option

Using the methods and results derived in this chapter, we can examine PPAI with a return option. The results of this section were obtained jointly with Sebastian Johann [Joh15].

Definition 6.7. The buying decision is reached as usual, i.e., a node $v \in V$ buys once $p_C(v) \geq \pi$. Once the influenced value drops below the price, they can return and claim a refund of $\alpha \cdot \pi$ for $0 \leq \alpha \leq 1$. The set of returners

$$D_t(\pi) := \left\{ v \in C_{t-1}(\pi) : p_{C_{t-1}(\pi)} < \pi \right\}$$

denotes the nodes that returned the product in round t, whereas $D_{\leq t}$ is the set of *returners* in the first t rounds and $D(\pi)$ the entirety of them.

These nodes may not rebuy in the standard model and their influence ceases in round $t+1$, i.e., $C_t = (C_{t-1} \cup B_t) \setminus D_t$. Together with the nodes $v \in N_t$ that have never bought this partitions $V = C_t \dot\cup D_{\leq t} \dot\cup N_t$.

The revenue for a fixed price π is

$$R(\pi) := \pi \cdot |C(\pi)| + (1 - \alpha) \cdot |D(\pi)| .$$

In the more general model each node is allowed to buy and return up to $\eta \in \mathbb{N}$ times.

Problem 5: PPAIηR *(Product Pricing with Additive Influences and Return Option)*

Instance: *A directed graph $G = (V, A)$, initial values $p(v) \in \mathbb{Z}$ for the nodes $v \in V$, influences $w(u, v) \in \mathbb{Z} \setminus \{0\}$ for the arcs $(u, v) \in A$, up to $\eta \in \mathbb{N}$ returns per node, refund percentage $0 \leq \alpha \leq 1$, and some revenue $\hat{R} \in \mathbb{N}$.*

Question: *Is there a price $\pi \in \mathbb{N}$ such that its revenue $R(\pi) \geq \hat{R}$?*

Remark 6.8. The influenced values can only change if C_t does, i.e., either a node buys or returns. Since every node can do both at most η times, the set $C_{2n\eta}$ is final and the selling process stops.

Similar to Algorithm Sell we can compute the revenue for given π in $\mathcal{O}(\eta m)$ time.

Allowing the return option in the instance $\mathcal{I}$ constructed for Theorem 3.9 does not result in any returns. Therefore, the reduction still holds.

Corollary 6.17. *PPAIηR is NP-complete for $\eta = \text{poly}(n)$.*

Remark 6.9. By setting $\eta = \left\lceil \frac{T}{2} \right\rceil$ and $d = 1$ for all $v \in V$ we allow the nodes to break in every round for both, i.e., perishable and returnable instances. With $\alpha = 0$ even the revenue is the same. For PPAIηR-v and PPAIηR-S the reduction also holds. Hence, the computational complexity proofs work analogously.

In this chapter we examined the basic product pricing model for not necessarily increasing customer sets. As long as the amount of changing customers — through break or return/rebuy — is limited we can

7 Diverse Product Pricing Aspects

Thus far we have studied deterministic synchronous selling in discrete rounds. This chapter includes elementary results for variations to the previous product pricing problems with additive influence.

7.1 Cooperative Pricing

So far we assumed that the nodes do not share information with each other. They were oblivious and bought once their influenced value was raised high enough. Thereby, with the exception of perishable goods and the return option in Chapter 6, they could not change their mind. This implies that nodes that were once eager to buy, now regret their decision.

On the other side, nodes could not coordinate a simultaneous purchase with their neighbors. In this section we examine the basic product pricing model with the possibility to build coalitions amongst the potential buyers.

Definition 7.1. A set $S \subseteq V$ is a *stable buyers set* for price π if $p_S(v) \geq \pi$ for $v \in S$ and $p_S(u) < \pi$ for $u \notin S$. If only $p_S(v) \geq \pi$ holds this is called an *internally stable buyers set.*

We focus on the stable buyers sets and define the corresponding problem CooPPAI.

Problem 6: CooPPAI *(Cooperative Product Pricing with Additive Influences)*

Instance: *A directed graph $G = (V, A)$, initial values $p(v) \in \mathbb{Z}$ for the nodes $v \in V$, influences $w(u, v) \in \mathbb{Z} \setminus \{0\}$ for the arcs $(u, v) \in A$, and some revenue $\hat{R} \in \mathbb{N}$.*

Question: *Is there a price $\pi \in \mathbb{N}$ with a stable buyers set S such that its revenue $R(\pi) = \pi \cdot |S| \geq \hat{R}$?*

Remark 7.2. We can check for price π and set S whether S is a stable buyers set in linear time. Therefore, CooPPAI $\in$ NP.

Note that this is closely related to Chapter 6. In this context, if we reach the last round with a break, the remaining customers are a stable buyers set. In general, finding a stable buyers set is akin to the Party Affiliation Game [AB92].

Remark 7.3. We do not have discrete synchronous selling rounds in CooPPAI, since all decisions and influences are covered before anyone actually buys.

We identify one case in which the uniqueness of S is guaranteed. If the proverb "the more the merrier" holds, i.e., every influence is positive, we can extend any given internally stable buyers set by "interested" nodes. This is the result of Lemma 3.12 on PPAI. We extend this to a maximal stable buyers set.

Lemma 7.1. *For a positive graph and any two stable buyers sets S_1, S_2 of price π, there exists a set $S \supseteq S_1 \cup S_2$ that is also stable for π.*

Thus there is a unique (cardinality and inclusion-wise) maximal buyers set.

Proof. The union $S_1 \cup S_2$ is already internally stable since the influenced values are non-decreasing in a positive graph.

To achieve a fully stable buyers set, we simply add nodes $v \notin S$ with $p_S(v) \geq \pi$ until no such node exists anymore. □

With Algorithm FixLowest we identify the maximum stable buyers sets.

Theorem 7.2. *Algorithm FixLowest correctly computes the optimal revenue for positive graphs with cooperation in $\mathcal{O}(m + n \log n)$ time.*

Proof. In contrast to FixHighest we start with fixing the lowest valued nodes.

Since the influences are positive, there is no higher influenced value for v than $p_V(v)$. Every node is part of the stable buyers set for $\min_{v \in V} p_V(v)$. Hence, in order to determine higher values $p^{\#}(v)$ we know that these fixed nodes do not buy and influence anymore.

The proof follows by induction, analogously to Theorem 3.14. □

Algorithm 12: FixLowest(G)

```
Data: A positive graph $G = (V, A)$ with initial values $p$ and influences $w$.
Result: The optimal revenue $R^*$.
Initialize: Revenue $R^* = 0$, $F = \emptyset$, and $p^{\#}(v) = p_V(v)$ for all $v \in V$.
// $p^{\#}(v)$ denotes the price up to which $v$ is part of a stable buyers set
while $V \setminus F \neq \emptyset$ do
    Compute $\pi^{\#} = \min_{v \in V \setminus F} p'(v)$ and $N = \left\{ v \in V \setminus F : p^{\#}(v) = \pi^{\#} \right\}$.
    $F = F \cup N$
    Update $p^{\#}(v) = \max\{p_{V \setminus F}(v), \pi^{\#}\}$ for all $v \in V \setminus F$.
Compute for sorted $p^{\#}(v_1) \geq \cdots \geq p^{\#}(v_n)$ the revenue $R^* = \max_{i=1,\dots,n} i \cdot p^{\#}(v_i)$.
return $R^*$
```

Corollary 7.3. *Problem* CooPPAI *is in* P *for positive graphs.*

For arbitrary graphs, CooPPAI *is shown to be* NP*-complete in [To14].*

As a product seller the main interest is a bigger revenue. So the natural question is: How much more or less revenue can be achieved given communication between the potential customers? We compare the revenue of positive graphs for uncooperative and cooperative product pricing, i.e., we compare the solutions of FixHighest and FixLowest.

Remark 7.4. The nodes can buy for prices $\pi > p_{\max}$ when cooperation is allowed. As extreme example we have $p(v) = 0$ for all $v \in V$ and $w(u, v) = M$ for all $u, v \in V$. In the uncooperative setting, this yields no revenue at all, whereas the cooperative instance has $R^* = M \cdot n(n-1)$.

Recall the values $p^*(v)$ of Remark 3.15, i.e., the maximum price for which $v \in V$ buys the product given a positive graph.

Lemma 7.4. *For a positive acyclic graph $G = (V, A)$ the values $p^\#(v) = p^*(v)$ coincide for all $v \in V$.*

Proof. We show this by induction on the topological sorting $v_1, \ldots, v_n$. Node v_1 is not influenced, hence, $p(v) = p^*(v) = p^\#(v)$.

Now assume that the claim holds for nodes $v_1, \ldots, v_k$. Thus we have the exact same buying behavior in both uncooperative and cooperative case, for every node $u \in \mathcal{N}^-(v_{k+1})$. It follows that v_{k+1} does also have the same values $p^\#(v_{k+1}) = p^*(v_{k+1})$. $\square$

More studies on the topic of stable buyers sets, coalitions, and general game theoretic aspects were done in [To14]. Amongst others, the viable coalitions are restricted to a family $\mathcal{S} \subseteq 2^V$ to enable further research.

7.2 Initial Value as Parameter

So far we have always set the the entirety of the parameters p and w. We assume now that for one particular node $v \in V$ the initial value is not set.

This is in preparation for future mechanism design approaches with one parameter (cf. [AT01], [APTT03], where the initial values would serve as the private information. The question is: How does the choice of $p(v)$ affect the sale of our product.

Remark 7.5. Setting $p(v)$ can potentially decrease or increase the revenue.

If the influences from v are highly negative, setting the initial value to $p_{\max}$ stops any purchase after the first round. On the other side, setting the initial value such that $p_C(v) \leq 0$ for all customer sets C, practically deletes v from the selling process, thereby allowing the product to achieve a high revenue.

We identify one particular case with guaranteed monotonicity — positive graphs.

Theorem 7.5. *For a positive graph we can determine the values $p^*(u)$ for $u \in V$ and all choices of $p(v)$ in $\mathcal{O}\left(nm + n^2 \log n\right)$ time.*

Proof. We compute the values $p^*(u)$ of nodes $u \in V \setminus \{v\}$ with Algorithm FixHighest for G in $\mathcal{O}(m + n \log n)$. This gives us all potential customer sets *before* a purchase of v itself.

Hence, we exactly know the influences on v and can determine $p^*(v)$ for every initial value $p(v)$. The function $f(p(v)) \mapsto p^*(v)$ is piecewise linear and increasing. An illustration of it is given in Figure 7.1.

By the choice of $p(v)$ we determine which nodes are uninfluenced by v — the nodes H with higher $p^*(u)$. So for prices $\pi > p^*(v)$ we already know the exact customer set and the optimum revenue.

Now going on to the lower prices $\pi \leq p^*(v)$, the range where v is actually changing something. On this range we know that nodes $H \cup \{v\}$ are always buying. Which means we can run Algorithm FixHighest again on $L := V \setminus H$ with H already fixed and $p^*(v)$ as parameter left.

In any case, node v gets fixed in the first iteration of FixHighest. We set $p^*(v) = \min\limits_{u \in H} p^*(u)$ and determine the values $p^*(u)$ for $u \in L \setminus \{v\}$. For any other choice of $p^*(v)$ with the same set H, we know that $p^*(u)$ is either the value we just computed or $p^*(v)$, as it was capped at this value.

In total we need one initial call to FixHighest and $n - 1$ more to determine all $p^*(u)$ subject to $p(v)$. □

Even though one value is missing, we can easily compute the buying behavior for positive graphs.

7.3 Two Product Pricing

The product we have marketed so far did not have any competition, i.e., getting another (similar) product was not a choice at all.

We introduce a rivaling second product with separate influences and its own price.

Definition 7.6. Let $p^{(1)}, w^{(1)}$ be the values and weights for the first product and $p^{(2)}, w^{(2)}$ those for the second.

The influences $w^{(1)}, w^{(2)}$ are separate for the two products, i.e., the purchase of one product does only change the influenced values concerning this product.

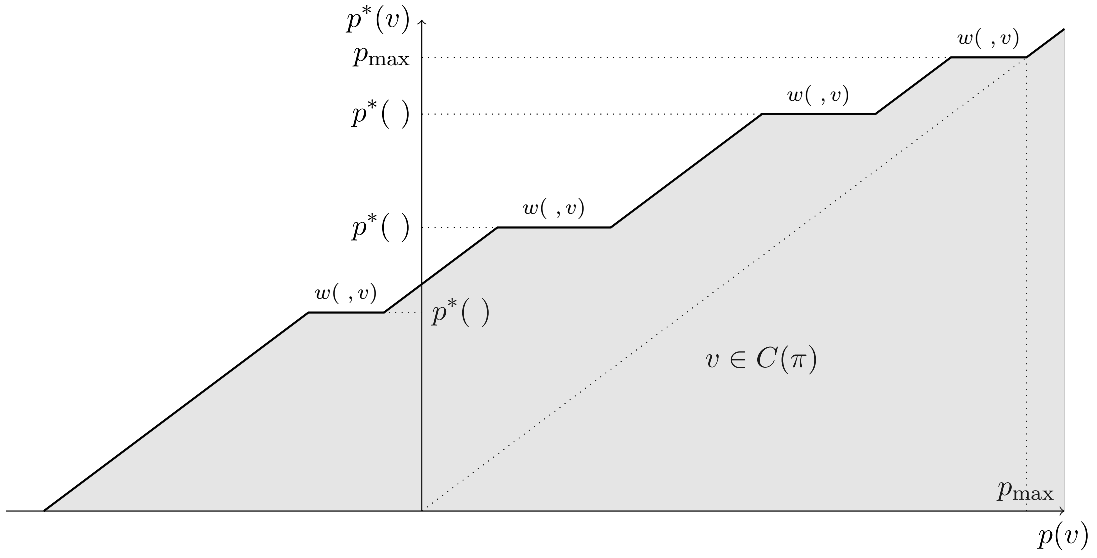

Figure 7.1: Buying behavior of v depending on $p(v)$

Given a price pair $\pi = (\pi^{(1)}, \pi^{(2)})$, we compute for each round the values $p'(v) \coloneqq p_{C^{(1)}(\pi)}(v) - \pi^{(1)}$ and $q'(v) \coloneqq p_{C^{(2)}(\pi)}(v) - \pi^{(2)}$.

Node v buys the first product if $p'(v) \geq \max\{0, q'(v)\}$, while v buys the second product if $q'(v) \geq \max\{0, p'(v)+1\}$, i.e., v buys whichever product has more benefit and the first if they are equal.

Remark 7.7. For some price pair $\pi = (\pi^{(1)}, \pi^{(2)})$ we can compute the customers $C^{(1)}(\pi)$ and $C^{(2)}(\pi)$ with a modified Sell in $\mathcal{O}(m)$ (cf. [Ger14]).

Behavior in a competition with separate positive influences: We assume that all influences are positive.

Lemma 7.6. *Given a graph $G = (V, A)$ with two products and positive influences and $v \in C^{(1)}(\pi^{(1)}, \pi^{(2)})$, the node v also buys for $(\rho^{(1)}, \rho^{(2)})$ with $\rho^{(1)} \leq \pi^{(1)}$ and $\rho^{(2)} \geq \pi^{(2)}$.*

Given one price $\pi^{(2)}$ we therefore have prices $\pi^{(1)} \in (0, p^*(v)]$ for which v buys the first product, and prices $\pi^{(1)} \in (q^*(v), p^{(1)}_{\max}]$ for which v buys the second product.

Proof. The pair $\pi' = (\rho^{(1)}, \pi^{(2)})$ only decreases the price for the first product. Since the price for the second product remains unchanged, the monotonicity of Lemma 3.12 still holds.

The same holds analogously for price $\rho' = (\pi^{(1)}, \rho^{(2)})$ and therefore for ρ itself. □

Observation 7.8. The pricing remains rather simple for one fixed price. However, even by having $\pi^{(1)} = \pi^{(2)}$, we loose the monotonicity described above.

In [Ger14] the pricing for two products is examined in detail. In particular for positive graphs it holds:

Theorem 7.7. *It is* NP*-complete to determine whether there is a price pair $\pi^{(1)} = \pi^{(2)}$ that achieves a collective revenue $\hat{R}$.*

In a sense, the positive influences of the rivaling product are negatively influencing the original.

7.4 Bounded Additive Influence

In our basic model the influenced value $p_C(v)$ can differ arbitrarily from the initial value $p(v)$. In [Ger14] the problem is examined for bounded influence.

Definition 7.9. We cap the influence with factor $\omega > 1$ to

$$p_C(v) \in \left[\frac{1}{\omega} p(v), \omega \cdot p(v)\right].$$

Remark 7.10. The reductions of Chapter 3 can be reused. In particular we can use the Shift Gadget 3.26.

For $\omega = 2$ we can satisfy the upper bound by adding $x := \sum_{a \in A^+} w(a)$ to every initial value. Thereby, we know that $p_C(v) \leq 2p(v)$. Analogously, by adding $y := -2 \cdot \sum_{a \in A^-} w(a)$ we know that $p_C(v) \geq \frac{1}{2} p(v)$.

In the next step in [Ger14] the influenced values are altered altogether.

Definition 7.11. For every $v \in V$ the function $\phi_v : \mathbb{Z} \to \mathbb{Z}$ changes the influenced value to

$$p_C(v) := p(v) + \phi_v \left(\sum_{u \in \mathcal{N}^-(v) \cup C} w(u,v) \right).$$

The functions ϕ_v are increasing and have diminishing returns, i.e.,

$$\phi_v(x+1) - \phi_v(x) \geq \phi_v(x+2) - \phi_v(x+1) \text{ for all } x \in \mathbb{N}$$

Furthermore, we require $\phi_v(0) = 0$, ϕ_v is Lipschitz continuous, and $\phi_v(x)$ is computable in polynomial time.

This model covers the problem that nodes may have influences from a big variety of neighbors, but their actual value is only dependent on a "dampened" group of customers.

The thesis [Ger14] goes on to show that the problem to find a price π achieveing a revenue $R(\pi) \geq \hat{R}$ is NP-complete for the dampened influenced values of Definition 7.11.

8 Outlook

We derived various results on computational complexity for deterministic Product Pricing with Additive Influence problems. While we have almost settled the complexity-landscape of these problems, it remains open whether PPAI on trees is NP-complete.

For a deeper understanding of the properties that are required to create fragments, it is an interesting endeavor to find the bare necessities of a graph with a high amount of fragments. While a polynomial number of fragments implies polynomial time solvability, it is not clear whether the converse implication holds. More specifically, is it true that a polynomial time algorithm for a graph class implies that the amount of fragments has to be polynomial as well? An answer in the affirmative would imply that Algorithm Frag is the universal polynomial time algorithm.

Concerning possible applications of the model, the actual — or at least more realistic — influenced values have to be determined first. First and foremost, a more accurate model needs to include bounds on the amount and magnitude of influences. In particular, in current social networks, one can typically choose whether to encounter other customers. Thus the positive influences can be considered prevalent. On the other side, if an encounter with a disliked individual is inevitable, this can darken ones mood towards the product or service immensely.

Furthermore, we assumed that the influenced value does not naturally decrease over time. Especially with respect to perishable goods and subscriptions, the benefit of a product is likely to diminish after some period. In connection to the return option, a second-hand market is a natural extension.

To find a proper influence model, some kind of survey for a couple of different products is advisable. This also applies to the actual structure of the social network considered. While there are plenty of graph libraries with social networks, there is none in relation to influences associated with a product. Considering proper instances, the actual performance of our algorithms can be studied and compared with formulations as integer program or graph dynamical system.

In any real-world application, we have to consider strategic and/or irrational behavior of the potential customers. Friends are communicating and plan together whether they want to use some product or service. To cater this, retailers are offering discounted bundles, thereby providing incentive to persuade others. For more realistic models, one would have to include among other things (partly) random behavior of the customers.

We only briefly tackled these game theoretic aspects. With a better understanding of the purchase behavior, one could design mechanisms to manage the sale a product. In the best-case, this would yield us truthful values and a high revenue. The question remains, how to manage the reward for customers. Currently influential individuals are often rewarded through affiliate marketing.

Finally, the different variants we introduced in this thesis and the possible variants we sketched here, are yet mostly unconnected.

Bibliography

[AB92] John H. Aldrich and William T. Bianco. A game-theoretic model of party affiliation of candidates and office holders. *Mathematical and Computer Modelling*, 16(8):103 – 116, 1992.

[APMS$^+$99] Giorgio Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1999.

[APTT03] Aaron Archer, Christos H. Papadimitriou, Kunal Talwar, and Éva Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. *Internet Mathematics*, 1(2):129–150, 2003.

[AT01] Aaron Archer and Éva Tardos. Truthful mechanisms for one-parameter agents. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 482–491, 2001.

[BFO10] Allan Borodin, Yuval Filmus, and Joel Oren. Threshold Models for Competitive Influence in Social Networks. In *Proceedings of the 6th International Conference on Internet and Network Economics*, WINE'10, pages 539–550, Berlin, Heidelberg, 2010. Springer-Verlag.

[BHM$^+$07] Chris Barrett, Harry B. Hunt, Madhav V. Marathe, S.S. Ravi, Daniel J. Rosenkrantz, Richard E. Stearns, and Mayur Thakur. Predecessor existence problems for finite discrete dynamical systems. *Theoretical Computer Science*, 386(1):3 – 37, 2007.

[Blu93] L. Blume. The statistical mechanics of strategic interaction. *Games and Economic Behavior*, 5:387–424, 1993.

[BS79] I. N. Bronstein and K. A. Semendjajew. Taschenbuch der Mathematik. In G. Grosche and V. Ziegler, editors, *Taschenbuch der Mathematik*. BSB B. G. Teubner Verlagsgesellschaft, Nauka-Verlag, Leipzig, Moskau, 19 edition, 1979.

[CBO12] Ozan Candogan, Kostas Bimpikis, and Asuman Ozdaglar. Optimal pricing in networks with externalities. *Oper. Res.*, 60(4):883–905, July 2012.

[CKM66] J. S. Coleman, E. Katz, and H. Menzel. *Medical innovation: a diffusion study.* Bobbs Merrill, 1966.

[CLS+10] Wei Chen, Pinyan Lu, Xiaorui Sun, Yajun Wang, and Zeyuan Allen Zhu. Pricing in social networks: Equilibrium and revenue maximization. *CoRR*, abs/1007.1501, 2010.

[CSRL01] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms.* McGraw-Hill Higher Education, 2nd edition, 2001.

[Die05] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics).* Springer, August 2005.

[DR01] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 57–66, New York, NY, USA, 2001. ACM.

[Ell93] G. Ellison. Learning, local interaction, and coordination. *Econometrica*, 61(5):1047–1071, 1993.

[FKLL13] Michal Feldman, David Kempe, Brendan Lucier, and Renato Paes Leme. Pricing public goods for private sale. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, EC '13, pages 417–434, New York, NY, USA, 2013. ACM.

[FT87] Michael L. Fredman and Robert Endre Tarjan. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *J. ACM*, 34(3):596–615, July 1987.

[Ger14] Patrick Gerhards. Algorithms for Pricing Problems in Social Networks. Master's thesis, University of Kaiserslautern, 2014.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[GLM01] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3):211–223, 2001.

[HMS08] Jason Hartline, Vahab Mirrokni, and Mukund Sundararajan. Optimal marketing strategies over social networks. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 189–198, New York, NY, USA, 2008. ACM.

[Joh15] Sebastian Johann. Algorithmic Aspects of Pricing Problems in Networks. Bachelor's thesis, University of Kaiserslautern, 2015.

[KKM$^+$10] Chris J. Kuhlman, V. S. Anil Kumar, Madhav V. Marathe, S. S. Ravi, and Daniel J. Rosenkrantz. Finding critical nodes for inhibiting diffusion of complex contagions in social networks. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD'10, pages 111–127, Berlin, Heidelberg, 2010. Springer-Verlag.

[KKM$^+$11] Chris J. Kuhlman, V. S. Anil Kumar, Madhav V. Marathe, Henning S. Mortveit, Samarth Swarup, Gaurav Tuli, S. S. Ravi, and Daniel J. Rosenkrantz. A General-purpose Graph Dynamical System Modeling Framework. In *Proceedings of the Winter Simulation Conference*, WSC '11, pages 296–308. Winter Simulation Conference, 2011.

[KKM$^+$15] Chris J. Kuhlman, V. S. Anil Kumar, Madhav V. Marathe, S. S. Ravi, and Daniel J. Rosenkrantz. Inhibiting diffusion of complex contagions in social networks: theoretical and experimental results. *Data Mining and Knowledge Discovery*, 29(2):423–465, 2015.

[KKT03] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the Spread of Influence Through a Social Network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[KST14] Sven Oliver Krumke, Florian D. Schwahn, and Clemens Thielen. Being Negative Makes Life NP-hard (for Product Sellers). In *Fun with Algorithms - 7th International Conference, FUN 2014, Lipari Island, Sicily, Italy, July 1-3, 2014. Proceedings*, pages 277–288, 2014.

[Led97] J. Ledyard. Public Goods: A Survey of Experimental Research. Levine's Working Paper Archive 509, David K. Levine, Aug 1997.

[Mor00] S. Morris. Contagion. *Review of Economic Studies*, 67:57–78, 2000.

[MR08] Henning S. Mortveit and Christian M. Reidys. *An Introduction to Sequential Dynamical Systems.* Universitext. Springer, 2008.

[Pap94] Christos M. Papadimitriou. *Computational complexity.* Addison-Wesley, Reading, Massachusetts, 1994.

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization.* Prentice-Hall, Inc., 1982.

[SB75] Bruce A. Scherr and Emerson M. Babb. Pricing public goods. *Public Choice*, 23(1):35–48, 1975.

[Swa14] Anand Swaminathan. An Algorithm for Influence Maximization and Target Set Selection for the Deterministic Linear Threshold Model. Master's thesis, Virginia Polytechnic Institute and State University, 2014.

[To14] Thanh To. Game Theoretic Aspects of Pricing Problems in Social Networks. Master's thesis, University of Kaiserslautern, 2014.

[Val79] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

[Val95] T.W. Valente. *Network Models of the Diffusion of Innovations.* Communication Series. Hampton Press, 1995.

[You06] H. Peyton Young. *Economy as an evolving complex system*, volume 3, chapter The Diffusion of Innovations in Social Networks, pages 267–282. Oxford University Press US, 2006.

List of Figures

List of Problems

List of Integer Programs

List of Algorithms

Curriculum Vitae

Florian David Schwahn

Education

- 03/2006 Graduation (Abitur)
  Rudi-Stephan-Gymnasium Worms
- 4/2006 – 9/2012 Studies in Mathematics
  University of Kaiserslautern
  Minor: Chemistry (undergraduate), Computer Science (graduate)
- 9/2012 Diploma in Mathematics (Dipl.-Math.)
  Diploma Thesis: Canadian Traveler Problems — Online Graph Exploration with Blocked Edges
- 11/2012 – 03/2013 Research Assistant
  Department of Mathematics University of Kaiserslautern
- since 04/2013 Scholarship Ph. D. Student
  Optimization Working Group, Department of Mathematics, University of Kaiserslautern

Wissenschaftlicher und beruflicher Werdegang

Florian David Schwahn

Ausbildung

- 03/2006 Abitur
  Rudi-Stephan-Gymnasium Worms
- 4/2006 – 9/2012 Studium der Mathematik
  Technische Universität Kaiserslautern
  Anwendungsfach: Chemie (Vordiplom), Informatik (Hauptstudium)
- 9/2012 Diplom in Mathematik (Dipl.-Math.)
  Diplomarbeit: Canadian Traveler Problems — Online Graph Exploration with Blocked Edges
- 11/2012 – 03/2013 Wissenschaftlicher Mitarbeiter
  Fachbereich Mathematik, Technische Universität Kaiserslautern
- seit 04/2013 Promotionsstudent
  AG Optimierung, Fachbereich Mathematik, Technische Universität Kaiserslautern