ONLINE DYNAMIC PRICING OF COMPLEMENTARY PRODUCTS

Marco Mussi
Politecnico di Milano
marco.mussi@polimi.it

Marcello Restelli
Politecnico di Milano
marcello.restelli@polimi.it

ABSTRACT

Traditional pricing paradigms, once dominated by static models and rule-based heuristics, are increasingly being replaced by dynamic, data-driven approaches powered by machine learning algorithms. Despite their growing sophistication, most dynamic pricing algorithms focus on optimizing the price of each product independently, disregarding potential interactions among items. By neglecting these interdependencies in consumer demand across related goods, sellers may fail to capture the full potential of coordinated pricing strategies. In this paper, we address this problem by exploring dynamic pricing mechanisms designed explicitly for complementary products, aiming to exploit their joint demand structure to maximize overall revenue. We present an online learning algorithm considering both positive and negative interactions between products' demands. The algorithm utilizes transaction data to identify advantageous complementary relationships through an integer programming problem between different items, and then optimizes pricing strategies using data-driven and computationally efficient multi-armed bandit solutions based on heteroscedastic Gaussian processes. We validate our solution in a simulated environment, and we demonstrate that our solution improves the revenue w.r.t. a comparable learning algorithm ignoring such interactions.

1 Introduction

The advent of digital technologies, combined with the chance to access large amounts of data, is reshaping the way businesses set prices, increasing the adoption of *dynamic pricing* algorithms (Den Boer, 2015). These algorithms leverage advanced machine learning tools to analyze a multitude of factors, from consumer behavior and market trends to external variables such as product availability and competitor pricing. By adjusting prices in response to these dynamic factors, businesses employing dynamic pricing strategies aim to determine the optimal pricing strategy to maximize profit within a given time horizon. One of the key factors in dynamic pricing consists of understanding the *demand* for a product. The demand represents the *willingness* of the customers to buy an item, given a price. The extent to which demand varies as the price of an item changes defines the so-called *elasticity* of a product. Such a demand usually takes into account *only* the price at which we sell the item under analysis. However, considering all the products as independent prevents us from learning the correlations between the products in the catalog and other baskets, as well as carryover phenomena that can be exploited to maximize profit (Yan and Bandyopadhyay, 2011). The literature distinguishes between two types of product interdependencies: *complementarity* and *substitutability*. Complementary products "complement" each other (Nicholson and Snyder, 2017); these kinds of products are usually bought together (e.g., a printer and its ink cartridges), and the sale of one implies an increment in the probability of selling the other. On the other hand, substitutable goods serve the same purpose and are rarely bought together; instead, they tend to cannibalize each other (e.g., two identical items of different brands). Complementary and substitutable products have different effects on the demand function for the related products. The concept of *cross-price elasticity* is useful to describe these effects: it measures the proportionate change in the quantity of a good demanded in response to a proportionate change in the price of some other good (Nicholson and Snyder, 2017). For complementary products, the cross-price elasticity is *negative*: this is intuitive since a lower price of product a leads to higher sales of product a and, since the demand of product a drives the demand of the complementary product b, we can also expect the quantity of product b to increase. On the contrary, the effect for substitutable products is the opposite: the cross-price elasticity is *positive* since the lower demand for product c leads to an increase in the demand for its competing product d. The most challenging task in jointly optimizing pricing strategies involves identifying complementary relationships. Indeed, complementarity relationships exhibit characteristics such as *asymmetry* and *non-transitivity* (Kocas et al., 2018;

Yu et al., 2019; Xu et al., 2020). Furthermore, two products can be complementary while belonging to semantically different areas, so complementarity is inherently harder to capture and cannot be identified simply by similarity. On the other hand, substitutability relationships are symmetrical and transitive, and concern only products that satisfy the same need, which can be more easily identified also from a *semantic* point of view.

Original Contributions. In this work, we aim to jointly optimize the pricing strategies of strictly related products to reach optimal solutions, taking into account the products' positive and negative interactions. We focus our attention on the more challenging problem of complementary relations and propose a computationally and statistically efficient solution to identify complementary products and determine their optimal pricing strategy, given a target index. Our solution avoids the request for information that is complex to retrieve and requires only purchase transaction data to detect complementary relations and estimate the optimal pricing strategy. More in detail, our contributions can be summarized as follows:

- In Section 3, we formulate the problem of finding the optimal pricing strategy in the presence of both positive and negative interactions among products. We introduce the setting, the decision variables, the underlying assumptions, and the corresponding learning problem.
- In Section 4, we introduce `Complementary Products Pricing` (`CPP`), a three-stage algorithm for optimizing pricing strategies in environments with interacting products. Sections 5, 6, and 7 detail each stage of the algorithmic framework. We first aggregate substitutable products into meta-products, which can be efficiently identified and are mutually non-competing. Then, we design a new solution for detecting the most *profitable* complementary relations, avoiding at the same time a combinatorial explosion of complexity w.r.t. the number of products in the catalog. Finally, we derive a computationally and statistically efficient procedure for learning the optimal pricing policies.
- In Section 8, we validate our approach through a simulation study that allows us to test different levels of correlation among products. We compare our proposed methods against a baseline based on non-parametric regression models that price products independently. The results highlight the learning performance of our algorithms and provide insights into when exploiting inter-product relationships becomes particularly beneficial.

In Section 2, we present the relevant background needed to read the work properly. In Section 9, we discuss the relevant literature. Finally, in Section 10, we summarize our work, and we draw some possible research perspectives.

Notation. Throughout this paper, we interchangeably use lower- and upper-case letters (e.g., $z,\ Z$) to indicate scalar values, lower-case boldface letters (e.g., $\mathbf{v}$) to indicate (column) vectors, upper-case boldface letters (e.g., $\mathbf{A}$) to indicate matrices, and calligraphic letters (e.g., $\mathcal{X}$) to indicate sets. Given a vector or a matrix, we indicate with $(\cdot)^\top$ the transpose operator. Given a vector $\mathbf{v}$, we define $\mathsf{diag}(\mathbf{v})$ as the operator returning a diagonal matrix with $\mathbf{v}$ on the diagonal. Given a matrix $\mathbf{A}$, we indicate with $\det(\mathbf{A})$ its determinant. We indicate with $\mathbf{I}_Z$ the identity matrix of size Z (we omit the index Z when it can be inferred from the context). Given a scalar value $Z \in \mathbb{N}_{\geq 1}$, we define $[\![Z]\!] \coloneqq \{1, 2, \dots, Z\}$. Given a finite set $\mathcal{X}$, we denote with $|\mathcal{X}|$ its cardinality. Given an event/condition e, we define the indicator function $\mathbb{1}\{e\}$, which takes the value 1 if its argument e is true, and 0 otherwise. We use $\mathcal{O}(\cdot)$ to present results – on both statistical and computational complexity – omitting constants.

2 Gaussian Processes for Pricing

In this section, we provide the necessary background to fully understand this work.

Gaussian Processes (GPs, Rasmussen and Williams, 2006) are a powerful and flexible machine learning technique, widely used for *regression* and *classification* tasks. We focus on GPs for regression, where we want to learn a function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \in \mathbb{R}^d$ is the input space. To learn this function, we observe T noisy samples of such a function. For every $t \in [\![T]\!]$, samples are drawn from a Gaussian distribution with expected value $f(\mathbf{x})$, formally $y_t = f(\mathbf{x}_t) + \epsilon_t$, where ϵ_t is a zero-mean σ^2-subgaussian random noise, independent conditioned to the past.[1] A GP, denoted by $\mathcal{GP}(\mu(\cdot), k(\cdot,\cdot))$, is a collection of random variables $(f(\mathbf{x}))_{\mathbf{x}\in\mathcal{X}}$, such that every sub-collection of random variables $(f(\mathbf{x}_i))_{i\in[\![J]\!]}$ is jointly Gaussian with mean $\mathbb{E}[f(\mathbf{x}_i)] = \mu(\mathbf{x}_i)$, and covariance $\mathbb{E}[(f(\mathbf{x}_i) - \mu(\mathbf{x}_i))(f(\mathbf{x}_j) - \mu(\mathbf{x}_j))] = k(\mathbf{x}_i, \mathbf{x}_j)$. We assume that function $f(\cdot)$ belongs to a *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}_k$ (Berlinet and Thomas-Agnan, 2004).

Given the vector of points $[\mathbf{x}_1, \dots, \mathbf{x}_T]^\top$, we define the *kernel matrix* $\mathbf{K}_T = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j\in[\![T]\!]}$. Then, considering the corresponding vector of observed rewards $\mathbf{y}_T = [y_1, \dots, y_T]^\top$, and given a target point $\mathbf{x}$, we can compute a GP

[1] A zero-mean random variable ξ is σ^2-subgaussian if $\mathbb{E}[\exp(\lambda\xi)] \leq \exp\left(\frac{\lambda^2\sigma^2}{2}\right)$, for every $\lambda \in \mathbb{R}$.

estimate of the model $\widehat{\mu}_T(\mathbf{x})$ and the uncertainly $\widehat{\sigma}^2_T(\mathbf{x})$:

$$\widehat{\mu}_T(\mathbf{x}) = \mathbf{k}_T(\mathbf{x})^\top \left(\mathbf{K}_T + \sigma^2 \mathbf{I}_T\right)^{-1} \mathbf{y}_T$$
$$\widehat{\sigma}^2_T(\mathbf{x}) = k(\mathbf{x},\mathbf{x}) - \mathbf{k}_T(\mathbf{x})^\top \left(\mathbf{K}_T + \sigma^2 \mathbf{I}_T\right)^{-1} \mathbf{k}_T(\mathbf{x})$$

where $\mathbf{k}_T(\mathbf{x}) = [k(\mathbf{x}_1,\mathbf{x}),\dots,k(\mathbf{x}_T,\mathbf{x})]^\top$.[2]

The kernel should be chosen according to the covariance model over the different samples in the problem under analysis. Due to its versatility, the *squared exponential* (SE) kernel is the most commonly used. It is defined as:

$$k(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\ell^2}\right),$$

where $\ell > 0$ represents the *length scale*. The kernel's hyperparameters (in this case, just ℓ) are typically optimized by maximizing the marginal likelihood.

2.1 Online Learning with Gaussian Processes

GPs can be used to learn *online* the best point $\mathbf{x}^* \in \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$, i.e., the one maximizing function $f(\cdot)$. Several algorithms allow us to find $\mathbf{x}^*$ minimizing, at the same time, the cumulative regret $\widetilde{R}(\boldsymbol{\pi}, T) \coloneqq Tf(\mathbf{x}^*) - \sum_{t=1}^{T} f(\mathbf{x}_t)$, where samples $\mathbf{x}_t$ are drawn according to algorithm/policy $\boldsymbol{\pi}$.[3] Nearly-optimal algorithms (Scarlett et al., 2017) in this sense are `GP-UCB` (Srinivas et al., 2010) and its improvement `IGP-UCB` (Chowdhury and Gopalan, 2017). In `IGP-UCB`, which we will adopt later in this paper, at every time step $t \in [\![T]\!]$ we want to play an optimistic action, according to the usual exploration/exploitation trade-off we have in online learning. To select such an action, the posterior mean $\widehat{\mu}_{t-1}(\mathbf{x})$ and uncertainty $\widehat{\sigma}^2_{t-1}(\mathbf{x})$ are computed using all samples available up to time $t-1$, for every $\mathbf{x} \in \mathcal{X}$. The best action is then selected according to:

$$\mathbf{x}_t \in \arg\max_{\mathbf{x}\in\mathcal{X}} \ \widehat{\mu}_{t-1}(\mathbf{x}) + \beta_t \sqrt{\widehat{\sigma}^2_{t-1}(\mathbf{x})}, \tag{1}$$

with:

$$\beta_t = B + \sqrt{2\sigma^2(\gamma_{t-1} + 1 + \mathsf{ln}(1/\delta))}, \tag{2}$$

where B is an upper bound on the RKHS norm ($\|f\|_k \leq B$) and $\gamma_{t-1} \coloneqq \mathsf{ln}(\mathsf{det}(\mathbf{I}_{t-1} + \sigma^{-2}\mathbf{K}_{t-1}))/2$ is the maximum information gain at time $t-1$, that for SE kernels is $\mathcal{O}((\mathsf{ln}t)^{d+1})$. For more details, we refer the reader to (Srinivas et al., 2010; Chowdhury and Gopalan, 2017; Vakili et al., 2021).

Gaussian Processes for Bernoulli Random Variables. GPs are naturally designed to work with Gaussian random variables. However, in pricing, we aim to model demand curves, which are designed by learning a demand model starting from Bernoulli samples. Nevertheless, the versatility of GPs allows them to extend beyond purely Gaussian settings and to model a broader class of subgaussian random variables (Chowdhury and Gopalan, 2017). This fact enables the chance to use them to model Bernoulli random variables, as every random variable bounded in the range $[a,b]$ is σ^2-subgaussian with $\sigma^2 = (b-a)^2/4$. Given this, a Bernoulli random variable is subgaussian with $\sigma^2 = 1/4$.[4]

Computational Complexity. Computing $\widehat{\mu}_T$ and $\widehat{\sigma}^2_T$ requires the inversion of kernel matrix $\mathbf{K}_T$, which scales as $\mathcal{O}(T^3)$ with the number of training points T. If we want to utilize online learning solutions in this setting, we need to perform such estimates iteratively, which leads to an overall complexity of $\mathcal{O}(T^4)$. Such computational burden, even if polynomial w.r.t. T, is very demanding from the practical perspective. It can be reduced by observing that kernel matrices are incrementally built by adding one row and one column to the previous kernel matrix, and thus enables the chance to use *block-wise matrix* inversion (Lu and Shiou, 2002) and reduce the overall complexity of online learning with GPs to $\mathcal{O}(T^3)$.

3 Problem Formulation

In this section, we formalize the problem we address in this paper. We start in Section 3.1 by defining the setting and the related assumptions. Then, in Section 3.2, we present the learning problem, and we define the metric we use for evaluating the quality of our solution.

[2]This expression of $\widehat{\sigma}^2_T(\mathbf{x})$ considers just the epistemic uncertainty of the model, not the aleatoric one.

[3]This objective is equivalent to maximizing the cumulative reward during the interaction.

[4]At the moment, there are no specific results on GPs with Bernoulli rewards (Mussi et al., 2024).

3.1 Setting

We want to find the optimal pricing strategy for a set of products $\mathcal{P}$, with $|\mathcal{P}| = P$. Our goal is, given a time horizon T, to set for every time $t \in [\![T]\!]$ a vector of percentage margins (from now on, margins) $\mathbf{m}_t = (m_{1,t}, \ldots, m_{P,t})$ where $m_{i,t} \in \mathcal{M}$ is the price we choose for product i at time t, and $\mathcal{M} \subset \mathbb{R}$ is the set of possible margins, which we assume to be finite with $|\mathcal{M}| = M$. We can change our pricing strategy at every time step $t \in [\![T]\!]$. A single time step t can be an arbitrary, given, time interval (e.g., a week). We define the (percentage) margin $m_{i,t}$ as:

$$m_{i,t} \coloneqq \frac{p_{i,t} - c_i}{c_i},$$

where $p_{i,t}$ is the selling price for product i at time t, and c_i its acquisition cost. For a generic product $i \in \mathcal{P}$, we denote as $\overline{v}_i(\mathbf{m})$ the average volumes (sales), which we assume to depend on the whole margin vector $\mathbf{m}$. We assume that, given the number of impressions $n_{i,t}$ (i.e., the number of times the product i and its price is shown to a customer at time t) of a product and its demand function $d_i(\mathbf{m}_t)$, the number of realized sales are modeled as a binomial random variable with mean $d_i(\mathbf{m}_t)$ and number of trials $n_{i,t}$, that is: $v_{i,t} \sim \text{Binomial}\,(n_{i,t},\, d_i(\mathbf{m}_t))$, so that the expected value satisfies $\mathbb{E}[v_{i,t}(\mathbf{m}_t)] = \overline{v}_i(\mathbf{m}_t) = \overline{n}_i\, d_i(\mathbf{m}_t)$, where $\overline{n}_i$ is the average number of impressions for product i in a single time step.

We consider a scenario in which we have both positive and negative interactions among the products, i.e., we assume that the purchase of a product may affect the purchase probability of all the other products. We identify two types of relationships between products: *substitutability* and *complementarity*. Consider two products $a, b \in \mathcal{P}$. On the one hand, we call a and b substitutable products if an increase in the sales of one product implies a decrease in the sales of the other. On the other hand, we call a and b complementary products if an increase in the sales of one product implies an increase in the sales of the other.

Assumptions and Available Data. We consider non-perishable products with unlimited availability. These assumptions, nowadays, hold in several cases. For example, the unlimited availability virtually holds for e-commerce websites adopting the *dropshipping* (Singh et al., 2018) paradigm. We assume to be in a stationary environment, where both demand curves and acquisition costs remain constant over time.[5] We consider a scenario in which we do not have access to information related to the products we are selling to be more general, except for the information related to substitutable goods, i.e., the one satisfying the same need.[6] We only make the mild assumption to have access to transaction data reporting all the sales for every product $i \in \mathcal{P}$, divided by baskets, and provided with the timestamp of the sale. We also assume that we can observe when a customer chooses not to buy a product to be able to compute demand curves.[7]

3.2 Learning Problem

The goal of our learning problem is to find the vector of the optimal margin $\mathbf{m}^*$, i.e., the vector maximizing our objective function $f(\mathbf{m})$. Formally:

$$\mathbf{m}^* \in \underset{\mathbf{m} \in \mathcal{M}^P}{\arg\max}\, f(\mathbf{m}), \tag{3}$$

where the objective function $f(\mathbf{m})$ is the *profit*:

$$f(\mathbf{m}) \coloneqq \sum_{i \in [\![P]\!]} f_i(\mathbf{m}) = \sum_{i \in [\![P]\!]} m_i\, c_i\, \overline{v}_i(\mathbf{m}), \tag{4}$$

over all the products.[8] We call $\boldsymbol{\pi} = (\pi_t)_{t \in [\![T]\!]}$ the (history-dependent) policy returning at each time t a vector of margins $\mathbf{m}_t$. We define the cumulative reward of such a policy as:

$$R(\boldsymbol{\pi}, T) \coloneqq \sum_{t \in [\![T]\!]} f(\mathbf{m}_t). \tag{5}$$

The goal of our algorithm is to find a policy maximizing the expected cumulative reward $\mathbb{E}\left[R(\boldsymbol{\pi}, T)\right]$, where the expectation is taken over the randomness of the realizations and the possible randomness of the algorithm.[9]

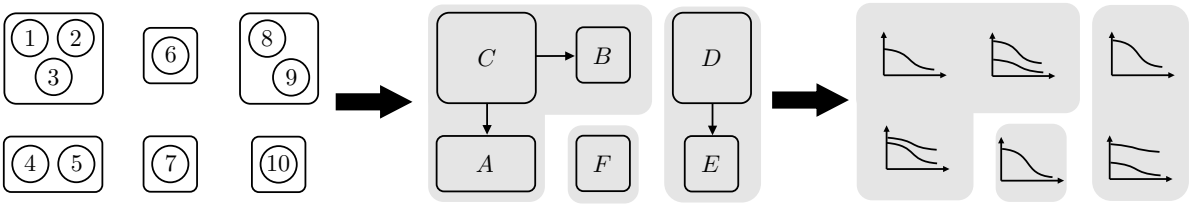
[5] The extension to non-stationary environments has been discussed several times for the independent product pricing (Bauer and Jannach, 2018; Nambiar et al., 2019; Javanmard et al., 2020; Mussi et al., 2022) and is out of the scope of this work.

[6] As already discussed in Section 1, this is a mild request as these relations are easy to detect.

[7] This is a mild assumption as well in practice, as, for example, an e-commerce website can keep track of the times in which a webpage of an item has been visited.

[8] The objective function can be made more general by considering a convex combination of revenue and profit. In this case, the objective function f becomes $f(\mathbf{m}) \coloneqq \sum_{i \in [\![P]\!]} (m_i + \alpha)\, c_i\, v_i(\mathbf{m})$, with $\alpha \in [0, 1]$. In this formulation, it is easy to observe that if we select $\alpha = 0$ we optimize the *profit* (as in Equation 4), while if we select $\alpha = 1$ we optimize the *revenue*.

[9] This is equivalent to minimizing the expected cumulative regret $\mathbb{E}[\widetilde{R}(\boldsymbol{\pi}, T)] \coloneqq T f(\mathbf{m}^*) - \mathbb{E}[R(\boldsymbol{\pi}, T)]$.



Stage 1: Aggregate Substitutable Products (Section 5)

Stage 2: Mining Complementary Relations (Section 7)

Stage 3: Demand Curve Modeling and Pricing Strategy Optimization (Section 6)

Figure 1: Algorithm outline.

4 Algorithm: Overview

In this section, we present an overview of `Complementary Products Pricing` (`CPP`), an algorithm that identifies and optimizes the pricing strategy of complementary products in an online fashion. The algorithm takes as input the product catalog, including information about substitutable products, and all the order records, and provides as output a coherent pricing strategy for all products. The algorithm is structured in 3 stages, which can be summarized as follows:

(Stage 1) *Aggregate Substitutable Products*: all the products satisfying the same need are aggregated in meta-products. These new meta-products will be the actual products we use in the next stages (we will refer to them simply as products for the sake of simplicity, as the original products are no longer considered). This operation is performed only once at the algorithm's startup and is repeated only in the case of a change in the catalog.

(Stage 2) *Mining Complementary Relations*: all the products are analyzed to search for the *leader-follower sets* which are more profitable to exploit. This operation is repeated at regular intervals, as we want to avoid performing it at every time step to strike a balance between the algorithm's performance and computational complexity.

(Stage 3) *Demand Curve Modeling and Pricing Strategy Optimization*: We utilize previously detected complementary relations to estimate demand curves, considering also *cross-selling* dynamics. These demand curves are then used to define an exploratory pricing strategy.

A graphical representation of such a pipeline is provided in Figure 1, where an example of this aggregation is illustrated. We can observe how 1, 2, and 3 satisfy the same need, and they are aggregated into the meta-product C. In the next phases, we consider only meta-products, such as C. Then, we identify complementary relations and define the leaders and followers. In the figure, we observe how C and D are leaders, A, B, and E are followers, while F is not related to any other item. Finally, we define the demand curve for these clusters of products and their subsequent pricing strategies.

In the following sections, we present the three stages in detail. We first present the stage in which we aggregate substitutable products and the rationale behind this operation (Section 5), then we present our demand learning model for both independent and complementary products (Section 6), and, finally we present our solution to detect the relevant complementary relations (Section 7). The rationale behind this inverted order w.r.t. the algorithm's flow is that to understand the solution to detect complementary products, we need to know how to estimate demand curves.

5 Algorithm: Aggregate Substitutable Products

In this section, we discuss why we need to aggregate substitutable products. Indeed, the first step of our algorithm involves aggregating substitutable products into meta-products, each representing a specific *need*. Then, we will define (Sections 6 and 7) the optimal margin for the meta-product that will be applied to all the products that constitute it.[10]

The need to cluster substitutable products stems from the characteristics of these types of products. Two substitutable products compete with each other and are often subject to the phenomenon of cannibalization (Moorthy and Png, 1992) since they satisfy the same need. Indeed, pricing these products separately would exacerbate cannibalization and could lead to them competing against each other, resulting in a reduction of profit. Clustering these products together and considering them as a single product, where the objective function is the total sales, removes the chance that they will

[10]More elaborate solutions on how we can find margins for products that are part of a meta-product should be considered. However, we believe this problem deserves a standalone work.

cannibalize each other. This problem of cannibalization can be viewed from a game-theoretic perspective, where we can choose whether to opt for a cooperative solution or two competitive algorithms when pricing two substitutable products. We know that the optimal *stable* solution for the cooperative case is always better or equal than the one for non-cooperative games (i.e., a possibly suboptimal *Nash equilibrium*, Nash 1950, 1951). In the remainder of this paper, we refer to meta-products calling them products for the sake of simplicity, as the original products are no longer considered.

6 Algorithm: Demand Curves Modeling and Pricing Strategy Optimization

In this section, we introduce an efficient approach to demand learning based on heteroscedastic Gaussian processes. Although in general the demand for a product can depend on the prices of all other products, modeling such dependencies is highly challenging due to the extremely large action space. Consequently, it becomes essential to find a way to reduce both the statistical complexity (i.e., the number of samples needed for learning) and the computational complexity (i.e., the number of operations required to estimate the optimal margin or action). To address this, we adopt a *leader-follower* framework, wherein the set of products is partitioned into disjoint subsets, each consisting of one *leader* and zero, one, or more *followers*. When a leader has no followers, we say the product is *independent* of the others. In this section, we assume that the leader-follower sets are given; the procedure for determining these associations is deferred to Section 7. Our objective is to construct the vector of margins $\mathbf{m}_t \in \mathcal{M}^P$ for each time step $t \in [\![T]\!]$. Given the partition into leader-follower sets, we compute the optimal margins by processing each set individually. We begin by presenting the demand model and the resulting strategy for selecting the optimistic margin for *independent products* (Section 6.1), and then we extend our analysis to the case of *leader-follower sets* (Section 6.2).

6.1 Independent Products

Given an independent product i, we now want, assuming that its demand depends only on its own margin, to estimate such a demand from historical data and perform an optimistic exploration to learn its optimal margin m_i^*.

What we observe during a single time step (e.g., a week) is a sequence of visits to, e.g., the online page of a product, and then we observe the realization of the Bernoulli indicating whether the customer bought or not the product. In principle, we can create a demand curve in this way using standard GPs between margins and a binary-valued variable to represent the realization of a Bernoulli (1 if the product has been sold, 0 otherwise), generating in this way a model of the demand curve, considering that every sample has $\sigma^2 = 1/4$.[11] However, this approach incurs a significant computational burden at each time step $t \in [\![T]\!]$, with a per-step complexity of $\mathcal{O}(t^3\overline{n}_i^3)$. This results in an overall complexity of $\mathcal{O}(T^4\overline{n}_i^3)$.[12] Afford such complexity is not feasible, especially in the context of the pipeline we aim to implement. In the following, we present how heteroscedastic GPs can be leveraged to substantially reduce this computational overhead.

Demand Learning via Heteroscedastic GPs — A First Solution. We can aggregate information by creating a data structure in which we store information about sales. Such a structure at time t for independent products should contain, for every $\tau \in [\![t]\!]$ the margin $m_{i,\tau}$ for product i at time τ, its actual sales $v_{i,\tau}$ and the number $n_{i,\tau}$ of *impressions*, i.e., the times the product and the related margin has been observed (no matter if a customer bought it or not) during the time step τ. With this information, we can estimate the demand in a computationally efficient way using heteroscedastic GPs. In this model, given a target margin m, at time $t+1$ we can compute:

$$\widehat{\mu}_{i,t}(m) = \mathbf{k}_{i,t}(m)^\top \left(\mathbf{K}_{i,t} + \mathsf{diag}\left(\frac{\sigma^2}{\mathbf{n}_{i,t}}\right)\right)^{-1} \mathbf{y}_{i,t} \tag{6}$$

$$\widehat{\sigma}^2_{i,t}(m) = k(m,m) - \mathbf{k}_{i,t}(m)^\top \left(\mathbf{K}_{i,t} + \mathsf{diag}\left(\frac{\sigma^2}{\mathbf{n}_{i,t}}\right)\right)^{-1} \mathbf{k}_{i,t}(m) \tag{7}$$

with kernel matrix $\mathbf{K}_{i,t}$ and vectors $\mathbf{n}_{i,t}$, $\mathbf{y}_{i,t}$ and $\mathbf{k}_{i,t}$ are defined as:

$$\mathbf{K}_{i,t} \coloneqq [k(m_{i,j}, m_{i,h})]_{j,h \in [\![t]\!]}$$
$$\mathbf{n}_{i,t} \coloneqq [n_{i,1}, \dots, n_{i,t}]^\top$$
$$\mathbf{y}_{i,t} \coloneqq [y_{i,1}, \dots, y_{i,t}]^\top$$

[11] This is an upper bound to the variance of the Bernoulli, which corresponds to the real variance only if $\mathbb{E}[y_{i,t}|m_{i,t}] = 1/2$, the real variance may be lower but we cannot do better than upper bound it.

[12] While this can be mitigated to some extent through incremental matrix inversion (Lu and Shiou, 2002), the computational cost remains prohibitively high.

$$\mathbf{k}_{i,t}(m) \coloneqq [k(m_{i,1}, m), \dots, k(m_{i,t}, m)]^\top$$

where $y_{i,\tau} = v_{i,\tau}/n_{i,\tau}$, i.e., the number of sales divided by the number of impressions for product i at time τ (this value represents a realization of the demand). The rationale behind this model is that the quality of every sample $y_{i,\tau}$ depends on the number of impressions $n_{i,\tau}$, i.e., the number of times we observed the realization of the Bernoulli for such a margin. Indeed, we can add $n_{i,\tau}$ samples whose average is $y_{i,\tau}$ with variance σ^2 each or add just one sample with variance $\sigma^2/n_{i,\tau}$, we get the same result (Binois et al., 2018). In this way, the computational complexity of learning online in this setting can drop to $\mathcal{O}(T^3)$ (if we consider efficient matrix inversion updates).

Demand Learning via Heteroscedastic GPs — A More Efficient Solution. Even if the solution above improves the algorithm's efficiency, the computational complexity is still too high. We can improve it by using the same idea of considering an equivalent GP with heteroscedastic noise, but considering one sample for each margin only. To do so, we have to build a dataset in which we have sales statistics $\widetilde{v}_{i,t,j}$ and $\widetilde{n}_{i,t,j}$ for product i at time t for each margin m_j with $j \in [\![M]\!]$. Such counters are defined as:

$$\widetilde{v}_{i,t,j} = \sum_{\tau \in [\![t]\!]} v_{i,\tau} \cdot \mathbb{1}\{m_{i,\tau} = m_j\} \qquad \widetilde{n}_{i,t,j} = \sum_{\tau \in [\![t]\!]} n_{i,\tau} \cdot \mathbb{1}\{m_{i,\tau} = m_j\} \qquad \widetilde{y}_{i,t,j} = \frac{\widetilde{v}_{i,t,j}}{\widetilde{n}_{i,t,j}}.$$

Given such aggregated counters, we can design a new heteroscedastic GP (whose kernel matrix size will be bounded by the number of actions M) as:

$$\widehat{\mu}_{i,t}(m) = \widetilde{\mathbf{k}}_{i,t}(m)^\top \left(\widetilde{\mathbf{K}}_{i,t} + \mathsf{diag}\left(\frac{\sigma^2}{\widetilde{\mathbf{n}}_{i,t}}\right)\right)^{-1} \widetilde{\mathbf{y}}_{i,t} \tag{8}$$

$$\widehat{\sigma}^2_{i,t}(m) = k(m,m) - \widetilde{\mathbf{k}}_{i,t}(m)^\top \left(\widetilde{\mathbf{K}}_{i,t} + \mathsf{diag}\left(\frac{\sigma^2}{\widetilde{\mathbf{n}}_{i,t}}\right)\right)^{-1} \widetilde{\mathbf{k}}_{i,t}(m) \tag{9}$$

where matrix $\widetilde{\mathbf{K}}_{i,t} \coloneqq [k(m_j, m_h)]_{j,h \in [\![M]\!]}$ and vectors $\widetilde{\mathbf{n}}_{i,t}$, $\widetilde{\mathbf{y}}_{i,t}$ and $\widetilde{\mathbf{k}}_{i,t}$ are defined as:

$$\widetilde{\mathbf{n}}_{i,t} \coloneqq [\widetilde{n}_{i,t,1}, \dots, \widetilde{n}_{i,t,M}]^\top$$
$$\widetilde{\mathbf{y}}_{i,t} \coloneqq [\widetilde{y}_{i,t,1}, \dots, \widetilde{y}_{i,t,M}]^\top$$
$$\widetilde{\mathbf{k}}_{i,t}(m) \coloneqq [k(m_1, m), \dots, k(m_M, m)]^\top.$$

We highlight that the heteroscedastic GP of Equations (6) and (7) and the one of Equations (8) and (9) gives the same result in terms of $\widehat{\mu}_{i,t}(m)$ and $\widehat{\sigma}^2_{i,t}(m)$, for every $m \in \mathcal{M}$ (Binois et al., 2018).[13]

Optimistic Action Choice. Once at time t we have an estimate of $\widehat{\mu}_{i,t-1}(m)$ and $\widehat{\sigma}^2_{i,t-1}(m)$ for every $m \in \mathcal{M}$, we can make use of the optimistic bound of `IGP-UCB` (Chowdhury and Gopalan, 2017) to optimistically select the margin maximizing our objective function:

$$m_{i,t} \in \arg\max_{m \in \mathcal{M}} \widehat{f}_{i,t}(m),$$

where $\widehat{f}_{i,t}(m)$ is the optimistic counterpart of $f_i(\mathbf{m})$ presented in Equation (4):

$$\widehat{f}_{i,t}(m) = m\, c_i\, \widehat{d}^{\,\texttt{OPT}}_{i,t-1}(m)\, \widehat{n}_{i,t-1}, \tag{10}$$

where $\widehat{n}_{i,t-1}$ is an estimate of the average number of impressions per-stage calculated with data up to time $t-1$, and the estimated optimistic demand $\widehat{d}^{\,\texttt{OPT}}_{i,t-1}(m)$ is calculated as:

$$\widehat{d}^{\,\texttt{OPT}}_{i,t-1}(m) = \widehat{\mu}_{i,t-1}(m) + \beta_t \sqrt{\widehat{\sigma}^2_{i,t-1}(m)}, \tag{11}$$

with β_t chosen as in Equation (2), considering $\sigma^2 = 1/4$, as we need to consider the variance proxy of the original process, and using as maximum information gain (Binois et al., 2018):

$$\gamma_{i,t-1} = \frac{1}{2}\mathsf{ln}(\mathsf{det}(\mathbf{I} + \mathbf{D}_{i,t-1}\widetilde{\mathbf{K}}_{i,t-1}\mathbf{D}_{i,t-1})),$$

where $\mathbf{D}_{i,t-1} = \mathsf{diag}\left(\sigma^2/\widetilde{\mathbf{n}}_{i,t-1}\right)^{-1/2}$.

In this optimistic bound, the exploration term is applied just to the estimate of the demand curve, as we have no statistical uncertainty on terms m and c_i. Regarding $\widehat{n}_{i,t-1}$, the reason for which we do not need an exploration bonus is twofold. First, this term is just a rescaling factor and does not change the optimal margin we select. Second, we do not need any form of exploration on that, given that we will refine its estimate at any step, no matter the chosen margin.

[13] In the model above, margins $m_j \in \mathcal{M}$ whose corresponding $n_{i,t,j}$ is 0 should not be included in the GP model, as their variance will be undefined. In this case, no efficient matrix update is possible.

Computational Complexity. The regression model presented above significantly mitigates the computational complexity, as learning online requires $\mathcal{O}(TM^3)$ operations, where M is the number of margins, which means a strong improvement w.r.t. $\mathcal{O}(T^3)$ given that we usually have $M \ll T$.

6.2 Leader-Follower Sets

In this part, we consider the pricing of a single set of complementary products. Then, we will repeat the process for all the other sets. In this set $\mathcal{S}$, we assume to have one leader and one or more followers. With a little abuse of notation, we call the leader l and the followers f_i, for $i \in [\![F]\!]$ where F is the number of followers of the set under analysis. We have to compute, given a vector of margins $(m_l, m_{f_1}, \ldots, m_{f_F})$, the overall estimated return from all the products belonging to this set. This is given by:

$$\widehat{f}_{\mathcal{S},t}(m_l, m_{f_1}, \ldots, m_{f_F}) = \widehat{f}_{l,t}(m_l) + \sum_{i \in [\![F]\!]} \widehat{f}_{f_i,t}(m_l, m_{f_i}), \tag{12}$$

and our goal is to choose:

$$(m_{l,t}, m_{f_1,t}, \ldots, m_{f_F,t}) \in \underset{(m_l, m_{f_1}, \ldots, m_{f_F}) \in \mathcal{M}^{F+1}}{\arg\max} \widehat{f}_{\mathcal{S},t}(m_l, m_{f_1}, \ldots, m_{f_F}). \tag{13}$$

This means we want to jointly optimize the prices we choose for the leader and the followers. In this formulation, the demand of the leader depends only on the margin of the leader m_l, while the demand of a follower f_i depends on the margin of such a follower m_{f_i} and the one of its leader m_l. As such, the optimistic objective function of the leader $\widehat{f}_{l,t}(m_l)$ is defined as:

$$\widehat{f}_{l,t}(m_l) = c_l \; m_l \; \widehat{d}^{\,\texttt{OPT}}_{l,t-1}(m_l) \; \widehat{n}_{l,t-1},$$

where $\widehat{n}_{l,t-1}$ is the average number of impressions we observed (per stage) up to stage $t-1$ for leader product l, and the optimistic demand is:

$$\widehat{d}^{\,\texttt{OPT}}_{l,t-1}(m_l) = \widehat{\mu}_{l,t-1}(m_l) + \beta_t \sqrt{\widehat{\sigma}^2_{l,t-1}(m_l)}.$$

On the other hand, $\widehat{f}_{f_i,t}(m_l, m_{f_i})$ is defined as:

$$\widehat{f}_{f_i,t}(m_l, m_{f_i}) = c_{f_i} \; m_{f_i} \; \widehat{d}^{\,\texttt{OPT}}_{f_i,t-1} \; \widehat{n}_{f_i,t-1},$$

where $\widehat{d}^{\,\texttt{OPT}}_{f_i,t-1}$ is the equivalent optimistic demand for the follower f_i, defined as:

$$\widehat{d}^{\,\texttt{OPT}}_{f_i,t-1} = \; \widehat{p}_{l,f_i,t-1} \, \widehat{\mu}_{l,t-1}(m_l) \; \widehat{\overline{d}}^{\,\texttt{OPT}}_{f_i,t-1} + ((1 - \widehat{p}_{l,f_i,t-1}) + \widehat{p}_{l,f_i,t-1}(1 - \widehat{\mu}_{l,t-1}(m_l))) \; \widehat{\widetilde{d}}^{\,\texttt{OPT}}_{f_i,t-1},$$

where $\widehat{p}_{l,f_i,t-1}$ is the sample-based estimate up to time $t-1$ of $\overline{p}_{l,f_i}$, i.e., the probability that a user observes f_i when leader l is in the basket, and the demands are defined as:

$$\widehat{\overline{d}}^{\,\texttt{OPT}}_{f_i,t-1} = \; \widehat{\overline{\mu}}_{f_i,t-1}(m_{f_i}) + \beta_t \sqrt{\widehat{\overline{\sigma}}^2_{f_i,t-1}(m_{f_i})} \qquad \text{and} \qquad \widehat{\widetilde{d}}^{\,\texttt{OPT}}_{f_i,t-1} = \; \widehat{\widetilde{\mu}}_{f_i,t-1}(m_{f_i}) + \beta_t \sqrt{\widehat{\widetilde{\sigma}}^2_{f_i,t-1}(m_{f_i})},$$

where $\widehat{\overline{\mu}}_{f_i,t-1}(m_{f_i})$ and $\widehat{\overline{\sigma}}^2_{f_i,t-1}(m_{f_i})$ are the estimators related to the demand computed using only the impressions and sales generated for product f_i when leader l is in the same basket, and $\widehat{\widetilde{\mu}}_{f_i,t-1}(m_{f_i})$ and $\widehat{\widetilde{\sigma}}^2_{f_i,t-1}(m_{f_i})$ the ones related demand computed using the sales and impressions generated when product l is not in the same basket. The rationale is to trade off the sales we can expect for the follower, taking into account both the probability that a customer concurrently sees the leader and the probability that such a customer buys the leader product. All the demands are estimated using the model proposed in Section 6.1, taking care of creating a new dataset in which we divide the impressions, and the sales generated must be stored together with the information about the products bought together.

Computational Complexity. These regression models present the same computational complexity order per-product as that of Section 6.1, i.e., $\mathcal{O}(TM^3)$. However, once we have computed all models for all products in the cluster, we also need to search for the optimal margin vector. To maintain an acceptable level of computational complexity, the search for the optimal solution (Equation 13) within the given space must be carefully structured. This necessity arises because the search space lacks convexity/concavity properties, which preclude the use of gradient-based optimization methods and instead roughly require a grid-search approach, leading to a complexity in the order of $\mathcal{O}(M^{F+1})$.[14] If we accept that all followers share the same margin (i.e., $m_{f_1,t} = m_{f_2,t} = \cdots = m_{f_F,t}$), the complexity of this search is limited to $\mathcal{O}(M^2)$ for each set (this will introduce a suboptimality in the maximum revenue reachable).

[14]The grid search can be substituted with more elaborated heuristic approaches (see, e.g., Vikhar, 2016).

7 Algorithm: Mining Complementarity Relations

In this section, we focus on identifying complementary relations among the products we aim to price. These relations, in general, may involve numerous products with intricate interaction dynamics, posing significant challenges for effective modeling. Detecting general complementary relations requires navigating an extensive hypothesis space, which can lead to issues such as model variance and sensitivity to noise. Moreover, our goal extends beyond merely identifying products that are frequently purchased together. Instead, we aim to discover products whose optimal joint pricing maximizes overall value, surpassing the benefits of pricing them independently. To mitigate such a high complexity, we focus specifically on *leader-follower* complementary relations. In this framework, each product is classified as either (i) a leader to one or more products, (ii) a follower of another product, or (iii) not involved in any complementary relation (independent).[15]

We begin by presenting the concept intuitively, providing a foundation for understanding. Then, we introduce a practical example that helps transition to a formal definition of the problem.

The goal is to determine the optimal partitioning of a set of P products to maximize revenue. To accomplish this, using the models introduced in Section 6, we construct a real-valued $P \times P$ matrix $\mathbf{V} = [v_{i,j}]_{i,j \in [\![P]\!]}$ that represents the optimistic revenues for all possible leader-follower pairs. In this matrix, on-diagonal elements $v_{i,i}$ represent the highest possible revenue obtainable when product i is priced independently (see Section 6.1), i.e., $v_{i,i} = \max_{m \in \mathcal{M}} \widehat{f}_{i,t}(m)$ (Equation 10). On the other hand, off-diagonal elements $v_{i,j}$ (with $i \neq j$) in the matrix corresponds to the maximum estimated revenue from treating product i as the leader and product j as the follower, with their margins jointly optimized (see Section 6.2) to maximize the objective function (Equation 12) minus the value of the leader $v_{i,i}$ if priced independently (to avoid to count it several times if it is leader of more than one follower).[16]

With this matrix in place, we want to identify the most profitable combinations of products. With this aim, we define a second $P \times P$ binary-valued matrix $\mathbf{X}$. Here, each element $x_{i,j}$ indicates whether product i is selected as the leader of product j ($x_{i,j} = 1$) or not ($x_{i,j} = 0$). To determine the optimal solution, we formulate a *binary programming* problem that maximizes the sum of all the elements of the matrix resulting from the Hadamard (element-wise) product between the matrices $\mathbf{V}$ and $\mathbf{X}$, subject to a set of constraints governing the placement of zeros and ones in $\mathbf{X}$ we will present later.

To better understand this solution, before formalizing the problem, we present an example of what we expect as output.

Example. *Consider a set of $P = 6$ products. Suppose we compute, using our models, the expected return of all the couples, and we get a matrix $\mathbf{V}$ as follows:*

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 10 & 20 & 30 & 40 & 50 & 60 \\ 10 & 20 & 30 & 40 & 50 & 60 \\ \mathbf{30} & \mathbf{50} & \mathbf{30} & 40 & 50 & 60 \\ 10 & 20 & 30 & \mathbf{40} & \mathbf{80} & 60 \\ 10 & 20 & 30 & 40 & 50 & 60 \\ 10 & 20 & 30 & 40 & 50 & \mathbf{60} \end{bmatrix} \end{array} \tag{14}$$

One optimal solution involves selecting C as the leader for products A and B, selecting D as the leader for E, and leaving F independent. The corresponding binary matrix $\mathbf{X}$ is:

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \end{array} \tag{15}$$

[15] For computational efficiency, a product designated as a follower cannot simultaneously act as a leader for other products, as this would significantly increase the complexity of determining the optimal prices, given that all the generated subsets will not be independent and we cannot perform an independent optimization of the different subsets.

[16] From the practical perspective, it may be useful to subtract a small penalty factor to off-diagonal elements to avoid the detection of *ghost* relations, as the joint performance is lower bounded by the independent performance, and the noise may easily lead to detect false complementary relations.

In this illustrative example, it is easy to observe that the estimated reward is given by the sum of the values of the matrix in Equation (14) *in the positions where in Equation* (15) *we have the terms* 1*. Given that, the overall estimated return can be seen as the sum of all the elements in the matrix* $\mathbf{V} \circ \mathbf{X}$*, where* $\circ$ *is the Hadamard product.* □

After presenting the example, we formalize the problem as an integer programming problem with linear constraints (Wolsey, 2020). The objective function aims to maximize the sum of all elements in the matrix obtained from the element-wise product:

$$\sum_{i \in [\![P]\!]} \sum_{j \in [\![P]\!]} x_{i,j} \; v_{i,j},$$

where $v_{i,j}$ are positive real-valued elements, and $x_{i,j}$ are the binary variables over which we optimize. If product i is a leader (i.e., $i \in \mathcal{L}$), the following conditions (constraints) must hold:

$$\textstyle\sum_{j \in [\![P]\!] \setminus \{i\}} x_{i,j} \geq 1, \qquad \forall i \in \mathcal{L} \tag{16}$$

$$\textstyle\sum_{j \in [\![P]\!] \setminus \{i\}} x_{j,i} = 0, \qquad \forall i \in \mathcal{L} \tag{17}$$

$$x_{i,i} = 1, \qquad \forall i \in \mathcal{L} \tag{18}$$

where Equation (16) ensures that the leader has at least one follower, Equation (17) imposes that it has no leaders (it must not be a follower of any other product), and Equation (18) ensures we consider the value of the leader product in the formulation.

On the other hand, if product i is a follower (i.e., $i \in \mathcal{F}$) or independent (i.e., $i \in \mathcal{I}$), it has to satisfy the following conditions:

$$\textstyle\sum_{j \in [\![P]\!]} x_{j,i} = 1, \qquad \forall i \in \mathcal{F} \cup \mathcal{I} \tag{19}$$

$$\textstyle\sum_{j \in [\![P]\!] \setminus \{i\}} x_{i,j} = 0, \qquad \forall i \in \mathcal{F} \cup \mathcal{I} \tag{20}$$

where Equation (19) ensures that each follower is linked to exactly one product (including itself if independent), and Equation (20) ensures that a follower does not lead any other product.

To enforce that each product is *either* a leader or a follower/independent, we impose the following condition:

$$(\text{Eq. (16)} \;\wedge\; \text{Eq. (17)} \;\wedge\; \text{Eq. (18)}) \;\oplus\; (\text{Eq. (19)} \;\wedge\; \text{Eq. (20)}) \tag{21}$$

where $\wedge$ and $\oplus$ represent the logical *and* and the *exclusive or*, respectively. This will ensure that $\mathcal{L} \cup \mathcal{F} \cup \mathcal{I} = \mathcal{P}$ and their (even pair-wise) intersections are the empty set. To implement the *exclusive or*, we consider P additional binary optimization variables $(z_i)_{i \in [\![P]\!]}$ into our integer programming problem. As an implementation choice, a variable z_i will be equal to 0 if the corresponding product is a leader and 1 otherwise. With these additional quantities, we can define our optimization problem by implementing the *exclusive or* using linear constraints. The integer programming problem is then defined as:

$$\max_{\substack{(x_{i,j})_{i,j \in [\![P]\!]} \\ (z_i)_{i \in [\![P]\!]}}} \sum_{i \in [\![P]\!]} \sum_{j \in [\![P]\!]} x_{i,j} \; v_{i,j} \tag{22}$$

$$\begin{aligned}
\text{subject to:} \quad & 0 \leq x_{i,j} \leq 1, && \forall i,j \in [\![P]\!] && (23)\\
& 0 \leq z_i \leq 1, && \forall i \in [\![P]\!] && (24)\\
& \textstyle\sum_{j \in [\![P]\!] \setminus \{i\}} x_{i,j} \geq 1 - P z_i, && \forall i \in [\![P]\!] && (25)\\
& \textstyle\sum_{j \in [\![P]\!] \setminus \{i\}} x_{j,i} \leq 0 + P z_i, && \forall i \in [\![P]\!] && (26)\\
& x_{i,i} \leq 1 + P z_i, && \forall i \in [\![P]\!] && (27)\\
& x_{i,i} \geq 1 - P z_i, && \forall i \in [\![P]\!] && (28)\\
& \textstyle\sum_{j \in [\![P]\!]} x_{j,i} \geq 1 - P(1 - z_i), && \forall i \in [\![P]\!] && (29)\\
& \textstyle\sum_{j \in [\![P]\!]} x_{j,i} \leq 1 + P(1 - z_i), && \forall i \in [\![P]\!] && (30)\\
& \textstyle\sum_{j \in [\![P]\!] \setminus \{i\}} x_{i,j} \leq 0 + P(1 - z_i), && \forall i \in [\![P]\!] && (31)
\end{aligned}$$

Some comments are in order. First, we observe that now the optimization variables are $P^2 + P$, even if the ones involved in the objective function still remain P^2. Second, we use z_i variables to determine the type of product and the constraints to be applied. In Constraints (25), (26), (27) and (28) referring to leader products, we add/subtract (given the type of constraint) a term $P\, z_i$ to *lift* the constraints in the case in which the product under analysis is a follower/independent.[17] For equality constraints $x = Z$, we impose these constraints by imposing that $x \leq Z$ and $x \geq Z$.[18] In Constraints (29),

[17] For simplicity and readability, terms $P\, z_i$ and $P\,(1 - z_i)$ are used consistently, although smaller values could suffice to lift the constraints in some cases.

[18] In some cases, we just need one constraint as the other is already satisfied by construction.

(30) and (31), we do the opposite, and to model the case in which products are followers/independent, and the term $P(1 - z_i)$ lift the constraints in the case we are facing a leader.

It is important to note that when there is more than one follower for a single leader, this optimization problem yields an *optimistic estimate* of the impact of joint product pricing. In fact, given a leader l, the margin m_l that maximizes the overall revenue for a follower f_i may be suboptimal for another follower f_j. Addressing this issue rigorously would significantly increase the computational complexity. Therefore, we accept this slight suboptimality to maintain a reasonable computational burden.

Computational Complexity. Finding the optimal solution for binary programming problems is known to be NP-hard in general (Karp, 1972; Schrijver, 1998; Papadimitriou and Steiglitz, 1998; Wolsey and Nemhauser, 1999). Given that, we should make considerations and adopt strategies for reducing the computational burden. First, we can determine a rough split of the products, generating disjoint subsets. We should then run the algorithm multiple times, once on each subset, which significantly reduces the computational complexity of this pipeline step. Moreover, this step should not be performed at every $t \in [\![T]\!]$. The reason for avoiding running this procedure at every step is twofold. First, to reduce computational complexity, as this operation is computationally heavy. Second, to prevent frequent changes to product pairings during learning, as this could negatively impact pricing stability (see Section 8).

8 Experimental Validation

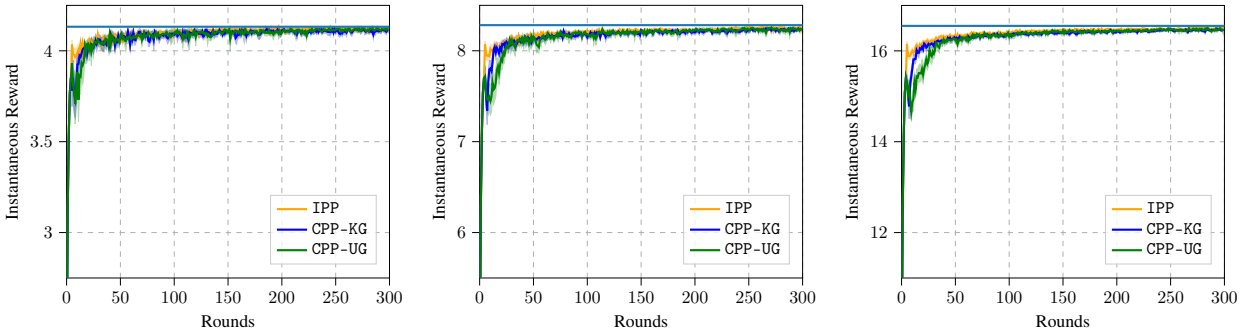
In this section, we empirically assess the effectiveness of the proposed `Complementary Products Pricing` (`CPP`) algorithm through a set of controlled simulations. The goal of this experimental campaign is twofold. First, we aim to evaluate the capability of `CPP` to learn optimal pricing strategies in environments with different types of product interactions. Second, we seek to compare its performance with suitable baselines to isolate the contribution of each algorithmic component. We start by introducing the algorithms under comparison, including two `CPP` variants and a baseline which disregards product interdependencies. Then, we design two distinct experimental scenarios. First, in Section 8.1, we consider the case in which product demands are independent. Then, in Section 8.2, we consider the case where products exhibit complementary relations which may influence the optimal behavior. Finally, in Section 8.3, we analyze the computational performance of all algorithms to assess their scalability and practical viability. The code used in this section is available at `https://github.com/marcomussi/ComplementaryProductsPricing`.

Evaluated Algorithms and Baseline. We consider two versions of the `CPP` algorithm and a coherent baseline. The first variant, `CPP-KG` (*known graph*), considers an implementation of `CPP` in which we assume that the complementarity graph between products is known in advance. In this version, the algorithm focuses exclusively on learning demand curves and optimizing prices given a fixed interaction structure. This configuration allows us to isolate and assess the effectiveness of the demand modeling and pricing optimization stages in a simpler scenario where the relational structure does not need to be inferred. The second variant, `CPP-UG` (*unknown graph*), represents the complete version of the algorithm. In this version, neither the structure of the complementarity graph nor the demand functions are known. The algorithm must therefore learn the underlying relations between the products by solving the integer programming problem described in Section 7, while simultaneously learning the demand models and updating pricing strategies online. This setup enables the assessment of the algorithm's capacity to autonomously discover profitable complementary relations and exploit them for coordinated pricing. We run the `CPP-UG` by recomputing the graph through the integer programming problem at every step, even though it can be run at regular intervals to reduce computational complexity. To provide a meaningful point of comparison, we also include an independent pricing baseline, denoted as `IPP` (`Independent Products Pricing`). This algorithm employs the same learning framework as `CPP` for modeling demand and optimizing prices, but treats each product as independent, ignoring any potential cross-product relations (following the model presented in Section 6.1). By comparing `CPP-KG` and `CPP-UG` against `IPP`, we can quantify the advantage of explicitly modeling interactions among complementary products and evaluate the impact of structure learning on both performance and computational cost.

8.1 Independent Environment

We start by analyzing the learning performances of the algorithms in the case of independent demand curves.

Setting. We consider scenarios involving different sets of products $\mathcal{P}$ with varying cardinalities $P \in \{5, 10, 20\}$. For each product, we define a discrete set of possible margins $\mathcal{M} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, which can be applied to all the products. At each round, we simulate 100 impressions per product, assuming a concurrence probability $\overline{p}_{i,j} = 1$ for every $i, j \in \mathcal{P}$ for the sake of simplicity in the presentation of the results. Demand curves are synthetically generated under the constraint of being monotonically decreasing with respect to the margin, reflecting standard economic



(a) 5 Products. (b) 10 Products. (c) 20 Products.

Figure 2: Independent demands environments (30 trials, mean $\pm$ 95% C.I.).

assumptions (this assumption is not needed for the correct execution of the algorithms). The algorithms' performance is evaluated in terms of the obtained rewards. Specifically, we report the instantaneous rewards (mean $\pm$ 95% confidence interval), while the optimum is represented as a horizontal line in the figures.[19] The optimal value is computed via Monte Carlo simulations using 10^5 samples for each action to ensure accurate estimation.

Results. The results are presented in Figure 2. In this scenario, regardless of the number of products considered, all algorithms achieve optimal performance, confirming that the demand learning model behaves as expected. In particular, the independent pricing algorithm (`IPP`) exhibits faster convergence to the optimal reward. This outcome is reasonable, as `IPP` represents the simplest model among the three, yet it retains sufficient representational capacity to learn the optimal action correctly in this simpler scenario. This behavior becomes even more evident as the number of products increases. In Figure 2b, which considers 10 products, and in Figure 2c, which extends the analysis to 20 products, we observe that both competing algorithms are outperformed by `IPP` in the first few rounds of the learning process. In particular, `CPP-UG` shows, as expected, the worst performance at the beginning as it has to explore more to learn also the graph structure.

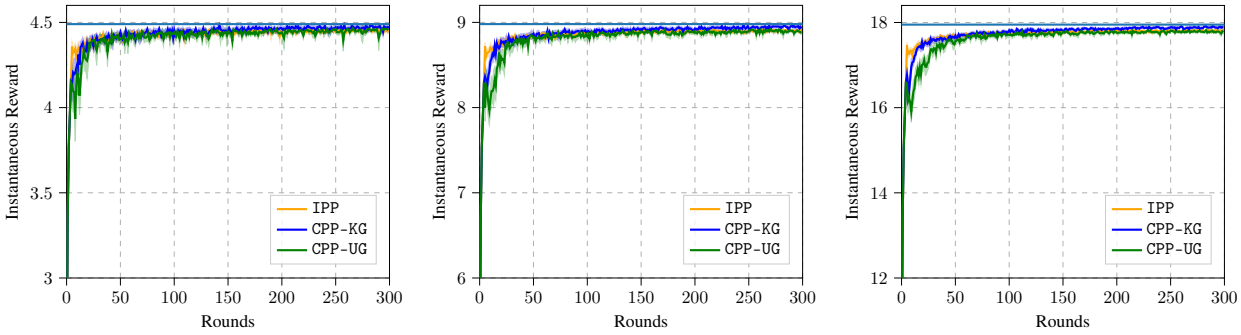
8.2 Complementary Environments

We now move to the scenario in which we consider product interactions influencing demands.

Setting. We consider a setting in which product interactions arise between demand curves following a leader–follower structure. As in the previous experiment, we analyze scenarios involving different sets of products $\mathcal{P}$ with cardinalities $P \in \{5, 10, 20\}$. For each product, we define a discrete set of possible margins $\mathcal{M} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, which can be applied uniformly across all products. At each round of the algorithm, we simulate 100 impressions per product, assuming a concurrence probability $\overline{p}_{i,j} = 1$ for every $i, j \in \mathcal{P}$. When a follower product is proposed after a customer purchases its leader, we model the increase in demand as a multiplicative factor applied to the base demand (i.e., the demand the product would exhibit in the absence of the leader). Two levels of demand interaction are considered, corresponding to increases of 10% and 40%, to emulate two levels of interaction. As in the previous experiment, the performance of the proposed methods is evaluated in terms of instantaneous rewards, and the optimal value is estimated via Monte Carlo simulations with 10^5 samples per action.

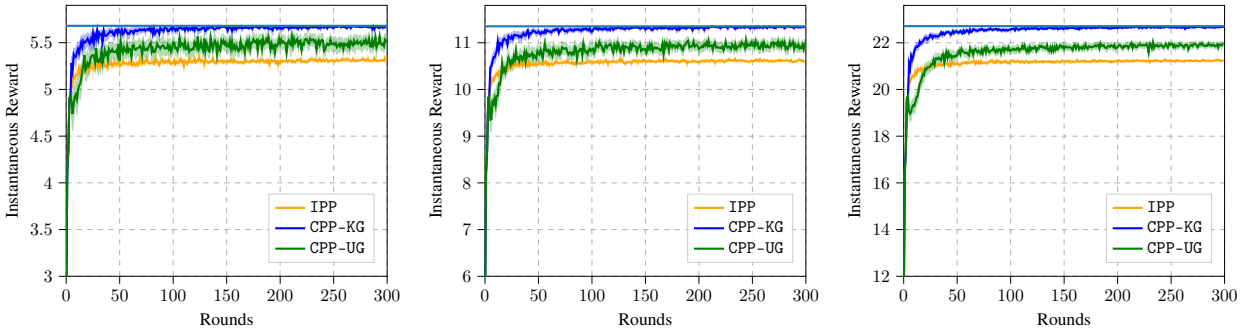
Results. In Figures 3 and 4, we show the results for the complementary environment for the demand increases of 10% and 40%, respectively. We can observe from Figure 3 that, across all product sets, when the increment in the follower's demand is mild (10%), all algorithms exhibit comparable performance and successfully reach the optimal reward. The same considerations discussed in the previous case of independent items also apply here, in terms of convergence speed and behavior, as the number of products increases. In contrast, Figure 4, where the follower's demand increases significantly (40%), illustrates a markedly different pattern. In this setting, where the pull effect becomes substantial, the independent algorithm converges rapidly but only to a suboptimal reward, while the algorithms exploiting product complementarities achieve higher performance. In particular, the algorithm that does not need to learn the complementary graph structure reaches the optimum, whereas the version that simultaneously learns this structure also converges, but at a slower rate. A detailed analysis suggests that this slower convergence is due to an

[19] We report the instantaneous reward instead of the cumulative one, as in this way it is simpler to visualize when we reach the optimum.



(a) 5 Products. (b) 10 Products. (c) 20 Products.

Figure 3: Complementary demand environments with mild complementarity dynamics (30 trials, mean $\pm$ 95% C.I.).



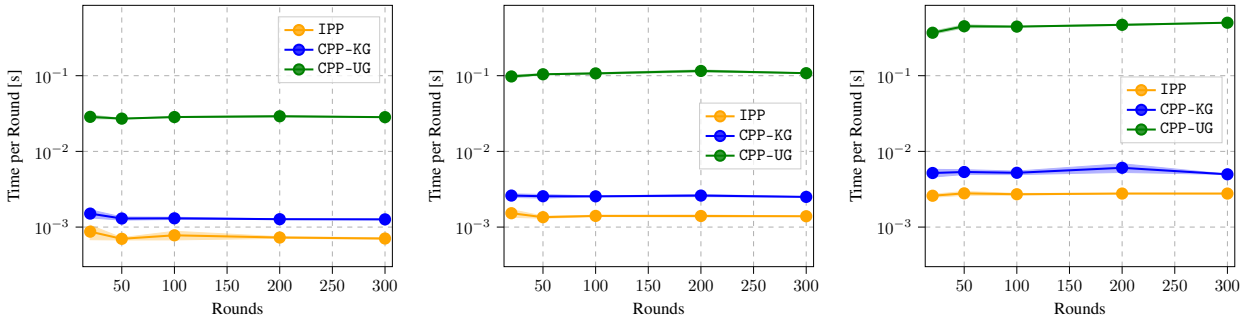
(a) 5 Products. (b) 10 Products. (c) 20 Products.

Figure 4: Complementary demands environments (30 trials, mean $\pm$ 95% C.I.).

over-exploration required for learning the graph of complementarity relationships. The two figures thus reveal distinct behaviors, raising the question of why, in the mild increment scenario, the algorithms that exploit complementarities do not show any advantage. We observed that the reason lies in the nature of the optimal actions. When the increment is small, the optimal solution obtained for each product individually coincides with that obtained by jointly optimizing over the cluster of complementary items. Conversely, in the high-increment scenario, the optimal joint action for the leader–follower group differs from the one derived by treating products independently. In particular, we observe that reducing the margin of the leader decreases its individual reward, but simultaneously induces a boost in the follower's demand, ultimately leading to a higher overall reward for the cluster.

8.3 Computational Performances

The algorithm proposed in this paper has been designed to be lightweight and to overcome several limitations induced by the adopted learning model (i.e., GPs), which generally scale poorly with respect to the number of samples. As a result, all experiments can be efficiently executed on a standard consumer laptop. Specifically, all tests were conducted on a MacBook Pro equipped with an Apple Silicon M4 processor and 24 GB of RAM. All experiments were run on a single core, although the algorithm can be partially parallelized. Figure 5 reports the normalized running time of the algorithm for 5 (Figure 5a), 10 (Figure 5b), and 20 (Figure 5c) products for different target horizon $T \in \{20, 50, 100, 200, 300\}$, corresponding to the circle marks in the figures. All plots share the same scale to facilitate comparison. Each figure shows the running time of a single execution, normalized by the number of rounds. We observe that, thanks to the heteroscedastic formulation introduced, the algorithm does not suffer from scalability issues with respect to the time horizon. A closer inspection of the results reveals that both `IPP` and `CPP-KG` scale linearly with the number of products. Conversely, the version with an unknown graph structure, `CPP-UG`, is slightly more computationally demanding; however, it still exhibits a very reasonable running time. It is worth noting that the reported computational request of `CPP-UG` corresponds to a worst-case scenario, as here the integer optimization problem is solved at every step. In practice, this optimization can be performed less frequently, which would bring the computational performance of `CPP-UG` close to that of `CPP-KG`.



(a) 5 Products. (b) 10 Products. (c) 20 Products.

Figure 5: Running time of the algorithms normalized by the number of rounds (10 trials, mean $\pm$ 95% C.I.).

9 Related Works

In this section, we review the main contributions related to dynamic pricing, with a focus on studies employing machine learning approaches and on research concerning complementary products. For a comprehensive survey on dynamic pricing methodologies, readers are referred to the work of Den Boer (2015).

Among the available ML frameworks, *multi-armed bandits* (MABs) have emerged as one of the most widely adopted techniques for dynamic pricing, as they effectively address the exploration-exploitation trade-off and ensure efficient sample utilization. The foundational study that introduced the concept of active learning of the demand curve in a pricing context was conducted by Rothschild (1974). Subsequent research expanded on this idea in various ways. Kleinberg and Leighton (2003) approached the challenge of continuous demand functions by discretizing price values and providing formal performance guarantees. Parametric versions of the demand function were later proposed by Besbes and Zeevi (2009) and Broder and Rusmevichientong (2012). Besbes and Zeevi (2015) imposed monotonicity constraints on the demand model and demonstrated the effectiveness of linear models for demand estimation. Building on these ideas, Shukla et al. (2019) developed a pricing algorithm that integrates customer-specific characteristics and assumes a monotonic willingness-to-pay function. Mussi et al. (2023) propose a MAB algorithm to learn demand curves integrating a data-driven approach to volume discounts.

9.1 Complementary Products

Among studies that explicitly consider complementarities in pricing, (Mulhern and Leone, 1991) is the earliest contribution empirically demonstrating that adopting multi-product pricing strategies can improve retail performance, although their focus was on product bundling rather than online learning. After that, several works tried to exploit complementary relations for pricing. Sun et al. (2015) applied association rule mining to uncover implicit item-to-item relationships while accounting for their asymmetric nature. Zhao et al. (2017) constructed a directed graph of complementary products using natural language processing and the Skip-Gram model (Mikolov et al., 2013) to generate product embeddings. Wang et al. (2018) combined catalog metadata with transaction information to construct a graph capturing both substitutability and complementarity, applying category and path constraints to identify new connections. Feng et al. (2018) modeled substitutability and complementarity within a discrete choice framework, but their approach lacks the adaptability and uncertainty modeling provided by ML techniques. Stradi et al. (2024) study the problem of dynamic pricing for complementary items shown sequentially to customers under uncertainty and sales constraints. It models the task as a constrained Markov decision process and develops a primal-dual online algorithm. Then, there are approaches, such as (McAuley et al., 2015), that rely heavily on textual data to infer complementarity, which may not accurately reflect the actual purchasing dynamics within a specific e-commerce environment, where transaction data provides a more accurate signal for pricing. Additionally, most of these studies require extensive datasets, including textual descriptions, product reviews, or co-view information, whereas the present work assumes access only to transaction data to infer complementary relations. Hao et al. (2020) developed a graph-based representation learning model that integrates textual and behavioral data to recommend diverse, complementary items, while Xu et al. (2020) leveraged co-view data and product descriptions to build knowledge graph embeddings. Although these methods effectively capture relationships between products, their primary goal differs from the focus of this paper. In particular, they do not aim to simplify the graph structure or jointly optimize pricing decisions to exploit relations that generate a higher revenue/profit. On the theoretical perspective, Mussi et al. (2025) demonstrate that synchronization among actions is crucial for achieving optimal performance in sequential decision-making problems.

10 Discussion and Conclusions

In this paper, we addressed the problem of finding the optimal pricing strategy for products exhibiting complementary relations. We began by formalizing the problem and defining the associated learning objective. Then, we introduced `Complementary Products Pricing`, a novel three-stage algorithm designed for online learning in such environments. The algorithm first aggregates substitutable products using the available information. Subsequently, it identifies complementary relations through a specifically crafted integer programming formulation and estimates the corresponding demand curves using a statistically and computationally efficient implementation of Gaussian Processes that leverages heteroscedasticity to reduce computational complexity. To the best of our knowledge, this is the first complete pipeline that jointly identifies complementary products and optimizes pricing decisions while explicitly accounting for computational constraints. We conducted an extensive experimental campaign to validate the soundness and effectiveness of our approach. The results show that knowing the structure of complementarities significantly simplifies the learning process, leading to faster convergence and more stable performance. When such relations are unknown, estimating them through our proposed approach provides a clear improvement over treating products as independent. However, some improvement can still be beneficial to enhance the stability of the integer programming stage and mitigate the effects of over-exploration. Overall, the experiments confirm the robustness of the proposed framework and its ability to adapt to different levels of product interaction.

Future Works. In the short term, we plan to analyze the computational complexity of the integer programming component presented in Section 7, and to design heuristic or approximate solutions improving stability and scalability. In the longer term, we aim to integrate a model of customer rationality into the framework, enabling a more realistic representation of consumer behavior and a richer set of strategic interactions in dynamic pricing environments.

References

Arnoud V. Den Boer. Dynamic pricing and learning: Historical origins, current research, and new directions. *Surveys in Operations Research and Management Science*, 20(1):1–18, 2015.

Ruiliang Yan and Subir Bandyopadhyay. The profit benefits of bundle pricing of complementary products. *Journal of Retailing and Consumer Services*, 18(4):355–361, 2011.

Walter Nicholson and Christopher Snyder. *Microeconomic theory: basic principles and extensions*. Cengage Learning, 2017.

Cenk Kocas, Koen Pauwels, and Jonathan D. Bohlmann. Pricing Best Sellers and Traffic Generators: The Role of Asymmetric Cross-selling. *Journal of Interactive Marketing*, 41:28–43, 2018.

Hang Yu, Lester Litchfield, Thomas Kernreiter, Seamus Jolly, and Kathryn Hempstalk. Complementary Recommendations: A Brief Survey. In *International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*, pages 73–78, 2019.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Product Knowledge Graph Embedding for E-commerce. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*, pages 672–680, 2020.

Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer, 2004.

Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. Lower bounds on regret for noisy gaussian process bandit optimization. In *Annual Conference on Learning Theory (COLT)*, volume 65 of *Proceedings of Machine Learning Research*, pages 1723–1742. PMLR, 2017.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1015–1022. Omnipress, 2010.

Sayak R. Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 844–853. PMLR, 2017.

Sattar Vakili, Kia Khezeli, and Victor Picheny. On information gain and regret bounds in gaussian process bandits. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130 of *Proceedings of Machine Learning Research*, pages 82–90. PMLR, 2021.

Marco Mussi, Simone Drago, and Alberto M. Metelli. Open problem: Tight bounds for kernelized multi-armed bandits with bernoulli rewards. *CoRR*, abs/2407.06321, 2024.

Tzon-Tzer Lu and Sheng-Hua Shiou. Inverses of 2×2 block matrices. *Computers & Mathematics with Applications*, 43(1-2):119–129, 2002.

Gurpreet Singh, Harjot Kaur, and Amitpal Singh. Dropshipping in e-commerce: A perspective. In *Proceedings of the International Conference on E-Business, Management and Economics (ICEME)*, page 7–14. ACM, 2018.

Josef Bauer and Dietmar Jannach. Optimal pricing in e-commerce based on sparse and noisy data. *Decision Support Systems*, 106:53–63, 2018.

Mila Nambiar, David Simchi-Levi, and He Wang. Dynamic Learning and Pricing with Model Misspecification. *Management Science*, 65(11):4980–5000, 2019.

Adel Javanmard, Hamid Nazerzadeh, and Simeng Shao. Multi-product dynamic pricing in high-dimensions with heterogeneous price sensitivity. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2652–2657. IEEE, 2020.

Marco Mussi, Gianmarco Genalti, Francesco Trovò, Alessandro Nuara, Nicola Gatti, and Marcello Restelli. Pricing the Long Tail by Explainable Product Aggregation and Monotonic Bandits. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 3623–3633. ACM, 2022.

Sridhar Moorthy and Ivan P. L. Png. Market Segmentation, Cannibalization, and the Timing of Product Introductions. *Management Science*, 38(3):345–359, 1992.

John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

John F. Nash. Non-cooperative games. *Annals of Mathematics*, 54:286–295, 1951.

Mickael Binois, Robert B. Gramacy, and Mike Ludkovski. Practical heteroscedastic gaussian process modeling for large simulation experiments. *Journal of Computational and Graphical Statistics*, 27(4):808–821, 2018.

Pradnya A. Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *International conference on global trends in signal processing, information computing and communication (ICGTSPICC)*, pages 261–265. IEEE, 2016.

Laurence A. Wolsey. *Integer programming*. John Wiley & Sons, 2020.

Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

Laurence A. Wolsey and George L. Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 1999.

Michael Rothschild. A two-armed bandit theory of market pricing. *Journal of Economic Theory*, 9(2):185–202, 1974.

Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: bounds on regret for online posted-price auctions. In *IEEE Symposium on Foundations of Computer Science*, pages 594–605. IEEE, 2003.

Omar Besbes and Assaf Zeevi. Dynamic Pricing Without Knowing the Demand Function: Risk Bounds and Near-Optimal Algorithms. *Operations Research*, 57(6):1407–1420, 2009.

Josef Broder and Paat Rusmevichientong. Dynamic pricing under a general parametric choice model. *Operations Research*, 60(4):965–980, 2012.

Omar Besbes and Assaf Zeevi. On the (surprising) sufficiency of linear models for dynamic pricing with demand learning. *Management Science*, 61(4):723–739, 2015.

Naman Shukla, Arinbjörn Kolbeinsson, Ken Otwell, Lavanya Marla, and Kartik Yellepeddi. Dynamic Pricing for Airline Ancillaries with Customer Context. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2174–2182. ACM, 2019.

Marco Mussi, Gianmarco Genalti, Alessandro Nuara, Francesco Trovò, Marcello Restelli, and Nicola Gatti. Dynamic pricing with volume discounts in online settings. In *Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pages 15560–15568. AAAI Press, 2023.

Francis J. Mulhern and Robert P. Leone. Implicit Price Bundling of Retail Products: A Multiproduct Approach to Maximizing Store Profitability. *Journal of Marketing*, 55(4):63–76, 1991.

Zhu Sun, Guibing Guo, and Jie Zhang. Exploiting Implicit Item Relationships for Recommender Systems. In *User Modeling, Adaptation and Personalization*, Lecture Notes in Computer Science, pages 252–264. Springer, 2015.

Tong Zhao, Julian McAuley, Mengya Li, and Irwin King. Improving recommendation accuracy using networks of substitutable and complementary products. In *International Joint Conference on Neural Networks (IJCNN)*, pages 3649–3655. IEEE, 2017.

Tomás Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013.

Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. A Path-constrained Framework for Discriminating Substitutable and Complementary Products in E-commerce. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 619–627. ACM, 2018.

Guiyun Feng, Xiaobo Li, and Zizhuo Wang. On substitutability and complementarity in discrete choice models. *Operations Research Letters*, 46(1):141–146, 2018.

Francesco E. Stradi, Filippo Cipriani, Lorenzo Ciampiconi, Marco Leonardi, Alessandro Rozza, and Nicola Gatti. A primal-dual online learning approach for dynamic pricing of sequentially displayed complementary items under sale constraints. *CoRR*, abs/2407.05793, 2024.

Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring Networks of Substitutable and Complementary Products. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794. ACM, 2015.

Junheng Hao, Tong Zhao, Jin Li, Xin L. Dong, Christos Faloutsos, Yizhou Sun, and Wei Wang. P-Companion: A Principled Framework for Diversified Complementary Product Recommendation. In *Proceedings of the ACM International Conference on Information & Knowledge Management (CIKM)*, pages 2517–2524. ACM, 2020.

Marco Mussi, Simone Drago, Marcello Restelli, and Alberto M. Metelli. Factored-reward bandits with intermediate observations: Regret minimization and best arm identification. *Artificial Intelligence*, 347:104362, 2025.