Automated Repricing and Ordering Strategies in Competitive Markets

Rainer Schlosser [a,*], Carsten Walther [a], Martin Boissier [a], and Matthias Uflacker [a]
[a] *Hasso Plattner Institute, University or Potsdam, Potsdam, Germany*
E-mail: {first.last}@hpi.de

Abstract.
Merchants on modern e-commerce platforms face a highly competitive environments. They compete against each other using automated dynamic pricing and ordering strategies. Successfully managing both inventory levels as well as offer prices is a challenging task as (i) demand is uncertain, (ii) competitors strategically interact, and (iii) optimized pricing and ordering decisions are mutually dependent. We show how to derive optimized data-driven pricing and ordering strategies which are based on demand learning techniques and efficient dynamic optimization models. We verify the superior performance of our self-adaptive strategies by comparing them against different rule-based as well as data-driven strategies in duopoly and oligopoly settings. Further, to study and to optimize joint dynamic ordering and pricing strategies on online marketplaces, we built an interactive simulation platform. To be both flexible and scalable, the platform has a microservice-based architecture and allows handling dozens of competing merchants and streams of consumers with configurable characteristics.

Keywords: dynamic pricing, inventory management, demand learning, oligopoly competition, e-commerce

1. Introduction

Online markets have become highly dynamic and competitive. Merchants automatically adjust prices to react to changing market situations, cf. [1]. Similarly, they can flexibly reorder items taking into account (i) estimated demand, (ii) delivery times, (iii) ordering costs, and (iv) inventory holding costs.

Computing well-performing pricing and ordering strategies is challenging as demand is uncertain and markets are steadily changing (cf. [2], [3], [4]). What is more, pricing and ordering strategies mutually affect each other [5].

As testing is potentially hazardous when done in production, simulating the performance of automated ordering and pricing strategies is crucial. Nevertheless, there is a distinct lack of simulation platforms which allow evaluating data-driven strategies under various competitive setups. Existing platforms, e.g., [6], [7], are limited in their capabilities: Simulations run (i) on a single machine, (ii) offer a limited set of consumer behaviors, (iii) simulate solely short sales horizons, and (iv) price updates or orders are restricted to predefined discrete points in time.

Resembling production marketplaces such as Amazon or eBay [8, 9], we built a continuous time framework to simulate dynamic pricing and ordering under competition. The setup allows for customers with heterogeneous buying behaviors. Further, the competitors' offers include multiple dimensions such as price and product quality.

Our platform supports large numbers of merchants to compete simultaneously. Each merchant can run his preferred ordering and repricing strategy to order products and adjust prices on the marketplace, respectively. Market situations steadily change due to the strategic interaction of competing merchants' price reactions. Simulating random streams of interested customers allows generating realized sales events and the firms' sales revenues. The firms' inventory levels, their holding costs as well as their ordering costs depend on their ordering strategies and can be easily evaluated. By visualizing price evolutions, inventory levels, and profits over time, the user can easily study the complex interplay of ordering and repricing strategies and, most importantly, compare short and long-term profits.

The platform logs each interaction such as orders, price updates, stock-outs, new offers, sales, etc. This historic data – which is defined as partially observable as sales are typically private knowledge – is requested

[*]Corresponding author. E-mail: {first.last}@hpi.de.

and numerically analyzed by data-driven merchants. Our system supports self-adapting learning strategies. Various state-of-the-art machine learning approaches can be applied to quantify how demand (i.e., sales probabilities) is affected by a merchant's pricing decisions.

In addition, merchants are able to develop own optimization models [10–12] which are calibrated by estimated sales probabilities to compute *optimized* data-driven pricing and ordering strategies. In this context, it is even possible to learn competitors' strategies in order to take *anticipated* price reactions into account.

Our framework also allows controlling and measuring the influence of (i) the customers' buying behavior, (ii) price adjustment frequencies, as well as (iii) the exit or entry of competitors on a strategy's performance. In addition, different demand learning techniques and optimization approaches can be compared regarding their accuracy and efficiency.

In this paper, we make the following contributions:

- We present a platform to simulate competing pricing and ordering strategies.
- We show how to estimate demand from partially observable market data.
- We derive effective data-driven dynamic pricing and ordering strategies.
- We evaluate the complex interplay of various strategies in duopoly and oligopoly scenarios.
- We verify that our data-driven strategy outperforms different rule-based strategies.

This paper is organized as follows. In Section 2, we discuss related work. In Section 3, we describe the main components of our platform. In Section 4, we introduce our stochastic dynamic optimization model. We show how to estimate demand from historical market data and how to compute optimized pricing and ordering decisions. In Section 5, we present performance evaluations of our data-driven strategy derived. In Section 6, we provide implementation details for our merchants. Conclusions are summarized in the final Section 7.

2. Related Work

Inventory control problems and dynamic pricing problems have been extensively studied for decades, cf., e.g., [13], [14] for pure ordering or [15], [16] for pure pricing problems. Joint dynamic pricing and ordering problems are reviewed in the survey by [17]. Solutions are proposed for different problem scenarios, if demand is known, cf. [10], [18], or [19].

Scenarios with uncertain demand are less well studied. Future demand must be estimated from market observations. Typical approaches are to investigate specific classes of parameterized demand distributions and propose methods to find parameters, so that the demand distribution fits the experienced sales best, cf. [20]. [21] propose Bayesian based approaches for ordering and pricing problems with uncertain demand. Adida, Perakis (2006) study this problem in a multi-product scenario without backordering.

Further, in recent literature there are approaches to also incorporate competition. The surveys [4] and [22] provide an overview about the dynamic pricing problem under competition for single-product and multi-product scenarios. Finite time horizon settings have been studied, e.g., by [23]. Data-driven repricing strategies for infinite horizon oligopolies are derived in [12]. In [3], the authors consider joint pricing and inventory control in a duopoly.

The combined problem of joint ordering and pricing, demand learning, and oligopoly competition is highly challenging; usually heuristics have to be used. For analyzing and evaluating the complex interplay of data-driven and rule-based ordering and pricing strategies under competition, simulation platforms, cf. [24], can be used. For testing and evaluating merchant strategies, we use the platform Price Wars (cf. Section 3), a framework to simulate dynamic pricing competition on online marketplaces. The platform has advantages over other solutions (e.g., [25–27]) as (i) its continuous time model makes the platform similar to real online marketplaces like Amazon and (ii) users are unrestricted in their choice and implementation of merchant strategies.

3. Platform Description

The platform used for this project is called *Price Wars*, an open source platform for dynamic pricing research [9, 28]. The simulation platform is built with a microservice-based architecture for scalability and flexibility. Each service implements one business artifact, whereby services can be scaled out for large simulations. By having separated services, additional components can be added during *running* simulations at any time.

Merchants update their products' prices based on the current market situation which they can request at any time. Arbitrary strategies, e.g., rule-based or data-driven strategies, can be applied. Data of observed market situations as well as a merchant's sales data can be used to estimate sales probabilities (for current market situa-

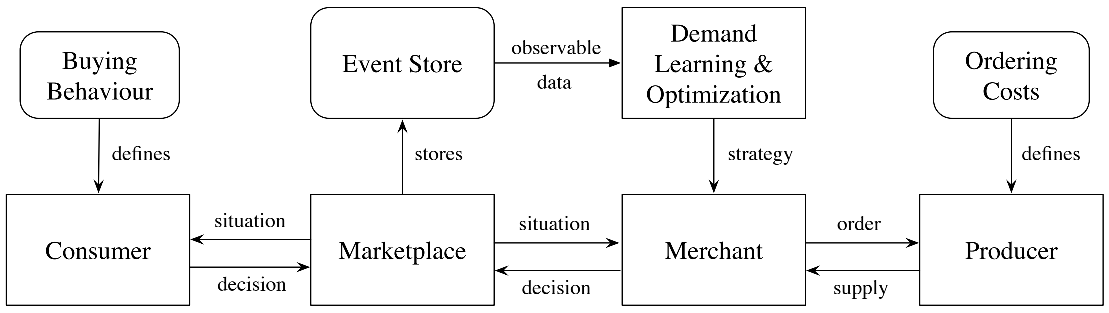

Fig. 1. Depiction of the platform's components and their interaction in a dynamic pricing scenario with inventory replenishment.

tions) using various machine learning techniques (e.g., logistic regression, boosted trees [29], reinforcement learning [30], or neural networks). Merchants can be easily added to the simulation or updated as long as they confirm to the HTTP/REST interface of the platform (cf. Section 6).

The centre of the simulation is the *marketplace* which manages all product offers, cf. Figure 1. The marketplace is the access point for the *consumer* component which creates a random stream of interested customers. Any customer choice behavior can be defined. The decision whether a customer buys a product and which offer is chosen, is probabilistically modelled and can depend on all parameters of the current market situation.

The *event store* logs platform events (price adjustments, sales, etc.) and provides CSV files for data-driven merchants. The *producer* provides products ordered by the merchants.

The *merchants* regularly request current market situations and decide on price updates and orders. For optimized well-matched pricing and ordering decisions, they can apply *demand learning* to estimate sales probabilities to be used in (dynamic) *optimization models*. We applied efficient dynamic programming techniques, which are described in the following sections.

The *HTML-based front end*, see Figure 9 in the Appendix, enables the user to configure (i) the customer behavior, (ii) the merchants' strategy setup, as well as (iii) all cost parameters (fixed/variable ordering costs, holding cost rates [31]). The front end also allows observing prices (see Figure 2), inventories (see Figure 3), and profits over time. The competing strategies' short-term and long-term performances are measured by different KPIs, including realized profits, revenues, holding costs, ordering costs, etc.

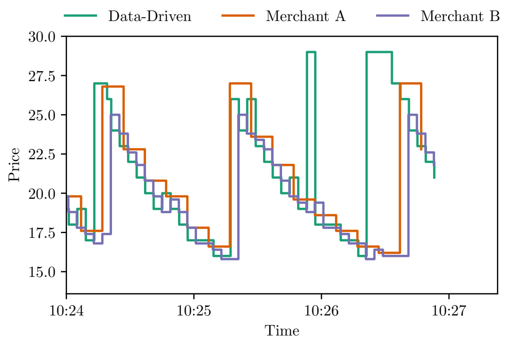

Fig. 2. Example of three competitors' prices over time.

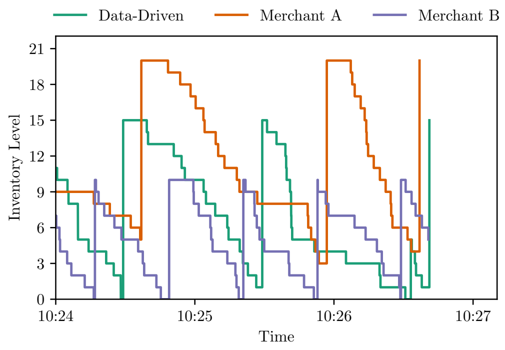

Fig. 3. Example of competitors' inventory levels over time.

4. Computation of Pricing and Ordering Strategies under Competition

In this section, we show how to derive optimized data-driven strategies. In Section 4.1, we introduce our stochastic dynamic pricing and ordering model. In Section 4.2, we show how a firm's observable historical market data can be used to estimate sales probabilities under competition. In Section 4.3, we propose an efficient mechanism to compute optimized joint pricing and ordering decisions.

4.1. Model Description

We consider the situation in which a firm seeks to sell a durable good over time. The time horizon is not restricted. We assume that (i) demand is uncertain and has to be estimated from historical data, (ii) prices can be adjusted over time, and (iii) items can be reproduced or reordered. Further, we assume several competitors for our products. In our model, we include substitution effects in demand as customers might compare prices of competitors. The goal is to derive data-driven pricing and ordering decisions to maximize expected discounted long-term profits.

If a sale takes place shipping costs c have to be paid, $c \geqslant 0$. Moreover, we consider inventory holding costs. We assume that each unsold item leads to holding costs of l per unit of time (e.g., one hour or one day), $l \geqslant 0$. We also include discounting in the model. For one unit of time, we use the discount factor δ, $0 < \delta < 1$. This corresponds to a discount rate α, $\alpha > 0$, given by $\alpha = \ln(\delta^{-1})$. A list of variables and parameters is given in the Appendix, cf. Table 6.

Due to customer choice, the demand for a firm's product particularly will depend on a firm's offer price a and the current competitors' prices $\vec{p} = (p^{(1)}, ..., p^{(K)})$, where K is the number of competitors at a certain point in time. W.l.o.g, we assume time homogeneous demand. To this end, the sales intensity of a firm (i.e., the average demand within one unit of time in case of stable prices) is denoted by λ, $a \geqslant 0$, $p^{(k)} \geqslant 0$, $k = 1, ..., K$,

$$\lambda(a, \vec{p}) \tag{1}$$

Our firm's random inventory level at time t is denoted by N_t, $t \geqslant 0$. If all items are sold, we let $\lambda(\cdot,\cdot) = 0$ (no back orders). If a firm does not offer items, we let its offer price $a := 0$. Items can be ordered at any time t, $t \geqslant 0$. The number of items ordered at time t are denoted by b_t. Ordered products are assumed to be delivered, e.g., with a delay of one unit of time, i.e., in time $t+1$ the inventory level N_{t+1} raises by b_t. The set of admissible order quantities is denoted by B. Ordering costs $C(b)$ are paid in advance and characterized by fixed and variable cost parameters $c_{fix}, c_{var} \geqslant 0$, $b \in B$, $C(0) = 0$,

$$C(b) := c_{fix} \cdot 1_{\{b>0\}} + c_{var} \cdot b \tag{2}$$

Prices can also be updated at continuous points in time t, $t \geqslant 0$. The set of admissible prices is denoted by A. However, prices cannot be adjusted infinitely often. As in real-life, we assume a certain limit for the number of updates processed within a certain time frame.

We call strategies (a_t, b_t) admissible if they belong to the class of Markovian feedback policies, i.e., pricing decisions $a_t \geqslant 0$ and ordering decisions $b_t \geqslant 0$ may depend the current inventory level N_t and the current competitor prices $\vec{p}_t$.

To account for ordering costs, by Z_t, we denote the (random) number of positive orders (cf. $1_{\{b_s>0\}}$) initiated up to time t, $t \geqslant s \geqslant 0$, $Z_0 := 0$. By X_t, we denote the random number of sales up to time t, $t \geqslant 0$, $X_0 := 0$. A firm's profits are characterized by its sales and orders which are connected to the inventory process N_t and the order process Z_t. Given a pricing and ordering strategy (a_t, b_t), a firm's random accumulated future profits (i.e., sales revenues minus holding costs minus order costs) from time t on (discounted on time t) amount to, $t \geqslant 0$,

$$G_t := \int_t^\infty e^{-\alpha\cdot(u-t)} \cdot (a_{u-} - c) dX_u$$

$$- \int_t^\infty e^{-\alpha\cdot(u-t)} \cdot l \cdot N_u du$$

$$- \int_t^\infty e^{-\alpha\cdot(u-t)} \cdot C(b_{u-}) dZ_u \tag{3}$$

The objective is to determine a non-anticipating feedback pricing and ordering policy that maximizes the expected discounted total profit $E\left(G_t \,|N_t, \vec{p}_t\right)$, cf. (1)-(3), conditioned on the current inventory level N_t and the current market situation $\vec{p}_t$ at time t.

4.2. Estimation of Sales Probabilities

The goal of this section is to estimate sales probabilities from historical market data. As in real-life applications, in our framework, merchants cannot continuously track markets over time. Typically, merchants have to request the marketplace to observe the current market situation. Then, based on the current market situation a price adjustment is sent back according to a certain repricing rule or strategy. Each merchant can also observe his/her realized sales as *private* knowledge.

Typically firms observe market situations shortly before they adjust their prices. Prices are kept constant until the next price update, i.e., a market request, takes place. A firm seeks to quantify how the numbers of observed sales within different time intervals are affected by the relation of a firm's offer price and the competitors' prices.

In the following, we assume that a firm has historical data for J time intervals. Observable data includes offer prices $a_{t^{(j)}}$, competitor prices $\vec{p}_{t^{(j)}}$ at times $t^{(j)}$, and realized sales $y_{t^{(j)}}$, i.e., the number of products sold within the time intervals $(t^{(j)}, t^{(j+1)})$, $j = 0, ..., J-1$, see Table 1.

j	$t^{(j)}$	$t^{(j+1)}$	$y_{t^{(j)}}$	$a_{t^{(j)}}$	$p^{(1)}_{t^{(j)}}$	$p^{(2)}_{t^{(j)}}$	...	$p^{(K)}_{t^{(j)}}$
0	0.0	1.4	4	15	11	13	...	/
1	1.4	2.5	12	10	11	13	...	17
2	2.5	2.8	8	12	9	13	...	16
3	2.8	4.0	5	15	9	11	...	/
...	...	...	...	...	...	...	...	...
$J-1$	$t^{(J-1)}$	$t^{(J)}$	...	...	...	...	...	...

Table 1

Illustration of observable and private data of a firm: Competitors' offer prices $\vec{p}$ at discrete points in time $t^{(j)}$ and number of sales between $t^{(j)}$ and $t^{(j+1)}$ at price $a_{t^{(j)}}$, $j = 0, 1, ..., J-1$.

A firm that plans to set a price at time t for the length of, e.g., h units of time, seeks to estimate sales probabilities for the time frame $(t, t+h)$ given the current market prices $\vec{p}$ observed at time t. The own offer price a can be chosen from a set A of admissible prices. In this context, a firm seeks to derive an *estimation* of the (true) conditional sales probabilities, $h > 0$, $i = 0, 1, ...,$ $t \geqslant 0$, $a, p^{(k)} \in A$, $k = 1, ..., K$,

$$P_{t,t+h}\,(i, a|\vec{p}) \tag{4}$$

There are several approaches to estimate sales probabilities, cf. (4), from data sets as described in Table 1, see e.g., [12]. Common approaches are, e.g., least squares, logistic regression, gradient boosted trees (e.g., XGBoost [29]), neural networks, etc.

In the following, we illustrate a simple way to estimate demand. To explain the dependent variable $y_{t^{(j)}}$, $j = 0, ..., J-1$, we can use, e.g., a robust least squares regression model (LS model). Using the LS model, we aim to specify average expected sales for a time span of length h conditioned on initial (not necessarily stable) market prices $\vec{p}$, a, $p^{(k)} \in A$, $k = 1, ..., K$, $h \geqslant 0$,

$$\tilde{\lambda}(h, a|\vec{p}; \vec{\beta}) := \vec{x}(h, a, \vec{p})'\vec{\beta} \tag{5}$$

where $\vec{\beta} = (\beta_1, ..., \beta_M)$ is the unknown parameter vector that is associated to the vector $\vec{x} = (x_1, ..., x_M)$ of M explanatory variables. The regressors $\vec{x}(h, a, \vec{p})$ can be a function of offer price a and market prices $\vec{p}$. The optimal coefficients $\vec{\beta}^* = (\beta_1^*, ..., \beta_M^*)$ can be easily obtained using standard methods.

The resulting intensities $\tilde{\lambda}^*(h, a|\vec{p}) := \tilde{\lambda}(h, a|\vec{p}; \vec{\beta}^*)$, cf. (5), can be used to estimate P, cf. (4): For h units of time, we let the estimated sales probabilities $\tilde{P}_{t,t+h}(\cdot, a|\vec{p})$ be Poisson distributed with rate $\tilde{\lambda}^*(h, a|\vec{p})$, $a \in A$, $h \geqslant 0$, $t \geqslant 0$, $i = 0, 1, ...,$ i.e.,

$$\tilde{P}_{t,t+h}(i, a_t|\vec{p}_t) := \frac{\tilde{\lambda}^*(h, a_t|\vec{p}_t)^i}{i!} \cdot e^{-\tilde{\lambda}^*(h, a_t|\vec{p}_t)} \tag{6}$$

To illustrate the approach, in the following definition, we give simple examples of explanatory variables $\vec{x}$.

Definition 4.1. We define the following regressors x_m, $m = 1, ..., 7$, for given data $a_{t^{(j)}}, p^{(k)}_{t^{(j)}} \in A$, $k = 1, ..., K_{t^{(j)}}$, $h^{(j)} := t^{(j+1)} - t^{(j)}$, $t^{(j)} \geqslant 0$, $j = 0, ..., J-1$:

(i) constant / intercept

$$x_1(h^{(j)}, a_{t^{(j)}}, \vec{p}_{t^{(j)}}) = 1$$

(ii) own price at time $t^{(j)}$

$$x_2(h^{(j)}, a_{t^{(j)}}, \vec{p}_{t^{(j)}}) := a_{t^{(j)}}$$

(iii) rank of own price a within prices $\vec{p}$ at time $t^{(j)}$

$$x_3(h^{(j)}, a_{t^{(j)}}, \vec{p}_{t^{(j)}}) := rank(a_{t^{(j)}}, \vec{p}_{t^{(j)}})$$

(iv) price gap between $a_t^{(j)}$ and best competitor

$$x_4(h^{(j)}, a_{t^{(j)}}, \vec{p}_{t^{(j)}}) := a_{t^{(j)}} - \min_{k=1,...,K_{t^{(j)}}} \{p^{(k)}_{t^{(j)}}\}$$

(v) number of competitors at time $t^{(j)}$

$$x_5(h^{(j)}, a_{t^{(j)}}, \vec{p}_{t^{(j)}}) := K_{t^{(j)}}$$

(vi) availability of our product at time $t^{(j)}$

$$x_6(h^{(j)}, a_{t^{(j)}}, \vec{p}_{t^{(j)}}) := 1_{\{N_{t^{(j)}}>0\}} = 1_{\{a_{t^{(j)}}=0\}}$$

(vii) interval length $h^{(j)}$

$$x_7(h^{(j)}, a_{t^{(j)}}, \vec{p}_{t^{(j)}}) := h^{(j)}$$

Our framework allows to measure the impact of a firm's offer price in the presence of competitors' market prices. Note, also non-linear versions of explanatory variables can be used. In this general framework, further explanatory variables can be easily defined to capture the impact of additional effects, such as time, product quality, etc.

Note, the quality of estimations of sales probabilities (for the entire range of prices or for prices that often occur during the competition) can be analyzed and compared for different demand learning techniques. As the true sales probabilities are characterized by the defined customer behavior and the competing merchant's strategies they can be determined using, e.g., Monte Carlo simulations, cf. [12].

In general, regression results are better if prices are more randomized, cf. [32]. In this context, our platform can also be used to study the impact of a selection bias caused by a firm's strategy as well as the competitors' strategies. Moreover, the impact of various effects of the model can be studied, such as distribution and length of reaction times, customer arrival intensity, customers' buying behavior, or number of competitors, etc.

Finally, the (estimated) conditional probabilities (4) and (6) are affected by both, the customer behavior as well as the strategic interplay of competitors' price adjustments. Recall, a firm's demand learning does neither anticipate competitors' strategies nor their reaction times. The estimated probabilities (6), however, allow to *indirectly* measure the average impact of competitors' price adjustments and, thus, account for the fact that market situations may change between two price adjustments of a firm.

Our platform allows to test and to validate different demand learning approaches in different competitive markets. In addition, components of the demand estimation can be further improved (exploration phases, sampling, feature selection, etc.). While not focus of this paper, further issues, such as missing variables, IIA assumption, unobservable demand shocks, etc., can be addressed, cf. to recent literature, e.g., [33] or [34]. To this end, our model can be used to study to which extent such effects influence the quality of different demand learning approaches.

The goal of the next section is to derive effective ordering and pricing decisions that are based on estimated conditional probabilities for current market situations, cf. (6). Further, we seek to account for holding costs, ordering costs, and discounting.

4.3. Dynamic Model and Solution Approach

There are two major problems to derive applicable pricing strategies in competitive markets: (i) as demand is affected by many parameters (e.g., dozens of competitors' prices) a model's state space explodes and the problem becomes intractable, and (ii) in general, as competitors' strategies are not known, their price adjustments cannot be effectively anticipated.

Our approach deals with both problems. Most importantly, instead of computing complete feedback strategies, we compute prices for one period only based on the current market situation that occurs during a sales process. To compute controls for single time periods, in general, the current state as well as potential future states have to be taken into account. As price reactions of competitors occur with a certain delay the short-term evolution of the market can be well approximated by the current market situation. The long-term evolution of the market, however, can hardly be predicted. Our approach is motivated by the fact that the optimal price for one period mostly depends on the current state and is much less affected by specific potential states in the future, see [35] for finite horizon pure pricing problems.

For a current state, we manage problem (i) as follows: We roughly approximate future market situations by using sticky prices. While the degree of inaccuracy is acceptable, we gain a structure that makes it possible to circumvent the curse of dimensionality, cf. problem (i), as the states of our dynamic system (i.e., the market situation) are not coupled and can be decomposed. Thus, for single states decisions can be computed independently, which makes it possible to consider current market situations only.

The second key idea is to compensate the model's inaccuracy as well as the lack of price anticipations, cf. problem (i), by frequent price adjustments, which in turn are possible as the model's simplicity allows for fast re-computations.

Due to price adjustments, exits, or entries of firms, in general, market situations are not stable. In our model, we consider (estimated) conditional sales probabilities, cf. (6), $a \geqslant 0$, $h > 0$, $t \geqslant 0$, $i = 0, 1, ...$,

$$\tilde{P}^{(h)}(i, a|\vec{p}) := \tilde{P}_{t,t+h}\,(i, a|\vec{p}) \qquad (7)$$

for selling i items within the time span $(t, t+h)$ at price a under the condition that at time t the market situation is $\vec{p}$ (and may change within the period due to competitors' price reactions). Note, following our assumptions, in (7) demand is time homogeneous, i.e., considering the interval $(t, t+h)$ only the period length h matters. However, time-dependent sales probabilities are also possible (e.g., seasonal and cyclic effects).

W.l.o.g., in the following, we consider a firm with an average price adjustment delay of $h = 1$. As described in the beginning of this section, we use a simplified dynamic programming approach based on a discrete time model. In this context, a firm's random accumulated future profits G_t, cf. (3), from time t on (discounted on time t) amount to, $t = 0, 1, 2, ...,$

$$G_t := \sum_{s=t}^{\infty} \delta^{s-t} \cdot \begin{pmatrix} (a_s(N_s, \vec{p}_s) - c) \cdot (X_{s+1} - X_s) \\ -l \cdot N_s - C\left(b_s(N_s, \vec{p}_s)\right) \end{pmatrix} \tag{8}$$

In a given state $(n, \vec{p})$ at time t, the best expected discounted future profits $E(G_t | N_t = n, \vec{p}_t = \vec{p})$, cf. (8), are independent of time and described by the value function $V^*(n, \vec{p})$, $n = 0, 1, ..., N$, $p^{(k)} \geqslant 0$, $k = 1, ..., K$.

If a period's (random) demand is i items and b items are ordered (with delivery delay), the transition of the current inventory level n to the next period's level is given by $n \to \max(n-i, 0) + b$. To avoid an unbounded state space, we use the upper limit N_{max}, which – if chosen sufficiently large – does not affect the optimal solution $(a^*(n, \vec{p}), b^*(n, \vec{p}))$, which is characterized by the associated Hamilton-Jacobi-Bellman equation, $n = 0, ..., N_{max}$, $p^{(k)} \geqslant 0$, $k = 1, ..., K$,

$$V^*(n, \vec{p}) = \max_{a \in A, b \in B} \left\{ \sum_{i=0,...,N_{\max}} \tilde{P}^{(1)}(i, a | \vec{p}) \right.$$

$$\left. \cdot \begin{pmatrix} (a - c) \cdot \min(i, n) - l \cdot n - C(b) \\ +z \cdot \delta \cdot V^*\left(\min\left((n - i)^+ + b, N_{\max}\right), \vec{p}\right) \end{pmatrix} \right\} \tag{9}$$

where z, $z \geqslant 0$, is an additional penalty/discount parameter which allows (i) to control the *speed of sales* of the feedback policy, and (ii) to account for expected general *long-term* market trends (decay of average prices, product attractiveness, etc.). For the time being, we let $z := 1$. The set of admissible prices A and order quantities B can be chosen arbitrarily.

The optimal joint pricing $a^*(n, \vec{p})$ and ordering strategy $b^*(n, \vec{p})$, $n = 0, ..., N_{max}$, $p^{(k)} \geqslant 0$, $k = 1, ..., K$, is given by the *arg max* of (9). If optimal prices or ordering quantities are not uniquely determined, we choose the largest numbers.

The solution of the system of equations (9) can be derived using standard methods like value iteration or policy iteration. Alternatively, the system can also be solved using a (nonlinear) solver. Note, the number of variables and constraints is $N_{max} + 1$.

Value iteration does not need a solver to approximate the value function. For a given "large" number T, we use the terminal condition

$$V_T(n, \vec{p}) := 0 \tag{10}$$

for all numbers n and market situations $\vec{p}$. Using the recursion, $t = 0, 1, ..., T-1$, $n = 0, ..., N_{max}$, $p^{(k)} \geqslant 0$, $k = 1, ..., K$,

$$V_t(n, \vec{p}) = \max_{a \in A, b \in B} \left\{ \sum_{i=0,...,N_{\max}} \tilde{P}^{(1)}(i, a | \vec{p}) \right.$$

$$\left. \cdot \begin{pmatrix} (a - c) \cdot \min(i, n) - l \cdot n - C(b) \\ +z \cdot \delta \cdot V_{t+1}\left(\min\left((n - i)^+ + b, N_{\max}\right), \vec{p}\right) \end{pmatrix} \right\} \tag{11}$$

we can compute the values $V_t(n, \vec{p})$, $t = 0, 1, ..., T-1$. The number of iteration steps T can be chosen such that the approximation error between V and V^* is sufficiently small. The approximation error can be estimated via the discount factor δ.

Finally, the associated (optimal) strategies $a_0(n, \vec{p})$ and $b_0(n, \vec{p})$, $n = 0, ..., N$, are given by the *arg max* of (11) at the last recursion step, cf. $t = 0$.

Note, due to the size of the state space it is *not* possible to compute prices $a_t(n, \vec{p})$ for all states $\vec{p}$ in advance. The following algorithm, however, circumvents the curse of dimensionality and allows to derive viable heuristic joint pricing and ordering strategies in competitive markets with a large number of competitors.

Algorithm 4.1. We define the following pricing and ordering heuristic:

Step 1: For every period t observe the new state, i.e., the current inventory level N_t and the current market situation $\vec{p}_t$. Compute the probabilities $\tilde{P}^{(1)}(i, a | \vec{p}_t)$ for all $i = 0, 1, ..., N_t$, $a \in A$.

Step 2: Solve either (9) for $V^*(N_t, \vec{p}_t)$ and $a^*(N_t, \vec{p}_t)$, $b^*(N_t, \vec{p}_t)$, or use T recursion steps to compute the specific value $V_0(N_t, \vec{p}_t)$, cf. (10) - (11), and obtain the associated offer price $a_0(N_t, \vec{p}_t)$ and the ordering decision $b_0(N_t, \vec{p}_t)$.

The key idea is to just compute decisions for single market situations and to regularly refresh them in response to changing market situations. Due to the small dimensionality of the state space, a single recomputation is very fast. Further, our solution is *scalable* as the algorithm's complexity does neither increase with the number of competitors, the number of offer dimensions, nor the number of explanatory variables.

Remark 4.1. The recomputations of Algorithm 4.1 can be speed up as follows:

(i) Typically it is sufficient to consider a small subset of admissible prices A and order quantities B. Suitable subsets can be derived from previous computations for $a^*(n,\vec{p})$ and $b^*(n,\vec{p})$.

(ii) The computation of decisions via (11) can be dramatically accelerated by using suitable starting values $v(n)$, $n=0,1,...,N_{max}$, for the terminal condition $V_T(n,\vec{p}):=v(n)$, cf. (10). Suitable starting values can be derived from previous computations, i.e., $v(n):=V_0(n,\vec{p}')$ for market situations $\vec{p}'$ similar to $\vec{p}$.

(iii) Further, if the computation time shall be below a certain time limit (e.g., 0.1 seconds) the number of recursion steps, cf. (11), can either be chosen sufficiently small or the recursive approximation is stopped accordingly.

(iv) Reaction times are an competitive advantage. However, the number of market requests is often limited. Our framework allows to balance the accuracy of solutions and the required computation time. The number of price updates processed can be effectively controlled.

5. Numerical Examples and Evaluation

The platform allows simulating strategic interaction of rule-based and data-driven strategies in different market scenarios characterized by product portfolios, customer behaviors, oligopoly settings, and cost definitions. In Section 5.1, we describe our setup and give examples of rule-based merchants. In Section 5.2, we study how our data-driven strategy performs in duopoly setups. In Section 5.3, we evaluate an oligopoly scenario.

5.1. Merchant Description and Simulation Setup

Our merchant implementation with the proposed optimization model is called *data-driven merchant*. A new training on all training data is processed every minute. The period length is four seconds. Further, we let $N_{max}=40$, $T=40$, $A:=\{0.1,0.2,...,100\}$, $B:=\{0,1,2,...,N_{max}\}$, and $\delta=0.9999$.

Besides our data-driven merchant, we consider the following two rule-based merchants. The *cheapest merchant* always undercuts the cheapest competitor by configurable amount (here, 0.30). Only if the cheapest competitor price is higher than the upper price bound of 30, the cheapest merchant sets a price of 30 instead. If no competitor offer is available, the cheapest merchant sets the price to the upper price bound. The merchant makes a new order when the inventory level falls below six items. In that case, the merchant orders as many items as needed to refill the inventory to 20 items. Price updates are made every four seconds.

The second rule-based merchant, called *two bound merchant*, undercuts the competitor with the lowest price by 0.30, similar to the cheapest merchant. However, the merchant has a upper and lower price bound. If the cheapest competitor's offer price is below the lower price bound of 17, the two bound merchant sets the price to the upper price bound, 30. Moreover, if no competitor offers are available or all competitor prices are above the upper price bound, the price is also set to the upper price bound. This merchant makes a new order if the inventory level falls below four items. In that case, the merchant orders as many items as needed to refill the inventory to 15 items. Price updates are also made every four seconds.

Consumers are configured to visit the marketplace at an average rate of 100 consumers per minute. The time between arriving consumers is exponentially distributed with a mean of 0.6 seconds. They dismiss offers costing 80 or more. Assume the remaining J offers $\vec{o}=(o_1,o_2,\dots,o_J)$ have prices $\vec{p}=(p_1,p_2,\dots,p_J)$. The maximal price in $\vec{p}$ is denoted by p_{max}, the sum of these prices is denoted by p_{sum}. An arriving consumer buys one item from the remaining offers j at random with the probability distribution, $j=1,\dots,J$,

$$P(\text{Buy from } o_j)=\frac{p_{max}+1-p_j}{J\cdot(p_{max}+1)-p_{sum}} \qquad (12)$$

If there are no offers with prices below 80 (willingness to pay), the consumer leaves the marketplace without buying anything.

Simulations have a duration of 15 minutes. Ordering costs are defined by $c_{fix}=10$ and $c_{var}=15$. Holding costs are three per minute per item for all merchants. That corresponds to $l=3/(60s/4s)=0.2$ for the data-driven merchant. If not mentioned otherwise, the following simulations are run with the configuration listed here.

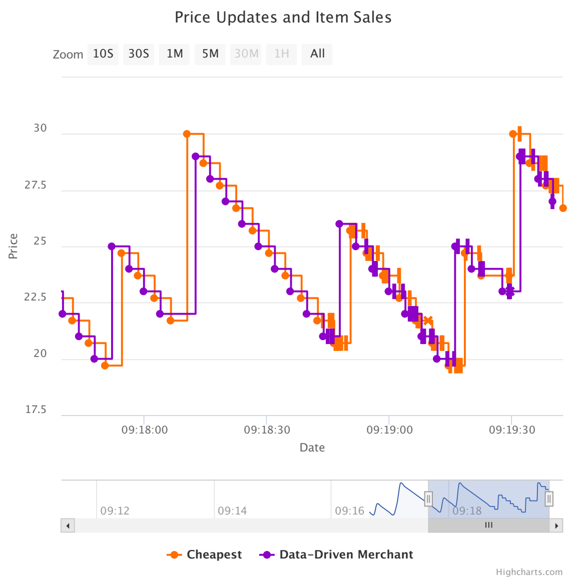

Fig. 4. Screenshot of the Price Wars simulation: Price trajectories in a duopoly of our data-driven merchant and the cheapest merchant. Dots in the chart are price updates and bars are sales events. Configurations are listed in Section 5.1 and Section 5.2.1.

5.2. *Duopoly Simulation*

In this section, we investigate the profitability of the proposed merchant in different duopoly scenarios. Our data-driven merchant competes with each of the two rule-based merchants. In the last duopoly scenario, two data-driven merchants with the same strategy compete with each other. In the simulations, the competitors' strategies are mutually not observable.

5.2.1. *Data-Driven Merchant vs. Cheapest Merchant*

We simulated a duopoly between the data-driven and cheapest merchant for 15 minutes. Figure 4 shows how both merchants' prices undercut each other. The data-driven merchant does not reduce the price below 20; the price is raised to 25-30. The increased price has lower sales probabilities but a higher profit margin. Surprisingly, we observe that the cheapest merchant increases the price sometimes. When the data-driven merchant is out of stock, there is no competitor offer for the cheapest merchant to undercut. When this is the case, the cheapest merchant uses a default price. The final performance results of this simulation are shown in Table 2. The data-driven merchant has overall more ordering and holding costs. However, profits are higher as costs are overcompensated by higher revenues compared to the cheapest merchant.

Merchant	Profit	Revenue	Holding	Ordering
Data-Driven	7 285.78	20 599.00	588.22	12 725.00
Cheapest	5 796.11	17 165.10	418.99	10 950.00

Table 2

Performance results of a duopoly with our data-driven merchant and the cheapest merchant. Configuration as described in Section 5.1 and Section 5.2.1.

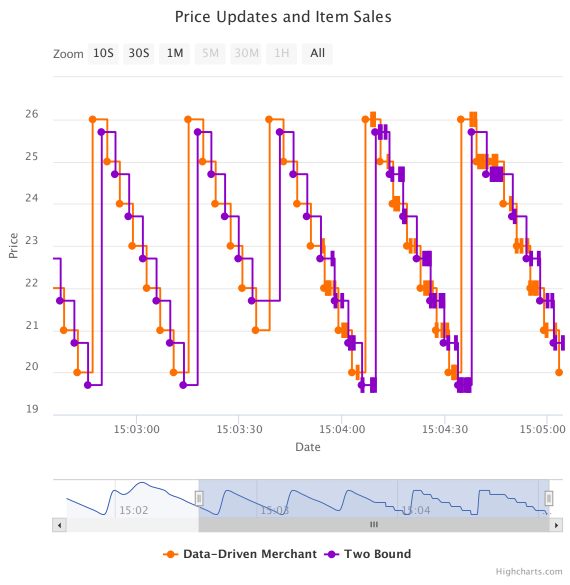

Fig. 5. Screenshot of the Price Wars simulation: Price trajectories in a duopoly of our data-driven merchant and the two bound merchant. The two bound merchant restocks the inventory to 25 and reorders if the inventory falls below 7 instead of four items to reduce the number of stock-outs. All other parameters are as defined in Section 5.1 and Section 5.2.2.

5.2.2. *Data-Driven Merchant vs. Two Bound Merchant*

In a second setup, we study the competition between the data-driven merchant and the two bound merchant. Compared to the cheapest merchant, we used a different ordering policy. The two bound merchant restocks to 25 items (instead of 15) and makes a new order whenever the inventory level falls below 7 items (instead of 4). This reduces the risk of having stock-outs.

The data-driven merchant expects the most profit from undercutting the competitor. This results in both merchants undercutting each other as shown in Figure 5. The two bound merchant is programmed to raise the price if it falls below a certain threshold (price 17). However, the data-driven merchant was the first to increase the price in this simulation (if prices are below 20). The performance results are shown in Table 3. Both merchants made less profit compared to the previous setting, see Section 5.2.1.

Merchant	Profit	Revenue	Holding	Ordering
Data-Driven	5 858.79	18 984.00	595.20	12 530.0
Two Bound	5 230.10	16 952.70	527.60	11 195.0

Table 3

Simulation results of a duopoly with our data-driven merchant and the two bound merchant. The two bound merchant holds more items in the inventory to reduce the number of stock-outs. The two bound merchant restocks the inventory to 25 instead of 15 items and reorders if the inventory falls below 7 instead of 4 items to reduce the number of stock-outs. Other parameters as described in Section 5.1 and Section 5.2.2.

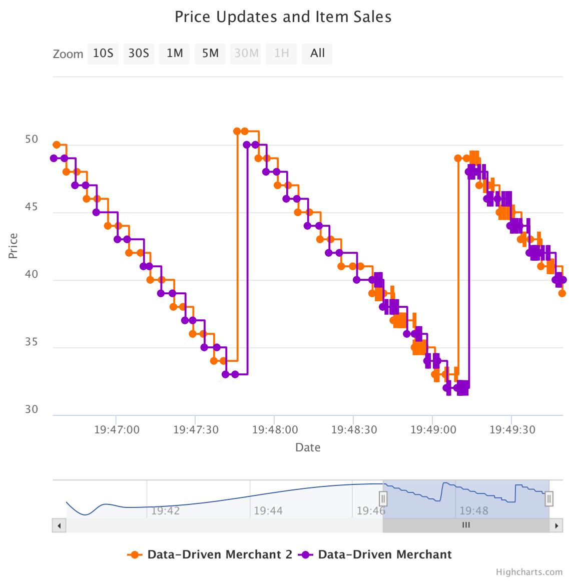

Fig. 6. Screenshot of the Price Wars simulation: Price trajectories in a duopoly of two identically configured data-driven merchants. Parameters as described in Section 5.1 and Section 5.2.3.

5.2.3. *Data-Driven Merchant Against Itself*

The third setup analyzes the competition between two identical instances of our data-driven merchant. The price chart in Figure 6 shows again the typical zig-zag pattern of two merchants undercutting each other and periodically pushing the price up to restore the price level and, in turn, to increase profit margins. The competition between two data-driven merchants happens at a higher price level (around 32-51) compared to the previous simulation (around 20-26). This results in an overall higher profit for both competing merchants. The automated data-driven strategies suggest to order new items whenever the inventory level falls below six items and restock the inventory to around 28 items.

Results of this simulation are shown in Table 4. We observe that performance results are overall similar but not entirely symmetric. Merchants do not gather the

Merchant	Profit	Revenue	Holding	Ordering
Data-Driven	17 361.30	30 936.0	804.69	12 770.0
Data-Driven 2	15 650.58	27 282.0	721.41	10 910.0

Table 4

Simulation results of a duopoly with two (symmetric) data-driven merchants. Configurations as described in Section 5.1 and Section 5.2.3; simulation duration of 30 minutes.

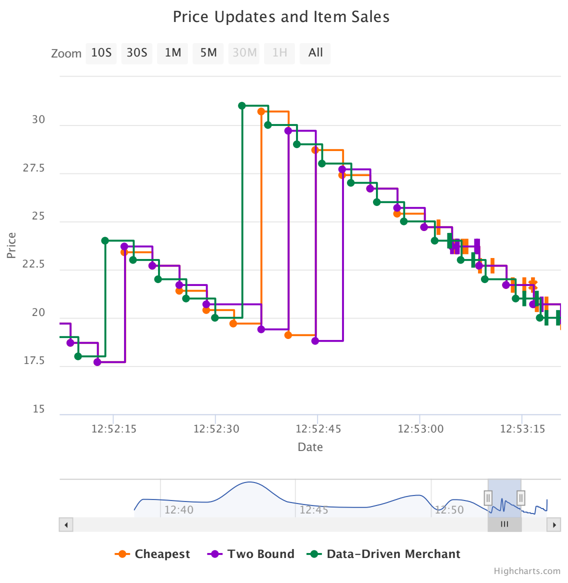

Fig. 7. Screenshot of the Price Wars simulation: Price trajectories in an oligopoly scenario on the platform. Our data-driven merchant competes with two rule-based merchants. Configurations as described in Section 5.1 and Section 5.3; simulation duration of 30 minutes.

same sales events which may result in different pricing and ordering policies. Further, as price reactions of both merchants have the same frequency and occur almost equidistantly, the mutual price reaction times can be uneven. Hence, the percentage of time a merchant has the most recent price update can be skewed, which leads to different results. To circumvent this issue, reaction times can randomized, cf. [36]. This makes it also harder to anticipate price reaction times in order to choose the timing of price updates strategically.

5.3. *Oligopoly Simulation*

This section illustrates how our data-driven merchant performs in an oligopoly. We simulated the competition between the data-driven merchant, the cheapest merchant, and the two bound merchant. The evolution of pricing and ordering decisions during the simulation are depicted in Figures 7 and 8. Again, the data-driven

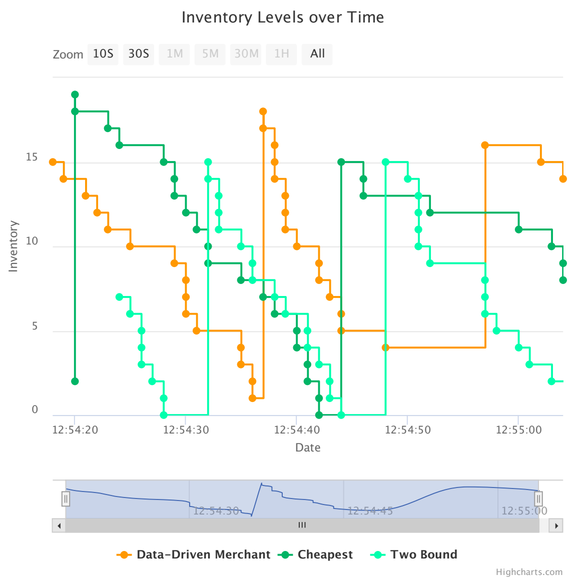

Fig. 8. Screenshot of the Price Wars simulation: Inventory levels over time in an oligopoly scenario on the platform. Our data-driven merchant competes with two rule-based merchants. Configurations as described in Section 5.1 and Section 5.3; simulation duration of 30 minutes. Note, while the simulation of sales is in continuous time, the plot grid is discrete.

merchant learns the advantage of undercutting competitors' offers. However, this creates a high price competition and average price levels decrease. With shrinking profit margins, it becomes unprofitable to set the price below competitors' prices. The data-driven merchant pushes the price up in such a situation. This motivates the competitors to also increase their offer prices.

Table 5 shows the results of a 30 minutes competition between the three merchants. The data-driven merchant outperformed all competitors. Our data-driven merchant made around 10% more profit than the cheapest merchant and 18% more than the two bound merchant. Interestingly, our merchant did not make the most revenue (the cheapest merchant did). The cheapest merchant sold the most items but with low profit per item. Our merchant made the most profit by saving a lot of order cost compared the to cheapest merchant. The data-driven merchant orders on average more items than the competitors. This results in higher holding costs but saves on fixed order cost.

We find that fully automated data-driven strategies – combined with efficient dynamic programming optimization techniques – clearly outperform rule-based strategies after a sufficiently large data set has been gathered for demand learning. It can also be studied to which extent jointly optimized pricing and ordering

Merchant	Profit	Revenue	Holding	Ordering
Data-Driven	5 944.13	21 938.00	943.87	15 050.00
Cheapest	5 386.90	23 770.80	903.89	17 480.00
Two Bound	5 038.63	20 148.30	644.67	14 465.00

Table 5

Simulation results of an oligopoly scenario with the data-driven, the cheapest, and a two bound merchant. The data-driven merchant made the most profit. The cheapest merchant made the most revenue. Configurations as described in Section 5.1 and Section 5.3; simulation duration of 30 minutes.

strategies outperform different combinations of single ordering and pricing benchmark strategies.

Moreover, the platform can be used to study short-term as well as long-term performance of self-adapting strategies that iteratively improve over time.

6. Merchant Implementation Details

Merchants on the platform can be written in any language as long as they comply with the platform's REST APIs. We decided to implement our merchant in the Python programming language [37] for the following reasons. Python has great library support for numerical computing. These libraries allow a concise and efficient implementation without reinventing the wheel. Our platform offers a Python implementation of the RESTful API for the merchant to communicate with the platform's services. Lastly, it is possible to quickly create prototypes in Python.

Our merchant consists of four components. The *main loop* is the central component. It regularly checks the marketplace for open offers, updates prices, and orders items from the producer. After enough time has passed, the merchant requests new market and sales data from the event log and provides it to the *demand learning* component to analyze demand.

The merchant makes ordering and pricing decisions based on policies that are computed by the *policy component*. The policy component contains the dynamic programming approach. The merchant provides all arguments that are necessary for the policy creation and sales probabilities are requested from the demand learning component. The dynamic programming function is the computational most expensive part of the merchant. An efficient implementation reduces the time needed for a pricing and ordering decision. We create a vector that has the dimensions inventory levels, ordering decisions, pricing decisions, and demand. The expected profit is calculated for each possible situation and decision that occur in this vector. The expected profits are used to find the most profitable decisions and to create the ordering and pricing policy. We use fast and vectorized array operations from the Numpy library [38] to compute the policies. Python is a high-level programming language and has a lot of computational overhead [39]. Numpy provides data structures and functions implemented in the C programming language to overcome Python's overhead for numeric computations.

The merchant's demand learning component is responsible for estimating sales probabilities and for bringing market and sales data into a form that can be used for training. The module uses linear regression to learn and predict the demand. We use the scikit-learn library [40] for a reliable and fast linear regression implementation. As an additional benefit, it is easy to change between regression algorithms using scikit-learn. The demand learning is implemented in a way that make it easy to add new or change existing explanatory variables. Only single function (named `extract_features`) must be changed to add new explanatory variables.

The *merchant server* receives sales events from the marketplace and triggers the appropriate action. In our case, the merchants prints a message to notify the user whenever an item was sold. Moreover, the server receives configuration updates from the platform frontend and applies them.

7. Conclusion and Future Work

In this work, we presented a distributed and scalable platform resembling real-world e-commerce applications. Both practitioners and researchers can investigate the strategic interaction of various adaptive data-driven pricing and ordering strategies and develop, test, and evaluate their own approaches.

Further, we have proposed a data-driven approach to derive effective pricing and ordering strategies. We combine private sales data with partially observable data of the competitors' offers to efficiently predict sales probabilities in competitive markets. Using estimated sales probabilities, we have set up a dynamic model including discounting, ordering costs, and holding costs. Our strategies are even applicable if the number of competitors' products is large. Our solution approach is characterized by a simplified frequently updated dynamic programming model, in which only current market situations have to be considered.

Our framework can be easily extended in several ways: (i) further offer dimensions (quality, ratings, shipping time, etc.), (ii) the consideration of perishable products, and (iii) substitution effects between different products. While the simulation platform allows to take these extensions into account, the presented optimization model has to be adapted. To address markets with multiple offer dimensions (cf. [41]) the demand learning component needs to be extended. In our setting, additional characteristic explanatory variables can be easily defined. The consideration of perishable products requires finite horizon models. Such models can be solved using recursive dynamic programming techniques. In multi-product models, the state space as well as the action space can be enormous. To manage this complexity, relaxation approaches and decomposition techniques can be used. The demand learning component has to be extended such that substitution effects are taken into account, e.g., [42].

References

[1] A.R. Greenwald and J. Kephart, Shopbots and pricebots, in: *International Joint Conference on Artifical Intelligence*, Vol. 1, 1999, pp. 506–511.

[2] W.-H. Tsai and S.-J. Hung, Dynamic pricing and revenue management process in internet retailing under uncertainty: An integrated real options approach, *Omega* **37**(2–37) (2009), 471–481.

[3] E. Adida and G. Perakis, Dynamic pricing and inventory control: Uncertainty and competition, *Operations Research* **58**(2) (2010), 289–302.

[4] M. Chen and Z.-L. Chen, Recent Developments in dynamic pricing research: Multiple products, competition, and limited demand information, *Production and Operations Management* **24**(5) (2015), 704–731.

[5] A. Rajan, Rakesh and R. Steinberg, Dynamic pricing and ordering decisions by a monopolist, *Management Science* **38**(2) (1992), 240–262.

[6] J.O. Kephart, J.E. Hanson and A.R. Greenwald, Dynamic pricing by software agents, *Computer Networks* **32**(6) (2000), 731–752.

[7] J.M. DiMicco, P. Maes and A. Greenwald, Learning curve: A Simulation-based approach to dynamic pricing, *Electronic Commerce Research* **3**(3–4) (2003), 245–276.

[8] M. Boissier, R. Schlosser, N. Podlesny, S. Serth, M. Bornstein, J. Latt, J. Lindemann, J. Selke and M. Uflacker, Data-Driven Repricing Strategies in Competitive Markets: An Interactive Simulation Platform, in: *Proceedings of the Conference on Recommender Systems, RecSys*, 2017, pp. 355–357.

[9] PriceWars, Price Wars Repository on GitHub: https://git.io/pricewars, *Last accessed: Sep 30th 2018* (2018).

[10] S. Transchel and S. Minner, The impact of dynamic pricing on the economic order decision, *European Journal of Operational Research* **198**(3) (2009), 773–789.

[11] A. Yabe, S. Ito and R. Fujimaki, Robust quadratic programming for price optimization, in: *International Joint Conference on Artificial Intelligence*, 2017.

[12] R. Schlosser and M. Boissier, Dynamic pricing under competition on online marketplaces: A data-driven approach, in: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2018)*, 2018, pp. 705–714.

[13] K.J. Arrow, *Studies in the mathematical theory of inventory and production*, Vol. 1, Stanford University Press, 1958.

[14] H.E. Scarf, A survey of analytic techniques in inventory theory, *Multistage Inventory Models and Techniques* **7** (1963), 185–225.

[15] K.T. Talluri and G.J. Van Ryzin, *The theory and practice of revenue management*, Vol. 68, Springer Science & Business Media, 2004.

[16] R.L. Phillips, *Pricing and revenue optimization*, Stanford University Press, 2005.

[17] W. Elmaghraby and P. Keskinocak, Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions, *Management Science* **49**(10) (2003), 1287–1309.

[18] A. Federgruen and A. Heching, Combined Pricing and Inventory Control Under Uncertainty, *Operations Research* **47**(3) (1999), 454–475.

[19] D. Simchi-Levi, X. Chen and J. Bramel, Integration of inventory and pricing, in: *The Logic of Logistics*, Springer, 2014, pp. 177–209.

[20] K.S. Azoury, Bayes solution to dynamic inventory models under unknown demand distribution, *Management Science* **31**(9) (1985), 1150–1160.

[21] A. Bisi and M. Dada, Dynamic learning, pricing, and ordering by a censored newsvendor, *Naval Research Logistics* **54**(4) (2007), 448–461.

[22] A.V. den Boer, Dynamic pricing and learning: historical origins, current research, and new directions, *Surveys in Operations Research and Management Science* **20**(1) (2015), 1–18.

[23] V. Martínez-de-Albéniz and K.T. Talluri, Dynamic Price Competition with Fixed Capacities, *Management Science* **57**(6) (2011), 1078–1093.

[24] R. Schlosser, C. Walther, M. Boissier and M. Uflacker, Data-driven inventory management and dynamic pricing competition on online marketplaces, in: *27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, 2018, pp. 5856–5858.

[25] J. Morris, A simulation-based approach to dynamic pricing, Master's thesis, Massachusetts Institute of Technology, 2001.

[26] J.M. DiMicco, P. Maes and A. Greenwald, Learning Curve: A Simulation-Based Approach to Dynamic Pricing, *Electronic Commerce Research* **3**(3–4) (2003), 245–276.

[27] T. Pinto, Z.A. Vale, T.M. Sousa, I. Praça, G. Santos and H. Morais, Adaptive learning in agents behaviour: A framework for electricity markets simulation, *Integrated Computer-Aided Engineering* **21**(4) (2014), 399–415.

[28] S. Serth, N. Podlesny, M. Bornstein, J. Latt, J. Lindemann, J. Selke, M. Boissier, M. Uflacker and R. Schlosser, An interactive platform to simulate dynamic pricing competition on online marketplaces, in: *21st IEEE International Enterprise Distributed Object Computing Conference, EDOC*, 2017.

[29] T. Chen and C. Guestrin, XGBoost: A scalable tree boosting system, in: *International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[30] E. Kutschinski, T. Uthmann and D. Polani, Learning competitive pricing strategies by multi-agent reinforcement learning, *Journal of Economic Dynamics and Control* **27**(11–12) (2003), 2207–2218.

[31] R. Schlosser, Dynamic pricing and advertising models with inventory holding costs, *Journal of Economic Dynamics and Control* **57** (2015), 163–181.

[32] A.V. Den Boer and B. Zwart, Simultaneously Learning and Optimizing Using Controlled Variance Pricing, *Management Science* **60**(3) (2013), 770–783.

[33] M. Fisher, S. Gallino and J. Li, Competition-Based Dynamic Pricing in Online Retailing: A Methodology Validated with Field Experiments, *Management Science* (2017).

[34] G. Vulcano, G. van Ryzin and R. Ratliff, Estimating Primary Demand for Substitutable Products from Sales Transaction Data, *Operations Research* **60**(2) (2012), 313–334.

[35] R. Schlosser and M. Boissier, Dealing with the dimensionality curse in dynamic pricing competition: Using frequent repricing to compensate imperfect market anticipations, *Computers and Operations Research* **100** (2018), 26–42.

[36] R. Schlosser and M. Boissier, Optimal price reaction strategies in the presence of active and passive competitors, in: *International Conference on Operations Research and Enterprise Systems*, 2017, pp. 47–56.

[37] G. van Rossum, Python Programming Language, in: *Proceedings of the USENIX Annual Technical Conference*, 2007.

[38] P.F. Dubois, K. Hinsen and J. Hugunin, Numerical Python, *Computers in Physics* **10**(3) (1996), 262–267.

[39] S. Behnel, R. Bradshaw, C. Citro, L. Dalcín, D.S. Seljebotn and K. Smith, Cython: The Best of Both Worlds, *Computing in Science and Engineering* **13**(2) (2011), 31–39.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort et al., Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research* **12** (2011), 2825–2830.

[41] S. Kachani and K. Shmatov, Competitive pricing in a multi-product multi-attribute environment, *Production and Operations Management* **20**(5) (2010), 668–680.

[42] S. Ito and R. Fujimaki, Optimization beyond prediction: Prescriptive price optimization, in: *International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1833–1841.

Appendix A. Notation Table and Additional Figures

Symbol	Description	Symbol	Description
a	Offer price	A	Set of admissible prices
$\vec{p}$	Competitors' prices	K	Number of competitors
δ	Discount factor for future profits	l	Holding costs per item and period
t	Time	b	Number of items ordered
B	Set of admissible order quantities	c_{fix}	Fixed order costs
c_{var}	Variable order costs	$C(b)$	Total order costs for ordering b items
$\lambda(a,\vec{p})$	Mean sales for one period with stable prices	$P_{t,t+h}(i,a\vert\vec{p})$	Probability to sell i items within $(t,t+h)$
N_t	Random inventory level at the time t	X_t	Random number of sold items until time t
Z_t	Number of orders made until time t	G_t	Random accumulated disc. profit from time t on
y	Dependent variable (number of sales)	$\vec{x}(h,a,\vec{p})$	Explanatory variables
M	Number of explanatory variables	h	Price reaction time
$\vec{\beta}$	Weights vector for linear regression	$\vec{\beta}^*$	Optimal weights for specific training data
$\tilde{\lambda}(h,a\vert\vec{p})$	Estimated mean sales for a time span h	$\tilde{P}^{(h)}(i,a\vert\vec{p})$	Estimated probabilities for a time span h
n	Inventory level	N_{max}	Maximum inventory capacity
$V^*(n,\vec{p})$	Value function	$V_t(n,\vec{p})$	Approximated value function
T	Number of periods/recursion steps	$v(n)$	Starting value for value function V_T
$a^*(n,\vec{p})$	Optimal pricing decision	$b^*(n,\vec{p})$	Optimal ordering decision

Table 6
List of variables and parameters.

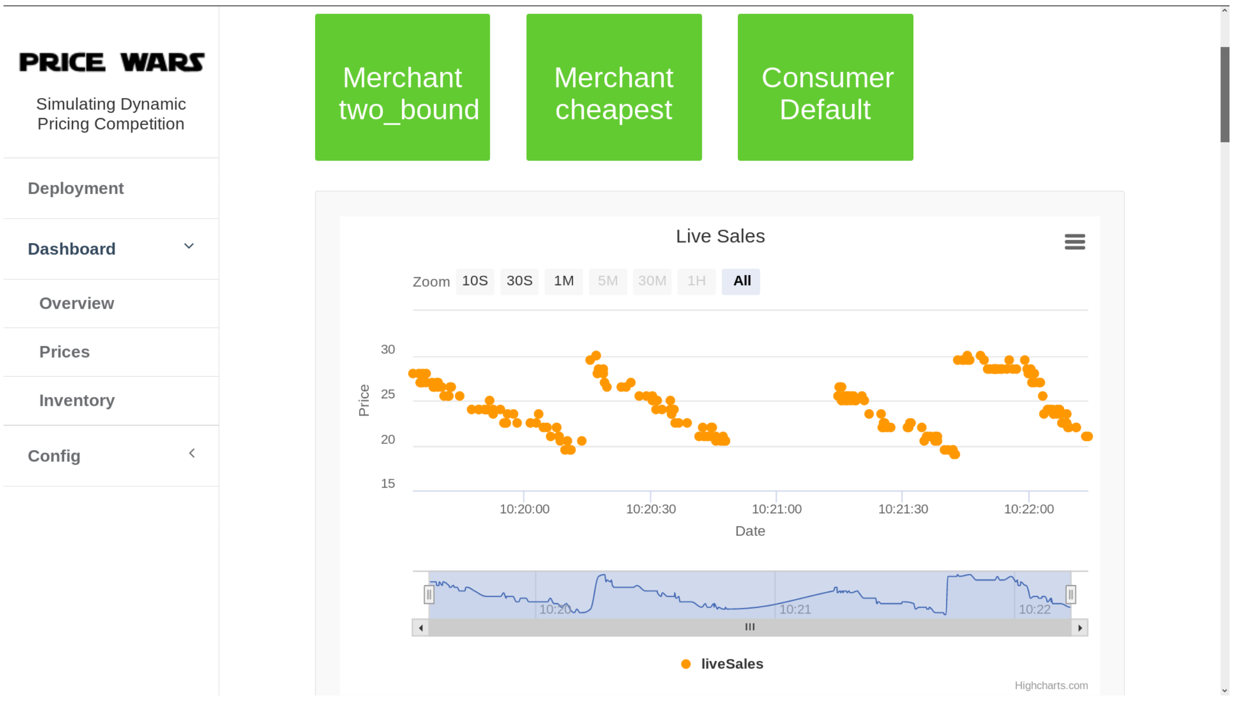

Fig. 9. Dashboard of the HTML-based frontend.