An Interactive Platform to Simulate Dynamic Pricing Competition on Online Marketplaces

Sebastian Serth*, Nikolai Podlesny*, Marvin Bornstein*, Jan Lindemann*, Johanna Latt*,
Jan Selke*, Rainer Schlosser‡, Martin Boissier‡, Matthias Uflacker‡
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
Email: *{firstname.lastname}@student.hpi.de, ‡{firstname.lastname}@hpi.de

***Abstract*—E-commerce marketplaces are highly dynamic with constant competition. While this competition is challenging for many merchants, it also provides plenty of opportunities, e.g., by allowing them to automatically adjust prices in order to react to changing market situations. For practitioners however, testing automated pricing strategies is time-consuming and potentially hazardously when done in production. Researchers, on the other side, struggle to study how pricing strategies interact under heavy competition. As a consequence, we built an open continuous time framework to simulate dynamic pricing competition called *Price Wars*. The microservice-based architecture provides a scalable platform for large competitions with dozens of merchants and a large random stream of consumers. Our platform stores each event in a distributed log. This allows to provide different performance measures enabling users to compare profit and revenue of various repricing strategies in real-time. For researchers, price trajectories are shown which ease evaluating mutual price reactions of competing strategies. Furthermore, merchants can access historical marketplace data and apply machine learning. By providing a set of customizable, artificial merchants, users can easily simulate both simple rule-based strategies as well as sophisticated data-driven strategies using demand learning to optimize their pricing strategies.**

I. E-Commerce & Dynamic Pricing Competition

The ongoing rise of e-commerce continuously changes the opportunities and challenges of marketplaces that merchants have to deal with. Merchants are now able to observe the market at any given point in time. Thus, they can react to changing conditions immediately. At the same time, the pressure to steadily adapt and react is increasing, while most merchants have only limited experience with such highly competitive markets and their long-term effects.

In this paper, we present the architecture and the technical implementation of our open source platform *Price Wars* that provides a playground for automated repricing strategies, also called *dynamic pricing*. The field of *dynamic pricing* is concerned with the creation, analysis, and application of highly dynamic strategies that change prices of products in reaction to certain variables such as the competitor's actions, demand estimations, or certain consumer behaviors.

Usually, the first strategy being applied in competitive markets is based on simple rules that the merchant defines based on his experience (and gut feeling). This approach is also used by the majority of current repricing services, where merchants provide thresholds for certain rules (e.g., "set the second lowest price, unless price is smaller than threshold t").

But more recently, research and industry started to combine achievements of price optimization from the research field of *operations research* with the achievements in data-driven procedures from the field of *computer science* such as machine learning [1, 2]. This development is basically a catching-up with similar approaches already being applied in the field of algorithmic trading or high-frequency trading on stock exchanges. These approaches are far more sophisticated than currently observable approaches on marketplaces such as Amazon, where most merchants thrive to be amongst the cheapest competitors, eventually leading to a typical *race to the bottom*. More sophisticated strategies optimize for long-term profits, consider restocking, and predict competitor actions.

Unfortunately, for practitioners as well as for researchers, a common platform to develop, test, and evaluate pricing strategies is missing. Merchants lack the possibility to test their strategies appropriately before deploying them in production, potentially causing significant economic problems. For researchers, there are no open platforms for simulating large pricing competitions with various pricing strategies competing with each other. More importantly, there are no platforms that provide the means to deploy data-driven strategies.

We picked up the challenge to create an environment to imitate different market situations and test how pricing strategies interact when influencing each other, including simple rule-based merchants as well as sophisticated data-driven merchants using machine learning models.

Throughout this paper, we make the following contributions:

- We discuss how a distributed microservice-based architecture helps to achieve two of our mains goals (Section III): (i) scalability to handle many concurrent merchants and a large stream of consumers and (ii) flexibility to cover a wide set of pricing scenarios.
- We present our simulation platform for dynamic pricing competition. The characteristics of the platform mirror online marketplaces such as Amazon or eBay in order to simulate production pricing competition (Section IV).
- We show how to handle communication and data flow in such a distributed marketplace simulation and present the provided bootstrap behaviors both for merchants as well as consumers (Section V).
- We simulate and evaluate the outcome of dynamic pricing competition with multiple concurrent merchants deploying a wide range of strategies (Section VII).

2325-6362/17 $31.00 © 2017 IEEE
DOI 10.1109/EDOC.2017.17

II. Related Work

Simulation platforms have been built in various fields in order to simulate complex competition scenarios, e.g., in the field of real-time bidding [3], marketing [4], or electricity markets [5]. For this work, we focus on competition in e-commerce applications.

Selling products is a classical application of revenue management theory. The problem is closely related to the field of dynamic pricing, which is summarized in the books [6], [7], and [8]. The surveys [2] and [9] provide an excellent overview of recent pricing models under competition.

Marketplaces in the real world are characterized by many competitors, complex offers, and limited demand information. The derivation of sophisticated pricing strategies is challenging. Theoretical models with multiple competitors, multiple products, multiple product features and stochastic demand intensities that are not highly stylized are analytically not tractable. Hence, simulation approaches have to be used.

With the increasing use of online marketplaces in the last decades, cf. [10], the dynamic pricing topic got more and more important due to the possibility to change virtual prices within seconds, whereas physical prices printed on products in stores were much harder and slower to influence. In addition, the consumers' behavior changes in this context as Kannan and Kopalle [11] found by comparing the physical value chain with the virtual-information-based value chain.

This showed that the consumer behavior is just as important in dynamic pricing contexts as the pricing behavior is. However, little effort has been made to build simulation platforms that ease the development and evaluation of pricing algorithms while taking all of these factors into account. Morris [12] developed such a platform in 2001. However, it is limited in its capabilities, especially since it does not allow large e-commerce simulations. The simulation program runs on a single machine, offers a limited set of consumer behaviors, simulates solely finite sales horizons, and seller pricing updates happen only in discrete time intervals that are predefined by the system. Therefore, reactions to other merchants are very limited. This does not represent the current situation on online marketplaces very well and thus restricts possible simulation scenarios. Other platforms have similar limitations in their capabilities (e.g., Pinto et al. [5] or DiMicco et al. [13]).

III. Design Goals

Given the current, very limited possibilities to simulate a huge, dynamic online marketplace, we came up with a set of specific design goals and restrictions, that we wanted to fulfill to make the platform and the resulting simulation as realistic, dynamic, and reactive as possible.

Event-Driven Communication: Allow pricing algorithms to do fully dynamic price updates and look-ups in continuous time, i.e., at any time without being refrained by discrete time intervals to enable event-driven pricing strategies. All merchants act asynchronously and have to be able to deal with potentially outdated data (see [14]).

Realism: Allow to define arbitrary streams of customers with complex buying behavior. Enforce restrictions present in production marketplaces such as a limited amount of price updates per time interval to avoid advantages by constantly changing prices.

Scalability and Adaptability: The system should be scalable and easily expandable to account for high loads, the simulation of large marketplaces, or the addition of completely new components and features.

Flexibility: Provide the required flexibility to make the system adaptable to user needs, research questions, and new possible design goals.

Classes of Strategies: Also, the simulation of both, rule-based and data-driven pricing strategies should be possible, the latter using machine-learning techniques to model the customer choices and sales probabilities.

Usability: Allow merchants to directly participate in the competition with minimal overhead, i.e., by providing a simple bootstrap merchant written in Python. Furthermore, access to historical data for learning purposes shall be easy via a simple interface such as CSV files.

In [15], Hamilton lists additional aspects we had to tackle with building a microservice-based (or the more recent term "cloud-native") application.

IV. Architecture

Confronted with the challenge of creating a highly flexible infrastructure for simulating a marketplace with different merchants and consumers, a microservice-based architecture was created allowing the user to scale, exchange, or add single services ad-hoc and on demand. Each service within our architecture implements one business artifact. This architecture pattern comes with the cost of a communication overhead and requires farsighted API design.

Figure 1 depicts the architecture. We understand a single instantiation of this architecture as one simulation universe, meaning that key components are unique in this setup.

When initiating a new simulation universe, it comes along with the marketplace component, as well as the producer component and a management user interface (UI) for controlling each service. While the producer offers products, the marketplace holds the current market situation, handles price updates and purchases of goods. Each transaction processed by the producer and marketplace is logged to a stream database, namely Apache Kafka [16]. Further, those logs are being analyzed and aggregated through a streaming data processing component which, in our case, is Apache Flink [17] and written back into a new Kafka topic. Those details can then be accessed selectively through a web socket connection or REST interface provided by our *Kafka Reverse Proxy* service, which is responsible for both securing accesses to Kafka as well as caching. To generate the actual marketplace load, streams of different kinds of customers are simulated. Merchants may join and participate in the simulation. By default, six merchants are deployed with predefined strategies. Their behaviors are

described in Section V while the choreography of the single services is delineated in Section IV-A.

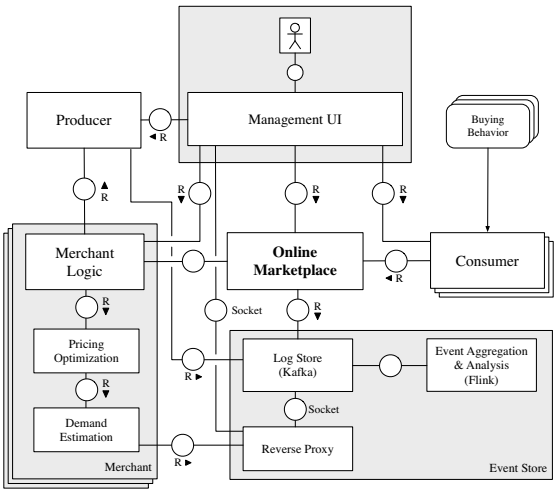

Fig. 1. FMC diagram of the platform's architecture

Our model is characterized by the components displayed in Figure 1. Their interplay can be described as follows:

Marketplace: Collects and updates the product offers of the competing merchants. The competitors' offers include two features: price and quality. The marketplace manages interested customers and buying events.

Consumer: Arbitrary (random) streams of interested customers can be defined. Various customer choice behaviors can be defined. The decisions whether a customer buys a product and which offer/merchant is chosen can be defined by probabilities that can depend on all parameters of the current market situation.

Merchants: Update prices of their products based on requests of current market situations. Rule-based as well as data-driven strategies can be applied. Data of observed market situation as well as a merchants' sales data are stored in the log store and can be used to estimate sales probabilities using various machine learning techniques.

Producer: Organizes the replenishment of all merchants. Merchants are provided with new products according to a (fair) distribution. This way the performances of competing strategies are not affect by asymmetric reordering strategies. However, the model would also allow to let the merchants choose their replenishment policy.

Management UI: Allows to adjust the customer behavior as well as model parameters such as adjustment limits or replenishment rules. Secondly, the UI allows to observe the evolution of current market situations and price trajectories over time. The strategies' performances are measured by different *key performance indicators* (KPIs), including average or accumulated profits/revenues, number of sales, number of price adjustments, and more.

A. Service Choreography

We developed our platform using a microservice-based architecture. The communication between the various components is done via RESTful APIs using JSON. This helps to fulfill the design goal of reactivity, since each service is totally autonomous and can contact every other service via their exposed RESTful API at any time. There is no need for explicit simulation ticks or action requests.

Due to the microservice-based architecture and the design goal to allow fair competition of different merchants without fraudulent actions, all important routes are secured by authorization tokens. For authenticating all participants in the simulation without the need of a centralized authentication server, a hash-based token and identification system was introduced. It enables the ID-based logging of event messages corresponding to one merchant or consumer. One of the causes to implement this decentralized authentication system was to reduce the number of requests during the simulation.

B. Event Log Analysis with Kafka and Flink

All transactions during the simulation are persisted using the stream database Kafka. The log is sourced by the marketplace and the producer with JSON-formatted messages and stored in so-called *topics*. Each topic can be consumed independently with the preserved order of events. For example, Flink jobs read all messages within a predefined time range (such as one minute or one hour). Each Flink job aggregates different events from multiple topics and logs the result in an own Kafka topic. The jobs are executed every ten seconds or every minute, depending on the desired aggregation level.

The aggregated messages contain the calculated market share per merchant per time slice and are used by the management UI to display corresponding statistics. Because Kafka only offers stream-based access, the Kafka reverse proxy is responsible for fetching a preset number of events on request to serve these via a RESTful API. Amongst other reasons, this additional layer is required for technical reasons since available Kafka libraries use timeouts to wait for new events during read or require a fix number of messages to fetch. The reverse proxy uses web sockets to push new events directly to the management UI in browsers for live statistic updates. In addition to being the main statistic source for the management UI, the reverse proxy is also responsible for serving a cleaned event log to merchants and to export topics to CSV files. These exports are mainly used for data-driven merchants. The messages contain a specific merchant ID and are filtered for that ID when exporting data. Therefore, the corresponding merchant token is required and used to remove sales data of other merchants.

V. Behaviors

The simulation framework provides several behaviors for merchants, as well as the consumers by default. In this section, we will outline and describe the underlying behaviors which are currently available.

A. Consumer Behavior

The platform allows to define various consumer behaviors. In our model, we distinguish between arriving interested consumers and buying consumers that decide to choose one of the offers after reviewing all available offers.

In our continuous time framework, we simulate randomized streams of interested consumers. In our implementation, we simulate stochastic arrival processes that are based on uniformly or exponentially distributed random variables. This way, we obtain randomized waiting times between two occurring consumers. Furthermore, the intensity, i.e., the average number of interested consumers within a certain time interval can be specified to model scenarios with low or high demand.

The buying behavior of consumers is modelled as a weighted mixture of different predefined selection behaviors. Those behaviors range from very subtle approaches like "buy the cheapest offer", or the n^{th}-cheapest according to a predefined probability distribution up to more sophisticated methods allowing to describe more realistic consumer behaviors.

In order to imitate consumers that balance an offer's features such as prices and quality according to specific weights, we use randomized scoring functions. The weighting parameters for specific markets can be calibrated using real world data. In our model, we also used coefficients which were extracted from Amazon market data provided by a big book retail company, cf. [18]. The authors use a weighted logistic regression model to quantify the buying behavior for different products. This model is included as one of the default consumer behaviors. Other behaviors can be easily added. The decisions whether a consumer buys a product and which offer/merchant is chosen can be defined by probabilities that can depend on all parameters of the current market situation.

Our framework also allows to reconfigure the buying behavior on-the-fly by adjusting arrival intensities and the consumer's selection probabilities.

B. Merchants Strategies

The merchant component has one main task: The price adjustment for a given product in a current market situation. This can either be adding a new product purchased from the producer, or updating an existing product that is already on the marketplace. This calculation has to trade-off between maximizing the probability to sell a product and maximizing the own profit into account. To allow an easy start, the platform already offers a set of five rule-based strategies and one data-driven approach that implements a demand learning strategy based on logistic regression [19].

The replenishment of each merchant is organized as follows. Whenever an item is sold, a randomized product with a randomized quality is assigned to the corresponding seller. This way, the performance comparison of different repricing strategies is not affected by the sellers' ordering policies.

1) Rule-Based Strategies: The simple, rule-based behaviors include response strategies such as "Be the cheapest", "fixed price", "Randomly be the 1st, 2nd or 3rd cheapest ('random third')", and the "gas station strategy" (also called "two-bound" in our system, cf. [20]). The "two-bound" strategy sets a minimum and maximum price, or a minimum and maximum profit margin, respectively, that can be added to the product's purchase price. The strategy keeps undercutting the cheapest competitor price as long as that price exceeds the minimum bound. If the cheapest competitor price is either below the minimum bound or above the maximum bound, the strategy adjusts the price to the upper price bound. If the "two-bound" strategy is played against, e.g., another "two-bound" merchant or the "Be the cheapest" merchant, we obtain cyclic price patterns that are of staircase type.

The rule-based strategies described above can also be refined such that the quality level of different products is taken into account, e.g., by using specific mark ups on price. All of these classes of behaviors have certain base settings that further determine the behaviors and that can be adjusted during run time. Minimum profit margins as well as upper price bounds can also be defined.

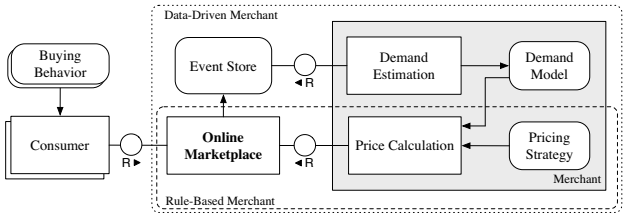

Fig. 2. FMC diagram depicting the general process flow of the simulation platform [21].

2) Data-Driven Strategies: On the one hand, our framework allows to apply rule-based strategies, which are widely used in practice as they are transparent and simple to implement. On the other hand, we supply the means to implement data-driven strategies by providing access to historical market data. Data-driven strategies are increasingly used in production and have the potential to replace rule-based strategies in the future. The application and computation of data-driven strategies requires an extended framework which allows to store and to process market data. Hence, in our framework, merchants have to be able to exploit their observable data to estimate demand using various machine learning techniques. As a result they can set up a "demand model" which allows, e.g., to quantify sales probabilities for all potential offer prices and any market situation. This makes it possible to support the price calculation (see Figure 2). The estimated sales probabilities can, for instance, be used to optimize expected profits.

To demonstrate the possibility to apply data-driven strategies, we implemented such a merchant. The merchant has access to his own sales data and historical market situations for training the demand model. The access to sales data of other merchants is restricted by the system, similar to real world marketplaces (see Section IV-B). The data used to learn a regression model are past market situations in which the merchant offered his own products. The key idea is to identify the likelihood of selling items conditioned on the underlying market situation.

The dependent variable will typically be the number of sales within a certain time interval. The explanatory variables can be used to describe the attractiveness of an offer compared to the competitors' offers. Finally, the estimated sales probabilities can be used to optimize the pricing strategy.

Our data-driven approach follows a pricing strategy aimed at maximizing expected short-term profits. To do so, a logistic regression model is trained, based on features, such as (i) the distance to the cheapest competitor, (ii) the price- and quality-rank, (iii) the number of competitors, or (iv) the average price of the product on the marketplace.

As a result, the logistic regression model outputs the probability with which a product is sold within a certain period of time, given a certain market situation and a specific offer price for that product. To maximize the expected profit, this merchant comes up with a set of possible prices, calculates the associated selling probabilities, and multiplies these probabilities with the corresponding price. The expected profits for all possible prices are compared and the price adjustment that maximizes the expected short-term profits is chosen.

A bootstrap merchant in Python was implemented to facilitate an easy on-boarding and extension of the platform by adding custom merchants. All default merchants are based on this bootstrap merchant and included for further reference.

VI. User Interface

The HTML-based *management front-end* enables users to configure, operate, and orchestrate the different microservices all in one place without any programming effort. Users can start and stop a simulation, merchants as well as other components directly using the front-end. All exposed settings for the individual behavior of each merchant and the consumer can be viewed and updated. The stateless components of the simulation, such as the producer and the marketplace, cannot be stopped or started, but configured here as well.

If a user wants to register a new merchant to participate in the simulation, they can use the front-end to register a new endpoint under which the merchant is running, and in return receive a secret token that is used for authorization.

The front-end consists of several pages where each page focusses on a particular aspect that practitioners or researchers might be interested in. The main screen of the platform is the dashboard. The dashboard visualizes all streaming sales in real-time and shows which registered merchants are currently participating. Further, the dashboard allows compare the revenues that are associated to the merchants' pricing strategies. Figure 3, for instance, shows current revenues of the competing firms over time. This way, it can be observed how certain changes affect the merchants' performance. Users can zoom in and out in this graph, e.g., to analyze long-term behavior or observe trends.

Furthermore, the front-end visualizes the pricing interaction of competing merchants, simplifying the process of comparing different pricing strategies (see [21] for more details). Figure 4 illustrates the price trajectories of three competing merchants over time. Mutual price reactions typically lead to cyclic pattern

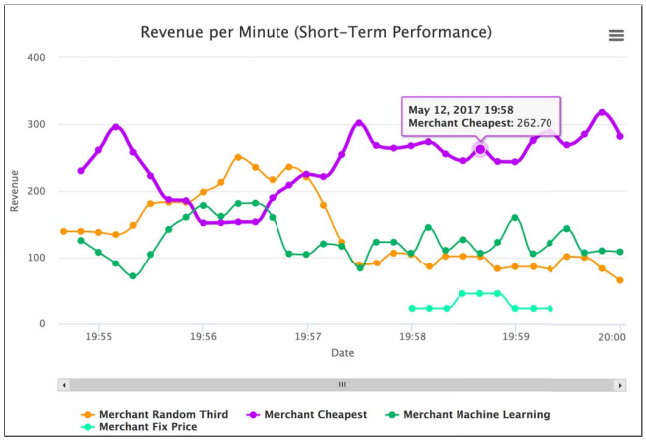

Fig. 3. Screen shot of a platform graph comparing merchant revenues over time using window aggregations of Flink.

in which competitors keep undercutting each other (so-called *raise to the bottom*), a pattern often observed in practice. If the price level is sufficiently low, some pricing strategies start to significantly raise the price in order to enable future profits (cf. the merchant in orange).

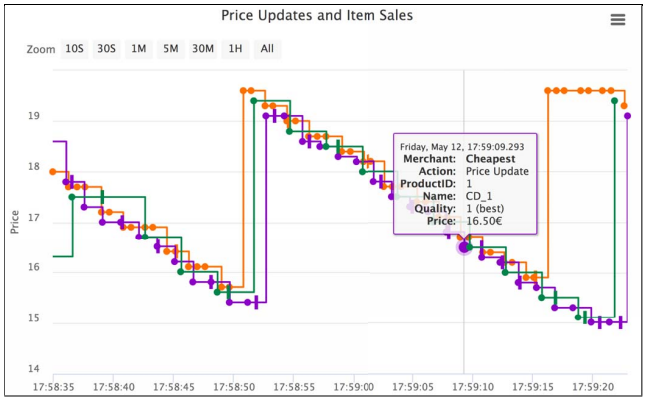

Fig. 4. Platform screen shot of price trajectories over time.

VII. Evaluation

We used the platform to get an overview of the provided merchants' behavior and performance. We expect that the exemplary data-driven approach based on a logistic regression model to estimate the current demand promises higher profits than common rule-based behaviors (cf. [18]), due to a better adjustment to the consumer behavior in pricing (e.g., changing consumer behaviors over time or willingness to pay for better product qualities).

First, we performed a one-on-one evaluation of the six provided merchant behaviors (see Section V-B). We ran each evaluation for 20 minutes, with one data-driven consumer (see Section V-A), and one product with 4 (random) qualities.

We observe that the realized profits are significantly influenced by the two competing strategies, where revenues are

significantly lower when at least one merchants pursues a rather aggressive pricing strategy and consumers are price sensitive. Also, we saw that the data-driven strategy outperforms all rule-based strategies in the one-on-one evaluation.

Second, we ran an oligopoly evaluation with all merchants running at the same time to see whether the data-driven merchant would beat the other rule-based merchants. The results are shown in Table I. The numbers are the profit gained by each merchant after 20 minutes of simulation.

TABLE I
OLIGOPOLY: GENERATED PROFITS BY ALL MERCHANTS COMPETING CONCURRENTLY AGAINST EACH OTHER.

Merchant	Generated Profit
Cheapest	817.80
Second Cheapest	784.00
Random Third	679.79
Two-Bound	1449.21
Fix Price	585.60
Machine Learning	1826.05

The data-driven strategy dominates the rule-based strategies in a one-on-one setup as well as in an oligopoly setup. The best rule-based strategy was the two-bound strategy which is reasonable aggressive but not destructive. However, the two-bound strategy lost against the data-driven merchant in both of our simulations. To get more comprehensive results, further tests have to be run with other data-driven strategies to investigate how they perform when playing against each other. The respective consumer behavior also has a large impact.

Our results demonstrate the usefulness and applicability of the platform. We are able to measure the performances of strategies of competing merchants in different setups. Our examples show the importance of sophisticated pricing strategies as they can have a vast impact on profits. Moreover, we showed that data-driven strategies are able to outperform simple rule-based strategies.

Our framework allows to further study the unforeseeable interplay of various repricing strategies. It enables the user to run quick and complex simulations of real-world setups with an almost arbitrarily high variety of merchants and strategies. At the same time, it offers comprehensive evaluation metrics. The user sees immediate feedback on the performance of each merchant, fulfilling our design goals (Section III).

VIII. CONCLUSION

We presented a distributed and scalable platform to simulate dynamic pricing competition allowing both practitioners and researchers to study the effects of automated repricing mechanisms competing with each other using market scenarios that mimic real-world marketplaces. For practitioners, the platform further provides a possibility to evaluate their pricing strategies appropriately before releasing them in production.

The platform has a scalable microservice-based architecture and is able to handle dozens of concurrent merchants and processing thousands of consumer requests per seconds. We built the platform in a way that one can participate and deploy own merchants with only a few lines of Python code. The platform provides an easy access to historical market data for facilitating data-driven strategies.

Moreover, we compared traditional rule-based strategies with simple data-driven strategies. Data-driven merchants are superior to rule-based approaches as soon as a sufficiently large data set has been gathered.

The platform's source code and the technical documentation are publicly available on GitHub[1].

REFERENCES

[1] R. Schlosser and M. Boissier, "Optimal price reaction strategies in the presence of active and passive competitors," in *Proc. ICORES*, 2017, pp. 47–56.

[2] A. V. den Boer, "Dynamic pricing and learning: Historical origins, current research, and new directions," *Surveys in Operations Research and Management Science*, vol. 20, no. 1, pp. 1–18, 2015.

[3] W. C. Wu, M. Yeh, and M. Chen, "Predicting winning price in real time bidding with censored data," in *Proc. 21th ACM SIGKDD*, 2015, pp. 1305–1314.

[4] Y. Tkachenko, M. J. Kochenderfer, and K. Kluza, "Customer simulation for direct marketing experiments," in *IEEE DSAA*, 2016, pp. 478–487.

[5] T. Pinto *et al.*, "Adaptive learning in agents behaviour: A framework for electricity markets simulation," *Integrated Computer-Aided Engineering*, vol. 21, no. 4, pp. 399–415, 2014.

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

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

[8] I. Yeoman and U. McMahon-Beattie, *Revenue management: a practical pricing perspective*. Springer, 2011.

[9] M. Chen and Z.-L. Chen, "Recent developments in dynamic pricing research: multiple products, competition, and limited demand information," *Production and Operations Management*, vol. 24, no. 5, pp. 704–731, 2015.

[10] D. Popescu, "Repricing algorithms in e-commerce," *INSEAD Working Paper Series*, 2015.

[11] P. Kannan and P. K. Kopalle, "Dynamic pricing on the internet: Importance and implications for consumer behavior," *International Journal of Electronic Commerce*, vol. 5, no. 3, pp. 63–83, 2001.

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

[13] J. M. DiMicco, P. Maes, and A. Greenwald, "Learning curve: A simulation-based approach to dynamic pricing," *Electronic Commerce Research*, vol. 3, no. 3-4, pp. 245–276, 2003.

[14] P. Helland, "Data on the outside versus data on the inside," in *CIDR 2005, Online Proceedings*, 2005, pp. 144–153.

[15] J. R. Hamilton, "On designing and deploying internet-scale services," in *Proc. LISA*, 2007, pp. 231–242.

[16] J. Kreps, N. Narkhede, J. Rao *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, 2011, pp. 1–7.

[17] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache Flink™: Stream and batch processing in a single engine," *IEEE Data Eng. Bull.*, vol. 38, no. 4, pp. 28–38, 2015.

[18] R. Schlosser *et al.*, "How to survive dynamic pricing competition in e-commerce," in *Proc. Poster Track of the 10th ACM RecSys*, 2016.

[19] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.

[20] M. D. Noel, "Edgeworth price cycles, cost-based pricing, and sticky pricing in retail gasoline markets," *The Review of Economics and Statistics*, vol. 89, no. 2, pp. 324–334, 2017/04/25 2007.

[21] M. Boissier *et al.*, "Data-driven repricing strategies in competitive markets: An interactive simulation platform," in *Proceedings of RecSys '17, Como, Italy, August 27-31, 2017*, 2017.

[1] Platform repository: https://github.com/hpi-epic/masterproject-pricewars