Algorithms in the Wild: Experimental Evidence from an Online Marketplace

Vito Stefano Bramante
University of Bologna

Emilio Calvano
University of Rome-Tor Vergata

Giacomo Calzolari
European University Institute

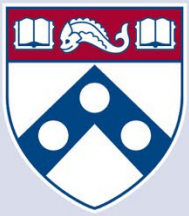
Maximilian Schaefer
Yale University & University of Bologna

EODS
Economics of Digital Services
@Penn

Center for Technology, Innovation & Competition
University of Pennsylvania

The Warren Center
for Network & Data Sciences

Penn
University of Pennsylvania



Economics of Digital Services (EODS)

EODS is an initiative of the University of Pennsylvania's Center for Technology, Innovation & Competition (CTIC) and Warren Center for Network & Data Sciences. Its aim is to generate independent research on the ways digital platforms access user data in support of their business models and the implications for antitrust policy and law. The initiative was funded by a major grant from the John S. and James L. Knight Foundation to support scholarly inquiry and novel approaches in the evolving digital age.

To learn more about the initiative, visit www.pennEODS.org.

Center for Technology, Innovation & Competition (CTIC)

CTIC is an interdisciplinary academic center at the University of Pennsylvania Carey Law School that bridges law and technology for academia and students. Focusing on intellectual property, competition, Internet and privacy, emerging technologies, health care, and entertainment and media law and policy, CTIC conducts foundational research that shapes the way legislators, regulatory authorities, and scholars develop policy and legal frameworks. CTIC also produces programming that explores the full range of scholarly perspectives, engages with industry experts, and educates the next generation of technology scholars, lawyers, and policymakers. *www.pennCTIC.org*

The Warren Center for Network & Data Sciences

The University of Pennsylvania's Warren Center fosters research and innovation in interconnected social, economic, and technological systems. Collaborating with Penn affiliates, it focuses on the role of data and algorithms to understand networked systems and how they can improve lives. The center also produces events that connect researchers, students, and entrepreneurs across the spectrum of network science. *www.warrencenter.upenn.edu*

Algorithms in the Wild: Experimental Evidence from an Online Marketplace

PRELIMINARY, PREPARED FOR THE EODS RESEARCH SYMPOSIUM
Click here for most recent version.

Vito Stefano Bramante[*] Emilio Calvano[†]

Giacomo Calzolari[‡] Maximilian Schaefer[§¶]

September 1, 2022

Abstract

Can off-the-shelf repricing algorithms used in online marketplaces learn collusive strategies that harm consumers? To shed light on the sophistication of commercial repricing technology, we deploy our own repricing software on an online platform. We implement a EXP3 repricing algorithm and compare its performance against the artificial intelligence algorithm of a selected commercial repricer. We start by establishing a performance benchmark for myopic pricing strategies when faced with a mechanical repricing rule that undercuts rivals' prices. When competing against the mechanical rule, our EXP3 algorithm achieves a better performance than the commercial software. Additionally, our EXP3 algorithms out-competes the commercial repricing software in a direct competition. These results cast doubt on the sophistication of the selected commercial repricing software. Designing algorithms that allow for intertemporal trade-offs is a prerequisite for collusion to arise. In simulations, we show that forward-looking strategies can be learned at low costs. This provides the basis for a more in-depth investigation of forward-looking algorithms, and, hence, collusion in future iterations of this work.

JEL Codes: D21, D43, D83, L12, L13
Keywords: Algorithms, Collusion, Platform Markets

[*]University of Bologna, vitostefano.bramant2@unibo.it
[†]University of Rome - Tor Vergata, emilio.calvano@gmail.com
[‡]European University Institute, giacocalzolari@gmail.com
[§]Yale University & University of Bologna, maximilian.schaefer@yale.edu
[¶]This research is funded by research grants from the Center for Technology, Innovation and Competition at the University of Pennsylvania, the Digital Economics Research Network, the European University Institute Research Council Funding, and PRIN 2017.

1 Introduction

The use of repricing software has become ubiquitous in recent years: One example is the case of gasoline markets in which the widespread adoption of repricing software by large companies has attracted the attention of both academia (Assad et al., 2020) and the media.[1] However, the use of repricing software is not limited to large firms. The emergence of peer-to-peer marketplaces such as Amazon, Airbnb, Booking, and eBay has given rise to a rich landscape of software companies that offer affordable off-the-shelf repricing solutions for small sellers.[2]

In the context of online platforms, sheer speed is often advertised as the main benefit of using repricing software: Automatically monitoring and reacting to rival prices allows to relentlessly undercut slower sellers, who set prices manually. This, in turn, allows sellers to capture consumer demand because online platforms prominently feature the seller with the lowest price. While fierce price competition is likely to benefit customers through lower prices, the possibility that repricers might be powered by intelligent algorithms has also raised concerns that they might autonomously learn sophisticated strategies to charge higher prices, i.e. that they might autonomously learn to collude.[3]

The study of Calvano et al. (2020) has provided the proof-of-concept that even simple artificial intelligence algorithms have the capability of sustaining collusion by autonomously learning to punish rivals who deviate from the collusive agreement. This lends credibility to the concerns related to algorithmic collusion, especially since many repricing companies advertise the use of artificial intelligence algorithms.

However, empirical evidence assessing the real-world effects of repricing software remains scarce. While the finance literature has started studying the subject of algorithmic trading already one decade ago, it does not directly speak to collusion (Hendershott et al., 2011; Chaboud et al., 2014). Additionally, it is questionable how findings from sophisticated financial markets, where sellers and buyers both use software, extend to consumer mass markets, where typically only the seller-side uses software. Assad et al. (2020) is among the first studies finding evidence consistent with a chilling effect on competition in a consumer market.

One shortcoming of existing real-world studies is the imperfect ability of researchers to identify the algorithms in use, as companies are naturally reluctant to reveal their pricing technology. This raises questions about the sophistication of the adopted repricing technologies and, ultimately, leaves open the possibility that reported findings might not capture the effect of machine intelligence. For example, it has been pointed out that high prices might be the consequence of the

[1] See https://www.economist.com/finance-and-economics/2017/05/06/price-bots-can-collude-against-consumers and https://www.wsj.com/articles/why-do-gas-station-prices-constantly-change-blame-the-algorithm-1494262674 (last accessed: August 22, 2022).

[2] A Google search for "algorithmic repricer" on August 22, 2022 returned 52,800 results.

[3] In fact, the possibility that algorithms might have the ability to collude has already drawn the attention of competition authorities around the globe. For example, the topic of algorithmic collusion has been discussed at the 7th session of the FTC Hearings on competition and consumer protection (November 2018). Furthermore, white papers of the OECD (2017) and the British CMA (2021) also discuss the subject.

algorithm's failure to optimize, providing an interesting contrast to the prevailing narrative (Cooper et al., 2015).

Our study sets out to address this shortcoming by leveraging the environment created by peer-to-peer marketplaces, which allows researchers to implement real-world repricing software in a controlled yet realistic environment. To this end, we create two seller accounts on a online marketplace, stock them with goods, and engage in sales activity. In addition, we create our own repricing software, which allows us to create arbitrarily sophisticated opponents against which we let commercial repricing software compete.

One major benefit of our setting is that it enables the controlled implementation of commercial repricing software and the extensive monitoring of the market environment, ruling out possible alternative explanations for increasing prices, such as changes in demand and supply conditions.[4]

In this article, we present the results from an experimental protocol in which we compare the performance of our own algorithm against the performance of a selected commercial repricing company that advertises the use of Artificial Intelligence algorithms.[5] Our own algorithm is a so-called EXP3 algorithm, which belongs to the class of *no-regret* reinforcement learning algorithms. No-regret algorithms guarantee a payoff close to the payoff of the ex-post optimal myopic strategy.

To compare the performance between our own algorithm and the commercial algorithm, we start by characterizing the optimal strategy of a seller when confronted with an opponent that always undercuts her prices. In the online marketplace selected for our experiment, the cheapest seller is prominently advertised to customers arriving on the product page. The strategy to marginally undercut rival prices to be the cheapest seller and to capture consumer demand constitutes a plausible competitive benchmark in the myopic-seller setting we consider. In fact, it is an option that many repricing companies offer as a default repricing rule.

After establishing the optimal response, we let both our own and the commercial repricing software compete against the opponent always undercutting rivals' prices. Our EXP3 algorithm is more successful in learning a strategy close to the optimal strategy than the commercial algorithm. Additionally, our algorithm outperforms the commercial repricing software in direct competition in the sense that it obtains the

[4]The benefits of this research approach, however, are to be weighed against the complications researchers face when dealing and interacting with actual markets. Approaches based on simulations, lab experiments and empirical analysis, developed so far, do not require to interact directly with actual market. We had instead to address a series of issues that are uncommon to academic research.

[5]As appears to be common in the industry, the selected company does not provide information on the type of algorithm used.

advertised seller position more often than the commercial repricing software. These results suggest that the level of sophistication of the selected commercial repricing software is not particularly evolved.

While our experimental protocol provides a measure to benchmark the performance of algorithmic repricers for myopic strategies, the case of algorithms able to implement intertemporal trade offs is not yet covered. The ability to incorporate the future consequences of current decisions is a prerequisite for proper collusion, which requires balancing the trade-offs between the short-term benefits of subverting the collusive agreement with the associated long-term costs of lower prices resulting from the subversive action.

In this context, the main challenge is to develop an algorithm that is capable of integrating intertemporal trade-offs while learning in a reasonable time to avoid incurring prohibitive learning costs. Algorithms that quickly learn strategies that allow for intertemporal trade-offs pose a significant challenge that researchers in laboratory environments can afford to ignore.[6] We present findings based on an algorithm that successfully learns the optimal intertemporal strategy against the mechanical algorithm relentlessly undercutting rivals' prices. This proof-of-concept from the laboratory setting still needs to be validated in a real world environment.

Looking ahead, the case of algorithms capable of implementing intertemporal trade-offs needs to be elaborated further. It remains to be determined whether it is possible to quickly train forward-looking algorithms that can successfully compete against a wider variety of strategies - not only the algorithm relentlessly undercutting rival prices. Our goal is to assess the feasibility of algorithmic collusion in real world market places using an algorithm that learns quickly and that accommodates a wide range of possible rival strategies.

In future work, we aim to deploy a revised experimental protocol, which will cover the myopic and forward-looking case, on a broader variety of repricing companies to provide a more comprehensive overview about the typical capabilities of off-the-shelf repricing solutions offered to small sellers in online marketplaces.

The remainder of the article is structured as follows: Section 2 introduces background information about the online marketplace and the commercial repricer we selected for our experiment. Section 3 introduces our current experimental protocol, the optimal myopic strategy, and the myopic EXP3 algorithm. Section 4 presents the results obtained from the experimental protocol. Section 5 deals with algorithms able to learn intertemporal trade-offs. Section 6 concludes.

[6]For example, Calvano et al. (2020) allow for several hundreds of thousands of periods of learning.

2 Background

The Online Marketplace

Our experiment is implemented in an online marketplace. We create two seller accounts and engage in real sales activity. We chose a cheap to supply product to implement our experiment. When customers search the product, they are displayed a picture of the product together with the price of the advertised seller. The online marketplace does not directly disclose how the advertised seller is selected. However, in the context of our experiment, the seller with the strictly cheapest price was almost certainly the advertised seller as is shown in Figure 1.

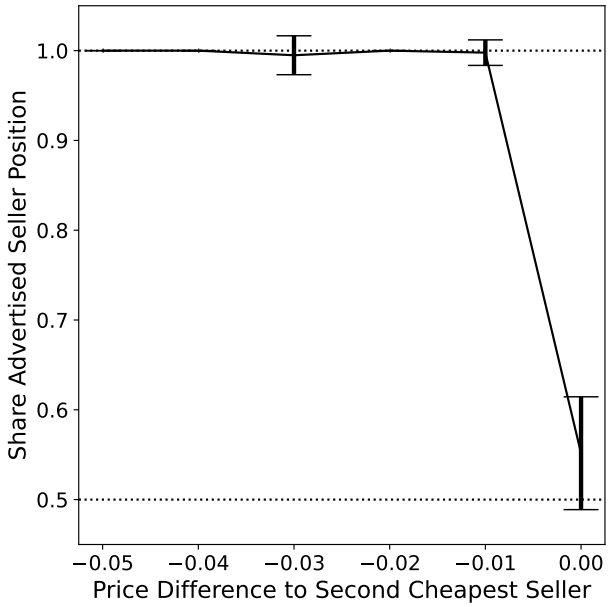


Figure 1: Share of Advertised Seller Position

Notes: The graph shows the share of the advertised seller position for the cheapest seller as a function of the price difference to the second cheapest seller. The two dotted horizontal lines show the 50 and 100 percent thresholds, respectively. The data were obtained from one of our experiments. The error bars show the 95% confidence intervals. We observe no variation for price differences of two and five cents.

Consumers can select the offer of the cheapest seller by clicking on a salient link which allows them to buy the product. The prices of non-advertised sellers can be accessed by clicking on a non-salient link, which leads to the list of non-advertised

sellers. There is no reliable information on the aggregate share of consumers ordering products from non-advertised sellers. However, it is clear that being the advertised seller increases the likelihood of sales significantly.

The Commercial Repricing Software

The repricing company we selected for our experiment offers off-the-shelf repricing solutions for several online marketplaces. It offers a variety of repricing features, such as rule based repricers and AI-driven repricers. Typically, rule-based repricers allow to undercut specific rivals such as the advertised sellers, or a designated seller (which can be identified by her seller id).

The company does not disclose any information about the nature of the AI-driven repricing algorithms. However, we were able to select among various options such as AI-driven repricers designed to maximize sales or profits. The results of the present article are based on the AI-algorithm to maximize profits.

The repricer is connected to the seller account through a designated API. In practice, the seller can connect her account to the repricing solution by following a series of easy-to-implement steps. Once connected, the seller can design own mechanical repricing rules or select prespecified mechanical or algorithmic repricing rules. Different rules can be applied to different products.

One noteworthy feature is that *all* repricing rules require the specification of a minimum and maximum price. When the minimum price is reached, the seller has to specify which action the repricer should take, which could be to either stay at the minimum price, or to revert to the maximum price. Throughout the experiment, we instruct the algorithm to revert to the maximum price in case the minimum is reached. The significance of this choice will become clear later.

Own Repricing Solution

We hired a software developer to create the architecture to interact with the seller accounts using the same API as the one used by commercial repricing companies. The software allows us to implement algorithms of arbitrary complexity using near real-time data from the online marketplace. We use the Python programming language to program our own algorithms. The software solution also allows us to closely monitor the market: We obtain a full picture of all prices and the identity of the advertised seller at one minute intervals. To the best of our knowledge, this granularity of data is unprecedented in academic research examining online marketplaces.

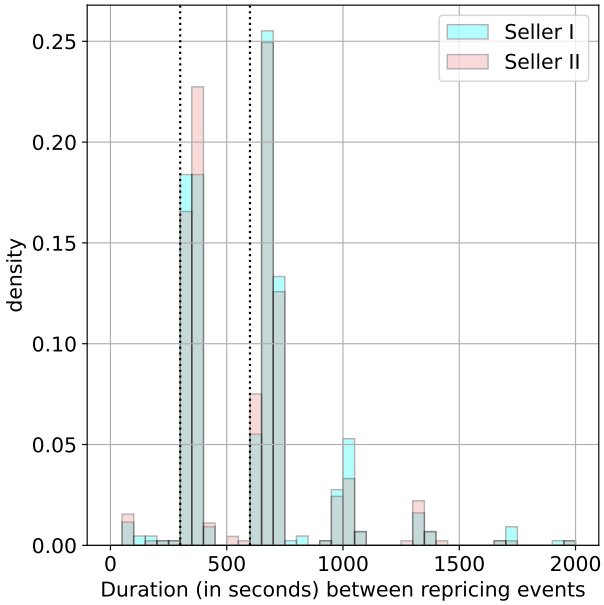


Figure 2: Distribution of Time-Intervals Between Price Events

Notes: The two dotted vertical lines mark the five and ten minutes time intervals, respectively. The data were obtained from one of our experiments.

Latency in Price-Setting

Two important aspects for competition are price-stickiness and whether rivals implement actions simultaneously or sequentially. Our data reveal that the typical time-interval between any two price changes is either five or ten minutes. Hence, once a new price is set, the seller is committed to this price for a given amount of time. Figure 2 shows the distribution of the time interval between price changes for both sellers. Our data also reveal that the prices of all sellers for a given product are typically updated simultaneously.

Price-stickiness is important because it implies that a seller who only reacts to price changes will need five minutes to implement a new price. This provides an advantage to first-movers. For example, player A who sets a new price p_t^a and who knows that rival B only passively reacts by undercutting the new price by an amount x such that $p_{t+1}^b = p_t^a - x$ can anticipate the rival move and instruct that $p_{t+1}^a = p_t^a - (x+\epsilon)$. By repeating this strategy $\forall t$, player a can ensure that she always has the cheapest offer and, thus, is the advertised seller. The minimum increment for price changes is one cent, i.e. we have that $\epsilon = 0.01$.

3 Experimental Design

In this Section, we describe our experimental approach to test the sophistication of the commercial repricing software. We start by introducing the overall design which relies on a benchmark strategy. Subsequently, we will discuss the myopic best response to the benchmark strategy and provide a detailed description of our own artificial intelligence algorithm.

Benchmark Strategy and Overall Design

The goal of the project is to assess the sophistication of commercial repricing software. Naturally, this assessment has to be done taking into consideration the environment within which the commercial repricing software will be deployed. One key feature of our online marketplace is that, by prominently displaying the cheapest seller no matter how small the price advantage, it implements an almost ideal version of Bertrand competition.

Assuming, for simplicity, that i and j denote the two cheapest sellers, we have

$$\Pi_i(p_i, p_j) = \begin{cases} p_i - c & \text{if } p_i < p_j \\ \frac{1}{2}(p_i - c) & \text{if } p_i = p_j \\ 0 & \text{if } p_i > p_j \end{cases}, \tag{1}$$

where we assume that the position as the advertised seller is equally shared in the scenario of price parity (Figure 1 corroborates this assumption).[7] Note that Equation (1) assumes symmetric marginal costs; we will maintain this assumption throughout the following exposition.

The Bertrand environment provides a strong incentive to undercut rivals. Thus, it appears natural that a seller would instruct a mechanical repricer to undercut the cheapest rival price by one cent while simultaneously setting a price floor which captures the marginal costs. In other words, it appears natural that a seller would instruct a repricer to implement the Bertrand strategy. Thus, our benchmark strategy is defined as the Bertrand reaction function (p_{-i} denotes the set of rival prices):

$$BR_i(p_{-i}, c) = max\{min\{p_{-i}\} \ - \epsilon, c\}. \tag{2}$$

[7] We tested this assumption more rigorously by analyzing how the advertised seller position is split between sellers in an experiment in which both sellers set the same price for a long period of time: The advertised seller position is nearly perfectly split between both sellers, as is shown in Figure 9.

The general approach of our design is to establish a best response to the Bertrand strategy and to assess the performance of the commercial repricer and our own repricer relative to this best response. This approach provides a metric for the performance of repricing software, which is derived from a sensible myopic benchmark strategy.

Myopic Best Response to Benchmark Strategy

The myopic best response strategy to the Bertrand reaction function exploits the price-stickiness discussed in Section 2 and the ensuing advantage for first-movers. The strategy we will introduce can be thought of as the best response of a strategic but myopic seller, who wants to ensure herself a large share of the advertised seller position, even in the event when she is confronted with a seller implementing a Bertrand strategy.

We call the best response to the Bertrand strategy the *relentless cycling strategy*. The relentless cycling strategy consists of continuously lowering the price of the seller implementing the strategy by $\epsilon + 0.01$. By continuously lowering her own price, the seller exploits the price latency and the fact that the Bertrand strategy is only reactive: In period t, the Bertrand strategy will implement a price change such that $p_{t+1} = p_t - \epsilon$, while the relentless cycling strategy will implement a price change such that $p_{t+1} = p_t - (\epsilon + 0.01)$. This way, the player with the relentless cycling strategy will have a by one cent lower price and obtain the advertised seller position.

Eventually, the relentless cycling strategy will reach the minimum price, which will lead to a price-reset to the allowable maximum price. As a result, the seller will loose the advertised seller position. However, the Bertrand strategy will instruct to follow the price increase, which will re-initiate the downward dynamic previously described. The relentless cycling strategy ensures full control over the advertised seller position in all periods, except when the cycle is re-initiated.

The reset to the maximum price follows the template of commercial repricing companies that typical offer either the option to stay at the minimum price or to reset to the maximum price. Note that resetting the maximum price is necessary to exploit the price-stickiness on the downward trajectory, which is the key feature of the relentless cycling strategy.

While the relentless cycling strategy has a forward-looking component because it anticipates the next move of the Bertrand strategy, it is still to be understood as a myopic strategy in the sense that it does not compute and solve intertemporal trade-offs. As we will discuss in Section 5, it is typically not optimal to wait until reaching the minimum price before reverting to the maximum price. However, this

requires to solve the trade-off between the benefits of a higher average price and the immediate loss associated with loosing the advertised seller position when resetting.

We conclude the discussion of the relentless cycling strategy by noting that the combination of Bertrand best response and relentless cycling does not constitute a Nash-Equilibrium. From the perspective of the Bertrand repricer, the best response to the relentless cycling strategy would be to stick to the minimal price once it is reached and not to follow price increases. Not following price increases is a setting a seller might choose. Note, however, that such a strategy requires anticipating future behavior of the relentless cycling strategy and foregoing an immediate best response (undercutting a higher price).

The EXP3 Algorithm

For our own repricer, we employ the EXP3 algorithm, which belongs to the class of no-regret reinforcement learning algorithms. No-regret algorithms minimize the expected loss of a sequence of actions relative to an ex-post optimal action, i.e. they minimize the "regret" from not knowing the ex-post optimal action ex-ante.

More precisely, denote by $u_t(a_j)$ the payoff from taking action $j \in 1, \cdots, K$ in period $t \in 1, \cdots, T$. The objective of the EXP3 algorithm is to minimize the regret function

$$\frac{1}{T} \sum_{t=1}^{T} \Big(u_t(a_j) - u_t(a^\star) \Big), \tag{3}$$

where $a^\star$ denotes the ex-post optimal *single* action over all rounds T. In each period t, the algorithm only observes the payoff of the action selected during that period. EXP3 is a popular online-learning algorithm because of its low informational requirements and ease of implementation.

Algorithm 1 describes the implementation rules for the EXP3 algorithm. Initially, equal weights w are assigned to each action. Over the periods, the weights are updated such that actions that yielded high rewards are chosen with higher probability. The parameter $\gamma \in [0, 1]$ is a tuning parameter governing the exploration-exploitation trade-off, which is common to all reinforcement learning algorithms. The higher the parameter γ, the more likely it is that the algorithm will choose an action at random, favoring exploration over exploitation. Throughout the analysis, we set $\gamma = 0.1$, which is a typical choice in the computer science literature.

In Equation (3), the payoffs for the same action are allowed to change over time. This implies that the weights of the algorithm might not change monotonically, which, in turn, implies that the algorithm might not converge towards playing one

single action over time. However, it can be shown that the regret from Equation (3) is bounded by $\sqrt{Klog(K)E_t(u_t(a^\star))}$ (Auer et al., 2002).

Algorithm 1: EXP3 Algorithm

Initialization: Set $w_{t=0}(a_j) = 1 \quad \forall j$

Algorithm: For $t = 1 \cdots T$:

1. $\forall j$, set $p_t(a_j) = (1-\gamma)\frac{w_t(a_j)}{\sum_{j=1}^{K} w_t(a_j)} + \frac{\gamma}{K}$

2. Draw one action from the probability distribution p_t, denote the selected action by a'

3. Set $w_{t+1}(a') = w_t(a') \exp\left(\frac{\gamma u(a')}{K p_t(a')}\right)$

Notes: In each period t, only the weight $w(a')$ is updated. γ is a tuning parameter that governs the exploration-exploitation trade-off.

The bound derived by (Auer et al., 2002) suggests that the number of actions K has a negative effect on the performance of the EXP3 algorithm. To keep K small, we opt to use deviations from the current market price as our action set $\mathcal{A}$.

More precisely, we define $\mathcal{A} = \{-0.04, -0.02, 0.00, 0.02, 0.04\}$, and the price implemented by the EXP3 algorithm if action $a' \in \mathcal{A}$ is chosen is $p_{t+1} = min\{\mathbf{p_t}\} + a'$, where $\mathbf{p_t}$ denotes the price vector of all sellers in period t. Thus, the EXP3 algorithm increases or decreases the minimum price by an increment a'. Our reward function is given by $u(a') = (min\{\mathbf{p_t}\} + a') \times \mathbb{I}\{AS_{t+1} = True\}$, where the indicator function $I\{AS_{t+1} = True\}$ takes the value one if the seller controlled by the EXP3 algorithm has the advertised seller position in period $t+1$.

The action space we select ensures that the EXP3 algorithm will, by design, implement prices close to the competitive price. Under the specified reward function, the EXP3 algorithm seeks to maximize the price-weighted share of times it acquires the advertised seller position.

Finally, we need to specify an ad-hoc rule for the instance that the EXP3 algorithm selects a price weakly below the marginal costs. We instruct the algorithm to revert to the maximum price if the minimum price (i.e. the marginal costs) are reached. Note that, due to the definition of the action space, the EXP3 algorithm cannot itself decide to revert to the maximum price. The ad-hoc rule of reverting to the maximum price is also implemented for the relentless cycling strategy, ensuring a level playing field between both algorithms.

Additional Remarks

Our goal is to implement a realistic repricing software that achieves good performance in real markets, which are characterized by rivals who might change their strategy. In this respect, one concern is that the relentless cycling strategy and the EXP3 algorithm might be overspecialized to the scenario of a rival implementing a Bertrand strategy.

For instance, the relentless cycling strategy involves lowering the own price preemptively to prevent the reactive Bertrand-player to successfully undercut. Note that this strategy would be non-optimal if the Bertrand-player would decide to give up and implement a higher static price. In this case it would be optimal for the relentless cycling strategy to stop cycling and simply undercut the new higher price.

While the relentless cycling strategy that we implement does not account for this scenario, it would be easily feasible to modify the code to monitor the activity of the rival (for instance by counting the number of price changes in a given time-interval) and to instruct the algorithm to switch modes in case the rival becomes inactive. Because this case is trivial both economically and technically, we do not explicitly consider it in our experiments.

A similar reasoning holds with respect to the EXP3 algorithm: If the rival becomes inactive, the EXP3 can be instructed to undercut the rival. When the rival resumes activity, the EXP3 algorithm can simply be reactivated and continue using the weights it had learned prior to the period of inactivity.

4 Results

In this Section, we present and discuss the results of our experiments. We start with the scenario in which a Bertrand strategy competes against a relentless cycling strategy. The results of this first experiment will establish the performance benchmark for the two subsequent experiments in which the commercial repricing software and our EXP3 algorithm will compete against the Bertrand strategy. Finally, we will show the results of the experiment in which our EXP3 algorithm competes directly against the commercial repricing software.

All the experiments where conducted with a maximum allowable price of 2.65 and a minimum allowable price of 2.0. The minimum allowable price can be thought of as representing the marginal costs. The prices where chosen with the goal to ensure that the two algorithms competing against each other are below the minimum price of all other sellers active in the market. We opted for this approach in an attempt to make sure that the prices of other sellers do not interfere with our experiments.

Each experiment lasted for approximately one-and-a-half days.

Bertrand Strategy Against Relentless Cycling Strategy

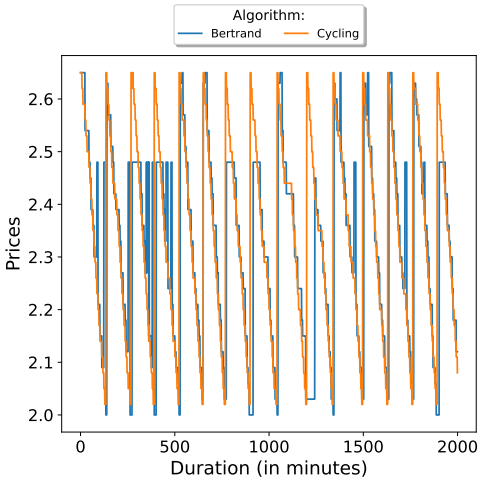
Figure 3a shows the prices set by the Bertrand and the relentless cycling strategy over the course of the entire experiment. Note that the Bertrand strategy occasionally does reset the price cycle to a price just below 2.65. This is due to a professional seller who unexpectedly decreased her price which led to interferences with our experiment. Figure 3b shows one selected cycle of the experiment: It illustrates that the relentless cycling strategy manages to prevent the Bertrand strategy from successfully undercutting most of the times.

Figure 3c shows the share of the advertised seller position that the relentless cycling strategy obtains. The share is obtained over rolling windows of 250 minutes, which corresponds to roughly two full cycles. The share of the advertised seller position is our success criterion. When computing the share, we remove instances in which the Bertrand strategy did not reset to a price above 2.5 to avoid that our results are affected by the prices of the professional seller.

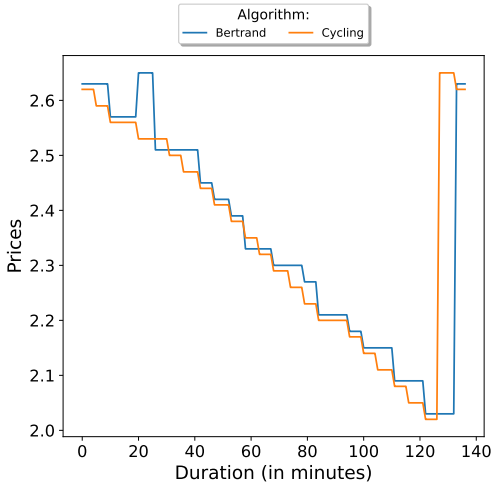
The share of the advertised seller position of the relentless cycling strategy oscillates around 86.5%. The red dotted horizontal line indicates the share of the advertised seller positions that the relentless cycling strategy should theoretically attain. In theory, the relentless cycling strategy should always charge the lowest price except when it initiates a price increase to reset the cycle. Taking into consideration that a cycle takes 23 price updates to complete, we would therefore expect that the theoretical share of the relentless recycling strategy is approximately equal to $1 - 1/23 = 95.5\%$.

The main reason for the discrepancy between the observed and the theoretical value is that the pricing API used to control prices exhibits unexpected behavior. For instance, the time between consecutive pricing events varies and price updates might occur asynchronously, which explains why the Bertrand strategy is occasionally able to successfully undercut the relentless cycling strategy (as is shown Figure 3b). This results in the Bertrand strategy being able to claim a larger share of the advertised seller position than expected from theoretical considerations.

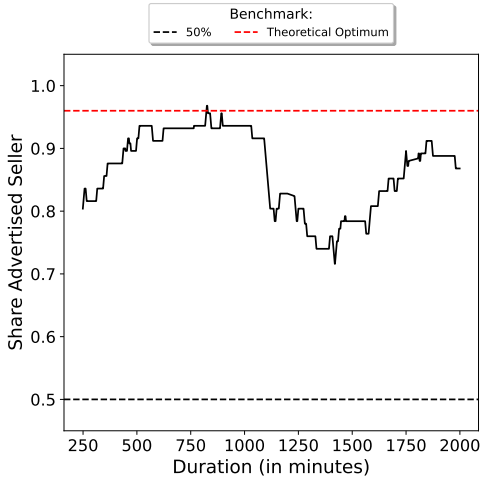
The discrepancy illustrates how real world technical issues hinder a perfect implementation of the intended strategies. As a result, hypothetical benchmarks may never be reached. Therefore, instead of the theoretically optimal share, we will use the average share of the advertised seller position observed for the relentless cycling strategy as the benchmark for the other experiments.



(a) Prices Paths Entire Experiment



(b) Price Paths for Selected Cycle



(c) Share of Advertised Seller Position for Relentless Cycling Strategy

Figure 3: Bertrand Best-Response vs. Relentless Cycling Strategy

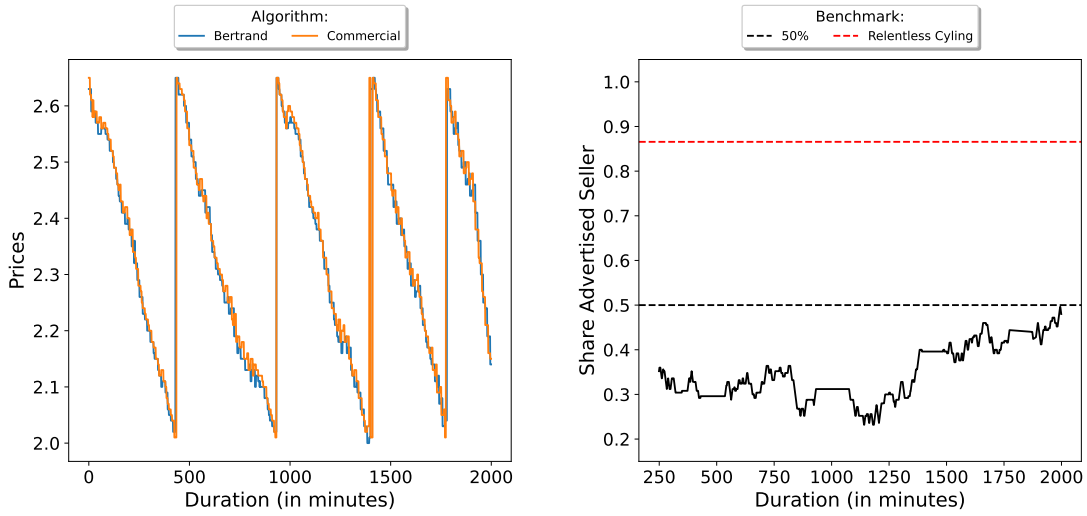
Notes: In the upper-left panel, the Bertrand strategy occasionally resets the price cycle to a price of 2.48. This is due to another seller setting a price of 2.5, prompting the Bertrand best-response strategy to ignore the price of the relentless cycler.

Bertrand Strategy Against Commercial Repricing Software

Figure 4a shows the prices set by the commercial repricer and the Betrand strategy. Figure 4b shows the share of the advertised seller position obtained by the commercial repricing software. The share obtained by the commercial repricer is consistently below 50%. This suggests that the Bertrand strategy is sufficient to outperform the AI of the commercial repricing company.

The prices observed in Figure 4a show that the commercial repricer is occasionally holding constant, or increasing the prices. This also explains why the cycle length observed in the experiment between the commercial repricer and the Bertrand strategy is significantly longer (approximately 500 minutes) than in the previous setting.

Theoretically, one advantage of extending the cycle length is that it decreases the costs associated with resetting the cycle to the maximum price, as this is associated with loosing the advertised seller position. Note, however, that this strategy would only make sense if the rival would cooperate in extending the cycle length by not always undercutting. As is evident from our experiment, extending the cycle length against a Bertrand strategy is clearly sub-optimal.



(a) Price Paths Entire Experiment

(b) Share Advertised Seller of Commercial

Figure 4: Commercial Repricing Software vs. Bertrand Strategy

While the performance of the commercial repricing software appears sub-optimal from the perspective of a myopic seller, the same cannot be said about a forward-looking seller. For instance, the commercial repricing software might try to deplete

the stock of the rival to obtain a monopoly position in the allowable price range that was specified by the seller (in our case from 2.65 to 2).

We note that there is no guarantee that the the rival will ever allow the stock to deplete, in which case the observed scenario is clearly sub-optimal. Additionally, if the strategy is indeed aimed at depleting the rival's stock, the observed price cycles implemented by the commercial software appear unnecessarily complex: The same could be achieved by allowing the Bertrand strategy to undercut a low static price.

Finally, it might be the case that the commercial repricer did not have sufficient time to learn and adapt to the Bertrand strategy. We cannot rule out this possibility. However, such concerns can be easily addressed by running longer experiments. Additionally, as we will show next, it is possible for an artificial intelligence algorithm to successfully adapt to the Bertrand Strategy within the time-span of one-and-a-half days.

EXP3 Against Bertrand Strategy and Commercial Repricing Software

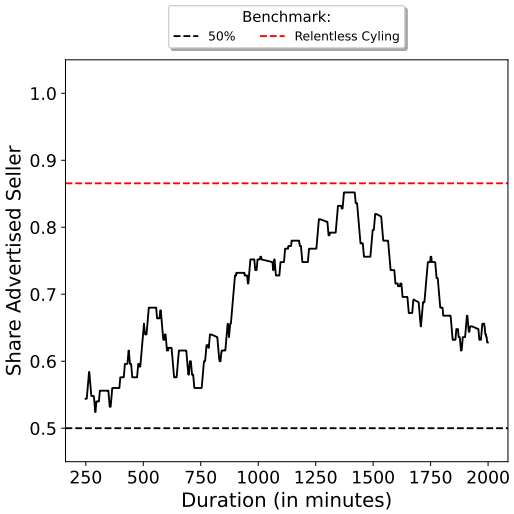
Figure 5a shows the performance the EXP3 algorithm against the Bertrand strategy. The share of the advertised seller position captured by the EXP3 algorithm is consistently above 50% and increasing over time.

Figure 5a shows the evolution of the probabilities with which the EXP3 algorithm chooses a certain action. In light of the discussion in Section 3, the optimal action is to choose to decrease the lowest price by four cents. Clearly, the algorithm learns to predominantly play the optimal action. Expectedly, the probabilities to choose an action that would weakly increase the currently lowest price converge to zero.

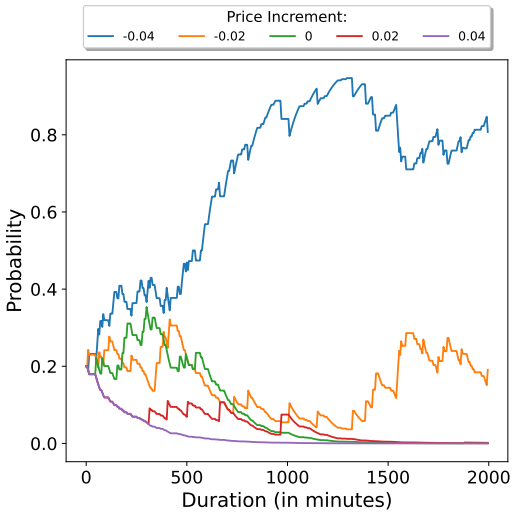
Decreasing the price by only two cents does not guarantee the advertised seller position because it will lead to a situation in which the Bertrand and the EXP3 player charge the same price. Note, however, that players charging the same price will be randomly assigned the advertised seller position. Therefore, the action of decreasing the lowest price by only two cents will yield occasional positive payoffs. This is the reason why the probability associated with choosing a price reduction of two cents does not decay to zero.

Figure 6a shows the performance of the EXP3 algorithm against the commercial repricing software. According to our success measure, the EXP3 algorithm substantially outperforms the commercial repricing software: The share of the advertised seller position shows a clear upward trend and is consistently above 50%.

Figure 6b shows the evolution of the probabilities with which the EXP3 algorithm chooses a certain action when confronted with the commercial repricer. Compared to the scenario in which the EXP3 algorithm competes against a Bertrand Strategy,

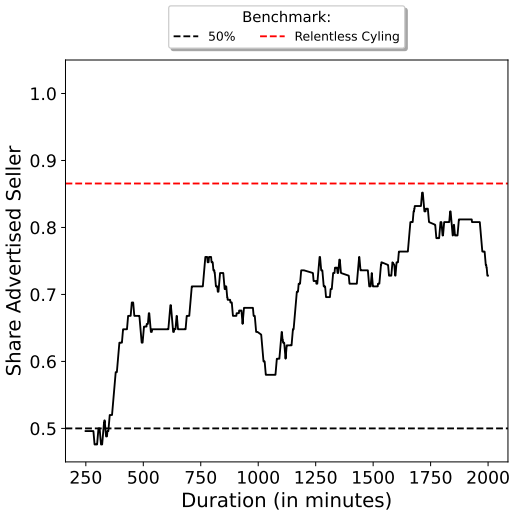


(a) Share Advertised Seller of EXP3

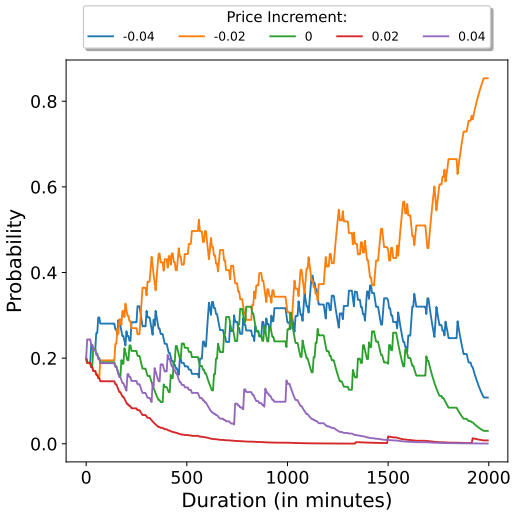


(b) Evolution of EXP3 Probabilities

Figure 5: EXP3 vs. Bertrand



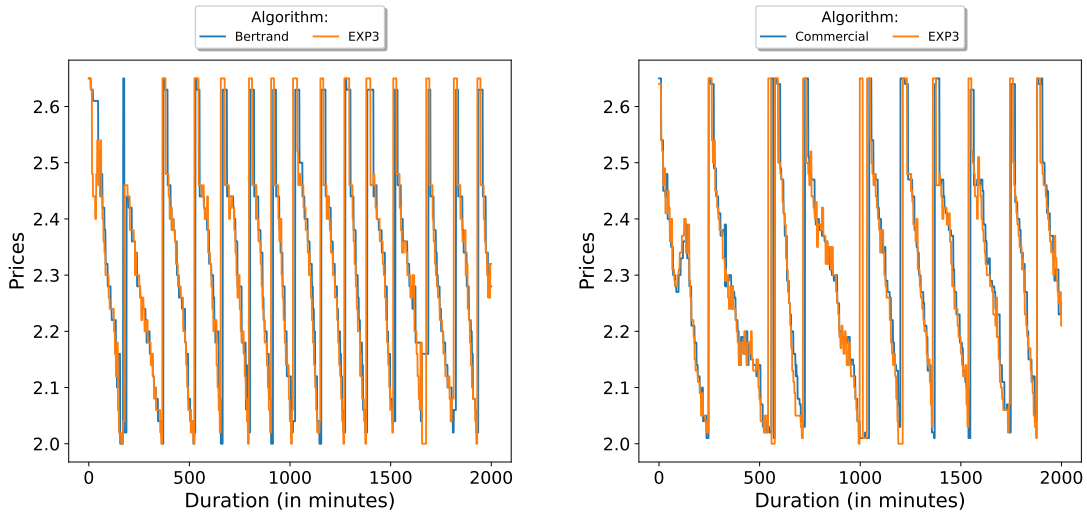
(a) Share Advertised Seller of EXP3



(b) Evolution of EXP3 Probabilities

Figure 6: EXP3 vs. Commercial

there does not appear to be one single action that the EXP3 algorithm predominantly chooses – although there appears to be a trend in favor of a price reduction of two cents towards the end of the experiment.



(a) Price Paths EXP3 vs Bertrand

(b) Price Paths EXP3 vs Commercial

Figure 7: Price Paths EXP3 Experiments

Over much of the experiment, the weights are consistent with a mixed strategy in which actions that lead to weak price reduction are chosen with approximately equal probabilities. Note that this "uniform" mixed strategy play does not appear to impact the performance of the EXP3 algorithm significantly compared to the later stages in which a single preferred action emerges. This appears consistent with the pricing pattern observed in Figure 4a for which we noted that the commercial repricer occasionally raises prices. In light of this, the EXP3 strategy of mixing across actions leading to a weak price decrease appears rational.

5 Forward-Looking Algorithms

So far, our research design has been limited to developing a protocol to test the sophistication of repricing technology assuming myopic players. Using one selected commercial repricing software and our own EXP3 algorithm, we demonstrated the applicability of our protocol. Our results indicated that the selected commercial

repricing software is outperformed by simple repricing rules as well as our own software.

Our findings might be driven by a mismatch between the potentially forward-looking objective function of the commercial software and the myopic objective function our protocol has been developed for. While we argue in Section 4 that the more plausible explanation for our results is the lack of sophistication of the commercial repricer, a more rigorous approach is needed. Particularly in view of the fact that other commercial repricing technologies, which we want to test in future iterations of this work, might be more sophisticated.

The ability to learn strategies that make the myopically profitable action of undermining the collusive agreement unattractive is a key aspect of tacit collusion. Solving intertemporal trade-offs is therefore a prerequisite that algorithms must fulfill in order to be able to collude. In the remainder of this Section, we briefly discuss the main issue associated with forward-looking algorithms that learn in real time. We then review the class of Q-learning algorithms, and, finally, present results that highlight the possibility to efficiently learn intertemporal trade-offs.

Speed of Learning

Calvano et al. (2020) show that simple Q-learning algorithms are able to learn strategies that punish a deviation from the collusive agreement for a finite number of periods. One caveat associated with their findings is that the algorithms undergo extensive learning. The main results in Calvano et al. (2020) are based on algorithms trained over several millions of periods. While such extensive learning might be acceptable in markets characterized by a very high-frequency of price events (such as financial markets), it appears prohibitive in our setting.

A fast speed of learning appears crucial for repricing companies. Sellers experiencing extensive learning periods with sub-optimal behavior will likely discontinue the use of the repricing service. While offline learning, where the algorithm is trained on historical data before being deployed in the real environment, might be a partial remedy, such algorithms will typically lack the ability to optimally adjust to the idiosyncrasies of the market environment in which they will be deployed.

Digression: Q-learning Algorithms

Q-learning algorithms are designed to solve Markov decision problems. Given a state space S and an action space A, they find the optimal policy, i.e. a mapping $S \to A$ that maximizes the sum of discounted payoffs $\sum_{t=0}^{t=T} \delta^t u(a_t, s_t)$. Markov decision problems are characterized by the property that the probability to reach a certain

state s' in period $t+1$ only depends on the state and action taken in period t, s and a, i.e. the (transition) probability to reach state s' given s and a is given by $prob(s'|s,a)$.

To apply the Q-learning algorithm, one starts by initializing a Q-matrix with dimensions $S \times A$. Each cell of the Q-matrix contains initial guesses for the Q-values, denoted by $Q(s,a)$. The Q-values describe the continuation value of choosing action $a \in A$ when in state $s \in S$. Once the Q-learning algorithm has been applied, the optimal policy can be read-off from the Q-matrix by finding $\text{argmax}_{a \in A} Q(s,a) \forall s$, i.e. by determining for each state s which action maximizes the continuation value.

Algorithm 2: ϵ-GREEDY Q-LEARNING ALGORITHM

Initialization: Randomly set $Q(s,a), \forall (a,s)$, and initialize a state s

Algorithm: For $t = 1 \cdots T$, repeat:

1. With prob. ϵ select $a^\star \in A$ uniformly at random (exploration); with prob. $1-\epsilon$ select $a^\star = \text{argmax}_{a \in A} Q_t(s,a)$ (greedy action)
2. Given $a^\star$ and s, select s' following $prob(s'|s,a^\star)$
3. Apply Equation (4) to update $Q_t(s,a^\star)$

One popular way to learn the optimal Q-matrix is the ϵ-greedy Q-learning algorithm: After initiating the Q-matrix and selecting an initial state s, learning proceeds in rounds t. In each round, the algorithm either selects one action $a \in A$ uniformly at random with probability ϵ, or, with probability $1-\epsilon$, it selects the greedy action, $\text{argmax}_{a \in A} Q_t(s,a)$. Given the selected action, $a^\star$ and the current state, s, the next state is reached according to $prob(s'|s,a^\star)$. In each round, the Q-values are updated following:

$$Q_{t+1}(s,a^\star) = (1-\alpha) Q_t(s,a^\star) + \alpha \big[u(s,a^\star) + \delta max_{a \in A} Q_t(s',a) \big]. \tag{4}$$

α denotes the learning rate, which is selected by the researcher. It has been shown that, under fairly general conditions, ϵ-greedy Q-learning converges to the Q-matrix that solves the Markov decision problem (Watkins and Dayan, 1992). Algorithm 2 provides the pseudo-code to implement ϵ-greedy Q-learning.

Optimistic Q-learning

Optimistic Q-learning is a variant of Q-learning that follows Algorithm 2 with two important distinctions. Firstly, the Q-values are not randomly initiated but are

initiated "optimistically", using the highest possible continuation value that is theoretically reachable in the environment. Secondly, optimistic Q-learning is performed without any exploration (i.e. $\epsilon = 0, \forall t$), i.e. chooses the greedy action at each step of the learning process. Optimistic Q-learning is popular as it is considered *sample efficient* and, hence, offers an avenue to circumvent issues related to extensive learning (Even-Dar and Mansour, 2001; Neustroev and de Weerdt, 2020).

Applying Optimistic Q-learning to Augment The Relentless Cycling Strategy

We now demonstrate, using a simulation approach, that an optimistic Q-learning algorithm can be used to find the optimal dynamic strategy against a myopic Bertrand player. Crucially, the learning process does only impose minimal costs on the seller. As discussed in Section 3, the relentless cycling strategy can only be considered myopically optimal. The reason is that the relentless cycling strategy only resets the price-cycle when it reaches the allowable minimum price.

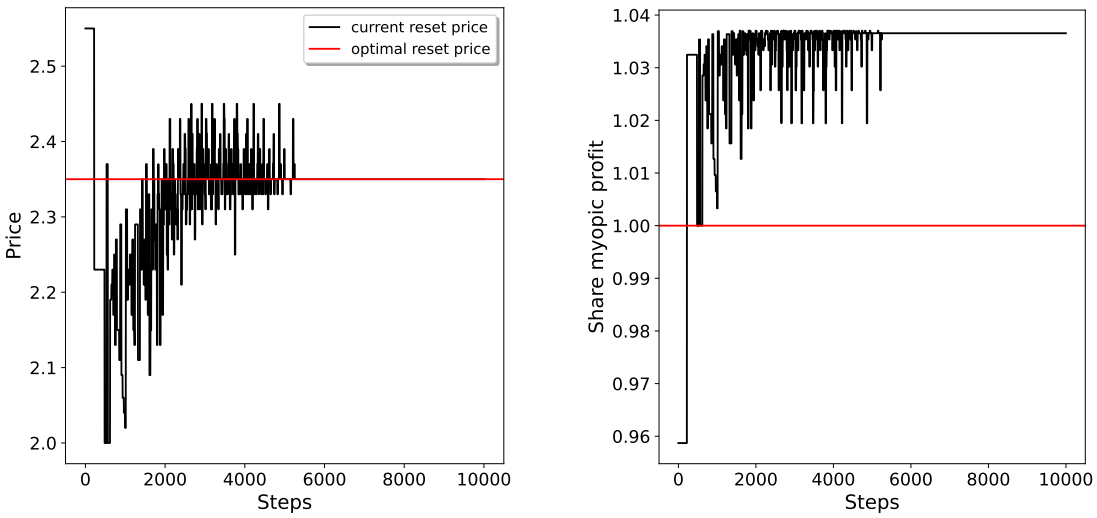
By contrast, a forward-looking seller will choose the optimal reset-price. This optimal reset-price will solve the trade-off between the costs of resetting the price-cylce, i.e. loosing the advertised seller position, and the benefits of achieving a higher average price by resetting before reaching the minimum price. For example, if the consider the scenario of a price range between 2.65 and 2 and a Bertrand rival, it can be shown that the optimal reset price for a relentless cycling strategy is 2.35.[8]

For the following simulation, we let the state space S of our Q-learning algorithm contain all the 65 possible price points in interval $[2, 2.65]$. The algorithm is allowed to choose three action: (i) no price change, (ii) reset the price-cycle, and (iii) reduce the price by $\epsilon + 0.01$, where $\epsilon = 0.01$ is the amount by which the Bertrand strategy decides to undercut. Note that preventing the Bertrand strategy from successfully undercutting is embedded in the algorithm by design. Thus, whenever the algorithm decreases its price, it automatically exploits the main idea of the relentless cycling strategy.

For our simulation, we set $\delta = 0.999$ and $\alpha = 0.1$. To implement the optimistic Q-learner, we initiate all the Q-values at the continuation value that results from resetting the price at the optimum value, i.e. 2.35.

Figure 8 shows one representative run of the simulation. Figure 8a shows the reset price that the algorithm is currently implementing. Figure 8b shows the ratio between the discounted profit that would result from keeping the current reset price and the discounted profit that would be obtained if the algorithm would use the

[8] We use numerical optimization techniques to find the optimal reset price.



(a) Evolution Reset Price

(b) Profit Compared to Myopic Strategy

Figure 8: Simulation Results Optimistic Q-learning

minimum price as a reset price. Note that using the minimum price as a reset price is the myopic optimal strategy. Therefore, the ratio indicates how much the seller would loose or gain from permanently implementing sub-optimal reset prices instead of simply playing the myopic strategy.

While Figure 8a reveals that the algorithm needs 5000 periods to settle on the optimal reset price (which corresponds to roughly 17 days in reality), Figure 8a shows that the typical learning process comes at almost no costs when taking the myopic optimal strategy as the benchmark. In fact, even in the learning stages, when the optimal reset price is not yet achieved, the optimistic Q-learning algorithm achieves a higher profit than the myopic benchmark. Thus, optimistic Q-learning, when combined with an ad-hoc implementation of the optimal myopic strategy, allows to learn dynamically optimal strategies against Bertrand strategies at no costs.

6 Conclusion

We develop a protocol to test the sophistication of commercial repricing software assuming myopic sellers. To this end, we create two seller accounts on an online platform and develop our own repricing software, which allows us to implement

arbitrarily complex algorithms.

After establishing a performance benchmark based on implementing the best response against a seller using a Bertrand strategy, we let the commercial and our own software compete against this same Bertrand strategy. Our software is based on EXP3, which is a workhorse reinforcement learning algorithm for online learning

We find that our software comes closer to the performance benchmark than the commercial software, which suggests that a simple EXP3 algorithm is superior to the commercial software, at least in the myopic setting. This finding is further confirmed by an experiment in which we let the EXP3 and the commercial software compete directly: our own algorithm significantly outperforms the commercial software.

We discuss the limitations of our current findings, which could capture a failure of using the right benchmark. For instance, it could be that the commercial repricing software has a long-term objective that our protocol does not capture. We provide a proof-of-concept demonstrating the ability of algorithms to learn long-term startegies efficiently. This lays the groundwork for a protocol testing the sophistication of commercial repricing software assuming forward-looking sellers.

References

Assad, S., Clark, R., Ershov, D., and Xu, L. (2020). Algorithmic pricing and competition: Empirical evidence from the german retail gasoline market. *CESifo Working Paper*.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002). The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

Calvano, E., Calzolari, G., Denicolo, V., and Pastorello, S. (2020). Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review*, 110(10):3267–97.

Chaboud, A. P., Chiquoine, B., Hjalmarsson, E., and Vega, C. (2014). Rise of the machines: Algorithmic trading in the foreign exchange market. *The Journal of Finance*, 69(5):2045–2084.

CMA (2021). Algorithms: How they can reduce competition and harm consumers. *CMA Whitepaper*.

Cooper, W. L., Homem-de Mello, T., and Kleywegt, A. J. (2015). Learning and pricing with models that do not explicitly incorporate competition. *Operations research*, 63(1):86–103.

Even-Dar, E. and Mansour, Y. (2001). Convergence of optimistic and incremental q-learning. *Advances in neural information processing systems*, 14.

Hendershott, T., Jones, C. M., and Menkveld, A. J. (2011). Does algorithmic trading improve liquidity? *The Journal of finance*, 66(1):1–33.

Neustroev, G. and de Weerdt, M. M. (2020). Generalized optimistic q-learning with provable efficiency. In *AAMAS*, pages 913–921.

OECD (2017). Algorithms and collusion: Competition policy in the digital age. *OECD Whitepaper*.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.

7 List of Additional Figures

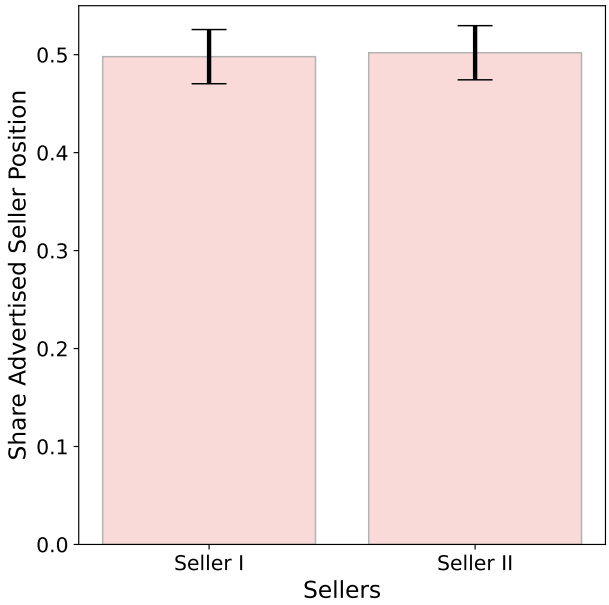


Figure 9: Share of Advertised Seller Position

Notes: The graph shows the share of the advertised seller position when both sellers set the same cheapest price. The experiment lasted for approximately two days. The error bars show the 95% confidence intervals.