Algorithms, Artificial Intelligence and Simple Rule-Based Pricing

Qiaochu Wang Yan Huang Param Vir Singh Kannan Srinivasan

Automated pricing strategies in e-commerce can be broadly categorized into two forms – simple rule-based such as undercutting the lowest price, and more sophisticated artificial intelligence (AI) powered algorithms, such as reinforcement learning (RL) algorithms. Although simple rule-based pricing remains the most widely used strategy, a few retailers have started adopting pricing algorithms powered by AI. RL algorithms are particularly appealing for pricing due to their abilities to autonomously learn an optimal policy and adapt to changes in competitors' pricing strategies and market environment. Despite the common belief that RL algorithms hold a significant advantage over rule-based strategies, our extensive pricing experiments demonstrate that when competing against RL pricing algorithms, simple rule-based algorithms may result in higher prices and benefit all sellers, compared to scenarios where multiple RL algorithms compete against each other.

To validate our findings, we estimate a non-sequential search structural demand model using individual-level data from a large e-commerce platform and conduct counterfactual simulations. The results show that in a real-world demand environment, simple rule-based algorithms outperform RL algorithms when facing other RL competitors. Our research sheds new light on the effectiveness of automated pricing algorithms and their interactions in competitive markets, and provides practical insights for retailers in selecting the appropriate pricing strategies.

Keywords: Algorithmic pricing, competition, rule-based pricing, reinforcement learning.

1 Introduction

Automated pricing algorithms exhibit varying degrees of sophistication, with some being simple rule-based systems that operate according to predetermined human agent-defined protocols. Others are more advanced, artificial intelligence (AI) powered, featuring self-learning capabilities that enable them to autonomously identify the optimal price points that maximize long-term profits through iterative pricing experiments. In this paper, we investigate the impact of the use of automated pricing algorithms on market outcomes. The primary objective of this study is to investigate the effectiveness of simple rule-based pricing algorithms versus AI powered (specifically, Reinforcement learning (RL) algorithms) in a competitive pricing environment.

Most major e-commerce marketplaces, such as Amazon, eBay, Shopify, Walmart, and Google Shopping, offer simple rule-based pricing algorithms as free pricing tools to 3rd-party sellers. Typically, these algorithms permit sellers to establish prices that are below, equal to, or above the lowest price in the pre-defined market, with a user-specified listing condition and fulfillment method.[1] If greater flexibility is required, such as matching the price of a specific competitor or alternating between different strategies based on market conditions, more sophisticated rule-based pricing solutions are available on various 3rd-party repricer services such as ChannelEngine, RepricerExpress, Aura, and Informed.co. Analyzing a sample of 1600 best selling items on Amazon, Chen et al. (2016) find that more than one third of sellers were using simple rule based pricing strategies in 2015. A 2017 E-Commerce sector inquiry found that 50% of retailers in the European Union track their online competitors' prices, and 70% of them automatically adjust their prices.

RL algorithms have recently been employed to solve various business problems, such as display advertisement measurement (Tunuguntla, 2021), sequential coupon targeting (Liu, 2022, Wang et al., 2021), ad sequencing (Rafieian, 2022), and ad recommendations (Aramayo et al., 2022). Not surprisingly, these algorithms have demonstrated significantly better performance than traditional methods. However, the evaluation of RL algorithms in competitive settings has only recently gained attention. In particular, in repeated price competitions in simulated market environments, RL algorithms have been recognized for their ability to learn sophisticated pricing strategies that soften competition and enable supra-competitive prices to be maintained in competitive markets (Calvano et al., 2020, Hansen et al., 2021, Klein, 2021, Johnson et al., 2020, Asker et al., 2022).

Empirical evidence regarding the utilization of RL algorithms for pricing by sellers operating in competitive environments is limited to only by 1st party sellers on major platforms.

[1]See `https://sellercentral.amazon.com/gp/help/external/200836360?ref=efph_200836360_cont_201186860&language=en_US`.

Researchers from the Chinese e-commerce company Alibaba found that employing reinforcement learning-based pricing algorithms resulted in significantly better profits for thousands of products, surpassing expert manual pricing (Liu et al., 2019).

The central research question of this study is whether a seller should adopt a simple rule-based pricing algorithm or an RL pricing algorithm when competing against rivals using RL pricing algorithms. The study considers various variables such as the demand model, degree of product differentiation, demand uncertainty, type of algorithm, and the number of players in the market to determine their impact on the outcomes. The findings of this study have significant managerial implications as they offer valuable insights into how sellers can respond to competitors who use RL pricing algorithms.

We adopt a three-pronged strategy to address the research questions at hand. Our first approach involves following the established economics literature that examines competition among pricing algorithms, with particular attention to the canonical Logit demand model as demonstrated in previous works such as Calvano et al. (2020), Klein (2021) and Asker et al. (2022). This approach is simple yet effective, enabling us to compare our findings with existing literature. Our second approach utilizes data from a major e-commerce platform to estimate a more sophisticated structural demand model, which takes into account realistic model parameters and consumer behavior during search and purchase. We use this estimated demand model, along with marginal costs for sellers, to assess competition between automated pricing algorithms. In our third approach, we employ theoretical analysis with simplifying assumptions to identify boundary conditions that determine when Q-learning algorithms and rule-based algorithms may dominate each other.

In the baseline case, we use one of the most commonly applied rule-based pricing algorithms – *Undercut Lowest Price (UL)* as an example of simple rule-based pricing algorithms, and use one of the most basic but popular RL algorithm – *Q-learning* algorithm as an example of RL pricing algorithm. We compare the behavior of the algorithms in two distinct scenarios: the Q-Q scenario where the focal and the competing sellers adopt the Q-leaning algorithms, and the Q-R scenario, where the focal seller uses a simple rule-based algorithm whereas the competing seller adopts a Q-learning algorithm. In each scenario, we let the algorithms interact repeatedly. We record and compare the market outcomes in three aspects: 1) equilibrium price and profit, 2) learning dynamics, and 3) deviation and punishment strategies.

Our research findings indicate that the focal seller attains higher profits in the Q-R scenario than in the Q-Q scenario. We attribute this result to the learning dynamics inherent in each scenario and the specific properties of the simple rule based strategy. In the Q-Q scenario, two Q-learning algorithms learn thorough exploration simultaneously, leading to non-stationarity in the environment, making it harder for any algorithm to learn the most profitable collusive strategy. Conversely, when one seller uses a simple rule-based algorithm, the environment stabilizes as the strategy for the simple rule based algorithm is fixed, enabling the competitor's Q-learning algorithm to learn the optimal policy easily. Furthermore, the Q-learning algorithm's optimal learning benefits the simple rule-based algorithm, which is pegged to it. The simple rule based strategy, UL, in this case represents a variant of the "Tit for Tat" strategy which is well-known to be a hard to defeat strategy in repeated games. The best response to the "Tit for Tat" pricing strategy is an always cooperate strategy which is what the Q-learning algorithm is able to learn in the Q-R case. The "Tit for Tat" feature of the UL strategy allows the algorithms in the Q-R case to sustain the highly collusive prices as equilibrium benefiting both the sellers.

Our study also contrasts the off-equilibrium strategies learned by the Q-learning algorithm in both scenarios. In the Q-Q scenario, the algorithms learn a multi-period punishment strategy. However, in the Q-R scenario, the Q-learning algorithm adopts an "always cooperate" strategy, charging the monopoly price regardless of how the simple rule-based algorithm deviates. These results remain reliable when considering variations in product differentiation levels, the number of players involved, different simple rule-based pricing algorithms, more advanced RL algorithms, and varying levels of demand uncertainty.

Compared to the logit demand model, the structural demand model that we estimate is more complex and realistic. The structural model is estimated using individual-level click and purchase data obtained from a large e-commerce platform, utilizing a Hierarchical Bayes approach. The findings demonstrate that consumers engage in a costly non-sequential search and build a consideration set of products, to learn about the product fit, before making a purchase decision, with price playing a vital role in product inclusion. The use of simple rule-based pricing strategies, such as UL which undercuts the competitors' past prices, may push Q-learning algorithms to reduce prices and intensify competition. However, the study's counterfactual results, which consider competition among five sellers that vary in terms of both horizontal and vertical differentiation, suggest that a focal seller can increase profits by approximately 9% by using a simple rule-based algorithm to soften competition and increase market prices.

We next tackle the problem theoretically by employing a system of differential equations to model the learning dynamics. We utilize a linear demand model that enables us to derive theoretical boundary conditions for the discount factor and cross-price elasticity. These conditions dictate the circumstances under which a rule-based algorithm weakly dominates a Q-learning algorithm, assuming the competitor also employs Q-learning. Our analysis reveals that the focal seller will gain more by adopting a simple rule-based algorithm if the discount factor is sufficiently high (indicating that sellers are future-oriented).

The findings from the study across multiple market environments have important managerial implications. In highly competitive markets where competitors adopt advanced pricing technologies, it is more practical and efficient for a seller to use a simpler rule-based pricing technology instead of attempting to utilize the sophisticated RL pricing algorithm. For policymakers, instead of focusing excessively on tacit collusion facilitated by numerous sophisticated pricing algorithms, more attention should be directed towards the scenario where the majority of pricing algorithms employed are simple rule-based.

The rest of the paper is organized as follows: In Section 2, we discuss relevant literature.

Section 3 expounds on the model setup and presents the results of the experiments conducted in the oligopoly logit market environment. Section 5 outlines the data utilized to create the realistic market environment, details the construction and estimation of a non-sequential search structural model and reports the outcomes of the counterfactual price experiment. Section 6 proposes a theoretical framework. Finally, in Section 7, we provide a discussion and conclusion of our findings.

2 Relevant Literature

This study adds to the growing body of literature on algorithmic pricing and collusion. The literature recognizes two main types of pricing algorithms: prediction algorithms and learning algorithms. Prediction algorithms are primarily used to forecast elements associated with the market environment, such as demand levels (Miklós-Thal and Tucker, 2019, O'Connor and Wilson, 2021). Whereas, learning algorithms can identify optimal pricing rules through price testing and adaptation (Calvano et al., 2020).

The issue of how and why pricing algorithms might increase market prices is a prominent question in this literature. Several potential channels have been discussed. The ability of algorithms to predict future demand levels has been shown to affect the incentives to deviate to lower prices during period of high predicted demand and discourage collusion (Miklós-Thal and Tucker, 2019, O'Connor and Wilson, 2021). In contrast, a number of studies that focus on learning algorithms have demonstrated the ability of self-learning intelligent algorithms (i.e. RL) to learn to charge supra-competitive prices without any coordination or communication, a phenomenon known as tacit collusion. Proposed explanations for this phenomenon include the generation of prices that are correlated with those of other algorithms, which can lead to an overestimation of the algorithm's price sensitivity (Hansen et al., 2021). Intelligent algorithms with memory are shown to learn punishment strategies to thwart deviation and as a result sustain supra-competitive prices in a repeated pricing game (Calvano et al., 2020). Our study focuses on the reasons behind the ability of algorithms to charge supra-competitive prices and the extent of tacit collusion they can achieve. Our research makes two notable contributions to the existing literature. Firstly, we uncover a previously overlooked factor that could result in increased market prices. Our findings reveal that rule-based algorithms, when compared to intelligent reinforcement learning pricing algorithms, can lead to higher equilibrium prices in the presence of other reinforcement learning competitors. Secondly, we provide new insights into algorithmic collusion by demonstrating how the self-learning algorithms' first-explore-then-exploit nature can introduce non-stationarity to the market environment, thereby hindering their ability to search for optimal policies. In contrast, simple rule-based pricing algorithms can promote stationarity, enabling intelligent algorithms to learn to charge near-monopoly prices.

Currently, most of the papers on algorithmic pricing approach the problem by theoretical modeling or simulation. Empirical evidence in this area is somewhat lacking, with the notable exception of Assad et al. (2020), whose study of German gasoline retailers suggests that the (inferred) adoption of algorithmic pricing approaches may have led to an increase in margins of up to 38%. In another study, Brown and MacKay (2021) present compelling empirical evidence that the adoption of pricing algorithms significantly impacts pricing patterns. Additionally, Chen et al. (2016) offer empirical evidence on the effects of pricing algorithms on pricing behavior.

Empirically studying *learning algorithms* is even more challenging, and the literature has primarily investigated learning algorithms only via simulations on the canonical logit demand model (e.g. Calvano et al. (2020), Asker et al. (2022), Hansen et al. (2021), Johnson et al. (2020)). To make some progress, this paper steps forward a bit and takes an experimental-empirical approach: we not only study algorithmic competition in the logit demand model but also validate our results by estimating a more realistic structural demand model and performing counterfactuals on the estimated model. Finally, we also provide a theoretical framework to study competition between automated pricing algorithms and provide results on when simple rule-based and Q-learning algorithms may dominate each other.

Our paper also complements the studies on non-sequential search models (e.g. (Mehta et al., 2003, De Los Santos et al., 2012, Honka, 2014)) by estimating a structural demand model where consumers search for fit but not for prices using detailed dataset from a large e-commerce platform.

3 Competition With Pricing Algorithms

3.1 The Market

We study the oligopoly price competition in a repeated game setting. The market consists of I consumers ($i = 1, 2, ..., I$), where each consumer demands at most one product. There are J differentiated products in the market provided by J sellers ($j = 1, 2, ...J$). Each seller sells their own product (with a constant marginal cost mc_j) in each and every period without capacity constraints. At the beginning of each time period ($t = 0, 1, ...$), the sellers set prices for their products (p_j^t) simultaneously, and the price history from previous periods is common knowledge. At the end of each period, a per-period profit is realized for each seller $r_j^t = (p_j^t - mc_j)D_j^t$, where D_j^t is the per-period demand for seller j determined by a demand function $D_j^t(p_j^t, p_{-j}^t)$.

For model tractability, we use discretized prices throughout the paper. We use p_j^{Nash} and p_j^{Mono} to denote the Bertrand-Nash price (competitive price) and monopoly price of the stage game for seller j, respectively. We then construct a set of equally spaced grid points $A = \{p_{j,k}\}, k \in \{1, .., K\}$ where $p_{j,1} = p_j^{Nash}$, $p_{j,K} = p_j^{Mono}$, and $p_{j,2}$ to $p_{j,K-1}$ are evenly spaced in between. The space between two consecutive price grids $\Delta p_j = (p_j^{Mono} - p_j^{Nash})/(K-1)$. We use $K = 10$ throughout the paper unless specified otherwise.

3.2 The Algorithm

3.2.1 Rule-Based Algorithm

There are three main types of rule-based pricing, depending on how price changes are triggered. The first type is *Competitor-based pricing*, where price changes are triggered by competitors' actions. This is the most common strategy used in practice – most large e-commerce platforms and almost all large third-party repricing platforms provide such tools. For example, the 'Target Low Price'[2] tool provided by Amazon allows sellers to adjust prices to match the lowest price in the market. On *informed.co*, sellers can specify the competitors they want to compete with, by either star rating, fulfillment methods, or item condition (whether new or used). Sellers can then choose to always set prices equal to these competitors' prices or lower than them by a fixed amount or percentage. The seller can also choose a minimum price to make sure the price never goes below that price.[3] These repricers can react to competitors' price changes almost instantaneously, for example, according to *AlphaRepricer*, it can reprice a product every two minutes[4], which means that any change in competitors' prices will trigger a reaction within two minutes. The second type of rule-based algorithm is *Sales-based pricing*, where price changes are triggered by the change in sales volume in the past. For example, Amazon provides sellers with tools to automatically decrease the price if its sales volume drops below a certain threshold in a certain period of time.[5] The third type of rule-based algorithm is *Time-based pricing*, where price changes are triggered by the time of the day or day of the week. For example, using *RepricerExpress* – a third party repricing solution – sellers can stop repricing or raise their price at a pre-specified time of the day.[6] Table 1 summarizes the commonly used rule-based pricing algorithms.

Table 1: Rule-based Algorithm Type

Algorithm Type	**Provider**	**Example**
Competitor-based	Amazon, RepricerExpress, Informed.co, BQool, ChannelAdvisor, SellerEngine, SellerActive, Aura, AlphaRepricer	Match Low Price (Always set price equal to the lowest price in the market)
Sales-based	Amazon, SellerEngine,SellerActive	Sales Velocity Pricing
Time-based	RepricerExpress	Sleep-mode Pricing

[2]This 'Target Low Price' mechanism should not be confused with the 'Match Price Policy' or 'Match Price Guarantee' provided by retailers like Walmart and Target. The latter guarantees that exactly the same product cannot be found elsewhere at a lower price. The former is a tool that individual sellers can use to adjust prices according to competitors' prices, where the matches are not necessarily defined by multiple sellers selling identical products, and prices are not necessarily matched exactly.

[3]`https://help.informed.co/en/articles/2319357-build-your-own`

[4]`https://alpharepricer.com/blogs/why-is-alpha-repricer-better-than-other-repricers/`

[5]`https://sellercentral.amazon.com/gp/help/external/FRJDFLFPWZSAG67`

[6]`https://support.repricer.com/sleep-mode`

In this paper, we focus on the most commonly applied rule-based algorithm – the competitor-based pricing algorithm. Its popularity comes from the fact that it can react to opponent sellers' price changes almost instantaneously to prevent loss. By contrast, the sales-based pricing algorithm adjusts price only after the effect of opponent sellers' price change has been reflected in one's own sales. Most third-party repricing software gives sellers options to customize the competitor-based repricing rule, including choosing which seller they wish to compete with, choosing to match price exactly or undercut, and whether or not to set a minimum price to prevent loss. In the benchmark case, we focus on a widely used competitor-based pricing algorithm: *Undercut Lowest Price* (*UL*), where prices are adjusted to undercut the lowest price among competitors by a fixed amount or fixed percentage. In Section 4.2.1, we study three variants of *UL*: *Undercut Lowest Price with a Minimum Price*, *Target Lowest Price*, *Target Lowest Price with a Minimum Price.*

Specifically in our setting, the simple rule-based algorithm will always undercut the *last period* lowest price among competitors by one price grid. The policy function of the simple rule-based algorithm $\mathit{\Phi}_R$ is specified in Equation 1

$$\mathit{\Phi}_R : p_j^t = min(p_{-j}^{t-1}) - \Delta p_j, \forall t \tag{1}$$

3.2.2 Q-learning Algorithm

Following the emerging literature on algorithmic pricing in economics (e.g. Calvano et al. (2020), Asker et al. (2022), Johnson et al. (2020), Waltman and Kaymak (2008)), we use Q-learning as a representative example of AI-powered learning algorithm. There are several reasons for this choice. First, Q-learning is a *model-free* reinforcement learning algorithm, which means that it does not need any knowledge of the environment. It is well suited for pricing applications since market conditions and demand functions are often unknown to sellers. Second, Q-learning is one of the most popular RL algorithms used by computer scientists: the implementation is simple and the number of hyper-parameters to tune is significantly smaller compared with neural network based algorithms. Third, the logic behind

Q-learning is clear, which makes it possible for us (researchers) to study its learning dynamics.

A Q-learning algorithm can learn the optimal strategy in a stationary Markov decision process: in period t, given the state ($s^t \in S$), a Q-learning seller j takes an action ($p_j^t \in A$). A reward (r_j^t) is realized and the current state transits to a new state in the next period ($s^{t+1} \in S$) according to some time-invariant state transition rule. The goal of the Q-learning algorithm is to maximize the long-term discounted cumulative reward ($E[\sum_t \delta^t r_j^t]$), where $\delta < 1$ is the discount factor. The algorithm is forward-looking since it does not simply take the action that maximizes the current period reward. Instead, it also takes into account the effect of the current period action on future period payoffs through state transitions. The value to seller j given a state can be represented by the Bellman's value function

$$V_j(s^t) = \max_{p \in A} \{E[r_j^t | s^t, p] + \delta E[V(s^{t+1}) | s^t, p]\} \tag{2}$$

For the Q-learning agent, the long-term value of taking action p_j^t at state s^t is represented by a 'state-action value function' – Q-function

$$\begin{aligned} Q_j(s^t, p_j^t) &= E(r_j^t | s^t, p_j^t) + \delta V_j(s^{t+1}) \\ &= E(r_j^t | s^t, p_j^t) + \delta E[\max_{p \in A} Q(s^{t+1}, p) | s^t, p_j^t] \end{aligned} \tag{3}$$

If the state space (S) and action space (A) are finite, the Q-values can be stored in a $|S| \times |A|$ table. Given this Q-table, after observing state s^t, the algorithm will take the action in set A that maximizes the Q-value. The core of the Q-learning algorithm is to learn the Q-values, which can be achieved by interacting with the environment repeatedly and updating the Q-table constantly. Specifically, at time t an action p_j^t is taken, the Q-value $Q_j(s^t, p_j^t)$ will be updated as follows, while the Q-values for other state-action combinations remain unchanged:

$$Q_j(s^t, p_j^t) = (1 - \alpha) Q(s^t, p_j^t) + \alpha (r_j^t + \delta \max_{p \in A} (Q_j(s^{t+1}, p))) \tag{4}$$

where the additional parameter α is the learning rate. Intuitively, the update rule moves the current Q-value α "steps" towards the current-period reward plus the discounted value of the next period state.

Since Q-learning uses a table to store Q-values. Thus it can only work in finite action space. We use the price grid constructed earlier as the action space for seller j: $A_j = \{p_{j,k_j}\}, k_j \in \{1, .., 10\}$. Throughout the paper, we allow the Q-learning algorithm to have a single-period memory. That is, the state consists of the last-period prices of all sellers. Specifically, $S = \{(p_{1,k_1}, p_{2,k_2}, ..., p_{J,k_J}), k_1, k_2, ..., k_J \in \{1, .., 10\}$. The size of the state space $|S| = 10^J$. Although seemingly restricted, the number of potential strategies with one-period memory is actually extremely large. Specifically, there are in total 10^{10^J} potential strategies for the Q-learning algorithm to choose from.

Like other learning algorithms, Q-learning faces the exploration-exploitation trade-off. On the one hand, it tries to pick the optimal action based on the current knowledge; on the other hand, it needs to explore other options that have not been visited to learn about their potential. A commonly used exploration strategy is called the ϵ-greedy strategy. It takes the greedy action with probability $1 - \epsilon$ and randomly picks an action from the action space with probability ϵ. In practice, ϵ is often set to decrease over time, which allows the algorithm to explore intensively at first, then take the greedy action once the learning process is completed. Unless otherwise mentioned, in this paper we set ϵ to decay exponentially over time: $\epsilon(t) = e^{-\omega t}$, where ω is a parameter that controls how fast ϵ decays.

The Q-learning algorithm needs a stopping rule. In the paper, unless otherwise mentioned, we use a stopping rule similar to Calvano et al. (2020): We stop the Q-learning process when the optimal actions do not change for all states for at least 1000 periods. With regards to initialization, given that the learning algorithm possesses no prior knowledge of the competitor's strategy, we initialize the Q-value of taking action p in state s as the long-term profit by charging price p assuming the competitor selects the price randomly.

Once the learning is finished, the Q-learning algorithm's pricing policy Φ_Q can be specified as in Equation 5:

$$\Phi_Q : p_j^t = \arg\max_p Q(s^t, p), \forall t \tag{5}$$

3.3 Competition

We begin with the case with 2 sellers in the market ($J = 2$). We study two competitive scenarios: 1) when both sellers use a Q-learning algorithm (Q-Q scenario hereafter) and 2) when a seller uses a simple rule-based algorithm to compete with a Q-learning algorithm (Q-R scenario hereafter). Without loss of generality, we assume seller 1 always uses a Q-learning algorithm, and seller 2, the focal seller, uses Q-learning in the Q-Q scenario and the simple rule-based algorithms in the Q-R scenario. Algorithm 1 and Algorithm 2 provide the details of the competition in the Q-Q and Q-R scenarios in pseudo-code form, respectively.

Algorithm 1: Q-learning seller v.s. Q-learning seller

```
Parameters: $\alpha \in (0,1]$, $\delta \in (0,1)$, $\omega \in (0,\infty)$
Initialize $Q_1(s,p)$, $Q_2(s,p)$, for all $s \in S$, $p \in A$
Initialize $t = 1$
Initialize $s^1$
while not converge do
    $\epsilon \leftarrow e^{-\omega t}$
    $p_1^t \leftarrow \begin{cases} \arg\max_p Q_1(s^t,p) & \text{with probability } 1-\epsilon \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$
    $p_2^t \leftarrow \begin{cases} \arg\max_p Q_2(s^t,p) & \text{with probability } 1-\epsilon \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$
    Calculate $r_1^t$, $r_2^t$
    $s^{t+1} \leftarrow (p_1^t, p_2^t)$
    $Q_1(s^t,p_1^t) \leftarrow (1-\alpha)Q_1(s^t,p_1^t) + \alpha(r_1^t + \delta \max_{p\in A}(Q_1(s^{t+1},p)))$
    $Q_2(s^t,p_2^t) \leftarrow (1-\alpha)Q_2(s^t,p_2^t) + \alpha(r_2^t + \delta \max_{p\in A}(Q_2(s^{t+1},p)))$
    $t \leftarrow t+1$
end while
```

Algorithm 2: Q-learning seller v.s. Simple rule-based seller

```
Parameters: $\alpha \in (0,1]$, $\delta \in (0,1)$, $\omega \in (0,\infty)$, $\Delta p$
Initialize $Q_1(s,p)$, for all $s \in S$, $p \in A$
Initialize $t = 1$
Initialize $s^1$
**while** not converge **do**
    $\epsilon \leftarrow e^{-\omega t}$
    $p_1^t \leftarrow \begin{cases} \arg\max_p Q_1(s^t, p_1^t) & \text{with probability } 1-\epsilon \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$
    $p_2^t \leftarrow s^t(1) - \Delta p$
    Calculate $r_1^t$, $r_2^t$
    $s^{t+1} \leftarrow (p_1^t, p_2^t)$
    $Q_1(s^t, p_1^t) \leftarrow (1-\alpha)Q_1(s^t, p_1^t) + \alpha(r_1^t + \delta \max_{p\in A}(Q_1(s^{t+1}, p)))$
    $t \leftarrow t + 1$
**end while**
```

4 Market Environment 1: Logit Demand Model

Our initial focus is on the market setting in which the demand model adheres to the canonical logit demand model. We use a_j to denote the product quality of the product j, We use μ to denote the level of horizontal differentiation among the products. The demand function is as follows:[7]

$$D_j^t = \frac{\exp(\frac{a_j - p_j^t}{\mu})}{\sum_j \exp(\frac{a_j - p_j^t}{\mu}) + \exp(\frac{a_0}{\mu})} \tag{6}$$

where a_0 is a parameter that captures the demand for the outside good. As mentioned before, the marginal cost of product j is c_i, the profit for the seller j in period t is $r_j^t = (p_j^t - mc_j)D_j^t$. In the baseline model, we focus on the case where $J = 2$ and the two sellers are symmetric (i.e., $mc_1 = mc_2$, $a_1 = a_2$). We assume that the demand function is unknown to either party, this assumption highlights an advantage of using a sophisticated learning algorithm over a simple rule-based algorithm: The optimal price cannot be calculated *ex ante*, but can be learned through repeated interactions.

[7]We normalize the number of consumers to 1.

4.0.1 Algorithm Parameters

For model hyper-parameters we choose $\alpha = 0.15$, $\omega = 1.5 \times 10^{-5}$ in the Q-Q scenario, and $\alpha = 0.15$, $\omega = 1 \times 10^{-4}$ in the Q-R scenario. We use this configuration for the experiment unless otherwise mentioned. As for parameters for the demand function, in the baseline model, we set $a_1 = a_2 = 2$, $a_0 = 0$, $mc_1 = mc_2 = 1$, and $\mu = 0.25$.

4.1 Experiment Outcomes: Baseline Setting

In this section, we present the results obtained from the experiment and compare the effectiveness of the Q-learning and the simple rule-based algorithms in the Q-Q and Q-R scenarios respectively by analyzing three key dimensions: 1) equilibrium prices and profits, 2) learning process and dynamics, and 3) the punishment strategies learned by the algorithms.

4.1.1 Price and Profit Comparison

For both the Q-Q scenario and Q-R scenario, we run the experiment 100 times using the baseline parameters. The prices charged by seller 2 (averaging over the last 1000 periods) and the corresponding single period profit for seller 2 are reported in Table 2. The values within parentheses represent the standard deviations across the 100 runs.

	Q-learning Algorithm	**Rule-based Algorithm**
Price	1.787(0.070)	1.870(0.000)
Profit	0.320(0.021)	0.361(0.000)

Table 2: Price and Profit Comparison for Focal Seller (Baseline)

Note: The column corresponding to Q-learning algorithm (Rule-based algorithm) represents the results for the focal seller in the Q-Q (Q-R) scenario.

The results for the Q-Q scenario are similar to those shown in Calvano et al. (2020). Specifically, the two Q-learning algorithms learn to charge supra-competitive price (Bertrand-Nash price is 1.47), however, the price they converge to (around 1.79) is significantly lower than the monopoly price (1.92). By contrast, in the Q-R scenario, the Q-learning algorithms quickly learn to charge a price close to the monopoly price softening the competition even

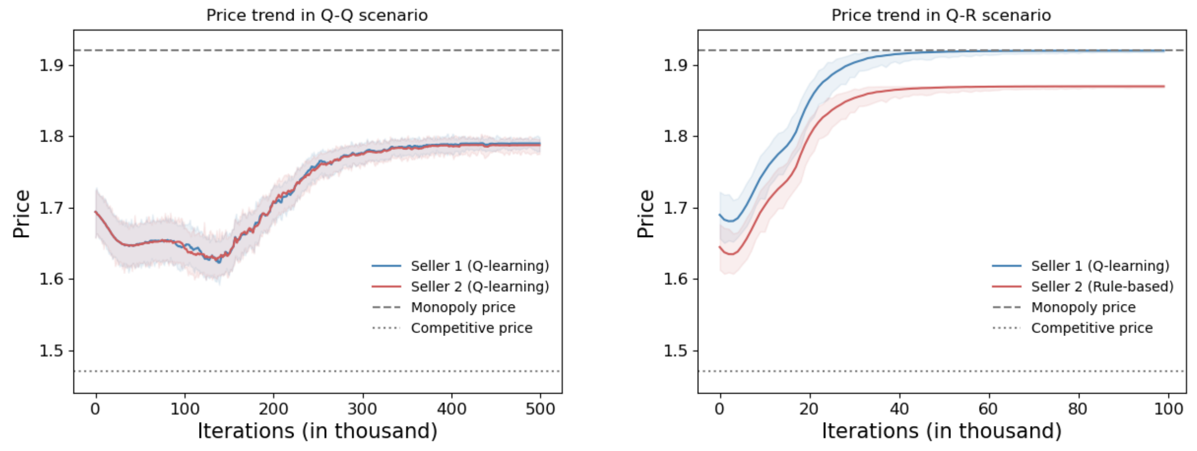


Figure 1: Learning dynamics

Note: In the figure, the points on the solid lines represent the moving average (window size = 1000) of the average price across the 100 experiments at any t (number of iterations). The upper and lower bound of the shaded area represent the maximum and the minimum value within the moving window.

further, which benefits the simple rule-based algorithm adopter (i.e., receive a higher profit) compared with the Q-Q scenario.

4.1.2 Learning Dynamic Comparison

After observing the distinctions in equilibrium price and profits in the two scenarios, we next compare their learning dynamics. From the 100 experiments in each scenario, we keep track of the prices charged by the algorithms in each period and report in Figure 1 the average price charged by the two sellers as a function of the number of iterations.

Learning Speed and Experiment Loss

We first compare the number of iterations needed for convergence between the Q-Q and Q-R scenarios. In the Q-Q scenario, roughly 350,000 iterations are needed for convergence. In the Q-R scenario, by contrast, only 40,000 iterations are needed. We next compare the loss incurred during the experimentation process. We take the equilibrium profit after prices converge as benchmark. In each period during experimentation, we compute the difference between the profit seller 2 gets and the benchmark profit, and we sum them up to measure the total loss in each scenario. It turns out that seller 2's experiment loss is on average 2.74

times higher in the Q-Q scenario compared with the Q-R scenario.

Learning Dynamics

Next, we examine the differences in learning processes between the two scenarios by utilizing the phase plot to illustrate the strategies employed by each seller. Specifically, we analyze the evolution of the strategy phase plots over time. The figure consists of four subfigures, each representing the strategy phase graph at a distinct learning stage, as indicated at the top of each sub-figure. The x-axis and y-axis in each phase graph represent the 10 different prices that can be chosen by seller 1 and seller 2, respectively. The 100 nodes in the graph represent the 100 states, which correspond to the price pair selected in the last period. For instance, the lower left node denotes the state in which both sellers chose Bertrand-Nash price in the previous period, while the upper right node represents the state where both sellers opted for the Monopoly price. The node size reflects the frequency at which a particular price pair is selected. Initially, all nodes are of equal size, as both algorithms choose prices randomly, resulting in all price pairs being selected equally frequently. The arrows in the graph denote the greedy strategies employed by the algorithms. That is, given a state, the arrow points to the price pair selected based on the Q values of the two algorithms. We observe that initially, all nodes point to the third-lowest node, which is the best response price when the competitor selects a price randomly.

When the learning starts, the algorithms transit from selecting prices randomly to favoring lower prices as the exploration rate diminishes. As the learning progresses to 39%, nodes located in the lower left corner are larger on average, indicating a higher frequency of selecting low-price pairs, which aligns with our expectations. Concurrently, as the competitor shifts from random price selection to favoring lower prices, the true Q values for states corresponding low prices decline. It is noteworthy that the Q values associated with low prices are updated more frequently and drop at a quicker rate than those for medium and high prices (since a Q-value is updated only when the corresponding action is taken, which happens more frequently for low prices). Eventually, the Q values for low prices dip below

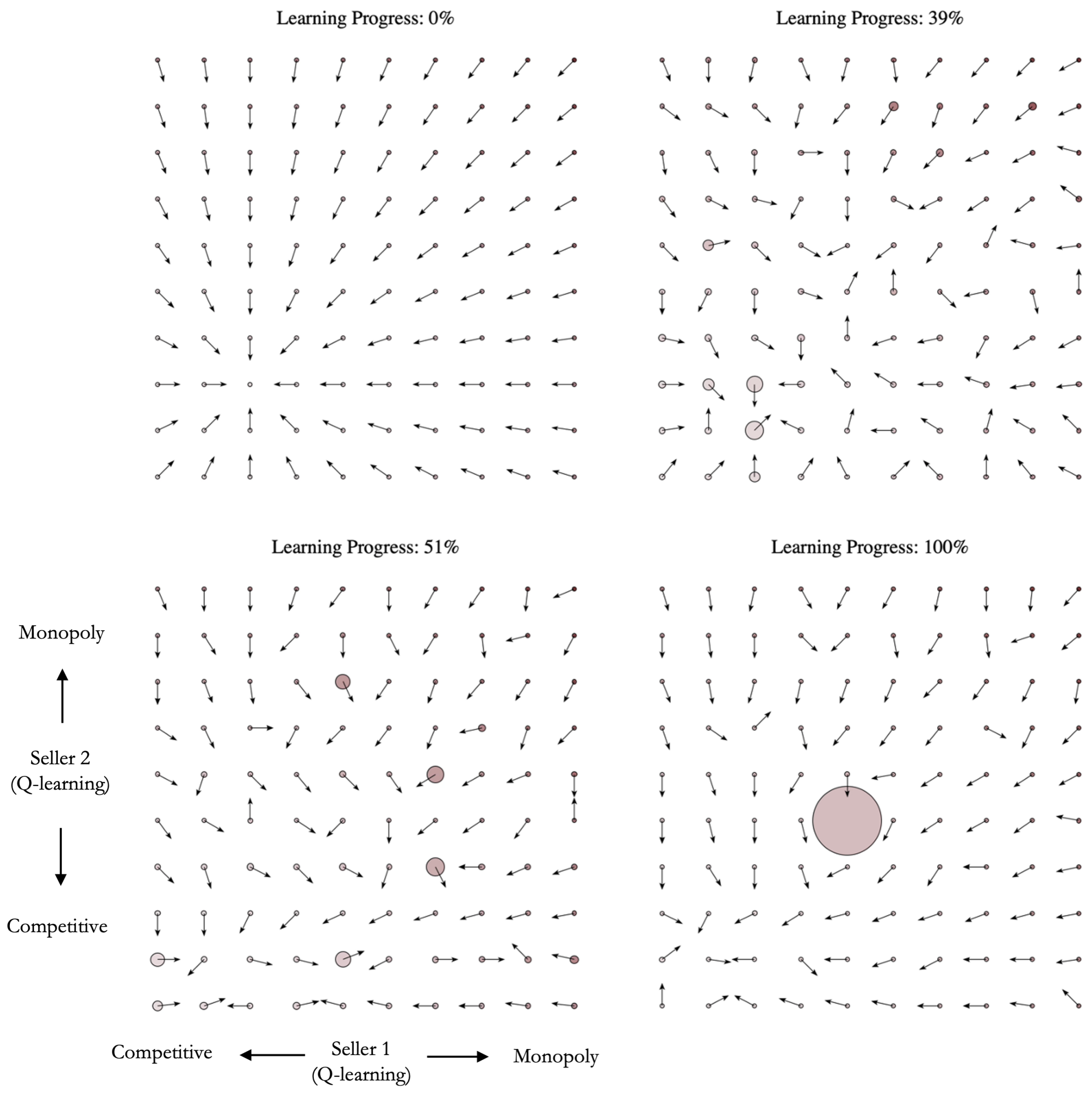


Figure 2: Learning Dynamics in the Q-Q scenario

the Q values for medium prices, prompting the algorithms' greedy strategies to switch from selecting low to medium prices. At the 51% mark, as shown in the third sub-figure, the large nodes move upward, indicating that the algorithms begin to coordinate on medium prices. At this stage, Q values for medium prices are updated more frequently, although they do not plummet as steeply as the Q values for low prices because the competitor adopts a less competitive strategy at this stage. Consequently, the Q values for medium prices never drop below those for high prices. As depicted in the final sub-figure, the upward trend eventually plateaus, resulting in both sellers repeatedly charging medium prices in the equilibrium.

The phase plots for the Q-R scenario are presented in Figure 3. In the Q-R scenario, the simple rule-based algorithm's strategy is fixed and the environment for the Q-learning algorithm is stable. The competitor faced by the Q-learning seller is not required to engage in random exploration, as observed in the Q-Q scenario. This advantageous feature enables the Q-learning algorithm to learn to coordinate on the highest price with remarkable speed.

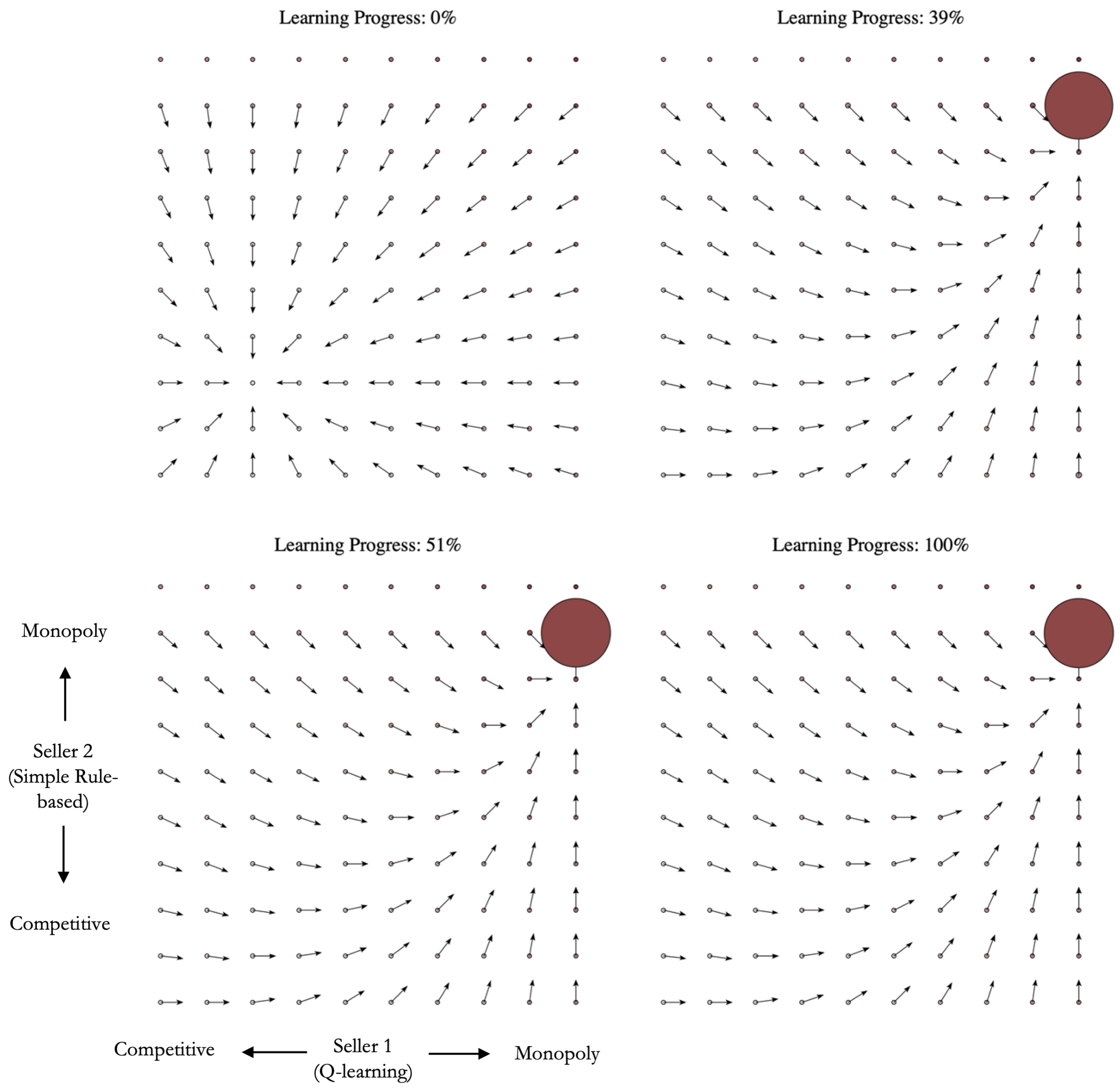


Figure 3: Learning Dynamics in the Q-R scenario

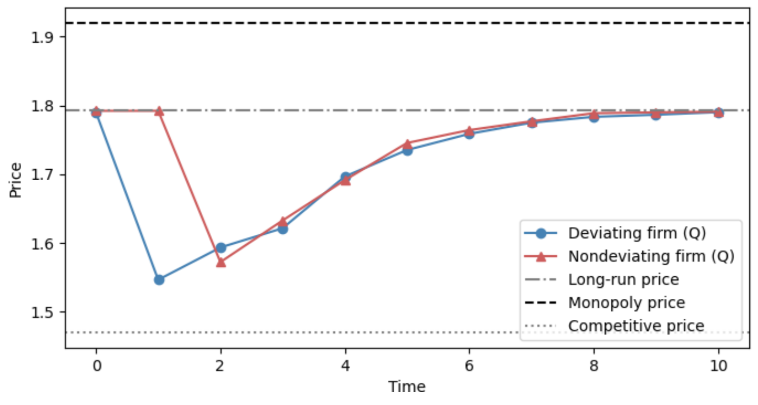
4.1.3 Deviation and Punishment Strategy Comparison

To understand the mechanism driving the competition outcome discussed in the previous section, we study the strategies learned by the algorithms. For each experiment, we retrieve the two sellers' converged Q-tables to derive the sellers' strategies (i.e., what action to take in each state (s), which is $\arg\max_{a \in A} Q(s, a)$). We then force seller 1 to deviate from the learned strategy in one period. Seller 2 continues to play according to its learned strategy in this period, and in the subsequent period, both sellers play their learned strategies. We find that in most of the sessions, the two sellers will come back to the pre-deviation price within 10 moves (periods). We then plot the impulse-response functions derived from this exercise for all 100 experiment sessions [8]. Specifically, shown in Figure 4a are the prices charged by the two sellers after the forced deviation of seller 1 in the Q-Q scenario. A similar plot is shown in Figures 4b and 4c for the Q-R scenario.

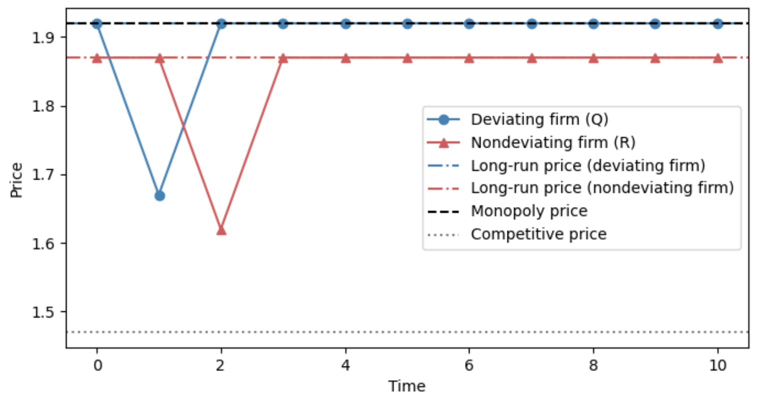
We then take a closer look at the punishment strategies learned by the Q-learning algorithm in the Q-Q and Q-R scenarios. We use a directed graph to represent the limit strategies. Specifically, each node represents a state (a price pair), and each edge represents the state transition given the two firms' strategies. For example, an arrow pointing from node A to node B means that at the state specified in node A, the two algorithms' strategies will generate a price pair as specified in node B. The square is the absorbing state where there are no arrows pointing out. The shade of the nodes denotes the profit gain, darker nodes mean more collusive price pairs. The nodes' size represents the node's importance in the graph(as measured by betweenness centrality).

In the Q-Q scenario, a deviation from the equilibrium state by seller 1 results in a return to the absorbing node rather than remaining stuck in another node. Thus, the punishment strategy in this scenario is not a grim trigger strategy. The graph illustrates a deviation-punishment-reconciliation path, represented by a blue line, which requires the two sellers to pass through several progressively darker nodes to return to the absorbing

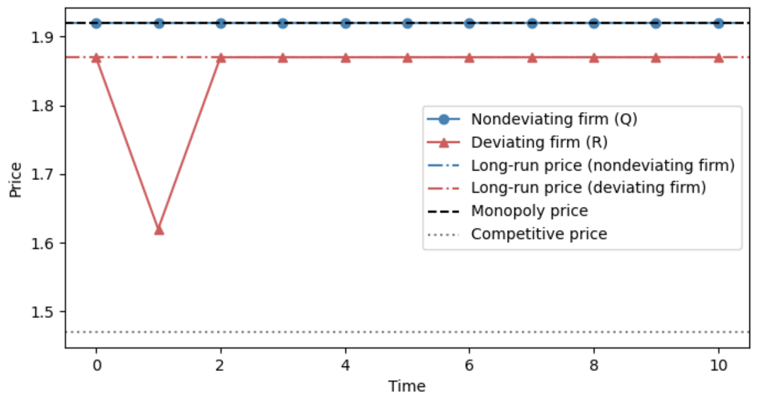
[8] In this experiment we focus on the sessions that do not lead to price cycles



(a) Q-Q scenario



(b) Q-R scenario



(c) Q-R scenario

Figure 4: Deviation-Punishment scheme

The plot shows how prices evolve according to both sellers' limit strategies after an exogenous price cut for one of the sellers in 3 different scenarios. Figure (a): In the Q-Q scenario, seller 1 cuts the price by 5 grids in period 1. Figure (b): In the Q-R scenario, seller 1 (Q-learning algorithm) cuts the price by 5 grids in period 1. Figure (c): In the Q-R scenario, seller 2 (Simple rule-based algorithm) cuts the price by 5 grids in period 1.

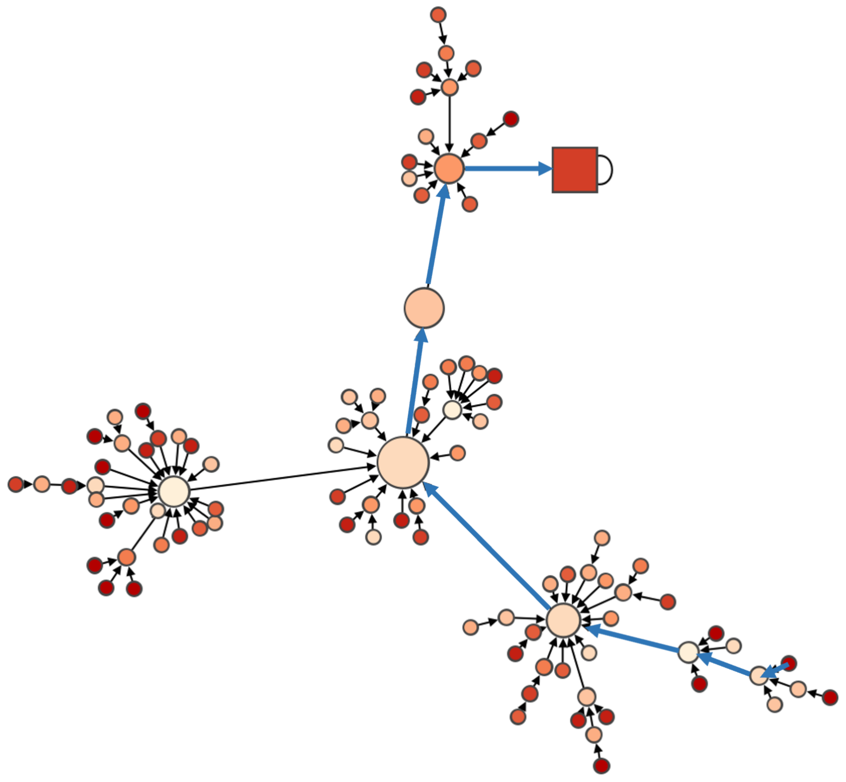


Figure 5: Graph of Strategies in Q-Q scenario

This is a visual representation of the strategies depicted through a directed graph. The square represents an absorbing node. All other nodes are represented by circles. The shades reflect the profit gain, with darker nodes indicating higher profit gains. The size of each node corresponds to its importance in the graph, as measured by betweenness centrality. In each graph, a blue line is used to highlight a specific deviation-punishment-reconcile path

node. This strategy effectively punishes the deviating seller for multiple periods and eases the punishment over time representing a stick and carrot strategy.

The strategy graph in the Q-R scenario exhibits a distinct pattern: no matter which node the seller deviates to, it comes back to the equilibrium state almost immediately. The Q-learning algorithm employed in this scenario learns an 'always cooperate' strategy. The rationale for this is that the simple rule-based algorithm does not facilitate reconciliation unless the Q-learning algorithm brings the price back to the original level. The Q-learning

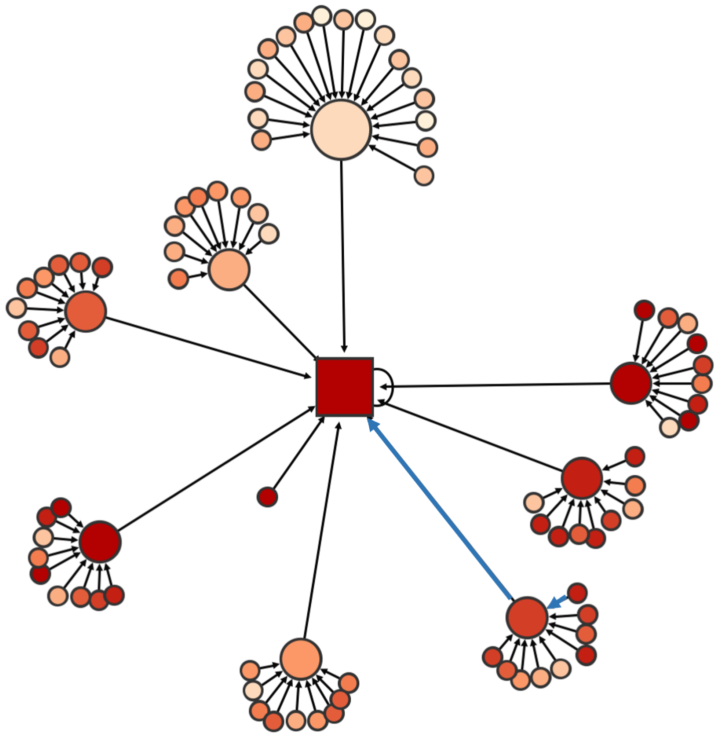


Figure 6: Graph of Strategies in Q-R scenario

algorithm picks this up from the past experience of deviation. Consequently, the algorithm increases the price immediately after a deviation to reduce the length of the punishment period and mitigate losses.

4.2 Experiment Outcomes: Alternative Settings

While the main intuition is shown in the baseline setting, we also work on several alternative settings to show the robustness of our results. In this section, we show how changes in logit demand parameters and different types of rule-based algorithms affect our main result. We put the study on multiple sellers and more advanced RL pricing algorithms in Appendix A.

4.2.1 Different Rule Based Pricing Algorithms

We now examine the performance of other rule-based algorithms when they compete with the Q-learning algorithm. Specifically, we consider the following three variants of UL: 1) *Undercut Lowest Price with a Minimum Price (ULM)*: The agent using ULM always charges one price grid below the competitor's last-period price but will never go below a user-defined minimum price p_{min}[9]. 2) *Target Lowest Price (TL)*: This algorithm will match the competitor's last-period price. 3) *Target Lowest Price with a Minimum Price (TLM)*: The agent using TLM always match the competitor's last-period price but will never go below a user-defined minimum price p_{min}.

The equilibrium prices and equilibrium profits for seller 2 by using the above-mentioned rule-based strategies are reported in Table 3, and the corresponding learning dynamics are shown in Figure 7. All these rule-based strategies outperform the Q-learning algorithm when facing a Q-learning competitor, in terms of giving seller 2 higher equilibrium profits. Therefore, our results from the baseline setting are robust to the choice of simple rule-based algorithms.

	ULM	**TL**	**TLM**
Price	1.870(0.000)	1.914(0.014)	1.918(0.008)
Profit	0.360(0.000)	0.337(0.001)	0.337(0.000)

Table 3: Price and Profit for Different Rule-based Algorithm Competing with Q-learning Algorithm

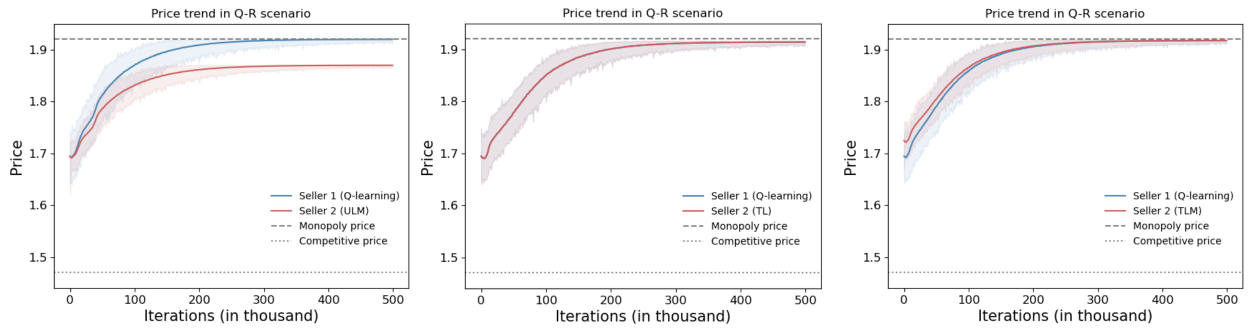


Figure 7: Learning Dynamics When Different Rule Based Algorithms Compete With a Q-Learning Algorithm

4.2.2 Alternative Market Structure

Horizontal Differentiation (Changes in μ):

[9]In this section, we set p_min to the fourth price grid

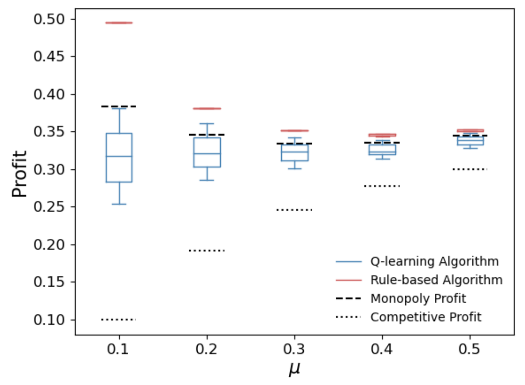
According to the demand function specified in Equation 6, horizontal differentiation between the products is captured by μ: When μ approaches 0, the products become perfect substitutes, and as μ increases, the *cross price elasticity of demand* decreases and the products become more independent with each other. We had set $\mu = 0.25$ in the baseline, and now we consider five different values of μ ($\mu \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$). Note that when μ changes, both the Bertrand-Nash price and the monopoly price will change accordingly. The results (seller 2's profit) are shown in Figure 8a. As μ decreases (degree of horizontal differentiation decreases), seller 2's equilibrium profit in the Q-Q scenario drops. This is because as the two products become similar, being undercut by competitors will result in lower short-term profits. Consequently, the algorithm will assign lower Q-values to high prices and higher Q-values to low prices initially, which makes coordinating on medium or high prices more difficult later on. However, this pattern is flipped in the Q-R scenario, since the Q-learning algorithm is always able to charge the highest price, and the lower the μ, the higher profit that a simple rule-based algorithm can get by always undercutting its competitor. Overall, for different levels of horizontal differentiation, the rule-based algorithm outperforms the Q-learning algorithm consistently.

Uncertainty in Demand (Changes in a_0):

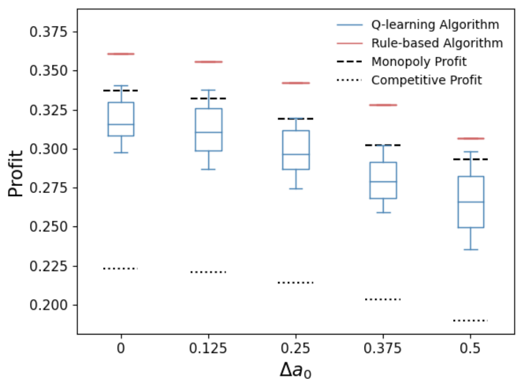
To induce additional demand uncertainty, we let the quality of the 'outside good' fluctuate from period to period. Specifically, in each period, the quality of the outside good (a_0) could be either $-\Delta a_0$ or Δa_0, each with 50% probability. We experiment with five different values of Δa_0 ($\Delta a_0 \in \{0, 0.125, 0.25, 0.375, 0.5\}$) in the study. Similar to the previous case, when Δa_0 changes, both the Bertrand-Nash price and the monopoly price will change accordingly. The results (seller 2's profit) is shown in Figure 8b. Again, the rule-based algorithm gives seller 2 higher equilibrium profit across all cases.

5 Market Environment 2: A Structural Demand Model

Up to this point, our analysis has been focused on investigating pricing algorithm competition in hypothetical environments, where the demand function is assumed to take a standard logit



(a) Horizontal differentiation



(b) Demand Uncertainty

Figure 8: Seller 2's profit by using Q-learning algorithm and rule-based algorithm under different values of (a) μ and (b) Δa_0.

Note: The boxes in the figure represent the distributions of the seller 2's profits from 100 experiments. The box extends from the first quartile to the third quartile, with a line at the median. The whiskers extend from the box by 0.5× the inter-quartile range (IQR).

form. In this section, we will work on a more sophisticated consumer non-sequential search demand model where model parameters are estimated from real-world data.

5.1 Data

The dataset employed in our study is transaction-level data from a prominent Chinese e-commerce platform, JD.com. The platform's operational mechanism involves presenting consumers with products from multiple sellers in each product category, accompanied by pertinent details such as prices, ratings, and key product features. Upon selecting a product listing by clicking on it, consumers can obtain more comprehensive information. Specifically, our dataset captures the click and purchase behavior of over 2.5 million consumers within a specific product category throughout March 2018.

The data contains 4 tables: *SKU Table* describes the characteristics of all 31,868 SKUs that belong to a single product category receiving at least one click during March 2018. For each SKU, it records the unique identifier SKU ID, whether this SKU is from a first-party seller or a third-party seller. *Consumer Table* describes the characteristics of each consumer, including the unique identifier consumer ID and a bunch of consumer demographics like age, gender, marital status etc. *Click Table* records the information of all the click events, including the consumer ID that initiates the click, the SKU ID that she clicks on, and time of the click. *Purchase Table* records the information of all the purchase events, including consumer ID, SKU ID, time of the purchase, purchase quantity, listed prices, and effective prices.

Instead of studying the competition among all SKUs, we focus on a small subset of them. The motivation of this choice is two-fold: 1), the sales of these products follow a 'long-tail distribution', where over 96% of SKUs have single-digit daily sales. We aim to concentrate on the subset of products that have a sizeable volume of sales. 2), we focus on a group of products that are close substitutes, rather than analyzing the competition between dissimilar products. For example, we would like to study the competition between electric shavers from two brands, instead of the competition between an electric shaver and a water flosser. We thus pre-process the data and identify a small subset in which all SKUs have decent sales volume and are close substitutes. In Appendix B, we provide details on how we determine if two products are close substitutes and construct the subset.

Our study concentrates on a specific subset of products that exhibit substantial sales volume and are identified as close substitutes, to derive an accurate demand model. This category comprises 5 distinct products, and these products have been clicked by more than 25,000 consumers. (In Appendix D, we report corresponding results for a case with another subset of 8 products for robustness.) The basic information of these 5 products and the involved consumers are shown in Table 4 and 5, respectively. For each purchase, we observe the purchase time and the original listed price, and the price discount that a consumer gets. Interestingly, we find the effective price (listed price minus discounts)[10] is almost identical for all consumers at any given time, indicating very little first- or third-degree price discrimination. The most common age segment is 26-35 years old and the most common education level is a Bachelor's degree. There are more females in the sample than males.

[10]Here we only consider direct discount but not volume discount since very few consumers buy more than one product.

The sample is approximately equally split between single and married customers.

	P1	P2	P3	P4	P5
Average price (¥)	68.84	52.27	57.94	71.54	69.05
1st/3rd party seller	1st	1st	3rd	3rd	1st
Unique clicks	14,171	15,478	5,351	4,512	4,683
Unique orders	3,942	3,490	1,247	1,179	726

Table 4: Product Information

Key Characteristics	Count(Percentage)
Number of consumers	25,339
Age	
$\leq$ 25 years old	6,335 (25.0%)
26 - 35 years old	11,567 (45.6%)
36 - 45 years old	5,604 (22.1%)
46 - 55 years old	1,107 (4.4%)
$\geq$ 56 years old	726 (2.9%)
Gender	
Male	5,657 (22.3%)
Female	19,682 (77.7%)
Marital Status	
Single	13,046 (51.5%)
Married	12,293 (48.5%)
Education Level	
Less Than High School	912 (3.6%)
High School or Equivalent	8,705 (34.4%)
Bachelor's Degree	13,620 (53.8%)
Post-graduate Degree	2,102 (8.3%)

Table 5: Descriptive Statistics of Consumer Characteristics

Table 6 provides a summary of the click and purchase behavior of consumers. Several key observations can be made from this table. First, 20.1% of consumers click on 3 or more products and 25.5% of consumers click on 2 products before making a purchase decision, suggesting that a substantial search cost is involved in the search-then-purchase process. Second, less than 1% of consumers purchase more than one product, indicating that it is highly unlikely that the product subset we have selected includes complementary items.

Key Characteristics	Count(Percentage)
Number of consumers	25,339
Choice Set Size	
1	13,775 (54.4 %)
2	6,459 (25.5 %)
≥ 3	5,105 (20.1 %)
Number of products purchased	
0	14,953 (59.0%)
1	10,192 (40.2%)
≥ 2	194 (0.8%)

Table 6: Descriptive Statistics of Consumer Clicks and Orders

5.2 Testing Search Models

In this study, we examine two major models that describe consumers' search-then-purchase behavior: the sequential search model (McCall, 1970, Weitzman, 1979) and the non-sequential search model (Stigler, 1961, Mehta et al., 2003, De Los Santos et al., 2012). While consumers' actual search behavior may fall somewhere between these two models, it's important to determine which model is more appropriate based on the context (De Los Santos et al., 2012). In our scenario, where consumers search for product fit rather than prices, we employ two empirical metrics to evaluate the search model. Our findings suggest that a non-sequential search model provides a better fit for our online search and purchase context. We use metrics such as recall and the conditioned probability of purchase to support our conclusion. We find that 58% of people show recall behavior and that the order in which products are searched has no significant impact on their purchase probability. We provide more details about the two metrics in Appendix C. Based on these results, we construct a structural model under the non-sequential search assumption to describe the market environment.

5.3 A Non-sequential Search Structural Model

In this section, we present a non-sequential search structural model that captures consumer search and purchase behavior within our empirical context. There are I consumers and J sellers in the market, each seller (seller j) supplies one product (product j) with marginal cost mc_j. Each consumer demands at most one product. We specify consumer i's utility of purchasing product j as:

$$u_{ij} = \eta_j + \beta_i p_j + \varepsilon_{ij} \tag{7}$$

where η_j is the product (or seller) fixed effect, p_j is the price of product j. β_i is individual specific price sensitivity: $\beta_i = \Gamma z_i + \varepsilon_\beta$ where z_i is a vector of individual characteristics for consumer i, Γ is a coefficient vector where the kth entry represents the part-worth of the kth characteristics on β_i, and ε_β is an unobserved heterogeneity component which is assumed to follow a normal distribution with mean 0 and variance Σ_β. ε_{ij} is the fit value between consumer i and product j. We assume ε_{ij} is $i.i.d$ distributed and follows a Type I extreme value distribution with a location parameter of zero and a scale parameter σ_ε ($\varepsilon_{ij} \sim EVT1(0, \sigma_\varepsilon)$).

Online marketplaces such as JD.com typically display important product information such as prices, ratings, number of reviews, and key features through thumbnail images or product listings. This allows users to quickly browse and access this information without the need to click on individual products. It is assumed that consumers have access to prices (p_j) and product values (i.e., product fixed effects, η_j) for all products upon entering the market. Further, consumers know the distribution of the fit value (ε) but cannot observe its realization at this point. To obtain the exact fit value, consumers must conduct a search by clicking on a specific product and examining the detailed product description, images, videos, and consumer reviews.

We model a non-sequential search process: in the first stage, consumers form a consideration set which is a subset of all the J products, based on the information available to them when they enter the market. In the second stage, consumers click on each of the products in the consideration set, learn the fit values, and decide which product to purchase or not to purchase at all (i.e., the outside option). Next, we explain these two stages in detail.

5.3.1 First Stage: Form Consideration Sets

Consumers determine their consideration set by balancing the expected benefit of inspecting all products within it against the search costs involved. On the one hand, having more products in the consideration set will increase the chances of finding great fit later on. On the other hand, consumers have to incur larger search costs for larger consideration sets (e.g., time and effort spent on visiting the product information pages). We use c to denote the cost of inspecting one product, and for simplicity, we assume the search cost is homogeneous across consumers and products. We use $\mathcal{S}$ to denote the universe of products, and consumers' consideration set is denoted as $S \subseteq \mathcal{S}$. Consumer i's expected utility of forming a consideration set S is

$$m_{iS} = \mathbb{E}[\max_{j \in S}\{u_{ij}\}] - c|S| \tag{8}$$

We can rewrite u_{ij} as $u_{ij} = \delta_{ij} + \varepsilon_{ij}$, where $\delta_{ij} = \eta_j + \beta_i p_j$. If we denote the CDF of the distribution of ε_{ij} as F, then the CDF of u_{ij} is $F(u - \delta_{ij})$, and the CDF of $\max_{j \in S}\{u_{ij}\}$ is $\prod_{j \in S} F(u - \delta_{ij})$. Then,

$$\mathbb{E}[\max_{j \in S}\{u_{ij}\}] = \int_{-\infty}^{\infty} u\rho(u)du \tag{9}$$

where is the pdf of $\max_{j \in S}\{u_{ij}\}$ and can be expressed as

$$\rho(u) = \frac{d}{du}[\prod_{j \in S} F(u - \delta_{ij})]$$

Plug $F(u) = \exp[-\exp[-u]]$ into Equation 9, and we get[11]

$$\mathbb{E}[\max_{j \in S}\{u_{ij}\}] = \gamma + \log[\sum_{j \in S} \exp(\delta_{ij})] \tag{10}$$

where γ is the Euler-Mascheroni constant.[12]

To smooth the choice set formation probability and allow consumers with similar levels of price sensitivity facing the similar product price to form different consideration sets, we follow De Los Santos et al. (2012) and assume that consumers are subject to an assessment error ζ when assessing the expected net benefit of forming consideration set S, where ζ_{iS} follows another Type I extreme value distribution with a location parameter of zero and a scale parameter of σ_ζ. Consumer i determine her consideration set by maximizing the expected utility associated with the consideration set plus the assessment error.

$$S = \arg\max_{S \in \mathcal{S}} [m_{iS} + \zeta_{iS}] \tag{11}$$

Consumer i's probability of forming a consideration set S follows a logit form:

$$P_{iS} = \frac{\exp[m_{iS}/\sigma_\zeta]}{\sum_{S' \in \mathcal{S}} \exp[m_{iS'}/\sigma_\zeta]} \tag{12}$$

[11]Specifically, let $v = \sum_{j \in S} \exp(-u + \delta_{ij})$, then we have $dv = -vdu$ and $u = \log(\sum_{j \in S} \exp(\delta_{ij})) - \log(v)$

$$\begin{aligned}
\mathbb{E}[\max_{j \in S}\{u_{ij}\}] &= \int_{-\infty}^{\infty} u \frac{d}{du}[\exp[-\sum_{j \in S} \exp(-u + \delta_{ij})]]du \\
&= \int_{-\infty}^{\infty} \frac{u \sum_{j \in S} \exp(-u + \delta_{ij})}{\exp[-\sum_{j \in S} \exp(-u + \delta_{ij})]} du \\
&= \int_{-\infty}^{\infty} \frac{uv}{exp(v)} du \\
&= -\int_{0}^{\infty} \frac{\log(\sum_{j \in S} \exp(\delta_{ij}))}{\exp(v)} dv + \int_{0}^{\infty} \frac{\log(v)}{exp(v)} dv \\
&= \gamma + \log[\sum_{j \in S} \exp(\delta_{ij})]
\end{aligned}$$

[12]We ignore this constant in the rest of the analysis since the constant term does not affect choice probability.

5.3.2 Second Stage: Purchase

Upon identifying the consideration set, a consumer proceeds to examine each product within the set and evaluates their respective fit values before deciding on which product to purchase, including the outside option. The likelihood of consumer i purchasing product j, conditioned on the formation of consideration set S, can be expressed as follows:

$$P_{ij|S} = \Pr(u_{ij} > u_{ik}, \forall k \neq j \in S) \tag{13}$$

Note that the fit value ε_{ij} is revealed to consumer i *after* she forms a consideration set and inspects all products within it. From the researchers' perspective, conditioned on the consideration set consumer i forms, the probability of the consumer purchasing product j, denoted as $P_{ij|S}$, follows a logit form:

$$P_{ij|S} = \frac{\exp[\delta_{ij}]}{\sum_{k\in S} \exp[\delta_{ik}] + \exp[\delta_0]} \tag{14}$$

The joint probability of consumer i forming consideration set S and purchase product j is

$$P_{ijS} = P_{ij|S} P_{iS} \tag{15}$$

The log-likelihood function of observing all consumers' consideration sets and their purchase decisions can be specified as

$$LL = \sum_i \log \hat{P}_{ijS} = \sum_i \log \hat{P}_{ij|S} \hat{P}_{iS} \tag{16}$$

Where $\hat{P}_{ijS}$ is the likelihood of consumer i selecting the observed consideration set and making the corresponding purchase decision. Consistent with the conventions of discrete choice demand models, the coefficients for the observable variables are identified in reference to the variability of unobserved factors. In order to achieve this, we first standardize the variance of the random utility component ε_{ij} to 1, followed by the estimation of the

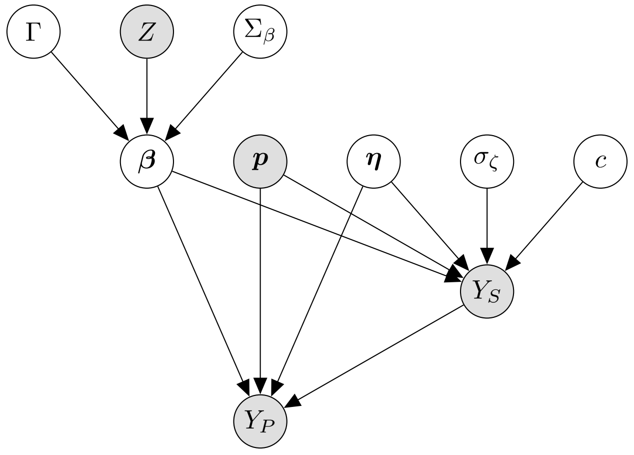


Figure 9: A Hierarchical Graph of the Non-sequential Search Model

In the graph, latent variables (e.g., parameters need to be estimated are represented by white nodes, and observed outcomes are represented by grey nodes. The directed edges represent the causal relationship between nodes. Subscripts are omitted.

coefficients for observed variables and the variance of the assessment error term σ_{zeta}.

5.3.3 Demand Estimation

We estimate the model using the Hierarchical Bayes approach (Rossi and Allenby, 2003) to account for observed and unobserved consumer heterogeneity. The parameters to estimate are: product (seller) fixed effects η_j, search cost c, the variance of the assessment error term σ_ζ^2, individual specific price sensitivity β_i, a coefficient vector that connects price sensitivity with individual characteristics Γ, the variance of the unobserved heterogeneity component Σ_β. The observables are: the price of the product p_j, consumer characteristics Z, consumers' choice sets Y_S, and consumers' purchase decisions Y_P. A hierarchy graph of all parameters and observables is shown in Figure 9.

We use Gibbs sampling to estimate the Hierarchical Bayes model. The idea of this procedure is to recursively draw each variable from the conditional distributions given other variables[13]. In each iteration, variables are drawn from the corresponding conditional distributions in the following order and are updated to form conditional distributions for other

[13]More precisely, given all other variables in its Markov blanket

variables:

$$\beta_i \mid \boldsymbol{p_i}, Y_{S_i}, Y_{P_i}, z_i, \boldsymbol{\eta}, \sigma_\zeta, c, \Gamma, \Sigma_\beta,\ \forall i \tag{17}$$

$$\Gamma \mid Z, \boldsymbol{\beta}, \Sigma_\beta \tag{18}$$

$$\Sigma_\beta \mid Z, \boldsymbol{\beta}, \Gamma \tag{19}$$

$$\boldsymbol{\eta} \mid \boldsymbol{p}, Y_S, Y_P, \boldsymbol{\beta}, \sigma_\zeta, c \tag{20}$$

$$\sigma_\zeta \mid \boldsymbol{p}, Y_S, Y_P, \boldsymbol{\eta}, \boldsymbol{\beta}, c \tag{21}$$

$$c \mid \boldsymbol{p}, Y_S, Y_P, \boldsymbol{\eta}, \boldsymbol{\beta}, \sigma_\zeta \tag{22}$$

The posterior (conditional) distribution of Γ is amultivariate normal distribution, and the posterior distribution distribution of Σ_β is an Inverse-Wishart distribution. These two parameters can be drawn directly from the corresponding posterior distributions. However, the posterior distribution of β_i, $\boldsymbol{\eta}$, σ_ζ and c does not have closed forms. Thus a Markov Chain Monte Carlo (MCMC) approach is used to obtain the draws. The exact forms of these distributions and the details of generating the draws in each iteration are specified in Appendix F. We recursively generate the draws by looping through Equation 17 to 22, until the distributions are stable. After convergence, we keep running the program for 1000 iterations to obtain statistics of interest such as mean and variance or the variables.

5.3.4 Identification and Results

Before presenting the estimation results, we explain how the parameters in our structural model can be identified.

As mentioned previously, we normalize the variance of the random utility component ε_{ij} in the utility function to 1. The remaining parameters in the consumer utility function, namely the product (seller) fixed effects (η_j) and consumer price sensitivity (β_i), can be identified from the consumers' purchase decisions conditioned on their consideration sets, which are observable in the available data. The systematic variation in purchase probabilities, conditioned on being considered, across different products help identify η_j. In our dataset, sellers on average change the price of their product 4 times during the one-month period, and the average price change is between 10% to 20%. With those price changes, $beta_i$ can be best identified if we can observe how each consumer's purchase behavior changes at different price points, as the product values remain constant over time. Although repeated purchases from the same consumers are infrequent in our data, we can leverage multiple consumers from the same demographic group and with the same characteristics.[14] The average impact of price changes on the probability of each product being purchased conditioned on being searched enable us to identify the mean price sensitivity of that group (Γz_i). The extent of variation in this effect across consumers within a demographic group helps identify the variance of the part of price sensitivity that cannot be explained by demographic characteristics (ϵ_β). Additionally, how Γz_i vary by z_i provides identification of Γ.

Once η_j and the distribution of $beta_i$ are identified, the search cost per product (c) is identified from the average size of the consideration set: a smaller average size indicates a larger value for c. Furthermore, by examining the extent of the variation in the composition of the consideration set among consumers within the same demographic group, we can identify the variance of the first-stage assessment error (σ_ζ).

The estimation results are shown in Table 7. On average, 1st party sellers have higher product (seller) fixed effects than 3rd party sellers. Average search cost is estimated to be 18.5 CNY (2.7 USD), about 25% of product prices. As mentioned earlier, the scale of ε_{ij} in the utility function is normalized to 1, and the scale parameter of the choice set–specific stochastic term (σ_ζ) is estimated to be 0.724, which suggests the impact of optimization error on the expected net benefit of the choice sets is relatively small. Consumer price sensitivity tends to decrease with age[15], while showing an increasing trend based on consumer education level (with Bachelor's Degree as the baseline group). In general, male consumers exhibit lower levels of price sensitivity.

[14](Note that in our data, demographic characteristics happen to be all categorical).

[15]Note that in the utility function, we did not include a negative sign before the price term. As a result, the greater the negativity of β_i, the higher the price sensitivity of consumer i becomes.

Variable	Notation	Coeff.	SE
Price Coefficient	Γ		
Intercept		-0.074	0.001
Age			
$\leq$ 25 years old		-0.008	0.001
36 - 45 years old		0.005	0.001
46 - 55 years old		0.007	0.001
$\geq$ 56 years old		0.008	0.002
Is Male		0.003	0.001
Is Married		0.000	0.001
Education Level			
Less Than High School		0.008	0.002
High School or Equivalent		0.010	0.001
Post-graduate Degree		-0.001	0.001
Seller Fixed Effect	η		
Seller 1		4.004	0.088
Seller 2		2.970	0.070
Seller 3		2.180	0.079
Seller 4		2.910	0.084
Seller 5		2.326	0.085
Search Cost	c	1.208	0.011
Unobserved Heterogeneity	Σ_β	0.001	0.000
Stochastic term of choice set	σ_ζ	0.751	0.009

Table 7: Estimation Results

5.3.5 Supply Side Estimation

In order to run counterfactual simulations, we need supply-side parameters, such as the marginal cost of the products. Following the common practice of the supply-side estimation, we assume sellers maximize their profit by taking into account the demand-side information and supply-side cost information. Denote seller j's marginal cost as mc_j, seller j' pricing strategy as σ_j (possibly mixed), and seller j' strategy space as $\Delta(\Phi_j)$, then a strategy profile $(\sigma_1,...,\sigma_J)$ constitutes a Nash equilibrium if it satisfies the following conditions:

$$\Pi_j(\sigma_j, \sigma_{-j}) \geq \Pi_j(\sigma_j^{'}, \sigma_{-j}), \quad \forall j, \quad \forall \sigma_j^{'} \in \Delta(\Phi_j) \tag{23}$$

where $\Pi(\cdot)$ is the profit function given all sellers' strategies. Let $\Phi_j^+ \subset \Phi_j$ denote the set of prices that seller j has set at least once (pure strategies played with positive probabilities), then the profit function can be expressed as:

$$\Pi_j(\sigma_j, \sigma_{-j}) = (p_j - mc_j)D(p_j, \sigma_{-j}), \quad \forall p_j \in \Phi_j^+ \tag{24}$$

The term $D(p_j, \sigma_{-j})$ above represents the demand for product j when the focal seller sets its price at p_j and the opponents set their prices according to strategy σ_{-j}, which can be derived from the previously estimated demand model. The conditions in Equation (23) require that $\forall j$:

$$(p_j - mc_j)D(p_j, \sigma_{-j}) = (p_j^{'} - mc_j)D(p_j^{'}, \sigma_{-j}), \quad \forall p_j, p_j^{'} \in \Phi_j^+ \tag{25}$$

$$(p_j - mc_j)D(p_j, \sigma_{-j}) \geq (p_j^{'} - mc_j)D(p_j^{'}, \sigma_{-j}), \quad \forall p_j \in \Phi_j^+, \quad \forall p_j^{'} \notin \Phi_j^+ \tag{26}$$

Since the exact formula of the demand function $D(\cdot)$ is complicated, we solve for the equilibrium prices that satisfy the above conditions numerically when estimating the marginal costs of the products.

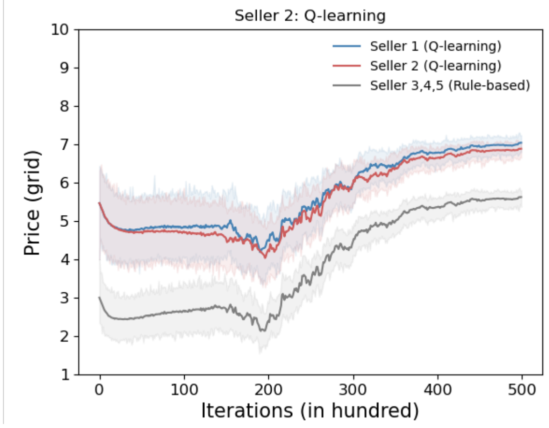
5.3.6 Counterfactual: Algorithmic Pricing

Using the estimated demand-side and supply-side parameters, we study the impact of using algorithmic pricing tools via counterfactuals. Specifically, we study two counterfactual scenarios: In counterfactual scenario 1, the seller with the highest selling volume (seller 1) uses a Q-learning pricing algorithm. The three sellers with the lowest selling volumes (seller 3, 4, and 5) use a simple rule-based pricing algorithm[16]. In counterfactual scenario 2, sellers 1 and 3 use Q-learning pricing algorithms. sellers 4 and 5 use simple rule-based algorithms. In both counterfactual scenarios, we compare seller 2's equilibrium price/profit when the seller chooses between a Q-learning algorithm and a simple rule-based algorithm.

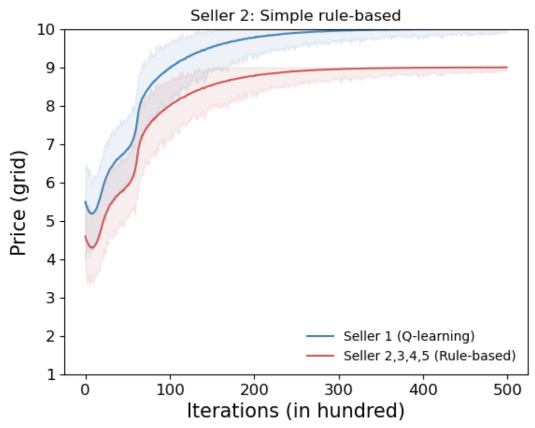
For both the simple rule-based and Q-learning pricing algorithms, we set the size of the price grids to 10 (i.e., there are 10 equally-spaced possible price levels), and these price grids are seller specific. Each seller's price grid ranges from its Nash equilibrium prices (competitive price) to its monopoly prices. When a seller uses a simple rule-based algorithm, the algorithm undercuts the lowest price of all Q-learning sellers by one price level (e.g., if Q-learning seller 1 uses its 7th price level, and Q-learning seller 2 uses its 5th price level, the simple rule-based algorithm will set price to its 4th price level). We set other hyper-parameters, including learning rate α, exploration decay rate β, and discount factor δ, to those used in the the baseline case (in Section 4). We simulate the market dynamics in each of the two counterfactual scenarios under each of Seller 2's possible algorithm choices (rule-based and Q-learning) 100 times.

The learning dynamics of the algorithms in the two scenarios are shown in Figures 10 and 11, and the mean and variance of prices and profits in the two scenarios are reported in Tables 8 and 9. In counterfactual scenario 1 (2), the focal seller's profit will increase by 4.9% (3.9%) by using a simple rule-based algorithm instead of a Q-learning algorithm. In the last column of the two tables, we also report the price and profit of our focal seller if it uses the 'optimal' simple rule-based pricing algorithm, in which the algorithm undercut competitors'

[16]Here again we use the 'Undercut Lowest Price' as a representative for simple rule-based algorithm

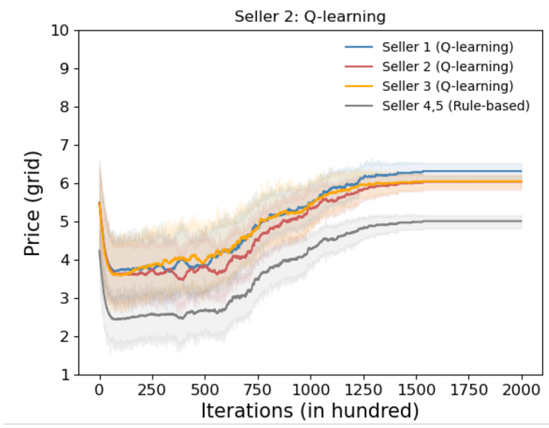


(a) Seller 2: Q-learning

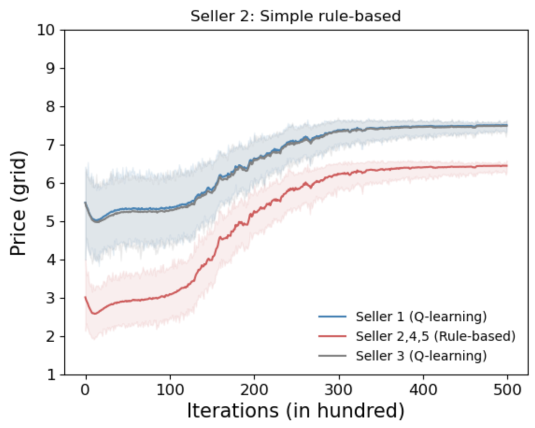


(b) Seller 2: Simple rule-based

Figure 10: Learning dynamics: Counterfactual Scenario 1



(a) Seller 2: Q-learning



(b) Seller 2: Simple rule-based

Figure 11: Learning dynamics: Counterfactual Scenario 1

prices by the number of price levels which results in the highest equilibrium profit. When using the 'optimal' rule-based strategy, the focal sellers' profit will increase by 9.02% (6.8%) in comparison to that from using a Q-learning algorithm.

In counterfactual scenario 1, seller 2, along with other sellers who employ simple rule-based algorithms, is able to drive the price of seller 1's product to near-monopoly levels. Seller 2 benefits from the relatively high average market price and consistently prices lower than its main competitor, seller 1. However, if seller 2 were to adopt the Q-learning algorithm concurrently with seller 1, the non-stationarity introduced by simultaneous exploration would

Seller 2:	**Q-learning**	**Simple rule-based**	**Rule-based(optimal)**
Price	57.6(1.2)	59.6(0.0)	56.8(0.0)
Profit	58052.9(1141.7)	60917.6(3.6)	63291.6(4.7)

Table 8: Price and Profit Comparison for Seller 2 (Counterfactual scenario 1)

Seller 2:	**Q-learning**	**Simple Rule-based**	**Rule-based(optimal)**
Price	56.9(0.6)	57.2(1.0)	55.7(1.0)
Profit	57994.8(1018.8)	60248.3(690.3)	61920.2(1212.6)

Table 9: Price and Profit Comparison for Seller 2 (Counterfactual scenario 2)

hinder both sellers from raising prices to monopoly levels, resulting in lower profits for seller 2.

In counterfactual scenario 2, the utilization of the Q-learning algorithm by both seller 1 and seller 3 creates a non-stationary environment. If seller 2 also implements the Q-learning algorithm, it would further exacerbate the non-stationarity, making it more challenging to search for a joint optimal price. This is because facing two competitors that explore randomly, as opposed to one, lowers the best response price. Consequently, the Q-learning algorithms assign higher Q-values to lower prices and lower values to high prices initially, making it difficult for the algorithm to move from low to middle or high prices. The graphical representation in Figure 11a demonstrates this issue, where initially, the prices sharply decline and struggle to rebound to the middle or high level. However, utilizing a simple rule-based algorithm by seller 2 can assist seller 1 and seller 3 in realizing the disadvantage of being stuck at a low price quickly. If they continue to charge low prices and, in addition, seller 2 charges an even lower price as a significant competitor, the two sellers would be incentivized to explore the high price region. This effect is evident in Figure 11b, where the initial price drop is much smaller than in Figure 11a, and the sellers increase the price to a much higher level in the later learning stage.

6 Theoretical Analysis

In the previous sections, we showed, through extensive simulations using the logit demand model and counterfactual analysis using an estimated demand model from real-world data, that simple rule-based algorithms provide higher profit compared to Q-learning algorithms to a seller when competing against others who use Q-learning algorithms. However, do these results hold under all conditions? To address this issue, we approach the problem theoretically and identify boundary conditions when simple rule-based algorithms would dominate Q-learning algorithms.

Modeling the RL pricing algorithm theoretically is challenging. The main source of complication is the stochastic and dynamic nature of RL algorithms. The updating process depends on the actions taken and the actions themselves may be determined by random draws. We develop a theoretical framework that captures the RL pricing dynamics to help understand how different equilibrium outcomes are reached in different competitive scenarios. In essence, the framework transforms the discrete-time learning problem into a continuous-time version, which allows us to characterize the Q-leaning dynamics using a system of differential equations. This framework enables us to study theoretically how the 'simultaneous exploration issue' affects algorithms' ability to coordinate on high prices and how stationarity brought by the rule-based algorithm solves this issue.

We put the details of the general theoretical framework in Appendix E. Here we present the result under a simple configuration. First, we limit the size of the price grid (m) to 2: $A = p_l, p_h$. Second, we assume both sellers fully explore ($\varepsilon = 1$ when $t \leq t_0$) then fully exploit ($\epsilon = 0$ when $t > t_0$). Third, we use a simple linear demand function $D_i(p_i, p_{-i}) = 1 - p_i + bp_{-i}$, where $b \in (0, 1)$ capture the cross-price elasticity of the two goods.

Considering that the size of the price grid is 2, we set p_l to the Bertrand-Nash price and

p_h to the monopoly price of the stage game:

$$p_l = \frac{1}{2-b}, \quad p_h = \frac{1}{2-2b} \tag{27}$$

The stage game payoff matrix to seller 1 is shown in Table 10, where seller 1 is the row player and seller 2 is the column player.

	p_l	p_h
p_l	$r_{ll} = \frac{1}{(b-2)^2}$	$r_{lh} = \frac{-b^2+2b-2}{2(b^3-5b^2+8b-4)}$
p_h	$r_{hl} = \frac{3b-2}{4\left(b^3-4b^2+5b-2\right)}$	$r_{hh} = \frac{1}{4(1-b)}$

Table 10: The Payoff Matrix of the Stage Game

The pricing strategies of sellers are determined by learned Q-values, and once learning has concluded, Q-values are examined to identify the equilibrium outcome. By examining the mapping of sellers' states to actions, we can identify the absorbing state(s) based on sellers' strategies. In the event that the Q-values for setting p_h are higher than those for setting p_l in any given state, the state (p_h, p_h) becomes the absorbing state, and both sellers will charge p_h in equilibrium. We put the details on the learning process (how Q-values evolve) in Appendix E and present the main result here. In the remainder of this section, we use $Q_{(p_{k_1},p_{k_2})-p_{k_0}}(t), \forall k_1, k_2, k_3 \in \{l, h\}$ denote seller's learned Q-values at time t that corresponding to charging price p_{k_0} in state (p_{k_1}, p_{k_2}).

Lemma 1 *The equilibrium outcome in the Q-Q scenario is determined by δ and b, as follows:*
1. If $\delta < \frac{b}{2-b}$, $\lim_{t\to\infty} Q_{(p_i,p_j)-p_l}(t) > \lim_{t\to\infty} Q_{(p_{k_1},p_{k_2})-p_h}(t), \forall k_1, k_2 \in \{l, h\}$, both sellers charge p_l repeatedly as $t \to \infty$.
2. If $\frac{b}{2-b} \le \delta < \sqrt{\frac{b}{2-b}}$, $\lim_{t\to\infty} Q_{(p_l,p_l)-p_l}(t) \le \lim_{t\to\infty} Q_{(p_l,p_l)-p_h}(t)$, $\lim_{t\to\infty} Q_{(p_h,p_h)-p_l}(t) > \lim_{t\to\infty} Q_{(p_h,p_h)-p_h}(t)$, both sellers alternate between p_l and p_h as $t \to \infty$.
3. If $\delta \ge \sqrt{\frac{b}{2-b}}$, $\lim_{t\to\infty} Q_{(p_l,p_l)-p_l}(t) < \lim_{t\to\infty} Q_{(p_l,p_l)-p_h}(t)$, $\lim_{t\to\infty} Q_{(p_h,p_h)-p_l}(t) \le \lim_{t\to\infty} Q_{(p_h,p_h)-p_h}(t)$, both sellers charge p_h repeatedly as $t \to \infty$.

The proof of Lemma 1 is available in Appendix E.1. Lemma 1 highlights that both algorithms will cooperate only when they attach great importance to the future (δ is high) and when the two products are not too similar (b is low). The reasoning behind this is that the focal seller will benefit more from charging p_l than p_h since, in all states, the competitor randomly chooses between p_l and p_h. As exploration decreases, both sellers frequently charge p_l, causing the Q-values assigned to p_l in state (p_l, p_l) to drop. If the Q-value for p_l never falls below that of p_h, neither seller has the incentive to increase the price, and both sellers become stuck in state (p_l, p_l). This situation arises when the initial value for p_h is extremely low, or equivalently when b is high and δ is low, making pricing above the competitor unprofitable in the short term. As soon as both sellers switch to charging p_h in state (p_l, p_l), they alternate between state (p_l, p_l) and state (p_h, p_h), and the Q-value assigned to charging p_l in state (p_h, p_h) decreases. If the Q-value for p_l drops below that of p_h in state (p_h, p_h), both sellers will learn to charge p_h repeatedly. This happens when the initial value for p_h is high, or when b is low and δ is high, making pricing above the competitor profitable in the short term, or making the seller less concerned about short-term profits.

As for the Q-R scenario, we assume seller 1 uses a Q-learning algorithm, while seller 2 uses a simple rule-based algorithm and always set a price that equals seller 1's last period price. The Q-dynamics can be derived similarly (see Appendix E for details). Using the same simple configurations as in Q-Q scenario, the final learning outcome is summarized in Lemma 2:

Lemma 2 *The equilibrium outcome in the Q-R scenario is determined by δ and b, as follows:*
1. If $\delta < \frac{1}{2}$, $\lim_{t\to\infty} Q_{(p_i,p_j)-p_l}(t) > \lim_{t\to\infty} Q_{(p_i,p_j)-p_h}(t), \forall i,j \in \{l,h\}$, both sellers charge p_l repeatedly as $t \to \infty$.
2. If $\frac{1}{2} \leq \delta$, $\lim_{t\to\infty} Q_{(p_h,p_h)-p_l}(t) \leq \lim_{t\to\infty} Q_{(p_h,p_h)-p_h}(t)$, both sellers charge p_h repeatedly as $t \to \infty$.

The proof for Lemma 2 can be found in Appendix E.2.

By comparing the learning outcomes in the Q-Q and the Q-R scenario, Theorem 1 summarizes conditions under which sellers get higher equilibrium payoff in the Q-R scenario than in the Q-Q scenario.

Theorem 1 *Both sellers get weakly higher equilibrium profit in the Q-R scenario than in the Q-Q scenario if* $\delta \geq \frac{1}{2}$.

The proof can be found in Appendix E.3. Theorem 1 tells us that when facing a competitor who adopts a Q-learning pricing algorithm, the focal seller should adopt a simple rule-based algorithm unless the discount factor is smaller than a threshold 0.5. Although this threshold may not seem too small, it should be noted that the length between time periods is also small[17]. In practice, it is highly unlikely that any sellers will encounter a discount factor smaller than this value, which is why the previous experimental results consistently show that a simple rule-based algorithm outperforms a Q-learning algorithm.

7 Discussion and Conclusion

This study investigates the issue of algorithmic pricing competition, which has gained significant attention in recent times. Specifically, this paper aims to answer a crucial question: what kind of pricing algorithm should a seller use when competing with opponents who employ sophisticated learning algorithms? Our findings indicate that a simple rule-based algorithm can remarkably increase the seller's profit, even in the presence of competitors using sophisticated learning algorithms. Notably, implementing complex learning algorithms on online marketplaces can be both costly and risky, as it entails a prolonged exploration period, during which losses are likely to occur. Such a scenario may pose significant challenges for small sellers, who may not be well-equipped to bear the risks, thus raising concerns that they may be at a disadvantage compared to the pre-algorithmic pricing era. Nevertheless, our results suggest that such concerns are largely unfounded, as the use of a simple rule-based algorithm can yield better outcomes for a seller than utilizing a sophisticated learning

[17]For example, if the length between time periods is 4 hours, it transfers to a daily discount factor of 0.015.

algorithm when competing with competitors using advanced algorithms.

In the legal and policymaking spheres, algorithmic tacit collusion has been a topic of intense discussion since 2015, with policymakers expressing concern that independent learning algorithms have the potential to charge supra-competitive prices even in the absence of communication and coordination among sellers. Our analysis contradicts the conventional wisdom that more advanced algorithms are better equipped to collude and sustain higher prices. Instead, we find that the use of a sophisticated learning algorithm by one seller, in conjunction with simple rule-based algorithms used by others, can result in the sustainment of higher prices. Policymakers concerned with algorithmic collusion should, therefore, direct their attention toward this specific scenario, as it presents a unique challenge that demands tailored interventions.

A Experiment Outcomes: Alternative Settings

A.1 More Advanced RL Algorithms

We examine whether our results are robust when we replace the Q-learning algorithm with two types of more advanced RL algorithms: *Deep Q Network* (DQN) and *Advanced Actor-Critic Method* (A2C). Specifically, we let seller 1 use an advanced RL algorithm, and compare seller 2's equilibrium price and profit when using the same advanced RL algorithm vs using a rule-based algorithm to compete. For each advanced algorithm (DQN or A2C), we run the experiment 100 times and report the mean and standard deviation of equilibrium price and profit in Table 11. The result shows that our main result holds regardless of which specific type of reinforcement learning algorithm is used.

Panel A: Deep Q Network (DQN)		
	DQN	Rule-based Algorithm
Price	1.680(0.024)	1.894(0.011)
Profit 2	0.291(0.015)	0.335(0.002)
Panel B: Advanced Actor-Critic Method (A2C)		
	A2C	Rule-based Algorithm
Price	1.586(0.043)	1.844(0.050)
Profit	0.265(0.019)	0.356(0.013)

Table 11: Simple Rule-based Algorithm Competes with More Advanced Reinforcement Learning Algorithms (DQN and A2C)

A.2 Multiple Players in the Market

In this section, we extend our baseline model to an oligopoly setting ($N > 2$). We study the case where there are three sellers in the market and examine when seller 1 and seller 2 use Q-learning pricing algorithms, how would seller 3's equilibrium price and equilibrium profit differ when it decides between a Q-learning algorithm and a simple rule-based algorithm.

In this extension, we compare seller 3's equilibrium price and equilibrium profit across 4 scenarios: when it uses 1) the same Q-learning algorithm as sellers 1 and 2, 2) a simple rule-based algorithm that undercuts seller 1's price by one price grid, 3) a simple rule-based algorithm that undercuts seller 2's price by one price grid, and 4), a simple rule-based algorithm that undercuts the lowest price (between sellers 1 and 2) by one price grid.

We run each experiment 100 times and the outcomes are reported in Table 12.

	Q-learning	Undercut seller 1	Undercut seller 2	Undercut the lowest
Price	1.464(0.030)	1.663(0.059)	1.667(0.069)	1.654(0.055)
Profit	0.148(0.009)	0.175(0.015)	0.172(0.017)	0.202(0.014)

Table 12: Equilibrium Outcomes When Seller 3 Uses Different Algorithm

B SKU Subset Selection Details

Unfortunately, we cannot see the detailed product information from the data set so we cannot manually select a subset of close substitutes. Instead, we use consumers' clicking data to infer the subsets which contain close substitutes: if two products are close substitutes, then consumers are very likely to click on both of them to compare. We say two products are 'co-clicked' by a consumer if this consumer clicks on both products within a relatively short period of time (for example, 5 hours). If two products are 'co-clicked' by many consumers, then these two products are very likely to be close substitutes. With this idea, we first create an un-directed weighted graph using the 2.5 million click events in the *Click Table* to describe the 'co-click' relationship among all SKUs. Specifically, each node in the graph represents an SKU, the weight of each edge in the graph represents the number of consumers who have co-clicked on these two SKUs. The adjacency matrix of this graph turns out to be very sparse which means that most SKU pairs are never 'co-clicked' by any consumer. Next, we would like to find a subset of SKUs within which all SKUs are close substitutes (i.e., the weight of all edges connecting these nodes are high). In graph theory, such activity is called 'community detection', which means detecting local communities in a graph that each member in it has a close relationship with others (The SKUs within a community are more densely connected than the SKUs across communities). Considering the size of our graph, we adopt the *Louvain method for community detection* (Blondel et al., 2008) to detect local communities which have algorithm complexity of $O(n \log n)$. One potential issue of this method is that if two products are complementary instead of substitutes, they will also be included in the same cluster. Later in the paper, we rule out this possibility by showing that very few consumers end up purchasing more than one product.

C Testing Search Models

Metrics I: Recall

Under the assumption of sequential search, a consumer starts from the most potential products (i.e., the one that gives the highest expected utility). If the realized utility of a product after searching is high, she will stop searching and make the purchase. If the opposite, she will keep on the searching process. However, she will never recall a product that she searched for earlier unless there are a finite number of products unless the consumer has searched for all options. Instead, under the non-sequential search model, this needs not to be the case since the search sequence is purely random. We examine the recall behavior for all the consumers in our data set and look at the percentage of consumers who has recall behavior. It turns out that among the consumers who search for more than one product but don't search for all 5 products, 58% of them have recall behavior.

Metrics II: Conditioned Probability of Purchase

Under the assumption of sequential search, a consumer continues the searching process if she finds the current product unsatisfied. Otherwise, she will stop searching and make the purchase. Consequently, consumers will purchase the latter searched product more often than the earlier searched product, even if they always start from the most preferred one. However, we find that the probability of being purchased for the earlier searched product is not significantly smaller than the purchase probability for the latter searched product.

The above results empirically support the non-sequential search model in this context. Thus we construct a structural model under the non-sequential search assumption to describe the market environment.

D Results for an Alternative SKU subset

The estimation results are shown in Table 13:

Variable	Notation	Coeff.	SE
Price Coefficient	Γ		
Intercept		-0.050	0.000
Age			
$\leq$ 25 years old		0.003	0.000
36 - 45 years old		0.002	0.000
46 - 55 years old		-0.006	0.001
$\geq$ 56 years old		0.001	0.001
Is Male		-0.001	0.001
Is Married		-0.001	0.001
Education Level			
Less Than High School		-0.006	0.001
High School or Equivalent		-0.003	0.000
Post-graduate Degree		-0.001	0.001
Seller Fixed Effect	η		
Seller 1		2.896	0.097
Seller 2		3.013	0.093
Seller 3		3.504	0.093
Seller 4		4.670	0.187
Seller 5		3.667	0.142
Seller 6		0.353	0.014
Seller 7		1.366	0.067
Seller 8		3.432	0.071
Search Cost	c	1.770	0.015
Unobserved Heterogeneity	Σ_β	0.001	0.000
Stochastic term of choice set	σ_ζ	0.811	0.008

Table 13: Estimation Results

E Theoretical Framework Details

We study the competition of algorithms in oligopoly pricing, which is set within a repeated game framework. The market consists of n differentiated products, each sold by one of n sellers. Without any capacity constraints, each seller offers their own product in every period. Sellers concurrently determine the prices of their respective products in each period, and previous price history is common knowledge. We assume sellers use a size m discrete price grid and have one-period memory, and we restrict our attention to the case $n = 2$. Specifically, the action space for the Q-learning algorithm is $A = \{p_i\}, i \in \{0, .., m-1\}$. The state contains the last period prices of all sellers $S = \{s_{ij} = (p_i, p_j)\}, i \in \{0, .., m-1\}, j \in \{0, .., m-1\}$.

In essence, the procedure of learning of the algorithm is the procedure of updating its Q-table. The size of the Q-table is $\|A\| \times \|S\| = m \times m^2$, that is, for each state, there are m Q-values representing the value of taking each of the m actions. We start with the Q-Q scenario. Denote p_k^t as the price charged by seller k at time t, respectively. Denote s^t as the state in time t, where $s^t = (p_1^{t-1}, p_2^{t-1})$. Seller k receives a reward $r_k(p_1^t, p_2^t)$, and the state transits to $s^{t+1} = (p_1^t, p_2^t)$ in the next time period. We focus on the case where the two sellers are symmetric and we derive everything from seller 1's perspective. We denote the seller 1's Q-value at time t that corresponds to taking action $p_{i'}$ in state (p_i, p_j) as $Q^1_{(p_i,p_j)-p_{i'}(t)}$ According to the Q-learning updating rule, at time t, one of the Q-values in the Q-table, $Q^1_{(p_1^{t-1},p_2^{t-1})-p_1^t}(t)$ will be updated as follows:

$$Q^1_{(p_1^{t-1},p_2^{t-1})-p_1^t}(t+1) = (1-\alpha)Q^1_{(p_1^{t-1},p_2^{t-1})-p_1^t}(t) + \alpha(r_1(p_1^t, p_2^t) + \delta \max_{p_i'}(Q^1_{(p_1^t,p_2^t)-p_i'})) \tag{E.1}$$

Where δ is the discount factor and α is the learning rate. From the above, we can see that the reward and state transition depend on both sellers' actions, which are not always deterministic (e.g., random exploration is needed in the early stage of learning). We are interested in how the Q-value evolves *in expectation.* We now re-formalize Equation E.1 using the two sellers' *probabilities of taking each action* instead of the actions that they *actually take.* Assume that given state $s = (p_i, p_j)$, seller k takes action $a_k = p_{i'}$ with probability $x^k_{(p_i,p_j)-p_{i'}}, \forall p_{i'} \in A$. Denote the probability that the state (p_i, p_j) appears as $P_{(p_i,p_j)}$, then $Q^k_{(p_i,p_j)-p_{i'}}$ will be updated *in expectation* in the following way:

$$\begin{aligned} Q^1_{(p_i,p_j)-p_{i'}}(t+1) = (1-\alpha)Q^1_{(p_i,p_j)-p_{i'}}(t) + \alpha P_{(p_i,p_j)} x^1_{(p_i,p_j)-p_{i'}} \Sigma_{j'}(x^2_{(p_i,p_j)-p_{j'}} r(p_{i'}, p_{j'})) \\ + \delta \Sigma_{j'}(x^2_{(p_i,p_j)-p_{j'}} \max_{p_{i''}}(Q^1_{(p_{i'},p_{j'})-p_{i''}})) \end{aligned} \tag{E.2}$$

The basic logic for Equation E.2 is: Q-values are state and action specific, and we want to weight the learning rate of a Q-value that corresponds to a certain state and a certain action with the probability that this specific state occurs and the specific action is taken. In essence, for the state that is frequently visited and the action that is frequently taken, the corresponding Q-value will be updated faster than those less frequently visited states and actions.

Now we transform the discrete-time learning problem into a continuous time version by compressing the time interval between two consecutive time periods from 1 to Δt. As Δt approaches 0, the Q-leaning dynamics can be captured by the following system of differential equations:

$$\begin{cases} \frac{dQ^1_{(p_i,p_j)-p_{i'}}(t)}{dt} & = \alpha P_{(p_i,p_j)} x^1_{(p_i,p_j)-p_{i'}} (\Sigma_{j'}(x^2_{(p_i,p_j)-p_{j'}} r_1(p_{i'}, p_{j'}) \\ & +\delta \Sigma_{j'}(x^2_{(p_i,p_j)-p_{j'}} \max_{i''}(Q^1_{(p_{i'},p_{j'})-p_{i''}})) \qquad \forall i, j, i' \in \{1, ..m\} \\ & -Q^1_{(p_i,p_j)-p_{i'}}(t)) \end{cases} \tag{E.3}$$

where $x^k_{(p_i,p_j)-p_{i'}}, \forall k \in \{1, 2\}$ are determined by the current Q-values and the exploration strategy. In particular, we use the ϵ greedy exploration strategy here, where the probability of taking a greedy action (the action with the largest Q-value) given a state is $1-\epsilon$ and the probability for taking any other actions is ϵ. Then we have:

$$x^k_{(p_i,p_j)-p_{i'}} = \begin{cases} \epsilon & \text{if} \quad p_{i'} = \arg\max_p Q^k_{(p_i,p_j)-p} \\ \frac{\epsilon}{(m-1)} & \text{otherwise} \end{cases} \quad \forall i, j, i' \in \{1, ..m\}, \forall k \in \{1, 2\} \tag{E.4}$$

$P_{(p_i,p_j)}$ is the frequency that state (p_i, p_j) is visited under sellers' current strategies, and can be determined by the following system of linear equations

$$\begin{cases} P_{(p_i,p_j)} = \Sigma_{i'j'} P_{(p_{i'},p_{j'})} x^1_{(p_{i'},p_{j'})-p_i} x^2_{(p_{i'},p_{j'})-p_j} & \forall i,j \in \{1,..m\} \\ \Sigma_{i'j'} P_{(p_{i'},p_{j'})} = 1 \end{cases} \tag{E.5}$$

The complete Q-dynamics can be found by solving Equations E.3 to E.5.

E.1 Learning Dynamics of The Q-Q scenario (Proof of Lemma 1)

E.1.1 Learning Stage 0: Initialization and Random Exploration

we initialize the Q-values using the discounted payoff that would arise for the focal seller if the opponent seller chooses the price randomly, which is also the Q-values that the Q-learning algorithm will learn when both sellers fully explore. For notation simplicity, we use $Q_{ij-k}(t)$ to denote seller 1's Q-value of taking action p_k in state (p_i, p_j), $\forall i, j, k \in \{l, h\}$.

$$\begin{aligned} Q_{ll-l}(0) = Q_{hh-l}(0) = Q_{lh-l}(0) = Q_{hl-l}(0) &= \frac{r_{ll} + r_{lh}}{2(1-\delta)} \\ Q_{ll-h}(0) = Q_{hh-h}(0) = Q_{lh-h}(0) = Q_{hl-h}(0) &= \frac{\delta(r_{ll} + r_{lh})}{2(1-\delta)} + \frac{r_{hl} + r_{hh}}{2} \end{aligned} \tag{E.6}$$

Since our initialization is consistent with the Q-values that the Q-learning algorithm will actually learn, the Q-values don't change in this stage. At the end of this stage, both sellers' greedy strategies are to play p_l at all of the states.

E.1.2 Learning Stage 1: The Drop of Q_{ll-l}

After the exploration is shut down at $t = t_0$, both sellers start to take their greedy actions: play p_l repeatedly. In this stage, only the stage (p_l, p_l) is visited so only Q_{ll-l} is updated. According to Equation E.1, the Q-dynamic for Q_{ll-l} in this stage is:

$$\frac{dQ_{ll-l}(t)}{dt} = \alpha(r_{ll} + \delta Q_{ll-l}(t) - Q_{ll-l}(t)) \tag{E.7}$$

With initial condition:

$$Q_{ll-l}(t_0) = \frac{r_{ll} + r_{lh}}{2(1-\delta)} \tag{E.8}$$

Therefore,

$$Q_{ll-l}(t) = \frac{r_{lh} - r_{ll}}{2(1-\delta)} \exp(-\alpha(t-t_0)(1-\delta)) + \frac{r_{ll}}{1-\delta} \tag{E.9}$$

It's straightforward to check that $Q_{ll-l}(t)$ decreases in t, the intuition is that: the Q-values learned in stage 0 is for the case when the opponent seller randomly pick between p_l and p_h, however, in stage 1, the opponent seller becomes more aggressive in setting the price (it charges p_l repeatedly). Thus, the Q values learned in stage 0 overestimate the true Q-values that appear in stage 1. As new experiences are gathered, the Q-values are updated downwards.

A sufficient condition that the two sellers are trapped in the (p_l, p_l) equilibrium is:

$$\lim_{t\to\infty} Q_{ll-l}(t) > Q_{ll-h}(t_0) \tag{E.10}$$

The intuition is that if the Q-value for taking p_l in state (p_l, p_l) never drops below the Q-value for taking p_h in state (p_l, p_l), the two sellers will never reach to other states in the first place, thus they will end up in state (p_l, p_l) forever. Take Equations E.19 and E.6 in Equation E.10, we get the sufficient condition for equilibrium outcome (p_l, p_l):

$$\delta < \frac{b}{2-b} \tag{E.11}$$

E.1.3 Learning Stage 2: The Drop of Q_{hh-l}

If Condition E.11 is not met, Q_{ll-l} will drop below Q_{ll-h} at t_1, the two sellers begins to charge p_h in state (p_l, p_l) but charge p_l in state (p_h, p_h). Thus the state will jump between (p_l, p_l) and (p_h, p_h), the two Q-values, Q_{ll-h} and Q_{hh-l} will be updated at the same time.

The Q-dynamics for these two Q-values are:

$$\begin{aligned}\frac{dQ_{ll-h}(t)}{dt} &= \frac{\alpha}{2}(r_{hh} + \delta Q_{hh-l}(t) - Q_{ll-h}(t)) \\ \frac{dQ_{hh-l}(t)}{dt} &= \frac{\alpha}{2}(r_{ll} + \delta Q_{ll-h}(t) - Q_{hh-l}(t))\end{aligned} \tag{E.12}$$

With initial conditions:

$$\begin{aligned}Q_{ll-h}(t_1) &= \frac{\delta(r_{ll} + r_{lh})}{2(1-\delta)} + \frac{r_{hl} + r_{hh}}{2} \\ Q_{hh-l}(t_1) &= \frac{r_{ll} + r_{lh}}{2(1-\delta)}\end{aligned} \tag{E.13}$$

The solutions turn out to be:

$$\begin{aligned}Q_{ll-h}(t) =& (((1+\delta)(1-\delta)(r_{ll} + r_{lh} - r_{hh} - r_{hl}) + 2(1-\delta)(r_{hh} - r_{ll}))\exp(\frac{-(t-t_1)(1+\delta)}{2}) \\ &+ (1+\delta)((r_{ll} + r_{lh} - r_{hh} - r_{hl})(1-\delta) + 2(r_{hh} - r_{lh}))\exp(\frac{-(t-t_1)(1-\delta)}{2}) \\ &- 4(r_{hh} + r_{ll}\delta))/(4(\delta-1)(\delta+1)) \\ Q_{hh-l}(t) =& -((1+\delta)(1-\delta)(r_{ll} + r_{lh} - r_{hh} - r_{hl}) + 2(1-\delta)(r_{hh} - r_{ll}))\exp(\frac{-(t-t_1)(1+\delta)}{2}) \\ &+ (1+\delta)((r_{ll} + r_{lh} - r_{hh} - r_{hl})(1-\delta) + 2(r_{hh} - r_{lh}))\exp(\frac{-(t-t_1)(1-\delta)}{2} \\ &- 4(r_{ll} + r_{hh}\delta)(4(\delta-1)(\delta+1))\end{aligned} \tag{E.14}$$

A sufficient condition that the two sellers end up with alternating between (p_l, p_l) and (p_h, p_h) in equilibrium is:

$$\begin{aligned}&\lim_{t\to\infty} Q_{ll-h}(t) > Q_{ll-l}(t_1)\\&\lim_{t\to\infty} Q_{hh-l}(t) > Q_{hh-h}(t_1)\end{aligned} \tag{E.15}$$

Take Equations E.14 and E.6 in Equation E.15, we get the sufficient condition for equilibrium outcome $(p_l, p_l) - (p_h, p_h)$:

$$\frac{b}{2-b} \leq \delta < \sqrt{\frac{b}{2-b}} \tag{E.16}$$

E.1.4 Learning Stage 3: Rise of Q_{hh-h}

If neither Condition E.11 nor E.16 is met, Q_{hh-l} will drop below Q_{hh-h} at t_2, the two sellers switch to charge p_h in state (p_h, p_h). Then only Q_{hh-h} will be updated afterward. The Q-dynamics can be shown:

$$\frac{dQ_{hh-h}(t)}{dt} = \alpha(r_{hh} + \delta Q_{hh-h}(t) - Q_{hh-h}(t)) \tag{E.17}$$

With initial condition:

$$Q_{hh-h}(t_2) = \frac{\delta(r_{ll} + r_{lh})}{2(1-\delta)} + \frac{r_{hl} + r_{hh}}{2} \tag{E.18}$$

Therefore the solution:

$$Q_{hh-h}(t) = \frac{r_{hh}}{1-\delta} - \frac{r_{hh} - r_{hl} + \delta(r_{hh} - r_{ll} + r_{hl} - r_{lh})}{2(1-\delta)} \exp(-\alpha(t-t_2)(1-\delta)) \tag{E.19}$$

It's straightforward to check that $\lim_{t\to\infty} Q_{hh-h}(t) = \frac{r_{hh}}{1-\delta} > Q_{hh-l}(t_2)$, Therefore, a sufficient condition for the (p_h, p_h) equilibrium is:

$$\delta \geq \sqrt{\frac{b}{2-b}} \tag{E.20}$$

E.2 Learning Dynamics of The Q-R scenario (Proof of Lemma 2)

The Q-learning dynamic can be found by solving Equations E.21 to E.23.

$$\begin{cases} \frac{dQ^R_{(p_i,p_j)-p_{i'}}(t)}{dt} & = \alpha P_{(p_i,p_j)} x^R_{(p_i,p_j)-p_{i'}} (r(p_{i'}, p_{\max(i-1,1)}) \\ & +\delta \max_{i''}(Q^R_{(p_{i'},p_{\max(i-1,1)})-p_{i''}})) \qquad \forall i,j,i' \in \{1,..m\} \\ & -Q^R_{(p_i,p_j)-p_{i'}}(t)) \end{cases} \tag{E.21}$$

$$x^R_{(p_i,p_j)-p_{i'}} = \begin{cases} \epsilon & \text{if} \quad p_{i'} = \arg\max_p Q^R_{(p_i,p_j)-p} \\ \frac{\epsilon}{(m-1)} & \text{otherwise} \end{cases} \qquad \forall i,j,i' \in \{1,..m\} \tag{E.22}$$

$$\begin{cases} P_{(p_i,p_j)} = \Sigma_{i'j'} P_{(p_{i'},p_{j'})} x^R_{(p_{i'},p_{j'})-p_i} \mathbb{1}\{p_j = p_{\max(i'-1,1)}\} & \forall i,j \in \{1,..m\} \\ \Sigma_{i'j'} P_{(p_{i'},p_{j'})} = 1 \end{cases} \tag{E.23}$$

However, finding the learning outcome in the Q-R scenario is much simpler. Since seller 2's pricing strategy does not alter, the environment is stationary for seller 1, and the Q-learning algorithm is guaranteed to find the optimal pricing policy. The eventual Q-values can be found by solving the following system of equations:

$$\begin{aligned}
Q_{ll-l} &= r_{ll} + \delta \max(Q_{ll-l} + Q_{ll-h}) \\
Q_{ll-h} &= r_{hl} + \delta \max(Q_{hh-l} + Q_{hh-h}) \\
Q_{hh-l} &= r_{lh} + \delta \max(Q_{ll-l} + Q_{ll-h}) \\
Q_{hh-h} &= r_{hh} + \delta \max(Q_{hh-l} + Q_{hh-h})
\end{aligned} \tag{E.24}$$

Depending on the sign of $Q_{ll-l} - Q_{ll-h}$ and $Q_{hh-l} + Q_{hh-h}$, we get 3 types of outcomes: When $\delta < 0.5$, $Q_{ll-l} > Q_{ll-h}$,$Q_{hh-l} > Q_{hh-h}$ and (p_l, p_l) will be the equilibrium outcome. When $0.5 \leq \delta < \frac{1}{2(1-b)}$, $Q_{ll-l} > Q_{ll-h}$,$Q_{hh-l} \leq Q_{hh-h}$ and both (p_h, p_h) and (p_l, p_l) will be the equilibrium outcome, according to our equilibrium selection criteria, (p_h, p_h) is selected since it Pareto dominates (p_l, p_l). When $\delta \geq \frac{1}{2(1-b)}$, $Q_{ll-l} \leq Q_{ll-h}$,$Q_{hh-l} < Q_{hh-h}$ and (p_h, p_h) will be the equilibrium outcome.

E.3 Proof of Theorem 1

The proof is straightforward by comparing Lemma 1 and Lemma 2 thus is omitted here.

F A Hierarchical Bayes Estimation Algorithm

Following (Netzer et al., 2008), we specify our Hierarchical Bayes estimation algorithm details in this section. All vectors and matrices are bold. Subscript i denotes consumer i, and N is the total number of consumers.

In each iteration, variables are drawn from corresponding conditional distributions in the following order and are updated to form subsequent conditional distributions for other variables:

$$\begin{aligned}
&\beta_i \mid p, Y_{S_i}, Y_{P_i}, z_i, \eta, \sigma_\zeta, c, \Gamma, \Sigma_\beta \\
&\Gamma \mid Z, \beta, \Sigma_\beta \\
&\Sigma_\beta \mid Z, \beta, \Gamma \\
&\eta \mid p, Y_S, Y_P, \beta, \sigma_\zeta, c \\
&\sigma_\zeta \mid p, Y_S, Y_P, \eta, \beta, c \\
&c \mid p, Y_S, Y_P, \eta, \beta, \sigma_\zeta
\end{aligned}$$

(1) Generate $\beta_i, \forall i$

$$\begin{aligned}
&f(\beta_i \mid p_i, Y_{S_i}, Y_{P_i}, z_i, \eta, \sigma_c, c, \Gamma, \Sigma_\beta) \\
&\propto \mathcal{N}(\beta_i \mid \eta, \sigma_c, c, \Gamma, \Sigma_\beta) L(Y_{S_i}, Y_{P_i}) \qquad \text{(F.1)} \\
&\propto |\Sigma_\beta|^{-1/2} \exp(-1/2(\beta_i - \Gamma' z_i)' \Sigma_\beta^{-1} (\beta_i - \Gamma' z_i)) L(Y_{S_i}, Y_{P_i})
\end{aligned}$$

Where $L(Y_{S_i}, Y_{P_i})$ is the likelihood function from Equation 16. We use the Metropolis-Hasting algorithm to draw new β_i from its conditional distribution. In each iteration, a new vector value β_i^{new} is drawn from a multivariate normal distribution $\mathcal{N}(\beta_i, V_D^2)$ where V_D^2 is chosen adaptively, and we accept the generated new draw with probability

$$Pr = \min\{\frac{\exp[-1/2(\beta_i^{new} - \Gamma' z_i)' \Sigma_\beta^{-1} (\beta_i^{new} - \Gamma' z_i)] L(Y_{S_i}, Y_{P_i} | \beta_i^{new})}{\exp[-1/2(\beta_i - \Gamma' z_i)' \Sigma_\beta^{-1} (\beta_i - \Gamma' z_i)] L(Y_{S_i}, Y_{P_i} | \beta_i)}, 1\} \qquad \text{(F.2)}$$

(2) Generate Γ

The vectorized Γ (denoted as $v\Gamma$) follows a multivariate normal distribution

$$v\Gamma \mid Z, \beta, \Sigma_\beta = \mathcal{N}(u_n, V_n) \qquad \text{(F.3)}$$

Where

$V_n = [(Z^{'}Z \otimes \Sigma_{\beta}^{-1}) + V_0^{-1}]^{-1}$,

$u_n = V_n[Z^{'} \otimes \Sigma_{\beta}^{-1}\beta^{*} + V_0^{-1}u_0]$,

$Z = (z_1^{'}, z_2^{'}, ..., z_N^{'})$ is an $N \times nz$ matrix,

$\beta^{o} = (\beta_1^{'}, \beta_2^{'}, ..., \beta_N^{'})$ is an $N \times n\beta$ matrix,

$\beta^{*} = vec(\beta^{o})$,

$nz = \dim(z_i)$,

$n\beta = \dim(\beta_i)$,

We set prior hyperparameters $u_0 = \{0\}_{n\beta \cdot nz}$, and $V_0 = 100 I_{n\beta \cdot nz}$.

(3) Generate Σ_β

$$\Sigma_{\beta} \mid Z, \beta, \Gamma \sim \mathcal{W}_{n\beta}^{-1}(f_0 + N, G_0^{-1} + \Sigma_{i=1}^{N}(\beta_i - \Gamma^{'}Z_i)^{'}(\beta_i - \Gamma^{'}Z_i)) \tag{F.4}$$

Where $\mathcal{W}^{-1}$ is the inverse Wishart distribution. $f_0 = n\beta + 5$ and $G_0 = I_{n\beta}$ are prior hyperparameters.

(4) Generate η

The procedure of generating η, σ_ζ, and c is similar to the procedure of generating β_i, where the Metropolis-Hasting algorithm is used to generate draws from the specified probability distribution. We use diffused priors instead of having prior distribution specified by other higher-level variables.

$$\begin{aligned} & f(\eta \mid p, Y_S, Y_P, \beta, \sigma_\zeta, c) \\ & \propto \mathcal{N}(\eta_0, V_{\eta 0}) L(Y_S, Y_P) \\ & \propto |V_{\eta 0}|^{-1/2} \exp(-1/2(\eta - \eta_0)^{'} V_{\eta 0}^{-1}(\eta - \eta_0)) L(Y_S, Y_P) \end{aligned} \tag{F.5}$$

In each iteration, a new draw η^{new} is generated from the multivariate normal distribution $\mathcal{N}(\eta, V_\eta)$ and will be accepted with the following acceptance probability

$$Pr = \min\{\frac{\exp(-1/2(\eta^{new} - \eta_0)' V_{\eta_0}^{-1}(\eta^{new} - \eta_0))L(Y_S, Y_P|\eta^{new})}{\exp(-1/2(\eta - \eta_0)' V_{\eta_0}^{-1}(\eta - \eta_0))L(Y_S, Y_P|\eta)}, 1\} \tag{F.6}$$

The diffusion prior hyperparameters are set as follows: $\eta_0 = \{0\}_{ns}$, $V_{\eta_0} = 100 I_{ns}$.

(5) Generate σ_ζ

$$\begin{aligned} & f(\sigma_\zeta \mid p, Y_S, Y_P, \eta, \beta, c) \\ & \propto \mathcal{N}(\sigma_{\zeta_0}, V_{\sigma_{\zeta_0}}) L(Y_S, Y_P) \\ & \propto |V_{\sigma_{\zeta_0}}|^{-1/2} \exp(-1/2(\sigma_\zeta - \sigma_{\zeta_0})' V_{\sigma_{\zeta_0}}^{-1}(\sigma_\zeta - \sigma_{\zeta_0})) L(Y_S, Y_P) \end{aligned} \tag{F.7}$$

In each iteration, a new draw σ_ζ^{new} is generated from a normal distribution $\mathcal{N}(\sigma_\zeta, V_{\sigma_\zeta})$ and will be accepted with the following acceptance probability

$$Pr = \min\{\frac{\exp(-1/2(\sigma_\zeta^{new} - \sigma_{\zeta_0})' V_{\sigma_{\zeta_0}}^{-1}(\sigma_\zeta^{new} - \sigma_{\zeta_0}))L(Y_S, Y_P|\sigma_\zeta^{new})}{\exp(-1/2(\sigma_\zeta - \sigma_{\zeta_0})' V_{\sigma_{\zeta_0}}^{-1}(\sigma_\zeta - \sigma_{\zeta_0}))L(Y_S, Y_P|\sigma_\zeta)}, 1\} \tag{F.8}$$

The diffusion prior hyperparameters are set as follows: $\sigma_{\zeta_0} = 0$, $V_{\sigma_{\zeta_0}} = 30$.

(5) Generate c

$$\begin{aligned} & f(c \mid p, Y_S, Y_P, \eta, \beta, \sigma_\zeta) \\ & \propto \mathcal{N}(c_0, V_{c_0}) L(Y_S, Y_P) \\ & \propto |V_{c_0}|^{-1/2} \exp(-1/2(c - c_0)' V_{c_0}^{-1}(c - c_0)) L(Y_S, Y_P) \end{aligned} \tag{F.9}$$

In each iteration, a new draw c^{new} is generated from a normal distribution $\mathcal{N}(c, V_c)$ and will be accepted with the following acceptance probability

$$Pr = \min\{\frac{\exp(-1/2(c^{new} - c_0)' V_{c_0}^{-1}(c^{new} - c_0))L(Y_S, Y_P|c^{new})}{\exp(-1/2(c - c_0)' V_{c_0}^{-1}(c - c_0))L(Y_S, Y_P|c)}, 1\} \tag{F.10}$$

The diffusion prior hyperparameters are set as follows: $c_0 = 0$, $V_{c_0} = 30$.

References

N. Aramayo, M. Schiappacasse, and M. Goic. A multiarmed bandit approach for house ads recommendations. *Marketing Science*, 2022.

J. Asker, C. Fershtman, A. Pakes, et al. Artificial intelligence, algorithm design and pricing. In *AEA Papers and Proceedings*, volume 112, pages 452–56. American Economic Association, 2022.

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

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10): P10008, 2008.

Z. Y. Brown and A. MacKay. Competition in pricing algorithms. Technical report, National Bureau of Economic Research, 2021.

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

L. Chen, A. Mislove, and C. Wilson. An empirical analysis of algorithmic pricing on amazon marketplace. In *Proceedings of the 25th international conference on World Wide Web*, pages 1339–1349, 2016.

B. De Los Santos, A. Hortaçsu, and M. R. Wildenbeest. Testing models of consumer search using data on web browsing and purchasing behavior. *American Economic Review*, 102 (6):2955–80, May 2012. doi: 10.1257/aer.102.6.2955.

K. T. Hansen, K. Misra, and M. M. Pai. Frontiers: Algorithmic collusion: Supra-competitive prices via independent algorithms. *Marketing Science*, 40(1):1–12, 2021. doi: 10.1287/mksc.2020.1276.

E. Honka. Quantifying search and switching costs in the us auto insurance industry. *The RAND Journal of Economics*, 45(4):847–884, 2014. doi: https://doi.org/10.1111/1756-2171.12073.

J. Johnson, A. Rhodes, and M. R. Wildenbeest. Platform design when sellers use pricing algorithms. *Available at SSRN 3753903*, 2020.

T. Klein. Autonomous algorithmic collusion: Q-learning under sequential pricing. *The RAND Journal of Economics*, 52(3):538–558, 2021.

J. Liu, Y. Zhang, X. Wang, Y. Deng, and X. Wu. Dynamic pricing on e-commerce platform with deep reinforcement learning: A field experiment, 2019.

X. Liu. Dynamic coupon targeting using batch deep reinforcement learning: An application to livestream shopping. *Marketing Science*, 2022.

J. J. McCall. Economics of information and job search. *The Quarterly Journal of Economics*, 84(1):113–126, 1970. ISSN 00335533, 15314650.

N. Mehta, S. Rajiv, and K. Srinivasan. Price uncertainty and consumer search: A structural model of consideration set formation. *Marketing Science*, 22(1):58–84, 2003. doi: 10.1287/mksc.22.1.58.12849.

J. Miklós-Thal and C. Tucker. Collusion by algorithm: Does better demand prediction facilitate coordination between sellers? *Management Science*, 65(4):1552–1561, 2019.

O. Netzer, J. M. Lattin, and V. Srinivasan. A hidden markov model of customer relationship dynamics. *Marketing Science*, 27(2):185–204, 2008. doi: 10.1287/mksc.1070.0294. URL `https://doi.org/10.1287/mksc.1070.0294`.

J. O'Connor and N. E. Wilson. Reduced demand uncertainty and the sustainability of collusion: How ai could affect competition. *Information Economics and Policy*, 54:100882, 2021.

O. Rafieian. Optimizing user engagement through adaptive ad sequencing. *Marketing Science*, 2022.

P. E. Rossi and G. M. Allenby. Bayesian statistics and marketing. *Marketing Science*, 22 (3):304–328, 2003.

G. J. Stigler. The economics of information. *Journal of Political Economy*, 69(3):213–225, 1961. ISSN 00223808, 1537534X.

S. Tunuguntla. Display ad measurement using observational data: A reinforcement learning approach. 2021.

L. Waltman and U. Kaymak. Q-learning agents in a cournot oligopoly model. *Journal of Economic Dynamics and Control*, 32(10):3275–3293, 2008.

W. Wang, B. Li, and X. Luo. Deep reinforcement learning for sequential targeting. *Available at SSRN 3487145*, 2021.

M. L. Weitzman. Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society*, pages 641–654, 1979.