Collusion and Artificial Intelligence: A computational experiment with sequential pricing algorithms under stochastic costs

Gonzalo Ballestero*

November 2021

Abstract

Firms increasingly delegate their strategic decisions to algorithms. A potential concern is that algorithms may undermine competition by leading to pricing outcomes that are collusive, even without having been designed to do so. This paper investigates whether Q-learning algorithms can learn to collude in a setting with sequential price competition and stochastic marginal costs adapted from Maskin and Tirole (1988). By extending a previous model developed in Klein (2021), I find that sequential Q-learning algorithms leads to supracompetitive profits despite they compete under uncertainty and this finding is robust to various extensions. The algorithms can coordinate on focal price equilibria or an Edgeworth cycle provided that uncertainty is not too large. However, as the market environment becomes more uncertain, price wars emerge as the only possible pricing pattern. Even though sequential Q-learning algorithms gain supracompetitive profits, uncertainty tends to make collusive outcomes more difficult to achieve.

Keywords: Competition Policy, Artificial Intelligence, Algorithmic Collusion

JEL Codes: D43, K21, L13

*Universidad de San Andrés. E-mail: gballestero@udesa.edu.ar. This is the final version of my master's degree thesis. I am grateful for valuable discussions with and support from Lucía Quesada. I would also like to thank Daniel Heymann, Martín Zimmermann, Florencia Gabrielli, Manuel Willington, Lucila Porto, Pablo Zárate and Franco Nuñez for encouragement and insightful comments. I also thank conference participants at the LVI Annual Meeting of the Argentine Association of Economic Policy. All errors and omissions are my own.

1 Introduction

In the modern economy there is a widespread use of algorithms to set prices particularly on online platforms such as Amazon and eBay (Chen et al., 2016) and even in the gasoline market (Assad et al., 2020). However, the use of algorithms entails potential challenges and risks to competition. A potential concern is that algorithms may lead to pricing outcomes that are collusive even without having been designed to do so and even without explicit communication [1]. This concern raises the question about what competition authorities can do to deal with such undesirable situations. In a recent paper, Harrington (2018) proposes an experimental test in order to determine whether an algorithm can be used by firms or it should be legally prohibited. It consists of designing an artificial marketplace and let pricing algorithms interact repeatedly under controlled conditions. Then the outcomes are evaluated. If supracompetitive prices emerge in this experimental setting, then these algorithms may be prohibited because there are reasons to believe that they could reduce competition and harm consumers.

A recent article by Calvano et al. (2020) considers an infinitely repeated Bertrand game where firms use Q-learning algorithms to set their prices simultaneously, and they find that algorithms can lead to collusive strategies. Similarly, Klein (2021) considers a setting adapted from Maskin and Tirole (1988) where pricing algorithms update their prices sequentially in a deterministic environment and he also finds that Q-learning algorithms often coordinate on collusive outcomes. Furthermore, it is shown that the algorithms can converge to a focal price equilibria above the competitive level or to an Edgeworth price cycle.

Despite these insights, the model presumes an unchanging market environment. Nevertheless, it is well-known that real markets are constantly subject to unexpected exogenous shocks such as shifts in cost or demand and, as a consequence, firms compete under uncertain conditions. Since it is well-known that uncertainty hinders tacitly collusive agreements (Ivaldi

[1] Apart from algorithmic collusion, algorithms could also influence competition by offering better demand predictions (Miklós-Thal and Tucker, 2019) or by serving as commitment devices (Brown and MacKay, 2021).

et al., 2003; Harrington, 2017), it is not clear whether sequential Q-learning algorithms are capable to learn collusive strategies in such complex environment. To answer this question, a computational experiment is conducted in this paper by extending the model of Klein (2021) and allowing stochastic marginal cost. This article contributes to the recent literature of algorithmic pricing by addressing how different factors influence the algorithm performance. My main result is that sequential Q-learning algorithms gain supracompetitive profits when they compete under uncertainty. The algorithms can coordinate on focal price equilibria or an Edgeworth cycle provided that uncertainty is not too large. Nonetheless, as the market environment becomes more uncertain, price wars emerge as the only possible pricing pattern. The intuition behind this results is the following. When marginal cost remains high the algorithms tend to coordinate on a focal price, but when cost decreases, algorithms have incentives to serve the entire market. Thus, a period of undercutting begins. Therefore, uncertainty tends to make collusive outcomes more difficult to achieve. This extends the results in Klein (2021) to the case where the marginal cost is stochastic and it identifies under which conditions focal price equilibrium can emerge after the repeated interaction of sequential Q-learning algorithms. Such result could potentially give predictions regarding when price wars or price rigidity would be observed in markets where firms use pricing algorithms. My finding is also aligned with previous results in the literature of computational methods for oligopoly models. Noel (2008) employs a dynamic programming approach to simulate the canonical model of Maskin and Tirole (1988) when marginal costs vary from period to period and through numerical simulations he also finds that the symmetric duopoly model only converges to an Edgeworth cycle. Also, my paper is closest to that of Calvano et al. (2021). These authors show how Q-learning algorithms can learn to collude in an environment in which the algorithms choose a quantity to produce under demand uncertainty and imperfect monitoring adapted from Green and Porter (1984). Their findings indicate that imperfect monitoring is not an obstacle to autonomous algorithmic collusion.

The rest of the paper is organized as follows. The next section describes the economic en-

vironment where the algorithms operate. Section 3 defines the algorithms and the baseline calibration. Section 4 discusses the results for two main cases. The first one refers to a deterministic scenario in which the marginal cost remains constant over time. In the second one the algorithms compete under uncertainty and the outcomes are analyzed for different calibrations. Section 5 provides several robustness analysis. Finally, section 6 concludes with a brief discussion of the limitations of the analysis and future research areas.

2 Economic Environment

Following Maskin and Tirole (1988) and Eckert (2004), I consider an alternating-move duopoly model with stochastic marginal costs. There are two firms, indexed by $i = 1, 2$, and competition between them takes place in infinitely repeated discrete time indexed by $t \in \{0, 1, 2, ...\}$. Firm 1 chooses a price in every odd period and firm 2 sets its price in even periods. This timing reflects commitments to price for two periods. The price space is discrete and it is given by $P = \{0, \frac{1}{k}, \frac{2}{k}, ..., 1\}$ where k is a parameter to be defined. The finite price grid also ensures that the best response function is well defined. For instance, if firm i sets a price $p_i = p > 0$, and firm j wants to undercut p_i, the optimal price it can set is $p_j = p - \frac{1}{k}$. Each firm aims to maximize its cumulative stream of discounted future profits

$$\max \sum_{t=0}^{\infty} \delta^t \pi_{it}(p_{it}, p_{jt}, c_t) \tag{1}$$

where $\delta \in (0, 1)$ is the discount factor. It is assumed that there are no fixed costs or capacity constraints, therefore the profit of firm i at time t can be written as

$$\pi_{it}(p_{it}, p_{jt}, c_t) = (p_{it} - c_t)\, D_i(p_{it}, p_{jt}) \tag{2}$$

where D_i is the demand function which is defined as follows

$$D_i(p_{it}, p_{jt}) = \begin{cases} 1 - p_{it} & \text{if} \quad p_{it} < p_{jt} \\ \frac{1}{2}(1 - p_{it}) & \text{if} \quad p_{it} = p_{j,t} \\ 0 & \text{if} \quad p_{it} > p_{jt} \end{cases} \tag{3}$$

Both firms have the same marginal cost, c_t, and it can be either high (c_H) with probability $\rho \in (0, 1)$ or low (c_L) with probability $1 - \rho$ in any period. The shocks are purely idiosyncratic and have no persistency over time. The absence of auto-correlation among the shocks implies that there is significant uncertainty. Define p_L^c and p_H^c as the competitive prices and p_L^m and p_H^m as the monopoly prices under low and high marginal costs, respectively. I follow Maskin and Tirole (1988) by imposing the Markov assumption: strategies only depend on variables that are directly payoff-relevant, which in this case are the price set by the rival in the previous period and the current marginal cost. Therefore, I assume that each firm observes the current marginal cost and the competitor's past price before setting its price. This assumption entails a reaction function that specifies a price response for every possible rival price and for each possible marginal cost.

The solution concept is the Markov Perfect Equilibrium, which is a subgame perfect Nash equilibrium under the Markov assumption. Maskin and Tirole (1988) shows in their canonical model that two sets of Markov Perfect Equilibrium are possible for a sufficiently high discount factor: focal price equilibria and Edgeworth cycle equilibria. Focal price equilibria are characterized by constant prices over time. Firms tacitly colluding and all charging the monopoly price in each period is an example. In contrast, in Edgeworth cycle equilibria firms gradually undercut each other. Once the prices have fallen to marginal cost, undercutting ceases, and firms play a war of attrition with each firm mixing between raising price back to the top of the cycle and remaining at marginal cost. Both firms have an incentive to raise their price, but prefer the other firm to do so.

I assume that firms delegate their strategic decisions to pricing algorithms called Q-learning

algorithms. Provided that marginal cost is constant over time, Klein (2021) finds that pricing algorithms can coordinate on a focal price equilibria or an Edgeworth cycle. However, in this paper I consider a significant departure from Klein (2021) by assuming stochastic marginal costs. Particularly, I investigate the collusion capacity of Q-learning algorithms under uncertainty. Maintaining the focus on the same type of algorithms helps to compare the results and identify the specific effects of time-varying costs. The next section provides a brief description of the Q-learning algorithms used in this paper.

3 Sequential Q-Learning

Q-Learning is a reinforcement learning algorithm that aims to maximize the expected discounted value of future rewards for unknown environments with repeated interaction. The algorithm only learns trough experience, associating states and actions with the payoff they generate. Specifically, an algorithm tries an action at a particular state, and evaluates its consequences in terms of the immediate reward or penalty it receives and its estimate of the value of the state to which it is taken. By trying all actions in all states repeatedly, it learns which are best overall, judged by long-term discounted reward (Watkins and Dayan, 1992). In this way, it may learn an optimal strategy. Formally, the objective of the algorithm i is to maximize

$$\max_{\{p_{it}\}_{t=0}^{\infty}} V_i(s_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \delta^t \pi_i(p_{it}, s_t)\right] \qquad \text{s.t} \qquad \begin{aligned} s_{t+1} &= g(p_{it}, s_t) \\ s_0 &\text{ given} \end{aligned} \tag{4}$$

In each period the algorithm observes a state variable $s_t \in S$ and then chooses an action $p_{it} \in P$. Given the state and the action chosen, the algorithm gets an immediate reward π_i that depends on the algorithm's own action p_{it} and on the state s_t. It is assumed that the reward function π_i is bounded. The future state is determined by the function g which

is the state transition function. Under the one-period memory assumption the state s_t is a pair (p_{jt-1}, c_t). A finite memory is necessary for the state space to be finite, given that the action space is finite as well. The solution of (4) is given by the optimal policy function $h_i^* : S \to P$, which is defined as follows

$$h_i^*(s_t) = \underset{p_{it}}{\operatorname{argmax}} V_i^*(s_t) \tag{5}$$

where V_i^* is the value function and gives the discounted cumulative reward obtained by following the optimal policy beginning at state s_t. Equation (5) suggests that to compute the optimal policy function it is necessary to calculate the value function. To do so, we can start rewriting the sequential problem (4) as a recursive problem by using the Bellman's principle of optimality

$$V_i^*(s_t) = \max_{p_{it}} \left[\pi_i(p_{it}, s_t) + \delta \pi_i(p_{it}, s_{t+1}) + \delta^2 V_i^*\left(s_{t+1}\right)\right] \tag{6}$$

Assuming that the agent has perfect knowledge of the immediate reward function π_i and the state transition function g, it can use the value iteration method to solve the recursive problem and, as a result, can learn the optimal policy function (Stokey et al., 1989). In other words, this means that the algorithms can predict in advance the exact outcome of setting a certain price in any state, and this assumption is not reasonable in some applications. Therefore in this paper it is assumed that π_i and g are unknown. Thus, instead of using the value iteration method, our algorithms use another method called Q-learning[2].

For our purposes, lets define the function $Q_i(p_{it}, s_t)$ which gives the maximum discounted cumulative reward that can be achieved starting from state s_t and applying the action p_{it} as the first action. In other words, the value of Q_i is the reward received immediately upon

[2]For a textbook treatment on Q-learning, see Sutton and Barto (2018)

executing action p_{it} from state s_t, plus the value of following the optimal policy thereafter:

$$Q_i(p_{it}, s_t) \equiv \pi_i(p_{it}, s_t) + \delta\pi_i(p_{it}, s_{t+1}) + \delta^2 V_i^*(s_{t+1}) \tag{7}$$

By taking the maximum of both sides and using equation (6) yields

$$\max_{p_{it}} Q_i(p_{it}, s_t) = V_i^*(s_t) \tag{8}$$

Therefore, we can rewrite the equation (5) in terms of Q as

$$h_i^*(s_t) = \operatorname*{argmax}_{p_{it}} Q_i(p_{it}, s_t) \tag{9}$$

It shows that if the agent learns the Q_i function instead of the V_i^* function, it will be able to select optimal actions even when it has no knowledge of the functions π_i and g. As equation (9) makes clear, the algorithm needs only consider each available action p_{it} in its current state s_t and choose the action that maximizes $Q_i(p_{it}, s_t)$. Hence, if the agent knew the Q-matrix, it could then easily calculate the optimal action for any given state.

Learning. Note that equation (8) allows rewriting equation (7) as

$$Q_i(s_t) = \pi_i(p_{it}, s_t) + \delta\pi_i(p_{it}, s_{t+1}) + \delta^2 \max_{p_{it+1}} Q_i\left(p_{it+1}, s_{t+1}\right) \tag{10}$$

This recursive definition of Q_i provides the basis for estimating the Q-matrix by an iterative approximation procedure. Starting from an arbitrary initial matrix Q_0, after choosing action p_{it} in state s_t, the algorithm observes π_i and s_{t+1} and updates the corresponding cell of the matrix $Q_t(p_{it}, s_t)$ according to the following equation:

$$Q_{it+1}(p_{it}, s_t) = (1-\alpha)Q_{it}(p_{it}, s_t) + \alpha\left[\pi_i(p_{it}, s_t) + \delta\pi_i(p_{it}, s_{t+1}) + \delta^2 \max_{p_{it+1}} Q_{it}(p_{it+1}, s_{t+1})\right] \tag{11}$$

where $\alpha \in (0, 1)$ is a learning parameter that regulates how quickly new information replaces

old information. When α is equal to 0 then there is no learning process since the agent does not take into account the new information acquired. Analogously, when α is equal to 1, the algorithm would immediately forget what it has learned in the past. Thus, to ensure that an effective learning process takes place, α must take any value between 0 and 1.

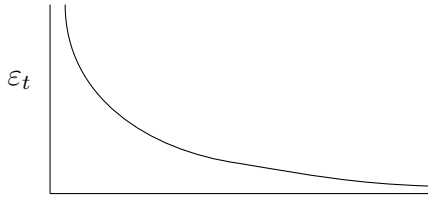
According to equation (11), the new estimate of the optimal long-run value given state s_t consists of three components: direct profit $\pi_i(p_{it}, s_t)$, next period profit $\pi_i(p_{it}, s_{t+1})$ when new state s_{t+1} realizes but the price has not changed (discounted for one period), and the highest possible Q-value $\max_{p_{it+1}} Q_{it}(p_{it+1}, s_{t+1})$ in this new state s_{t+1} (discounted for two periods). This enables an itaratively approximation in which initially the Q-values are imprecise, but over time, they become better estimates of the long-run consequences of choosing p_{it} in state s_t. It is worth noticing that Q-learning is slow because it updates only one cell of the Q-matrix at a time, and approximating the true matrix generally requires that each cell be visited many times. The larger the state or action space, the more iterations will be needed.

The action selection. I use a procedure called ε-greedy exploration: with probability ε_t it chooses a price randomly at period t and with probability $1 - \varepsilon_t$ it chooses the currently optimal action (i.e., the one with the highest Q-value in the relevant state). During the exploration phase, the algorithm explores the state-action space in order to discover the payoffs associated with a certain action in a given state.

$$p_{it} = \begin{cases} p \sim U(P) & \text{with} \quad \text{probability } \varepsilon_t \\ \underset{p_i}{\text{argmax}} \; Q_i(p_i, s_t) & \text{with} \quad \text{probability } 1 - \varepsilon_t \end{cases} \tag{12}$$

where $U(P)$ is a discrete uniform distribution over the action set P and the exploration rate is defined as $\varepsilon_t = (1 - \theta)^t$ with $\theta \in (0, 1)$. This implies that initially the algorithm chooses an action randomly, but then the probability of selecting the greedy choice increases as time goes by. Figure 1 shows ε_t over the curse of the simulation. In case of ties the algorithm randomizes over all currently optimal actions.

Figure 1: The ε-greedy policy with decreasing probability of exploration



Convergence. In a single-agent setting Q-learning converges to the optimal strategy under certain conditions (Watkins and Dayan, 1992). Intuitively, a sufficient condition for such convergence is that each action-state pair should be visited infinitely often. However, in a multi-agent setting, when two or more algorithms interact repeteadly with each other, the problem becomes non-stationary and history dependent and there is no theoretical result that guarantees ex-ante that the agents will learn an optimal policy. However, this does not imply that Q-learning is expected to behave badly; it simply means that theory is not able to say how well Q-learning is expected to work. In the absence of theoretical guidance, this paper takes an experimental approach to develop an empirical understanding of multi-agent Q-learning. I rely on computational simulations to derive results on algorithmic collusion. A pseudocode of the algorithm used in the simulations[3] is provided below.

3.1 Baseline specification

In order to facilitate comparisons, I use the same calibration as in Klein (2021). I set $k = 12$ price intervals between 0 and 1. The discount factor δ is 0.95 and the learning parameter is $\alpha = 0.3$. Each experiment consists of $T = 500,000$ periods and the experimentation parameter is $\theta = 2.75 \times 10^{-5}$ which implies that the probability of exploration gradually decreases from 100% at the beginning to 0.1% halfway through the run, reaching 0.0001% at the end (so $\varepsilon_{0.5T} = 0.001$ and $\varepsilon_T = 0.000001$). A lower value of θ implies a more extensive experimentation phase. If θ were equal to 0, the algorithm would never attain a stable

[3]The code was run in Matlab version 2017b and it can be obtained from the author upon request.

Algorithm 1: Pseudocode Sequential Q-Learning

Data
Set demand and learning parameters
Initialization
Initialize Q_i according to $Q_i(p_i, s) = 0$ for all (p_i, s)
Initialize (p_{it}, p_{jt}) and c_t for $t = 1, 2$
Initialize $i = 1$, $j = 2$ and $t = 3$
Loop over each period
    Compute (π_{t-2}, π_{t-1}) according to equation (2)
    Update $Q_i(p_{it-2}, s_{t-2})$ according to equation (11).
    Set $c_t = c_H$ with probability ρ or $c_t = c_L$ with probability $1 - \rho$
    Set p_{it} according to equation (12) and set $p_{jt} = p_{jt-1}$
    Update $t = t + 1$
    Update $i = j$ and $j = i$
Until $t = T$

behavior because it would be constantly experimenting. Analogously, a greater value of θ entails a shorter experimentation phase and, as a result, the algorithm would not be able to learn how to successfully play since it would not visit all the action-state pairs. As for the initial matrix Q_0, the baseline choice is to initiate the Q-values with all zeros. The fact that the initial Q-values are all equal can be interpreted as the algorithm not knowing the quality of their actions for any state at the beginning. Each experiment is run under the same set of parameters and the initial state s_0 is drawn randomly at the beginning of each experiment. To reduce the stochastic noise, I run $S = 300$ simulations.

4 Computational Results

The following subsections provides an analysis of the simulation results for two cases: the deterministic cost benchmark, where the marginal cost remains constant over time (*e.g.*, $\rho = 0$), and the stochastic cost scenario, where marginal cost in any period can be either high (c_H) with probability $\rho \in (0, 1)$ or low (c_L) with probability $1 - \rho$. This exercise aims to show that sequential Q-learning algorithms lead to supracompetitive outcomes even if they compete under structural uncertainty. Furthermore, by comparing the outcomes of

both cases it is possible to identify the specific effects time-varying costs have on collusion capacity. I analyzed the outcomes for the final 1000 periods. This is because the algorithms attain a stable behavior during the last periods, provided that the experimentation rate ε_t is decreasing over time. Consequently, by considering the last 1000 periods it is possible to study the strategies that algorithms have learned.

4.1 Sequential pricing algorithms under deterministic cost

In this subsection I analyze the outcomes for the baseline parametrization described in Section 3.1 when the marginal cost remains constant over time, $c_t = c$ for all $t = 1, 2, ..., T$. This analysis is performed for three different values $c \in \{0, 0.2, 0.4\}$. These magnitudes were deliberately chosen in order to ensure that the monopoly price increases as the marginal cost varies and all of them are in the action space. The results of the deterministic case replicates the findings of Klein (2021) and they serve as a benchmark to compare the collusive behavior under structural uncertainty.

4.1.1 Prices and profits

Table 1 summarizes pricing outcomes during the last 1000 periods. This table shows the types of behavior the algorithms learn after repeated interaction. It also shows the average market price, the minimum and maximum price for each possible marginal cost. The algorithms converge to a constant price in a significant fraction of the experiments whereas in the rest of them the algorithms converge to a non-constant price. This replicates the results in Klein (2021) and are also supported by the experimental evidence as documented by Leufkens and Peeters (2011). It is observed that for each possible marginal cost the average market price is above the competitive level regardless the pricing pattern the algorithms have converged to, which implies that algorithms manage to charge supracompetitive prices. Also it is observed that as the marginal cost increases, the average market price tends to increase as well. This finding suggests that the algorithms adapt their behavior to different market conditions.

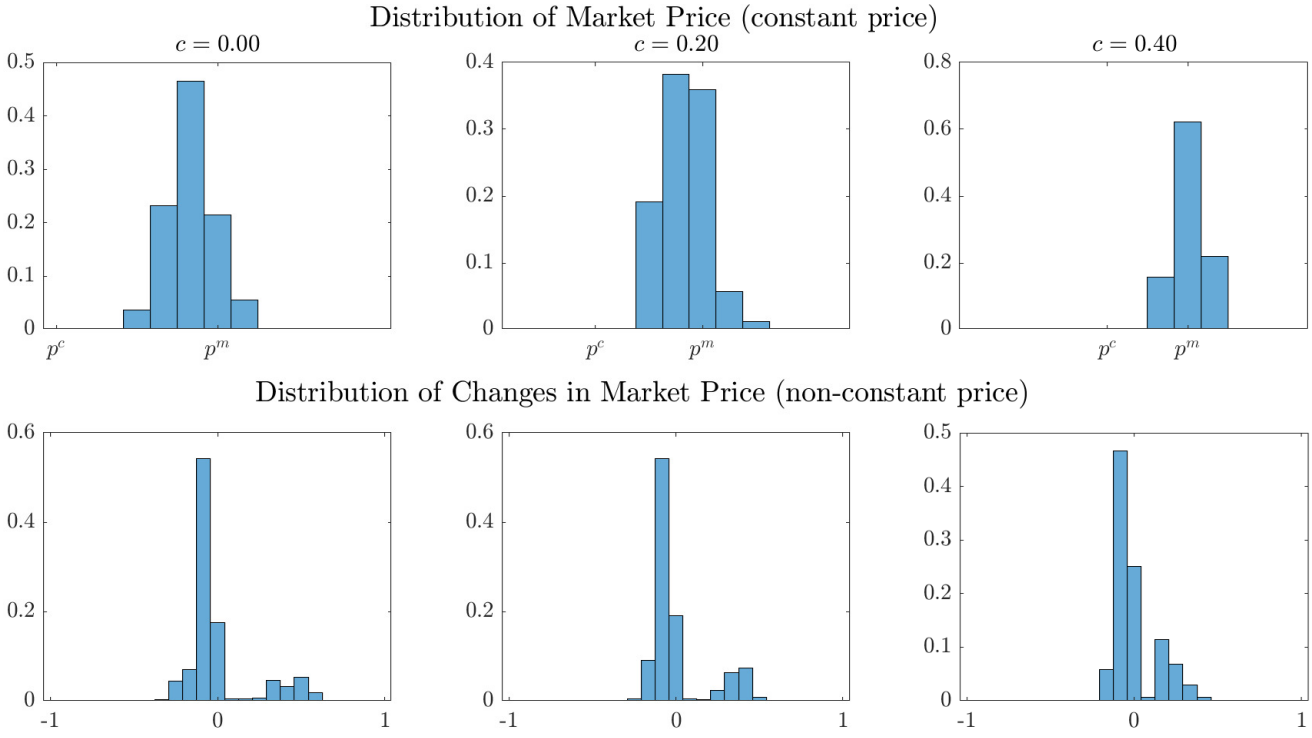
Table 1: Pricing outcomes

			constant price				non-constant price			
c	p^c	p^m	Freq.	p^{min}	$\bar{p}$	p^{max}	Freq.	p^{min}	$\bar{p}$	p^{max}
0.00	0.00	0.50	19%	0.25	0.42	0.58	81%	0.00	0.35	0.83
0.20	0.25	0.58	30%	0.42	0.53	0.75	70%	0.25	0.46	0.83
0.40	0.42	0.67	27%	0.58	0.67	0.75	73%	0.42	0.59	0.92

Note: The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$ and $S = 300$.

The upper panel of the Figure 2 illustrates the histogram of the market price charged during the last 1000 periods for those experiments that converge to a constant price. It is shown that the distributions are centered around the monopoly price p^m or one step away. Therefore, as it was suggested above, the algorithms coordinate on a constant price which is above the competitive level. The lower panel shows the distribution of changes in market price for all other simulations which have converged to a non-constant price. Across specifications, the lower panel reveals a recognizable pattern: price decreases occur much more often but are smaller in magnitude than price increases. This finding suggests that the algorithms generate a stylized fact which claims that *prices rise like rockets but fall like feathers* (Tappata, 2009). Figure 3 illustrates both types of price paths the algorithms display after repeated interaction. Just for the sake of illustration, I only consider the final 50 periods. The upper panel shows the price path for a particular simulation that have converged to a constant price. It shows that both algorithms sustain a fixed price near the monopoly level, which can be interpreted as the algorithms converging to a focal price equilibrium. On the other hand, the lower panel shows the price path for a particular simulation that has converged to a non-constant price. It shows that algorithms generate a pattern that resembles to an Edgeworth cycle, where algorithms successive undercut their competitor and after they reach the competitive price, they return to a price level that is above the monopoly price. This pricing pattern pushes the average market price above its competitive level. The lower panel also reveals that algorithm 2 always resets the price cycle. This is because Q-learning is restricted to playing pure strategies.

Figure 2: Histogram of pricing outcomes



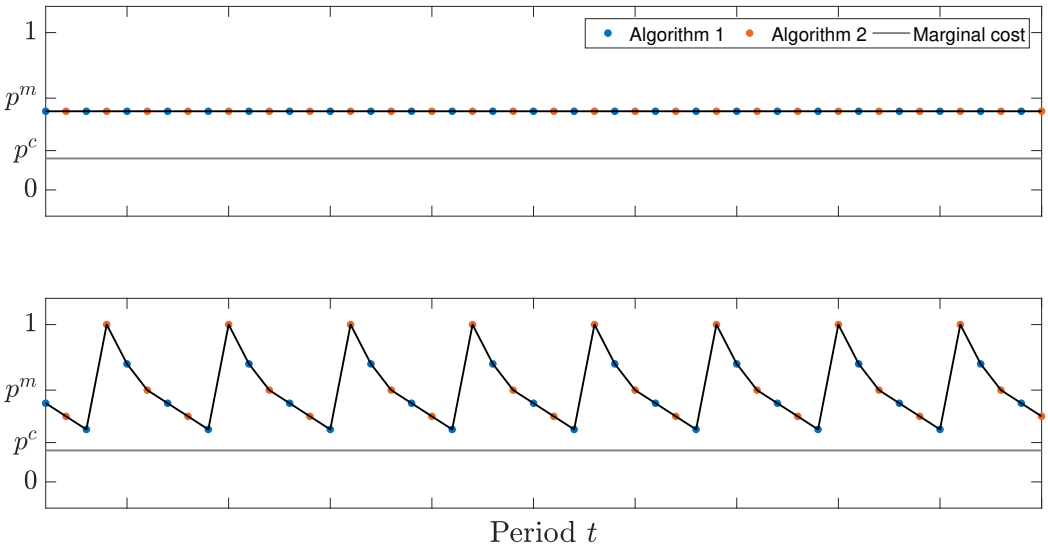
Note: The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$ and $S = 300$.

Since in some experiments the algorithms coordinate on a supracompetitive fixed price and in the rest of them they converge to a price cycle that is reset after reaching the competitive level, these pricing dynamics suggest profits that are on average supracompetitive. To explore this hypothesis, following Calvano et al. (2020) I compute the normalized aggregate profit for each experiment, which is defined as follows

$$\Delta = \frac{\bar{\pi} - \pi^c}{\pi^m - \pi^c} \tag{13}$$

where $\bar{\pi}$ is the average aggregate profit during the final 1000 periods, π^c is the profit in perfect competition and π^m is the monopoly profit. Thus, $\Delta = 0$ corresponds to the competitive outcome and $\Delta = 1$ to the perfectly collusive outcome. Figure 4 shows the average profit gain for each possible marginal cost. It is observed that algorithms converge to supracompetitive

Figure 3: Focal price and Edgeworth price cycle



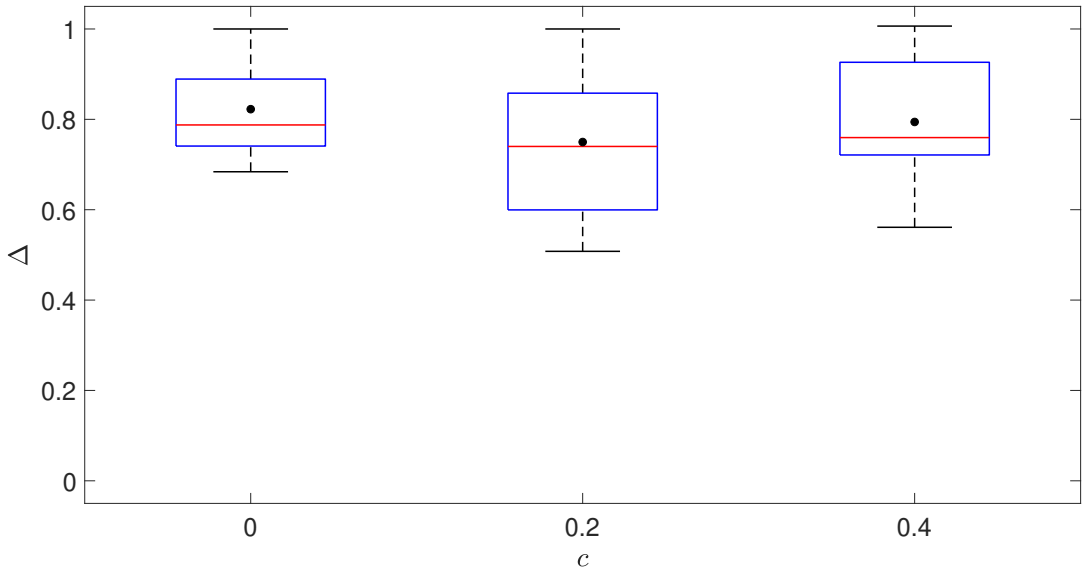
Note: Price path during the final 50 periods of a simulation that have converged to a constant price (upper panel) and a non-constant price (lower panel). The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$ and $S = 300$.

average profits. For example, the average profit gain is $\Delta = 82.25\%$ for $c = 0$ and $\Delta = 74.98\%$ for $c = 0.2$.

4.1.2 Punishment strategies

Besides the fact that algorithms obtain supracompetitive profits on average, to assure the existence of algorithmic collusion it is necessary to determine whether they learn strategies that embody a reward-punishment scheme to sustain such supracompetitive outcome. Therefore, following the approach used in Calvano et al. (2020) and Klein (2021), I analyze whether the algorithms have learned to punish deviations only for those simulations in which the algorithms have converged to a constant price above the competitive level. Starting from the average fixed price that the algorithms have converged to, I exogenously force one algorithm to deviate in one period by slightly undercutting its competitor. The other algorithm instead continues to play according to its learned strategy. By doing this, I observe the reaction of the algorithms in the subsequent periods when the forced cheater reverts to its

Figure 4: Normalized profits



Note: For each marginal cost c, the box extends form the first quartile to the third quartile with a red line at the median. The whiskers show the range of the data. The black dot stands for the mean. The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$ and $S = 300$.

learned strategy as well. The deviation price is always just below the pre-deviation price. Thus the deviating agent always choose the most profitable one-period deviation. Table 2 reports the resulting average market price for each value of c. In all cases deviations are punished since the one-period deviation in $\tau = 0$ is followed by a price war. After the price cut, the other algorithm sets a lower price which implies that deviation triggers a punishment to restore the focal price. The algorithms gradually return to their predeviation behavior after 5-7 periods.

Table 2: Pre and post-deviation market prices

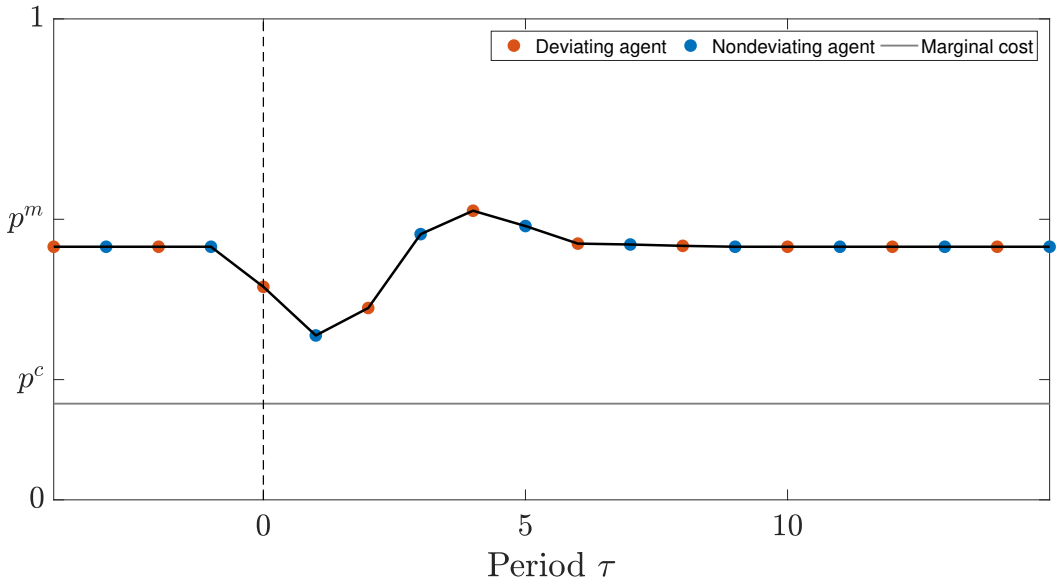
				Period τ											
c	Freq.	p^c	p^m	-1	0	1	2	3	4	5	6	7	8	9	10
0.00	19%	0.00	0.50	0.42	0.33	0.21	0.20	0.38	0.53	0.51	0.47	0.43	0.42	0.42	0.42
0.20	30%	0.25	0.58	0.53	0.44	0.34	0.40	0.55	0.60	0.57	0.53	0.53	0.53	0.53	0.53
0.40	27%	0.42	0.67	0.67	0.59	0.50	0.53	0.63	0.71	0.69	0.68	0.67	0.67	0.67	0.67

Note: The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$ and $S = 300$.

Figure 5 illustrates the price reaction to defection for $c = 0.2$. It suggests that the algorithms

learn to punish price deviations by playing a reward-punishment scheme. The non-deviating agent reacts to defection by lowering its price in $\tau = 1$, the period after defection. The defecting agent expects this punishment and sets her price near the punishment price as well. Then both agents simultaneously start to slowly increase their prices. Already after five periods, the prices return to the pre-defection level.

Figure 5: Deviation and Punishment



Note: One algorithm is forced to deviate in period $\tau = 0$. The deviation lasts for one period only. The figure plots the average market price for those simulations in which the algorithms have converge to a fixed price. The parameter values are $k = 12$, $c = 0.2$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$ and $S = 300$.

4.2 Sequential pricing algorithms under stochastic cost

In the previous subsection, I replicated the results of Klein (2021) as a benchmark to compare how uncertainty influences the algorithm performance. In this subsection I analyze the outcomes when the marginal cost is stochastic and it can takes two values, $c_t = c_L$ with probability $1 - \rho$ or $c_t = c_H$ with probability ρ for all $t = 1, 2, ..., T$. The shocks are purely idiosyncratic and have no persistency. The absence of auto-correlation among the shocks implies that the algorithms face considerable uncertainty. Low marginal cost is set to $c_L = 0$ and the high marginal cost varies according to particular specifications as well as

the probability of the shock. I consider $c_H \in \{0.2, 0.40\}$ and $\rho \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. As it is shown below, when sequential Q-learning algorithms compete under uncertainty, they achieve supracompetitive profits.

4.2.1 Prices and profits

Table 3 summarizes the pricing outcomes under uncertainty as the probability ρ varies and for each value of high marginal cost c_H. As in the deterministic benchmark, I start by classifying the pricing outcomes. To make such classification I follow a practical criterion: constant pricing pattern is deemed to be achieved if the market price remains constant over at least 50% of the last 1000 periods. That is, a simulation attains a constant pricing pattern if the market price remains without changes in most periods. Based on this criterion, I find that the algorithms coordinate on a constant price if the probability that costs change is not too large. Panel (a) of Table 3 reports that the algorithms learn to play a fixed price in a considerable fraction of experiments for $\rho = 0.10$ and $\rho = 0.90$. Alternatively, and similar to findings of Noel (2008), as the probability that costs change increases, algorithms only converge to a non-constant pricing pattern. Panel (b) of Table 3 shows that when $\rho \in \{0.30, 0.50, 0.70\}$ the algorithms charge a non-constant market price for each possible value of high marginal cost.

The table also reports the average market price, $\bar{p}$, and to facilitate comparison it also shows the theoretical average competitive price, $\bar{p}^c$, which is the average of the competitive prices that would occur in one-shot games: p_H^c with probability ρ and p_L^c with probability $1-\rho$. Similarly, the theoretical average monopoly price, $\bar{p}^m$, is the average of the monopoly prices that would occur in one-shot games: p_H^m with probability ρ and p_L^m with probability $1-\rho$. It is observed that algorithms always charge supracompetitive prices on average, regardless the type of behavior they display. For a given probability ρ, the average market price increases as the size of the high marginal cost increases. The same finding is observed when the high marginal cost c_H remains constant and the probability of the cost shock increases. All these

observations suggest that algorithms adjust their prices by increasing them when they face higher uncertainty. This implies that algorithms learn how to adapt their behavior in order to operate under uncertainty.

Table 3: Pricing outcomes under uncertainty as ρ varies

(a) constant price

ρ	c_H	Freq.	$\bar{p}^c$	$\bar{p}^m$	$\bar{p}$
0.10	0.20	15%	0.08	0.50	0.38
	0.40	11%	0.08	0.50	0.45
0.90	0.20	18%	0.25	0.58	0.49
	0.40	41%	0.42	0.67	0.61

(b) non-constant price

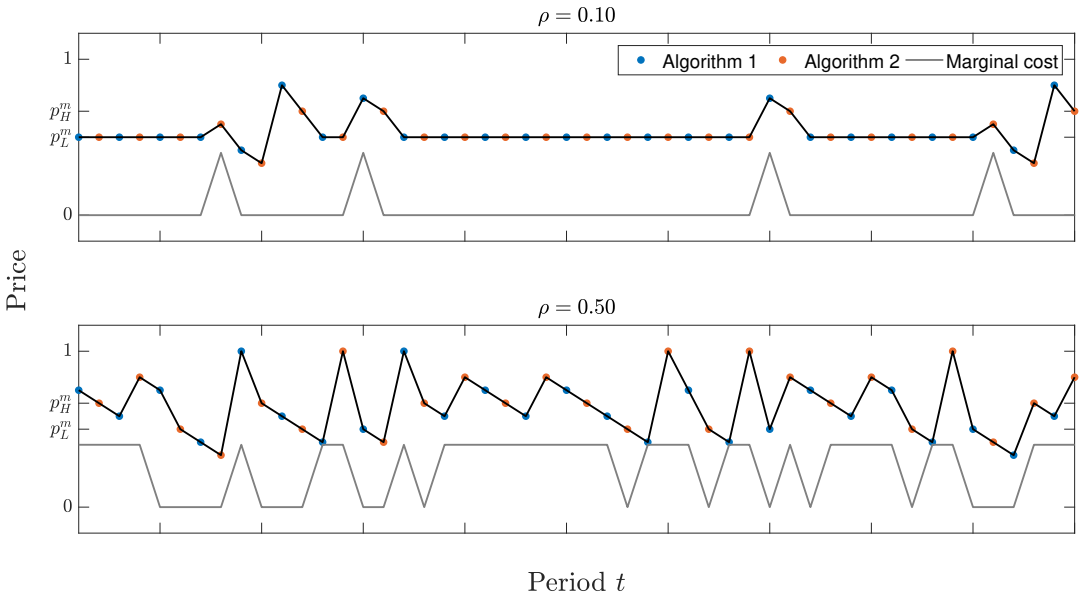
ρ	c_H	Freq.	$\bar{p}^c$	$\bar{p}^m$	$\bar{p}$
0.10	0.20	85%	0.08	0.50	0.38
	0.40	89%	0.08	0.50	0.40
0.30	0.20	100%	0.08	0.50	0.39
	0.40	100%	0.17	0.58	0.43
0.50	0.20	100%	0.17	0.58	0.41
	0.40	100%	0.25	0.58	0.47
0.70	0.20	100%	0.25	0.58	0.44
	0.40	100%	0.33	0.67	0.52
0.90	0.20	82%	0.25	0.58	0.47
	0.40	59%	0.42	0.67	0.58

Note: The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$ and $S = 300$.

Figure 6 illustrates both types of price paths the algorithms play after repeated interaction. Just for the sake of illustration, I only consider the final 50 periods. The upper panel shows the price path for a particular simulation that has converged to a constant price for $\rho = 0.10$ and $c_H = 0.40$. It shows that both algorithms sustain a fixed price near the monopoly level when the marginal cost is low, but after the shock the algorithms enter into a price war that lasts for several periods and then revert to the pre-deviation price. Thus a constant pricing pattern can be understood as a focal price equilibria. In addition, the lower panel shows the price path for a particular simulation that has converged to a non-constant price for $\rho = 0.50$ and $c_H = 0.40$. As in the deterministic benchmark, the pricing path follows a pattern similar to an Edgeworth cycle that is reset after reaching the competitive price level.

Table 4 reports the normalized profits under uncertainty. It shows that they are closed to the collusive level, and the minimum is attained when $\rho = 0.50$. Thus, when algorithms face considerable uncertainty they perform worse than in other specifications. To asses the

Figure 6: Constant and non-constant price path



Note: Price path during the final 50 periods of a simulation that have converged to a constant price (upper panel) and a non-constant price (lower panel). The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$ and $S = 300$.

Table 4: Normalized profits under uncertainty as ρ varies

c_H \ ρ	0.10	0.30	0.50	0.70	0.90
0.20	82.53%	78.67%	70.24%	74.16%	74.21%
0.40	84.39%	81.44%	71.42%	77.14%	79.27%

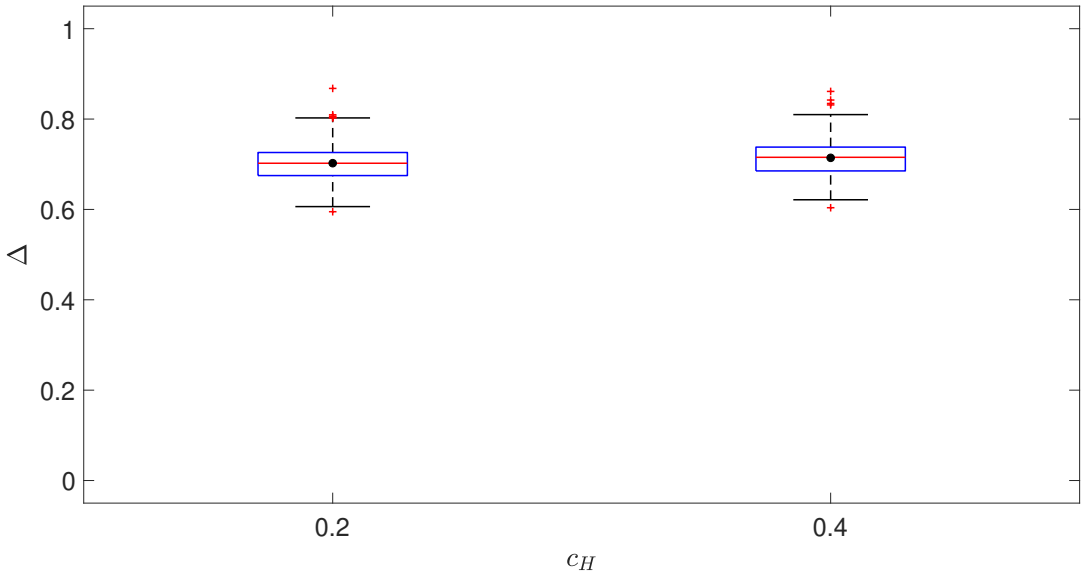
impact of stochastic marginal cost on collusion, I compare the normalized profits gained in the deterministic case for $c = 0.2$ (see Section 4.1.1) and the normalized profits gained in the stochastic scenario when $c_L = 0.2$, $c_H = 0.4$ and $\rho = 0.5$. Both cases are comparable since they are characterized by the same average marginal cost of $\bar{c} = 0.2$. Therefore, in this way I can identify the effect of the stochastic marginal cost on the algorithm performance. Table 5 reports that the difference between profit gains under deterministic marginal cost and stochastic marginal cost is 3.56% in absolute terms. It implies that uncertainty tends to make collusive outcomes more difficult to achieve with sequential pricing algorithms. The magnitude is also consistent with the finding of Calvano et al. (2021).

Table 5: The impact of uncertainty on collusive outcomes

Deterministic marginal cost	Stochastic marginal cost
74.98%	71.42%

Figure 7 extends the computational results reported in Table 4 by showing the distribution of normalized profits gain for $\rho = 0.5$ as c_H varies. It is observed that for each possible value of c_H, the distribution of profits does not display a remarkable statistical dispersion and the mean remains approximately constant regardless the value of the high marginal cost.

Figure 7: Normalized profits under uncertainty as c_H varies



Note: For each marginal cost c, the box extends form the first quartile to the third quartile with a red line at the median. The whiskers show the range of the data. The black dot stands for the mean. Values beyond the whiskers are considered outliers and are plotted as individual points. The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$ and $S = 300$.

4.2.2 Punishment strategies

As in the deterministic benchmark, high price levels alone are not proof of algorithmic collusion as learning to play those prices could be merely myopic. In other words, the algorithm could have learned to play certain prices without developing an underlying understanding of the strategic components of the market environment. From a theoretical perspective,

punishment strategies are vital for collusion to be sustainable in the long run. Thus, to confirm that the market price of the algorithms are indeed collusive, it is necessary to establish whether the algorithms learn to punish price deviations. To test this hypothesis I basically perform the same analysis mentioned in Section 4.1.2. Starting from the strategies the algorithms have converged to, I let them interact and then I observe what happens if one algorithm is forced to deviate in one period by slightly undercutting its competitor while the other continues to play according to its learned strategy. However, since marginal cost varies over time, a decrease in price after the deviation maybe due to the fact that the other algorithm triggers a punishment to restore the focal price or simply because the cost state has changed from high to low state and the other algorithm sets a lower price. To avoid such confounding effects, the marginal cost is set equal to the high state and it remains constant over time. Table 6 summarizes the average market price for those simulations that were able to converge to a single fixed price given that the marginal cost is constant. In period $\tau = -1$ both algorithms play according to the strategy they have learned. Then, in period $\tau = 0$, one algorithm is forced to deviate by undercutting the price of the competitor. Afterwards, both algorithms play according to their learned strategies again. It is shown that the average market price is close to the average monopoly price for all calibrations. Additionally, as in the deterministic benchmark, deviations are followed by price wars and the algorithms gradually return to their predeviation behavior after a few periods. Therefore, these results suggest that algorithms do not only learn to play high prices but also strategies that make collusion incentive compatible, even when they operate under structural uncertainty. Figure 8 complements these computational results by illustrating the price reaction to defection for $c_H = 0.4$ and $\rho = 0.5$.

4.2.3 Price wars in equilibrium

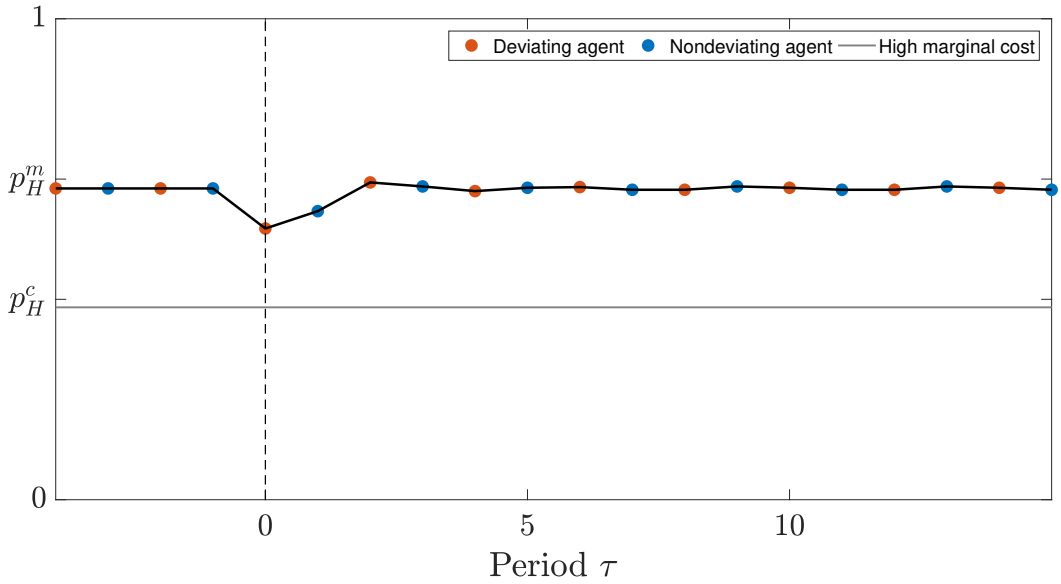
In the previous sections, I argued that the algorithms can learn collusive strategies and they coordinate on a focal price equilibrium or an Edgeworth cycle provided that the probability

Table 6: Pre and post-deviation market prices

					Period τ											
ρ	c_H	p_H^c	p_H^m	Freq.	-1	0	1	2	3	4	5	6	7	8	9	10
0.10	0.20	0.25	0.58	8%	0.54	0.46	0.63	0.58	0.57	0.58	0.56	0.57	0.55	0.56	0.55	0.56
	0.40	0.42	0.67	10%	0.63	0.55	0.73	0.76	0.66	0.65	0.64	0.64	0.64	0.64	0.64	0.64
0.30	0.20	0.25	0.58	9%	0.52	0.43	0.40	0.78	0.6	0.55	0.53	0.52	0.52	0.52	0.52	0.52
	0.40	0.42	0.67	18%	0.63	0.54	0.70	0.70	0.68	0.64	0.64	0.64	0.63	0.62	0.64	0.64
0.50	0.20	0.25	0.58	11%	0.52	0.44	0.46	0.51	0.47	0.49	0.57	0.53	0.52	0.52	0.52	0.52
	0.40	0.42	0.67	20%	0.65	0.56	0.58	0.65	0.66	0.65	0.63	0.65	0.64	0.64	0.64	0.64
0.70	0.20	0.25	0.58	15%	0.49	0.40	0.33	0.33	0.72	0.58	0.53	0.50	0.49	0.49	0.49	0.49
	0.40	0.42	0.67	31%	0.64	0.56	0.51	0.65	0.64	0.67	0.64	0.64	0.64	0.64	0.64	0.64
0.90	0.20	0.25	0.58	18%	0.51	0.43	0.27	0.54	0.53	0.61	0.52	0.51	0.51	0.51	0.51	0.51
	0.40	0.42	0.67	41%	0.66	0.58	0.49	0.57	0.66	0.69	0.67	0.66	0.66	0.66	0.66	0.66

Note: The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$ and $S = 300$.

Figure 8: Deviation a Punishment starting from high marginal cost state



Note: One algorithm is forced to deviate in period $\tau = 0$. The deviation lasts for one period only. The figure plots the average market price for those simulations in which the algorithms have converge to a fixed price, starting from the high marginal cost state. The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$, $S = 300$, $c_H = 0.40$ and $\rho = 0.50$.

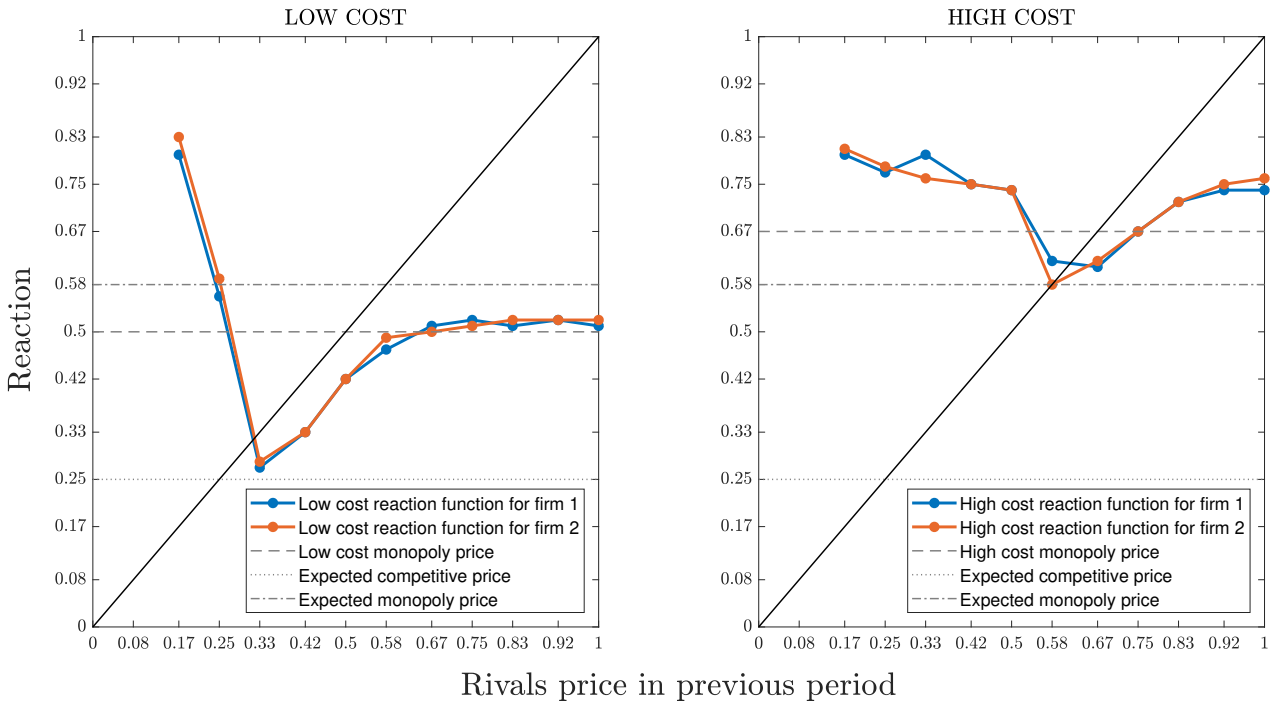
that costs change is not too large. However, as the probability that costs change increases, algorithms only converge to a an Edgeworth cycle. Therefore, when the market environment becomes more uncertain, the likelihood that a price war will occur increases. In this section, I study the underlying mechanism that may drive such price wars by analyzing the reaction functions the algorithms have learned for the case in which the high marginal cost is $c_H = 0.4$

and the probability is $\rho = 0.50$. Given the assumption of one-period memory, the reaction function of firm i, $R_i(p_{jt-1}, c_t)$, depends on the price set previously by i's rivals, namely firm j, and on the current marginal cost. This reaction function specifies a price response for every possible rival price and for each possible marginal cost. Since the algorithms learn their strategies by experimentation, the reaction function that the algorithms converge to depend quite sensitively on the specific history of the interaction between them. As a result, there is a considerable variation in the learned reaction functions. Averaging across the 300 simulations, eliminates much of the noise and reveals a clear pattern. Figure 9 depicts the average reaction functions the algorithms converge to for each possible state of marginal cost and Table 7 illustrates numerically the reaction function for firm 1. The table gives the price and corresponding period profits when cost is low or high. To facilitate the interpretation of the strategies, Figure 9 also depicts the monopoly price in each state and the expected competitive and monopoly price. The upward-sloping straight line on each graph has a slope of 1 and represents the price matching line. Where the reaction function is below this line represents an undercut, and where the reaction function is above this line corresponds to price restorations.

The figure shows some remarkable behaviors. Firstly, firms repeatedly undercut each other at high prices if costs are high, and at lower prices if costs are low. In the low-cost state the algorithms successively undercut their competitor and before they reach the expected competitive price, they return to a price level that is above the expected monopoly price. In the high-cost state, the algorithms undercut until they reach the expected monopoly price, then they restore the cycle by setting a price that is substantially above the high cost monopoly price. These observations suggest that the undercutting phase is more aggressive in the low-cost state. The intuition behind this finding is similar to that of Rotemberg and Saloner (1986), in periods in which cost is low, the temptation to undercut the competitors' price is greater than when cost is high. Therefore, decreases in cost induce price wars since a firm facing low cost has incentive to serve the entire market.

Secondly, the dynamic reaction functions reveals that firms tend to gravitate more easily towards a focal price in the high-cost state than in the low-cost state. The high-cost focal price is given by the expected monopoly price, where the reaction functions overlap the price matching line. This is consistent with the results depicted earlier in Figure 8, where it was shown that the algorithms play reward-punishment scheme to sustain focal price equilibria in the high-cost state.

Figure 9: Reaction functions



Note: The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$, $S = 300$, $c_H = 0.40$ and $\rho = 0.50$.

Both findings suggest that if costs remain low, firms wish to take advantage of the temporarily low costs by serving the entire market. As a result, a period of undercutting begins. Alternatively, if costs remain high, firms tend to coordinate on the expected monopoly price. With strategies of this sort, price wars occur in stochastic environments with one-period memory. Such finding could potentially predict that when firms use sequential Q-learning algorithms to set prices, then price wars would be observe provided that firms compete under uncertainty.

Table 7: Reaction function for firm 1

p_2	$R_1(p_2, c_L)$	$\pi_1(p_2, c_L)$	$R_1(p_2, c_H)$	$\pi_1(p_2, c_H)$
1.00	0.51	0.25	0.74	0.09
0.92	0.52	0.25	0.74	0.09
0.83	0.51	0.25	0.72	0.09
0.75	0.52	0.25	0.67	0.09
0.67	0.51	0.25	0.61	0.08
0.58	0.47	0.25	0.62	0.00
0.50	0.42	0.24	0.74	0.00
0.42	0.33	0.22	0.75	0.00
0.33	0.27	0.20	0.80	0.00
0.25	0.56	0.00	0.77	0.00
0.17	0.80	0.00	0.80	0.00

Note: Average reaction function over 300 simulations and profits

5 Robustness

The above results suggest that sequential Q-learning algorithms achieve supracompetitive profits even when they face an environment with stochastic marginal cost. This finding is robust for each combination of c_H and ρ. In this section, I consider that the baseline calibration of the stochastic experiment is defined by $c_H = 0.4$ and $\rho = 0.5$ and then I analyze how robust the baseline results are to changes in the economic environment. In the following subsections, I briefly report the main results of this analysis.

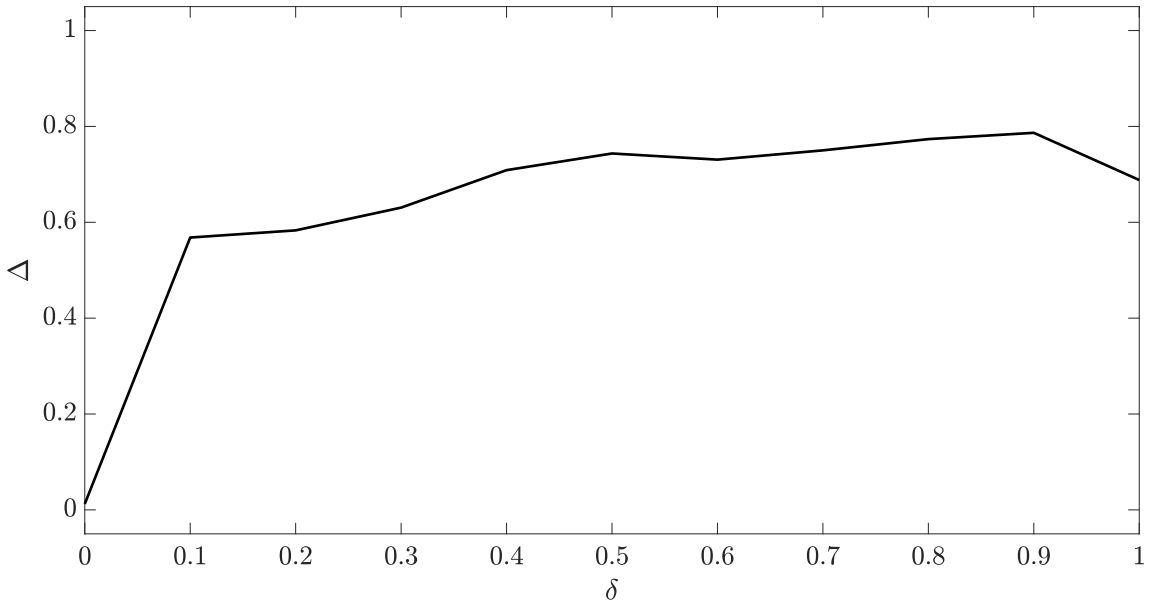
5.1 Alternative action set

I explore the consequences of enlarging the discrete price grid by setting $k = 24$. I find that the normalized aggregate profit gain is $\Delta = 76.60\%$ when $k = 24$, so it remains above the competitive level. It is worth noting that a higher cardinality of the action set implies that the Q-matrix is much larger than the baseline model. Consequently, it is more difficult that algorithms converge to supracompetitive outcomes *a priori*. Nonetheless, the normalized aggregate profit remains approximately constant in comparison with the results reported in Table 4. Hence, results are robust to an increase in the amount of pricing intervals k.

5.2 Discount factor

In the baseline parametrization I have set disconunt factor $\delta = 0.95$ reasonably close to 1. In this subsection I analyze how the results change as the discount factor takes lower values. Figure 10 summarizes how the normalized aggregate profits varies with δ. Consistent with the theoretical literature and the experimental evidence (Bruttel, 2009; Ivaldi et al., 2003), it is observed that a lower discount factor is an obstacle to achieve collusive outcomes. In particular, it reveals that when δ is low, algorithms consistently learn to coordinate on a competitive outcome. Particularly, when $\delta = 0$ the average profit gain is close to $\Delta = 0$.

Figure 10: Normalized profits Δ as a function of the discount factor δ



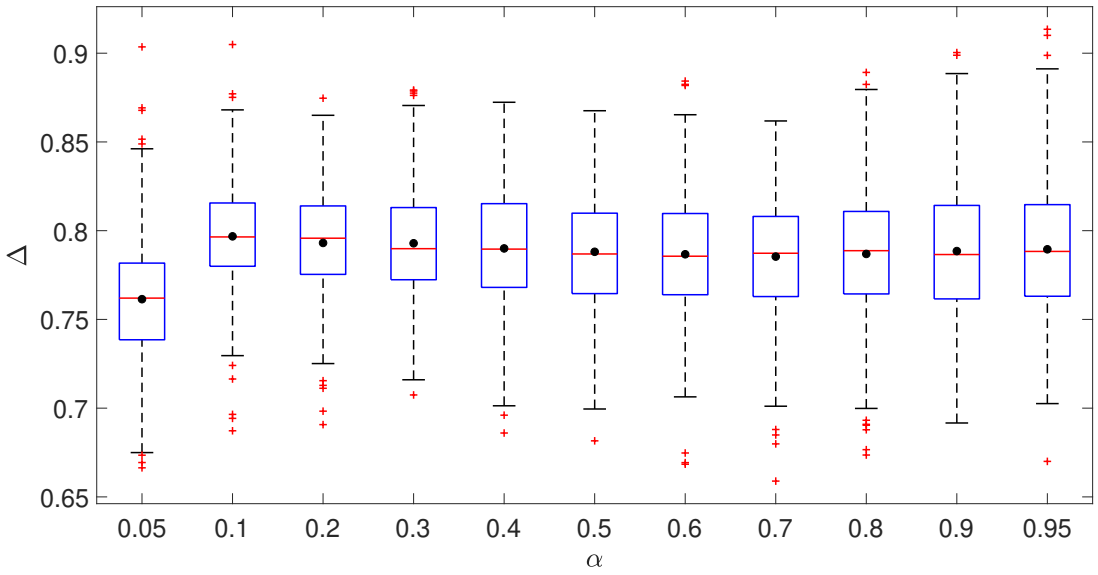
Note: The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$, $S = 300$ and $c_H = 0.40$.

5.3 Learning parameter

In the baseline stochastic experiment, I have set the learning parameter $\alpha = 0.3$. Figure 11 shows that the results appears to be robust under different learning rates, and the maximum average normalized profit is attained when $\alpha = 0.1$. This finding is in line with Calvano et al. (2020), since the authors argue that a value of 0.1 is commonly used in the computer science

literature. Interestingly, It is observed that the average profit gain for $\alpha = 0.05$ is lower than the average profit gain for $\alpha = 0.95$. However this difference is not statistically significant. Intuitively, when algorithms operate in an environment that is constantly changing due to exogenous shocks, there is no advantage in acquiring new information rather than relying solely on what has been learned in the past.

Figure 11: Normalized profits Δ as α varies



Note: For each marginal cost c, the box extends form the first quartile to the third quartile with a red line at the median. The whiskers show the range of the data. The black dot stands for the mean. Values beyond the whiskers are considered outliers and are plotted as individual points. The parameter values are $k = 12$, $\delta = 0.95$, $\alpha = 0.3$, $T = 500,000$, $S = 300$, $c_H = 0.40$ and $\rho = 0.50$.

5.4 Higher uncertainty

In the baseline stochastic model I have assumed that marginal cost can take two values, $c_L = 0$ and $c_H = 0.4$ with probability $\rho = 0.5$. In this subsection, I consider three marginal cost levels: $c_L = 0$, $c_M = 0.2$ and $c_H = 0.4$. Each cost state has the same probability of occurring, thus $p(c_L) = p(c_M) = p(c_H) = \frac{1}{3}$. This implies that the algorithms face higher uncertainty. I examine how the level of collusion varies if uncertainty increases. I find that the averaged normalized profit gain is $\Delta = 74.98\%$. This robustness check confirms

that sequential Q-learning algorithms manage to achieve supracompetitive profits despite of facing structural uncertainty.

5.5 Triopoly

As a final robustness check, I extend the baseline stochastic model by considering a three firm game since it is a well-documented finding in the literature that tacit collusion becomes less likely as the number of firms in the market increases (Huck et al., 2004). The reason behind this fact is twofold. On one hand, a larger number of market participants implies higher deviation profits, which increases the incentive to deviate from a collusion price level. On the other hand, with more firms in the market the cardinality of the state space also increases since agents have to condition their behavior on one additional variable, namely, the past price chosen by the extra competitor. This increase in strategic complexity may further hinder collusion.

In the three-firm setting, each firm can adjust its price every third period and its price is fixed for the following two. Firm 1 adjusts its price in period $3t+1$, firm 2 in $3t+2$ and firm 3 in $3t+3$ where $t = 0, 1, 2, ...$ In this setting, the learning equation (11) is rewriting as follows:

$$Q_{it+1}(p_{it}, s_t) = (1-\alpha)Q_{it}(p_{it}, s_t) + \alpha \left[\pi_i(p_{it}, s_t) + \delta\pi_i(p_{it}, s_{t+1}) + \delta^2\pi_i(p_{it}, s_{t+2}) + \delta^3 \max_{p_{it+2}} Q_{it}(p_{it+2}, s_{t+2})\right] \quad (14)$$

where $s_t = (p_{jt-2}, p_{lt-2}, c_t)$ with $j, l \in \{1, 2, 3\}$, $i \neq j$, $i \neq l$ and $j \neq l$. The per period profit function is the standard formulation. The lowest priced firm serves the entire market, or if two or more firms have the lowest price, they split the market in halves or thirds accordingly. As equation (14) illustrates, the algorithms should care about its profits gain during three periods when they have to choose the price. Consistently with the findings in the literature, I find that in a market with three firms the average profit gain is $\Delta = 53.30\%$, and thus significantly lower than the average profit of the baseline model $\Delta = 71.42\%$. Therefore,

collusive outcomes are more difficult to achieve when there are more players.

6 Conclusion

This article presents an extension of Klein (2021) by allowing stochastic marginal costs. It was shown that sequential Q-learning algorithms leads to supracompetitive profits and this finding is robust to parameter changes and various extensions. The algorithms can coordinate on focal price equilibrium or an Edgeworth cycle provided that uncertainty is not too large. However, as the market environment becomes more uncertain, price wars emerge as the only possible pricing pattern. The underlying mechanism that triggers such price wars is similar to that of Rotemberg and Saloner (1986). When marginal cost remains high the algorithms tend to coordinate on a focal price, but when cost decreases, algorithms have incentive to serve the entire market. Thus, a period of undercutting begins. Therefore, even though sequential Q-learning algorithms gain supracompetitive profits, uncertainty tends to make collusive outcomes more difficult to achieve. This main finding extends the results in Klein (2021) by identifying under which conditions focal price equilibria can emerge after the repeated interaction of sequential Q-learning algorithms. Such result could potentially give predictions regarding when price wars or price rigidity would be observed in markets where firms use pricing algorithms.

This paper can be extended in several dimensions. First, I assume only one pricing algorithm, but in real markets, companies may use a wider class of algorithms. In this sense, a future area of research is to address what happens in a market environment where heterogeneous algorithms compete against each other. Secondly, in this paper I have analyzed the outcomes generated by a repeated interaction between algorithms, but in platforms the algorithms also interact with human beings (Crandall et al., 2018; Werner, 2021). Thus, it is necessary to study whether humans and machines can learn to cooperate to achieve collusive outcomes in the framework considered in this paper.

References

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

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

Bruttel, L. V. (2009). The critical discount factor as a measure for cartel stability? *Journal of Economics*, 96(2):113–136.

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

Calvano, E., Calzolari, G., Denicolò, V., and Pastorello, S. (2021). Algorithmic collusion with imperfect monitoring. *International Journal of Industrial Organization*, page 102712.

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

Crandall, J. W., Oudah, M., Ishowo-Oloko, F., Abdallah, S., Bonnefon, J.-F., Cebrian, M., Shariff, A., Goodrich, M. A., Rahwan, I., et al. (2018). Cooperating with machines. *Nature communications*, 9(1):1–12.

Eckert, A. (2004). An alternating-move price-setting duopoly model with stochastic costs. *International Journal of Industrial Organization*, 22(7):997–1015.

Green, E. J. and Porter, R. H. (1984). Noncooperative collusion under imperfect price information. *Econometrica: Journal of the Econometric Society*, pages 87–100.

Harrington, J. E. (2017). *The theory of collusion and competition policy*. MIT Press.

Harrington, J. E. (2018). Developing competition law for collusion by autonomous artificial agents. *Journal of Competition Law & Economics*, 14(3):331–363.

Huck, S., Normann, H.-T., and Oechssler, J. (2004). Two are few and four are many: number effects in experimental oligopolies. *Journal of economic behavior & organization*, 53(4):435–446.

Ivaldi, M., Jullien, B., Rey, P., Seabright, P., and Tirole, J. (2003). The economics of tacit collusion. Technical report, Report for DG Competition, European Commission.

Klein, T. (2021). Autonomous algorithmic collusion: Q-learning under sequential pricing. *RAND Journal of Economics*, pages 1–21.

Leufkens, K. and Peeters, R. (2011). Price dynamics and collusion under short-run price commitments. *International Journal of Industrial Organization*, 29(1):134–153.

Maskin, E. and Tirole, J. (1988). A theory of dynamic oligopoly, ii: Price competition, kinked demand curves, and edgeworth cycles. *Econometrica: Journal of the Econometric Society*, pages 571–599.

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

Noel, M. D. (2008). Edgeworth price cycles and focal prices: Computational dynamic markov equilibria. *Journal of Economics & Management Strategy*, 17(2):345–377.

Rotemberg, J. J. and Saloner, G. (1986). A supergame-theoretic model of price wars during booms. *The American economic review*, 76(3):390–407.

Stokey, N., Lucas, R., and Prescott, E. (1989). *Recursive methods in economic dynamics*. Harvard University Press.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tappata, M. (2009). Rockets and feathers: Understanding asymmetric pricing. *The RAND Journal of Economics*, 40(4):673–687.

Watkins, C. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Werner, T. (2021). Algorithmic and human collusion.