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

Gonzalo Ballestero*

August 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 fluctuating marginal costs adapted from Maskin and Tirole (1988) and Eckert (2004). By extending a previous model developed in Klein (2021), I find that sequential Q-learning algorithms converge to supracompetitive outcomes despite they operate in a complex environment with structural uncertainty. Finally it is shown that these findings are robust to various extensions.

Keywords: Artificial Intelligence, Algorithmic Collusion, Competition Policy

JEL Codes: D43, K21, L13

*Universidad de San Andrés. E-mail: gballestero@udesa.edu.ar. This is a preliminary version of a 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. 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). 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. 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 sequential Q-learning algorithms indeed often coordinate on collusive equilibria. Furthermore, the author shows that the algorithms can converge to a focal price above the competitive level or to an stable Edgeworth price cycling pattern.

Despite these insigths, 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, there is considerable uncertainty. Since one of the factors that hinders collusion is cost uncertainty, 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 fluctuating marginal cost. In this way, this article contributes to the recent literature of algorithmic pricing by addressing how different factors influence the algorithm performance. Also, this 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. My main result is that Q-learning algorithms converge to supracompetitive outcomes despite they operate in a complex environment with structural uncertainty. Hence, uncertainty it is not a factor that prevents algorithms to collude. This extends the results in Klein (2021) to cases where marginal costs fluctuates. In comparison with their results, I find that algorithms only converges to an Edgeworth price cycling pattern when they operate under uncertainty. It is shown that these pricing pattern push average prices above their competitive level and, as a result, it leads to supracompetitive profits.

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

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

Marginal cost in any period can be either high (c_H) with probability $\rho \in (0, 1)$ or low (c_L) with probability $1 - \rho$. 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 assume that each firm observes the current marginal cost and the competitor's past price before setting its price. Finally, firms delegate their strategic decisions to pricing algorithms. Following Calvano et al. (2020, 2021) and Klein (2021), I investigate the collusion capacity of Q-Learning algorithms because they are simple and their parameters have a clear economic interpretation. Also maintaining the

focus on the same type of algorithms helps to compare the results and identify the specific effects of fluctuating marginal cost. 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 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 agent 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

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

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). However, the assumption that the algorithms or its human programmer can predict in advance the exact outcome of setting a certain price in a particular state it is no longer plausible. In cases where either π_i or g is unknown, we can not rely on the value iteration method. Instead of using the value iteration method, our algorithms use another method called Q-learning[1].

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

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

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

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

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

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 need 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. 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. Indeed, Watkins and Dayan (1992) proved that in a single-agent setting Q-learning converges to the optimal strategy under certain mild conditions. However, in a multi-agent setting, when two or more algorithms interact repeteadly with each other, there is no theoretical result that guarantees ex-ante that the agents will learn an optimal policy.

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)

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

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 $\theta > 0$. This implies that initially the algorithm chooses an action randomly, but then it selects the greedy choice as time increases. In case of ties the algorithm randomizes over all currently optimal actions. 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. A pseudocode of the algorithm used in the simulations is provided below.

3.1 Baseline specification

In the baseline specification of the model I set $k = 12$ price intervals between 0 and 1. As in Klein (2021), an experiment consists of $T = 500,000$ periods. In order to facilitate comparisons, I use the same learning and experimentation parameters. Thus $\alpha = 0.3$ to ensure that learning process is not too slow (α close to 0) nor the algorithm does not forget too

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 $t = T$

rapidly what it has learned in the past (α too close to 1). Additionally, I set $\theta = 2.75 \times 10^{-5}$ 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 $\varepsilon_T = 0.000001$). The discount factor δ is 0.95. As for the initial matrix Q_0, the baseline choice is to initiate the Q-values with all zeros. This can be interpreted as the algorithm does not know 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 operate in a complex environment with structural uncertainty. Furthermore, by comparing

the outcomes of both cases it is possible to identify the specific effects of fluctuating marginal cost. I analyzed the outcomes for the final 1.000 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 1.000 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 perform for five different values $c \in \{0, 0.15, 0.3, 0.45, 0.6\}$. 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 benchmark are similar to Klein (2021).

4.1.1 Prices and profits

Table 1 summarizes pricing outcomes during the last 1.000 periods. This table shows the types of behavior the algorithms learn. The algorithms converge to a single, stable fixed price in a fraction of the experiments whereas in the rest of them the algorithms converge to a non-fixed price. This replicates the results in Klein (2021) and are also supported by the experimental evidence as documented by Leufkens and Peeters (2011). Naturally, it is observed that as the marginal cost increases, the algorithms tend to converge to a fixed price more frequently. I argue that this is because a higher marginal cost implies that some prices are no longer profitable and, as a consequence, it is easier to coordinate to a single price. In the limit, if there were only one profitable price, then the algorithms would learn to play that price. Additionally, this table provides quantiles of the market price charged by the algorithms. Regarding to those runs that converge to a single fixed price, I find that the median market price is above the competitive level although is below the monopoly price.

Furthermore, it is worth to notice that as the marginal cost increases, the median market price tends to increase as well. On the other hand, when the algorithms converge to non-fixed price, it is shown that the price in the 5th percentile is above the competitive level and the price in the 95th percentile is at most two step further away from the monopoly price. This implies that algorithms charge supracompetitive prices.

			Runs with fixed price						Runs with non-fixed price					
c	p^c	p^m	N	5th	25th	50th	75th	95th	N	5th	25th	50th	75th	95th
0	0	0.50	56	0.33	0.33	0.42	0.5	0.58	244	0.08	0.25	0.33	0.50	0.58
0.15	0.17	0.58	68	0.42	0.42	0.50	0.58	0.67	232	0.25	0.33	0.42	0.58	0.67
0.30	0.33	0.67	82	0.50	0.58	0.58	0.67	0.67	218	0.33	0.42	0.50	0.67	0.75
0.45	0.50	0.75	145	0.58	0.67	0.67	0.67	0.75	155	0.50	0.50	0.58	0.75	0.83
0.60	0.67	0.83	225	0.67	0.67	0.75	0.75	0.75	75	0.67	0.67	0.75	0.83	0.83

Table 1: Price market outcomes for the final 1.000 periods. The 5th percentile, the 25th percentile, the 50th percentile, the 75th percentile, the 95th percentile of the market prices are reported. Results are in case of $k = 12$ price intervals, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

The upper panel of the Figure 1 illustrates the histogram of the market price charged during the last 1.000 periods for those experiments that converge to a fixed price. The lower panel shows the distribution of changes in market price for all other simulations which have converged to a non-fixed price. Across specifications, there is a clear pattern: price decreases occur much more often but are smaller in magnitude than price increases. In other words, the algorithms generate an stylized fact which claims that *prices rise like rockets but fall like feathers*. Figure 2 illustrates this behaviour for a representative non-fixed price simulation. Particularly, it shows that algorithms generate a pattern that resemble to an Edgeworth Cycle, where algorithms successive undercuts its competitor and after they reach the competitive price, they return to a price level that is above the monopoly price.

Since in some experiments the algorithms converge to a supracompetitive fixed price and in the rest of them the algorithms 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 Klein (2021) I compute the average profit for the final

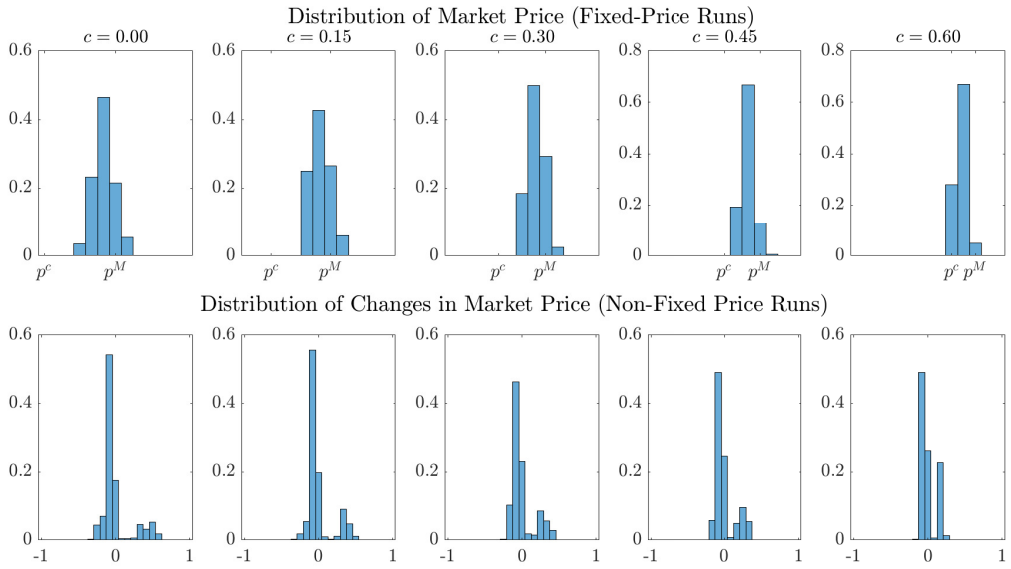

Figure 1: Histogram of the market prices and changes in market prices for the last 1.000 periods. The upper panel shows the distribution for those simulations that have converged to a fixed price and the lower panel shows the distribution of changes in market price for those runs that have converged to a non-fixed price. Results are in case of $k = 12$ price intervals, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

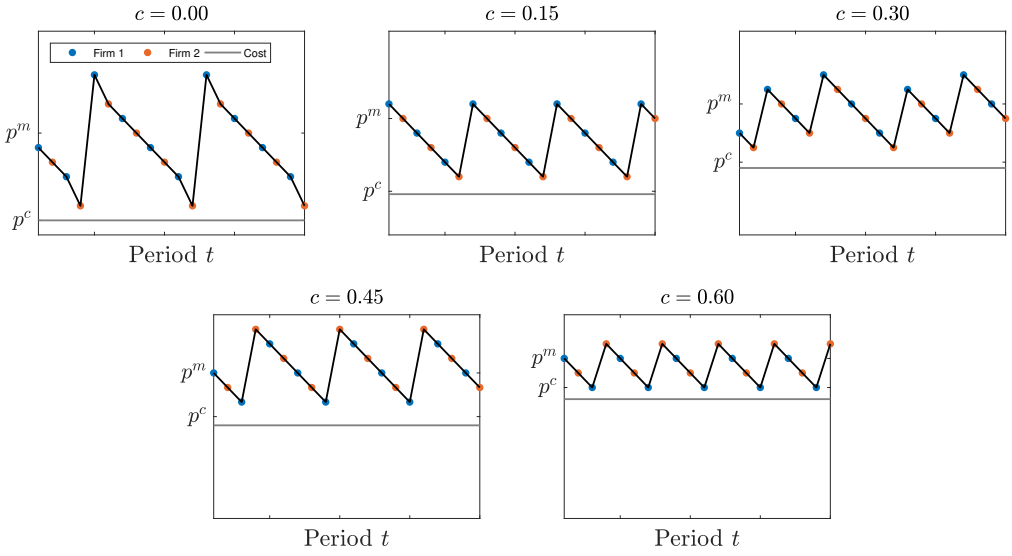

Figure 2: Edgeworth Cycles. This figure illustrates the prices charged by the algorithms and the marginal cost during the final 20 periods of a representative experiment that have converged to a non-fixed price. Results are in case of $k = 12$ price intervals, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

1.000 periods for each experiment, which is defined as follows

$$\Pi_i = \frac{1}{1000} \sum_{t=T-1000}^{T} \pi_i(p_{it}, p_{jt}, c_t) \tag{13}$$

Figure 3 shows that algorithms converge to supracompetitive profits. In some experiments the algorithms converge to the joint-profit maximizing level π^m, but in some cases they converge to an average profit that is below this level. In all cases the profits are above the competitive level and this finding is robust for all magnitudes of the marginal cost.

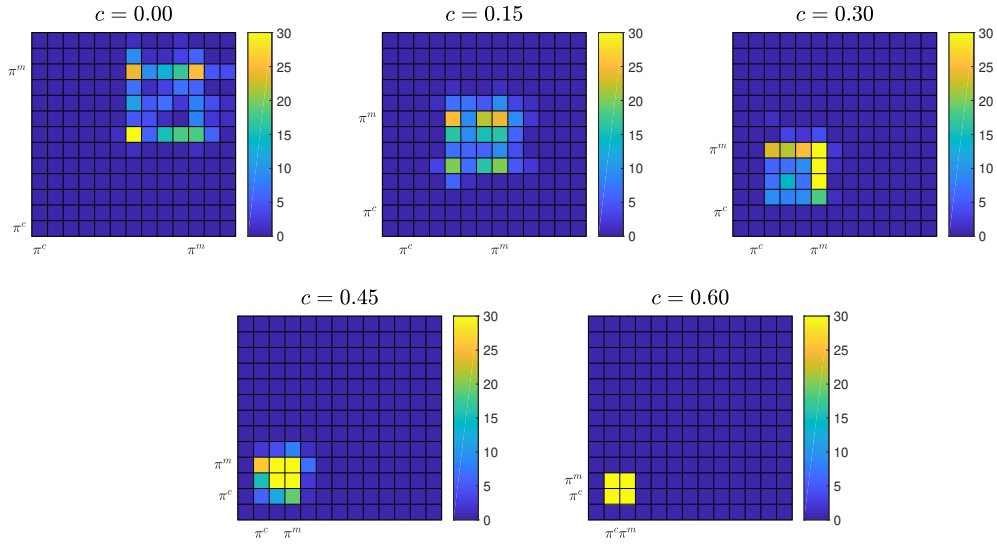

Figure 3: Distribution of profits. Results are in case of $k = 12$ price intervals, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

4.1.2 Punishment strategies

Besides the algorithms obtain on average supracompetitive profits, to assure the existence of algorithmic collusion it is necessary to determine if 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 fixed 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 his learned strategy. By doing this, I observe the reaction of the algorithms in the subsequent periods when the forced cheater reverts to his learned strategy as well. Figure 4 shows that deviations are punished since it is observed that the one-period deviation is followed by a price war. However, the algorithms gradually return to their predeviation behavior after 5-7 periods and this pattern is robust across specifications.

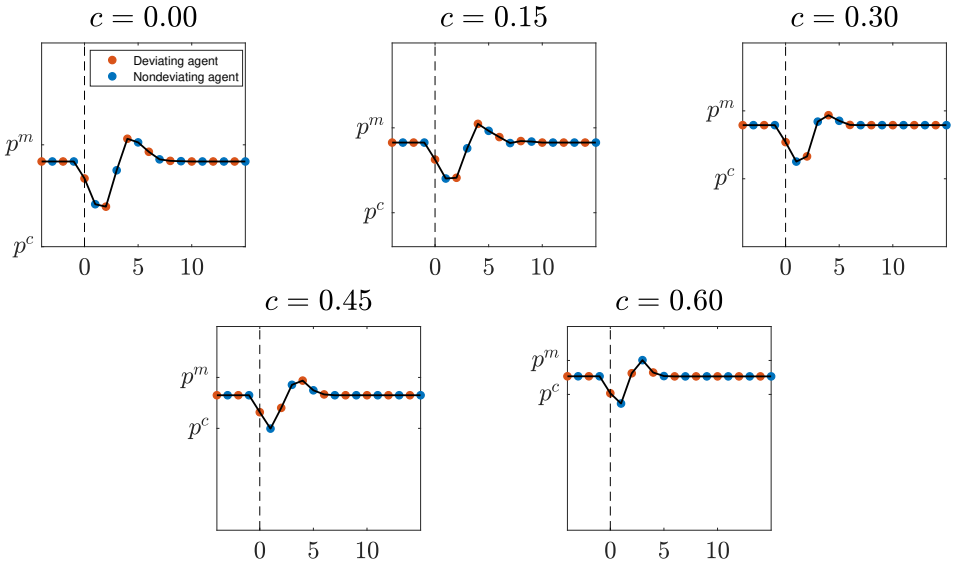

Figure 4: **Deviations and Punishments.** One algorithm is forced to deviate in period $\tau = 0$. The deviation lasts for one period only. The figure plots the average prices for those simulations in which the algorithms have converge to a fixed price. Results are in case of $k = 12$ price intervals, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

4.2 Sequential pricing algorithms under stochastic cost

In this subsection I analyze the outcomes for the baseline parametrization described in Section 3.1 when the marginal cost c_t fluctuates over time 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 $c_H \in \{0.15, 0.3, 0.45, 0.6\}$ and $\rho \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ so there are 20 specifications to be evaluated. As it is shown below, sequential Q-Learning algorithms leads to collusive outcomes even when they operate under significant uncertainty.

4.2.1 Prices and profits

I find that when the marginal cost fluctuates over time, the algorithms never converge to a constant price. Figure 5 illustrates this behaviour for a particular simulation. 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 2 summarizes average market

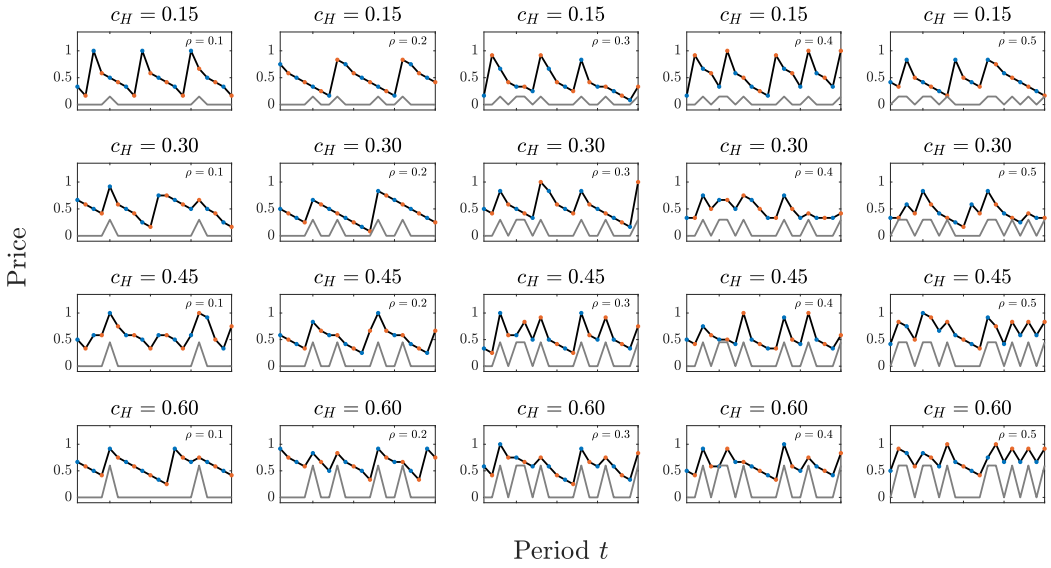

Figure 5: Edgeworth Cycles. This figure illustrates the prices charged by the algorithms and the fluctuating marginal cost during the final 20 periods of a particular simulation. Results are in case of $k = 12$ price intervals, low marginal cost $c_L = 0$, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

price and this outcome is compared with the theoretical weighted average competitive price, $\bar{p}^c$, and the theoretical weighted average monopoly price, $\bar{p}^m$, which are defined as

$$\bar{p}^z = (1 - \rho)\, p_L^z + \rho\, p_H^z \qquad z = c, m \tag{14}$$

It is shown that the average market price charged by the algorithms lies between $\bar{p}^c$ and $\bar{p}^m$ for all parametrizations. Hence the Edgeworth Cycle pricing pattern pushes the average price above its competitive level. Furthermore, for a given probability ρ, the average market price increases as the size of the high marginal cost increases. Additionally, the same finding is observed when the high marginal cost c_H remains constant and the probability of the shock increases. All these observations suggest that when the algorithms face higher uncertainty, they adjust their prices in the sense that it would be expected theoretically.

					Probability ρ														
					0.1			0.2			0.3			0.4			0.5		
c_H	p_L^c	p_H^c	p_L^m	p_H^m	$\bar{p}^c$	$\bar{p}$	$\bar{p}^m$	$\bar{p}^c$	$\bar{p}$	$\bar{p}^m$	$\bar{p}^c$	$\bar{p}$	$\bar{p}^m$	$\bar{p}^c$	$\bar{p}$	$\bar{p}^m$	$\bar{p}^c$	$\bar{p}$	$\bar{p}^m$
0.15	0	0.17	0.5	0.58	0.02	0.37	0.51	0.03	0.37	0.52	0.05	0.38	0.52	0.07	0.39	0.53	0.08	0.40	0.54
0.3	0	0.33	0.5	0.67	0.03	0.39	0.52	0.07	0.40	0.53	0.10	0.40	0.55	0.13	0.42	0.57	0.17	0.44	0.59
0.45	0	0.50	0.5	0.75	0.05	0.42	0.53	0.10	0.44	0.55	0.15	0.45	0.58	0.20	0.47	0.60	0.25	0.50	0.63
0.6	0	0.67	0.5	0.83	0.07	0.45	0.53	0.13	0.49	0.57	0.20	0.52	0.60	0.27	0.55	0.63	0.33	0.59	0.67

Table 2: Average market price for the final 1.000 periods, $\bar{p}$. The table also reports the theoretical weighted average competitive price, $\bar{p}^c$, and the theoretical weighted average monopoly price, $\bar{p}^m$. Results are in case of $k = 12$, low marginal cost $c_L = 0$, $T = 500,000$ periods, learning parameter $\alpha = 0.3$ and discount factor $\delta = 0.95$

Figure 6 illustrates the histogram of the market price charged during the last 1.000 periods for all the experiments. Complementing the results reported in Table 2, this figure shows that the distribution shifts to the right as the high marginal cost increases. Figure 7 shows that algorithms converge to supracompetitive profits on average. To facilitate interpretations, the figure also reports the theoretical weighted average competitive profit, $\bar{\pi}^c$, and the theoretical weighted average monopoly profit, $\bar{\pi}^m$, which are defined as

$$\bar{\pi}^z = (1-\rho)\ \pi_L^z + \rho\ \pi_H^z \qquad z = c, m \tag{15}$$

It is seen that when the high marginal cost or the probability of the shock increases, the coordination is even better. The intuition behind that is twofold. On one hand, as it was mentioned above, if the high marginal cost increases, there are less profitable prices. On the other hand, as the probability shock is higher, the algorithms tend to visit all the states more frequently which enables them to achieve a better learning. Both observation explain

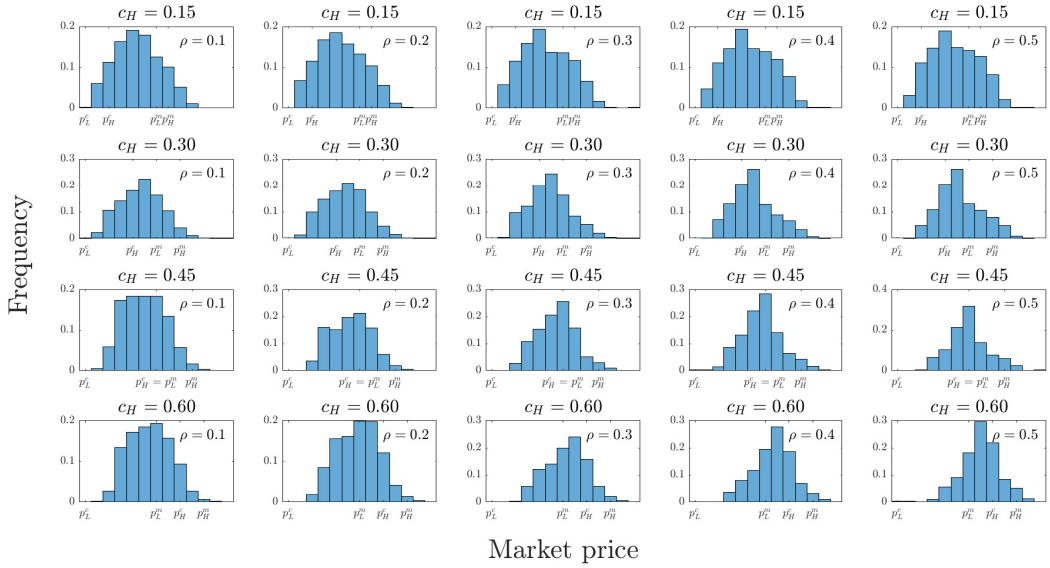

Figure 6: Histogram of the market prices for the last 1.000 periods. Results are in case of $k = 12$ price intervals, low marginal cost $c_L = 0$, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

why the distribution of average profits presents less variability as we move to the southeast.

Table 3 summarizes the aggregate average profits and it shows that they are closed to the

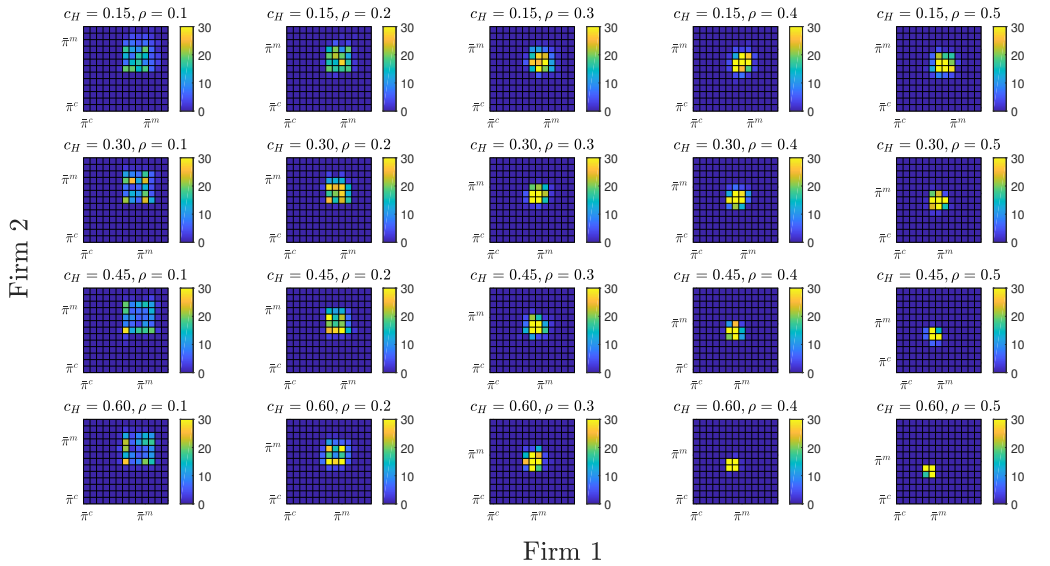

Figure 7: Distribution of profits. Results are in case of $k = 12$ price intervals, low marginal cost $c_L = 0$, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

collusive level. This finding is robust with respect to parameter changes.

					Probability ρ														
					0.1			0.2			0.3			0.4			0.5		
c_H	π_L^c	π_H^c	π_L^m	π_H^m	$\bar{\pi}^c$	$\bar{\pi}$	$\bar{\pi}^m$	$\bar{\pi}^c$	$\bar{\pi}$	$\bar{\pi}^m$	$\bar{\pi}^c$	$\bar{\pi}$	$\bar{\pi}^m$	$\bar{\pi}^c$	$\bar{\pi}$	$\bar{\pi}^m$	$\bar{\pi}^c$	$\bar{\pi}$	$\bar{\pi}^m$
0.15	0	0.01	0.25	0.18	0	0.2	0.24	0	0.19	0.24	0	0.18	0.23	0.01	0.17	0.22	0.01	0.17	0.22
0.3	0	0.02	0.25	0.12	0	0.2	0.24	0	0.18	0.22	0.01	0.17	0.21	0.01	0.16	0.2	0.01	0.15	0.19
0.45	0	0.03	0.25	0.08	0	0.2	0.23	0.01	0.18	0.22	0.01	0.16	0.2	0.01	0.15	0.18	0.01	0.13	0.16
0.6	0	0.02	0.25	0.04	0	0.2	0.23	0	0.18	0.21	0.01	0.16	0.19	0.01	0.14	0.17	0.01	0.12	0.14

Table 3: Average aggregate profits for the final 1.000 periods, $\bar{\pi}$. The table also reports the theoretical weighted average aggregate competitive profit, $\bar{\pi}^c$, and the theoretical weighted average aggregate monopoly profit, $\bar{\pi}^m$. Results are in case of $k = 12$ price intervals, low marginal cost $c_L = 0$, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

4.2.2 Adaptation

As it was showed in Section 4.2.1, the algorithms trained in an uncertain environment converge to price war paths that resemble to an Edgeworth Cycle. This phenomenon occurs because algorithms successive undercut each other, which means that their behavior is non-cooperative. But what happens if these algorithms, that were trained under uncertainty and learn not to cooperate, are put to work in a deterministic environment? Shall we expect that these algorithms continue to behave in a non-cooperatively way? Are they capable to adapt their behavior and cooperate? To answer these questions I perform the following exercise. Starting from the strategies the algorithms have converged to, I let them interact in a deterministic environment by setting the marginal cost equal to the high state and it remains constant over time. Table 4 summarizes the average market price for those simulations that were able to converge to a single fixed price. For all calibrations, the average market price is close to the theoretical monopoly price, which means that the algorithms were capable to adapt their behaviour to this new environment. Then, I test 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. So, I perform the same analysis mentioned in Section 4.1.2. Figure 8 shows that after the deviation the algorithms gradually return to their predeviation behavior after 5 periods and this pattern is robust

			Probability ρ									
			0.1		0.2		0.3		0.4		0.5	
c_H	p_H^c	p_H^m	N	$\bar{p}$	N	$\bar{p}$	N	$\bar{p}$	N	$\bar{p}$	N	$\bar{p}$
0.15	0.17	0.58	21	0.49	14	0.49	10	0.47	15	0.46	21	0.48
0.3	0.33	0.67	32	0.59	40	0.59	42	0.57	36	0.57	58	0.58
0.45	0.50	0.75	28	0.68	54	0.67	64	0.64	59	0.65	57	0.65
0.60	0.67	0.83	35	0.79	48	0.76	49	0.77	44	0.75	50	0.74

Table 4: Average market price for the final 1.000 periods, $\bar{p}$, only for those runs that have converged to a fixed single price. Results are in case of $k = 12$ price intervals, low marginal cost $c_L = 0$, learning parameter $\alpha = 0.3$ and discount factor $\delta = 0.95$

across specifications.

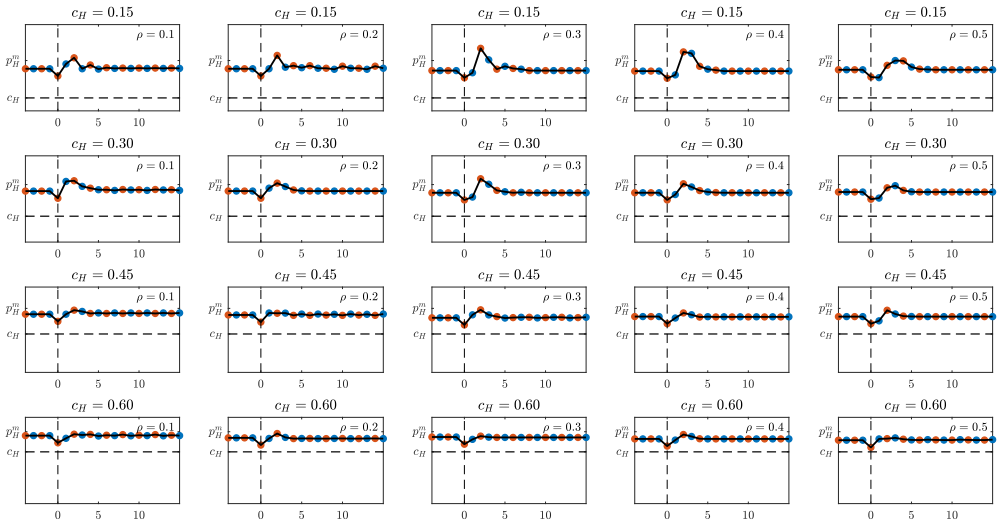

Figure 8: **Deviations.** One algorithm is forced to deviate in period $\tau = 0$. The deviation lasts for one period only. The figure plots the average prices for those simulations in which the algorithms have converge to a fixed price. Results are in case of $k = 12$ price intervals, low marginal cost $c_L = 0$, learning parameter $\alpha = 0.3$ and discount factor $\delta = 0.95$

5 Robustness

The above results suggest that sequential Q-learning algorithms can manage to coordinate on supracompetitive outcomes even when they face an environment with fluctuating marginal cost. This finding is robust accross the 20 specifications, one for each combination of c_H and

ρ. In this section, I consider that the baseline parametrization of the stochastic experiment is defined by $c_H = 0.3$ and $\rho = 0.5$ and then I analyze how robust are the baseline results to changes in the economic environment. In the following subsections, I briefly report the main results of this robustness analysis.

5.1 Action set

I explore the consequences of enlarging the discrete price grid by setting $k = 24$. Table 5 shows that when the amount of pricing intervals k increases, the average aggregate profit 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 average aggregate profit remains constant in comparison with the results reported in Table (3).

π_L^c	π_H^c	π_L^m	π_H^m	$\bar{\pi}^c$	$\bar{\pi}$	$\bar{\pi}^m$
0	0.02	0.25	0.12	0.01	0.15	0.19

Table 5: Average aggregate profits for $k = 24$ price intervals. Results are in case of low marginal cost $c_L = 0$, high marginal cost $c_H = 0.3$, probability $\rho = 0.5$, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

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. Table 6 reports how the average aggregate profits varies with δ. According to economic theory, lower discount factors is an obstacle to achieve collusive outcomes, and this is confirmed by the results summarized in Table 6. It shows that as discount factor decreases, the average aggregate profits tend to decrease as well.

						Discount factor δ				
						0.55	0.65	0.75	0.85	0.95
π_L^c	π_H^c	π_L^m	π_H^m	$\bar{\pi}^c$	$\bar{\pi}^M$	$\bar{\pi}$	$\bar{\pi}$	$\bar{\pi}$	$\bar{\pi}$	$\bar{\pi}$
0	0.02	0.25	0.12	0.01	0.19	0.13	0.14	0.14	0.14	0.15

Table 6: Average aggregate profits under different discount factors. Results are in case of $k = 12$ price intervals, low marginal cost $c_L = 0$, high marginal cost $c_H = 0.3$, probability $\rho = 0.5$, learning parameter $\alpha = 0.3$, $T = 500,000$ periods and $S = 300$ simulations

5.3 Learning parameter

In the baseline stochastic experiment, I have set the learning parameter $\alpha = 0.3$. Table 7 shows that the results appears to be robust under different learning rates. Additionally, it is observed that the average aggregate profit decreases with higher values of α, since the algorithms tend to forget more rapidly what they have learned in the past and these experiences are highly valuable in an environment that is constantly changing.

						Learning parameter α				
						0.1	0.2	0.3	0.4	0.5
π_L^c	π_H^c	π_L^m	π_H^m	$\bar{\pi}^c$	$\bar{\pi}^M$	$\bar{\pi}$	$\bar{\pi}$	$\bar{\pi}$	$\bar{\pi}$	$\bar{\pi}$
0	0.02	0.25	0.12	0.01	0.19	0.15	0.15	0.15	0.14	0.14

Table 7: Average aggregate profits under different learning rates. Results are in case of $k = 12$ price intervals, low marginal cost $c_L = 0$, high marginal cost $c_H = 0.3$, probability $\rho = 0.5$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

5.4 Higher uncertainty

In the baseline stochastic model I have assumed that marginal cost can take two values, $c_L = 0$ and $c_H = 0.3$ with probability $\rho = 0.5$. In this subsection, I consider three equiprobable marginal cost levels: $c_L = 0$, $c_M = 0.3$ and $c_H = 0.6$. This implies that the algorithms face higher uncertainty. Nevertheless, Table 8 shows that the average aggregate profit gain is supracompetitive. This robustness check confirms that uncertainty is not an obstacle for sequential Q-learning algorithms.

c_L	c_M	c_H	π_L^c	π_M^c	π_H^c	π_L^m	π_M^m	π_H^m	$\bar{\pi}^c$	$\bar{\pi}$	$\bar{\pi}^m$
0	0.3	0.6	0	0.02	0.01	0.25	0.12	0.04	0.01	0.11	0.14

Table 8: Average aggregate profits with higher uncertainty. Results are in case of $k = 12$ price intervals, learning parameter $\alpha = 0.3$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

5.5 Triopoly

As a final robustness check, I extend the baseline stochastic model by considering a three firm game. In this setting, each firm can adjust its price every third period and its price is fixed for the following two. Firm 1 adjusts its price en period t, firm 2 in $t+1$ and firm 3 in $t+2$. In this setting, the learning equation (11) is rewriting as follows:

$$Q_{it+1}(p_{it}, s_t) = (1-\alpha)Q_{it}(p_{it}, s_t) + \alpha\left[\pi_i(p_{it}, s_t) + \delta\pi_i(p_{it}, s_{t+1}) + \delta^2\pi_i(p_{it}, s_{t+2}) + \delta^3 \max_{p_{it+2}} Q_{it}(p_{it+2}, s_{t+2})\right] \quad (16)$$

where $s_t = (p_{jt-2}, p_{lt-2}, c_t)$ with $j, l \in \{1, 2, 3\}$, $i \neq j$ 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 (16) illustrates, the algorithms should care about its profits gain during three periods when they have to choose the price. Table 9 shows that results are robust to an increase in the number of firms. Consistently with theory, collusive outcomes are more difficult to achieve when there are more players. The average aggregate profit is less in comparison with the results of the baseline model. However, with three firms the average aggregate profit gain is still above the competitive level.

π_L^c	π_H^c	π_L^m	π_H^m	$\bar{\pi}^c$	$\bar{\pi}$	$\bar{\pi}^m$
0	0.02	0.25	0.12	0.01	0.12	0.19

Table 9: Average aggregate profits with three firms. Results are in case of $k = 12$ price intervals, low marginal cost $c_L = 0$, high marginal cost $c_H = 0.3$, probability $\rho = 0.5$, discount factor $\delta = 0.95$, $T = 500,000$ periods and $S = 300$ simulations

6 Conclusion

This article presents an extension of Klein (2021) by allowing fluctuating marginal costs. It was shown that when the sequential Q-learning algorithms face structural uncertainty, they converge to a pricing pattern that resembles to an Edgeworth Cycle. This pricing pattern push average prices above their competitive level and, as a result, it leads to supracompetitive profits. This finding is robust with respect to parameter changes and various extensions. Therefore, uncertainty it is not a factor that prevents sequential Q-learning algorithms to achieve collusive outcomes. Nevertheless, this paper presents several practical limitations. First, I assume a specific demand function that does not cover all potential markets. Similarly, 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. Finally, 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. Thus, it is necessary to study whether humans and machines can learn to cooperate to achieve collusive outcomes in the setting considered in this paper.

References

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.

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

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.

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.

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