Collusive Outcomes in a Sequential Pricing Duopoly with Q-Learning Agents[*]

Côme Campagnolo[†]

21st April, 2024

Abstract

Recently, automated pricing algorithms relying on reinforcement learning methods achieve to reach outcomes above the competitive benchmarks in oligopoly markets. This occurs although there is no communication between pricing algorithms. This recent evolution of pricing systems, also known as algorithmic collusion, challenges the current understanding of collusive behaviors when games played by automated agents.

Using Q-learning methods, we investigate how the size of the price grid influences automated agents' pricing strategies in a sequential pricing duopoly. The simulation models this market design as a dynamic programming problem of an infinitely repeated sequential pricing game, which is inspired by Maskin and Tirole (1988) and its adaptation to Q-learning by Klein (2021). Our findings suggest that the size of the price grid affects AI-pricing agents' average profits and their behaviors, especially the way a player chooses to undercut his rival or not.

These results not only widen theoretical knowledge of automated economic agents but are also crucial for real-world markets where automated agents predominate, such as in financial trading and online retail. A better understanding of automated-pricing behaviors would help to renew the regulatory approaches to AI-driven markets.

[*]Supervision by Thierry Foucault, HEC Foundation Chaired Professor of Finance

[†]Master of Economics (M1), ENS Paris-Saclay, France

Keywords: Artificial Intelligence, Collusion, Markov Decision Process, Pricing Algorithms, Q-learning, Sequential Duopoly.

Contents

1 Introduction

Recently, the use of algorithmic pricing has widely spread across data-driven marketplaces, such as financial markets or online retailers, but also for material supermarkets or gasoline retailers (Assad et al., 2020). This surge in usage is attributed to the ability of pricing algorithms to adjust prices more frequently and to swiftly react to the evolution of the market, which should make the pricing mechanism more efficient and precise. Algorithmic pricing offers several pro-competitive effects. On the demand side, they estimate more accurately consumers' willingness to pay, and on the supply side, they adapt more quickly to the variations of supply conditions. However, there are two major issues for the consumers: price discrimination, because firms use consumer data to discriminate consumers more accurately, which could lead to a reduction of consumer welfare, and algorithmic collusion. AI pricing algorithms have shown the ability to reach supra-competitive prices, deviating from outcomes predicted by the economic theory. Notably, this seems to happen without communication between the agents. Q-learning algorithms are well suited to model this phenomenon as their initial aim is to solve dynamic stochastic optimization problems (Watkins, 1989). Various oligopoly models of repeated price competition based on Q-learning algorithms have been developed in order to understand how collusive strategies can emerge although they are model-free, i.e. they do not have any prior knowledge about the market structure. We will consider the dynamic programming problem of an infinitely repeated sequential pricing game from Maskin and Tirole (1988), which is adapted to Q-learning agents by Klein (2021)'s paper. Algorithmic pricing is not a new phenomenon, but two generations of pricing algorithms can be distinguished. The first generation, known as "adaptative" algorithms, aimed to maximize profits based on predefined market models. Collusion in these algorithms generally takes the form of a "collusion by code", that is to say that the code can reveal an intention to adopt collusive strategies, which is a keypoint for the legal interpretation of collusion. The second generation, the AI-pricing algorithms, is based on deep-learning methods and do not require prior knowledge of the market in order to compete through prices (Calvano et al, 2019). The Q-learning method that we will use is a basic reinforcement learning method, and we are not going to consider more sophisticated or 'state-of-the-art' algorithms, as we prefer to have an easily understandable algorithm for the purpose to discuss the theoretical implications. In contrary to the adaptative ones, the collusion between such AI agents is more subtile because, to some extent and under specific conditions, they autonomously learn anti-competitive behaviors. They can set supra-competitive prices without explicitly communicating with one another, and even without knowing the existence of other agents outside through the effect of their actions on the price, that is to say without having any previous knowledge about collusion. The antitrust legislation struggles with such cases, as it primarly focuses on proving a will to collude.

2 Literature on Algorithmic Collusion

The literature on AI pricing and algorithmic collusion is relatively recent and dynamic, as many papers were published from the late 2010s to the present. There are already several experimental papers, but relatively few empirical papers. While there is substantial evidence that pricing algorithms are widely used, especially on online markets (Chen et al, 2016), there are relatively few empirical papers on cases of collusion between AI algorithms, except those of Brown et al. (2020), Musolff (2021) and Assad et al. (2023). The first empirical analysis on this topic, based on the shift from mechanically-set prices to algorithmic pricing in Germany's retail gasoline market, concludes that the adoption of AI had a significant effect on pricing (Assad et al, 2023). In duopoly markets (two gasoline stations near to each other) where both competitors adopted AI-pricing, the market level margins increased by an average of 28%, whereas there was no significant change when only one of them adopted AI-pricing. It clearly shows that the emergence of tacit collusion is very dependent on the number of competitors (monopoly, duopoly, etc.) and on the adoption of the AI pricing technology by one or both competitors. It seems that tacit collusion emerges more easily when both competitors use algorithmic pricing. There are many precautions to take in order to empirically measure and make the distinction between the effect of collusion and other various price effects. The asymmetry in pricing technology (price adjustment frequency, commitment, etc.) has a considerable impact on the prices, as shown in the case of over-the-counter allergy medications for online retailers (Brown et al, 2020). This underlines a limit of most models in the literature which assume symmetry to avoid this additional layer of complexity. This lack of empirical evidence has many reasons, like the difficulty to access to the information, as firms are unwilling to share their algorithms, the emergent nature of the phenomenon and the methodological difficulty to distinguish the effects of collusion from other price effects. The practical consequences of algorithmic pricing on the competitivity of many markets are numerous in the literature between economics and law. The literature focuses on the sustainability of algorithmic collusion in real markets, diverging from the debates among computer scientists and economists about its theoretical sustainability (Ezrachi, Stucke, 2019). There is here a divergence between academic fields, as the legal literature tends to consider more clearly that there is indeed forms of sustainable algorithmic collusion on many real-life markets, whereas the economics literature (as well as the computer science literature) underlines that such equilibria are not easily achievable (Schwalbe, 2018). For the juridiction, the main issue about algorithmic collusion is the difficulty to prove its intentionality in the case of AI pricing. The legal literature agrees on the fact that algorithmic collusion is not a violation of the antitrust law and that the current juridiction is not adapted to this situation (Harrington, 2018). Even if tacit collusion is observed, it is harder to control for competition authorities and there is a de facto area of legality for anticompetitive behavior, as documented by the OECD in 2017. Solutions have been proposed, such as ex ante auditing to prevent collusive outcomes, and holding businesses accountable for the automated decisions of their pricing algorithms, enforcing a "compliance by design" rule (Marty, 2017). Some research papers suggest that a platform can indeed be designed so as to improve competition in the presence of AI pricing algorithms (Johnson et al, 2023). However, the risk of creating suboptimal systems that restrict rational behaviors unrelated to tacit collusion remains a valid concern.

3 Model

3.1 Economic Framework: A Sequential Pricing Duopoly

We simulate a sequential pricing duopoly inspired from the setting of Maskin and Tirole (1988). **Market Design.** This model is characterized by a market demand curve $D(p) = 1–p$. Two firms $i \in (1, 2)$ are competing on a market and update their prices alternatively. We assume the production to be costless. The demand for good addressed to firm i, given its own price p_{it} and the other firm's last price p_{jt}, is as follows:

$$D_i(p_{i,t}, p_{j,t}) = \begin{cases} 1 - p_{i,t} & \text{if} p_{i,t} < p_{j,t} \\ 0.5(1 - p_{i,t}) & \text{if} p_{i,t} = p_{j,t} \\ 0 & \text{if} p_{i,t} > p_{j,t} \end{cases} \tag{1}$$

If the price fixed by the firm is lower than the other firm's price, firm i takes all the demand. Conversely, if the price fixed by the other firm is lower, the demand drops to 0. When prices are equal, the demand is equally split between both firms. Then, profits for firm i at time t are $\pi_i(p_{it}, p_{jt}) = p_{it} \cdot D_i(p_{it}, p_{jt})$ for $i, j \in (1, 2)$.

This economic setting can be seen as a Markov Decision Process. 2 agents sequentially set prices and receive rewards (profits) depending on their decision.

Definition 1. *Markov Decision Process (MDP)*

A Markov Decision Process is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ *with:*

- $\mathcal{S}$ *a finite set of states*

- $\mathcal{A}$ *a finite set of actions*

- $\mathcal{P}$ *a state transition probability matrix*[1]*:* $\mathcal{P}^a_{s,s'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$

- $\mathcal{R}$ *a reward function* $([\mathcal{S}, \mathcal{A}] \longrightarrow \mathbb{R}_+)$ *,* $R^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$

- γ *a discount factor* ($\gamma \in [0, 1]$)*.*

Definition 2. *Policy*

A policy (or strategy) π *in an MDP is a distribution over actions given states that defines the behavior of the agent for every situation.*

[1]A matrix that describes transition probabilities from state s to state s^{prime} for $s, s^{prime} \in \mathcal{S}$

$\forall a \in \mathcal{A}, s \in \mathcal{S}, \pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$

Definition 3. *State-value Function*

The state-value function of an MDP gives, for a given policy π the expected return starting from state s:

$$V(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]$$

So the state-value function is a sum of the return for the current period and weighted expected future returns depending on a state and a policy. However, using a **Bellman equation** allows to directly substitute the calculation of the policy π_s into the value function, by backward induction. The objective is to determine a policy such that the state-value function is maximized. For this purpose, we introduce a Q-learning algorithm.

3.2 A Sequential Q-Learning Algorithm

Q-learning is a reinforcement learning method first introduced by Watkins (1989) which aims to find an optimal action-selection policy in infinite-horizon Markov decision processes (MDP). The state is the set of all elements taken into account by the agent when taking a decision.

In this model-free pricing algorithm, the agents do not know elements such as the market demand. They can only decide their prices based on the state, which contains the entire "payoff-relevant" history of prices at a given period t. All states satisfy the following property.

Proposition 1. *Markov Property* $\mathbb{P}[s_{t+1} \mid s_t] = \mathbb{P}[s_{t+1} \mid (s_i)_{i \in [1,t]}]$

Players are assumed to use only strategies under the Markov assumption (Maskin and Tirole, 1988).

Definition 4. *Markov Assumption*

The Markov assumption states that a firm's strategy depends only on the payoff-relevant state, that is to say those variables that directly enter its payoff function.

In our model, the payoff-relevant state is restricted to the price set by the other firm at the previous period. So the strategy of firm i is a dynamic reaction function $R_i(\cdot)$, where in its turn $p_{it} = R_i(p_{j,t-1})$. One Nash Equilibrium is the static reaction function in which firms always price at marginal cost or just over. We may wonder why this restriction on the payoff-relevant history, as we could imagine that further past prices influence the action at period t. This is explained by the fact that we use the economic framework from Maskin and Tirole (1988) which focuses on short-run commitments, and makes the hypothesis that the current behavior is not heavily influenced by past actions.

Then, the objective is to find a form of Nash Equilibrium that satisfy this assumption, that is to say a Markov-Perfect Equilibrium.

Definition 5. *Markov Perfect Equilibrium (MPE)*

A pair of dynamic reaction functions (R^1, R^2) [2] *constitutes a Markov Perfect Equilibrium (MPE) if and only if:*

(i) Given a_{2k-1}*,* $a_{2k} = R^2(a_{2k-1})$ *maximizes firm 1's intertemporal profit at any time* k*, and assuming that each firm* i *moves according to* R^i

(ii) the respective condition is respected for firm 2

Proposition 2. *A Markov-Perfect Equilibrium (MPE) is a subgame perfect Nash Equilibrium with regard to the Markov assumption.*

The state, or “payoff-relevant” history, is restricted in this model to the price chosen by the other player at the previous episode. This makes a difference with the algorithm from Calvano et al. (2020) in which the state corresponds to the set of past prices in the last k periods for all players ($s_t = \{\mathbf{p}_{t-1}, ..., \mathbf{p}_{t-k}\}$) and they use in their experiment a one-period memory. In our framework, action space and state space are therefore identical, because both players have the same action spaces. This keeps the computation process relatively light, as for each player i we have $|\mathcal{A}| = m$ and $|\mathcal{S}| = m$, which corresponds to m^2 action-state pairs. In a model with a broader payoff-relevant history, for example a k-period memory, $|A| = m$

[2]These dynamic reaction functions are also called Markov strategies

and $|S| = m^k$, there would be m^{k+1} action-state pairs (m^3 for one-period memory with 2 players). This leads to an exponential growth of the computation process[3].

Then, the Bellman equation is as follows:

Definition 6. *Bellman Equation*

$$V_i(p_{jt}) = \max_p \left[\pi_i(p, p_{jt}) + \mathbb{E}_{p_{j,t+1}}\left[\delta\pi_i(p, p_{j,t+1}) + \delta^2 V_i(p_{j,t+1})\right]\right] \tag{2}$$

Q-learning is an adaptation of this dynamic programming problem to reinforcement learning. Each player has a Q-table, a matrix containing all action-state pairs. The action space is a finite set of equally and discretely distributed prices available for the agent, such as $(0, 0.2, 0.4, 0.6, 0.8, 1)$ for example, or more generally $\forall n \in \mathbb{N}', P(n) = (k/n | k \in [0, n])$. At the beginning of the game, each player's Q-table is randomly initialized. At each period, the agent chooses an action and receives a "reward" from the environment (capturing the market demand). Based on this reward, the agent updates the value of this action-state pair. The updating rule is as follows:

$$Q_i(p_{i,t}, p_{j,t}) \leftarrow (1-\alpha)\cdot Q_i(p_{i,t}, p_{j,t}) + \alpha\cdot[\pi(p_{i,t}, p_{j,t}) + \gamma\cdot\pi(p_{i,t}, p_{j,t+1}) + \gamma^2\cdot\max_p Q_i(p, p_{j,t+1})] \tag{3}$$

$Q(p_{it}, p_{jt})$ represents the discounted previous estimate of the Q-value for the considered action-state pair. $\max_p Q_i(p, p_{j,t+1})$ represents the maximal Q-value for the next state. α is the learning rate of the algorithm. It determines how fast the algorithm learns from its observations. If this rate is too low, the agent will learn slowly, but if it is too fast, there might be an issue of over-fitting to the observations. γ is the discount rate that applies to future rewards. If $\gamma = 0$, agents only consider current profits. This parameter is generally close to 1. In this updating rule, future rewards are considered at two different episodes and become worth much less at a faster rate. This might be interpreted as a relatively high level of uncertainty or risk aversion.

[3]The same issue occurs when considering n players.

As the agent wants to maximize its total reward, and not only its reward at the current period, it should not always choose the action with the highest Q-value, especially at the beginning of the game, when it has little knowledge of the reward-matrix. A key factor in the learning process is the ability of the agent to gather information about the rewards of its possible actions in the various states, which implies that it needs to experiment new actions, especially in the early stage of the simulation. Therefore, the agent has an exploration strategy, i.e. it tests new actions, deviating from its current best estimate of optimal prices to experiment with different pricing strategies. This allows the agent to gather more information and potentially uncover better solutions, but at the cost of not exploiting the action with the highest-known reward. As its main goal remains to maximize the total reward, it also wants to exploit the action which has the highest expected value in a given state. As the agent gathers more and more information, the probability of uncovering better solutions decreases and the cost of opportunity of exploration becomes higher. Then, the agent will exploit its information more frequently. Exploitation refers to the agent's strategy of setting prices that it perceives as maximizing its expected reward (profit) based on its current knowledge. This trade-off is also known as exploitation-exploration dilemma. We can implement different rules for choosing between exploitation and exploration. The most basic one is the ϵ-greedy rule.

$$p_{it} \begin{cases} \sim U\{P\} & \text{with probability} \varepsilon_t \\ = \operatorname{argmax}_p Q_i(p, s_t) & \text{with probability} 1 - \varepsilon_t \end{cases} \tag{4}$$

This rules defines exploration as choosing any random price. More sophisticated decision rules allow to explore more specifically and to target parts of the Q-table. It has been pointed out that an insufficient exploration rate during the learning phase is one of the factors that decrease competition between Q-learning algorithms, suggesting that algorithmic sophistication could increase competition (Abada et al., 2023). There are also propositions of models that implement Q-values-based policies with a bias towards cooperation for example, so that they do not always choose the action associated to the highest Q-value (Compte, 2023). We stick to the basic ϵ-greedy rule.

Several papers such as Calvano et al. (2020) or Asker et al. (2021) consider a simultaneous pricing duopoly, but it is more likely in reality that pricing algorithms do not update their prices simultaneously. We consider a game in which players update their prices sequentially, that is player 1 can change its price in odd periods and player 2 in even periods. This aims to model the fact that a player commits to its action for a finite period of time while the other player can change its action, and that it reacts to the actions of its competitor. Here, each player's action is a commitment for 2 periods. This sequential pricing game seems closer to the conception of pricing strategies in the industrial organization literature than an infinitely repeated simultaneous game. Who moves first does not have an effect on the simulation.

As the agents play more and more action-state pairs during the simulation, they will progressively get better estimations of the Q-values, until each player's strategy stabilizes. The updating rule is a key element, the function which gives the expected rewards for every possible action in a given state. In comparison with other types of algorithms, Q-learning is relatively simple and transparent. The algorithm has zero prior knowledge, which means it only knows its set of available actions (the price grid) and observes the payoff from these actions afterwards in order to learn, that is to say to update its Q-values. A key point for us is to get a deeper insight on the pricing strategies of Q-learning agents depending on how fine the price grids are.

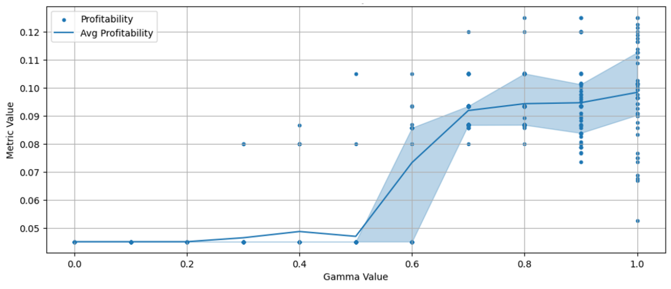
Over time, the agent will have a better guess of the rewards of each action-state pair and become able to choose prices that maximize its total reward over the entire game, with a discount factor for future profits: $\max \sum_{s=0}^{\infty} \delta^s \pi_i(p_{i,t+s}, p_{j,t+s})$.

3.3 Simulation Setup

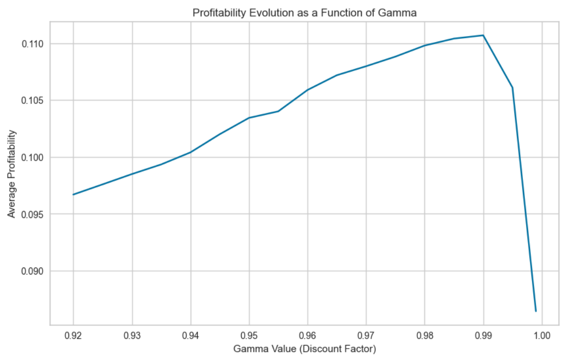
The main performance metric is the profitability, defined as half of the average joint-profit of both players during the last 1000 episodes of the simulation.

Definition 7. *Profitability*

$$\Pi_i = \frac{1}{2} \cdot \frac{\sum_{t=T-1,000}^{T} \pi_i(p_{i_t}, p_{jt})}{1,000} \tag{5}$$



(a) Discount Factor (0 to 0.99)



(b) Discount Factor (0.9 to 0.999)

Figure 1: Profitability Depending on Discount Factor

The representative simulation uses the following parameters for the Q-learning setup: $\alpha = 0.3$ (learning rate), $\gamma = 0.95$ (discount factor), $\epsilon = \delta^t$ with $\delta = 0.9997879$ (exploration rate). These parameters are fixed so as to have an efficient learning process without a prohibitive computation time.

For example, Figure 1 shows that the discount factor has a major impact due to the updating rule of our model. It needs to be sufficiently close to one, as a low discount factor drastically decreases the profits of both players, especially under 0.7. However, taking a closer look at the higher values for the discount factor (Figure 1b), the results suggest that a discount factor too close to 1 ($\gamma > 0.999$) also dramatically decreases the profitability. Thus, we set a value of 0.95.

The exploration rate (ϵ) progressively decreases from 1 to 0.0001 after 100,000 episodes according to the following formula: $\epsilon = \delta^t$ with $\delta = 0.9997879$. For comparison, Calvano et al. (2020)'s paper uses a much lower decay rate of $\delta = 4e-6$, which corresponds to an exploration rate of 0.03 after 2,000,000 episodes, and a lower learning rate $\alpha = 0.15$. We use higher values in order to have shorter convergence process and computation time, while still finding some results. The exploration probability decreases over time, as the agent should have less exploration and more exploitation during the progression of the game. The **decay rate** δ determines how quickly the exploration rate decreases ($\epsilon = \delta^t$). A too large decay rate would increase the probability of converging to sub-optimal rewards.

To compare the profits made by the Q-learning agents to the economic theory, we set two benchmarks: a monopoly benchmark and a dynamic competitive benchmark. In this setting, the perfect collusion or joint-profit maximizing price is $p^C = 0.5$. In this case, each firm makes a profit of $\pi_i = 0.125$. This corresponds to the monopoly benchmark.

The second benchmark is a competitive benchmark, but not the marginal cost competitive benchmark of a one-period duopoly problem. It has to be suited to the sequential pricing game with infinite horizon. For this purpose, we use Edgeworth price cycles, which are the most competitive Markov-Perfect Equilibrium (MPE) identified by Maskin and Tirole (1988). It can be considered as a dynamic alternative to the competitive equilibrium of a Bertrand model, in which prices equal marginal costs. This benchmark will be furthered explained when analyzing the non-constant pricing strategies.

4 Results

4.1 Convergence properties of Q-learning algorithms

Watkins and Dayan (1992) gave the theoretical proof of convergence in the case of a single player. They assume that the evolution of states adheres to a Markov Decision Process, where each state-action pair in one episode has a probability to transition to a state in the next episode and the algorithm has no prior knowledge of these transition probabilities. However, for multiple-agents situations, the results are more nuanced. Hu and Wellman (1998) extended the demonstration of convergence under specific conditions for stochastic games with multiple-agents scenarios. As there is no theoretical guarantee of convergence in the games played by Q-learning

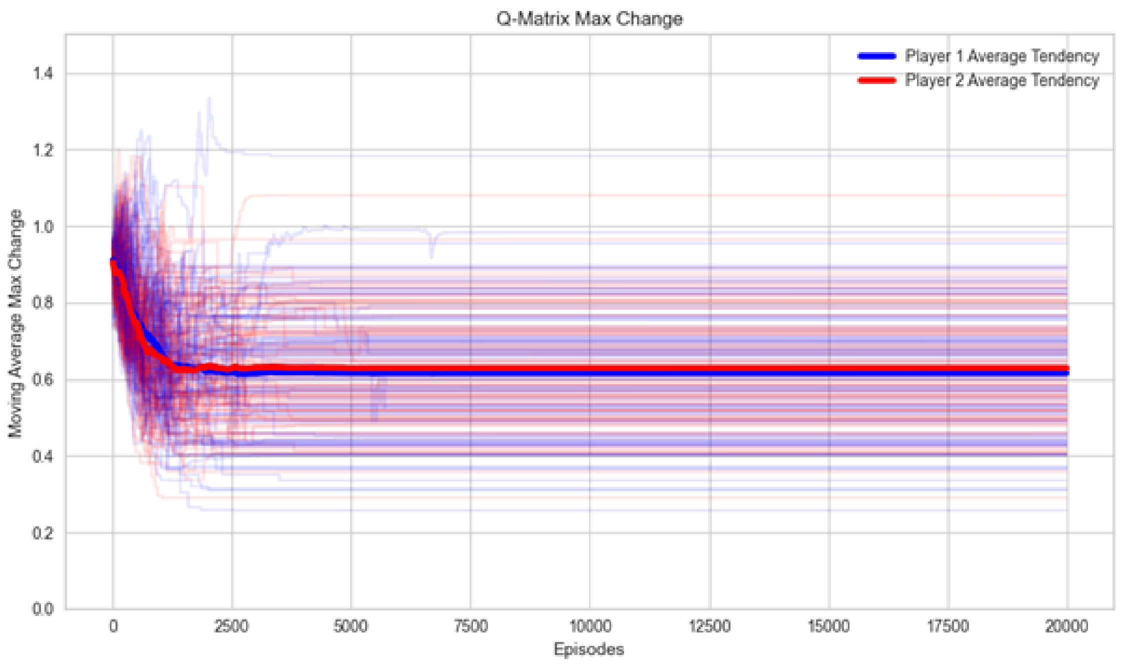


Figure 2: Q-table maximal variation between 2 consecutive periods

algorithms, the literature generally relies on a practical criterion for convergence, for example "if for each player the optimal strategy does not change for 100 000 consecutive periods" (Calvano et al., 2020).

Another element to consider is the stabilization of Q-values. If the maximal change between all the Q-values of a Q-table decreases, we can consider that the algorithm's policy function has become more stable. However, Q-values may stabilize even if the agent did not find the globally optimal policy, because it converged to a sub-optimal policy (Compte, 2023). Stability does not imply optimality of the strategies.

Figure 2 shows that the maximal variation in consecutive periods between Q-values always stabilizes at some point, when all the Q-values have been visited enough times. Consequently, for finer price grids, this stabilization process takes more time, but it is still valid. However, we also find that it never converges to zero, but always to a substantially higher constant. This means that Q-values continue to be updated by a significant amount at each period, even for a very large number of periods. This is probably a consequence of the non-stationary environment, so that each agent continuously adapts its policy to its rival.

4.2 Frequency and profitability of collusive outcomes depending on the size of the price grid

From an economic point of view, the price grid – or action space for the algorithm – matters a lot. In security markets for example, traders have to post their quote on a pre-determined grid. There is generally a mandatory minimum price variation that is chosen by market organizers and it is a key factor of the determination of the best ask and bid prices (Cordella and Foucault, 1999). The size of the price interval can affect the pricing strategies of the players in our simulation, as well as it does affect the economic behaviors of agents in real-life markets. Prices are equally and discretely distributed in [0, 1]. The Q-learning algorithm is slow by construction because it updates only one Q-value at each period. This method is called 'asynchronous' updating rule in opposition to the 'synchronous' updating rule, in which case all the Q-values are updated at each iteration (Asker et al., 2021). Moreover, the action space needs to be discretized, as Q-learning only works in a finite action space. To consider continuous action spaces, we would need to implement more sophisticated mechanisms using approximation functions. Discrete action spaces with a large number of values make the convergence process increase quadratically, due to the size of the Q-tables. For example, if we set a price interval of 0.1, the action space of each player contains 11 prices equally distributed in $[0,1](\mathcal{A} = \mathcal{S} = (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1))$, and a Q-table of size 121. For a price granularity of 0.01, we have a price grid size of 101, and a Q-table of size 10,201, and so on.

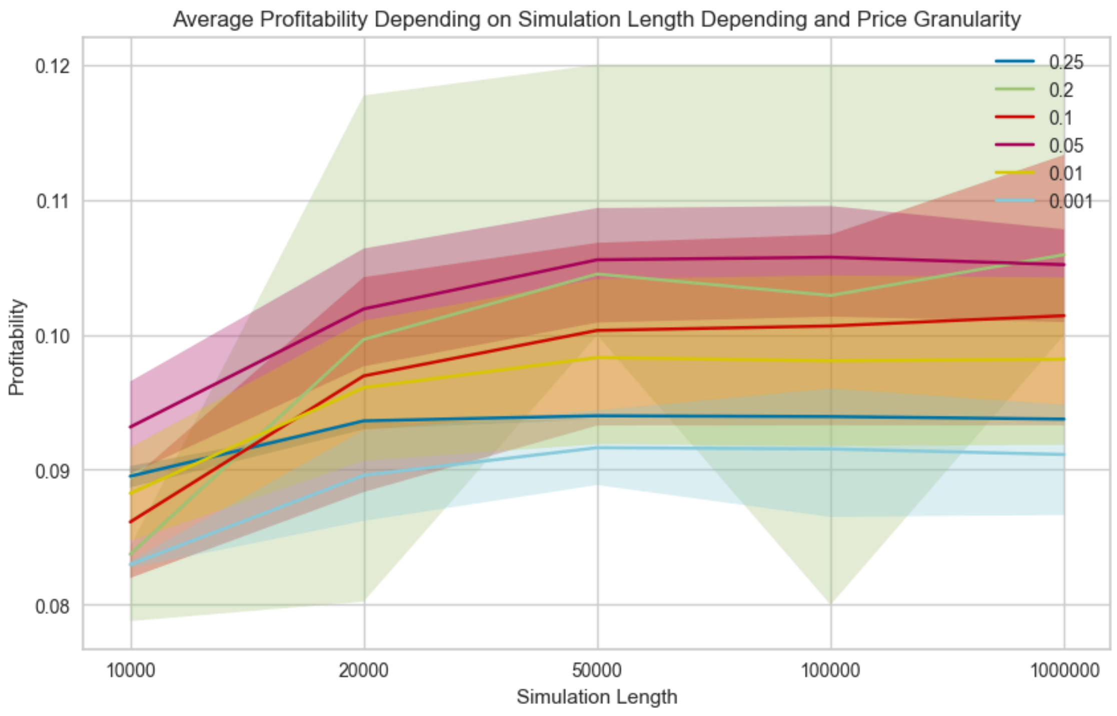


Figure 3: Mean Profitability Depending on Simulation Length and Price Granularity

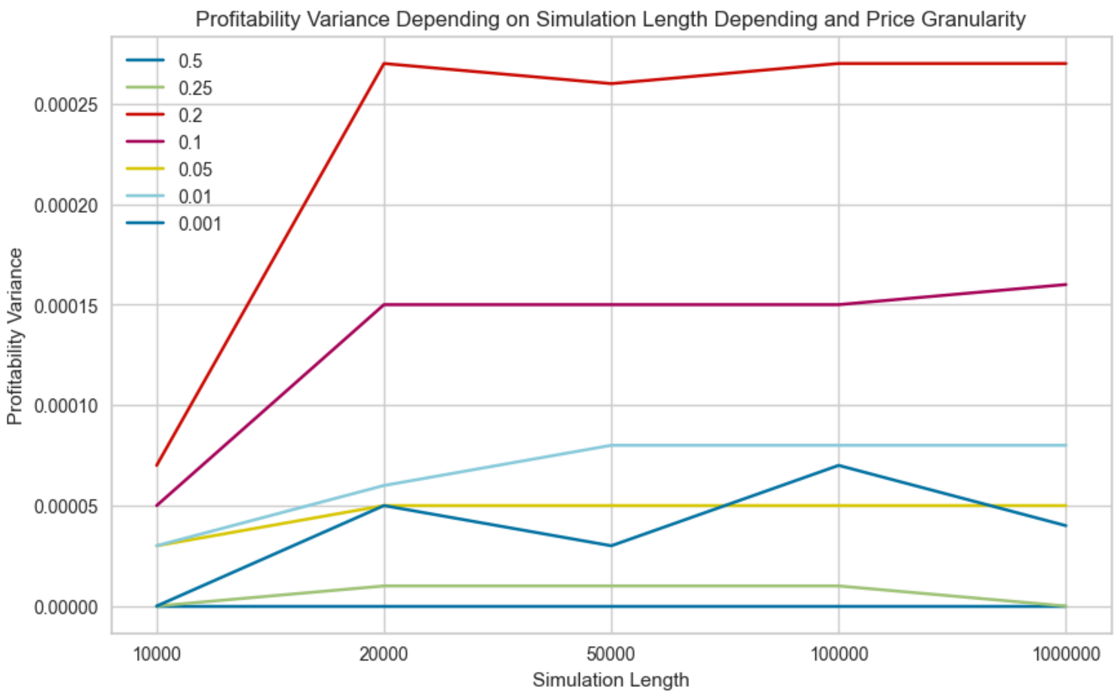


Figure 4: Profitability Variance Depending on Simulation Length and Price Granularity

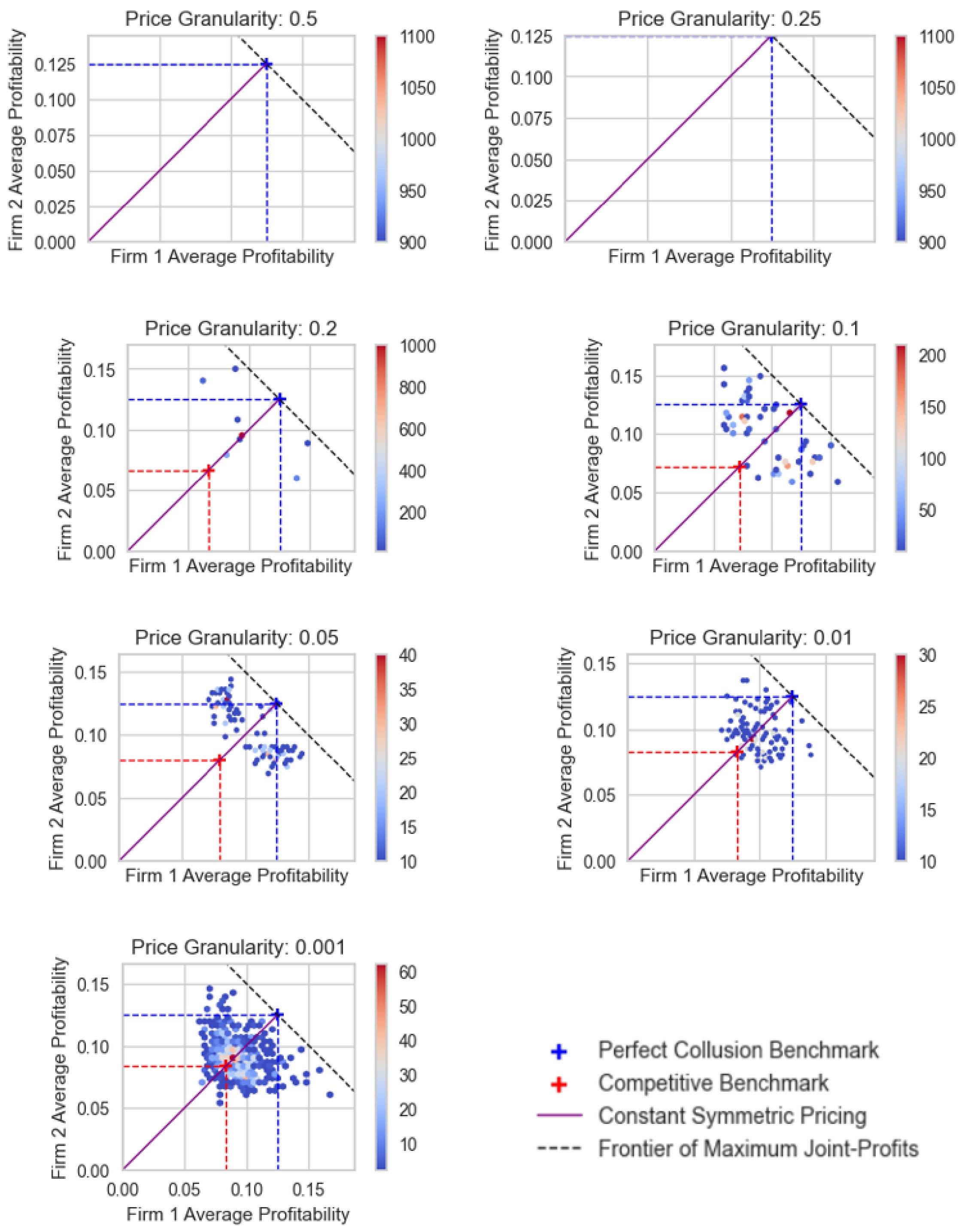


Figure 5: Profitability Distribution After 1 Million Episodes Depending on the Price Grid Size

For a large majority of simulations, we find that the agents converge to outcomes above the competitive benchmark, always in average above 0.09 for every price grid size after 1,000,000 episodes. As the simulation length increases, the average profitability increases. This confirms that the learning process is effective, and agents tend to converge to more profitable pricing strategies. The very slight increase between 100,000 and 1,000,000 episodes could be linked to finer adjustments that occur over very long periods, but it could also be a consequence of the Q-learning parameters, such as the exploration rate which continues to decrease. The detailed results in Table 7.2 show that even the first quartile of profitability is rarely under 0.09 after a high number of episodes, except for a price granularity $k = 0.001$.

We also find that the pricing grid does have a significant impact on the level of profitability: a price grid with 21 prices (0.05) reaches the highest value overall, whereas price grids with 5 values ($k = 0.25$) or with 1001 values ($k = 0.001$) have on average a lower profitability. For the price interval 0.5, as the agents can choose only between 0, 0.5 and 1, they converge as expected to the joint-profit maximizing value (0.125, 0.125) (see Table 7.2), which is in this case the only outcome with a strictly positive profit.

Figure 5 describes the profitability distribution across simulations for each player, depending on the price interval. In the Figures 10a and 10b, we get a deeper insight by comparing the evolution of profitability distribution at different stages of the simulation (after 10,000, 20,000, 50,000, 100,000 and 1,000,000 episodes). Depending on the price interval, we see that the distribution of profitability after a given number of episodes vary, as well as the number of possible outcomes. We find that the number of possible outcomes decreases as the simulation length increases, a result which is coherent with the idea of convergence of the pricing strategies over time. The distribution graphs also suggest that smaller price intervals lead to a higher level of dispersion of the outcomes. For price grids with only 3 or 5 prices ($p \in (0, 0.5, 1) or p \in (0, 0.25, 0.5, 0.75, 1)$), the players rapidly converge towards symmetric profits (displayed on the purple line). When the price interval increases a little more, especially for the price intervals in $(0.125, 0.1, 0.05)$, we observe a higher dispersion of the outcomes, which means a higher frequency of asymmetric situation between both players. For even finer price grids ($k \in (0.01, 0.001)$), we see again some level of concentration around the symmetric profitability levels. However, in contrary to the case with $k \in (0.5, 0.25)$, there is still a lot of dispersion. This reveals that players have closer asymmetric profits than for the previous price grids, but still asymmetric profits. The fact that players reach collusive outcomes with symmetric or asymmetric profitability more often for some price grids than for others suggest the existence of specific pricing strategies that appear more or less according to the price grid size.

4.3 Pricing Strategies

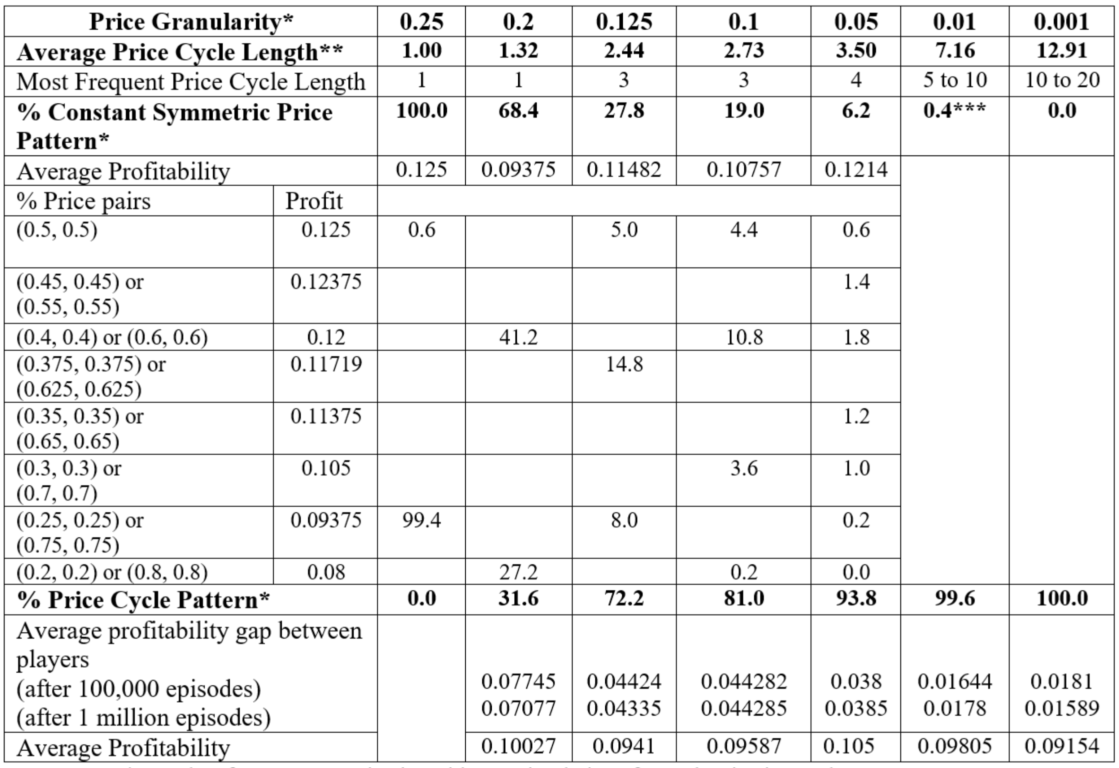
The results show that players can converge to different types of policy depending on price granularity. These policies can be analyzed through the pricing patterns of each player at the end of the game.

4.3.1 Pricing Cycles Analysis

It stems from the Table 6 and Figure 7 that the more prices are available to each player, the less they converge to constant pricing strategies. The price cycles consistently get longer as the price grids becomes finer, which could be interpreted as players adopting more detailed strategies. It is also noteworthy that this relation closely follows a log-linear trend (Figure 8). Among the various pricing patterns observed throughout the simulations, two frequent patterns hold our attention: **constant pricing** and **undercutting price cycles** - or 'price war' -.

4.3.2 Constant pricing strategies

In this pricing pattern, both players always choose the same action in the end phase of the game. Figure 6 shows that, for a price granularity $k \geq 0.25$, players almost always converge to a constant pricing strategy. As the price granularity decreases to 0.2 (6 available prices), we see for the first time non-constant pricing strategies, but constant pricing is still the most frequent outcome. However, for a price grid with 9 prices ($k = 0.125$) or more, it is not the case anymore. For a price



Price Granularity*		0.25	0.2	0.125	0.1	0.05	0.01	0.001
Average Price Cycle Length**		**1.00**	**1.32**	**2.44**	**2.73**	**3.50**	**7.16**	**12.91**
Most Frequent Price Cycle Length		1	1	3	3	4	5 to 10	10 to 20
% Constant Symmetric Price Pattern*		**100.0**	**68.4**	**27.8**	**19.0**	**6.2**	**0.4*****	**0.0**
Average Profitability		0.125	0.09375	0.11482	0.10757	0.1214		
% Price pairs	Profit							
(0.5, 0.5)	0.125	0.6		5.0	4.4	0.6		
(0.45, 0.45) or (0.55, 0.55)	0.12375					1.4		
(0.4, 0.4) or (0.6, 0.6)	0.12		41.2		10.8	1.8		
(0.375, 0.375) or (0.625, 0.625)	0.11719			14.8				
(0.35, 0.35) or (0.65, 0.65)	0.11375					1.2		
(0.3, 0.3) or (0.7, 0.7)	0.105				3.6	1.0		
(0.25, 0.25) or (0.75, 0.75)	0.09375	99.4		8.0		0.2		
(0.2, 0.2) or (0.8, 0.8)	0.08		27.2		0.2	0.0		
% Price Cycle Pattern*		**0.0**	**31.6**	**72.2**	**81.0**	**93.8**	**99.6**	**100.0**
Average profitability gap between players (after 100,000 episodes) (after 1 million episodes)			0.07745 0.07077	0.04424 0.04335	0.044282 0.044285	0.038 0.0385	0.01644 0.0178	0.0181 0.01589
Average Profitability			0.10027	0.0941	0.09587	0.105	0.09805	0.09154

* Results after 100,000 episodes with 500 simulations for each price interval

** Results after 1,000,000 episodes with 100 simulations for each price interval

*** Non-significant

Figure 6: Pricing Patterns Depending on Price Granularity

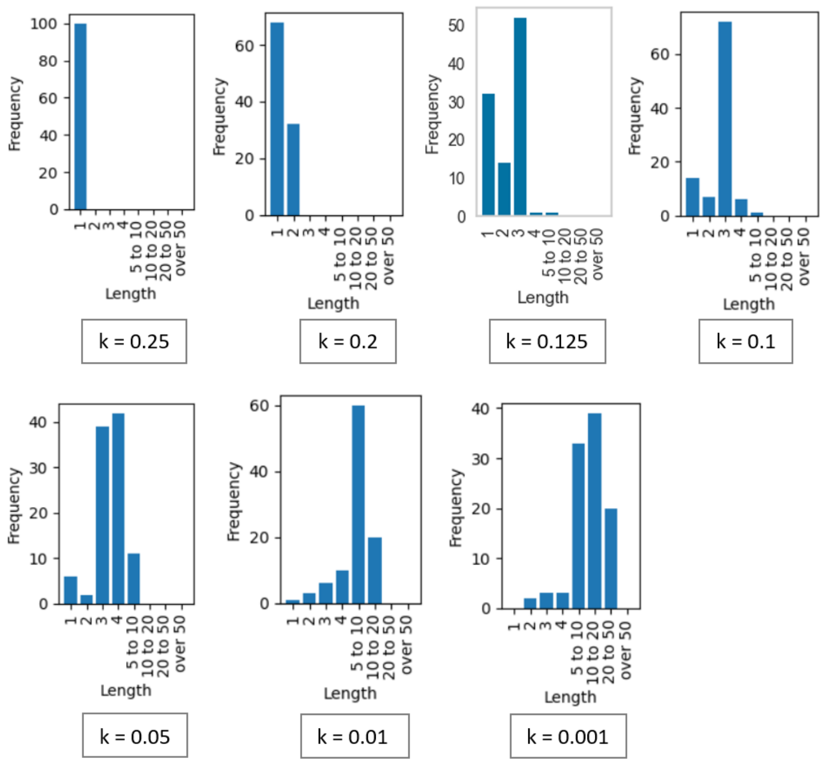


Figure 7: Price Cycle Length Frequency Depending on Price Granularity

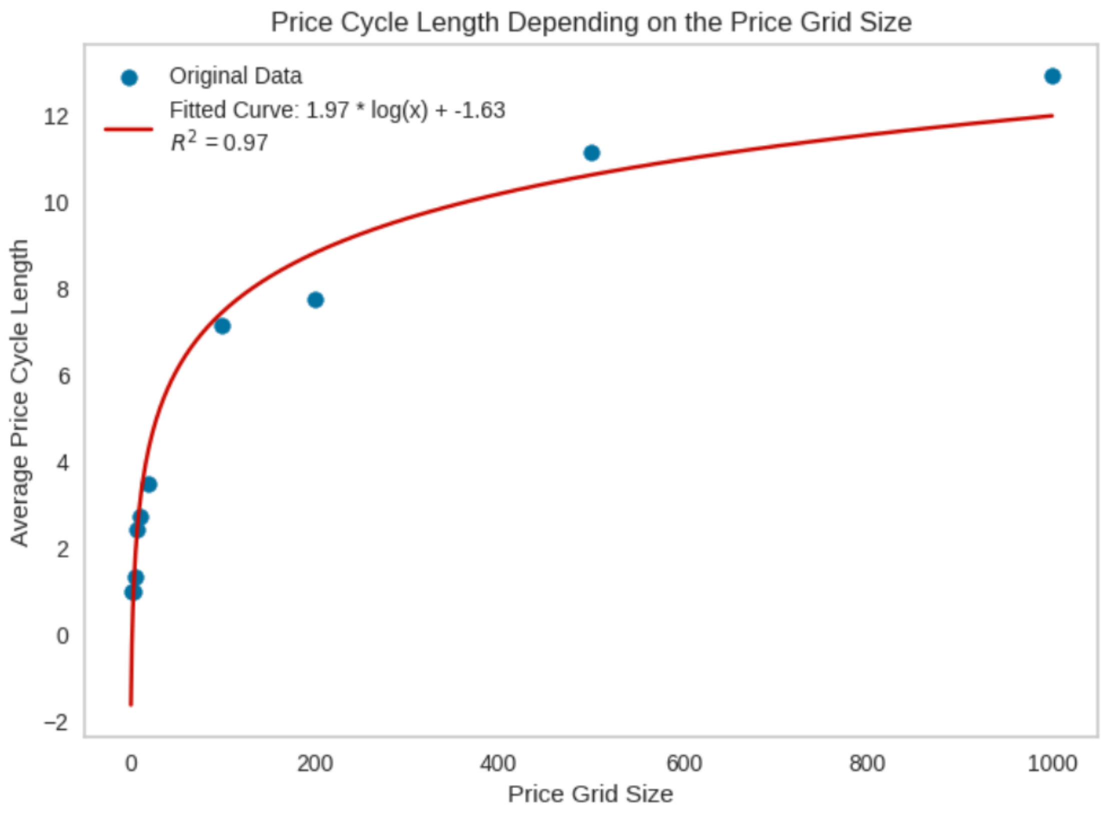


Figure 8: Price Cycle Length Depending on the Price Grid Size

grid with 21 prices ($k = 0.05$), it becomes a very unlikely outcome that occurs in only slightly more 1 out of 20 simulations. For an even more detailed price grid (101 prices or 1,001 prices), constant pricing disappears from the outcomes. A constant price equilibrium is characterized by a strictly positive profit, so it has to be strictly positive: we never find the outcomes $(0, 0)$ (the Bertrand Price model Nash Equilibrium) or $(1, 1)$.

Definition 8. *Constant Symmetric Pricing*

Let $\mathcal{P}$ be the action space, $R_i(p)$ the reaction function of player $i \in (1, 2)$.

A constant symmetric[4] pricing strategy is defined by the following dynamic reaction functions: $\exists! p^{prime} \in \mathcal{P}, p^{prime} = R_1(p^{prime}) = R_2(p^{prime})$ and,

$\forall p \in \mathcal{P} \backslash (p^{prime}), \exists k \in \mathbb{N}, R_1^{(k)}(p) = R_2^{(k)}(p) = p^{prime}$ where$R_i^{(k)}(p)$ is the set of the possible reactions to a price p after k periods.

Proposition 3. *A constant symmetric price strategy is a Markov Perfect Equilibrium.*

Proof. Let P the action space (set of available prices), (p_1^*, p_2^*) a Nash Equilibrium of the duopoly pricing game. Assume both players played this action-pair at the last period.

Q-learning agent 1 applies the updating rule to its Q-value.

$Q_{1,t}(p_1, p_2^*) = (1-\alpha) \cdot Q_{1,t}(p_1, p_2^*) + \alpha \cdot [\pi(p_i, p_j^*) + \delta \cdot \pi(p_i, p_{j,t+1}) + \delta^2 \cdot \max_p Q_i(p, p_{j,t+1})]$.

By definition of a Nash Equilibrium: $\forall p \in P, \pi(p_1^*, p_2^*) \geq \pi(p, p_2^*)$ and $\pi(p_2^*, p_1^*) \geq \pi(p, p_1^*)$. Then, we know that $\max_p Q_{1,t}(p) = Q_{1,t}(p_1^*)$. This implies: $\max_p Q_{1,t+1}(p) = Q_{1,t+1}(p_1^*)$.

Q-learning agent 1 will set again the price p_1^* and player 2 will have the same strategy. By induction, both players will play the same action-pair (p_1^*, p_2^*) infinitely, all other things being equal.

(R_1, R_2) is thus a Markov-Perfect Equilibrium. □

In order to test the reaction of the players, we introduce a market deviation, by which player 1 plays a price below the market price once. A market deviation

[4]We never find constant asymmetric outcomes, that is to say outcomes in which the two players would infinitely set two constant different prices.

of this type is often followed by two types of reactions. Either player 2 keeps its price at the same level, in which case player 1 generally comes back to its constant pricing strategy at the next turn, or player 2 undercuts player 1. Then, player 1 reacts in the same way. After a few episodes, both players come back to the initial market price, which is coherent with the constant pricing strategy.

4.3.3 Non-Constant Pricing Strategies and Undercutting Patterns

In the outcomes with non-constant pricing strategies, there is a very frequent pattern of **undercutting**. Players undercut each other until the price reaches a low level, at which point one player eventually reverts to a higher price, and the cycle restarts. We could also call it a 'price war' pattern and it has some similarities with the Edgeworth price cycles that is used as competitive benchmark.

Definition 9. *Edgeworth price cycle equilibrium*

An Edgeworth price cycle equilibrium is a mixed strategy defined by the following dynamic reaction functions:

$$\forall p \in \mathcal{P}\backslash(p_{min}), \forall i, j \in (1,2), R_i(p) = p_j - k$$

(with k the price interval) and if $p = p_{min}$,

$$R_i(p) = \begin{cases} p_{min} & \textit{with a given probability } x \\ p_{max} & \textit{with probability 1-x} \end{cases}$$

The Edgeworth price cycle equilibrium has been identified as the most competitive MPE (Maskin and Tirole, 1988), given a fine enough price grid[5].

This pattern could be interpreted as a form of price war followed by some phase of relenting when both players stay at a low price level waiting for the other one to raise its price. Both would like the price to be higher, but each firm wants the other to do it in order to be able to undercut. As we can see from the definition, the resolution of this free-rider problem is a mixed strategy. However, in the simulations, it is always the same player who resets the price at a higher level. This is a limitation

[5]They considered the case with 7 different prices. This is coherent with our simulations, as for a price interval $k < 0.2$ (i.e. less than 6 prices), we find only constant pricing strategies.

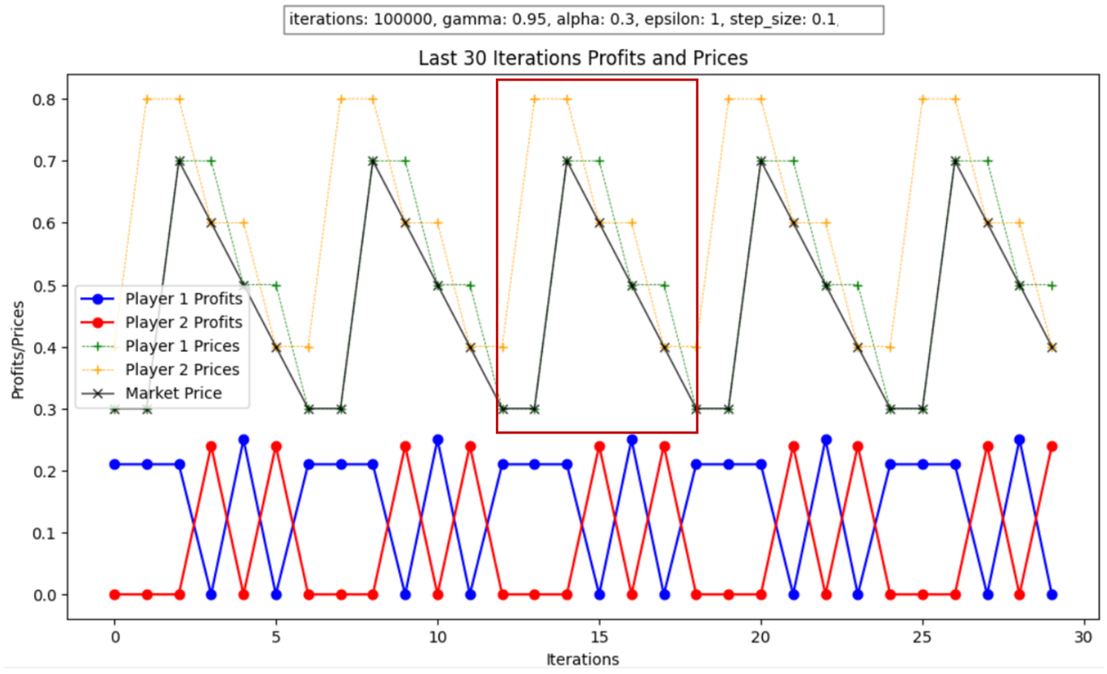


Figure 9: Example of an Undercutting Price Cycle Pattern

of our Q-learning model, as it can only play pure strategies. More sophisticated algorithms based on Q-function approximation could handle mixed strategies[6]. This leads to a strong asymmetry in profits in favor of the firm which does not reset its price. In Figure 9, we picked as example a simulation with a price granularity $k = 0.1$, that is to say prices from 0 to 1 with an increment of 0.1 and looked more closely at the last 30 iterations[7] This example shows that player 1 supplies the market demand 4 out of 6 times in each price cycle and has therefore a per period profit of 0.1467 against only 0.06 for player 2. The dynamic reaction functions played by both players are not exactly the same in an Edgeworth price cycle equilibrium, as the maximal and minimal values are never played, which seems coherent with the fact that profits get smaller as prices are less close to $p = 0.5$.

However, the undercutting patterns found in our simulations are not perfect Edgeworth price cycles, although they share some similarities. The main differences

[6]Methods such as linear function approximation and deep Q-learning can be used to approximate Q-functions. The idea is to approximate the Q-function rather than calculate its exact value. These methods scale better as they eliminate the need for a large Q-table and provide estimates for Q-values that have not been visited.

[7]As each player commits to its action two episodes, 30 iterations correspond to 60 episodes.

are that the price level at which the undercutting starts and the price level at which it resets can vary from one simulation to another, but it is generally not at the highest or lowest price. That means that both players seem to stay in a price area where profits are relatively high. In that way, we find a form of cooperation between both players and these differences are the reason why Q-learning agents frequently yield higher profits than our dynamic pricing competitive benchmark. It is also noteworthy that the increment by which each firms undercuts the other is not necessarily the smallest price interval, and that the undercutting process does not always follow a regular price interval.

5 Conclusion

Introducing Q-learning agents in a sequential pricing duopoly model allowed us to test their ability to reach collusive outcomes for various parameters, with a focus on the price grid. Similarly to the literature, our simulations confirmed that Q-learning agents consistently reach collusive outcomes in this setting. More specifically, this paper focused on the analysis of the size of the price grid, a key feature of market design that is sometimes overlooked compared to other settings.

Our findings suggest that the price grid size does have a significant impact on the mean level of profitability reached in this model. However, a finer price grid does not imply a higher level of collusion or inversely. Taking a deeper insight into the pricing strategies adopted by each player and the pricing patterns at the end of the simulations, we find that the average price cycle length increases as the price grid becomes finer, following a log-linear relationship. Prices have therefore a higher volatility, in the sense that they change more often as the price grid size increases. At the same time, this does not mean that players' strategies are less stable, as they are still robust to market deviations for example. Comparing the results with the economic theory, we find in a significant number of simulations that Q-learning agents learn behaviors that are close to well-known strategies, namely constant pricing and undercutting, although not identical.

As we have seen throughout the results, the presence of collusive outcomes and their extent is very dependent on the parameters and the market setup that is considered. Although we consistently find some level of collusion in our experiment, other modifications of the environment could challenge the robustness of those results. The literature has already shown that algorithmic collusion was robust to specific settings. An interesting question would be to compare the performance of algorithms in symmetric and asymmetric situations, as the literature tends to show that **symmetry** makes collusive strategies easier (Ezrachi, Stucke, 2019). Several papers have also tested Q-learning agents under **imperfect monitoring**, where rivals' strategies are more difficult to observe (Calvano et al, 2021) or with exposure to adverse selection, in the context of securities markets (Colliard, Foucault, Lovo, 2023). Imperfect monitoring can be useful for financial markets or markets for electricity, when aggregate outcomes are easier to observe than actual individual behaviors. Another research direction, closely related to the price granularity parameter that we test, would be to introduce a **delay mechanism** into this sequential pricing duopoly model. Then, we could investigate the impact of the grid size on transaction costs in financial markets, as in Cordella and Foucault (1999). These are only examples of settings that would help to understand which specific features of markets are more - or less - favorable to algorithmic collusion. Having a deeper insight into the strategies of Q-learning agents - and more generally, AI-pricing agents - also allows to search which mechanisms could be implemented in order to adapt markets to this new type of economic agents.

6 References

ABADA, I., LAMBIN, X., TCHAKAROV, N. (2023). Collusion by Mistake. Does Algorithmic Sophistication Drive Supra-Competitive Profits? *ESSEC Business School Research Paper*, No. 2023-01.

ASKER, J., FERSHTMAN, C., PAKES, A. (2021). Artificial Intelligence and Pricing: The Impact of Algorithm Design. *NBER Working Paper*, No. w28535.

ASKER, J., FERSHTMAN, C., PAKES, A. (2022). Algorithmic Pricing: Artificial Intelligence, Algorithm Design, and Pricing. *American Economic Association Papers and Proceedings*, 112, 452-56.

ASSAD, S., CLARK, R., ERSHOV, D., XU, L. (forthcoming). Algorithmic Pricing and Competition: Empirical Evidence from the German Retail Gasoline Market. *Journal of Political Economy.*

BROWN, Z. Y., MACKAY, A. (2020). Competition in Pricing Algorithms. *NBER Working Paper*, No. w28860

CALVANO, E., CALZOLARI, G., DENICOLÒ, V., PASTORELLO, S. (2019). Algorithmic Pricing: What Implications for Competition Policy? *Review of Industrial Organization*, 55, 155–171.

8. CALVANO, E., CALZOLARI, G., DENICOLÒ, V., PASTORELLO, S. (2021). Algorithmic collusion with imperfect monitoring. *International Journal of Industrial Organization*, 79.

CALVANO, E., CALZOLARI, G., DENICOLÒ, V., PASTORELLO, S. (2020). Artificial Intelligence, Algorithmic Pricing, and Collusion. *American Economic Review*, 110 (10), 3267-97.

CHEN, L., MISLOVE, A., WILSON, C (2016). An empirical analysis of algorithmic pricing on Amazon marketplace. *Proceedings of the 25th international conference on World Wide Web*, 1339–1349

CORDELLA, T., FOUCAULT, T. (1999). Minimum Price Variations, Time Priority, and Quote Dynamics. *Journal of Financial Intermediation*, 8 (3), 141-173.

COLLIARD, J.-E., FOUCAULT, T., LOVO, S. (2022). Algorithmic Pricing and Liquidity in Securities Markets. *HEC Paris Research Paper, Working Paper.*

COMPTE, O. (2023). Q learning with biased policy rules. *Paris School of Economics, Working Paper.*

EZRACHI, A., STUCKE, M. (2019). Sustainable and Unchallenged Algorithmic Tacit Collusion. *Northwestern Journal of Technology and Intellectual Property*, 17.

HARRINGTON, J. E. (2018). Developing Competition Law for Collusion by Autonomous Artificial Agents. *Journal of Competition Law & Economics*, 14, 331–363.

JOHNSON, J., RHODES, A., WILDENBEEST, M. (2023). Platform Design When Sellers Use Pricing Algorithms. *Econometrica*, 91 (5), 1841-1879.

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

MARTY, F. (2017). Algorithmes de prix, intelligence artificielle et équilibres collusifs. *Revue internationale de droit économique*, De Boeck Université, 0 (2), 83-116.

MASKIN, E., TIROLE, J. (1988). A Theory of Dynamic Oligopoly, I: Overview and Quantity Competition with Large Fixed Costs. *Econometrica*, 56 (3), 549–569.

MASKIN, E., TIROLE, J. (1988). A Theory of Dynamic Oligopoly, II: Price Competition, Kinked Demand Curves, and Edgeworth Cycles. *Econometrica*, 56 (3), 571–599.

MUSOLFF, L. (2022). Algorithmic Pricing Facilitates Tacit Collusion: Evidence from E-Commerce. *Proceedings of the 23rd ACM Conference on Economics and Computation.*

SCHWALBE, U. (2018). Algorithms, Machine learning, and collusion. *Journal of Competition Law and Economics.*

WATKINS, C. (1989). Learning from Delayed Rewards (Ph.D. thesis). *University of Cambridge*

WATKINS, C., DAYAN, P. (1992). Q-learning. *Mach Learn*, 8, 279–292.

7 Appendix

7.1 Pseudo-code of the Q-learning algorithm for a Sequential Pricing Duopoly

```
Input:
    iterations: Total number of episodes
    gamma: Discount factor for future rewards
    alpha: Learning rate for the udpating rule of Q-values
    epsilon: Exploration rate
    step_size: Price interval
---------------------------------------------------------------------------
Procedure:
        Define a set of possible actions using step_size.
        Create Q1 and Q2 with random values for Players 1 and 2.
        Function action_selection(Q, state, epsilon):
            With probability epsilon, choose an action randomly. (Exploration)
            Otherwise, select the action with the highest Q-value
            for the current state. (Exploitation)


    For t from 1 to iterations:
        Adjust epsilon using the decay rate delta.


        For Player 1:
            Calculate the Q-value for the current state-action pair:
                Q1[action1, previous_action2] =
                    (1 - alpha) * Q1[action1, previous_action2] +
                    alpha * (profit(current price, opponent's previous price) +
                    gamma * future profit estimate + gamma^2 * max future Q-value)


            Update Player 1's action with action_selection(Q, state, epsilon).
            Store the chosen prices and compute profits.
        For Player 2 (next period t+1):
            In a similar way, update Q2, choose the next action
            based on the updated policy and store prices and profits.

        Update states for the next iteration:
            Current actions become previous actions for the next period.

    Compute average profitability.
```

7.2 Tables and Graphs

Average Profitability, Q1, Q3 and Profitability Variance Depending on Simulation Length and Price Granularity

Table 1: Average Profitability Depending on Simulation Length and Price Granularity

Price Interval	0.5	0.25	0.2	0.1	0.05	0.01
Simulation Length						
10000	0.114248	0.089521	0.086632	0.086132	0.093155	0.088252
20000	0.123749	0.093619	0.096632	0.096948	0.101913	0.096076
50000	0.124997	0.093999	0.099252	0.100322	0.105551	0.098314
100000	0.125	0.093937	0.098425	0.10065	0.10574	0.098068
1000000	0.125	0.09375	0.099842	0.101413	0.105191	0.098194

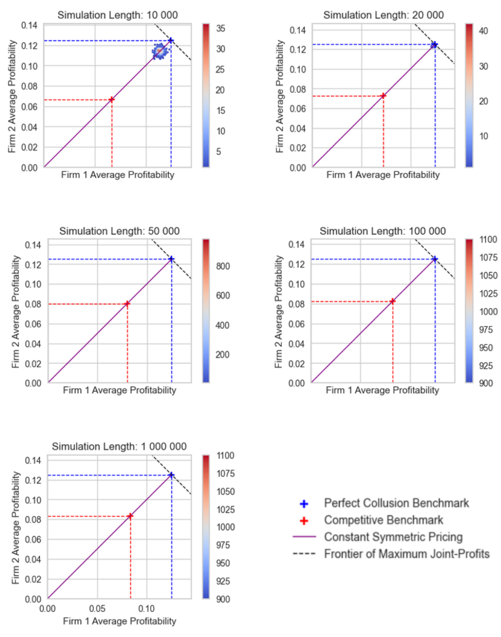
(n = 1000 for simulations with less than 100,000 episodes, n = 500 simulations with 100,000 episodes and n = 100 simulations with 1,000,000 episodes for each price interval)

Table 2: Profitability Variance and Quartiles Depending on Simulation Length and Price Granularity

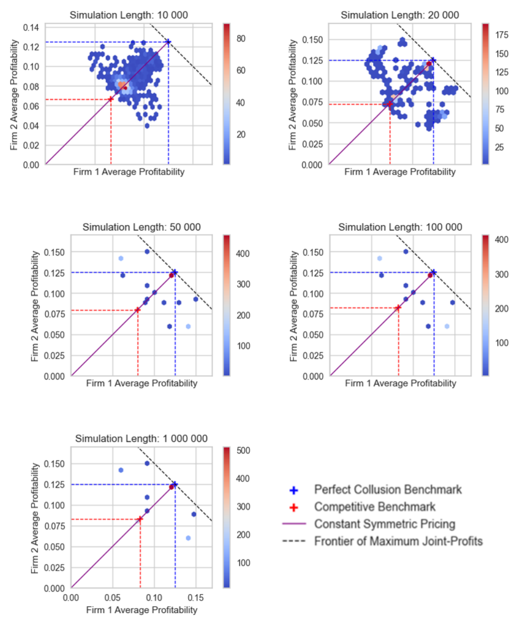
Iterations	Step Size					
	0.5		0.25		0.2	
	Var	Q1, Q3	Var	Q1, Q3	Var	Q1, Q3
10000	2.78e-06	0.1133, 0.1154	2.36e-06	0.0887, 0.0903	4.61e-05	0.0807, 0.09
20000	3.02e-07	0.1235, 0.1243	1.21e-05	0.093, 0.0935	1.49e-04	0.0928, 0.0989
50000	6.19e-10	0.125, 0.125	7.75e-06	0.0938, 0.0938	1.63e-04	0.0938, 0.1
100000	0	0.125, 0.125	5.82e-06	0.0938, 0.0938	1.57e-04	0.0938, 0.1
1000000	0	0.125, 0.125	0	0.0938, 0.0938	1.72e-04	0.0938, 0.12

Iterations	Step Size					
	0.1		0.05		0.01	
	Var	Q1, Q3	Var	Q1, Q3	Var	Q1, Q3
10000	4.56e-05	0.081, 0.09	2.65e-05	0.0847, 0.0916	2.58e-07	0.0827, 0.0832
20000	1.49e-04	0.0928, 0.0989	5.47e-05	0.0977, 0.1064	6.09e-05	0.0907, 0.101
50000	1.63e-04	0.0938, 0.1	5.09e-05	0.1009, 0.1094	8.43e-05	0.0919, 0.1041
100000	1.57e-04	0.0938, 0.1	4.71e-05	0.1013, 0.1095	7.69e-05	0.0918, 0.1044
1000000	1.72e-04	0.0938, 0.12	4.86e-05	0.1009, 0.1078	8.11e-05	0.0919, 0.1042

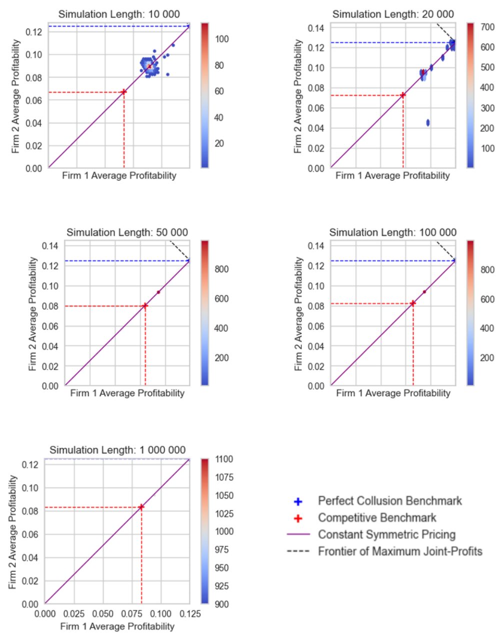
Iterations	Step Size	
	0.001	
	Var	Q1, Q3
10000	5.46e-06	0.0849, 0.0880
20000	3.10e-05	0.0866, 0.0938
50000	7.82e-05	0.0863, 0.0970
100000	7.06e-05	0.0865, 0.096
1000000	4.41e-05	0.0866, 0.0948



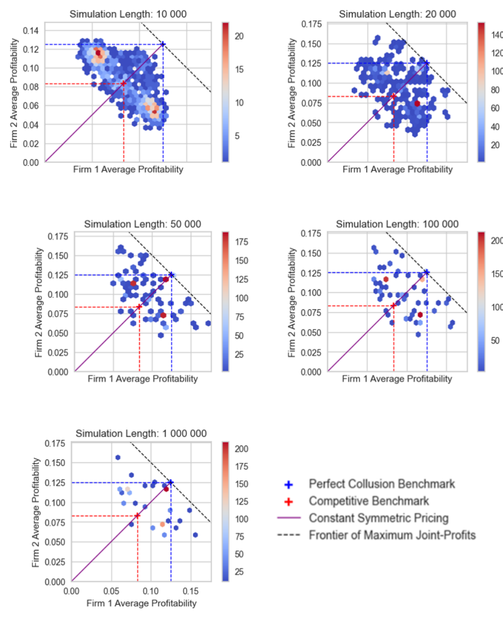
(a) Price Interval = 0.5



(c) Price Interval = 0.2

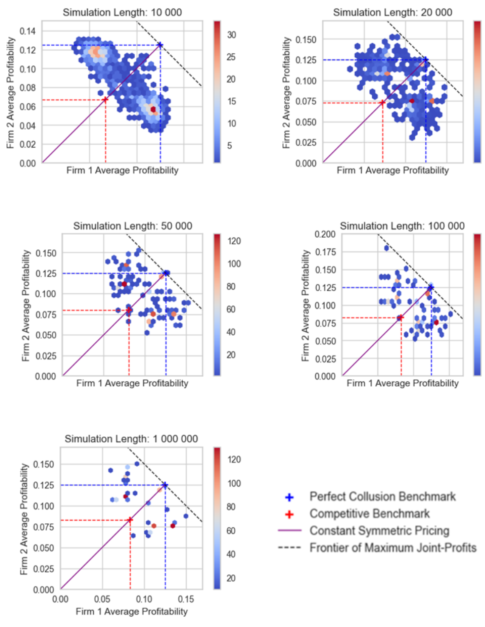


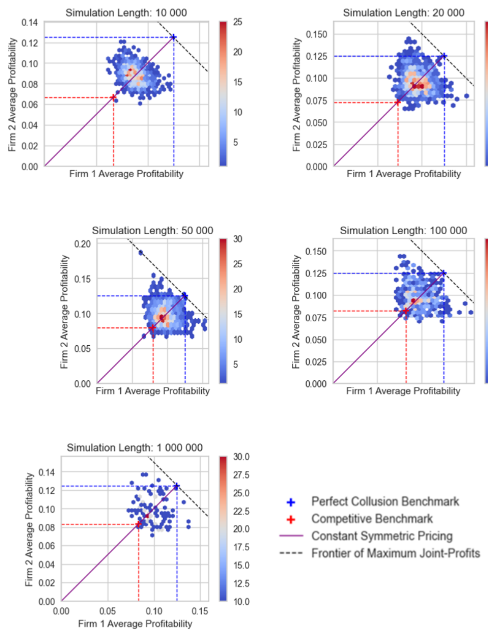
(b) Price Interval = 0.25

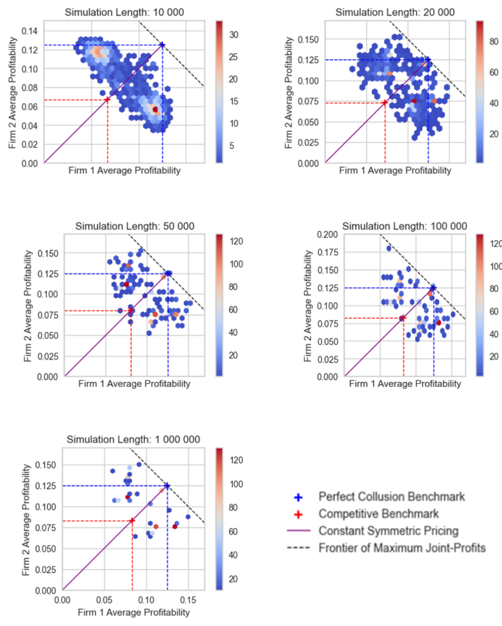


(d) Price Interval = 0.125

Figure 10: Profitability Distribution Depending on the Simulation Length for Each Price Interval


(a) Price Interval = 0.1


(c) Price Interval = 0.01


(b) Price Interval = 0.05

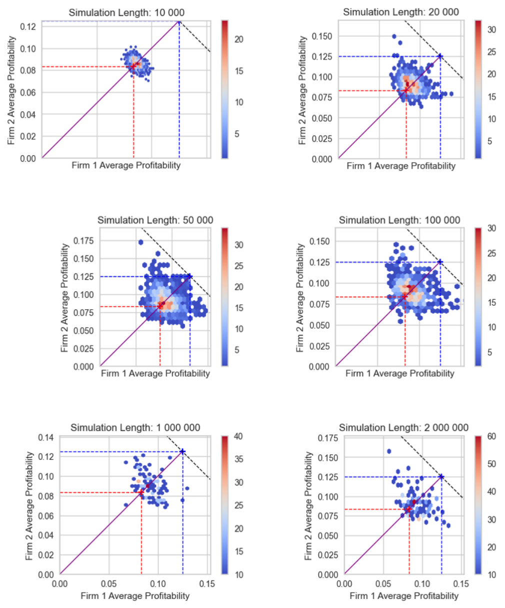

(d) Price Interval = 0.001

Figure 11: Profitability Distribution Depending on the Simulation Length for Each Price Interval