Forside

Eksamensinformation
NDAB19101E - Bachelorprojekt i datalogi-økonomi, Datalogisk Institut (Johanne og Morten)

Besvarelsen afleveres af
Johanne Emilie Wismann
fnw311@alumni.ku.dk

Morten Vestergaard Karlsen
zrp776@alumni.ku.dk

Eksamensadministratorer
DIKU Eksamen
uddannelse@diku.dk

Bedømmere
Anders Munk-Nielsen
Eksaminator
amn@econ.ku.dk
+4535324426

Niels Enemærke
Censor
ne@kfst.dk

Besvarelsesinformationer

Titel, engelsk: Price setting with artificial intelligence
Tro og love-erklæring: Ja

UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

Bachelor's thesis

Morten Vestergaard Karlsen & Johanne Emilie Wismann

Price setting with artificial intelligence

-A simulation study of collusive behaviour and asymmetric pricing technology

Supervisor: Anders Munk-Nielsen
Submitted on: 10. June 2022
Character count: 55865

Abstract

This paper examines what happens when pricing algorithms compete against each other, in particular when pricing technology is asymmetric. It determines that autonomous algorithmic collusion is possible. This is found through simulations wherein two algorithms compete in a sequential Bertrand game alternating in setting prices. These algorithms are deployed using a type of reinforcement learning called Q-learning, implemented in Python. The baseline experiment consists of two Q-learners competing, extending the study to include simulations where a Q-learner faces other types of pricing algorithms. Namely the Tit for Tat strategy, a Q-learner that is restricted by sticky pricing, and a myopic Q-learner.

Collusive behaviour is identified in the baseline experiment in more than 650 cases when averaging across 1000 repeated games. It is shown that an unrestricted Q-learner manages to exploit the Tit for Tat player. The Q-learner also reaches a collusive outcome when playing against the restricted Q-learner, but when playing against a myopic Q-learner collusion is hindered.

Table of contents

1 Introduction

Still, more firms are adopting pricing algorithms and, likely, this trend will continue. The algorithms have become more autonomous, can learn faster and make better decisions. This raises concerns regarding the algorithms ability to raise prices above a competitive level, even without explicit instruction or communication.

The amount of literature concerning algorithmic pricing is increasing as of late, as it gains the attention of more economists and computer scientists. Most scientific articles are simulation studies that examine whether algorithms set prices above the competitive level. This includes the papers Calvano et al. (2020), Klein (2021), Asker et al. (2021), and more.

One paper within empirical domain stems from an analysis of the German gasoline market, where price algorithms were adopted and led to increased profits, by Assad et al. (2020). Additionally, Brown and MacKay (2022) discover that sophisticated pricing technologies for online retailers selling over-the-counter allergy medicine lead to collusive behaviour.

This paper is motivated by the question; *what happens when we leave the role of price setting in the hands of artificial intelligence? In particular, when pricing technology is asymmetric.*

We set out to uncover this with a simulation study where we construct pricing algorithms using Q-learning and make them compete with each other within a repeated sequential Bertrand setup following the methods and parameter values in Klein (2021). Our work is meant to both confirm his findings as well as build upon it, by adding an insight into the case of asymmetric pricing technology.

Just like Klein, we find that collusion is present between two Q-learning algorithms. The algorithms learn to play collusive strategies and cooperate on prices that are well above the static Bertrand equilibrium and gain profits that are reaching the joint-profit maximizing level. The ability to set prices above the competitive level is based on reward-punishment systems, that are characteristic of collusion. Collusive outcomes are observed both in the form of price cycles and focal pricing.

These findings are concerning due to the difficulty of detecting tacit collusion, where no explicit agreement has been made. Since we see a fast-moving development within the field, it is vital that the competition authorities can keep up with these changes and make the regulations

necessary to keep competition thriving.

We further test our algorithm against different types of pricing strategies to see how it adapts and how the market and competition is affected by asymmetric pricing technology. It is clear to see that it indeed is adaptable, and exploits less sophisticated players, such as the Tit for Tat strategy and a Q-learner restricted by sticky pricing. When playing against these strategies, our unrestricted Q-learner still reaches a profit higher than that of the competitive level. However, collusion is not reached in all cases. The unrestricted Q-learner was greatly hindered when playing against a myopic Q-learner. This restriction and other possible regulations were presented in Wilk (2022) as ways to tame pricing algorithms.

Since Q-learning is known to be slow, we wanted to optimize it to run faster. This was achieved with the Python library Numba, as we experienced immense speedup by implementing this. It made the process of carrying out experiments to answer the question posed much quicker.

The rest of this paper is organized as follows. Section 2 presents the related literature on the subject that has been the foundation of our paper, and much of our theory and research behind algorithmic collusion and adoption of pricing algorithms are inspired by these. Section 3 includes the economic theory of dynamic competition, with definitions of equilibrium and the Bertrand game. We provide a review of collusive behaviour and what makes it feasible, as well as go deeper into the subject of tacit collusion. In section 4 we introduce the concept of reinforcement learning and give a thorough description of Q-learning and all its contents. This section also defines the setup used in this paper. Section 5 presents all of our results. Section 6 discusses the implementation of the algorithms as well as possible optimizations. Section 7 discusses our findings and their implications and section 8 concludes the paper.

2 Related literature

The literature on the subject of collusive strategies by reinforcement learning pricing algorithms is quite limited, as most concerns regarding this issue have been raised as of late. Most literature consists of simulation studies such as ours, as it is rather difficult to analyse occurrences of algorithmic pricing in the real world where the setup is not as regulated and controlled as you can make it in a simulation study. The research mostly differs in the way the environment

for the pricing and the algorithms are set up.

When we were first exposed to the subject of reinforcement learning pricing algorithms we were introduced to the pioneering work of Calvano et al. (2020). His findings are among, if not the, first to document the emergence of collusive strategies among autonomous pricing agents. His work is perhaps the first real sign of reason to be concerned regarding tacit collusion in algorithmic pricing. Calvano et al. and their research closely mirrors that of Klein (2021). Both achieve simulation of collusive strategies among autonomous Q-learning agents, but the clearest distinction is the fact that Calvano simulates an environment of infinitely repeated simultaneous pricing, while Klein instead simulates sequential pricing. In addition, Calvano conditions prices on past non-payoff relevant prices, while Klein does not. Once introduced to Klein (2021), the modifications made to the environment and algorithms seemed very sensible thus we decided to begin our research based on the theory and procedures presented in his paper.

The initial research on the subject also led to the paper by Asker et al. (2021). This research is based on epsilon-greedy reinforcement learning algorithms setting prices in a Bertrand game as well. The big difference is here that the authors differentiate between asynchronous and synchronous learning. Asynchronous learning allows the Artificial Intelligence Algorithms (AIA) to learn only from actions that are actually taken much like Klein (2021) and Calvano et al. (2020) while synchronous learning allows the AIA to conduct counterfactuals to assist learning. Asker et al. (2021) find that only in the case of asynchronous setting is collusion observed and suggest that asynchronous learning often is equivalent to Q-learning. One could consider implementing synchronous learning as enforcement of competitiveness.

When writing a paper involving equilibria in a discrete price grid setting, one inevitably comes by the work of Maskin and Tirole (1988). Their research is vital to the analysis of staggered pricing and the equilibria which can be found in that environment namely Edgeworth price cycles and focal pricing both of which are referenced in this paper.

In the process of writing this paper, we've come across two sources using real-world examples of algorithmic tacit collusion. Assad et al. (2020) manage to find evidence of collusion while monitoring prices set by German gas stations using certain conditions. Those involve the number of price changes made in a day, the average size of price changes, and the response time of a station's price update given a rival's price change. This is one of the first times algorithmic

collusion has been observed in the wild and has made it very clear that policymakers and competition authorities will have to deal with and take a stand regarding this issue if they haven't already. Brown and MacKay (2022) observe data from pharmaceutical companies and find that firms update their prices at regular intervals during the week and that higher frequency means a better response to rivals' prices thus yielding higher profits. Overall they find that frequency, commitment and asymmetry in pricing lead to higher prices in equilibrium.
Their research suggests algorithms fundamentally change the pricing game, as fully collusive strategies are possible even with very simple pricing models.

3 Theory

This section contains the theoretical pillars upon which we have built this paper. We cover the environment in which our algorithms operate, and the equilibria found therein. This includes a definition of the Bertrand game and a description of collusion and what makes it feasible, going into detail about tacit collusion.

3.1 Dynamic Competition

In game theory, a dynamic game is one in which players move sequentially or repeatedly. It is assumed that every participating agent knows the rules of the game, and which options are available to each player. The variation of dynamic competition we will be simulating is one with dynamic prices as the product is unchanging while the price fluctuates.

3.1.1 Nash equilibrium, SPNE, and MPE

Nash equilibrium is a term from game theory. It describes the set of actions where each player's strategy is optimal, considering the decisions of other players, therefore no player has incentive to deviate.
When playing an extensive game, we introduce history. An extensive game is one that consists of smaller games, subgames, that are repeated thus another type of equilibrium is introduced, the subgame perfect Nash equilibrium, SPNE. For a Nash equilibrium to be subgame perfect it has to be a Nash equilibrium in every subgame of the game (Osborne and Rubinstein, 2020).

In a repeated game, the player takes into account the present and the future implications of the actions and this can often lead to actions that do not seem rational in a single-stage game alone. But since we are repeating the same game many times, the seemingly irrational choice in that particular case might be part of the best possible strategy for the entire game. (Yildiz, 2012) Lastly, we have MPE which is the Markov perfect equilibrium. The MPE is a refinement of the Nash equilibrium. It is a SPNE under the Markov assumption which is that strategies only depend on directly payoff-relevant variables. (Klein, 2021)

3.1.2 Bertrand game

This section introduces the Bertrand game and its equilibria in both a static and dynamic setting.

The Bertrand game is a game with two producers of a good. They compete in prices, and the consumer will always choose to purchase goods from the firm that offers the lowest price.

We assume no marginal costs and no fixed costs. This results in a simple profit function:

$$\pi_i(p_{it}, p_{jt}) = p_{it} D_i(p_{it}, p_{jt}) \tag{1}$$

where $D_i(p_{it}, p_{jt})$ is the demand player i experiences, given player i and its competitor j's prices. Since the firm with the lowest price receives the entire market, their opponent gets nothing. If they choose the same price they share the market.

The demand function for firm i is as follows:

$$D_i(p_{it}, p_{jt}) = \begin{cases} 1 - p_{it} & \text{if } p_{it} < p_{jt} \\ 0.5(1 - p_{it}) & \text{if } p_{it} = pjt \\ 0 & \text{if } p_{it} > p_{jt} \end{cases} \tag{2}$$

The static Bertrand Nash equilibrium is given by both competing firms choosing their marginal cost as their price. To prove this, one must simply consider a firm's options once its price has been set to marginal costs. Increasing its price, profit remains 0 and if the price is reduced, profit becomes negative. At any other price level than marginal costs, competitor's will attempt to undercut and gain the entire market, thus leading to both firms setting prices equal to their marginal costs.

On the other side of the spectrum, we have the monopoly price. In the case of monopoly, the demand would be given by $1 - p$, since it is the only firm on the market. We can calculate the monopoly price for our Bertrand game by profit-maximizing.

$$\pi = p \cdot (1 - p) \quad \leftrightarrow \quad \pi = p - p^2$$

We take the derivative with respect to the price and set it equal to zero, we now have

$$1 - 2p = 0$$

Isolating p we get that the monopoly price is

$$p = 1/2$$

The original work of Bertrand is based on an environment in which prices are set simultaneously. This paper will consider sequential pricing instead, meaning two competing firms take turns adjusting the price. Sequential pricing is arguably closer to real-world data, as it is nearly impossible to set prices at exactly the same time, and most prices are a reaction to a competitor's price.

This creates a dynamic setting, thus the equilibrium found in static Bertrand no longer holds. Instead we turn to the value function defined in equation 3. A pair of strategies (consisting of response functions given by the two players of the Bertrand game) is a Nash equilibrium, if the condition given by the value function holds for all prices. Further, it is a MPE if it also includes off-equilibrium prices.(Klein, 2021)

$$V_i(p_{jt}) = max_p[\pi_i(p, p_{jt}) + \mathbb{E}_{p_{j,t+1}}[\delta\pi_i(p, p_{j,t+1}) + \delta^2 V_i(p_{j,t+1})]] \tag{3}$$

3.2 Collusion

Collusion is when two agents decide to cooperate in setting a price that is higher than the competitive price. As we have seen before, the equilibrium price in a static Bertrand game is equal to the marginal cost. In this section we will elaborate on collusion and its implications, including an example where we calculate what the discount factor has to be, to have cooperation be equilibrium.

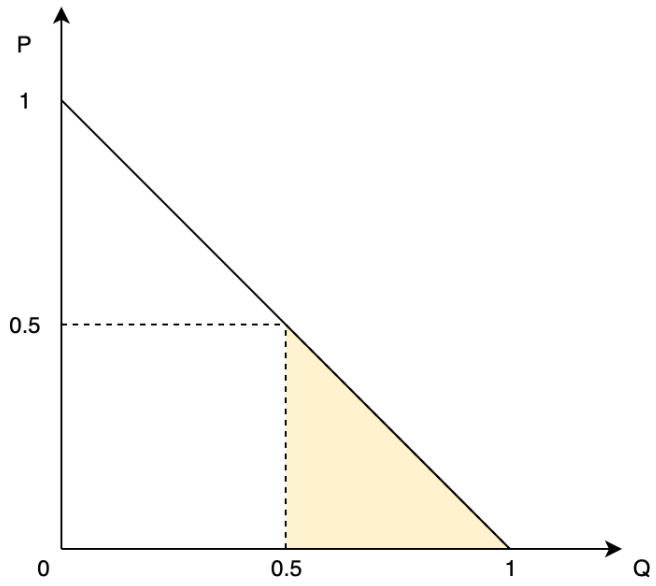

Figure 1: Welfare loss

Choosing to set the price equal to marginal cost leaves the firms with zero profit. It is only desirable for a firm to set a higher price, if all other parties do the same, as they otherwise will be left with no profit in a Bertrand game. This means that firms agreeing to cooperate will set an unnaturally high price thereby receiving higher profit than when acting on their own. This is illegal and an issue because if consumers end up being charged the monopoly price, the market will see a welfare loss. The graph depicted in Figure 1 shows the demand curve from equation 2 and the welfare loss from perfectly collusive pricing.

We can calculate the total welfare loss occurring when firms set the monopolist price, by calculating the area of the yellow triangle highlighted in the figure. The points defining the triangle are the monopolist price, the corresponding demand and the point where the demand function intersects with the x-axis. The total welfare loss is:

$$\frac{(1-0.5)\cdot 0.5}{2} = 0.125$$

One of the key characteristics of collusive behaviour is the existence of reward-punishment strategies. This means that a firm is rewarded for sticking to the collusive price, getting a higher profit. This also means a punishment will follow, should the agent choose to deviate. This punishment varies based on the strategy. It does however consist of the other firm lowering their price and thereby forcing the profit of the deviating firm down. A well-known example

of this kind of strategy is the grim-trigger strategy. This involves initially cooperating and continuing to do so, until the other player doesn't, never backing down from the subsequent punishment. The grim-trigger strategy is a very harsh form of punishment, and can in fact lead to cooperation in infinitely repeated games when both players are following this strategy - provided they care sufficiently about the future. As mentioned earlier, that is when δ is high. It is possible to calculate what discount factor enables cooperation. This example is inspired by William Spaniel's work regarding the Grim-trigger strategy but adjusted to fit our scenario (Spaniel, 2016a).

Suppose that two competing firms are in a situation where they can either choose the joint-profit maximizing price of 0.5 or deviating and choosing the price 2/6. Then the payoff from that game would be given by:

	0.5	2/6
0.5	(0.125, 0.125)	(0, 0.222)
2/6	(0.222, 0)	(0.111, 0.111)

Table 1: Payoff in the Grim-trigger example

Since the game is infinitely repeated and we are playing the grim-trigger strategy, the two firms would get a profit of 0.125 forever if they choose to cooperate forever. In case a firm chooses to undercut, they would get 0.222 in profit for that period and 0.111 for the rest of the time. We write up an inequality from this, where we are discounting the future profits to determine what δ has to be to prefer cooperation.

$$0.125 + 0.125\delta + 0.125\delta^2 + ... \geq 0.222 + 0.111\delta + 0.111\delta^2 + ...$$

We now rewrite the geometric sum and isolate delta:

$$\frac{0.125}{(1-\delta)} \geq 0.222 + \frac{0.111\delta}{(1-\delta)}$$

$$0.125 \geq 0.222(1-\delta) + 0.111\delta$$

$$0.125 \geq 0.222 - 0.222\delta + 0.111\delta$$

$$0.111\delta \geq 0.097$$

$$\delta \geq 0.87387$$

This means that for a δ larger than 0.87387 the firms would cooperate if they were both playing the grim-trigger strategy. Once again this shows that if firms care enough about the future, then supra-competitive prices can be equilibrium play. (Spaniel, 2016a)
Another strategy that uses reward-punishment is the Tit for Tat strategy, which we will go further into later.

3.2.1 Tacit collusion

This section discusses what tacit collusion is and what makes it feasible. We refer to the findings of Maskin and Tirole (1988) to emphasize the theory.
Since collusion is illegal, it is in the participating firms' best interest to keep it a secret. To combat this, most governing bodies have formed competition committees and authorities to keep the markets in check and well-functioning. This is an attempt to make sure firms are following the competition act, and in particular, to prevent collusion. (DCCA)
Tacit collusion is when firms are cooperating without explicitly agreeing to do so. This type of collusion is incredibly hard to prove, which makes it an even bigger issue. Since there is no clear agreement on the choice to collude, the competition authorities have a hard time sanctioning the involved parties. In practise, it is hard to achieve tacit collusion between people, without resorting to some form of communication. This is not always the case when prices are set by algorithms.
If firms have a high discount factor, δ, more Nash equilibria are found. A high discount factor means the agent cares about the future when making a decision, crucial to reaching tacit collusion. Both focal pricing and Edgeworth price cycles are equilibria when this condition is met. Pricing is considered focal when firms agree on one fixed price. They do this based on a belief that the competitor would undercut them if they lowered the price, and that they wouldn't follow if they raised it. (Maskin and Tirole, 1988)
Edgeworth price cycles are when firms enter a price war and continue to undercut each other until a price where they can no longer undercut is reached. The firms both have an incentive to raise the price once again but prefer the other to do so. Edgeworth price cycles are Markov perfect equilibria and the profit gained from any MPE is bound away from the Bertrand equilibrium. These are results found in Maskin and Tirole (1988), and Figure 2 illustrates the behaviour found in these cycles. They prove that if firms place enough weight on future profits,

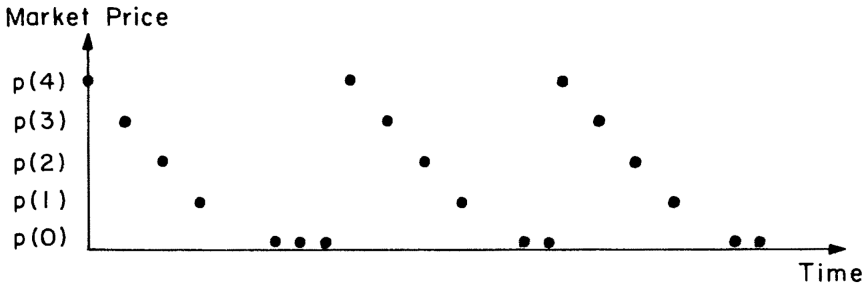

Figure 2: Edgeworth cycles (Maskin and Tirole, 1988)

we cannot expect low prices in equilibrium. This means the market experiences welfare loss without having a "smoking gun" meaning no parties have explicitly participated in collusion and left concrete evidence.

4 Reinforcement learning

In this section, we introduce the concept of reinforcement learning, particularly Q-learning and the way we have implemented our Q-learner, similarly to Klein (2021).

Reinforcement learning is a type of machine learning that learns from its mistakes. In the field of machine learning, we have supervised, unsupervised and reinforcement learning. Supervised learning is task-driven meaning it is given a correct set of actions for performing a task and is mapping between input and output. Unsupervised learning is data-driven and tries to find differences and similarities between data points. Reinforcement learning has similarities to both supervised and unsupervised machine learning, but what makes it different is that reinforcement learning algorithms learn in an interactive environment through trial and error. Furthermore, it is given a goal: to maximise the rewards throughout the learning period(Bhatt, 2018).

4.1 Introduction to Reinforcement learning

Reinforcement learning requires an introduction to a few of its key elements. We will do this in the following section where the terms specific to reinforcement learning are marked in italic. First of all, we have the *state* of the environment. This is based on previous actions by the algorithm and in our case, also the opponent. It can be described as the situation the learner is currently in. For us, that means that all previous actions and knowledge gained combined with the price chosen by the opponent, give us the current state. Secondly, we have the *rewards*

gained, in terms of feedback that the algorithm is getting from the environment. That is, a "good" decision leads to a high reward and vice versa for "bad" decisions. In our case, the reward is based on profit. This means that when making a "good" decision, you are rewarded with a high profit. Thirdly, the *policy* determines how the algorithm gets from a given state to an action. Ultimately the goal is to create an optimal policy, which leads to a maximization of rewards.

Reinforcement learning problems can also be referred to as a Markov decision process. A Markov decision process is a mathematical framework that is used for modelling decision-making and involves the same elements listed above. (Bhatt, 2018)

4.2 Q-learning

This section will introduce Q-learning, the specific type of reinforcement learning that we are going to use in our research.

Q-learning is a simple type of reinforcement learning used in unknown environments with repeated interaction. It can be divided into two modules, a *learning module* and an *action-selection module*. Learning consists of processing the information obtained from interacting with the environment, while action-selection determines how to act in the environment. An algorithm properly utilizing the Q-table table, a Q-learner, aims to maximize its intertemporal utility by choosing the largest Q-value given by its competitors price. The algorithm obtains these Q-values by adapting its behaviour to knowledge gained through previous interactions with the environment. This means a maximization of utility in the beginning of the learning period will be less optimal than one near the end. (Klein, 2021)

One of the main advantages of Q-learning is that it requires no prior knowledge regarding the problem it's facing. It is also rather straightforward to implement and the economic interpretation of the parameters is very clear. (Calvano et al., 2020)

The downside of Q-learning is that it learns quite slowly, only updating one entry in the table at a time. Q-learning algorithms will not find values for unseen states and perform counterfactual calculations as this will be considered an alternative form of reinforcement learning. This

unfortunately means that Q-learning requires quite a lot of experimenting to gain experience, which can be costly.(Klein, 2021)

4.2.1 Q-function

This section will elaborate upon the Q-function and how it works since it is the core part of Q-learning.

The Q-function is a recursive value function which is used to update the values in the Q-table. The Q-table is estimated iteratively starting from an arbitrary matrix. The Q-function is related to Condition (3), by approximating it, and ultimately converging towards it (remembering that we want to reach the optimal policy).(Klein, 2021)

When updating a Q-value, the Q-function takes into account the value that is currently at that entrance in the table, $Q_i(p_{it}, s_t)$, as well as a new estimate. The new estimate is based on the profit gained in this period from the algorithm's choice of price, given the opponent's price, $\pi(p_{it}, s_t)$. The second term in the new estimate is the profit from choosing that same price in the following period, $\pi(p_{it}, s_{t+1})$. Lastly, it includes an estimate of the optimal future Q-value. That is, the Q-value maximized by the current price given the opponent's choice in the next period, $max_p Q_i(p, s_{t+1})$. The last two terms in the new estimate are discounted with δ. This allows convergence towards a Q-table, Q^*, with accurate representations of the consequences associated with a certain price, given the current state.

The previous estimate and the new estimate of the Q-value are weighted by the step-size parameter α. This determines how much previous information is valued compared to new information.

The Q-function used for updating the Q-values is defined as:

$$Q_i(p_{it}, s_t) \leftarrow (1-\alpha) \cdot Q_i(p_{it}, s_t) + \alpha \cdot (\pi(p_{it}, s_t) + \delta\pi(p_{it}, s_{t+1}) + \delta^2 \text{max}_p Q_i(p, s_{t+1})) \quad (4)$$

4.2.2 Exploration vs. exploitation

The action-selection module consists of balancing exploration and exploitation. This is critical to optimal learning, and this section covers how we ensure a balance between the two actions. As mentioned, the Q-learning algorithm has to interact with the environment, in order to gain information and ultimately use this to optimize its policy. When doing so, the algorithm has to

balance exploring new states with maximizing the reward. We differentiate between two ways of interacting with the environment. The algorithm can either 1: perform a random action or 2: perform a greedy action.

These two options are referred to as exploration and exploitation respectively. Exploitation is seen as a greedy choice, since it involves picking the price that maximizes current profit over learning more about the environment and exploring other possible options and experiences. The choice of which to perform is determined by the parameter $\epsilon \in [0, 1]$. This ϵ-greedy method is described as:

$$p_{it} = \begin{cases} \sim U[p], & \text{with probability } \epsilon \\ \text{argmax}_p Q_i[p, s] & \text{with probability } 1 - \epsilon \end{cases} \tag{5}$$

where $U[p]$ is a uniform distribution of all possible prices.

An effective ϵ will balance exploration versus exploitation such that the algorithm initially almost exclusively explores and by the end of the learning period will be exploiting the information that it has collected in the Q-table.

4.3 Setup

To run our simulations, certain choices have been made regarding the value of parameters: α and δ as previously described but also k, the pricing interval, and ϵ, the rate of exploration. In this section we will justify parameter values.

α, the step-size parameter, often referred to as the learning rate is crucial to set at a reasonable level. This level is generally considered to be $[0; 1]$. If the step size parameter is set too high, the algorithm might find a best-response strategy quicker, but at the cost of having sub-optimally weighted strategies. Likewise, an α too small could end up requiring too many periods of learning. Unfortunately, this is not something that can be easily calculated but is best found through trial and error. (Brownlee, 2019).

We take inspiration from the parameter values presented in Klein (2021). Klein held trial runs with varying values of alpha ranging from $[0 : 1]$ and finds that 0.3 is an acceptable compromise between the need to ensure that the algorithm does not take its previous results for granted, while also making sure learning is not too slow. Calvano et al. (2020) use the baseline parameter of $\alpha = 0.15$, but his simulations greatly outnumber our with the fewest periods being 400.000.

This appropriately justifies our slightly larger learning rate through 500.000 periods. Note that 500.000 periods simply means that the agents collectively set 500.000 prices. A period can be translated to everything from multiple times a day to once every other week. This distinction is meaningful to the firms using pricing algorithms, as an algorithm taking too long to learn how to set the optimal price, could result in initial heavy losses. This is a consideration of adopting these pricing methods and sub-optimal real-world conditions could discourage firms from doing so.

δ is typically very close to 1, and through trial and error, 0.95 is found to be optimal in Klein (2021), with the same value is used in Calvano et al. (2020). Thus, $\delta = 0.95$ is used throughout this thesis.
δ lies in the interval $[0; 1]$ as well, where $\delta = 0$ results in a myopic learner, one that only learns about actions that produce an immediate reward. $\delta = 1$ will care about how the current choice affects the future and weigh the profit from the future equally in its new estimate. The balance here is to find a δ that is high enough to have the algorithm weigh past and future decisions, but not low enough to compromise performance. Intuitively, choosing $\delta = 0.95$ makes sense, since firms will generally care about the future and discount it accordingly. This could for instance, be due to alternate costs surrounding the use of capital. Alternatively, they could invest in bonds, an almost purely long-run oriented decision, prioritizing future reward.

k used to determine the discrete price grid used in our environment. Prices consists of k-spaced intervals in $[0; 1]$, meaning $k = 6$ gives the prices $[0, \frac{1}{6}, \frac{2}{6}, \ldots, 1]$. This was also the preferred value of Klein, and through trial and error, we found this to be the optimal value, as the difficulty of learning an optimal strategy increases with the number of prices on the grid.

In this paper we will determine the probability of exploration, ϵ_t, using the decay parameter, θ. This parameter will ensure that the algorithm performs actions 100% at random at first, gradually decreasing the frequency of random actions to 0.01% halfway through the run and ending the run at 0.0001%. This is done by defining:

$$\epsilon_t = (1 - \theta)^t \tag{6}$$

such that $\epsilon_{0.5T} = 0.001$ and $\epsilon_T = 0.000001$.

5 Results

This section contains the results found in our research, illustrated with graphs. We introduce the performance metrics we will use to inspect profit, profitability, as well as the degree of collusion between two Q-learners, complicity. We conduct a baseline experiment, involving two Q-learners setting prices against each other. We then proceed to pit our Q-learner against other strategies, measuring profitability as a reference to the level of competitiveness in the market.

5.1 Performance metrics

We want to measure how our algorithm is performing in our price-setting environment. There are a few interesting things to take a look at when measuring performance. First of all, we want to make sure that the algorithm converges, and actually reaches a point where it has obtained a Q-table with a final strategy, Q^*.

To ensure the Q-table has converged, we simply check if there are any changes in the best-response policies found in the Q-tables towards the final 1000 periods. If not, the table is considered converged. An average of 95.6% of runs experience convergence, therefore we are confident in performing the calculations in this section.

Another way to see if our algorithm is performing well is to see how much profit it is achieving. After all, the goal is to be able to respond optimally thereby gaining higher profit. We use the metric called profitability, which is also used in Klein (2021). Lastly, we tried to measure how optimal the choice made by our Q-learner was. But ultimately we decided to introduce a measure from Calvano et al. (2020) which determines to which degree the AI agents have begun colluding or whether a more competitive level has been reached.

5.1.1 Profitability

Profitability measures the profit that an algorithm has on average obtained throughout the previous 1000 periods.

We store the profit obtained by each algorithm throughout the entirety of the run and calculate

a rolling average.

$$\Pi_i = \frac{1}{1000} \sum_{t=T-1000}^{T} \pi_i \left(p_{it}, p_{jt}\right) \tag{7}$$

This means that we monitor profit in all periods except the first 1000.

In all graphs depicting profitability we illustrate two benchmarks, joint-profit maximizing and competitive as inKlein (2021). The joint-profit maximizing benchmark of 0.125, happens when both firms set the monopolist price $p = 0.5$. The competitive benchmark we use is 0.0611. As we have sequential pricing, this is not the profit of the Nash-equilibrium of setting the price to marginal cost as in the static game, but rather the most competitive MPE Edgeworth price cycle described in Maskin and Tirole (1988). This is the average profit of both firms where the price cycle consists of undercutting the other until the lower bound is reached followed by the firm first observing this price, resetting to one increment above monopoly price ($p = \frac{4}{6}$).

5.1.2 Complicity

This section introduces the performance metric complicity and our thoughts behind including this measure.

Initially, our goal was to replicate the work of Klein (2021), and this included his performance metric, optimality. Described as

$$\Gamma_i(p_{it}, p_{jt}) = \frac{Q_i(p_{it}, p_{jt})}{max_p Q_i^*(p, p_{jt})} \tag{8}$$

it is meant to measure the optimality of a choice made by a Q-learner, compared to a Q-table that was allowed to converge to an optimal policy given the current competitor strategy. This should theoretically produce $\Gamma_i = 1$ as an algorithm learns best-response behaviour. Klein defines Nash equilibrium as both Q-learners having $\Gamma_i = 1$. This metric was developed by the author purely for expositional purposes and has been described as "somewhat ad hoc". We found that in practise, it was difficult to replicate the measure due to a lack of detail in the paper. Furthermore, we believe it to be too computationally heavy and overall it seemed redundant when we could get a similar point across with simpler calculations.

Another performance metric was needed. We chose to focus on uncovering whether or not collusive behaviour is present. To measure the level of collusion the two algorithms reach at the end of a run, we instead take inspiration from Calvano et al. (2020). In this paper the

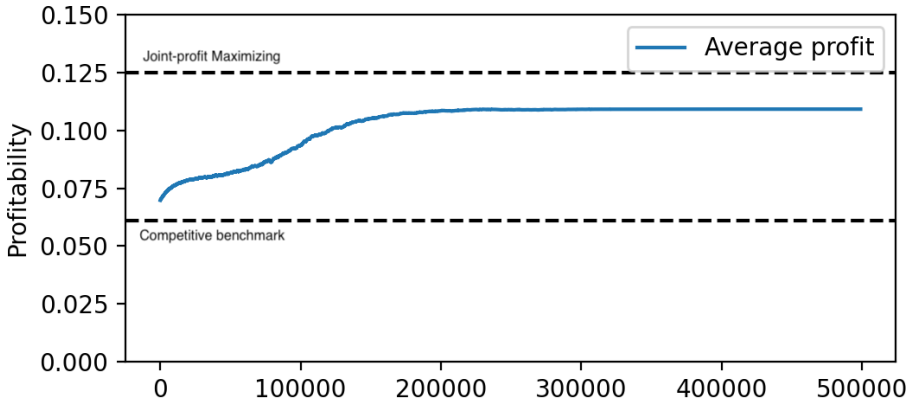

Figure 3: Average profitability: two Q-learners

metric complicity, denoted by Δ is used:

$$\Delta \equiv \frac{\bar{\pi} - \pi^N}{\pi^M - \pi^N} \tag{9}$$

where $\bar{\pi}$ is the average per-firm profit at the end of a run, π^N is the profit in the Bertrand-Nash static equilibrium, and π^M is the profit under full collusion (monopoly). As we operate with no marginal costs, $\pi^N = 0$ this modifies the measure, thus

$$\Delta = \frac{\bar{\pi}}{\pi^M}$$

This means that we simply measure to which degree the players have reached the Joint-profit maximizing profit. Therefore $\pi = 0$ means $\Delta = 0$ and $\Delta = 1$ is the perfectly collusive outcome and a profit of 0.125.

5.2 Baseline results

This section will illustrate the results found when letting two Q-learners set prices in an environment with parameters described in Section 4.3. We present graphs depicting the profitability, the distribution of the two players profitability combined and the distribution of the complicity measure. To observe the behaviour of Q-learners interacting, we conducted a baseline experiment by pitting two Q-learners against each other in the sequential Bertrand game. Figure 3 illustrates the average profitability of 1000 independent learners across 500.000 periods. As you can tell, the profit gained is well above the competitive benchmark, almost reaching the joint-profit maximizing point, as the simulation progresses. We observed an average profitability of

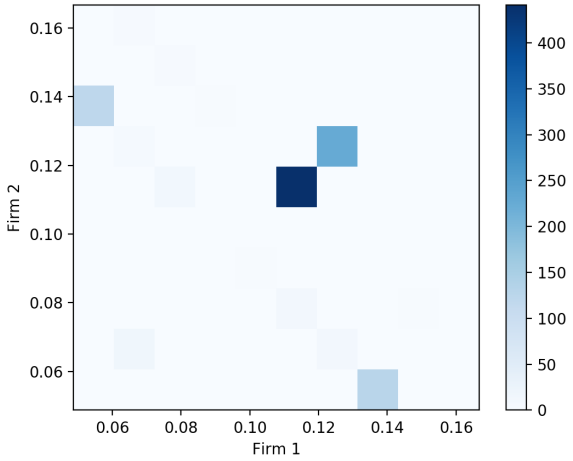

Figure 4: Distribution of profitability Π_i for two firms

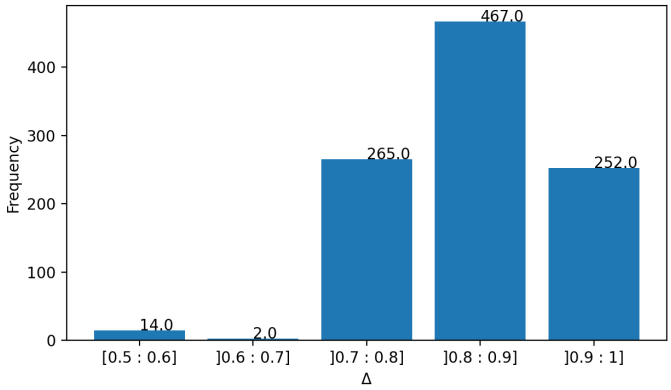

Figure 5: Distribution of Δ

0.1092. This tells us that the reinforcement learning algorithms on average are performing on a supra-competitive level.

To further investigate just how often the algorithms actually reach a collusive equilibrium and how the level of profitability is distributed, Figures 4 and 5 were generated. We have plotted the distribution of the profitability Π_i for the two firms as a heatmap, inspired by Klein (2021), as well as a bar chart of the average Δ values. The colours on the grid of the heatmap, are determined by how many times a specific combination of profitability values is observed. Notice that the most prevalent case is when both firms reach profitability of just below 0.12 roughly corresponding to $\Delta = [0.8 : 0.9]$. 250 cases of joint-profit maximizing profit of 0.125 per firm are also observed. From this, we can tell that in more than 650 runs, the algorithms did indeed learn to cooperate at a collusive level.

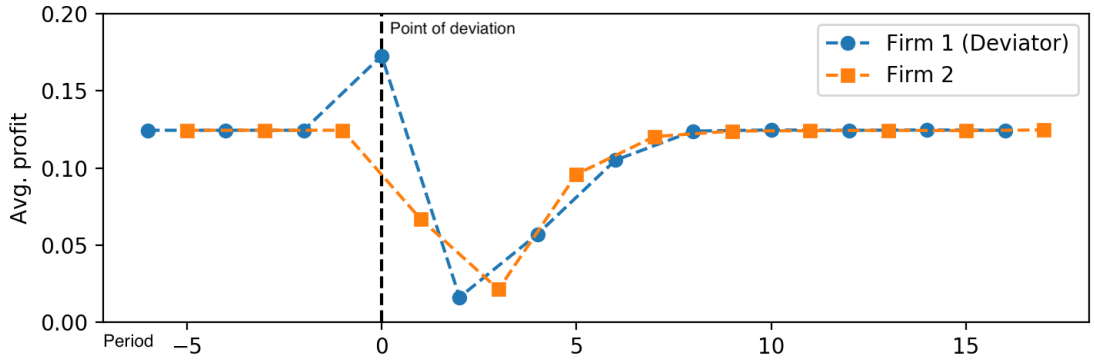

Figure 6: Average two-period profit at point of forced deviation

5.3 Forced deviation

In the presence of collusion, it is interesting to see whether the two firms have learned to return to their old strategy if one of them decided to deviate. To illustrate what happens in this situation, we are forcing a deviation from one of the firms in a run where focal pricing at the joint-profit maximizing price $p = 0.5$ is observed. The forced deviation is forced upon the algorithm in a period towards the end of the simulation, where exploration is unlikely to happen. We calculate a two-period average such that one point on the graph shows the average profit from the current period and the previous inspired by Klein (2021).

From Figure 6, it is clear to see that the punishment is temporary and the two firms return to their initial joint-profit maximizing strategy. After approximately 10 periods of punishment, they reunite at $p = 0.5$ and they once again obtain the profit of 0.125 until the end of the simulation.

5.4 Other players

We decided to pit the unrestricted Q-learner against a variety of conventional strategies to give us an idea of how adaptable the q-learner is to other types of players and to see what happens when there is asymmetry in pricing technologies. Theoretically, the Q-learner should be able to beat less complex and knowledgeable players, given enough time.

We were inspired to conduct these experiments based on the paper by Assad et al. (2020) about gasoline in Germany, where it was clear to see that more sophisticated pricing algorithms had an advantage in the market.

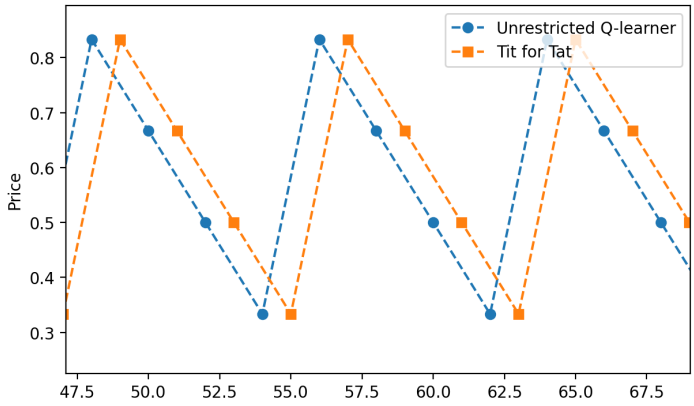

Figure 7: Prices set by Q-learner and Tit for Tat

5.4.1 Tit for Tat

The strategy called Tit for Tat is another strategy that also involves rewarding cooperation and punishing deviation. This section will cover the profitability of a Q-learner competing against Tit for Tat as well as an example of the exploiting that occurred.

In terms of punishment, Tit for Tat is less harsh than the grim trigger strategy which was discussed in Section 3.2. This strategy works very well in practise, even though it is not a SPNE[1]. Robert Axelrod held a tournament where he invited game theory experts and people of many different professions to submit algorithms to compete in a computer prisoners dilemma game. The winner ended up being the Tit for Tat strategy (Spaniel, 2016b).

With the Tit for Tat strategy, the player simply begins the game with cooperation and then copy its opponent's last move from there on. Herein lies the threat that should your opponent choose not to cooperate, then the player using Tit for Tat, will simply also stop cooperating as long as you do. This leaves room for forgiveness as it is possible to start cooperating again, but it must be initiated by the player facing a Tit for Tat player.

A game was set up where an unrestricted Q-learner, as described until now, faced a Tit for Tat player. We found that the Q-learner was able to dominate the market, in every single run, since it saw through the Tit for Tat player's strategy. This gave the Q-learner the possibility to exploit the Tit for Tat player by initiating a price cycle at the highest profitable price, $p_i = \frac{5}{6}$, and then undercutting its opponent by 1 price increment until reaching $p_i = 2/6$ followed by

[1]There can be found one delta where the strategy actually is SPNE, but generally, it is not.

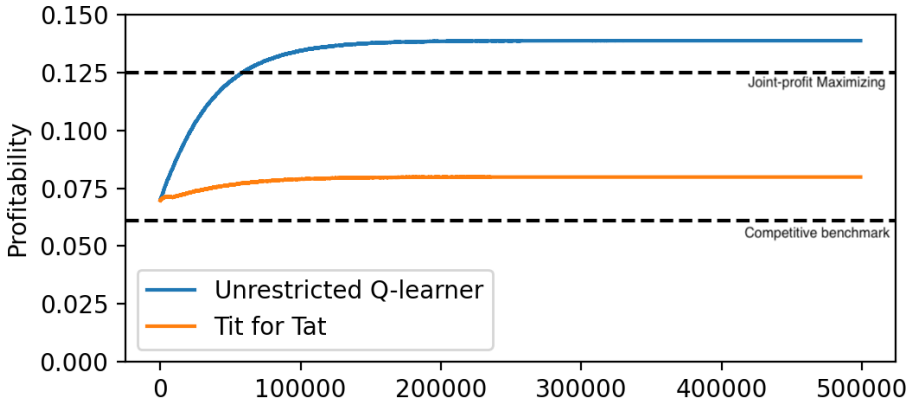

Figure 8: Profitability of unrestricted Q-learner and Tit for Tat player

resetting. The agent used its opponent's pattern of copying prices to design a cycle where it allows the other player to receive the full profit of $p_j = \frac{2}{6}, \pi_j = 0.2\bar{2}$ in one period, followed by either $\pi_j = 0$ or splitting the shared profit through the rest of the cycle.

One might wonder why the algorithm does not set the price of $p_i = \frac{1}{6}$, gaining $\pi_i = 0.1388888889$ followed by splitting the profit and then resetting. This turns out to result in an average profit of 0.1319 per cycle while leaving out $p_i = \frac{1}{6}$ yields a profit of 0.1388.

This pattern of price cycles can be observed in Figure 7.

This, of course, leads to a very desirable profit for the unrestricted Q-learner. The average profit gained by the Q-learner actually lies well above the profit realized when agents collude at monopoly price. This is shown in Figure 8.

5.4.2 Restricted Q-learner

The next thing we wanted to do was to impose some restrictions that could mimic ways you could be restricted in real life. We tried a few different restrictions on our unrestricted Q-learner but ultimately wanted to highlight the Q-learner restricted by sticky pricing. Sticky pricing, in this setting, is when you are prohibited from updating your prices as often as your opponent. That is the case in many different markets. One example could be hairdressers.They don't change the price of their haircut very often compared to for example gasoline, where prices are adjusted very often.

Even in the same market, frequency of updating prices can vary a lot. (Brown and MacKay, 2022)

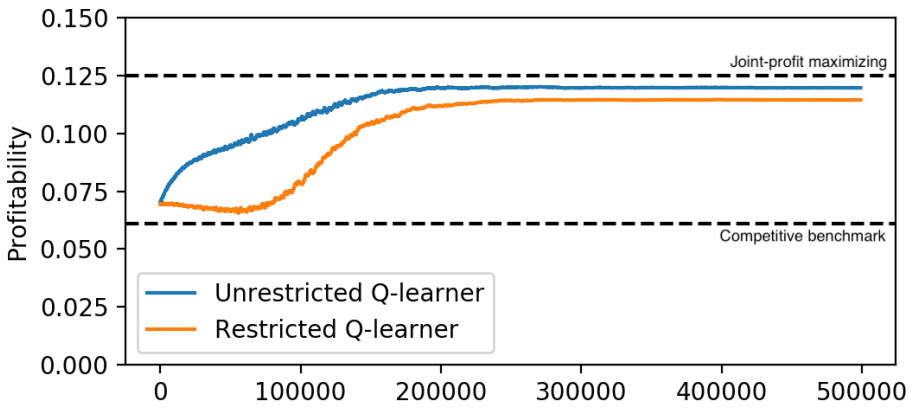

Figure 9: Profitability of restricted and unrestricted Q-learner

The Q-learner restricted with sticky prices is similar to the baseline Q-learner, but it is only allowed to update its price 50% of the time. This means that the firm generally is stuck with the price it set for a longer time than its opponent. Figure 9 shows how the profitability gained by both the restricted and unrestricted Q-learner is very high, very near that of joint-profit maximizing. It is worth noting that the unrestricted Q-learner on average has the upper hand, and generally obtains higher profitability. The advantage is more prominent at the beginning of the run and remains on top albeit less as exploitation becomes the more common choice.

Figure 10 illustrates how the restricted Q-learner is restrained by the sticky pricing. Here you can see when the restricted Q-learner is forced to keep the price set in the previous period regardless of the algorithm wanting to lower it (marked in green). One example, is period 302443, 302445, and 302447 where we can see the restricted Q-learner is forced to keep the price of 1, and immediately changes the price to undercut the competitor as soon as it is "allowed" to do so. In these periods the unrestricted Q-learner gets the whole market, instead of being undercut, and this results in an overall higher profit for the unrestricted Q-learner. These kinds of cycles mostly occur in the first half of the runs, and in a few cases, they continue all throughout the simulation, which could explains the gap between the two agents' profitability at the end. The unrestricted Q-learner also has the advantage of being able to explore more in the beginning thereby arriving at an optimal strategy faster. In the vast majority of the runs, they end in focal pricing, which is what ultimately leads to the very high profitability for both types of learners. In Brown and MacKay (2022) they highlight that firms with faster pricing technology generally have a lower price, thus gaining the market profit. This is in

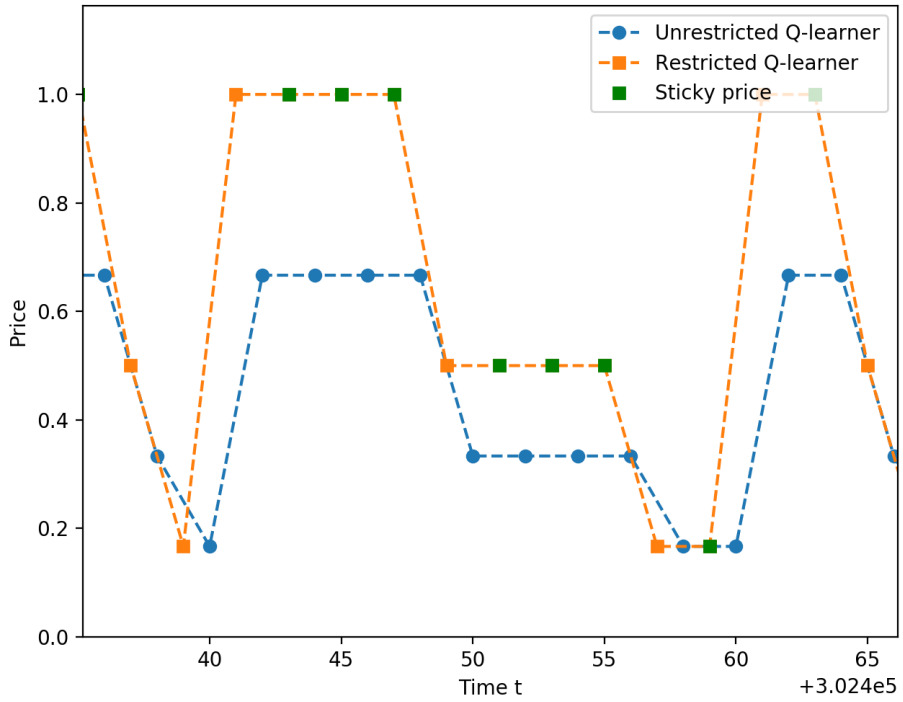

Figure 10: Example of prices played by restricted and unrestricted Q-learner.

line with what we observe when we restrict our Q-learner to only update the price in half of greedy periods. One thing to note here is that Brown and MacKay (2022) look at retailers who update prices at regular intervals which differs among the firms. However, it seems reasonable to argue that the same holds for our restriction since the general idea is the same; one firm has a frequency advantage in terms of setting the price compared to the other. This suggests that more advanced technology leads to a less competitive market. Also, they found that reacting to the competitor's price by committing to a strategy creates a credible threat which leads to collusive behaviour, just as we have seen. This stems from the very strong initial commitment when using algorithms. Having your strategy written down in an algorithm automatically makes it a clear commitment. This commitment is unlike human behaviour, as empathetic behaviour towards your opponent could happen, or more realistically worries regarding the agent's own profits. These factors could result in an altering of strategy, while an algorithm would remain unfazed.

5.4.3 Myopic learner

In the following section we try to restrain the Q-learner by letting it play against a myopic learner. We present a graph with the profitability gained by both learners, where its clear to see that the level of collusion has decreased.

A myopic learner is short-sighted. A pricing agent who doesn't look into the future, and only thinks about the current situation, and how to maximize the reward gained now. The idea of including a myopic learner came from (Wilk, 2022). The article discusses how pricing algorithms can be discriminatory against minority groups and the general ethics of programming a pricing algorithm. The issue of algorithms learning to collude is also raised. Wanting to avoid both of these things they seek a way of taming the pricing algorithms. Here they highlight some of the findings of Xu et al. (2022). For instance, that imposing price controls, consisting of an upper and lower limit, can lessen the collusive behaviour but more interestingly they also point out that agents with present bias prevent collusion. We thought this would be interesting to see in action, and we set our Q-learner up against a myopic Q-learner in the Bertrand game.

Figure 11 shows that playing against the myopic learner drags the profitability of the Q-learner down to just above the competitive benchmark defined by the competitive Edgeworth cycle, with an average profitability of 0.0694. This corresponds to a 63.67% decrease of profits relative to two Q-learners facing each other.

This profit happens when the two agents are both choosing the price $p = \frac{1}{6}$, one increment above marginal costs. At this point, the myopic learner has no incentive to undercut and reach $p = 0$, nor to raise the price. This is because the myopic learner is not looking towards the future thereby not seeing the possible profits from raising the price now. This pattern is learned by the unrestricted Q-learner and the agents end up settling for this relatively low pricing level, thus reaching a more competitive level of pricing.

Interestingly the myopic learner is initially able to gain a sizable profit over the Q-learner. This is most likely caused by the unrestricted Q-learner still exploring, making sub-optimal choices to learn while being undercut by the myopic learner. Even when making exploitative decisions it is considering future profit, not yet realising that its opponent does not, ending up losing the market.

This means that we observe results similar to the article, and can confirm that even just facing

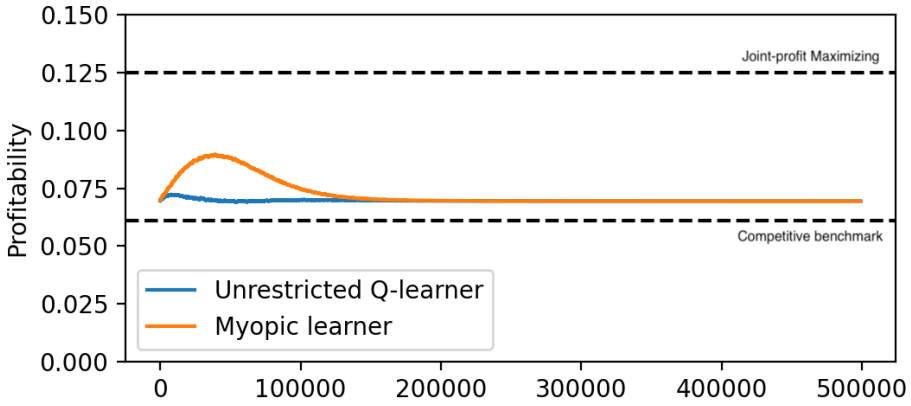

Figure 11: Profitability of myopic Q-learner and unrestricted Q-learner

a myopic player will eliminate collusive behaviour. This is also consistent with our knowledge about the discount factor. Remember that δ has to be high for the collusive strategies to be an equilibrium.

6 Implementation and optimization

Strategies using reinforcement learning and other varying complex calculations must be simulated to fully monitor its progress. This section will discuss our implementation of these simulations and the optimization of computational speed. Pseudo code for the algorithms referred to in this section can be found in Appendix A.

6.1 Implementation

We conducted our simulations using an executable script in python. The progress of this implementation very closely follows the method described in Klein (2021). This resulted in a program modelled as seen in Algorithm 1 and 2.

Once algorithm 2 is called, it returns 2 pairs of matrices of size order [runs : periods] and $\frac{[\text{runs:periods}]}{2}$ corresponding to the achieved profit as well as prices set for each agent in each period. These matrices were manipulated to produce the graphs seen in this paper.

6.2 Calculations

This section includes an estimate of the possible optimization, and will provide a perspective of the calculations necessary to perform our computational workload.

Taking a look at algorithm 1, the main body of calculations, we can see that for each run we are picking 4 random prices to initialize the algorithm and then initiating a loop which runs for T-4 periods. This loop is then repeated as many times as the number of runs we wish to perform, illustrated in algorithm 2. When pitting two Q-learners against each other, each iteration of the loop in algorithm 1 requires at least 16 floating-point operations performed. Floating-point operations are defined as either: addition, subtraction, multiplication or division using decimal numbers. One example would be calculating the profit when the algorithm has just set a price. That price has to be multiplied by the demand, which is determined by either 0, 1 or 2 operations, depending on whether the price set is equal, lower or higher than its opponent's price. Given the minimum 16 calculations averaging across 1000 runs for 500,000 periods, $1000 \cdot 500000 \cdot 16 = 8,000,000,000$ floating-point operations are required and perhaps more given the random elements of the algorithm.

6.2.1 Numba

This section gives an introduction to Numba, and how it optimized our workload.

Initially, we ran our simulations through regular python scripts using an Apple MacBook from early 2015 but quickly found this task to be overwhelmingly complex to perform in a manageable time.

This quickly made us consider the python library, Numba. Much like the extremely popular python library, NumPy, Numba is written in the programming language C and wrapped in Python for ease of use. Numba generates optimized machine code from Python code. This enables just-in-time compilation meaning compilation is performed during execution as opposed to before. If Python code either executes many loops, utilizes arrays or has a considerable amount of math, optimization should be possible.

As we are using just-in-time compilation, Numba will counter-intuitively output rather slow performance during the first calculation as it is compiling. This is however then replaced by extremely fast execution of code if implemented correctly.

When attempting our calculations without Numba in online notebooks to spare the GPU of our personal computers, the RAM allocated would be used up and the execution of our program would be stalled. Forced to run the scripts on personal computers, they had to be left on overnight or be left unusable for the better part of an entire day.

After modifying and replacing certain data-types within the script and adding the Numba just-in-time decorator, Numba was implemented and the 1000 runs of 500,000 periods can now be performed in 3 minutes on a personal computer. We measured an average of $20x$ speedup when performing 10 repeated runs, keeping in mind that this speedup will exponentially grow as repetitions increase.

Even more optimization of our scripts is possible, as we executed calculations on multidimensional arrays using NumPy methods that have not yet been made optimizable by the Numba developers. Once these are implemented, performance will be extremely fast compared to its complexity.

6.2.2 Parallelizing

When simulating 1000 independent runs, anyone familiar with parallel programming would instantly look into the parallelization of such a computation. This would mean utilizing multiple threads of a GPU to perform calculations independently of each other. However, this is a feat notoriously known to be difficult in Python and for the most part utilized in high-level languages such as C/C++. While Numba has an option for parallelizing, the data types necessary for the computations in this paper did not support this option. This would have been a natural, perhaps necessary, next step should heavier computations become necessary.

7 Discussion

Our research confirms the findings of Klein (2021), that algorithmic collusion occurs between Q-learning algorithms. This could however be considered a somewhat isolated issue, as the conditions under which this was possible have been decided by the authors. Since our study is based purely on simulations within an undisturbed environment, translating it directly to reality might lead to questions like; how will the algorithm perform transitioned from a training environment to actual competition? How does it act when facing multiple opponents each with

a different pricing approach? Which unforeseeable factors affect a real-world pricing setting? One might suspect that collusion similar to the one we witnessed in the training environment is unsustainable in a real-world setting. This claim is only strengthened by the point in Calvano et al. (2020) that the degree of collusion decreases when the number of competitor's increase. It is highly unlikely, if not impossible, that a perfect duopoly can exist in the, mind you, very complex real markets.

While this is true, one must consider the fact that both Assad et al. (2020) and Brown and MacKay (2022) found signs of algorithmic pricing and increased profit as algorithmic involvement became clear. Even though our findings of supra-competitive pricing by Q-learners were observed in a specially designed sandbox, they are still valuable. Less optimal conditions could still cause worry regarding collusion, albeit reduced.

Further, we've shown that a certain asymmetry in pricing technology can be crucial to your profits in a frequently updating market. Having humans or less sophisticated algorithms set prices against a Q-learner could result in a similar pattern to the one observed in section 5.4. These disadvantages cause major differences in profit, and witnessing this as management provides an incentive to invest in similarly advanced pricing software, equalizing the aforementioned asymmetry. This could be valid critique towards the relevancy of our work, but we argue that since technology is always improving, markets will never reach a point of perfect symmetry. We could perhaps come closer to perfect symmetry in pricing technologies, with the help from regulations. However, the technological development persists leaving research behind. This fact causes us to believe that asymmetric pricing technology is an issue that will remain.

Future research would include testing the reinforcement learning algorithms to an even greater extent in preparation of facing real-world opponents. Additionally, finding past data in an appropriate market, perhaps where pricing data is similar to Assad et al. (2020) or Rise (2021), would be optimal to test effectiveness compared to empirically utilized algorithmic pricing software.

Another interesting addition to the research in this paper would be to explore alternative regulations in order to further hinder collusion. Much research within this field is still experimental

and contributing to finding the most effective solution would no doubt be very useful to further existing research and assist authorities.

8 Conclusion

In this paper, we studied the behaviour of reinforcement learning pricing algorithms in a sequential Bertrand setting. We confirmed the work of Klein (2021), that autonomous algorithmic collusion is possible in a simulated environment.

Furthermore, we demonstrated that the agents participating in collusion with focal pricing will be punished by their opponent when deviating. The collusive nature of the agents meant a fast and steady return to the previous state. We observed that a Q-learner was able to exploit traditionally successful game strategies as well as realistically restricted Q-learning algorithms. We discuss the opportunity of preventing collusive behaviour in algorithms by implementing present bias, effectively creating a myopic learner.

This suggests that asymmetry in pricing technologies cause vastly different profitabilities and could be attributed with non-competitive behaviour.

Our research suggests that antitrust laws must soon be constructed or altered to reflect the reality of algorithmic pricing, as methods similar to ours are surely used, if not widely dispersed in plenty of industries already.

References

John Asker, Chaim Fershtman, and Ariel Pakes. Artificial intelligence and pricing: The impact of algorithm design. 2021. URL https://www.nber.org/papers/w28535#:~:text=The%20behavior%20of%20artificial%20intelligences,when%20there%20is%20market%20interaction.

Stephanie Assad, Robert Clark, Daniel Ershov, and Lei Xu. Algorithmic pricing and competition: Empirical evidence from the german retail gasoline market. 2020. URL https://www.cesifo.org/en/publikationen/2020/working-paper/algorithmic-pricing-and-competition-empirical-evidence-german.

Shweta Bhatt. Reinforcement learning 101, 2018. URL https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292. [accessed:19.05.2022].

Zach Y. Brown and Alexander MacKay. Competition in pricing algorithms. 2022.

Jason Brownlee, 2019. URL https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/#:~:text=The%20amount%20that%20the%20weights,range%20between%200.0%20and%201.0.

Emilio Calvano, Giacomo Calzolari, Vincenzo Denicolò, and Sergio Pastorello. Artificial intelligence, algorithmic pricing, and collusion. 2020. URL https://www.aeaweb.org/articles?id=10.1257/aer.20190623.

DCCA. Danish competition authority: Cartels. URL https://www.en.kfst.dk/competition/cartels/. [accessed: 17.05.2022].

Timo Klein. Autonomous algorithmic collusion: Q-learning under sequential pricing. 2021. URL https://onlinelibrary.wiley.com/doi/10.1111/1756-2171.12383.

Eric Maskin and Jean Tirole. A theory of dynamic oligopoly, ii: Price competition, kinked demand curves, and edgeworth cycles. 1988.

Martin J. Osborne and Ariel Rubinstein. *Models in Microeconomic Theory*. OpenBook Publishers, 2020.

Christina Rise. When pricing algorithms take over - an empirical application to the danish pharmaceutical market. 2021.

William Spaniel. Grim trigger in repeated prisoners dilemma, 2016a. URL https://www.youtube.com/watch?v=8jBouAQt9EY&t=1078s.

William Spaniel. Tit-for-tat in the repeated prisoner's dilemma, 2016b. URL https://www.youtube.com/watch?v=I_Ug4vHHtGo.

Ethan Wilk, 2022. URL https://www.scientificamerican.com/article/an-old-fashioned-economic-tool-can-tame-pricing-algorithms/. [accessed:26.05.2022].

Renzhe Xu, Xingxuan Zhang, Peng Cui, Bo Li, Zheyan Shen, and Jiazheng Xu. Regulatory instruments for fair personalized pricing. In *Proceedings of the ACM Web Conference 2022*. ACM, apr 2022. doi: 10.1145/3485447.3512046. URL https://doi.org/10.1145%2F3485447.3512046.

Muhamet Yildiz. Repeated games, lecture notes, 2012.

Appendix A

Algorithm 1 Simulating a single run

```
Set learning parameters
Initialize Q-tables
Randomly pick {p_it, p_jt} for t ∈ {1, 2}
function GAME(params, T)
    for t = 3 → T do
        if t % 2 = 1 then
            update Q_i according to equation (4)
            pick a new price p_i according to strategy
            save profit, prices
        else
            update Q_j according to equation (4)
            pick a new price p_j according to strategy
            save profit, prices
        end if
    end for
    return prices, profit
```

Algorithm 2 Simulating multiple runs

```
function MULTIPLE_GAMES(game, reps)
    for i = 0 → reps − 1 do
        simulate game[i]
        save prices, profit
    end for
    return prices, profit
```

*Entire code available at github.com/mortenvk/Bachelorprojekt