UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

Bachelor's Thesis

Conrad Lavesen Valentinus & Johan Søvndahl Kok

Collusion 2.0

-Artificial Intelligence as a Collusive Force in Modern Price Competition

Supervisor: Anders Munk-Nielsen

Submitted on: DATE

Abstract

Contents

1 Introduction

We live in the age of artificial intelligence (AI), which is rapidly transforming our world, while being in rapid development itself. Increasingly, AI technologies replace traditional processes, including how companies price products and thereby compete against competitors. While using AI algorithms for automating repetitive tasks is generally beneficial, using these algorithms for pricing risks consumers being the laughing stock by paying market prices, above normal competitive levels.

Setting prices above the competitive level can be considered collusive behavior. For this reason, algorithmic price setting has become an increasingly important topic in computer science and economics, and is the focus of this project. Simulation studies investigating algorithmic pricing and its implications include Calvano et al. (2020); Klein (2021), which find algorithmic collusion to be a possibility. Empirical support comes from studies by Assad et al. (2020) and Noel (2007), who both investigated the effects of algorithmic pricing in gasoline markets and confirmed that it leads to supra-competitive prices. These findings show that algorithmic collusion is observed in real markets and thereby not only theoretical.

This project follows the framework of Klein (2021), who uses the work of Maskin and Tirole (1988) as the economic base for the model. As Klein (2021), we use machine learning to simulate turn-based price competition between two firms. Specifically, we apply Q-learning in a sequential duopoly setting depicting a Bertrand economy. As a simulation-based proof of concept, our aim is to answer the overall question: *Does collusive behavior occur, and to what extent, for different price-setting intervals when reinforcement learning algorithms compete against each other? Additionally, why do these intervals affect the performance of the Q-learners?*

Since the Q-learners do not communicate or explicitly agree, any observed collusion is tacit. The pricing intervals are determined by the parameter k, essentially determining the number of pricing options in the environment. To assess collusive behavior, we use the most competitive Edgeworth cycle as the competitive benchmark, as Klein (2021), which varies with k. The other benchmark is the joint profit-maximizing level, derived from the monopolist profit. These benchmarks form the basis for a normalized measure $\zeta_{k,\bar{\pi}}$, introduced in Section 4, which is inspired by Calvano et al. (2020). Additionally, we analyze cycles from the outcomes to investigate how the cycles, and thereby result, change as the k values increase, and to investigate reward-punishment behavior, which is an indicator of collusion.

To answer our overall question, this project begins by introducing relevant economic theory to set up the environment in section 2. Section 3 covers the theory necessary to understand the foundations and considerations behind the implementation of the Q-learning algorithm. In section 4, we explain the simulation setup, performance evaluation, and parameter settings. After that, we present and analyze the results in section 5. Finally, in section 6, we discuss the results and their robustness, followed by the conclusion in section 7. Our conclusion underlines that collusive behavior occurs tacitly to some degree between firms sequentially setting prices using Q-learning. We observe reward-punishment behavior in analyzed cycle examples, with the overall degree of collusion tending to decline as the parameter k increases.

2 Economic Theory

In the following section we will focus on unraveling the Economic theory behind the dynamics of the game we will be simulating. The overall idea is that we are using machine learning to simulate price competition between two firms. This is being done using a reinforcement learning algorithm known as Q-learning in a duopoly setting known from a Bertrand economy. This framework is based on an article from Klein (2021), who uses the work of Maskin and Tirole (1988) as the economic base for the model. Therefore it is a dynamic, sequential setting, where the firms will take turns of setting their price. Like Klein (2021), we will investigate whether using algorithmic pricing can lead to, what can be seen as collusive behaviour.

We are investigating the dynamics of the game and specifically what happens when you change the number of prices that each firm can set.

2.1 Bertrand competition

The competitive framework adopted in this project builds on the classical Bertrand model, in which symmetric firms competing in a market for a homogeneous good set prices, while quantities are determined by the market demand (Varian, 2019, p. 532). While the Bertrand model can be generalized to oligopolies, we focus on a duopoly setting, reflecting the model's original formulation (de Bornier (1992)) and aligning with the approaches of Maskin and Tirole (1988) and Klein (2021), who examine dynamic pricing competition in a two-firm market setting.

Under the classical Bertrand assumptions of homogeneous goods and constant marginal costs, firms are predicted to price at marginal cost. However, in real markets with few sellers, such outcomes are rarely observed. Maskin and Tirole (1988) attribute this discrepancy to the static nature of the standard Bertrand model and propose a dynamic alternative in which firms set prices sequentially over time. Following the work of Klein (2021), we adopt a dynamic and sequential duopoly model. Unlike simultaneous, move frameworks such as that of Calvano et al. (2020), Klein (2021) argues that the sequential structure better captures real world pricing behavior. In practice, it is unlikely that firms update prices at precisely the same moment, instead it is more realistic that one firm moves first, with the other observing and responding accordingly.

2.2 Game Theory

The simulation framework applied in this project is based on game theory. We therefore begin by introducing key definitions and assumptions relevant for the setup of the simulations. Firms are modeled as rational agents whose objective is to maximize profits. The primary strategic variable for each firm is its price, and thus, pricing constitutes their set of available actions. Therefore, in our simulations, a firm's strategy is defined by its pricing choices, and different combinations of strategies may lead to distinct equilibrium outcomes.

2.2.1 Nash Equilibrium, SPNE and MPE

A **Nash equilibrium** is a strategy or set of strategies in a game where no agent has any incentive to deviate from their strategy, given what the other agents are playing. This ensures that in equilibrium, each agent's strategy is a best response to the other's (Tadelis, 2013, p. 80).

In our simulation we are playing a repeated game, which leads us to the definition of the Subgame Perfect Nash Equilibrium (SPNE). A SPNE is a refinement of the Nash equilibrium and means that every subgame of the game is a Nash equilibrium (Tadelis, 2013, p. 157).

Maskin and Tirole (1988) introduce the concept of a Markov Perfect Equilibrium (MPE). An MPE is a special type of SPNE with the assumption that the strategy of each player only depends on the current "state" of the game — not the full history. This assumption is known as the **Markov assumption**, and in our model, the state is simply the price set by the opponent in the previous round. The concept of MPE's will become important later, as they will lay the foundation for the competitive benchmark used in our simulations.

2.3 Economic environment

Our model adopts the sequential pricing duopoly and Q-learning simulation framework as outlined in Klein (2021). The following description of the economic environment is based on his paper.

Two firms, i and j take turns on setting prices in infinitely repeated discrete time indexed by $t = \{1, 2, 3, ...\}$. Thus, firm i sets price $p_{i,t}$ and firm j sets the price $p_{j,t}$ in period t. The prices that the firms can choose are discrete, evenly spaced and between 0 and 1. The number of different prices that the firms can set are determined by k, such that the price interval is denoted by $P = \{0, \frac{1}{k}, \frac{2}{k}, \frac{3}{k}, ..., 1\}$. Prices are set sequentially, such that in even time periods firm p_i sets their price, and in odd time periods p_j sets their price.
Assuming no marginal or fixed costs, the profit function for firm i, in time period t is described as:

$$\pi_i(p_{i,t}, p_{j,t}) = p_{i,t} D_{i,t}(p_{i,t}, p_{j,t}) \tag{1}$$

where p_{jt} is the price of the opposite firm, and D_{it} is the demand as a function of its own price $p_{i,t}$, and the price of the opposing firm, $p_{j,t}$. Firms discount future profits with $\delta \in [0, 1)$, such that it is the objective of each firm to maximize, at time t, the following:

$$\max \sum_{s=0}^{\infty} \delta^s \pi_i(p_{i,t+s}, p_{j,t+s}) \tag{2}$$

The demand function is defined as:

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

Which means that the firm with the lower price takes the whole market. If both firms set the same price, they split the market evenly.

The monopolist's profit corresponds to the joint-profit maximizing profit. Therefore, we calculate the monopolist's profit, such that it can later be used as a benchmark. This allows for a clearer interpretation of the profit levels observed in the simulation, by comparing them to the theoretical maximum achievable profit through full cooperation.
The monopolist's demand function is $D(p) = 1 - p$ yielding a profit of $\pi(p) = p \cdot D(p) =$

$p \cdot (1-p) \Leftrightarrow \pi(p) = p - p^2$. Maximizing the profit with respect to p:

$$\frac{d\pi(p)}{dp} = 1 - 2p = 0 \Rightarrow p^M = \frac{1}{2} \tag{4}$$

The joint profit maximizing price is thus $p^M = \frac{1}{2}$, with a corresponding profit for each firm of:

$$\pi_{ij}^M(p^M) = 0.5 \cdot \frac{1-0.5}{2} = 0.125 \tag{5}$$

As previously noted, following Klein (2021) and Maskin and Tirole (1988), we adopt the Markov assumption: firm strategies depend solely on the price set by the competitor in the previous round, $p_{j,t-1}$ for firm i and symmetrically for firm j, making earlier history irrelevant. This also means that there is no communication between the firms, beyond the observation of this price.

The following value function is a Nash equilibrium for all prices along the equilibrium path, if the condition holds for both firms:

$$V_i(p_{jt}) = \max_{\text{p}}[\pi_i(p, p_{jt}) + E_{p_{j,t+1}}[\delta\pi_i(p, p_{j,t+1}) + \delta^2 V_i(p_{j,t+1})]] \tag{6}$$

Equation (6) is written in the style of Klein (2021), who has the equation from Maskin and Tirole (1988). If you include off-equilibrium prices, for all prices a strategy pair (R_1, R_2) is MPE if equation (6) holds.

2.3.1 Edgeworth Price Cycles and Focal Pricing

Further, Maskin and Tirole (1988) show that if firms value future profits sufficiently [1], two kinds of MPE's exist, namely **focal pricing** and **Edgeworth price cycles**.

Focal pricing refers to an equilibrium where both firms continuously set the same price. This outcome is sustained by the mutual expectation that any attempt to undercut will be matched by the competitor, which would trigger a costly price war. Because both firms value future profits sufficiently, neither is willing to initiate such a war. Similarly, neither firm

[1]When Maskin and Tirole (1988) talk about valuing future profits 'sufficiently' high with a 'sufficiently' high discount factor, δ, they mention one close to 1, without giving a specific number. Klein (2021) uses a δ value of 0.95.

is inclined to raise its price, anticipating that the competitor would not follow and then it would lose the entire market by doing so (Maskin and Tirole (1988)).

Edgeworth price cycles are when firms gradually undercut each other, to increase market share, until the price war becomes too costly, and one of the firms resets the price to a 'high' price, and the undercutting begins again (Maskin and Tirole (1988)). In this way, a cycle pattern arises, which an example of can be seen in Figure 1.

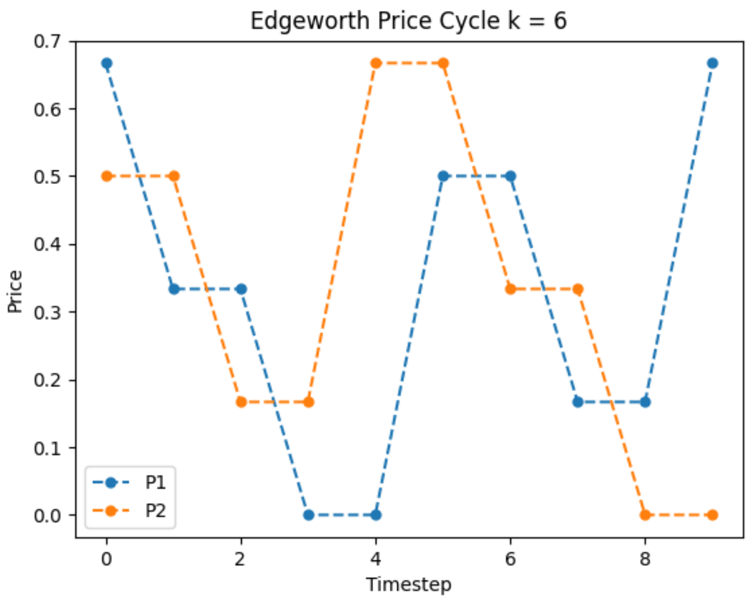

Figure 1: The most competitive Edgeworth price cycle for two firms with $k = 6$. Firms undercut each other until the first firm to observe the lowest price resets the price cycle. In this case firm 2 is the first to reset the cycle, and then the next reset is done by firm 1.

One particular form of this cycle is known as the **most competitive Edgeworth price cycle**, and it serves as a benchmark for competitive outcomes in Klein (2021). In this competitive cycle, firms successively undercut each other by a single price increment until the price reaches marginal cost (which is zero in this setting). Once the marginal cost is reached, the first firm to observe this price resets its price to one increment above the monopoly price, and the process of undercutting repeats. Figure 1 illustrates this specific cycle for $k = 6$.

The most competitive Edgeworth price cycle changes with the price interval, determined by k. This dependency is central to the later analysis, where we examine the effects of varying k. Therefore, we simulate the most competitive Edgeworth cycles for different values of k and calculate the corresponding per-period profits for the firms under equal sharing.

These profit levels will serve as a benchmark for competitive outcomes, consistent with the approach in Klein (2021). The results are shown in Table (1).

k	Per period profit of the most competitive Edgeworth cycle
6	0.0611
12	0.0699
24	0.0758
48	0.0793
100	0.0813

Table 1: The per period profit associated with the most competitive Edgeworth price cycle for different values of k.

Another potential competitive benchmark is the static Nash equilibrium, where both firms set prices at marginal cost (yielding a profit of 0), or one increment above marginal cost. However, due to the dynamics of the game, the static Nash outcome is not a suitable benchmark for a repeated game (Klein (2021)).

3 Reinforcement learning theory

3.1 Introduction to Reinforcement Learning

Reinforcement learning (RL) is a type of machine learning which differs from supervised and unsupervised learning. In supervised learning, the model learns from labeled input-output pairs to find the underlying function and pattern, such as ordinary least squares (OLS) regression fitting a linear model. Unsupervised learning uses unlabeled data to find patterns or structures, for example, K-means clustering, which groups data into clusters, again finding underlying patterns. RL, on the contrary, learns the best actions through repeated interaction with the environment. It follows a trial-and-error approach, trying actions and learning which ones yield the highest rewards in the given environment. A common example of an RL algorithm is Q-learning, which is the learning algorithm that will be used in this project (Albrecht et al., 2024, p. 19–21).

A more general definition of RL by (Albrecht et al., 2024, p. 20) describes it as algorithms that learn solutions for sequential decision processes via repeated interaction with an environment. This explains RL also being called a "trial and error" approach. Albrecht et al. (2024) will also serve as the primary source for this theory section and its subsections.
A sequential decision process involves an agent making decisions and observing outcomes, within an environment, one time step at a time. To define such processes, a decision process

model must guide decisions, and a learning objective must evaluate each strategy. Together, they form a reinforcement learning problem, illustrated in Figure 2 (Albrecht et al., 2024, p. 20). For our project, we used Markov Decision Processes (MDP) as the decision process model for our implementation of the Q-learning algorithm.

The environment in which we apply RL algorithms determines whether we are dealing with Single-Agent or Multi-Agent RL. Single-Agent RL involves an agent interacting alone in the environment, while Multi-Agent RL involves multiple agents interacting within the same environment (Albrecht et al., 2024, p. 19–21). In the following subsection, we focus on the single-agent framework and MDP, and in later subsections relate the general Q-learning setup to our economic environment.

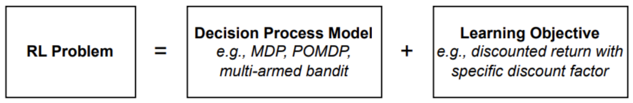
RL Problem
=
Decision Process Model
e.g., MDP, POMDP, multi-armed bandit
+
Learning Objective
e.g., discounted return with specific discount factor

Figure 2: Figure from (Albrecht et al., 2024, p. 19-21)

3.2 Single Agent Reinforcement Learning

3.2.1 Markow Decision Processes

As mentioned above, Markov Decision Processes (MDP) are the decision process model we focus on. MDP is the standard model for sequential decision processes and forms the foundation for Single-Agent RL (Albrecht et al., 2024, p. 19–22). Simply put, MDP provides a structured way to model how an agent makes a series of decisions over time within an (economic) environment.

Definition 1: Markov Decision Process (Albrecht et al., 2024, p. 22) A Discrete (MDP) consists of:

1. **Finite set of states** S, with a subset of terminal states $\bar{S} \subset S$.
2. **Finite set of actions** A.
3. **Reward function** $\mathcal{R} : S \times A \times S \to \mathbb{R}$.
4. **State transition probability function** $\mathcal{T} : S \times A \times S \to [0,1]$ such that

$$\forall s \in S, a \in A : \sum_{s' \in S} \mathcal{T}(s, a, s') = 1$$

5. **Initial state distribution** $\mu : S \to [0, 1]$ such that

$$\sum_{s \in S} \mu(s) = 1 \quad \text{and} \quad \forall s \in \bar{S}, \mu(s) = 0$$

Let us now explain the items of the definition. **Finite set of states** S represents all possible situations, in which an agent can be. In our economic environment, a firm's state is determined by the competitor's price. Instead of specific terminal states where the agent stops, we end our simulation by setting a maximum number of time steps, which serves the same purpose.

A **finite set of actions** A is equivalent to the possible actions an agent can take. In our economic environment, these are the firm's different pricing options, determined by the parameter k.

The **reward function** determines the reward of an action and thereby how well the action performs. In our economic environment, it corresponds to the profit function.

The **state transition probability function** describes and dictates how the environment changes and can be seen as the rulebook of the game. This function has only theoretical relevance in this project, which is clarified in Section 3.2.3.

The **initial state distribution** defines the probability for the agent to start in each of the possible states. For example, if a firm has two possible starting states s_1 and s_2 (determined by k), the initial state distribution might look like $P(s_1) = 30\%$ and $P(s_2) = 70\%$, meaning the firm has a 30% chance to start in state s_1 and a 70% chance to start in state s_2.

One might think that the Markov Decision Process relates to the Markov assumption (also called the Markov property) introduced in Section 2.2, and this is indeed correct. The Markov assumption presents that the future state (competitor price) and reward (profit) are conditionally independent of past states and actions, given the current state (competitor price) and action (own price). The assumption is mathematically expressed in Equation 7.

$$P(s_{t+1}, r_t | s_t, a_t, s_{t-1}, a_{t-1} \ldots, s_0, a_0) = P(s_{t+1}, r_t | s_t, a_t) \tag{7}$$

This means providing the agent with only the current state is sufficient for it to take optimal actions in an MDP (Albrecht et al., 2024, p. 23). Limiting the agent in this way also reduces coding complexity and computational demands, as past states and actions do not need to be stored or used.

3.2.2 Value functions and Bellman equation

Moving forward we will have to define a strategy function, that returns a probability from which the MDP chooses an action conditioned on the state. It will be denoted as: $\mathcal{P}(a_t, s_t)$. When working with MDP, one can define a solution as an optimal strategy $\mathcal{P}^*$, if the strategy results in the maximum expected discounted return. Therefore, in the following section $*$ denotes optimality. To identify this strategy we will use the Bellmann Equation, also known as the state value function, which can be seen in equation (8).

$$V^{\mathcal{P}}(s) = \sum_{a \in A} \mathcal{P}(a|s) \sum_{s' \in S} \mathcal{T}(s'|s,a)[\mathcal{R}(s,a,s') + \delta V^{\mathcal{P}}(s')] \tag{8}$$

Where s' denotes the future states and δ is the discounting parameter. The purpose of equation (8) is to value the expected return when selecting actions with strategy $\mathcal{P}$, starting in state s. From the same logic, we can now define the action value function $Q^{\mathcal{P}}(a,s)$ in equation (9).

$$Q^{\mathcal{P}}(a,s) = \sum_{s' \in S} \mathcal{T}(s'|s,a)[\mathcal{R}(s,a,s') + \delta \sum_{a' \in A} \mathcal{P}(a'|s') Q^{\mathcal{P}}(a',s')] \tag{9}$$

$Q^{\mathcal{P}}(a,s)$ returns the expected value but this time when selecting an action a, in state s and then following strategy $\mathcal{P}$ to select actions afterwards. Since we are in the MDP, the strategy is optimal if its corresponding value function is optimal. This then leads us to Bellman Optimality Equations seen in 10 and 11

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} \mathcal{T}(s'|s,a)[\mathcal{R}(s,a,s') + \delta V^*(s')] \tag{10}$$

$$Q^*(s,a) = \sum_{s' \in S} \mathcal{T}(s'|s,a)[\mathcal{R}(s,a,s') + \delta \max_{a' \in A} Q^*(s',a')] \tag{11}$$

Equations 10 and 11 illustrate that the optimal strategy in an MDP relies on its value function being optimal. To find the optimal strategies for given states, we select actions that maximize the value, as shown in equation 12.

$$\mathcal{P}^*(s) = \arg\max_{a \in A} Q^*(s,a) \tag{12}$$

Since the optimal strategy $\mathcal{P}^*$ can be recovered by choosing actions that maximize equations (10) and (11), making an explicit notation of $\mathcal{P}(a|s)$ in these, is unnecessary.

Two families of algorithms are used to calculate optimal value functions and thereby optimal strategies, respectively Dynamic programming and Temporal Difference learning. While Dynamic Programming estimates value functions using complete knowledge of the MDP, Temporal difference learning updates value estimates based on interactions with the environment, making the latter of relevance to Q-learning and by extension our project. (Albrecht et al., 2024, p. 26-28)

3.2.3 Temporal difference Learning and Q-learning

Temporal difference learning (TD) is a subset of reinforcement learning where value functions and optimal strategies are learned through experience generated in the following loop: an agent chooses an action, to which the environment responds by changing the state, which the agent observes a reward, through a reward function, before choosing the next action. The reward function $\mathcal{R}(s^t, a^t, s^{t+1})$ is now simplified to r^{t+1}. TD learning relies on interaction with the environment and therefore does not require full knowledge of the MDP, making the state transition function unnecessary in the update equations. As a result, the Bellman optimality equations (10) and (11) can be modified to the TD versions shown in equations (13) and (14).

$$V^*(s) = \max_{a \in A} [r^{t+1} + \delta V^*(s^{t+1})] \tag{13}$$

$$Q^*(s,a) = [r^{t+1} + \delta \max_{a' \in A} Q^*(s^{t+1}, a')] \tag{14}$$

In temporal difference learning, the update rule for learning the action value function $Q(s,a)$, uses equation (14) and is shown in equation (15) just below.

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha * [\underbrace{r^{t+1} + \delta \max_{a' \in A} Q(s^{t+1}, a')}_{\text{Equation 14}} - Q(s^t, a^t)] \tag{15}$$

The update rule in Equation 15 estimates the action value function $Q(s,a)$ as it is updating values for each state-action pair based on interactions with the environment at each time step. This effectively forms the Q-matrix of size $A \times S$. Here, α is the learning rate or step size, ranging between 0 and 1, which controls how much new interaction information is weighted against prior knowledge. Algorithm 1 brings all this together, presenting the pseudocode for the single-agent Q-learning algorithm that employs this update rule.

Algorithm 1 Q-learning for MDPs using the Epsilon Greedy Strategy Method

```
1: Initialize: $Q(s, a) = 0$ for all $s \in S, a \in A$
2: Repeat for every episode:
3: for $t = 0, 1, 2, \ldots$ do
4:     Observe current state $s^t$
5:     With probability $\epsilon$: choose random action $a^t \in A$
6:     Otherwise: choose action $a^t \in \arg\max_a Q(s^t, a)$
7:     Observe own reward $r^{t+1}$ and next state $s^{t+1}$
8:     $Q_i(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha[r^{t+1} + \delta \max_a Q(s^{t+1}, a) - Q(s^t, a^t)]$
9: end for
```

Algorithm 1 uses the Epsilon-Greedy Strategy Method. The method is a way for making sure the Q-learner explores the action-state space, while still converging to an optimal strategy. The method is shown below in equation (16)

$$a^t \begin{cases} \sim \mathrm{U}(A) & \text{with probability } \epsilon^t \\ \arg\max_a Q(s,a) & \text{with probability 1-}\epsilon^t \end{cases} \tag{16}$$

It shows how an action is chosen with a probability determined by ϵ to either pick a random action from a uniform distribution or an action that maximizes the action-value function $Q(s, a)$. This is lines 5–6 in Algorithm 1. The value of ϵ decreases gradually over time, reducing exploration and ensuring the Q-learner eventually converges to a stable strategy, meaning it stops taking random actions to estimate the values of the state-action space. (Albrecht et al., 2024, p.32-35)

3.3 Multi Agent Reinforcement Learning

Having established the theory and methods behind Single-Agent RL, we now move onto Multi-Agent RL. The relevance of showing both single- and multi-agent frameworks will now become clear. Figure 3 from (Albrecht et al., 2024, p. 44) illustrates the Multi-Agent RL hierarchy, showing that Repeated Normal-Form Games and MDPs are special cases of Stochastic Games, which themselves are special cases of Partially Observable Stochastic Games. The figure also demonstrates that the number of states and agents classify each type of game. In this project, we focus on Q-learning agents learning by competing against each other. Since the states are fully observed and the agents interact in the same environment, 3 shows why we limit our scope to Stochastic Games and MDPs.

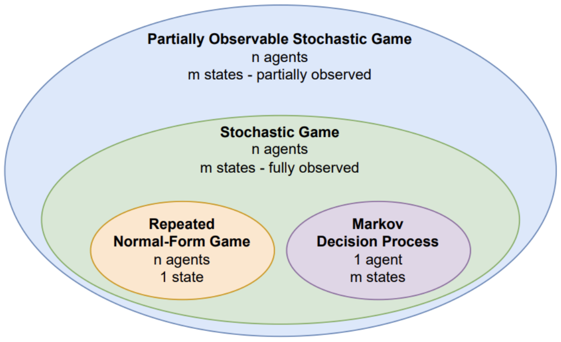
Partially Observable Stochastic Game
n agents
m states - partially observed
Stochastic Game
n agents
m states - fully observed
Repeated Normal-Form Game
n agents
1 state
Markov Decision Process
1 agent
m states

Figure 3: Multi-Agent RL Hierarchy

3.3.1 Stochastic games and Independent Learning

Most of the theory underlying Stochastic Games is similar to MDP's and therefore, to avoid repetition, the following will seem more superficial. A more formal definition of a Stochastic Game can be seen in appendix 9.1.1.

The definition of a stochastic game extends the MDP definition (Definition 1) by accommodating multiple agents in the same environment through the introduction of indices. This makes it of relevance since our game will be featuring 2 agents (firms). Like MDPs, stochastic games follow the Markov assumption shown in Equation 7, making it no surprise that stochastic games are also called Markov games (Albrecht et al., 2024, p. 48).

Multi-agent RL can be utilized to learn strategies for real-world problems, which vary widely, therefore requiring different approaches. In this project, we reduce the multi-agent RL game into multiple single-agent RL games within the multi-agent environment. This standard approach simplifies the problem by transforming multi-agent learning problems into single-agent learning problems. Specifically, there are two main methods to implement this, namely Centralized Learning and Independent Learning.

Centralized Learning handles the joint action space of multiple agents as a single action space and applies single-agent RL to learn a central strategy that selects actions for all agents. This approach is suitable in cooperative environments, such as robot lawn mowers collaborating to cut a field of grass most efficiently.

Independent Learning applies single-agent RL to each agent separately, treating other agents as part of the environment, so their actions are perceived as reactions from the environment, therefore reflected in the states (Albrecht et al., 2024, p. 95). Each agent learns its own strategy independently, essentially ignoring the direct presence of others. This approach suits competitive environments, where agents work against each other, which is why we use it in this project, where firms act as competitors.

The Independent Learning approach to multi-agent RL using Q-learning is shown in pseudocode in Algorithm 2 (Albrecht et al., 2024, p. 98). It closely resembles Algorithm 1, with the key difference being the indices for managing multiple agents and their corresponding state and action spaces.

Algorithm 2 Independent Q-learning (IQL) for stochastic games

```
Algorithm controls agent $i$
1: Initialize: $Q_i(s_i, a_i) = 0$ for all $s_i \in S_i, a_i \in A_i$
2: Repeat for every episode:
3: for $t = 0, 1, 2, \ldots$ do
4:     Observe current state $s_i^t$
5:     With probability $\epsilon$: choose random action $a_i^t \in A_i$
6:     Otherwise: choose action $a_i^t \in \arg\max_{a_i} Q_i(s_i^t, a_i)$
7:     (meanwhile, other agents $j \neq i$ choose their actions $a_j^t$)
8:     Observe own reward $r_i^{t+1}$ and next state $s_i^{t+1}$
9:     $Q_i(s_i^t, a_i^t) \leftarrow Q_i(s_i^t, a_i^t) + \alpha[r_i^{t+1} + \delta \max_{a_i} Q_i(s_i^{t+1}, a_i) - Q_i(s_i^t, a_i^t)]$
10: end for
```

3.4 Linking to Economic Environment

As mentioned in section 2.3, this project simulates a game between two firms, i and j competing in a sequential duopoly modeled as a Bertrand economy, therefore by setting prices. Translating the RL theory to this economic environment, we get the following action space, state space, and reward function for firm i are defined as follows, with a symmetrical setup for firm j:

$$A_i = P = \{0, \frac{1}{k}, \frac{2}{k}, \frac{3}{k}, ..., 1\}, \text{ where } p_i \in P$$

$$S_i = P = \{0, \frac{1}{k}, \frac{2}{k}, \frac{3}{k}, ..., 1\}, \text{ where } p_j \in P$$

$$\mathcal{R}(a_i, s_i) = \pi_i(p_i, p_j)$$

The action space is all prices the firm can set, determined by the parameter k. The state space corresponds to the competitor's price and is therefore also given by k. The reward function is now the firm's profit function.

Algorithm 2 shows Q-learning using independent learning, where firms play simultaneously, as all agents set prices at each time step. However, the decision process we simulate is turn-based, as firms take turns setting their price, just as in Klein (2021). This means that only one firm will set a price at each time step.

Our simulations implement this sequential turn-based pricing using the following reward definition, inspired by Fischer (2023), who also followed Klein (2021):

$$r_i^{t+1} = \pi_i(p_i^t, p_j^t) + \delta \pi_i(p_i^t, p_j^{t+1}) \tag{17}$$

The reward definition in equation 17 is tailored to this turn-based setting. It accounts for the fact that firm i never sets prices in two consecutive time steps by defining its reward as the sum of the current profit and the discounted profit in the next time step, when it is firm j's turn. The discount factor δ accounts for the future profit one time step ahead. Similarly, the action value function's $(Q(a, s))$ update rule must be tailored accordingly, resulting in a modified version of equation 15:

$$Q(p_j^t, p_i^t) \leftarrow Q(p_j^t, p_i^t) + \alpha * [r_i^{t+1} + \delta^2 \max_{p_i' \in p_i} Q(p_j^{t+1}, p_i') - Q(p_j^t, p_i^t)], \tag{18}$$
$$r_i^{t+1} = \pi_i(p_i^t, p_j^t) + \delta \pi_i(p_i^t, p_j^{t+1})$$

Again, notice that the Q-value is now discounted for 2 time steps (δ^2), to handle the sequential turn-based setting.

It is also worth noting the similarity between equation (18) and equation (6). As noted by Klein (2021), this relationship comes from the fact that Q-learning, via recursive updating, aims to solve a dynamic programming condition.

With the new update rule from 18 and corresponding game modification, we now present the final algorithm used in our simulations. Pseudocode for Algorithm 3, inspired by Fischer (2023), can be seen below.

Algorithm 3 Q-learning Sequential Bertrand Duopoly (Greedy ϵ)

```
1: Initialize: $Q_i(p_j, p_i) = 0$ **and** $Q_j(p_i, p_j) = 0$ for all $p_j, p_i \in P$
2: Randomly choose $p_i^t, p_j^t$ for $t = \{0, 1\}$
3: Repeat for every episode:
4: **for** $t = 2, 3, 4, \dots$ **do**
5:     Observe current state (competitor price) $p_j^t$
6:     With probability $\epsilon$: choose random action (price) $p_i^t \in P$
7:     Otherwise: choose action (price) $p_i^t \in \arg\max_{p_i} Q_i(p_j^t, p_i)$
8:     Observe next state (competitor price) $p_j^{t+1}$
9:     Observe own reward $r_i^{t+1} = \pi_i(p_i^t, p_j^t) + \delta\pi_i(p_i^t, p_j^{t+1})$
10:    $Q(p_j^t, p_i^t) \leftarrow Q(p_j^t, p_i^t) + \alpha * [r_i^{t+1} + \delta^2 \max_{p_i' \in p_i} Q(p_j^{t+1}, p_i') - Q(p_j^t, p_i^t)]$
11:    Update $\{i \leftarrow j, j \leftarrow i\}$ # Other players turn
12: **end for**
```

Compared to algorithm 2, algorithm 3 uses economic terminology, where states and actions correspond to competitor and own prices, respectively. Line 1 illustrates how the Q-values for firms i and j are stored in their corresponding $P \times P$ Q-matrices. All entries are initialized to zero, but any arbitrary value would suffice. This setup allows the Q-learning algorithm to try and find the best strategy based on the competitor's actions.

Under the right conditions, convergence to an optimal strategy is guaranteed for a single-agent Q-learning setting (Albrecht et al., 2024, p. 35). However, in our multi-agent setting, convergence to an optimal strategy, cannot be assured. A key challenge is the *moving target problem*, which occurs as agents continuously adapt their strategies in response to one another, causing the optimal strategy to shift and complicate learning, essentially making the optimal strategy a moving target (Albrecht et al., 2024, p. 97–98, p. 12).

4 Setup for simulation

In this section we will describe how we are running the simulation, what parameters are being used, and how we evaluate the final outcomes.

When running the simulations, we are letting two Q-learners compete against each other, as in the theoretical frame that we set up earlier, thereby using Algorithm 3.

The setup for the simulation is following the work of Klein (2021). A total of 1,000 independent simulation runs are conducted, each running for 500,000 time steps ($T = 500,000$). To reduce the influence of short-term fluctuations particularly those occurring during the learning and price experimentation phases of the Q-learning algorithms, a running average over 1,000 time steps is applied when presenting results graphically.

4.1 Parameters

As outlined in section 3, several parameters can be chosen for the Q-learning model: δ, α and θ. Below, we showcase our choices for these parameters.
The discount factor, δ, reflects how much future profits are valued. A value of:

$$\delta = 0.95$$

is chosen, which is justified by the small time periods in the model and aligns with the value used by Klein (2021).
The learning rate, α, determines the balance between new and existing knowledge in the Q-learning algorithm. A value of:

$$\alpha = 0.3$$

is selected. Klein (2021) found this value to be a reasonable compromise between ensuring that learning is not too slow, while also making sure that the algorithm does not forget what it has already learned.
The exploration parameter ϵ_t, which determines the probability of making random actions, is given by:

$$\epsilon_t = (1 - \theta)^t \tag{19}$$

This equation ensures thorough exploration at the start, with gradual reduction over time. The decay parameter θ controls the rate of decay, and we use a value of $\theta = 0.0000275$, which in the beginning of the simulation results in an exploration probability of:

$$\epsilon_1 = (1 - 0.0000275)^1 \approx 99.9\% \tag{20}$$

While it will in the end reach:

$$\epsilon_{500{,}000} = (1 - 0.0000275)^{500{,}000} \approx 0.0001\% \tag{21}$$

4.2 Evaluating performance

To evaluate algorithm performance after convergence, the average profit is calculated based on the final 1,000 time steps in each simulation. This approach provides an estimate of the profit level achieved once the learning process has become stable. It is important to note that while the algorithms appear to have converged by the end of the simulation, this does not necessarily imply convergence to an optimal strategy. Convergence is primarily driven by the decay of the exploration parameter, which is discussed in section 4.1.

To find and analyze the strategies (such as cycles or focal prices) to which the game has converged, we focus on the final 1000 time steps. This choice is justified by the low exploration probability at this stage, that indicate the algorithms have stopped exploring and therefore converged. We observe that cycles typically do not exceed around 32 steps, even when $k = 100$ (See figure 10), thus ensuring that all cycles are identified in our analysis.
In our simulation, we chose to classify an outcome as collusive if the resulting profit exceeds the profit associated with the competitive benchmark. This benchmark is defined as the per-period profit derived from the most competitive Edgeworth cycle (Described in section 2.3.1), with profit ranging between $\pi = 0.0611$ and $\pi = 0.0813$, depending on the specific value of the parameter k (see Table 1). Additionally, we use the joint profit-maximizing benchmark of $\pi = 0.125$, as shown in Equation (5), to provide a reference point for assessing the degree of collusiveness. Together, these benchmarks allow us to evaluate whether the simulated outcomes reflect behavior, which is consistent with competitive dynamics or with more collusive, profit-maximizing strategies.
To evaluate collusion levels formally, we introduce a normalized measure (ζ) for profit exceeding the competitive level. This measure is based on the approach by Calvano et al. (2020), who use the average profit (from the simulation), the static Bertrand Nash equilibrium profit (as the competitive benchmark), and joint profit maximization profit (as the collusive benchmark). In this project, our competitive benchmark differs. Thus, our version of the measure is defined in equation 22:

$$\zeta_{k,\bar{\pi}} = \frac{\bar{\pi}_k - \pi_k^{Competitive}}{\pi^M - \pi_k^{Competitive}} \cdot 100 \tag{22}$$

where $\bar{\pi}$ is the observed average profit, $\pi_k^{Competitive}$ is our competitive benchmark (Note that it changes with k), and π^M is the joint profit-maximizing profit. It is multiplied by 100 to get it in percent. Thus, a ζ value of 0 corresponds to a profit on the same level as the competitive benchmark, while a ζ value of 100 corresponds to the full collusion profit maximizing level.

5 Results

The following section has been divided into 2 subsections. **Q-learning Collusion**, which will dive into the results showcasing whether or not collusive behavior was detected in our simulations, and whether or not the level of collusion compared to the competitive benchmark changes with k. **Characterization of Equilibrium Dynamics**, which will dive into what kind of cycles, and thereby strategies the Q-learners end up converging to.

5.1 Q-learning Collusion

The following section has been divided into subsections **Q-learner vs Q-learner**, **Random vs Random**, and **Level of Collusion for k values**. The Random vs Random game is for comparison reasons.

5.1.1 Q-learner vs Q-learner

In this section we will see that Q-learners consistently reach profits above the competitive level. We start by seeing We find that with $k = 6$, the profits are high, thereafter declining sharply, followed by a gradual increase to a new high around $k = 50$, and thereafter declining.

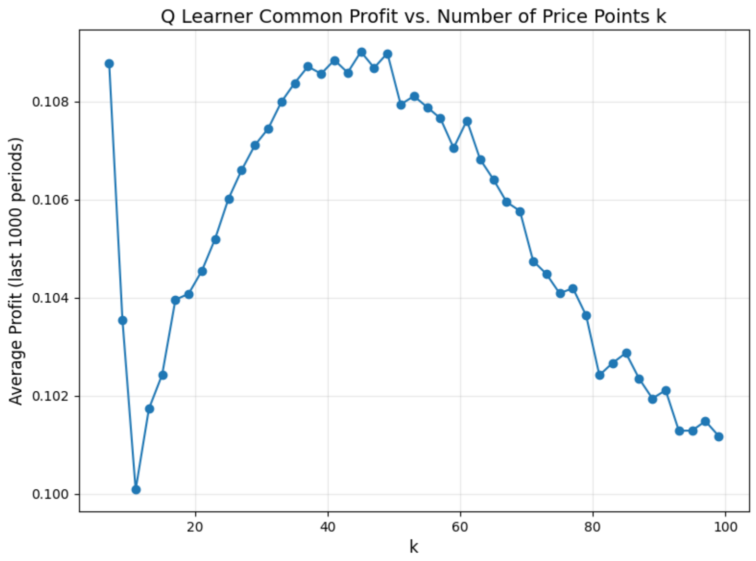

Figure 4: The average common profit under equal sharing as a result of k. Profit is calculated in the last 1000 steps of a 500.000 timestep simulation. Each value of k ran for 1000 simulations.

Figure (4) shows the trend for the profits as k rises. The declining trend observed after around $k = 50$, continues for values of $k > 100$ (See Figure 12 in the appendix). Note here that different values of k, constitute different games, in the sense that the competitive benchmark is not the same. As k rises, so does the competitive benchmark. We will further analyze Figure 4 in section 5.2.1, after introducing more relevant results in Figure 10 and 9.

Figure (5) and Figure (6) present the simulation results for two Q-learners competing when $k = 6$ and $k = 100$, respectively. Additional results for $k = 12, 24, 48$ are shown in the appendix. Across all values of k, the average common profit consistently lies above the competitive benchmark, indicating collusive behavior, but below the joint profit-maximizing

benchmark. This pattern is in line with the findings of Klein (2021).
A notable trend is that as k increases, the average profit gets closer to the competitive benchmark. For example, the profit in the simulation with $k = 100$ lies significantly closer to the competitive level than in the $k = 6$ case. This is due to both the profit falling, as seen in figure 4, as well as the competitive benchmark increasing. This declining degree of collusion with increasing k is further explored in Table (2).

Finally, the blue line in each figure, representing average common profit over time, also illustrates the learning process. Initially, when the exploration rate, ϵ, is high, the Q-learners engage in exploratory behavior and begin learning the environment, as illustrated in the gradual increase in average profits. As ϵ decays, the agents converge to stable strategies, reflected by the flattening of the profit curve.

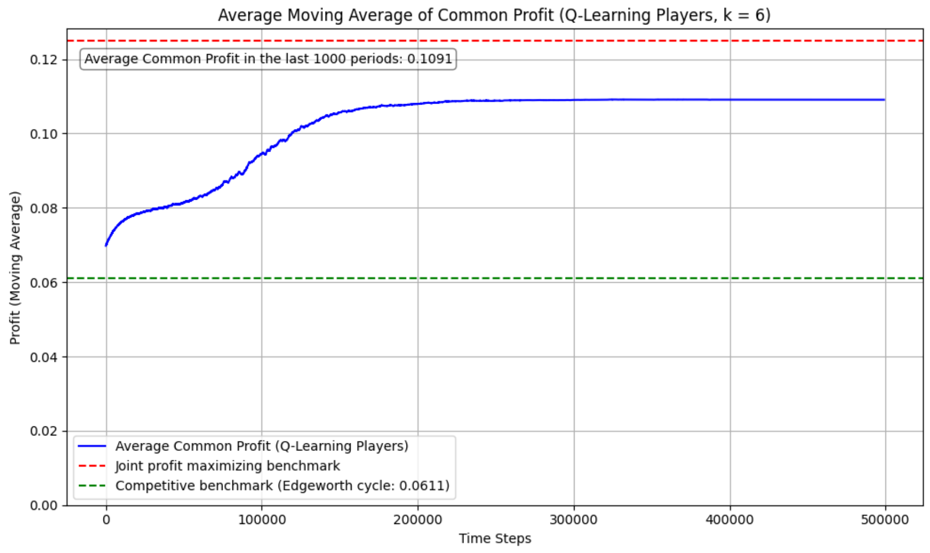

Figure 5: Q-learner vs Q-learner

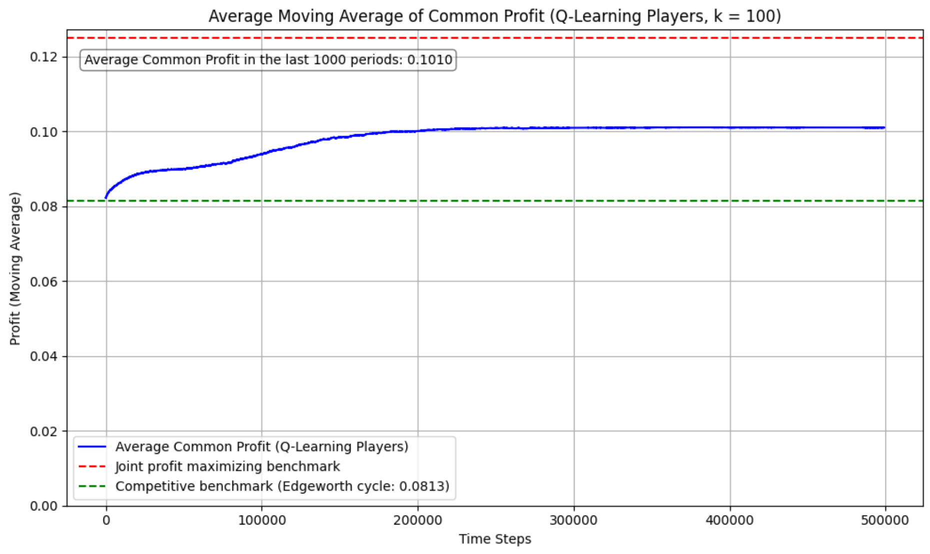

Figure 6: Q-learner vs Q-learner

5.1.2 Random vs Random

One might assume that the Q-learners' ability to reach profit levels exceeding the competitive benchmark reflects strategic sophistication. However as we shall see now, even players playing uniformly random achieve this.

Figure (7) and Figure (8) illustrate the simulation results for two random players interacting when $k = 6$ and $k = 100$, respectively. Corresponding results for $k = 12, 24, 48$ are included in the appendix. In these simulations, each player selects prices uniformly random when it is their turn, without any learning mechanism.

Unlike the Q-learning setting, these figures demonstrate an absence of learning, as evidenced by the flat path of average common profit (purple line) throughout the simulation. Although random players reach profit levels above the competitive benchmark, the magnitude of profit deviation from the benchmark is minimal compared to Q-learners. This underlines that the elevated profits in the Q-learning simulations are a result of adaptive behavior rather than random fluctuations.

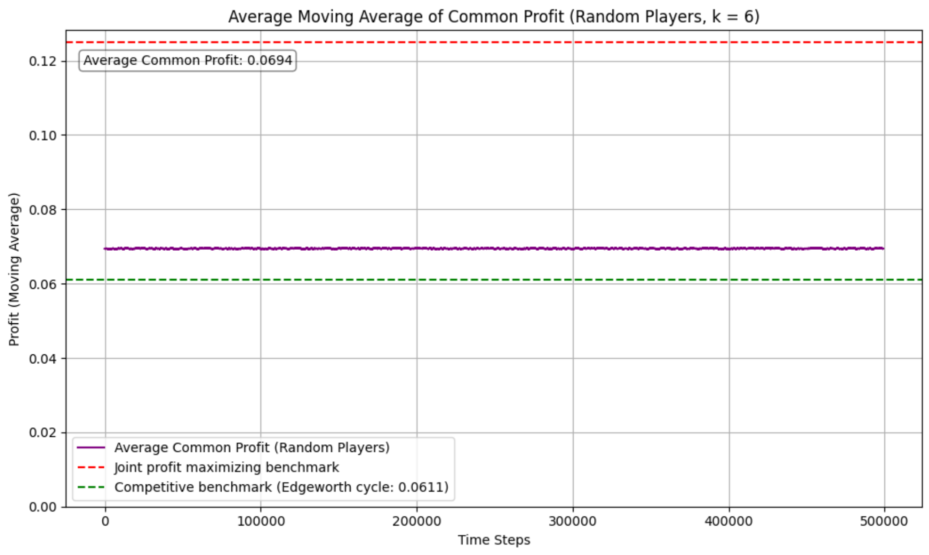

Figure 7: Random players, k = 6

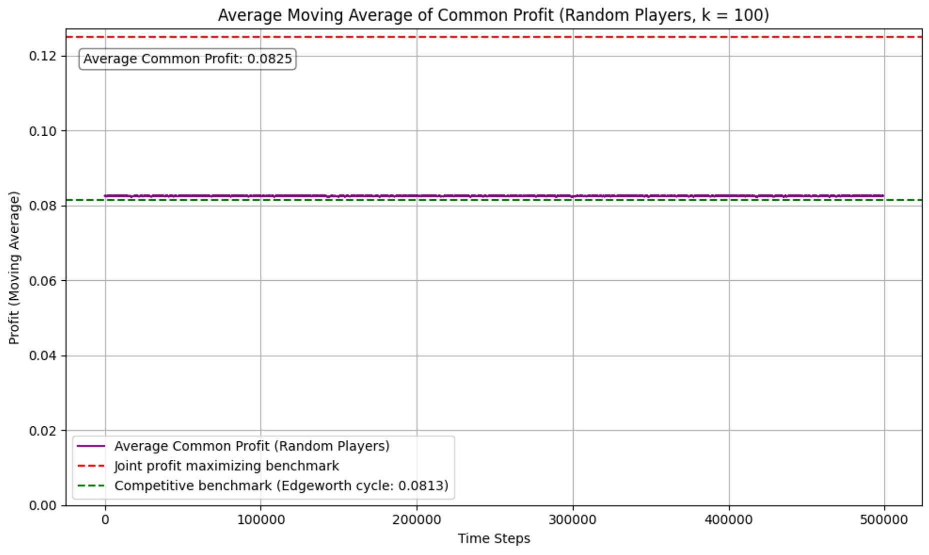

Figure 8: Random players, k = 100

5.1.3 Level of Collusion for k values

Table (2) presents the percentage deviation of the average profits calculated over the final 1,000 periods of each simulation from the respective competitive benchmark, across different values of the price interval parameter k. The table includes results for both Q-learners playing against Q-learners and random players playing against random players. Additionally, it reports the corresponding normalized collusion measure, ζ, for each game. These results are derived from the simulations shown in Section 5.1.1 and 5.1.2. For instance, when $k = 6$, the Q-learners achieved an average profit that was 78.6% above the competitive benchmark, with a corresponding ζ value of 75.1%.

k	% deviation from Comp. benchmark		ζ	
	Q-learners	Random	Q-learner	Random
6	78.6 %	13.6 %	75.1 %	13.0 %
12	46.4 %	9.3 %	58.8 %	11.8 %
24	39.2 %	5.4 %	60.4 %	8.3 %
48	36.6 %	2.9 %	63.5 %	5.0 %
100	24.2 %	1.5 %	45.1 %	2.7 %

Table 2: Illustrating the percentual deviation between the competitive benchmark and the average profit taken over the last 1000 periods for the Q-learners and the random players. In the last two columns we show the ζ values for corresponding k values.

The results in Table (2) reveal what appears to be an inverse relationship between the parameter k and the degree of collusive behavior observed. As k increases, the average profit approaches the competitive benchmark for both Q-learners and random agents, indicating a reduction of the collusive level. For example, with $k = 6$, Q-learners achieve profits 78.6% above the competitive benchmark, whereas at $k = 100$, this margin falls to just 24.2%. This pattern persists across intermediate k values (12, 24, 48), and also emphasizes the gap between the Q-learners and non-learners (random players).
The ζ values broadly confirm the overall trend that Q-learners exhibit decreasing levels of collusion as k increases. Specifically, ζ declines from 75.1% at $k = 6$ to 45.1% at $k = 100$, indicating a shift toward more competitive behavior. However, this decline is not strictly monotonic across all reported values of k. The pattern in the ζ values for the Q-learners mirrors the dynamics depicted in Figure 4: profits are initially high, then decrease sharply around $k = 12$ before gradually rising to a new peak near $k = 48$, after which they decline again. Notably, the ζ value at $k = 6$ exceeds that at $k = 48$, despite reaching similar profit levels. This discrepancy arises because the underlying game structure, and therefore the competitive benchmark, differs between these values of k.

5.2 Characterization of Equilibrium Dynamics

In this section we will further investigate what happens when the Q-learning algorithms have converged towards their strategies. Therefore we will look closer at the prices that the different Q-learners are playing in the last periods, to detect if and what cycles arise, or if they end in focal pricing, where both firms repeatedly are setting the same price.

5.2.1 Overview

We have written Python code which detects price cycles when the algorithms have converged. We are looking at the last 1000 prices set by the two firms. We define a price cycle as a situation where two firms, having previously set the same set of prices against each other, return to that exact same set of prices after a sequence of price adjustments, thus repeating the same pattern of prices indefinitely. A full cycle starts at an initial set of prices and continues through subsequent sets of prices until returning to that same initial set of prices. Notice that we are now investigating the prices set, instead of the profits that the firms are obtaining.
For all values of k, the shortest observed cycle length is 2. These instances are focal pricing and not cycles. However, they are identified as such, because of the sequential nature of the game. In this situation, both firms repeatedly set the same prices. We need to observe both firms setting their price. In step 1 firm i sets a price, which is then followed by step 2, where firm j sets a price, resulting in a 'cycle' of length 2.

As k increases, the share of runs ending with focal pricing decreases. This trend can be seen in figure 9.

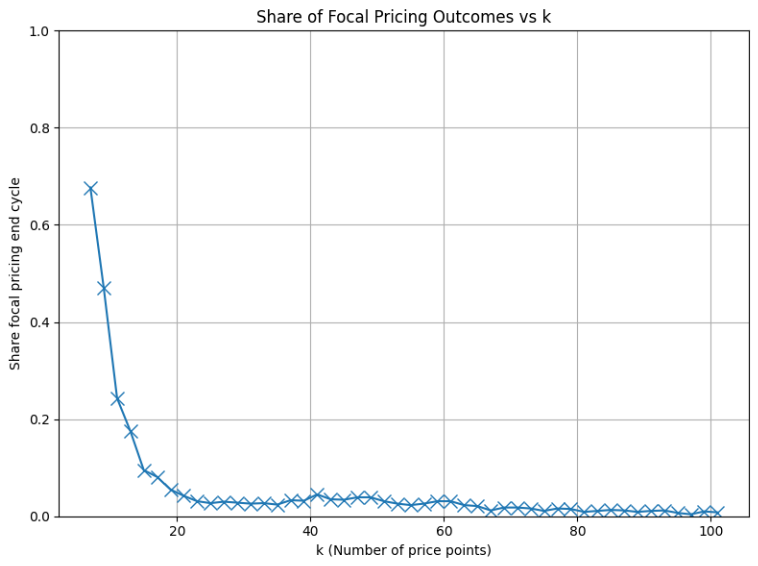

Figure 9: Share of focal pricing against k

From Figure 9 it is clear to see that when $k = 6$, about 70 % of the simulations end in focal pricing, whereas when k = 100 approximately 1 % of the runs end in focal pricing. This trend shows that the firms have a harder time coordinating on setting the same price, and thereby ending in focal pricing, as the price interval, k, grows.

The boxplots [2] in figure 10 is a visualization of the distributions of end-cycle lengths. As already seen, as k grows, the lengths of the cycles grows. This can be seen in both increases in the median and the mean. Furthermore, the longest cycle observed increases with k. This is to be expected, as smaller price intervals will allow for more and longer price cycles.

[2]The statistical software used was the library Matplotlib in Python. The box is from the quartiles Q_1 to Q_3, with the yellow line indicating the median. The whiskers extend from the box, and out to the furthest point lying within 1.5 times the inter-quartile range (IQR) from the box. Points further out than this are noted as points.

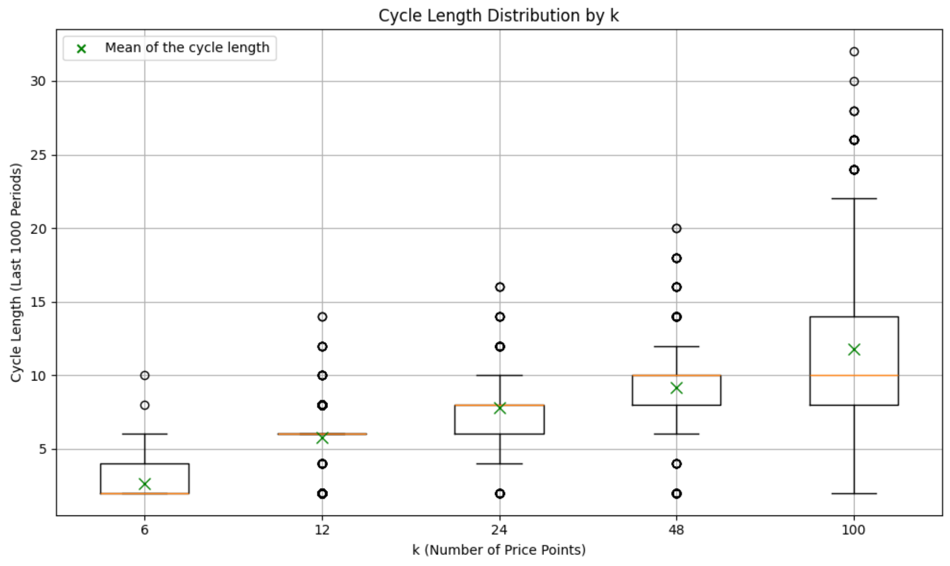

Figure 10: Boxplots for the distribution of the cycle lengths in the last 1000 periods for different values of k. Notice the mean of the cycle length marked with a green x.

Recall figure 4 which shows that profits are initially high, decline to a minimum near $k = 12$, then rise to a peak around $k = 48$, before decreasing again. We now seek to give a possible explanation to these dynamics. The primary puzzle is the notable dip in profits seen at $k \approx 12$. Drawing on Figures 9 and 10, we argue that this pattern is linked to the ability of the Q-learners to coordinate on focal (supra-competitive) prices, and the role of the "moving target problem."

At low values of k, we observe a high prevalence of focal pricing. This can be explained by the relatively limited state space, which keeps the moving target problem at a manageable level, and makes it easier for the algorithms to coordinate on (profitable) focal pricing. As k increases toward 12, the share of focal pricing declines sharply, and more of the outcomes end in cycles. But the number of possible pricing cycles remains limited due to the size of k, which is still moderate. As a result, profits fall, since the Q-learners are unable to effectively establish and maintain either profitable focal pricing or highly profitable cycles.

With an increase in k beyond this range, the situation changes: the larger action space allows for a greater variety of potential highly profitable cycles, while the moving target problem is not yet severe enough to prevent Q-learners from discovering and exploiting these cyclical strategies. Consequently, profits rise again as Q-learners increasingly learn to coordinate on more profitable cycles, as they become available. However, at around $k = 48$, the moving target problem becomes dominant. The action space is now so large that coordination, whether on focal prices or cycles, becomes difficult. This coordination problem leads to a decline in profits, despite the increase in availability of cycles with high profitability.

5.2.2 In-depth look at cycles

In this section we examine the specific pricing cycles that the algorithms converge to. As illustrated in Figure 10, these cycles vary in length, and within each cycle length, multiple distinct cycles may occur. Due to the large number of observed cycles, we focus on two representative examples: one for $k = 6$ and one for $k = 100$. To ensure representativeness, we selected the most frequently occurring cycle among those with lengths closest to the mean cycle length for each k respectively, as shown in Figure 10. For $k = 6$, although the mean is closest to a cycle length of 2, this reflects focal pricing rather than a price cycle. We therefore present a cycle of length 4. For $k = 100$, we present a cycle of length 12.

5.2.3 Cycle of length 4

Step	price setting (p_i,p_j)	profit (π_i, π_j)
1	(0.5,0.833)	(0.25, 0.0)
2	(0.5,0.333)	(0.0, 0.222)
3	(0.167,0.333)	(0.139, 0.0)
4	(0.167,0.833)	(0.139, 0.0)
Average Profit:		(0.132, 0.055)
Average Common Profit:		(0.0935)

Table 3: The most frequently observed four step cycle across 1,000 runs (29 occurrences) for k = 6. At each step, the red price indicates which firm sets the new price. Profits are shown, with blue marking the market winner. The final row presents average profits.

In Table 3 we have the most frequently occurring four-step cycle which was observed 29 times across 1,000 simulation runs for $k = 6$. In this particular cycle, firm i achieves a higher average profit per period than firm j (0.132 compared to 0.055). However, an identical cycle in which the roles are reversed, where firm j receives the higher profit, was observed 26 times. This near equal distribution of profit dominance suggests that while individual cycles may exhibit substantial differences in profit outcomes, these variations tend to balance out across repeated runs. Therefore, reporting the average profit across both firms (Average Common Profit), as we have done throughout the project, remains a meaningful and justified approach for summarizing overall performance.

We have extracted the final Q-matrices for firms i and j from the same simulation as Table

3; these Q-matrices are available in Appendix 9.2.4, as one combined Q-table. Based on the Q-table, we derive each firm's Q-learned best response (QBR) functions, shown in Equations 23 and 24.

$$QBR_i(p_j) = \begin{cases} 0 & \text{if } p_j \in \{0,\ 0.167\} \\ 0.167 & \text{if } p_j = 0.333 \\ 0.333 & \text{if } p_j = 0.5 \\ 0.5 & \text{if } p_j \in \{0.667,\ 0.833,\ 1\} \end{cases} \tag{23}$$

$$QBR_j(p_i) = \begin{cases} 0.167 & \text{if } p_i = 0.333 \\ 0.333 & \text{if } p_i = 0.5 \\ 0.667 & \text{if } p_i \in \{0.833,\ 1\} \\ 0.833 & \text{if } p_i \in \{0,\ 0.167,\ 0.667\} \end{cases} \tag{24}$$

These QBR functions explain the firms' behaviour in the observed cycle by showing their learned optimal responses to competitor prices.
In figure 11 we visualize these QBR functions.

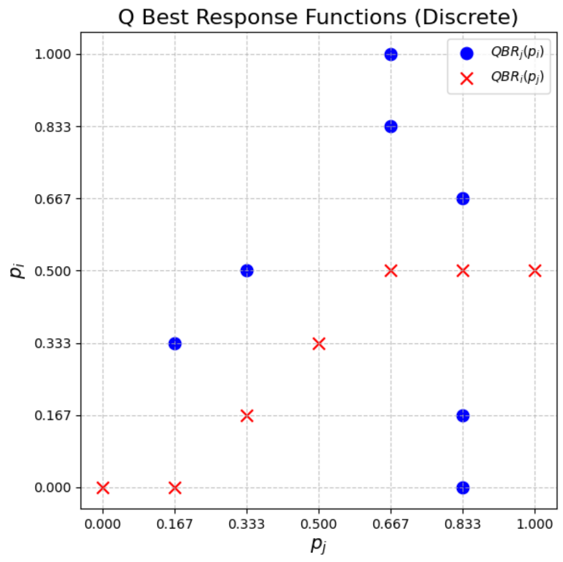

Figure 11: Visualization of QBR functions from Equation (23) and (24).

Figure 11 shows that no Nash equilibrium exists, as none of the QBR points overlap, indicating that the firms are never best responding to each other at the same price point. Furthermore the figures shows firm i's undercutting behavior, never playing prices above $p_i = 0.5$ (the monopolist's price). Firm j does not consistently undercut and frequently responds with $p_j = 0.833$. The only instance where firm j undercuts is when firm i sets

$p_i = 0.333$, to which firm j responds with $p_j = 0.167$.

Assuming the Q-matrices remain static, we also analysed whether any deviations could lead to different pricing cycles or focal pricing. From the functions 23 and 24 it is easily shown that no such deviations exist. For instance, if firm j deviates by setting a price of 0.5:

$$\begin{aligned} &QBR_i(0.5) = 0.333 \rightarrow QBR_j(0.333) = 0.167 \rightarrow \\ &QBR_i(0.167) = 0 \rightarrow QBR_j(0) = 0.833 \end{aligned} \tag{25}$$

Which leads back into the original cycle shown in Table 3. Similar analyses for all other possible deviations for both firms yield the same result, confirming the stability of the cycle under the learned QBRs.

Given this stability, we now turn to the question of whether any deviation from the cycle could nonetheless be profitable for one of the firms. Specifically, we assess whether deviating from the learned (stable) strategy yields a higher expected profit than sticking to it, or whether the resulting sequence of punishments renders deviation unprofitable. If the cost of deviating outweighs the short term gain, this would further suggest tacit collusion, as a central feature of such behavior is the ability to sustain high prices due to the expectation of a less profitable outcome following deviation (Klein (2021); Calvano et al. (2020)).

We have made the following table showcasing different expected profit averages for deviation within a specific step in the converged cycle shown in table 3

Deviating price (p)	i deviates	i deviates	j deviates	j deviates
	(p,0.333)	(p,0.833)	(0.167,p)	(0.5,p)
0	0.0	0.0	0.0	0.0
0.167	0.139	0.139	0.023	0.046
0.333	0.028	0.056	0.0	0.222
0.5	0.0	0.25	0.028	0.053
0.667	0.111	0.222	0.0	0.0
0.833	0.083	0.107	0.0	0.0
1	0.083	0.083	0.0	0.0

Table 4: The average profit a firm can obtain by one time deviation from the Q-learned stable cycle for different prices, and following the QBR's until the firms return to the stable cycle again, corresponding to the example process illustrated in Equation (25).
Black values indicate the direct profit from the price corresponding to the stable cycle strategy, meaning without any deviation: $\pi_i(0.167, 0.333) = 0.139$, $\pi_i(0.5, 0.833) = 0.25$, $\pi_j(0.167, 0.833) = 0.0$, $\pi_j(0.5, 0.333) = 0.222$.
Red values represent the average profits from one-time deviation cycles that are **less than or equal to** the corresponding stable cycle profit.
Green values indicate the average profits from deviations that are **greater than** the stable cycle profit.

Before analyzing Table 4, recall that all possible deviations eventually return to the stable cycle shown in Table 3.

From Table 4, it is evident that firm j has two deviation cycles, corresponding to $p_j = \{0.167, 0.5\}$, which yield an average one-time deviation cycle profit higher than the direct profit from $p_j = 0.833$ of $\pi_j = 0.0$ when responding to $p_i = 0.167$. While this suggests a potential short-term gain for firm j, it is important to consider the entire cycle leading back to the deviation price setting. This means returning not just to the cycle but to the exact point where the deviation occurred. This is illustrated for both deviation prices (green numbers where $p_j = \{0.167, 0.5\}$) in equations (26) and (27).

$$\begin{aligned}(p_i, p_j) = &\underbrace{(0.167, 0.167)}_{\pi_j=0.0695} \to \underbrace{(0, 0.167)}_{\pi_j=0.0} \to \underbrace{(0, 0.833)}_{\pi_j=0.0} \to \\ &\underbrace{(0.5, 0.833)}_{\pi_j=0.0} \to \underbrace{(0.5, 0.333)}_{\pi_j=0.222} \to \underbrace{(0.167, 0.333)}_{\pi_j=0.0}\end{aligned} \tag{26}$$

$$(p_i, p_j) = \underbrace{(0.167, 0.5)}_{\pi_j=0.0} \rightarrow \underbrace{(0.333, 0.167)}_{\pi_j=0.0} \rightarrow \underbrace{(0.333, 0.167)}_{\pi_j=0.139} \rightarrow \underbrace{(0, 0.167)}_{\pi_j=0.0} \rightarrow \underbrace{(0, 0.833)}_{\pi_j=0.0} \rightarrow \underbrace{(0.5, 0.833)}_{\pi_j=0.0} \rightarrow \underbrace{(0.5, 0.333)}_{\pi_j=0.222} \rightarrow \underbrace{(0.167, 0..33)}_{\pi_j=0.0} \quad (27)$$

From these equations, the average per-period profits are calculated to be $\pi_j = 0.048583$ and $\pi_j = 0.045125$ for deviation prices $p_j = 0.167$ and $p_j = 0.5$, respectively. Both are lower than firm j's average profit of $\pi_j = 0.055$ in the stable cycle prior to the deviation. Thus, a one-shot deviation results in a less profitable cycle in the longer term.

The implemented Q-learners can only play pure strategies (as also noted by Klein (2021)) and thus cannot play mixed strategies upon convergence. This means they cannot alternate between responding with different prices for the same competitor price. Consequently, one-shot deviations are practically impossible when the algorithms have converged.

Furthermore, if firm j changed its QBR to consistently respond with one of its deviation prices $p_j = \{0.167, 0.5\}$ when $p_i = 0.167$, the stable cycle would inevitably shift to instead follow the cycles shown in equations 26 and 27. This further illustrates why a permanent change would result in a short-sighted gain followed by a significant long-term loss.

From the above analysis, it is clear to see that the short-sighted gains are punished in the long term. Specifically, firms sustaining high prices due to fear of long-term profit loss is an indicator of collusion Klein (2021); Calvano et al. (2020).

5.2.4 Cycle of length 12

In table 5 we have a randomly selected 12-step cycle across the 1,000 simulation runs for k = 100. The cycle length of 12 was chosen because this is the average cycle length the simulations ended up in when k=100 (See 10).

Step	price setting (p_i,p_j)	profit (π_i, π_j)
1	(0.340,0.420)	(0.224, 0.0)
2	(0.340,0.280)	(0.0, 0.202)
3	(0.270,0.280)	(0.197, 0.0)
4	(0.270,0.140)	(0, 0.120)
5	(0.910, 0.140)	(0, 0.120)
6	(0.910, 0.590)	(0, 0.242)
7	(0.330, 0.590)	(0.221, 0.0)
8	(0.330, 0.240)	(0, 0.182)
9	(0.120, 0.240)	(0.106, 0.0)
10	(0.120, 0.980)	(0.106, 0.0)
11	(0.600, 0.980)	(0.240, 0.0)
12	(0.600,0.420)	(0.0, 0.244)
Average Profit:		(0.091,0.093)
Average Common Profit:		0.092

Table 5: An observed 12 step cycle across 1,000 runs for k = 100. At each step, the red price indicates which firm sets the new price. Profits are shown, with blue marking the market winner. The final rows presents average profits.

In Table 5 we see a cycle with no clear dominant firm. Even though firm j achieves a slightly higher average profit, the outcome is arguably balanced. Unlike the cycles observed for $k = 6$, both firms take turns resetting the cycle, after undercutting each other: firm i increases its price to 0.910 at step 5, and firm j resets at step 10 by raising its price to 0.980. This pattern resembles an Edgeworth cycle, however just not the most competitive one. Both firms obtain average profits exceeding the per period profit of the most competitive Edgeworth benchmark (0.092 compared to 0.081). A key reason for this is that the firms reset the cycle before prices reach marginal cost (0), indicating that they have learned to sustain more profitable strategies than the most competitive Edgeworth cycle.

6 Discussion

Our simulations confirms the findings of Klein (2021) by demonstrating that algorithmic collusion can occur. However, the extent of collusion varies depending on the parameter, k, which determines the number of possible prices in the Q-learning environment. The parameter k plays a crucial role in shaping the equilibrium dynamics: a lower k typically results in shorter cycle lengths, which typically results in more collusive behavior. In contrast, higher values of k increase the size of the action space, leading to longer price cycles and generally resulting in a reduced degree of collusion. As k increases, we observe a lengthening of undercutting phases within the price cycles, which then reduces the average profit obtained by the firms, for instance, from $k = 6$ to $k = 100$, average profits decline. This could suggest that coordination becomes increasingly difficult as the number of pricing options expands. Intuitively, a finer price grid introduces greater complexity to the environment, worsening the moving target problem and making it harder for the algorithms to identify and sustain supra competitive prices. Moreover, since the number of Q-values that must be learned grows with k, parameters that are calibrated for low k-values like $k = 6$ may no longer yield optimal performance in environments with larger action spaces such as $k = 100$.

We evaluated the algorithms in the end of the runs after convergence. However, this convergence may reflect the decay of the exploration rate, ϵ, rather than the algorithm learning an optimal equilibrium strategy. We still consider the algorithms to have converged because this stabilization occurs after an initial learning phase. Moreover, the extensively examined cycle (of length 4, for $k = 6$) showed no deviations that could disrupt it, indicating stability. Given the flattening of the profit curves (see Figures 5 and 6) towards the end, it is reasonable to assume similar stability across cycles in other runs.

Examining a specific converged cycle for $k = 6$, we found more evidence of collusive and Edgeworth-like cyclical behavior. Notably, one firm could have undercut for short-term gain but chose not to, which is a classic indicator of collusion Klein (2021); Calvano et al. (2020).

The cyclical pricing behavior we observed across different k values aligns with empirical evidence from Noel (2007), who documents price cycles in Toronto's gasoline market resembling Edgeworth cycles. Similar cycles appeared in our simulations, specifically in the 12-step cycle for $k = 100$. This correspondence indicates that our model captures realistic market pricing dynamics rather than purely theoretical patterns.

Additionally, the cyclical pricing observed in our simulations aligns with empirical findings by Musolff (2022), who analyzes Amazon Marketplace data and shows that automated pricing algorithms can induce tacit collusion via strategies that periodically reset prices to encourage competitors to raise theirs. These pricing cycles resemble Edgeworth cycles, and his study suggests such algorithmic pricing can sustain average prices near monopoly levels. This supports the notion that the cycles and collusion observed in our Q-learning simulations reflect behaviors present in some modern digital markets.

Further, Assad et al. (2020) studied the German gasoline market and found empirical evidence that the adoption of algorithmic pricing increased profits. In a duopoly, margins increased by 28% when both stations used algorithmic pricing. The findings of Assad et al. (2020); Musolff (2022); Noel (2007) support the validity of our simulation approach, as empirical evidence aligns with observed collusion and pricing patterns.

The parameters used may have been specifically tuned for a $k = 6$ Q-learner, which exhibits stronger collusive behavior. This suggests that the reduced collusion at $k = 100$ could result from parameters not being optimized for that setting. Consequently, it may imply that Klein's original simulation setup was biased toward achieving collusion. However, we argue that the best and most optimally chosen parameters most accurately represent Q-learning behavior, therefore making it unable to be biased towards collusive behavior. The algorithm is not designed to be collusive but rather to optimize its objective, which is to maximize profits. Any collusive behavior emerges as a byproduct of optimizing their strategy rather than intentional design.

Our simulations use a simplified economic environment. The profit function and the demand function remain fixed and do not adapt as they would in real markets. Consequently, the Q-learners' behavior here may differ from their performance in more complex, adaptive settings. Klein (2021) similarly notes that the stylized nature of the economic environment limits the Q-learning algorithm and that the Q-learner would face greater challenges in more realistic scenarios. Furthermore, Q-learning is a simple reinforcement learning algorithm, and real-world firms would likely use more advanced algorithms. Thus, just as Klein (2021) emphasizes, the Q-learning setup in this project serves primarily as a proof of concept for collusive behavior through algorithmic pricing.

One may question the realism of a k value of 6, corresponding to a price grid with only seven different possible prices. For gasoline markets, $k = 6$ is less realistic compared to $k = 100$, as gasoline prices typically change in small increments (as little as cents, U.S. Energy Information Administration (2024)), making a finer price grid more appropriate. When initially choosing to investigate larger k values, the increased realism was a key motivation, as well as an interest in how the results would change. Regardless of the k values, our simulations demonstrate that algorithmic pricing collusion occurs.

Collusive behaviour hurts the consumers, and collusive behaviour is illegal (MacKay and Weinstein (2022)). The collusion that we are observing in our simulations is on the form of tacit collusion. This mean that the algorithms are reaching collusive outcomes, without communicating. In general, collusion is difficult to detect, because the agents involved wants to keep it a secret. When speaking of tacit collusion it is even more difficult to discover and prove, as there is no explicit communication as evidence. However policy makers still have tools to combat this tacit collusion arising from algorithmic pricing. Asker et al. (2024) also dived into algorithmic pricing, and they differentiated between two different forms of Artificial Intelligence Algorithms, namely asynchronous and synchronous learning. The first is when the algorithms only learn from the paths and choices which they take, and is comparable to our setting. They found that algorithms in this setting leads to pricing close to monopoly level. However, they also investigated synchronous learning, where algorithms not only learn from the path they take, but also from the paths they dont take, and found that this type of learning leads to pricing at the competitive level. Therefore, an idea for a policymaker to combat the problem of collusive algorithmic pricing could be to demand users of algorithmic pricing, to use synchronous learning. However, the implementation of this could prove to be difficult.

Collusive behavior harms consumers and is illegal (MacKay and Weinstein (2022)). The collusive behavior observed in our simulations is tacit, meaning algorithms reach collusive outcomes without communication, which makes it particularly difficult to detect and prove. There might be tools for policymakers to address tacit collusion arising from algorithmic pricing. Asker et al. (2024) distinguishes between two types of artificial intelligence algorithms: asynchronous and synchronous learning. Asynchronous algorithms learn only from their own actions and paths they take, similar to our setting. This setting tend to produce prices near monopoly levels. Synchronous algorithms learn from both taken and untaken paths, resulting in prices at a competitive level (Asker et al. (2024)). Thus, policymakers could require the use of synchronous learning algorithms to combat collusive behavior arising from algorithmic pricing.

7 Conclusion

This project investigates whether algorithmic pricing via reinforcement learning (RL), specifically Q-learning, leads to collusive behavior in a dynamic Bertrand duopoly. The framework builds on Maskin and Tirole (1988) and Klein (2021), implementing a sequential pricing simulation where two Q-learners independently learn strategies to maximize their long-term profits in a turn-based setting.

The results confirm Klein's findings that Q-learners can sustain supra-competitive pricing, indicating tacit collusion. When comparing profits across different k values to a competitive benchmark based on the most competitive Edgeworth cycle, we observed consistent collusive profit levels. However, the degree of collusion declined as k increased, suggesting that coordination becomes more difficult as price intervals becomes smaller (larger k), likely due to increased coordination challenges and the moving target problem.

We analyzed the converged strategies underlying our results and found that Q-learners converged to focal pricing or cyclical patterns resembling Edgeworth cycles. As k increased, the share of simulations, which ended in focal pricing, declined significantly, while longer and more complex cycles became more frequent. Examining a specific cycle for $k = 6$, we observed that agents avoided short-term profitable deviations, indicating strategies aligned with long-term profit optimization. This behavior suggests that Q-learners refrained from deviating due to the threat of punishment, an indicator of collusion.

Overall, we find that algorithmic collusion is possible under specific market conditions. Even though the model is an idealized version of real market conditions, it serves as a proof of concept, which was empirically supported.

8 References

References

Albrecht, S. V., Christianos, F., and Schäfer, L. (2024). *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches.* MIT Press.

Asker, J., Fershtman, C., and Pakes, A. (2024). The impact of artificial intelligence design on pricing. *Journal of Economics & Management Strategy*, 33(2):276–304.

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

Calvano, E., Calzolari, G., Denicolò, V., and Pastorello, S. (2020). Artificial intelligence, algorithmic pricing, and collusion. *The American Economic Review*, 110(10):pp. 3267–3297.

de Bornier, J. M. (1992). The 'cournot-bertrand debate': A historical perspective. *History of Political Economy*, 24(3):623–656.

Fischer, J. L. (2023). Algorithmic pricing collusion using reinforcement learning. *Master's thesis, The University of Copenhagen.*

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

MacKay, A. and Weinstein, S. N. (2022). Dynamic pricing algorithms, consumer harm, and regulatory response. *Washington University Law Review*, 100:111.

Maskin, E. and 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*, page 32–33.

Noel, M. D. (2007). Edgeworth price cycles: Evidence from the toronto retail gasoline market. *The Journal of Industrial Economics*, 55(1):69–92.

Tadelis, S. (2013). *Game Theory: An Introduction.* Princeton University Press.

U.S. Energy Information Administration (2024). Gasoline price fluctuations. https://www.eia.gov/energyexplained/gasoline/price-fluctuations.php. Accessed: 2025-05-28.

Varian, H. R. (2019). *Intermediate Microeconomics with calculus - International Student Edition*. Norton.

9 Appendix

9.1 Q-learning collusion graphs

9.1.1 Definition 2 - Stochastic Game

Definition 2 (Stochastic game)(Albrecht et al., 2024, p. 47-48) A stochastic game consists of:

1. Finite set of agents $I = \{1, \ldots, n\}$
2. Finite set of states S, with a subset of terminal states $\bar{S} \subset S$
3. For each agent $i \in I$:
   - Finite set of actions A_i
   - Reward function $R_i : S \times A \times S \to \mathbb{R}$, where $A = A_1 \times \cdots \times A_n$
4. State transition probability function $T : S \times A \times S \to [0, 1]$ such that

$$\forall s \in S, a \in A : \sum_{s' \in S} T(s, a, s') = 1$$

5. Initial state distribution $\mu : S \to [0, 1]$ such that

$$\sum_{s \in S} \mu(s) = 1 \quad \text{and} \quad \forall s \in \bar{S} : \mu(s) = 0$$

9.2 Q learver profit vs k

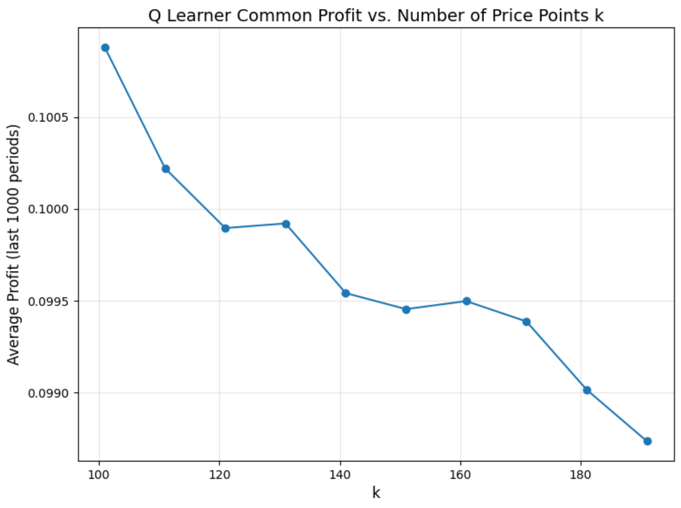

Figure 12: The average common profit under equal sharing as a result of k, for k values 100-200. Profit is calculated in the last 1000 steps of a 500.000 timestep simulation. Each value of k ran for 1000 simulations.

9.2.1 k = 12

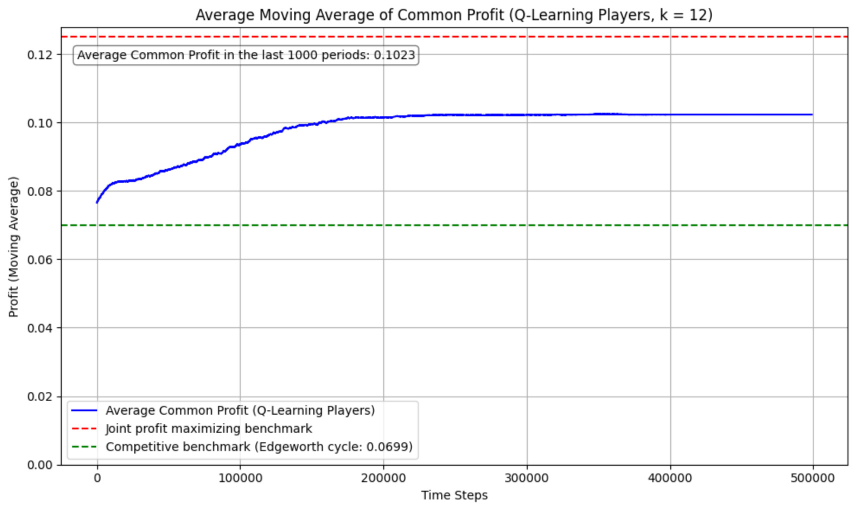

Figure 13: Q-learner vs Q-learner when k = 12

9.2.2 k = 24

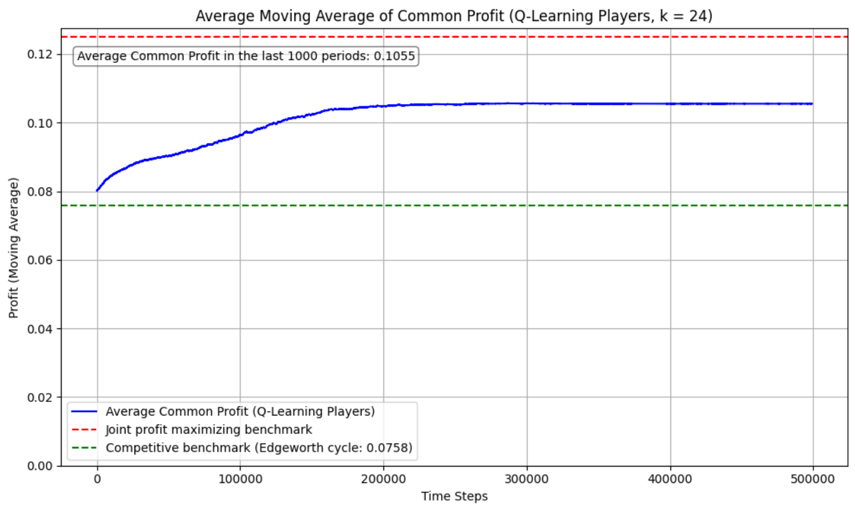

Figure 14: Q-learner vs Q-learner when k = 24

9.2.3 k = 48

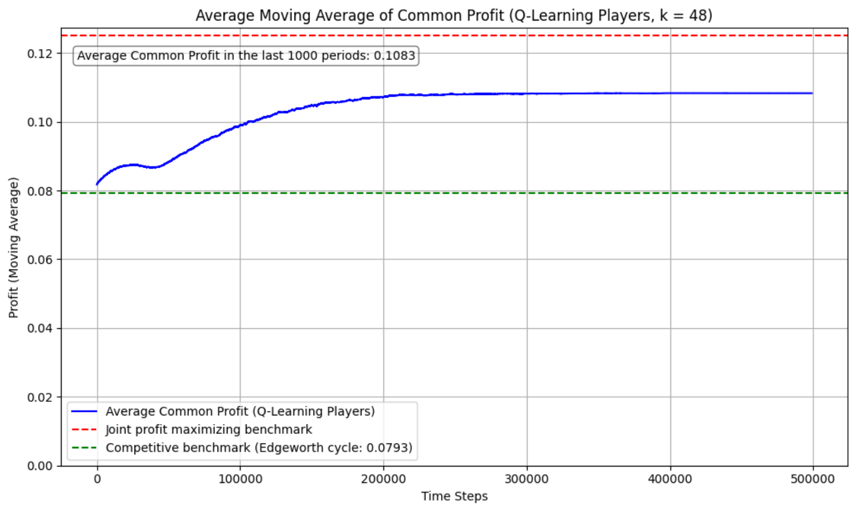

Figure 15: Q-learner vs Q-learner when k = 48

9.2.4 Q Matrix - cycle length 4 - k=6

Q	$\overbrace{s_1}^{p=0}$	$\overbrace{s_2}^{p=0.167}$	$\overbrace{s_3}^{p=0.333}$	$\overbrace{s_4}^{p=0.5}$	$\overbrace{s_5}^{p=0.667}$	$\overbrace{s_6}^{p=0.833}$	$\overbrace{s_7}^{p=1}$
a_1	2.361, 1.054	2.398, 1.014	2.289, 1.078	1.323, 0.999	2.132, 1.662	1.365, 1.503	1.763, 1.645
a_2	1.212, 1.061	1.197, 1.058	2.676, 1.142	1.308, 1.153	2.013, 1.564	2.051, 1.596	1.986, 1.525
a_3	1.165, 1.074	1.191, 1.010	2.275, 1.113	2.442, 1.198	1.706, 1.621	2.125, 1.599	1.618, 1.627
a_4	1.098, 1.061	1.185, 1.037	2.227, 1.101	1.331, 1.149	2.665, 1.673	2.665, 1.568	2.665, 1.627
a_5	1.483, 1.078	1.195, 1.062	2.404, 1.112	1.319, 1.110	2.098, 1.583	1.991, 1.609	1.874, 1.691
a_6	1.213, 1.093	1.194, 1.081	2.235, 1.117	1.327, 1.087	1.947, 1.726	2.255, 1.606	1.912, 1.621
a_7	1.212, 1.076	1.166, 1.052	2.202, 1.119	1.320, 1.103	1.844, 1.609	2.105, 1.483	1.868, 1.603

Matrix 1: Final Merged Q-matrices ($A \times S$) for firm i and j, from the K=6 run, shown in table 3. Each entry in the matrix is the tuple of Q-values from firm i and j ((1,1) = $((Q_i(a_1, s_1), Q_j(a_1, s_1)) = (2.361, 1.054))$. The Blue corresponds with firm i's maximizing action in that given state (competitor price) and red the same for firm j. It can be read as follows. If the state was s_5, (a competitor price of 0.667 (column 5)), firm i responds with action a_4 (setting a price of 0.5), since 2.665 is the highest Q-value for that given state/column. Given the same state firm j would respond with a_6 (setting a price of 0.833). This is when their actions are solely based on their learned Q-matrix, and thereby maximizing the Q-value given a state, which is how it would be at the end of the simulations as exploration.

Deklaration for anvendelse af generative AI-værktøjer (studerende)

På Københavns Universitet udfører vi vores arbejde med **ansvarsfølelse** og **respekt for samfund, kulturarv, miljø og mennesker** omkring os.
Integritet, ærlighed og **transparens** er forudsætninger for akademisk arbejde. Vi forventer derfor, at eksamenspræstationer afspejler **den studerendes egen læring og selvstændige indsats**.
Akademisk arbejde baserer sig altid på andres indsigter, viden og bidrag, men altid med **grundig anerkendelse, respekt** og **kreditering** af dette arbejde.
Dette gælder også ved brug af generativ kunstig intelligens.

Vejledning

Brug af generativ AI ved eksamen

I henhold til KU's regler for brugen af værktøjer, der er baseret på generativ AI (GAI), skal du være transparent om din anvendelse af teknologien, fx i dit metodeafsnit og/eller ved at udfylde og vedlægge nedenstående deklarationsskabelon som bilag til skriftlige opgavebesvarelser.
Når du skriver din deklaration, er det vigtigt, at læseren får et tydeligt billede af, om og hvordan generativ AI har bidraget til det endelige produkt.

Hvis det er besluttet, at du skal bruge skabelonen i dit fag, skal du også benytte den, når du *ikke* har anvendt GAI-værktøjer som hjælpemiddel. I dette tilfælde skal du dog blot krydse af, at du ikke har brugt GAI, og behøver ikke at udfylde resten.
Ved at deklarere din brug af GAI-værktøjer sikrer du, at der ikke opstår udfordringer i forhold til reglerne om eksamenssnyd.

I kurser, hvor brugen af GAI er integreret i fagligheden, kan refleksion og kritisk vurdering af anvendelsen af GAI-værktøjer også indgå som en del af et metodeafsnit i din opgave. Spørg din underviser eller vejleder, hvis du er tvivl, om det er tilfældet i dit kursus.

Hvis generativ AI er objekt for din undersøgelse, vil det fremgå af dine forskningsspørgsmål, din metodebeskrivelse, din analyse og konklusion, hvilken rolle GAI spiller i din opgave. Hvis du samtidig også bruger GAI som hjælpemiddel i processen, skal du deklarere denne anvendelse særskilt.

Opmærksomhedspunkter:

- Hvis GAI er et tilladt hjælpemiddel på dit kursus, må du anvende GAI til dialog og sparring under udarbejdelsen af din opgave, men du må **ikke** overlade udfærdigelsen af din opgavebesvarelse til GAI-værktøjer, selvom alle hjælpemidler er tilladt.
- Hvis materiale fra GAI inkluderes som kilde (direkte eller i redigeret form) i din besvarelse, gælder de samme krav om brug af citationstegn og kildehenvisning som ved alle andre kilder, da der ellers vil være tale om plagiat.
- Brug aldrig personhenførbare, ophavsretsbeskyttede eller fortrolige data i et AI-værktøj.
- Husk altid at undersøge gældende regler og retningslinjer for brug af generativ AI på KU.
- Læs kursusbeskrivelsen grundigt. Det er vigtigt at du ved, hvilke anvendelser der er tilladt i dit kursus. Der kan eventuelt være yderligere krav om dokumentation, fx at du skal beskrive dine centrale prompts og evt. kildemateriale (hvad har du givet af kontekst, hvad har du fodret værktøjet med, hvad har du bedt værktøjet om at gøre), beskrive outputtet (hvilke svar du fik af værktøjet), beskrive processen, f.eks. historik og iterationer (hvis du har skrevet frem og tilbage med værktøjet ad flere omgange for at komme frem til et brugbart svar).
- Tal med din underviser eller vejleder, hvis du er i tvivl.

Deklaration for anvendelse af generative AI-værktøjer (studerende)

☒ **Jeg/vi har benyttet generativ AI som hjælpemiddel/værktøj** *(sæt kryds)*

☐ **Jeg/vi har <u>IKKE</u> benyttet generativ AI som hjælpemiddel/værktøj** *(sæt kryds)*

Hvis brug af generativ AI er tilladt til eksamen, men du ikke har benyttet det i din opgave, skal du blot krydse af, at du ikke har brugt GAI, og behøver ikke at udfylde resten.

Oplist, hvilke GAI-værktøjer der er benyttet, inkl. link til platformen (hvis muligt):

OpenAI ChatGPT: https://chatgpt.com
GitHub Copilot: https://github.com/features/copilot

Beskriv hvordan generativ AI er anvendt i opgaven:

1) *Formål (hvad har du/I brugt værktøjet til)*
2) *Arbejdsfase (hvornår i arbejdsprocessen har du/I brugt GAI)*
3) *Hvad gjorde du/I med outputtet (herunder også, om du/I har redigeret outputtet og arbejdet videre med det)*
   ChatGPT er blevet brugt som sparringspartner i ideudviklingen og til at forstå stoffet. Derudover er det blevet brugt til at komme med ideer til hvordan vi kunne forbedre selve projektet skrivemæssigt (fx haft samtaler frem og tilbage om enkelte afsnit skulle rykkes rundt for bedre flow i teksten)
   GitHub Copilot er blevet anvendt som sparringspartner i udviklingen af kodningsdelen af projektet. Det er gjort således at den nogle gange er kommet med ideer til hvordan kode skulle udføres, og de ideer har vi så arbejdet videre med og nogle gange implementeret, og andre gange ikke.
   Vi har anvendt GAI i ideudviklingen og indsnævring af emnet, i udviklingen af koden som sparringspartner og i rettelsen af projektet hvor GAI er kommet med forslag til afsnit som kunne rykkes rundt for bedre flow i teksten.
   Vi har fået ideer i output fra GAI, som vi så har arbejdet videre med og nogle gange implementeret, og andre gange ikke.
   Outputtet fra GAI er ikke blevet anvendt ukritisk. I stedet har vi brugt det som inspiration og sparringspartner. Vi har altid forholdt os kritisk til alt output, og kun arbejdet videre med nøje udvalgte ideer fra GAI.

NB. GAI-genereret indhold brugt som kilde i opgaven kræver korrekt brug af citationstegn og kildehenvisning. <u>Læs retningslinjer fra Københavns Universitetsbibliotek på KUnet</u>.