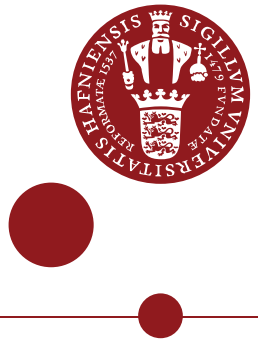
UNIVERSITY OF COPENHAGEN



BSc in Computer Science and Economics

Bachelor's Thesis

Algorithmic Pricing Collusion: Introducing Multi-Agent Reinforcement Learning

Nicklas Busk Jensen (vhr863) & Mikkel Foss Engelsted (hrx712)

Supervised by Anders Munk-Nielsen

June 2024

Nicklas Busk Jensen (vhr863) & Mikkel Foss Engelsted (hrx712)

Bachelor's Thesis

BSc in Computer Science and Economics, Tuesday 17th December, 2024

Supervisor: Anders Munk-Nielsen

Keystrokes: Approximately 74.000

University of Copenhagen

Faculty of Science

Bachelor's Degree in Computer Science and Economics

Universitetsparken 1

2200 Copenhagen N

Abstract

This thesis explores the phenomenon of algorithmic pricing collusion through the application of Multi-Agent Reinforcement Learning (MARL). The study focuses on a sequential Bertrand duopoly setting, where firms repeatedly adjust their prices in an attempt to maximize profits. The research investigates the collusive behavior of reinforcement learning algorithms; specifically Q-learning, WoLF-PHC, and JAL-AM, under conditions of complete and asymmetric information. Through extensive simulations, it is demonstrated that all three algorithms exhibit collusive tendencies, achieving supra-competitive profits. The study finds that JAL-AM outperforms the other algorithms in profitability, even in the presence of asymmetric information where one firm has incomplete information about the competitor's prices. These results highlight the robustness of MARL algorithms in adapting to less-than-ideal informational settings. They raise important considerations for policymakers and regulators concerning the potential for tacit collusion in algorithm-driven markets.

Contents

1 Introduction

Algorithmic pricing, a growing field at the intersection of economics and computer science, is transforming the landscape of market dynamics. Advanced machine learning techniques, particularly reinforcement learning, enable companies to employ sophisticated algorithms to optimize pricing strategies. While this shift towards data-driven decision-making has led to significant efficiency gains, it has also raised concerns about potential anti-competitive behaviors, such as collusion. The literature on this subject is limited but increasingly relevant as artificial intelligence becomes more accessible. Concerns about collusion have been examined in several scientific papers, including Calvano *et al.* (2020), Klein (2021), Fischer (2023), and others.

In this thesis, we explore the phenomenon of algorithmic pricing collusion, focusing on multi-agent reinforcement learning (MARL) and the effect of asymmetric information, by basing our study on the work of Klein (2021) and Fischer (2023). Most existing research in this area has implemented Q-learning; thus, we investigate alternative reinforcement learning algorithms, including WoLF-PHC and JAL-AM. Additionally, we consider asymmetric information to better resemble real-world settings, specifically when one firm is unable to observe the true price of its competitor. For instance, an algorithm collecting prices from websites might encounter inaccuracies, such as those found on Pricerunner.

We conduct simulations using a single-agent reinforcement learner and two multi-agent learners, comparing their behaviors under conditions of asymmetric information. Consistent with Fischer (2023), we find that Q-learning and WoLF-PHC algorithms exhibit collusive behavior, achieving profits well above those expected in competitive market dynamics. Additionally, we introduce the JAL-AM algorithm and find it achieves higher profitability than Q-learning and WoLF-PHC.

Our study also examines the dynamics of these algorithms under asymmetric information, where one firm has incomplete information when observing the competitor's price in the moment of action. We find that incomplete information negatively impacts average profitability, yet reinforcement learners can still maintain a collusive equilib-

rium. We delve into the dynamics of focal pricing equilibrium when one firm deviates, either above or below the equilibrium price.

These findings are concerning as tacit collusion, which is difficult to detect, remains possible even when true prices are harder to observe. This should alert policymakers and regulators to maintain fair market dynamics.

The algorithms used in this paper are known to be computationally intensive. To address this, we implemented the Numba library and parallelized our simulations using Python's `concurrent.futures` module, significantly reducing running time and expediting our results.

Following this introductory section, the next section reviews the related literature. Section 3 details our economic model and the relevant economic terms, introduces reinforcement learning, and describes the algorithms and metrics used. Section 4 summarizes the implementation and optimization of our simulations. Section 5 presents the results of our simulations. Section 6 discusses our findings and their implications, and Section 7 concludes the paper*.

*The associated code for this thesis is available at GitHub.
The use of ChatGPT and other Large Language Models have been used to correct grammatical errors and improve wording.

2 Related Literature

The study of algorithmic pricing and its implications on market dynamics has gained significant attention in recent years, although literature in this field remains relatively limited. As data-driven decision-making becomes more accessible, concerns about tacit collusion are increasing. Most studies, including ours, rely on simulated scenarios due to the challenges of determining and analyzing algorithmic pricing in real-world settings. The main differences in these studies lie in how they model the environment.

The first study we examined was by Klein (2021), which built upon the work of Calvano *et al.* (2020). Calvano et al. were among the first to explore algorithmic pricing using reinforcement learning. They provided experimental evidence that simple reinforcement learning algorithms could learn to collude without explicit communication, raising concerns about the regulatory implications of algorithm-driven markets. Calvano et al.'s work served as a proof-of-concept for tacit collusion, using Q-learning in an infinitely repeated simultaneous price-setting environment. Klein (2021) extended this research by investigating Q-learning in a sequential game, finding similar collusive behavior when two Q-learners interacted.

Subsequently, we reviewed the work of Fischer (2023), which was based on Klein (2021) but introduced Multi-agent Reinforcement Learning (MARL) to compare its performance against Q-learning in the same economic setting. Fischer found that MARL agents more frequently colluded on supra-competitive prices, closer to the joint-profit maximizing level.

An addition to the literature in this field is the study conducted by Dou *et al.* (2024), who explore the dynamics of tacit collusion under conditions of information asymmetry. Their work highlights the challenges posed by imperfect monitoring, where standard grim-trigger strategies are less effective due to difficulties in observing and accurately monitoring each other's actions. They found that tacit collusion can still be sustained through price-trigger strategies, as informed speculators can infer total order flows from market prices, thus maintaining collusive incentives. This insight is relevant for

our study as it underscores the resilience of collusive behavior even when information is incomplete.

Another contribution to this field is the paper by Lepore (2021), who introduced more advanced reinforcement learning algorithms such as Deep Q-Networks and Asynchronous Advantage Actor Critic (A3C). Lepore also explored methods to disrupt collusion, finding that it was possible but required significant market intervention.

In our further research, we encountered empirical studies demonstrating algorithmic pricing in practice. One such study by Wieting and Sapi (2021) investigated algorithmic pricing in the largest online marketplace in the Netherlands and Belgium. They found that competition between two sellers led to higher prices, but as the number of sellers increased, competition intensified. Another empirical study by Assad *et al.* (2024) examined the gasoline market in Germany. They provided evidence that algorithmic pricing used by German gas stations led to higher prices for consumers. This was the first real-world evidence of its kind, emphasizing the need for awareness and regulation by competition authorities.

3 Theory

This section presents the theoretical foundation of the project, covering the economic model and reinforcement learning theory, along with the settings for our simulations.

3.1 Economic Model

We adopt the economic model proposed by Maskin and Tirole (1988). This model represents a dynamic Bertrand duopoly competition, wherein the firms make sequential moves.

3.1.1 Dynamic competition

Dynamic competition refers to games, where firms take actions sequentially or repeatedly. In our model, firms will move sequentially. As a baseline assumption, we presume that all firms have knowledge of each other's prices, mirroring typical pricing strategies employed by companies.

3.1.2 Nash Equilibrium

In a game theoretic setting it's crucial to define the equilibria encountered in dynamic game. A Nash equilibrium (NE) occurs when each firm's chosen strategy is the optimal response to the strategies chosen by all other players. In simpler terms, no firm has an incentive to deviate from their strategy given the strategies of others. Formally, the Nash equilibrium is defined as follows (Tadelis, 2013):

Definition 1 (Nash equilibrium) *The pure-strategy profile* $s^* = (s_1^*, s_2^*, ..., s_{-i}^*)$ *for all* $s_i' \in S$ *is a Nash equilibrium if* s_i^* *is a best response to* s_{-i}^**, for all* $i \in N$.

Considering our repeated sequential game structure, where firms make decisions sequentially in multiple rounds, it's essential to introduce a more refined equilibrium concept known as the subgame perfect Nash equilibrium (SPNE). The SPNE arises due to the representation of each round as an extensive-form game, wherein firms factor in past actions when making decisions. Therefore, strategies must persist as a NE in every subgame within this framework. This concept is formally defined by Tadelis (2013):

Definition 2 (SPNE) *Let Γ be an n-player extensive-form game. A behavioral strategy profile $\sigma^* = \sigma_1^*, \sigma_2^*, ..., \sigma_n^*$ is a subgame-perfect (Nash) equilibrium if for every proper subgame G of Γ the restriction of σ^* to G is a Nash equilibrium in G.*

Additionally, we define a Markov Perfect Equilibrium (MPE), which serves as a refinement of the SPNE. Strategies in MPE adhere to the Markov assumption, meaning they depend solely on payoff-relevant variables (Maskin and Tirole, 1988; Klein, 2021).

3.1.3 Bertrand Competition

We use the infinitely repeated sequential move pricing duopoly presented by Maskin and Tirole (1988), as the environment for our firms. In this model, firm 1 adjusts its price when the time *t* is odd, while firm 2 adjusts prices when *t* is even - thus firms will have the same price for 2 periods. All prices are discrete and evenly spaced between 0 and 1, denoted as $p \in \mathcal{P} = \{0, \frac{1}{k}, \frac{2}{k}, ..., 1\}$, where $\mathcal{P}$ is all possible prices or *actions*. The firms sell perfectly homogeneous goods, where the firm with a lower price will take the entire market. In cases where the firms set the same price they split the market evenly. The demand function for firm *i* is defined as follows:

$$D_i(p_{it}, p_{jt}) = \begin{cases} 1 - p_{it} & \text{if} \quad p_{it} < p_{pjt} \\ \frac{1}{2}(1 - p_{it}) & \text{if} \quad p_{it} = p_{pjt} \\ 0 & \text{if} \quad p_{it} > p_{pjt} \end{cases} \tag{3.1}$$

With the demand function, $D_i(p_{it}, p_{jt})$, we assume the firms has no marginal costs; hence, the profit for firm *i* is given by:

$$\omega_i(p_{it}, p_{jt}) = p_{it} \cdot D_i(p_{it}, p_{jt}) \tag{3.2}$$

As firms compete in the infinitely repeated sequential move pricing duopoly, firms must weigh current profits against future profits. Thus, the firms' objective is to maximise future discount profits with discount factor $\gamma \in [0, 1)$ at time t:

$$max \quad \sum_{s=0}^{\infty} \gamma^s \cdot \omega_i(p_{i,t+s}, p_{j,t+s}) \tag{3.3}$$

In the simultaneous Bertrand game the Nash equilibrium is the marginal cost. To prove that the marginal cost is a Nash equilibrium consider the following: if both firms set prices higher than marginal cost, each would in turn benefit from undercutting the opponent until both reaches marginal cost. Once they reach marginal cost, it is no longer beneficial for either of the firms to undercut, as they instead would incur losses rather than zero profit. Hence, neither of the firms has any incentive to deviate from the strategy, resulting in marginal cost being a Nash equilibrium.

However, our setting uses sequential pricing, where the static equilibrium fails to hold. As Klein (2021) points out, an equilibrium in a sequential pricing setting is found when both firms follow the value function:

$$V_i(p_{jt}) = max_p[\, \omega_i(p, p_{jt} + E_{pj,t+1}[\gamma \cdot \omega_i(p, p_{j,t+1}) + \gamma^2 \cdot V_i(p_{j,t+1})]] \tag{3.4}$$

This equilibrium is categorised as an MPE if the value function also holds for off-equilibrium prices.

3.1.4 Asymmetric information

Asymmetric information occurs when firms possess different information while competing. This means that one or more firms have information that their competitors do not. This information disparity can lead to inefficient markets, as the less informed firm may set prices too high or too low, or misallocate resources. In the context of algorithmic price setting, it is interesting to investigate what happens when an algorithm receives incorrect pricing information. For example, an algorithm scraping prices from the web might encounter inaccuracies (e.g., on Pricerunner). In such a scenario, the algorithm might set an incorrect price based on the observed data. Subsequently, the firm could manually verify the true state or rely on a slower but more reliable source, and the algorithm would learn from this.

In our duopoly case, Firm 1 has complete information and can access all prices set by its competitor. However, Firm 2 operates under incomplete information and observes the state S_t, which is a stochastic variable: with probability μ, $S_t \sim \mathcal{U}(\mathcal{P})$, i.e. it is uniformly distributed on the price space $\mathcal{P} \subset \mathbb{R}$ and with the remaining probability, $1-\mu$, $S_t = p_{jt}$, the true price set by Firm 2. Thus we can express S_t as a function of μ:

$$S_t(\mu) = \begin{cases} p_{j,t} & \text{with probability} \quad 1-\mu \\ U_t & \text{with probability} \quad \mu \end{cases} \tag{3.5}$$

where $U_t \sim \mathcal{U}(\mathcal{P})$. Since Firm 1 does not know whether the true price is observed, it must maximise its expected profits conditional on what it observes. If we assume that $\gamma = 0$, or equivalently that the firm faces a static one-shot problem. Then the firms problem becomes:

$$\max_{p_{j,t}} \mathbb{E}_{p_{j,t}} \left[\omega(p_{i,t}, p_{j,t}) | S_t(\mu) = s \right] \tag{3.6}$$

Solving this problem involves determining the distribution of p_{jt} given $S_t = s$. This is because the expectation we are calculating is with respect to p_{jt}, which is the only known variable. When we know that $S_t = s$, it provides some information about p_{jt}, but it does not fully determine it. This uncertainty arises because S_t could be either p_{jt} or U_t. We can tell that the expectation in some realization of $S_t = s \in \mathcal{P}$, can be written as:

$$(1-\mu) \cdot \omega(p_{i,t}, s) + \mu \cdot \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \omega(p_{i,t}, p) \tag{3.7}$$

This equation intuitively states the expected profit for Firm 2, that is under incomplete information. The first term represents the expected profit when the true price p_{jt} is observed with probability $1-\mu$. So, with probability $1-\mu$, the firm observes the true price p_{jt} and calculates its profits based on the true price. The second term represents the expected profit when the firm observes a false price with probability μ. Here, $\frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \omega(p_{i,t}, p)$ calculates the average profit across all possible prices in the price space $\mathcal{P}$. Since the false price can be any price in $\mathcal{P}$ with equal probability, we take the average of profits at each possible price. Then, μ scales this average profit by the probability of observing a false price.

The framework for the algorithmic implementation of asymmetric information can be seen in Appendix A.

3.1.5 Monopoly

In section 3.1.3, we defined the demand in the Bertrand competition. In the Bertrand setting with a single firm, the demand is defined as $1 - p$, as there is only one firm in the market. Similarly, in a market with multiple firms engaging in full collusion, they effectively act as a single firm, resulting in the same demand function. Consequently, we can determine the monopolistic price, which maximises profit.

$$\omega = p \cdot (1 - p) \Leftrightarrow \omega = p - p^2 \tag{3.8}$$

To find the monopolistic price, we maximise profit by solving the first-order condition with respect to price:

$$\frac{\partial\, \omega}{\partial\, p} = 0 : \; 1 - 2p = 0 \Leftrightarrow p = \frac{1}{2} \tag{3.9}$$

Hence, the optimal price for the monopolist in the Bertrand setting is $p = \frac{1}{2}$. If the market contains one firm, the firm takes all profits, whereas if the market contains two firms engaging in full collusion, they share the profit equally. Specifically this is the joint-profit maximising price.

3.1.6 Collusion

Collusion is a phenomena occuring when firms actively and explicitly agree to coordinate on a joint price in order to increase their profits. In the Bertrand setting, when the firms do not collude, their profit is zero as the equilibrium price equals marginal cost. Firms thus, find it advantageous to set prices higher than marginal cost, creating an incentive to collude and maximise profits. However, the success of a collusion strategy depends on firms adhering to the agreed-upon price, as deviation by either firm results in profit gain for the deviator and loss for the non-deviator. Collusion leads to welfare loss for consumers, as firms choose to set high, supra-competitive prices, a practice deemed problematic and illegal under European law (EU, 2024).

Moreover, collusion often involves reward-punishment strategies. Firms are rewarded for adhering to the collusive price, gaining higher profits in the long run. Punishment is meted out to firms deviating from the collusive price to gain immediate rewards, but such deviation can result in profit loss in the long run. This loss occurs because the non-deviating firm purposely lowers its price to punish the deviating agent. The decision to refrain from deviating from a collusive price becomes beneficial when

agents sufficiently discount their future, indicating a high degree of future-oriented behavior (Tadelis, 2013).

3.1.6.1 Tacit Collusion

Due to the illegality of explicit collusion, firms engaging in such behavior seek to keep it off the record. To deter collusion, competition authorities have been enacted to maintain competitive and well-functioning markets (DCCA, 2024). Tacit collusion refers to a pricing scheme where competitors indirectly align their pricing strategies without explicit communication or formal agreements. Unlike explicit collusion, tacit collusion is typically considered legal, as no explicit agreement has been made. Consequently, competition authorities face a greater challenge in proving collusive behavior when it is tacit. With the emergence of algorithmic pricing, competition authorities has become more aware of tacit collusion as research have found that algorithms tend to set collusive prices (Klein, 2021; Maskin and Tirole, 1988; Calvano *et al.*, 2020).

When the discount factor γ is high, Klein (2021) and Maskin and Tirole (1988) identify two collusive equilibrium price strategies; focal pricing and Edgeworth price cycles. Focal pricing is when firms choose to maintain the same price. This strategy is based on the belief that if one firm were to lower its price, it would trigger a cycle of undercutting, leading to decreased profits for both. Similarly, if a firm were to raise its price, it expects the opponent to maintain the focal price and gain market control. Edgeworth price cycles occur when firms continuously undercut each other to gain market control and increase profits until undercutting becomes too costly. At this stage, one firm raises its price, prompting the other to follow suit and restart the cycle of undercutting. The equilibria found in Edgeworth price cycles result in supra-competitive prices.

3.2 Reinforcement Learning Theory

In this section, we introduce the basics of reinforcement learning, and some of the underlying theories, we use in this paper. This section is guided by the work of theory based on Sutton and Barto (2015).

3.2.1 Introducing reinforcement learning

Reinforcement learning is one out of three branches of machine learning, focused on teaching "agents", previously referred to as firms, to learn from their interactions. It is distinguished from the other types of machine learning by its use of the "trial and error" approach, where an agent learns the optimal decisions by trying different actions and observing the outcome.

The fundamentals of reinforcements learning includes several key concepts. The *agent* takes actions in an *environment*, which can be in different states. *Policy* is a strategy that guides the agent to take actions based on the current state of the environment. *Reward signal* is the feedback an agent receives, which measures the success of an action taken by the agent. The agent's goal is to maximise this reward over time. *Value function* is a prediction of the expected future rewards, used to evaluate which states are the most beneficial for the agent in the long run.

From these concepts, the agent takes an action given the state of the environment. This action is influenced by the immediate rewards and the anticipated future rewards. This introduces a dilemma between exploiting known rewards and exploring new possibilities to find more valuable actions.

The process of making a decision within an environment can be formally modeled using Markov Decision Processes (MDP), which provide a mathematical framework for reinforcement learning. MDP's can aid in designing of algorithms that efficiently learn, through experience, which actions gives the highest rewards.

When examining a reinforcement learning algorithm we can divide it into two components: the learning module and the action-selection module. *The learning component* contains the learning aspect of the algorithm, as it improves its policy by learning from past actions and rewards. *The action component* is where the agent will decide its next action, while considering whether to exploit, using its current knowledge or to explore and potentially discover greater future rewards. From these components we can better appreciate how reinforcement learning actually simulate the natural learning process.

3.2.2 Multi agent environment setting

Reinforcement learning can be divided into two different settings: single-agent and multi-agent setting. Both settings poses unique challenges and is suited for different types of problems. In a single-agent setting the agent learns the best policy to maximise the cumulative reward, given that the environment is static, meaning that the agent does not adapt to a changing environment. Unlike single-agent learning setting, the MARL setting, includes multiple agents, that can adapt and change strategies in response to the other agents actions, which could lead to complex dynamics such as cooperation, competition or a mixture of both.

When introducing a multi-agent setting it could lead to problems, such as convergence. If two single-agent RL algorithms plays against each-other, they could potentially try and adapt to each other infinitely, as they can have trouble converging to a stable policy.

3.3 Reinforcement Learning Algorithms

This section will introduce the specific algorithms we have chosen to use in our research, which is Q-learning, JAL-AM and WoLF-PHC. Each of the following subsections will include a brief classification of the specific algorithm introduced, along with the motivation for choosing the algorithm to solve our problem.

3.3.1 Q-Learning

The first algorithm we introduce is Q-learning, a versatile tool used in areas like pricing strategies and game designing to make optimal decisions. Q-learning was first introduced by Chris Watkins in 1989 (Sutton and Barto, 2015) to solve Markov Decision Process and is traditionally a single-agent RL algorithm. However it has been shown to work in a multi-agent setting (Klein (2021), Calvano *et al.* (2020), and more). In the multi-agent setting, Q-learning enables multiple agents to learn and adapt their strategies cooperatively or competitively, making it a flexible tool in fields ranging from robotics to economic modeling.

Q-learning is value-based, referring to the use of the value-function, typically referred to as the "Q-function", as part of the learning process. The policy Q-learning follows is implicit and can be derived directly from the Q-function, helping the algorithm to decide on the best action to take. In contrast, policy-based methods explicitly maintain and update a policy, which is a mapping from states to actions. These methods optimize the policy directly rather than deriving it from a value function.

The Q-function updates the Q-table, which is a matrix, organized into rows and columns to represent all states, S and all actions A. Hence the Q-table is a $|S|x|A|$ matrix. The states are all the prices the opponent agent can choose, while actions are all the prices the agent can choose. Consider the Q-table as a roadmap indicating the most optimal price to set in response to the opponent agent's price, informed by previous rounds played. As the Q-table is a matrix, the S and A, need to be discrete. As we assume the agents are able to set the same prices, we will refer to the action/state-space as P, and denote an element of the Q-table as $Q_i(p_{jt}, p_{it})$. An agent only updates its Q-function when it is their turn to set a price. The new value is a combination of the current value mixed with the reward received, adjusted by how important immediate rewards are compared to future rewards. The Q-table is updated for agent i as in Klein (2021) and is formulated as:

$$Q_i(p_{j,t}, p_{i,t}) \leftarrow (1-\alpha)Q_i(p_{j,t}, p_{i,t}) + \alpha \left(\omega(p_{j,t}, p_{i,t}) + \gamma \cdot \omega(p_{j,t+1}, p_{i,t}) + \gamma^2 \max_{\alpha'} Q_i(p_{j,t+1}, p') \right) \tag{3.10}$$

where $\alpha \in (0, 1)$ is the learning rate, which determines how fast the q-learner learns from new experiences. ω is the profit function, and $\gamma \in [0, 1)$ discounts the value of future rewards, controlling the allurement of immediate rewards.

As for our *action module* we have chosen to use the $\epsilon - greedy$ decision strategy to find the optimal policy for our q-learners, as Klein (2021). This strategy tries to balance exploration and exploitation by exploring widely at the start, but over time, it focuses more on prices that have worked well in the past.

$$p_{i,t} \begin{cases} = \arg\max_{p_i'} Q(p_{j,t-1}, p_i') & \text{with probability } 1 - \epsilon_t \\ \sim U(P) & \text{with probability } \epsilon_t \end{cases} \tag{3.11}$$

where $U(\mathcal{P})$ is uniform distribution over the action space and ϵ_t is a number between 0 and 1, $\lim_{t\to\infty} \epsilon_t = 0$.

To visualize the workflow of a sequential q-learner, we provide the pseudo-code in Algorithm 1. This shows a step by step guide of how we have implemented the algorithm for our environment.

Algorithm 1 Sequential Q-learning

1: **Initialize:**
Learning parameter and discount factors α, γ
Choose random prices in $\mathcal{P}$ for both players for $t = \{1, 2\}$
Q-tables
2: **for** t=3 to $\mathcal{T}$ **do**
3: Update $Q_i(p_{j,t-2}, p_{i,t-1})$:

$$Q_i(p_{j,t-2}, p_{i,t-1}) \leftarrow (1-\alpha) \cdot Q_i(p_{j,t-2}, p_{i,t-1}) +$$
$$\alpha \left(\omega(p_{j,t-2}, p_{i,t-1}) + \gamma \omega(p_{j,t-1}, p_{i,t-1}) + \gamma^2 \max_{p_i'} Q_i(p_{j,t-1}, p_i') \right)$$

4: Set p_{it} according to

$$p_{i,t} \begin{cases} = \arg\max_{p_i'} Q_i(p_{j,t-1}, p_i') & \text{with probability} \quad 1-\epsilon_t \\ \sim \mathcal{U}(P) & \text{with probability} \quad \epsilon_t \end{cases}$$

and $p_{j,t} = p_{j,t-1}$
5: change agents, $j \leftarrow i$ and $i \leftarrow j$
6: **end for**

3.3.2 JAL-AM

In Albrecht, Christianos and Schäfer (2024), "*Multi-agent Reinforcement Learning*", in chapter 6.3.2, they introduce an algorithm which they call Joint-action Learning with Agent Modeling (JAL-AM). Our exposition is an alteration of their chapter 6.3.2. The core idea of JAL-AM is inspired by the fictitious play concept from game theory, where agents iteratively update their strategies based on the observed actions of others. JAL-AM is an algorithm designed for learning optimal policies in stochastic games allowing agents to handle dynamic and uncertain environments. JAL-AM combines Q-values with agent models and best-response strategies to select actions and update Q-values. Each agent learns an empirical distribution of the actions taken by other agents in each state and uses this distribution to predict future actions and update its own strategy.

The agent model describes the subjective beliefs using the notation $\hat{\sigma}_{j,t}(p_{j,t}|p_{i,t})$ as the probability that agent j selects action $p_{j,t}$ in response to the state $p_{i,t}$. This notation captures the belief of agent i regarding the price $p_{j,t}$ that agent j will set, conditional

on the price $p_{i,t}$ that agent i has set. Let $C(p_{i,t}, p_{j,t})$ denote the number of times agent j selected action $p_{j,t}$ in state $p_{i,t}$, the agent model $\hat{\sigma}_{j,t}$ to the time, t, is defined as

$$\hat{\sigma}_{j,t}(p_{j,t}|p_{i,t}) = \frac{C(p_{i,t}, p_{j,t})}{\sum_{p'_j} C(p_{i,t}, p'_j)} \tag{3.12}$$

If the true strategy σ_j is fixed or converges, then at some point the agent model $\hat{\sigma}_{j,t}$ should converge towards the true strategy. JAL-AM uses the learned agent models to compute action values, $AV_i(p_{j,t}, p_{i,t})$ to update Q-values and set prices. The action value is calculated as

$$AV_i(p_{j,t}, p_{i,t}) = Q_{i,t}(p_{j,t}, p_{i,t}) \cdot \hat{\sigma}_{j,t}(p_{j,t}|p_{i,t}) \tag{3.13}$$

Here, it is understood that the product is the subjective belief about the probability that the action $p_{j,t}$ is played, when $p_{i,t}$ is played. The agent model is used when choosing an action through the ϵ-greedy strategy and is given as:

$$p_{i,t} \begin{cases} = \arg\max_{p'_i} AV_i(p_{j,t-1}, p'_i) & \text{with probability } 1 - \epsilon_t \\ \sim \mathcal{U}(\mathcal{P}) & \text{with probability } \epsilon_t \end{cases} \tag{3.14}$$

Additionally, the action value is implemented in the Q-function to incorporate the behaviors of other agents into the learning process. The Q-function is defined as followed:

$$Q_i(p_{j,t}, p_{i,t}) \leftarrow (1-\alpha)Q_i(p_{j,t}, p_{i,t}) + \alpha \left(\omega(p_{j,t}, p_{i,t}) + \gamma \cdot \omega(p_{j,t+1}, p_{i,t}) + \gamma \cdot max_{p'_i} AV_i(p_{j,t+1}, p'_i)\right) \tag{3.15}$$

By maximising $AV_i(p_{j,t+1}, p'_i)$, JAL-AM ensures that the learning agent considers the modeled behaviors of the other agent. Our implementation of JAL-AM for two agents in a sequential Bertrand duopoly is shown in Algorithm 2.

Algorithm 2 Sequential JAL-AM

1: **Initialize:**
Learning parameter α and discount factor γ
$i = 1$ and $j = 2$
Choose random prices in $\mathcal{P}$ for both players $t = 1, 2$
Q-tables as 3D arrays
Agent models $\hat{\sigma}_j(p_j|p_i) = \frac{1}{|\mathcal{P}|}$ for all $j \neq i, p_j, p_i \in \mathcal{P}$
Counters for $\{i, j\}$

2: **for** $t = 3$ to $\mathcal{T}$ **do**

3: Update Counter C_i and Agent model $\hat{\sigma}_j$:

$$C_i(p_{i,t-1}, p_{j,t-2}) = C_i(p_{i,t-1}, p_{j,t-2}) + 1$$

$$\forall p'_j \in \mathcal{P}\text{: } \hat{\sigma}_j(p'_j|p_{i,t-1}) \leftarrow \frac{C_i(p_{i,t-1}, p'_j)}{\sum_{p'_i \in P} C_i(p'_i, p'_j)}$$

4: Update Q-function

$$Q_i(p_{j,t-2}, p_{i,t-1}) \leftarrow Q_i(p_{j,t-2}, p_{i,t-1})$$
$$+\alpha \left(\omega(p_{j,t-2}, p_{j,t-1}) + \gamma \cdot \omega(p_{j,t-1}, p_{i,t-1}) + \gamma \cdot \max_{p'_i} AV_i(p_{j,t-1}, p'_i) - Q_i(p_{j,t-2}, p_{i,t-1}) \right)$$

5: Set $p_{i,t}$ according to

$$p_{i,t} \begin{cases} = \arg\max_{p'_i} AV_i(p_{j,t-1}, p'_i) & \text{with probability} \quad 1 - \epsilon_t \\ \sim \mathcal{U}(P) & \text{with probability} \quad \epsilon_t \end{cases}$$

and $p_{j,t} = p_{j,t-1}$

6: Change agents, $j \leftarrow i$ and $i \leftarrow j$

7: **end for**

3.3.3 WoLF-PHC

Win or Learn Fast Policy Hill Climbing (WoLF-PHC) is a policy-based reinforcement learning algorithm that was proposed and discussed in Bowling and Veloso (2001; 2002). The choice to use WoLF-PHC is similar to the decision of Fischer (2023), which is based on the popularity and its historically good reputation (Stimpson and Goodrich, 2003).

WoLF-PHC uses two distinct learning rates, δ_w and δ_l, to differentiate the learning rates for when it is winning and losing. As the algorithm needs to learn faster when loosing, $\delta_w < \delta_l$ must hold. Winning is determined by comparing the historical average performance to the current performance. So if the agent has a better performing

strategy than the historical average, it knows that it is winning. The formal selection of the learning rate δ is defined as:

$$\delta = \begin{cases} \delta_w & \text{if } \sum_{p'_i} \pi_i(p'_i|p_{j,t}) Q_i(p_{j,t}, p'_i) > \sum_p \bar{\pi}_i(p'_i|p_{j,t}) Q_i(p_{j,t}, p'_i) \\ \delta_l & \textit{otherwise} \end{cases} \tag{3.16}$$

In this way the algorithm can change its learning rate if its strategy performs better or worse than the historical average policy, $\bar{\pi}_i$.

The policy for agent *i*, π_i is updated, using the following update rule:

$$\pi_i(p_{i,t}|p_{j,t}) \leftarrow \pi_i(p_{i,t}|p_{j,t}) + \Delta(p_{j,t}, p_{i,t}) \tag{3.17}$$

where

$$\Delta(p_{j,t}, p_{i,t}) = \begin{cases} \delta & \textit{if} \quad p_{i,t} = \textit{arg max}_{p'_i} Q_i(p_{j,t}, p'_i) \\ \frac{-\delta}{|P|-1} & \textit{otherwise} \end{cases} \tag{3.18}$$

and δ being the learning rate, described in equation 3.16. This update process is critical in policy hill climbing, enabling iterative policy improvements through adjustments based on the difference between the current and the optimal action values.

To update the average policy for agent i, $\bar{\pi}_i$, WoLF-PHC keeps a counter, C_i, to track how many times the opponent agent has set different prices. This is simply updated as:

$$C_i(p_{j,t}) \leftarrow C_i(p_{j,t}) + 1 \tag{3.19}$$

The average policy is updated as:

$$\forall p_{i,t} \in P_{i,t} : \bar{\pi}_i(p_{i,t}|p_{j,t}) \leftarrow \bar{\pi}_i(p_{i,t}|p_{j,t}) + \frac{1}{C_i(p_{j,t})}(\pi_i(p_{i,t}|p_{j,t}) - \bar{\pi}_i(p_{i,t}|p_{j,t})) \tag{3.20}$$

For price selection in WoLF-PHC, we use the ϵ-greedy strategy, similar to the one used in Q-learning and JAL-AM. This means we have the same probabilities of exploring and exploiting. However, when exploting, WoLF-PHC uses it's policy, which is a distribution of actions to take based on the current state, instead of always choosing the maximised best action. This means there are different probabilities for each action. Our price selection strategy can be described as:

$$p_{it} \begin{cases} \sim \pi_i(\cdot|p_{j,t}) & \text{with probability } 1 - \epsilon_t \\ \sim U(\mathcal{P}) & \text{with probability } \epsilon_t \end{cases} \tag{3.21}$$

Moreover, the updating of the Q-table, like in Fischer (2023), is formulated as

$$Q_i(p_{j,t}, p_{i,t}) \leftarrow (1-\alpha)Q_i(p_{j,t}, p_{i,t}) + \alpha \left(\omega(p_{j,t}, p_{i,t}) + \gamma \cdot \omega(p_{j,t+1}, p_{i,t}) + \gamma^2 \cdot \max_{p_i'} Q_i(p_{j,t+1}, p_i') \right) \tag{3.22}$$

This integration of a dynamic learning rate and policy adjustment based on performance relative to historical averages allows WoLF-PHC to adapt effectively to varying environments, optimizing learning from both successes and setbacks. The pseudocode for WoLF-PHC can be seen in Algortihm 3.

Algorithm 3 Sequential WoLF-PHC

1: **Initialize:**
Learning parameter α and discount factor γ
Choose random prices in $\mathcal{P}$ for both firms for $t = \{1, 2\}$
Q-tables for both firms
policies for both firms, $\pi_i(p_i|p_j) = \frac{1}{|P|}$
average policies for both firms, $\bar{\pi}_i(p_i|p_j) = 0$
The counters for states visited, $C_i(p_j) = 0$

2: **for** t=3 to $\mathcal{T}$ **do**

3: Update $Q_i(p_{j,t-2}, p_{i,t-1})$:

$$Q_i(p_{j,t-2}, p_{i,t-1}) \leftarrow (1-\alpha) \cdot Q_i(p_{j,t-2}, p_{i,t-1}) +$$
$$\alpha \left(\omega(p_{j,t-2}, p_{i,t-1}) + \gamma \omega(p_{j,t-1}, p_{i,t-1}) + \gamma^2 \max_{p_i'} Q_i(p_{j,t-1}, p_i') \right)$$

4: Update $C_i(p_{j,t-2})$ and $\bar{\pi}_i(p_{j,t-2}, p_{i,t-1})$

$$C_i(p_{j,t-2}) = C_i(p_{j,t-2}) + 1$$

$$\forall p_i' \in \mathcal{P} : \bar{\pi}_i(p_i'|p_{j,t-2}) \leftarrow \bar{\pi}_i(p_i'|p_{j,t-2}) + \frac{1}{C_i(p_{j,t-2})}(\pi_i(p_i'|p_{j,t-2}) - \bar{\pi}_i(p_i'|p_{j,t-2}))$$

5: Update $\pi_i(p_{i,t-1}|p_{j,t-2})$ and constrain to a legal probability distribution.

$$\pi_i(p_{i,t-1}|p_{j,t-2}) \leftarrow \pi_i(p_{i,t-1}|p_{j,t-2}) + \Delta(p_{j,t-2}, p_{i,t-1})$$

where

$$\Delta(p_{j,t-2}, p_{i,t-1}) = \begin{cases} \delta & \text{if} \quad p_{i,t-1} = arg\ max_{p_i'} Q_i(p_{j,t-2}, p_i') \\ \frac{-\delta}{|\mathcal{P}|-1} & \text{otherwise} \end{cases} \tag{3.23}$$

and

$$\delta = \begin{cases} \delta_w & \text{if } \sum_{p_i'} \pi_i(p_i'|p_{j,t-2}) Q_i(p_{j,t-2}, p_i') > \sum_p \bar{\pi}_i(p_i'|p_{j,t-2}) Q_i(p_{j,t-2}, p_i') \\ \delta_l & \text{otherwise} \end{cases} \tag{3.24}$$

6: take action $p_{i,t}$ as

$$p_{i,t} \begin{cases} \sim \pi_i(\cdot|p_{j,t-1}) & \text{with probability} \quad 1-\epsilon_t \\ \sim \mathcal{U}(P) & \text{with probability} \quad \epsilon_t \end{cases}$$

and $p_{j,t} = p_{j,t-1}$

7: Change players, $i \leftarrow j$ and $j \leftarrow i$

8: **end for**

3.3.4 Configuration

To carry out our simulations, choices have been made. Previously, we introduced different rates and factors, which has not been given a specific value. In this section we will specify each variable's value and describe our rationale for these selections.

When doing a simulation, the first suitable variable to discuss is how long the simulation should last, $\mathcal{T}$. This variable indicates how many times the agents collectively sets a price. As Klein (2021) we will use $\mathcal{T} = 500.000$. Klein found this time frame large enough for the algorithms to find stability in their decision-making process.

For the learning rate and discount factor, we have chosen to use $\alpha = 0.3$ and $\gamma = 0.95$ based on the work of Klein (2021). Klein found that this learning rate, α was a suitable compromise between learning new information and forgetting old. For the discount factor, γ, Klein suggested setting this value close to 1 but not too close, for it to be ideal. $\gamma = 0.95$ allows the algorithm to consider future consequences but still gives enough weight to the immediate rewards, so that it does not neglect the importance of the decision made now.

As for the learning rates for WoLF-PHC, we will simply choose the same as Fischer (2023). When winning, the learning rate is $\delta_w = 0.1$, such that the agents learns at a slower pace. However, if the agent is loosing then the learning rate is $\delta_l = 0.6$ and they will learn rather aggressively.

For our $\epsilon - greedy$ strategy we will need to pick an ϵ which determines the probability of setting a random price to the time t. Klein (2021) used an epsilon that decreases from 100% probability of choosing a random price at $t = 1$. Halfway through $\mathcal{T}$, $t = 0.5T$ should be equal to a probability of 0.1% and a the end $t = \mathcal{T}$ it should have the probability of 0.0001% of choosing a random price. This can be formally written as the decay function $\epsilon_t = (1 - \theta)^t$ where

$$\theta = -\left(\left(\frac{1}{1000000}\right)^{\frac{1}{\mathcal{T}}}\right) + 1 \tag{3.25}$$

This ensures that when an agent progress through time, the agent increasingly relies on the information it has learned rather than setting random prices.

3.3.5 Performance metrics

In this section, we outline the performance metrics employed to evaluate the efficacy of our algorithms.

3.3.5.1 Profitability

To assess the performance of our algorithms, we will adopt the same profitability metric used by Klein (2021). Klein's approach evaluated the final profitability of a run lasting T periods by using the average profits from the last 1000 periods, expressed mathematically as:

$$\text{Profitability: } \prod_i = \frac{1}{1000} \sum_{t=\mathcal{T}-1000}^{\mathcal{T}} \omega(p_{i,t}, p_{j,t}) \tag{3.26}$$

The choice of the last 1000 periods is not trivial. Klein argued his decision by noting that prices often fluctuate, and the price-cycles occurring will always be significantly shorter than 1000 periods. This duration ensures the measurement captures a stable average.

In evaluation of the profitability metric we use two benchmarks; The joint-maximizing profit and a competitive benchmark. The joint-maximizing profit is 0.125 and is the maximum per-firm profit they are able to get. To obtain the joint-maximizing profit both firms set the monopolistic price of 0.5. The competitive benchmark we use is 0.0611 taken from Klein (2021). Klein argues that in sequential pricing environments pricing at or slightly above marginal cost is not subgame optimal in the long run. This means that while pricing at marginal cost might seem like a competitive benchmark in a static setting, it does not hold in a dynamic sequential setting. Firms might find themselves locked into a suboptimal pricing strategy if they don't consider the future consequences of their pricing decisions. The competitive benchmark approximates the most competitive outcome identified by Maskin and Tirole (1988), known as the Edgeworth price cycle. In this cycle, firms undercut each other incrementally until prices reach a lower bound, after which one firm resets prices to one increment above the monopoly price, and the cycle restarts.

3.3.5.2 Average Profit Gain

We find it important to evaluate the level of collusion between firms in our simulations. Thus, to measure the degree of collusion between the competing firms, we implement a measure introduced by Calvano *et al.* (2020) called the average profit gain. The average profit gain is denoted as:

$$\Delta \equiv \frac{\bar{\omega} - \omega^N}{\omega^M - \omega^N} \tag{3.27}$$

Where $\bar{\omega}$ is the average per-firm profit of the last 1000 periods, ω^N is the profit in the static Bertrand-Nash equilibrium, and ω^M is the joint-maximizing profit, when firms enter full collusion. As the static Bertrand-Nash equilibrium is marginal costs and in our case equal to zero, the average profit gain simplifies to:

$$\Delta \equiv \frac{\bar{\omega}}{\omega^M} \tag{3.28}$$

Thus the measure is a ratio of how close competing firms are to the joint-maximizing profit. Furthermore, when $\bar{\omega} = 0 \Rightarrow \Delta = 0$ firms are at a fully competitive profit. When $\bar{\omega} = 1 \Rightarrow \Delta = 1$ firms are at the fully collusive profit of 0.125.

4 Implementation and Optimization

This chapter presents the implementation of our simulations and the various optimization techniques employed to enhance performance.

4.1 Implementation

For our simulations, we utilized Python due to its extensive libraries and ease of use for data handling and algorithm implementation. We have created libraries for the simulations, so we could use an executable python script, Jupyter Notebook, to vizualise the results. Further, for the visualisation we have used NumPy for numerical computations and Matplotlib for creating graphs and figures. Given the necessity to perform multiple runs of each simulation, we implemented a robust data saving mechanism to store the resultant data efficiently. The volume of this data is substantial, making it necessary to consider optimization.

4.2 Optimization

4.2.1 Numba

Python, being an interpreted language, is inherently slower than compiled languages like C. The primary reason being that, Python is an interpreted programming language. This means that when Python code is run, every line has to be interpreted one by one, whereas compiled languages convert the entire code into machine code before execution. To mitigate Python's performance limitations, we employed Numba (2024), a Just-In-Time (JIT) compiler that translates Python functions into machine code at runtime.

Numba is particularly effective for optimizing numerical operations involving NumPy arrays, loops, and functions. By decorating functions with `@njit`, we ensure that the Python interpreter compiles these functions into machine code, significantly enhancing their execution speed. This approach allows us to execute a substantial portion of our code at near-C speed, thereby reducing the overall runtime of our simulations.

4.2.2 Parallelization

Despite the performance gains from JIT compilation, further optimization is possible through parallelization. Parallelization involves distributing computations across multiple CPU cores, allowing simultaneous execution and thus faster processing. Modern computers are equipped with multi-core processors, and utilizing these cores effectively can significantly improve performance.

In our implementation, we employed Python's (2024) `concurrent.futures` module, which provides a high-level interface for asynchronously executing tasks, allowing other tasks to proceed without waiting for the completion of the first tasks. The module offers two main classes of executors: `ThreadPoolExecutor` and `ProcessPoolExecutor`. For our simulations, we used `ProcessPoolExecutor` to parallelize the execution of simulation runs. This choice is advantageous as it avoids the Global Interpreter Lock (GIL) in Python, allowing true parallelism by running each simulation in a separate process.

By distributing simulation runs across multiple processes, we efficiently utilize available CPU cores, significantly reducing the total computation time. Careful consideration was given to data handling to avoid conflicts and ensure consistent results when multiple processes access shared data.

5 Results

This section presents the findings from our study, focusing on the behavior of the three reinforcement learning algorithms: Q-learning, JAL-AM, and WoLF-PHC, in a sequential price-setting game under both symmetric and asymmetric information.

Our research investigates how these algorithms perform, particularly in terms of their tendency to exhibit collusive behavior. We use the performance metrics introduced earlier to analyze the decision-making processes of these algorithms in different informational settings. We also examine how asymmetric information affects the strategies these algorithms adopt. Furthermore, we test the robustness of algorithms by varying the granularity of the price grid and altering the probability of observing incorrect prices.

5.1 Convergence

Before analyzing our results, we need to verify the convergence of the algorithms to ensure they have reached a final strategy. Since all algorithms utilize a Q-table, we will check for convergence by observing changes in the Q-table over the last 1,000 periods. To ensure the consistency of convergence, we conducted 1,000 simulations for each setting. Given the limitations of computer precision, we set a tolerance level of 10^{-10}.

Our findings indicate that all runs achieved convergence without issues, even when asymmetric information was introduced. This outcome is expected because the asymmetry was only implemented in the action-taking part. The agent with asymmetric information updates its strategy based on the true state, having no problem of converging. When the agent with incomplete information never observes the true state in the action module, the agent with complete information adapts to an opponent that selects prices randomly. Additionally, the algorithms are designed to converge because we set ϵ (the exploration rate) to decrease over time, making exploration very unlikely by the

end of a run. This ensures that we find a strategy, even if it is not the most optimal one. Therefore, we can confidently proceed with the analysis of our results.

5.2 Results with $k = 6$

In this section, we examine the behavioral patterns of the algorithms under symmetric and asymmetric information with $k = 6$. In symmetric information settings, the agents are homogeneous. In asymmetric information settings, Firm 1 has complete information, while Firm 2 has incomplete information. As described in section 3.1.4, under asymmetric information Firm 2 has a probability μ of observing a random price in the moment of action. We use $\mu = 0.05$, so the agent observes a random state 5% of the time, as we find 5% to be a realistic assumption.

5.2.1 Profitability

To measure the performance of our algorithms, we investigate the average profitability to determine if our algorithms exhibit collusive behavior. The average profitabilities over 1000 runs for Q-learning, JAL-AM and WoLF-PHC has been plotted in Figure 5.1.

Our results show that all algorithms exhibit collusive behavior with average profitability around 0.11, significantly higher than the competitive benchmark, approximating the most competitive Edgeworth price cycle Markov Perfect Equilibrium (MPE) identified by Maskin and Tirole (1988). Symmetric Q-learning ends with an average profitability at around 0.109, similar to the findings in Klein (2021). WoLF-PHC manages to converge to almost the same profitability, while JAL-AM converges to a slightly higher profitability at around 0.112.

When introducing asymmetric information with $\mu = 0.05$, all algorithms show a decrease in average profitability. However, they still gain supra-competitive profits, converging well above the competitive benchmark and close to the joint profit-maximizing level. Asymmetric information appears to impact WoLF-PHC's performance the most, decreasing its profitability from 0.109 to 0.104. Our expectation was that since WoLF-PHC can learn stochastic strategies, it would perform better than Q-learning. This was not the case. Q-learning decreases from around 0.109 to 0.106, a smaller change in

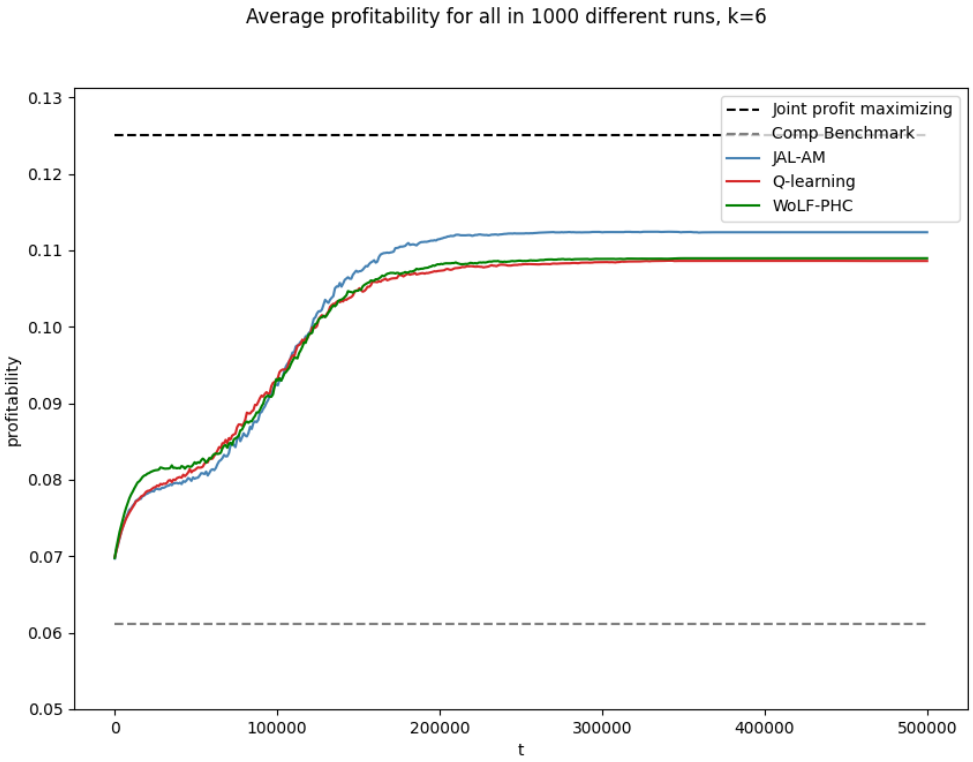


Figure 5.1: This graph shows the average profitability for 1000 different runs, where $\mathcal{T} = 500,000$ and $k = 6$

profitability than WoLF-PHC. It seems that, WoLF-PHC does not find a more profitable strategy than Q-learning, even though, it should be able to learn stochastic strategies.

JAL-AM's profitability decreases from approximately 0.112 to 0.109, a similar drop observed in Q-learning. While JAL-AM, like WoLF-PHC, can learn stochastic strategies, it does not outperform Q-learning when asymmetry is introduced. This indicates that the ability to learn stochastic strategies does not provide a significant advantage in this context. It is important to note that the individual profitability of the two firms changes, which accounts for the average profitability loss across the algorithms. These changes are illustrated in Appendix B. The graphs in the appendix show that with asymmetry introduced, the firm with complete information achieves higher profits compared to the firm with incomplete information, due to the deviations made by the asymmetric firm leading to lower profits. However, the incomplete information firm in the JAL-AM scenario is less affected, explaining the smaller decrease in average profitability.

The time required to learn the final strategy also increases under asymmetric information. With symmetric information agents need around 200,000 periods to reach

their highest profitability whereas under asymmetric information agents need around 250,000 periods to obtain their highest profitability. Under asymmetric information an additional element of stochasticity is introduced to the learning process in μ. We would argue that this is the reason for the slower convergence to the final strategy.

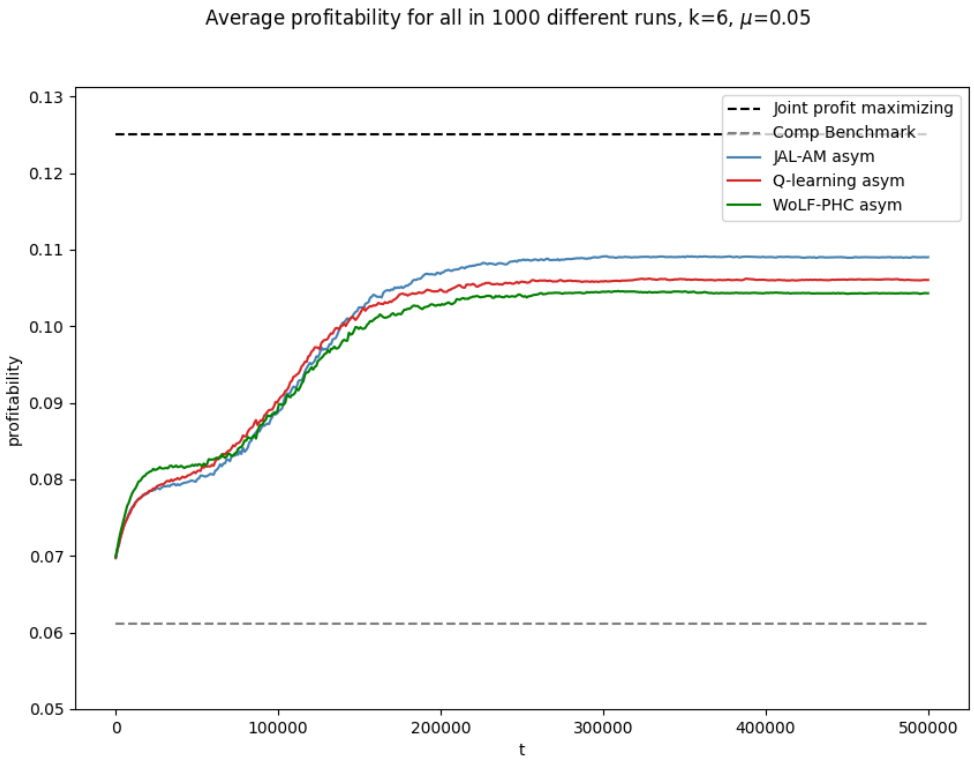


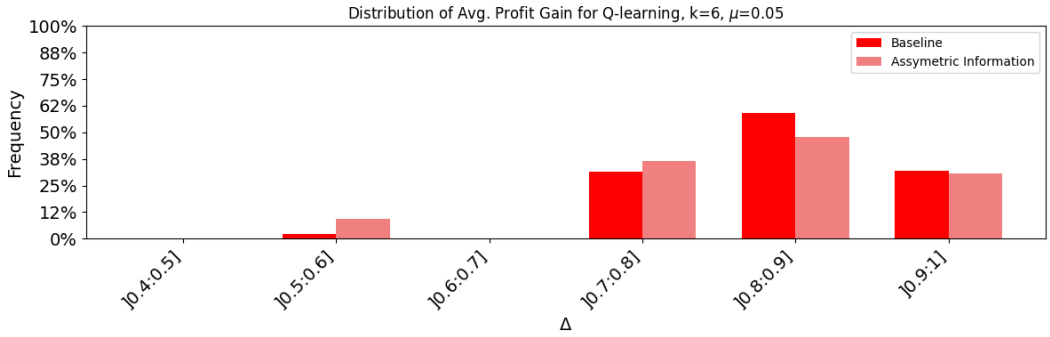
Figure 5.2: This graph shows the average profitability for 1000 different runs under asymmetric information, where $T = 500,000$, $k = 6$ and $\mu = 0.05$

5.2.2 Average Profit Gain

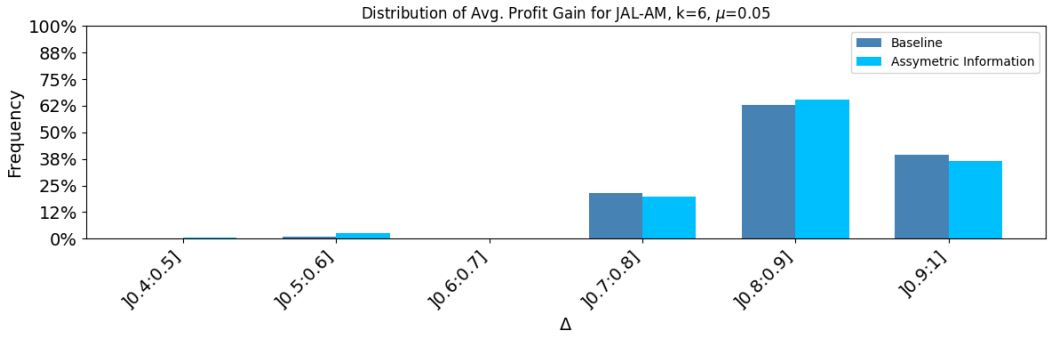
In this section, we investigate the degree of collusion through the average profit gain metric. This involves assessing how close the algorithms are to the joint-profit maximising profit and how often they learn to achieve it. Furthermore, this metric helps compare the degree of collusion under both symmetric and asymmetric information.

Figure 5.3 illustrate the distributions of the average profit gain over 1000 periods. All three algorithms, where the most frequent equilibrium is $\Delta]0.8, 0.9]$ which corresponds to profits of 0.1-0.1125, indicate that the average profitability is also the most frequent equilibrium. Under asymmetric information with $\mu = 0.05$, the frequency of high collusive behavior $\Delta]0.9, 1]$ slightly decreases, as expected.

(a) Q-learning



(b) JAL-AM



(c) WoLF-PHC

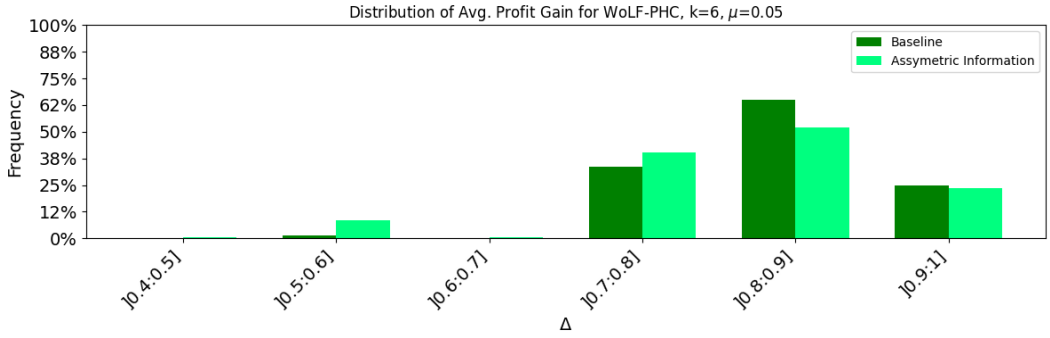


Figure 5.3: This plot shows the shows the distribution of Average Profit Gain for 1000 simulation, $\mathcal{T} = 500,000$, $k = 6$ and $\mu = 0.05$

For Q-learning, we see a relatively large increase in the frequency of equilibria in $\Delta]0.5, 0.6]$. Similarly, $\Delta]0.7, 0.8]$ increases under asymmetric conditions, resulting in a decrease in higher intervals $\Delta]0.8, 0.9]$.

JAL-AM is less affected by asymmetry, showing only small changes in the frequency of the Δ intervals. There is a slight increase in $\Delta]0.5, 0.6]$, which likely contributes to the decrease in profitability.

WoLF-PHC shows a pattern similar to Q-learning, where the decrease in $\Delta]0.8, 0.9]$ is distributed among $\Delta]0.5, 0.6]$ and $\Delta]0.7, 0.8]$.

Overall, at $k = 6$ and $\mu = 0.05$, asymmetric agents mostly affect Q-learning and WoLF-PHC, while JAL-AM remains relatively unchanged.

5.2.3 Price Cycles

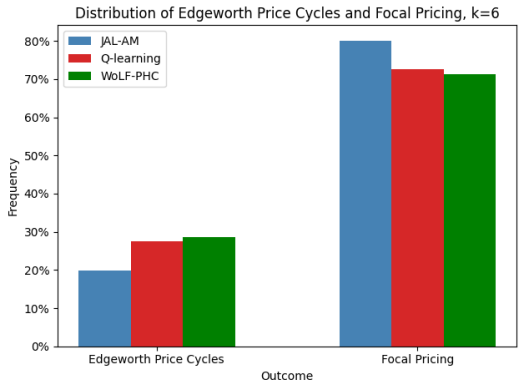
As described in section 3.1.6.1, when the algorithms have converged to a collusive strategy, we find two equilibria: Focal pricing and Edgeworth price cycles. This section illustrates the distributions of Edgeworth price cycles and focal pricing under symmetric and asymmetric information. Additionally, in focal pricing under asymmetric information, we provide examples of reward-punishment strategy.

In Figure 5.4 we have plotted the frequencies of focal pricing and Edgeworth price cycles for 1000 runs with $k = 6$ and $\mu = 0.05$ under asymmetric information. With symmetric information, the majority of outcomes end in focal pricing for all three algorithms. However, when asymmetric information is introduced, the pricing strategies change. For Q-learning and WoLF-PHC, there is almost an even 50/50 distribution between Edgeworth cycles and focal pricing. JAL-AM remains almost unchanged, with only a small increase in Edgeworth cycles under asymmetry.
This suggests that under asymmetric information with $k = 6$, Q-learning and WoLF-PHC results in more price cycles, while JAL-AM remain consistent in its strategy outcome. A possible reason for JAL-AM's consistency is its capability to learn stochastic policies, which theoretically should also be true for WoLF-PHC, but it does not achieve this in practice.

Additionally, in focal pricing under asymmetric information, we found occurrences of punishment and exploitation when a firm observed an incorrect price and therefore sat a sub-optimal price. This is interesting as reward-punishment strategies are often part of collusive behavior. Figure 5.5 shows an example of punishment and exploitation under asymmetric information for Q-learning. Note that these examples are from single runs and are not fully representative of all runs. These examples, though taken from single runs, showcase similar patterns for JAL-AM and WoLF-PHC.

In the example of punishment, Firm 2 observes an incorrect price and mistakenly deviates from a price of $\frac{3}{6}$ to set a lower price at $\frac{2}{6}$. Firm 1 then punishes Firm 2 by

(a) Symmetric information



(b) Asymmetric information

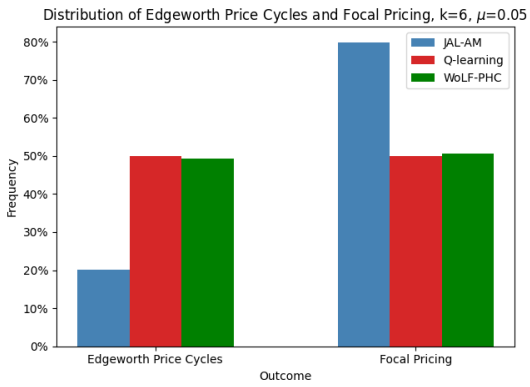
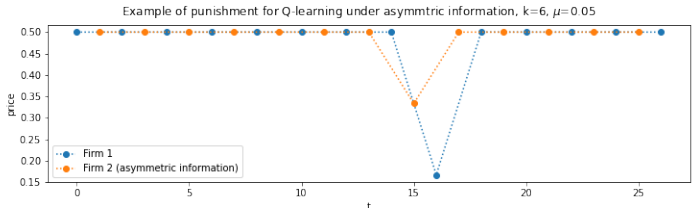


Figure 5.4: These plots show the distribution of Edgeworth Price Cycles and Focal Pricing, $\mathcal{T} = 500,000$, $k = 6$ and $\mu = 0.05$

undercutting to $\frac{1}{6}$. Once Firm 2 observes the correct price, it returns to the collusive price of $\frac{3}{6}$. These results resemble Klein (2021) findings when introducing a forced deviation, where the competing firms also return to collusive price levels.

(a) Punish



(b) Exploit

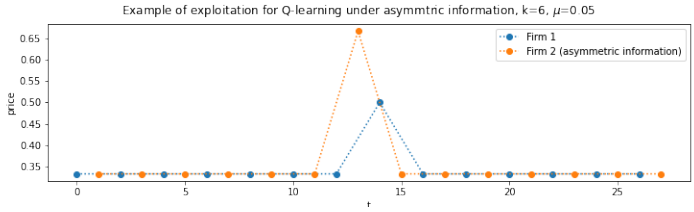


Figure 5.5: This graph shows examples of when a firm punishes and exploits, where $\mathcal{T} = 500,000$, $k = 6$ and $\mu = 0.05$

In the example of exploitation, Firm 2 observes an incorrect price and mistakenly deviates from a price of $\frac{2}{6}$ to set a higher price at $\frac{4}{6}$. This leads to Firm 1 exploiting Firm 2's action by setting the monopolistic price of 0.5. One might suggest that, Firm 1 hopes Firm 2 will also set the monopolistic price to achieve the joint-maximising profit. However, Firm 2 returns to the collusive price it learned before the deviation, maintaining the previously established collusive strategy.

5.3 Results with $k = 24$

In this section, we analyze the robustness of our result by increasing the fineness of the price grid. We examine the behavioral patterns of the algorithms under symmetric and asymmetric information with $k = 24$. As before, under asymmetric information, we use $\mu = 0.05$. We compare these results with results found when $k = 6$ both under symmetric and asymmetric information.

5.3.1 Profitability

When increasing the granularity of the price grid to $k = 24$, we find the same pattern as with $k = 6$, but the average profitability decreases slightly for all algorithms. Increasing the fineness of the price grid implies that the learning process becomes more complex. Furthermore, when the exploration module is fixed; while more prices are available, each state-action pair is less frequently visited, which could reduce the algorithm's ability to learn. One could argue that the firms have more opportunities to undercut each other and, therefore, on average have difficulties setting a higher collusive price. Additionally, firms gain a higher incentive to undercut the other firm, as they can take the whole market at almost the same price. As Klein (2021) argue, algorithms will tend to converge to Edgeworth cycles instead of focal pricing.

As shown in Figure 5.6 the profitability has decreased at $k = 24$, but once again, JAL-AM provides the highest average profitability. With symmetric information, we find that the algorithms learn to obtain collusive profits around 0.1065. Furthermore, we see that our learning curve becomes slightly flatter, indicating that it takes more time to converge to a final strategy. Fischer (2023) found that when k increased, Q-learning and WoLF-PHC's time needed for converging increased, while a decrease in the average profitability happened. This clearly coincides with our findings. We find that Q-learning drops from an average equilibrium around 0.109 to 0.106, while WoLF-PHC originates from an average profit of around 0.109 to 0.1064.

When the agents are under asymmetric information, we observe a further decrease in the profitability of the algorithms, although the same learning pattern is maintained. The results with asymmetric information are shown in Figure **??**. Notably, the gap between JAL-AM and the other algorithms becomes smaller. Here, all three algorithms converge to supra-competitive profits of around 0.104. Appendix C shows the individ-

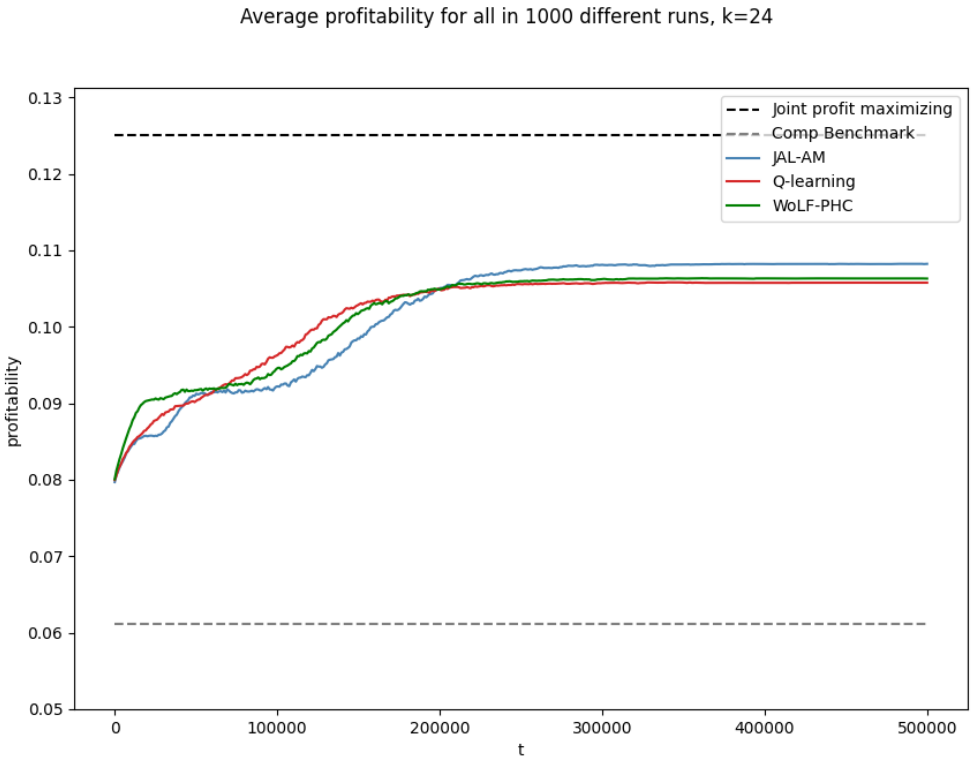


Figure 5.6: This graph shows the average profitability for 1000 different runs, where $\mathcal{T} = 500,000$ and $k = 24$

ual profitability of the firms. It reveals that the firm with complete information achieves higher profitability than the firm with incomplete information, which contributes to the overall decrease in average profitability. However the asymmetric firm still gains supra-competititve profits.

JAL-AM seems to need more time to converge than the other algorithms. However, this additional time to learn results in a higher average profitability than Q-learning and WoLF-PHC, but still with a decrease in profitability.

5.3.2 Average Profit Gain

In section 5.2.1 we found that the average profitability decreased when k increased. Hence, we expect the frequency of high collusive behavior to decrease.
Figure 5.8 depicts the distributions of average profit gain when $k = 24$ for all three algorithms. As expected, the distribution of average profit gain, when $k = 24$, saw a decrease in high collusive outcomes with $\Delta]0.9 : 1]$ for all algorithms under both symmetric and asymmetric information. As the frequency of $\Delta]0.9 : 1]$ decreased,

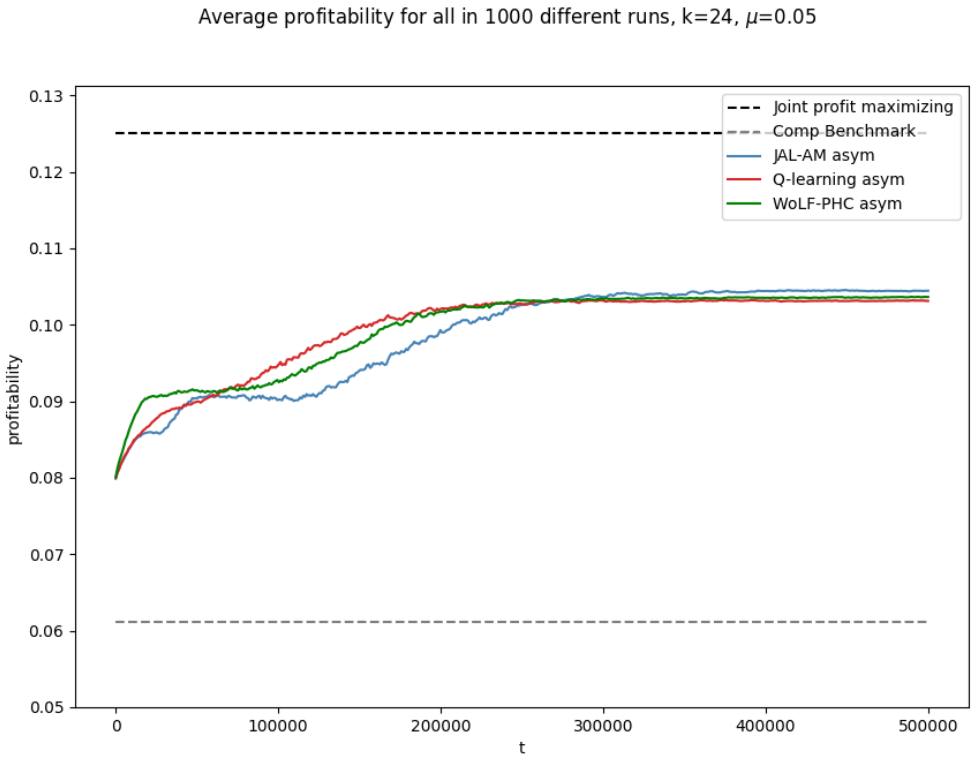


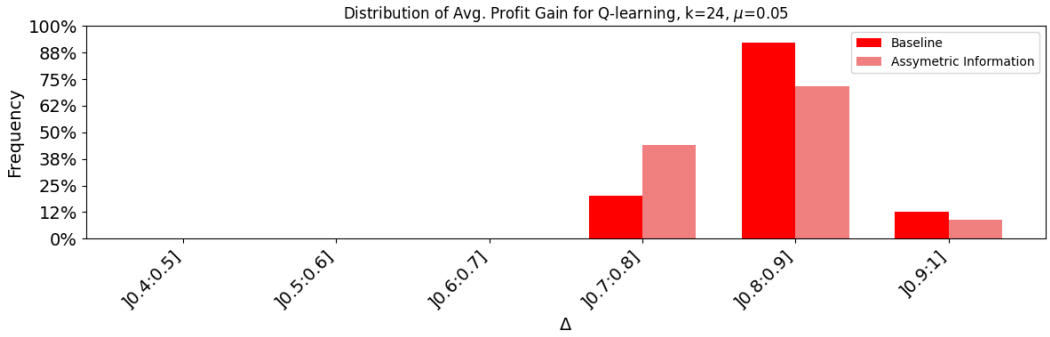
Figure 5.7: This graph shows the average profitability for 1000 different runs, where $\mathcal{T} = 500,000$, $k = 24$ and $\mu = 0.05$

more outcomes shifted to $\Delta]0.8 : 0.9]$, which overall saw an increase and where the majority of outcomes ended up. While the increase in k affected the frequency of high collusive behavior for all algorithms both under symmetric and asymmetric information, the effect was stronger under asymmetric information. Although the effect was stronger under asymmetric information, the algorithms still achieved supra-competitive profits.

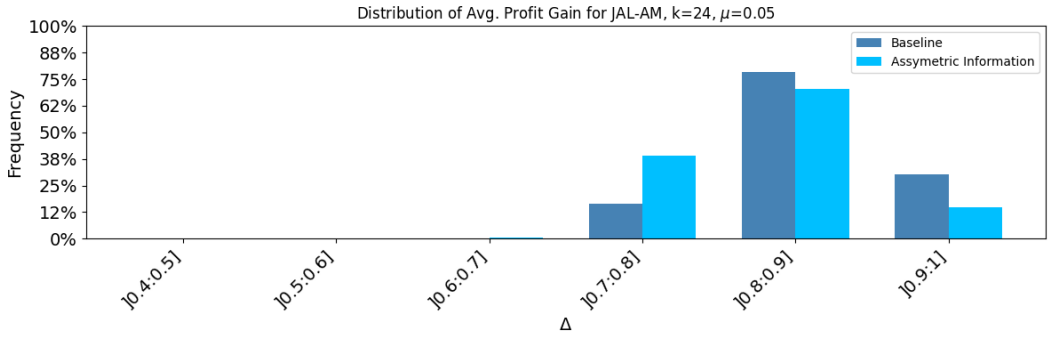
5.3.3 Price Cycles

With an increased fineness of the price grid we find that all three algorithms almost only end in Edgeworth price cycles both under symmetric and asymmetric information. This is seen in Figure 5.9.This was also found by Klein (2021), that when increasing k, Edgeworth price cycles get far more common as an equilibrium outcome. One thing to notice, however, is that JAL-AM seems to have a higher probability of setting focal prices.

(a) Q-learning



(b) JAL-AM



(c) WoLF-PHC

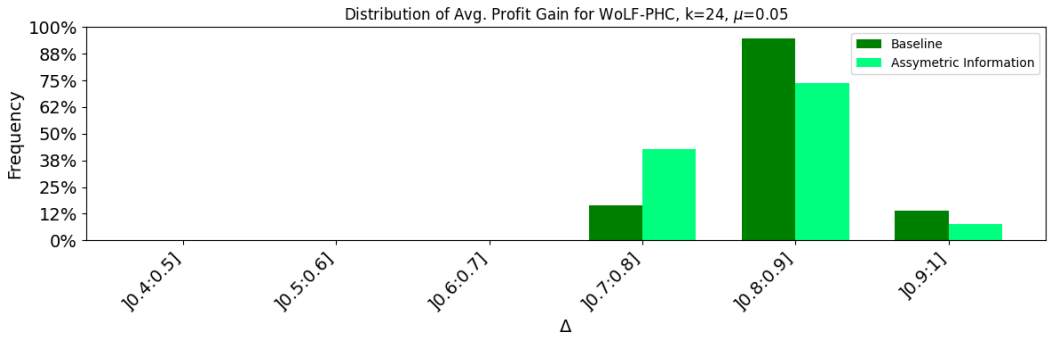


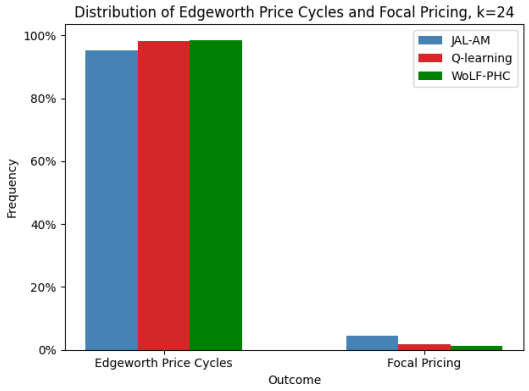
Figure 5.8: This plot shows the shows the distribution of Average Profit Gain for 1000 simulation, $\mathcal{T} = 500,000$, $k = 24$ and $\mu = 0.05$

Under asymmetric information, no significant changes has been made for Q-learning and WoLF-PHC. JAL-AM actually sees a slight increase in the number of focal pricing instances. This suggests that asymmetric information does not significantly influence the final strategy outcome when $k = 24$.

At $k = 24$, when the majority of final strategy outcomes result in Edgeworth price cycles, we do not find occurrences of punishment and exploitation as we did when $k = 6$. A possible reason for this is that when firms are engaged in price cycles, they are already punishing each other by undercutting until it becomes too costly. In other

words, there is not a firm that deviates from the current equilibria strategy by setting different price than anticipated.

(a) Symmetric



(b) Asymmetric information

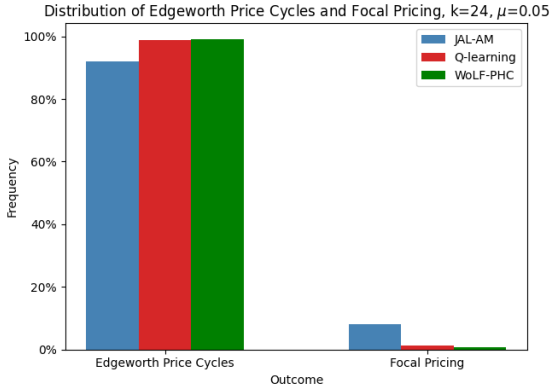


Figure 5.9: These plots show the distribution of Edgeworth Price Cycles and Focal Pricing for 1000 runs when $\mathcal{T} = 500,000$, $k = 24$ and $\mu = 0.05$

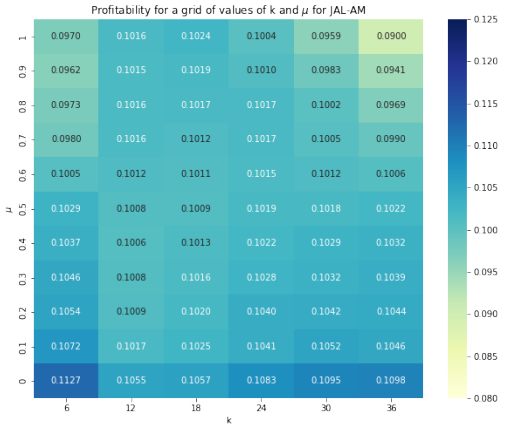
5.4 Robustness in μ and k

As the values of μ and k both influence the average profitability, we examined how they behave when each variable is held fixed. This is illustrated with heatmaps, where each "*scenario*" have been run 1000 times. The heatmap for Q-learning is shown in Figure 5.10a. When $k = 6$, the impact of asymmetric information has a greater effect on profitability. As μ increases we observe that the profitability decreases the most as when $k = 6$. However, when k is larger, the profitability seems to be more robust to asymmetry, as there are only slight changes to the profitability of a larger k when increasing μ.An explanation for this is that when we increase μ, the firm with complete information captures the profit from the firm with incomplete information whenever the incomplete firm makes deviations. This is illustrated in Appendix E; when $\mu = 0.3$, the firm with incomplete information achieves profitability close to our competitive benchmark.

The heatmap for JAL-AM is shown in Figure 5.10b. As JAL-AM is able to learn stochastic policies, it seems to outperform Q-learning and WoLF-PHC when $k = 6$. It still decreases in profitability alongside μ, but not as rapidly as Q-learning and WoLF-PHC. This is shown in Appendix D when $k = 6$ and $\mu = 0.3$, where the firm with incomplete information is able to gain higher profits than firms with incomplete information in Q-learning and WoLF-PHC. We observe the same pattern for JAL-AM as for Q-learning

(a) Q-learning

(b) JAL-AM



(c) WoLF-PHC

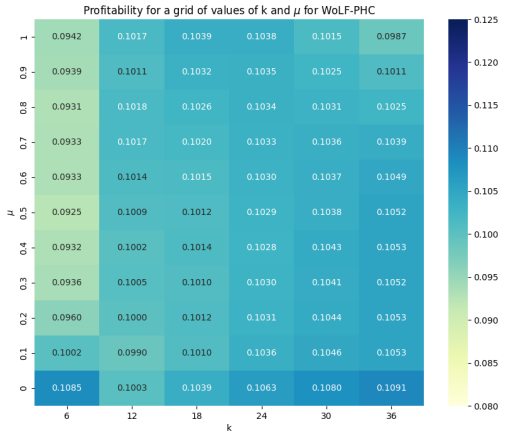


Figure 5.10: The heatmaps shows the average Profitability for a grid of values for k and μ in 1000 runs, $\mathcal{T} = 500,000$

when k and μ increase: the profitability stabilizes as k increases and is not affected by changes in μ.

The heatmap of WoLF-PHC is shown in 5.10c. We observe, that WoLF-PHC struggles with $k = 6$ just like Q-learning, as the profitability decreases rather quick. Even though WoLF-PHC is able to learn stochastic strategies, it does not achieve the effect on its outcome as JAL-AM. WoLF-PHC seems to be performing a lot like Q-learning, except for minor differences.

It seems that JAL-AM is more robust in finding higher equilibria than Q-learning and WoLF-PHC when $k = 6$.

6 Discussion

This study confirms the findings of Fischer (2023), demonstrating that MARL agents can potentially achieve higher profitability than Q-learning agents. Specifically, we found that while WoLF-PHC does not outperform Q-learning in terms of average profitability, the introduction of JAL-AM shows that certain MARL algorithms can indeed enhance profitability.

Our results align with Fischer's findings that WoLF-PHC does not converge to higher average profitability than Q-learning. However, JAL-AM was able to outperform both under certain conditions, suggesting potential advantages of advanced MARL algorithms in economic environments.

Under asymmetric information conditions, we observed that firms could still engage in collusive behavior, although the degree of collusion and the effectiveness of each algorithm varied based on the informational environment and the price grid granularity. Specifically, JAL-AM achieved higher equilibrium states than Q-learning and WoLF-PHC when asymmetric information was present. Notably, when $k = 6$, WoLF-PHC exhibited lower average profitability than Q-learning. Our results of finding supra-competitive profits and collusive behaviour under asymmetric information aligns with the conclusions drawn by Dou *et al.* (2024). They implement imperfect monitoring, which is similar to our asymmetric information, finding that firms also achieve supra-competitive profits. Although it is important to note, that all firms in their research were under imperfect monitoring, while we only had one firm with incomplete information. The findings on asymmetric information, particularly when μ is low, can be related to the concept of forced deviation observed by Klein (2021). Asymmetric information acts like frequent forced deviations, causing firms to temporarily deviate from equilibrium but eventually returning. This highlights the resilience of collusive equilibria even under uncertain conditions.

Furthermore, the introduction of asymmetric information significantly increased the convergence time for algorithms to settle on final strategies. Despite this, even in extreme cases, when μ is high, where agents more frequent observe random prices, they

managed to achieve supra-competitive average profits.However, this is because the firm with complete information is able to gain much higher profitability than the incomplete firm, and therefore skewing the average profitability. Brown and MacKay (2021, p. 42) argue, that policymakers should prohibit algorithms from directly conditioning on rivals prices to prevent such price increases. Our findings somewhat coincide with with Brown and MacKay, as when μ is high, the firm with incomplete information often conditions on a random price rather than directly on the competitors price and achieves a decrease in profits. One could imagine, that if both firms were under incomplete information, the profitability would decrease for both firms.

Our results rely on the use of the ϵ-greedy decision strategy. While this strategy is widely used in reinforcement learning, it may not be the most optimal choice for our specific algorithms. According to Fischer (2023), who compared the ϵ-greedy strategy with the Boltzmann strategy, the performance of algorithms can vary significantly depending on the decision strategy employed. Their study revealed that Q-learning, when utilizing the ϵ-greedy strategy, achieved a lower average profit compared to other MARL algorithms that employed different strategies and achieved higher profitabilities. Given these findings, it could be beneficial to explore the implementation of JAL-AM using the Boltzmann decision strategy to evaluate potential improvements in performance. However, it is important to note that the WoLF-PHC algorithm is not compatible with the Boltzmann decision strategy, which limits the scope of such an investigation for this particular algorithm.

The simulation-based nature of this study implies certain limitations. The constructed environment, though designed to imitate realistic settings, does not fully capture the disturbances and imperfections of real markets. For example, Calvano *et al.* (2020) and Eriksen *et al.* (2023) found that increasing the number of competing agents reduces the degree of collusion.

Future research should explore different forms of asymmetric information, particularly those affecting the learning phase rather than the action phase. Investigating real-world examples could provide valuable insights into how information availability influences collusive behavior and strategy convergence. Another avenue for future work is to examine different types of reinforcement learning algorithms, such as those incorporating deep learning and neural networks, which could provide insights into the scalability and adaptability of collusion strategies in more complex environments.

7 Conclusion

In this study, we have examined the behavior of multi-agent reinforcement learning (MARL) algorithms in comparison to Q-learning in a sequential Bertrand duopoly. We confirmed the findings of Fischer (2023), demonstrating that MARLs can yield higher profitability by introducing the MARL algorithm, JAL-AM. Our simulations reveal that all three algorithms exhibit a tendency towards collusion, achieving supra-competitive profits significantly higher than the competitive benchmark, with JAL-AM attaining the highest average profitability.

Furthermore, we investigated the behavior of these algorithms under conditions of asymmetric information. We found that while asymmetric information decreases profitability, the algorithms still manage to exhibit collusive behavior. Q-learning and WoLF-PHC, although effective, showed a more substantial decrease in profitability under asymmetric information compared to JAL-AM. This suggests that JAL-AM's ability to learn stochastic policies provides a more robust strategy in environments with incomplete information. The resilience of reinforcement learning, especially MARL, algorithms to informational asymmetry highlights their robustness in adapting to less-than-ideal informational settings, indicating a need for further investigation by policymakers and regulators.

8 Bibliography

Assad, Stephanie, Robert Clark, Daniel Ershov, and Lei Xu (2024). „Algorithmic pricing and competition: Empirical evidence from the German retail gasoline market“. In: *Journal of Political Economy* 132.3, pp. 000–000.

Bowling, Michael and Manuela Veloso (2001). „Rational and convergent learning in stochastic games“. In: *International joint conference on artificial intelligence*. Vol. 17. 1. Citeseer, pp. 1021–1026.

Bowling, Michael and Manuela Veloso (2002). „Multiagent learning using a variable learning rate“. In: *Artificial intelligence* 136.2, pp. 215–250.

Brown, Zach and Alexander MacKay (2021). „Competition in Pricing Algorithms“. In: *National Bureau of Economic Research*.

Calvano, Emilio, Giacomo Calzolari, Vincenzo Denicolo, and Sergio Pastorello (2020). „Artificial intelligence, algorithmic pricing, and collusion“. In: *American Economic Review* 110.10, pp. 3267–3297.

DCCA (2024). *Cartels are illegal*. `https://www.en.kfst.dk/competition/cartels/` (Accessed: 04/30/24).

Dou, Winston Wei, Itay Goldstein, and Yan Ji (2024). „Ai-powered trading, algorithmic collusion, and price efficiency“. In: *Jacobs Levy Equity Management Center for Quantitative Financial Research Paper*.

Eriksen, Sofus K, Christian Hempel, and Daan T Voorde (2023). *Exploring Multi-Agent Competition Dynamics: A Simulation Study on Reinforcement Learning and Collusion*.

EU (2024). *Consolidated version of the Treaty on the Functioning of the European Union*. `https://eur-lex.europa.eu/eli/treaty/tfeu_2008/art_101/oj` (Accessed: 23-04-2024).

Fischer, Julius Løve (2023). *Algorithmic Pricing Collusion using Reinforcement Learning*.

Klein, Timo (2021). „Autonomous algorithmic collusion: Q-learning under sequential pricing“. In: *The RAND Journal of Economics* 52.3, pp. 538–558.

Lepore, Nicolas (2021). „AI Pricing Collusion: Multi-Agent Reinforcement Learning Algorithms in Bertrand Competition“. MA thesis. Harvard College.

Maskin, Eric and Jean Tirole (1988). „A theory of dynamic oligopoly, II: Price competition, kinked demand curves, and Edgeworth cycles“. In: *Econometrica: Journal of the Econometric Society*, pp. 571–599.

Numba (2024). *Numba documentation*. `https://numba.readthedocs.io/en/stable/index.html` (Accessed: 04/06/24).

Python's (2024). *Concurrent.futures documentation*. `https://docs.python.org/3/library/concurrent.futures.html` (Accessed: 04/06/24).

Stimpson, Jeff L and Michael A Goodrich (2003). „Learning to cooperate in a social dilemma: A satisficing approach to bargaining“. In: *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 728–735.

Sutton, Richard S. and Andrew G. Barto (2015). "*Reinforcement Learning: An Introduction*. 2nd ed. The MIT Press.

Tadelis, Steven (2013). "*Game Theory: An Introduction*. 1st ed. Princeton University Press.

Wieting, Marcel and Geza Sapi (2021). „Algorithms in the marketplace: An empirical analysis of automated pricing in e-commerce“. In: *Available at SSRN 3945137*.

Appendix

9

A Pseudo-code for asymmetric information

Please note that the pseudocode for asymmetric information is generalized to only contain the change in the price setting step. All other steps remains the same.

Algorithm 4 Asymmetric information in action-module

Initialize:
    Learning parameters, discount factor, and needed variables and tables for the specific algorithm
    choose random prices $\in \mathcal{P}$ for both firms for $t = \{1, 2\}$
for t=3 to $\mathcal{T}$ **do**
    Update necessary variables, with the true state
    if t % 2=0 **then**
        Observe opponent price, S_t

$$S_t \begin{cases} = p_{j,t-1} & \text{with probability} \quad 1-\mu \\ \sim \mathcal{U}(\mathcal{P}) & \text{with probability} \quad \mu \end{cases}$$

        Choose price $p_{i,t}$ using S_t and set $p_{j,t} = p_{j,t-1}$
    else
        choose price using the true state
    end if
    change agents, $j \leftarrow i$ and $i \leftarrow j$
end for

B Individual profitability, $k = 6$ and $\mu = 0.05$

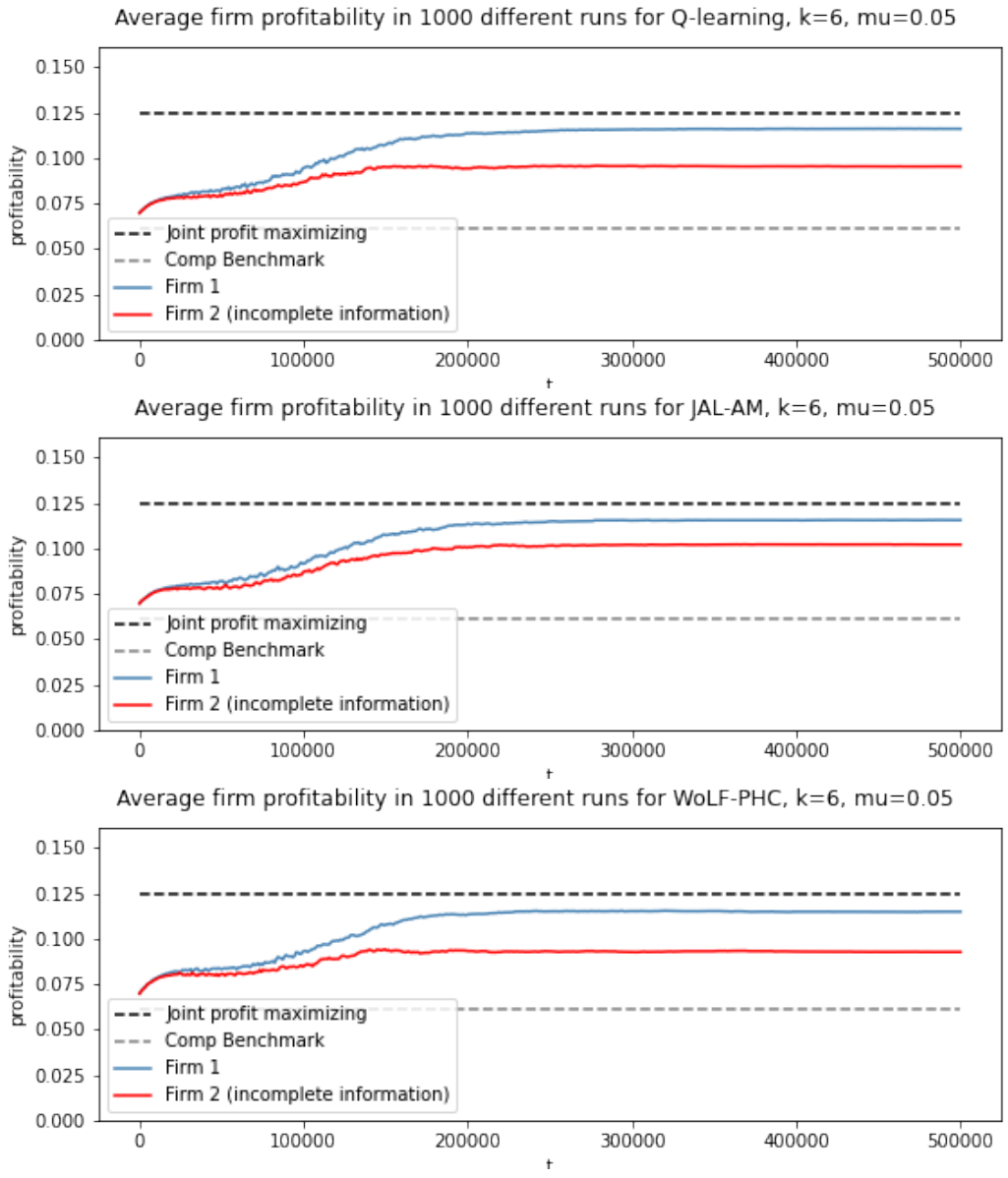


Figure 1: These graphs show the individual profitability when $k = 6$ and $\mu = 0.05$

C Individual profitability, $k = 24$ and $\mu = 0.05$

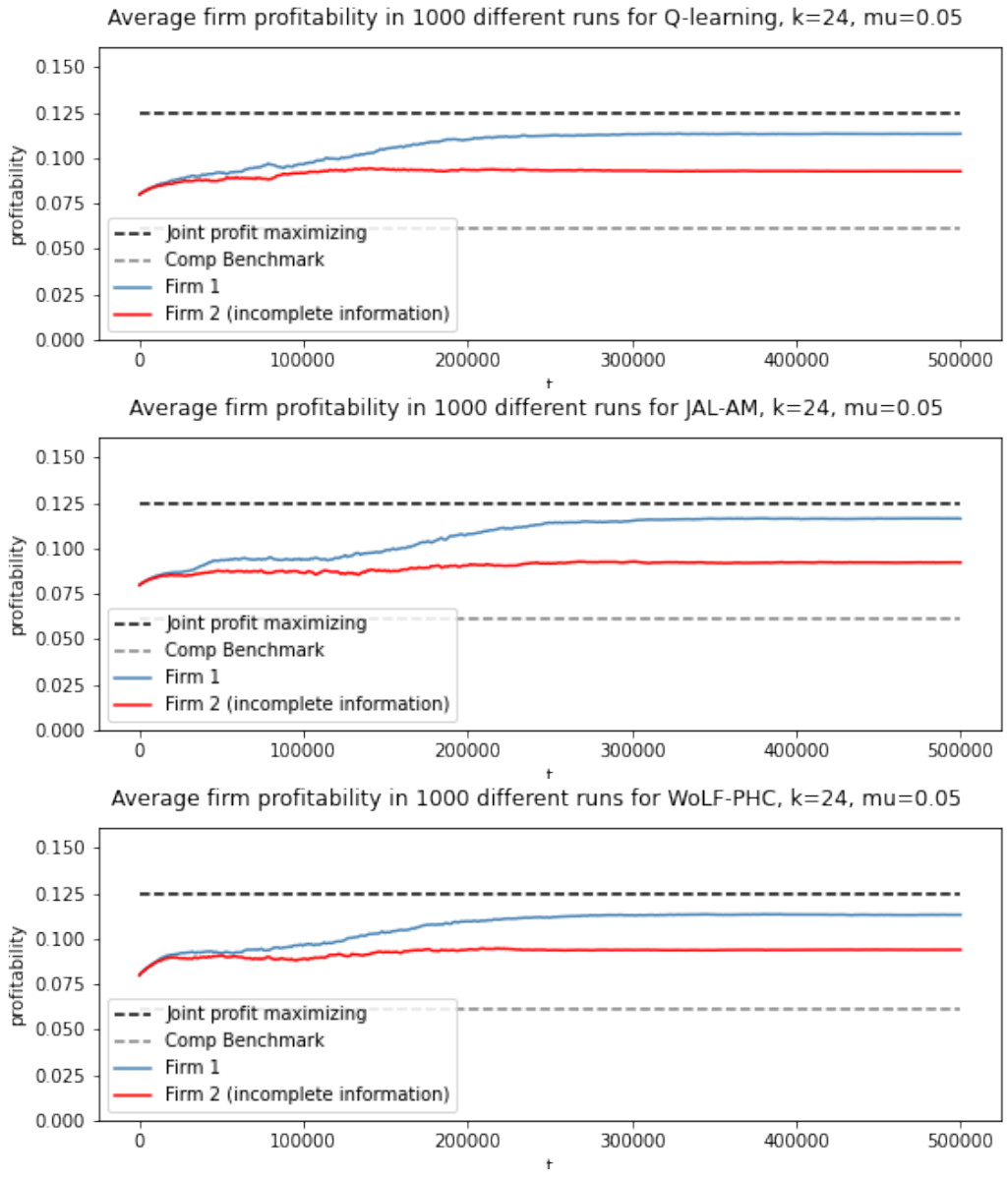


Figure 2: These graphs show the individual profitability when $k = 24$ and $\mu = 0.05$

D Individual profitability, $k = 6$ and $\mu = 0.3$

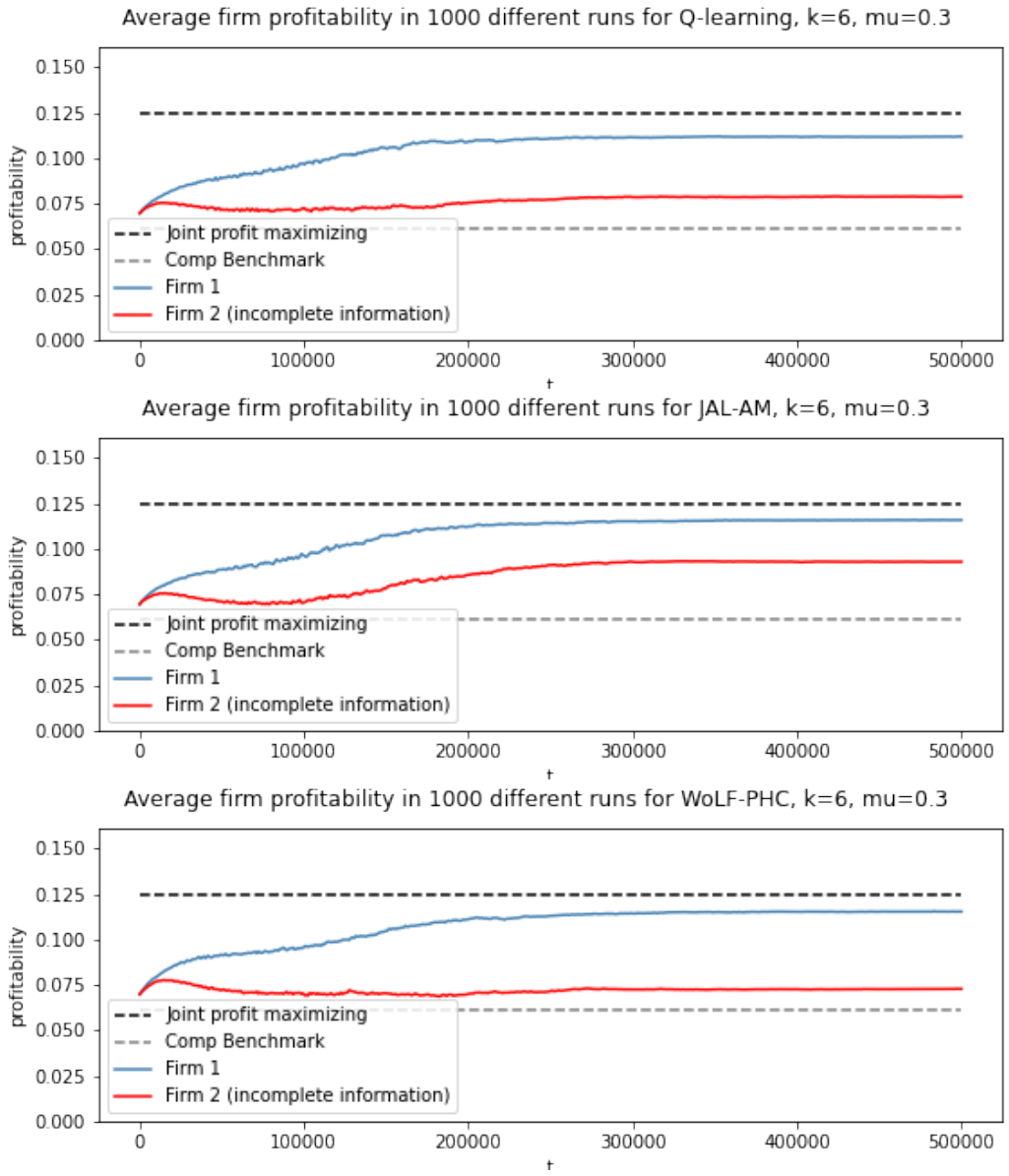


Figure 3: These graphs show the individual profitability when $k = 6$ and $\mu = 0.3$

E Individual profitability, $k = 24$ and $\mu = 0.3$

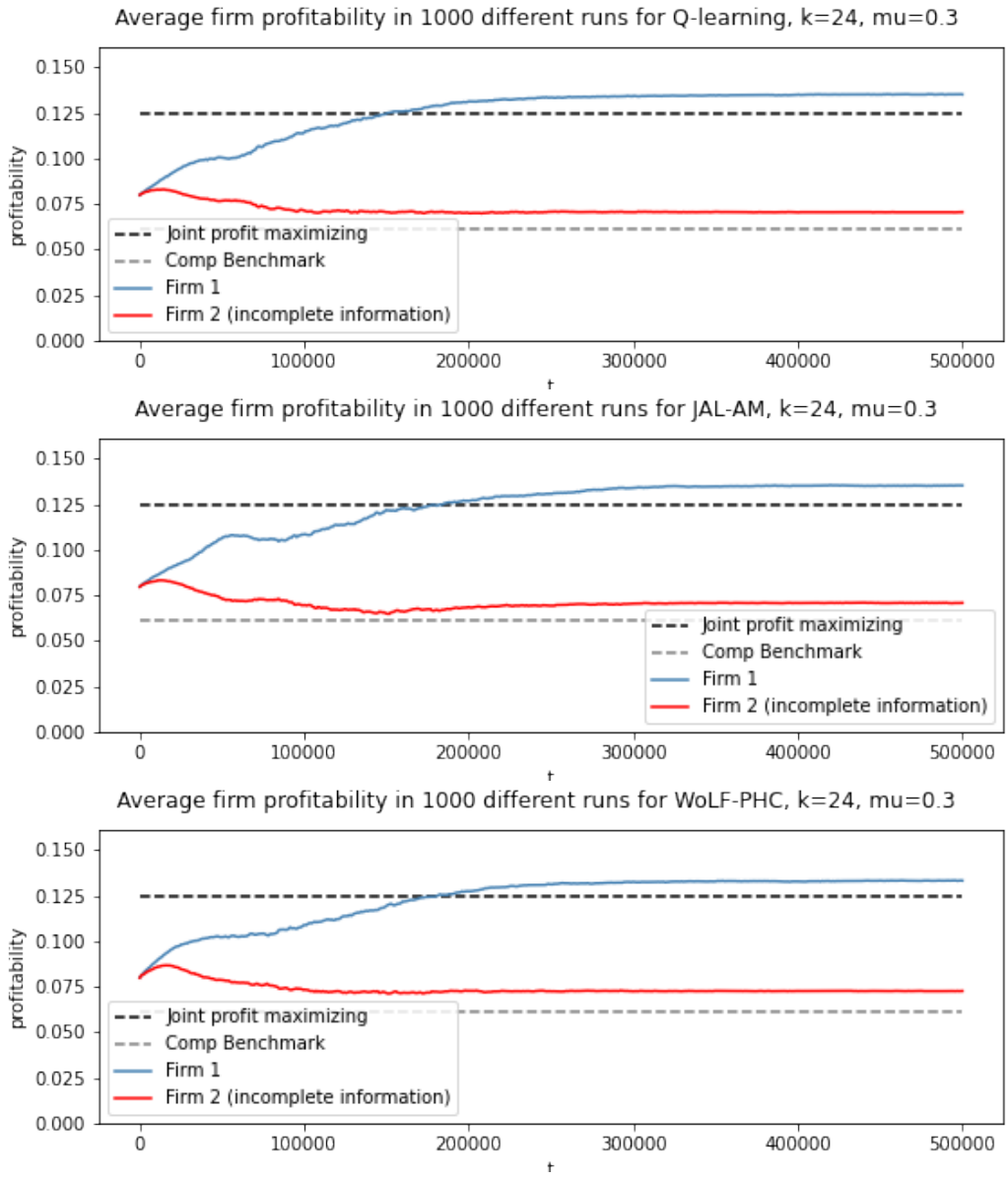


Figure 4: These graphs show the individual profitability when $k = 24$ and $\mu = 0.3$