

Forside

Eksamensinformation

ADØB19001E - Bachelorprojekt i datalogi-økonomi
(vejleder fra SAMF)

Besvarelsen afleveres af

Christian Hempel
fst669@alumni.ku.dk

Sofus Krogh Eriksen
bzk131@alumni.ku.dk

Daan Ten Voorde
xcj453@alumni.ku.dk

Eksamensadministratorer

Bente Andersen
bad@samf.ku.dk
☎ +4535323541

Bedømmere

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

Bent Jesper Christensen
Censor
bjchristensen@econ.au.dk
☎ +4529645180

Besvarelsesinformationer

Titel: Exploring Multi-Agent Competition Dynamics: A Simulation Study on Reinforcement Learning and Collusion

Titel, engelsk: Exploring Multi-Agent Competition Dynamics: A Simulation Study on Reinforcement Learning and Collusion

Antal tegn: 75679

Antal normalsider: 31,5

Tro og love-erklæring: Ja

Indeholder besvarelsen fortroligt materiale: Nej

Må besvarelsen gøres til genstand for udlån: Ja



BSc in Computer Science and Economics

Bachelor's thesis

**Exploring Multi-Agent Competition Dynamics: A Simulation Study
on Reinforcement Learning and Collusion**

Daan ten Voorde, Christian Hempel & Sofus Krogh Eriksen

Supervised by Anders Munk-Nielsen

12th of June 2023



Daan ten Voorde, Christian Hempel & Sofus Krogh Eriksen

Bachelor's thesis

BSc in Computer Science and Economics, 12th of June 2023

Supervisors: Anders Munk-Nielsen

Character count: 75,679

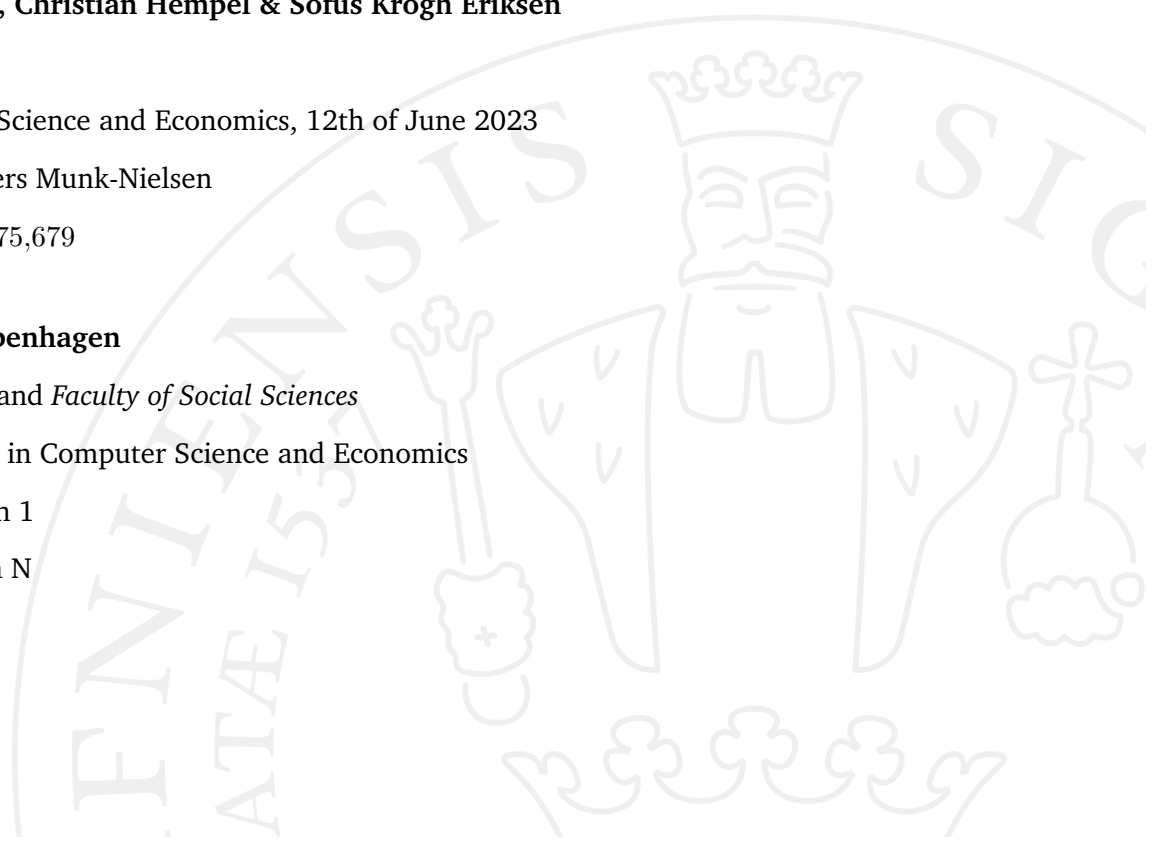
University of Copenhagen

Faculty of Science and Faculty of Social Sciences

Bachelor's Degree in Computer Science and Economics

Universitetsparken 1

2200 Copenhagen N



Abstract

Algorithmic price-setting has in recent years become more prevalent. Contemporary literature has shown that reinforcement learning algorithms are capable of colluding, leading to supra-competitive prices. This has created new problems for competition authorities, as the old policy is no longer sufficient for dealing with algorithmic collusion since no illegal agreements between humans have been made. We investigate what happens when reinforcement learning algorithms compete in a sequential game with a discrete set of prices. We replicate the results from other studies that show collusion between two players. By adding a third player, we expand upon the two-player game and demonstrate how in both cases, the players shift their pricing strategies as the price grid becomes finer. Finally, we show that there is still some form of price coordination occurring between the three players. However, we argue that it no longer constitutes collusion under our definition.

Contents

1	Introduction	1
2	Theory	3
2.1	Dynamic competition	3
2.1.1	Sequential games	3
2.1.2	Nash Equilibrium	4
2.1.3	Bertrand competition	6
2.1.4	Sequential pricing oligopoly	7
2.1.4.1	Monopoly	9
2.1.5	Collusion	10
2.1.5.1	Tacit collusion	10
2.2	Reinforcement-learning	11
2.2.1	The value-function	12
2.2.2	Q-learning introduction	15
2.2.2.1	Q-learning setup	15
2.2.2.2	Learning module	17
2.2.2.3	Action module	19
2.2.3	Q-learning with more players	19
2.2.4	Q-learning pricing algorithm setup	19
2.3	Performance metrics	20
2.3.1	Profitability	20
2.3.2	Average profit gain	21
2.3.3	Benchmarks	21
3	Implementation and optimization	23
3.1	Practical limitations	23

3.2	Optimization	24
3.2.1	Numba speedup	24
4	Results	26
4.1	Performance	26
4.1.1	Profitability	26
4.1.2	Average profit gain	28
4.2	Strategy outcomes	30
4.3	Forced deviation	35
4.4	Convergence	37
5	Discussion	39
5.1	Results summary	39
5.1.1	Performance	39
5.1.2	Strategy outcomes	41
5.1.3	Forced deviation	42
5.1.4	Convergence	42
5.2	Assumptions	42
5.3	Further improvements	43
6	Conclusion	45
7	Bibliography	47

Introduction

In recent years pricing algorithms have become more important in the way firms set their prices, hence affecting market dynamics and influencing outcomes for both firms and consumers (DCCA, 2021, p. 2). Algorithmic price cooperation has been shown to be possible in recent economic literature (Klein, 2021). This raises various concerns for competition authorities as algorithmic collusion can result in welfare loss for consumers and like classic tacit collusion, it can be hard to detect (Porter, 2005, ch. 1). Klein (2021) showed the collusive behavior of pricing algorithms in a competitive sequential duopoly environment. However, not much research has been done on the algorithmic behavior of **including more than two agents**.

Therefore, understanding how the addition of more players in a sequential game affects competition is a blind spot in the literature that warrants further investigation. We construct a sequential pricing game with a discrete price grid and use a reinforcement learning method called Q-learning. The Q-learners will act as agents in the game and will be set up to maximize their own (discounted) profit by choosing prices from the grid. By employing Q-learners as the strategic agents in our simulations, we create a dynamic environment that mirrors real-world competitive settings. We then investigate the performance of these Q-learners using a number of metrics in order to answer the question: *“What consequences does the addition of more agents have for the presence of algorithmic collusion?”*

Our examination encompasses various aspects, including firms’ profitability, average profit gains, pricing strategies, and the presence of possible collusion. **We replicate** the findings from Klein (2021, p. 548) that show the two Q-learners approaching joint-maximizing profits. They **punish deviation** from their strategies, indicating collusive behavior. We then ascertain that the introduction of **an extra player reduces profits**

by a substantial margin. Adding a third player, we find that the agents stop punishing deviation from collusive strategies, although they still display supra-competitive profits.

This outcome highlights that further complicating the environment by increasing the number of players leads to a decrease in coordinating behavior. Furthermore, sometimes an agent has completely unprofitable runs, suggesting that one of the Q-learners fails to converge to a working strategy. In general, agents play two types of strategies; fixed price strategies and price cycling strategies. We find that agents tend to prefer price cycles over fixed price strategies for finer price grids, while the opposite is found for coarser price grids.

Since Q-learning only updates one entry in the Q-table at a time, it runs quite slowly, especially in an interpreted language such as Python. We must take performance into consideration, as the code could take many hours or even days to run. We use a Python module called Numba, which implements a Just-In-Time (JIT) compiler, letting the code run within a reasonable time frame (Numba, 2023).

The outline of the paper is as follows. Section 2 provides a theoretical foundation for our study, exploring concepts such as sequential games, multiple Nash Equilibria, and the role of Q-learners. We delve into the relevant economic background to establish a solid theoretical framework for our analysis. Section 3 focuses on the practical implementation of our simulation, the practical limitations and the optimization techniques employed to ensure reliable and efficient results. Section 4 will present the results which section 5 will discuss in further detail. Finally section 6 will conclude the paper.

The code of this project can be found in the following GitHub repository:

<https://github.com/daanario/bachelor/>

Theory

In this section we present the theoretical foundations that underlie our paper. We cover the setting in which our algorithms operate as well as the equilibria that exist within this environment. Specifically, we provide a definition of a sequential Bertrand game and examine collusion and its feasibility. We delve into the specifics of tacit collusion, offering a detailed explanation.

2.1 Dynamic competition

From game theory we know that a dynamic game is a game where players can act repeatedly for a given time (Munk-Nielsen, 2022a, p. 16). When working with price-setting algorithms it therefore makes sense to use dynamic games for the competitive framework. This is a little studentious

2.1.1 Sequential games

A key element in the construction of our competitive environment is the definition of when the agents take an action. The agents in the game operate in discrete time periods. Assuming that a fixed number of actions are taken per period, it doesn't matter how long the period is, as the period could always be multiplied by some factor to represent another time unit. What's important is that between these periods no actions change. As the number of actions is constant in the period, the number of actions and their consequences (e.g. discounted profits) will scale linearly with the periods.

These time periods are dependent on the markets that they represent. In some markets, firms update their actions very frequently, while in other markets firms are constrained. For example in the Danish medicine market, there are legal constraints forcing firms to only update prices on predetermined days (Danish Medicine Agency, 2019).

In our game we use sequential instead of simultaneous competition because Klein (2021, p. 543) argues a sequential setting is a more natural reflection of algorithmic price competition than a simultaneous setting. Assuming continuous time it is extremely unlikely, if not impossible, that firms would update their prices at the exact same time. Therefore, it can be argued that sequential price-setting is a better reflection of the markets we choose to emulate since we want to see firms react to each other's prices consecutively in order to gain a competitive advantage.

weird way of saying it

2.1.2 Nash Equilibrium

A Nash equilibrium is a set of strategies for which all players choose a best response to the strategies of all other players. Thus, in a Nash equilibrium no player is able to obtain a higher payoff/reward by choosing any other strategy. Formally, the Nash equilibrium is defined in Tadelis (2013, ch. 5.1).

Definition 2.1.1 (Nash equilibrium) *The pure-strategy profile $s^* = (s_1^*, s_2^*, \dots, s_n^*) \in S$ is a Nash equilibrium if s_i^* is a best response to s_{-i}^* for all $i \in N$*

As our sequential game is extensive,^{form} the concept of history comes into play, hence it makes sense to define the sub-game. The sub-games represent smaller components of the extensive game and can be seen as individual games with their own equilibria. Hence each smaller game can now contain a Nash equilibrium. The sub-game-perfect Nash equilibrium (SPNE) is a refined Nash equilibrium that requires the players' strategies to be a Nash-equilibrium in all sub-games. Formally, the SPNE is defined in Tadelis (2013, ch. 8.2).

Definition 2.1.2 *Let Γ be an n -player extensive-form game. A behavioral strategy profile $\sigma^* = (\sigma_1^*, \sigma_2^*, \dots, \sigma_n^*)$ 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 .*

There is another refinement of the Nash Equilibrium that is of relevance to our study of this sequential game. Like Klein (2021, p. 544), we use the notion of a Markov Perfect Equilibrium (MPE) as introduced by Maskin and Tirole (1988). The MPE is a refinement of the subgame-perfect Nash equilibrium where the Markov assumption is imposed. The Markov assumption means that the agents' strategies only depend on variables that are directly payoff-relevant (Klein, 2021, p. 544).

It is also relevant to introduce the Folk theorem, as it tells us that payoffs need not be equally divided among the players. This is somewhat counterintuitive, as humans would want to divide the payoffs somewhat equally if the effort was equally shared (e.g. see the Dictator game (Engel, 2011, p. 583-585)). The formal definition of the Folk Theorem is given in Tadelis (2013, p. 211):

Theorem 2.1.1 *Let G be a finite, simultaneous-move game of complete information, let (v_1^*, \dots, v_n^*) denote the payoffs from a Nash equilibrium of G , and let (v_1, \dots, v_n) be a feasible payoff of G . If $v_i > v_i^*, \forall i \in N$, and if δ is sufficiently close to 1, then there exists a subgame-perfect equilibrium of the infinitely repeated game $G(\delta)$ that achieves an average payoff arbitrarily close to (v_1, \dots, v_n) .*

Although the theorem states that G is a simultaneous game, this is slightly awkward we choose to apply the same intuition to sequential games. Since we impose the Markov assumption, the subgame-perfect equilibria described in the theorem are also MPEs in our case. The theorem has an important implication that we will use in our paper; firms can divide profits unequally and still be on an equilibrium path, thus in an MPE.

2.1.3 Bertrand competition

The following section is based on Tadelis (2013, ch. 5.2.4). To establish a competitive setting for the algorithms, it is necessary to specify the type of competition in which they will participate. We have chosen to base the competition on a variation of the Bertrand game because they compete on prices. A Bertrand game with two players is a simultaneous game where the two firms produce the same good and compete in price. In this game, the prices are chosen from a set of prices $p_i \in S_i = [0; 1]$ and the marginal costs of the firms are $c \in [0; 1)$. The consumer will always pick the lowest price as products are homogeneous. Hence the firm with the lowest price takes the whole market, and in case the firms choose the same price they share the market. We then have the following demand function for firm i .

$$D_i(p_i, p_j) = \begin{cases} 1 - p_i & \text{if } p_i < p_j \\ 0.5(1 - p_i) & \text{if } p_i = p_j \\ 0 & \text{if } p_i > p_j \end{cases} \quad (2.1)$$

The profit is then defined as:

$$\pi_i(p_i, p_j) = (p_i - c)D_i(p_i, p_j) \quad (2.2)$$

The following proof is based on Munk-Nielsen (2022b, p. 34-35). In a simultaneous Bertrand game with two players the unique Nash equilibrium can be proven to be $(p_1, p_2) = (c, c)$. To prove this it makes sense to show that (p_1, p_2) for $p_1, p_2 > c$ is not a Nash equilibrium.

Consider (p_1, p_2) to be a Nash equilibrium with either $p_1 > c$ or $p_2 > c$. Lets assume a situation where $p_1 > p_2 > c$, then the strategy $\tilde{p}_1 = p_2 - \epsilon$ would always ^{wrong word} dominate for all $\epsilon > 0$. Player 2 would then do the same until $(p_1, p_2) = (c, c)$, hence there will be no Nash-equilibrium for $p_1 > c$ or $p_2 > c$. As prices can never be lower than marginal

cost we now just need to prove that $(p_1, p_2) = (c, c)$ is a Nash equilibrium. This is true because if $p_2 = c$ then player 1 will be indifferent between all price strategies as payoff is zero no matter what and the same goes for player 2. Hence $(p_1, p_2) = (c, c)$ is a unique Nash equilibrium.

2.1.4 Sequential pricing oligopoly

Motivated by the discussion in section 2.1.1 we choose a sequential variation of the Bertrand game with time periods t . In each period, one firm sets its price, after which it is stuck with that price until after the other firm has chosen its price in the next period. Prices are defined the same way as in Klein (2021, p. 543); as a discrete set of prices $p_i \in S_i \subset [0; 1]$, where the prices are evenly spaced in k intervals such that $p_i \in \left\{0, \frac{1}{k}, \frac{2}{k}, \dots, 1\right\}$. Since we want firms to be able to set the monopoly price (see section 2.1.4.1), k must be an even number. For n players we can generalize the demand function from (2.1) in a sequential setting. Let ς be the ~~set~~^{vector?} of prices, $\varsigma = \{p_{it}, p_{jt}, \dots, p_{nt}\}$ that ^{subscripts!!!} ~~firms~~^{aargh} i, j, \dots, n set in period t . Let p be a price in ς , $p \in \varsigma$. Let \mathcal{P} be the set of prices that are equal to p_{it} , $\mathcal{P} = \{p \in \varsigma : p = p_{it}\}$. We then have the generalized demand function for n players.

$$D_i(p_{it}, p_{jt}, \dots, p_{nt}) = \begin{cases} \frac{1}{|\mathcal{P}|} (1 - p_{it}) & \text{if } \forall p \in \varsigma : p_{it} \leq p \\ 0 & \text{if } \exists p \in \varsigma : p < p_{it} \end{cases} \quad (2.3)$$

Where $|\mathcal{P}|$ is the number of elements in \mathcal{P} .

The sequentiality of the game means players are always setting their prices in the same order consecutively. One could argue that this is not perfectly realistic as firms in reality might not need or have the capacity to be changing their prices every single period. However, the sequential nature of the game allows firms to react to one another and update their pricing in order to earn as much profit as possible over time. Therefore,

like in Klein (2021, p. 543) each firm's objective is to maximize the discounted sum of future profits with discount factor $\delta \in [0, 1)$, ~~such that:~~

$$\max \sum_{s=0}^{\infty} \delta^s \pi_i(p_{i,t+s}, p_{j_t+s}, \dots p_{n,t+s}) \quad (2.4)$$

Like Klein (2021, p. 543), we also ~~depart from the original Bertrand game by~~ assuming no marginal or fixed costs. This approach gives us the following simple profit function.

$$\pi_i(p_{it}, p_{jt}) = p_{it} D_i(p_{it}, p_{jt}) \quad (2.5)$$

Since we have a discrete grid of prices, this means we can no longer assume that the strategy $\tilde{p}_1 = p_2 - \epsilon$ would dominate, because ϵ can no longer be an infinitesimally small value, hence the proof of the unique Nash-equilibrium breaks. We therefore need a new analysis of the Nash-equilibria in this game. As an example, we have the stage game with the price grid for $k = 6$. From the figure 2.1 we can see player 1's

		Player 1 Player 2						
	0 / 6	1 / 6	2 / 6	3 / 6	4 / 6	5 / 6	6 / 6	
0 / 6	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	
1 / 6	(0.0, 0.0)	(0.069, 0.069)	(0.13, 0.0)	(0.13, 0.0)	(0.13, 0.0)	(0.13, 0.0)	(0.13, 0.0)	
2 / 6	(0.0, 0.0)	(0.0, 0.13)	(0.11, 0.11)	(0.22, 0.0)	(0.22, 0.0)	(0.22, 0.0)	(0.22, 0.0)	
3 / 6	(0.0, 0.0)	(0.0, 0.13)	(0.0, 0.22)	(0.125, 0.125)	(0.25, 0.0)	(0.25, 0.0)	(0.25, 0.0)	
4 / 6	(0.0, 0.0)	(0.0, 0.13)	(0.0, 0.22)	(0.0, 0.25)	(0.11, 0.11)	(0.22, 0.0)	(0.22, 0.0)	
5 / 6	(0.0, 0.0)	(0.0, 0.13)	(0.0, 0.22)	(0.0, 0.25)	(0.0, 0.22)	(0.069, 0.069)	(0.0, 0.0)	
6 / 6	(0.0, 0.0)	(0.0, 0.13)	(0.0, 0.22)	(0.0, 0.25)	(0.0, 0.22)	(0.0, 0.13)	(0.0, 0.0)	

Figure 2.1: Nash equilibrium $k = 6$ (discrete price grid)

best responses to player 2's prices as the orange marks, and the opposite for player 2 with the green marks. Hence when a discrete price grid is implemented in our sequential Bertrand game we now have two Nash-equilibria. One being the lowest price in the price grid not equal to zero, and the second being the MC which is zero. The above-zero NE can be generalized for all k , which we will show in section 2.2.1.

The timing of when a player should play a certain price becomes important because of the sequential nature of the game.

The timing depends on the total amount of players participating and their behavior. For two players each agent gets to set a price every other period, hence each player only has to consider their price “locked” for one period. With three players their prices are “locked” for two periods and so on. This introduces a greater element of complexity as the number of firms rises. Hence players will have to adapt their strategies in order to maximize the sum of their discounted profits. They need to account for the periods where they are stuck with their previously chosen price.

2.1.4.1 Monopoly

Previously we described the demand function for a Bertrand game shown in (2.1). A feature of this demand function is the monopoly price, which is the price that maximizes the profit for a monopolistic firm in the Bertrand game. It is also the optimal price for multiple firms to coordinate at.

For a monopoly, the demand function would be expressed as $1 - p$ since it is the only firm operating in the market. To find the monopoly price, we can calculate the price that maximizes profits.

$$\pi = p(1 - p) = p - p^2 \quad (2.6)$$

We solve for the FOC with respect to the price.

$$1 - 2p = 0 \iff p = \frac{1}{2} \quad (2.7)$$

As stated before, this will be the optimal price for the players to coordinate prices at. This does not change with the number of firms in the market, ^{imprecise} because demand adapts in response to the price in the Bertrand game (Tadelis, 2013, p. 90). In the Bertrand game, colluding firms would have to share the profit among more firms, diminishing the individual firm’s profit. ^{hmm, splitting it}

2.1.5 Collusion

When presenting collusion, it is important to state that there is no singular definition of the term. According to Varian (2014, p. 541), **collusion occurs when two or more agents agree to work together to establish a price that maximizes their joint profit.**

As demonstrated in section 2.1.3, when the price interval is continuous the static Bertrand game's equilibrium price is equivalent to the marginal cost. Opting to set the price ~~equivalent~~ to the marginal cost results in zero profit for the firms. **Firms only find it beneficial to establish a higher price if all other competitors follow suit.** do so Otherwise, they will be left with no profit in a Bertrand game. Therefore, firms that agree to cooperate will “artificially” raise the price, generating greater profit than if they acted alone. This action is problematic, particularly if consumers are charged supra-competitive prices, causing a welfare loss in the market (Varian, 2014, p. 467). Hence it is in the interest of competition authorities to detect and stop collusive agreements.

According to Calvano *et al.* (2018, p. 27), one of the main components of collusive behavior is the possibility of reward-punishment. This means that a firm will be rewarded for sticking to the ~~san~~collusive strategy, by gaining a higher profit over time. On the other hand, if a firm chooses not to collude, but to deviate, it will receive some kind of punishment. The punishment will vary from strategy to strategy, but will always consist of at least one of the opposing firms lowering their price. If one of the non-deviating opposing firms undercuts the deviator's price, the deviating firm will lose profit. In this paper, collusion will also be assumed to contain reward-punishment as a key characteristic.

2.1.5.1 Tacit collusion

Because **collusion is illegal**, participating firms have a vested interest in keeping it confidential (DCCA, 2020). To prevent this, governing bodies have established

competition committees and authorities to monitor and maintain well-functioning markets, enforcing the competition act and preventing collusion, such as the Danish Competition and Consumer Authority (DCCA).

Tacit collusion, as opposed to explicit collusion, occurs ^{to do what?} **when firms cooperate** without an explicit agreement, (Ivaldi *et al.*, 2003, p. 4). Competition authorities struggle to sanction the involved parties as it is hard to obtain clear evidence of collusion. According to Ivaldi *et al.* (2003, p. 7) tacit collusion can be difficult to achieve in practice among people since per definition, there is no explicit communication between the parties. As it is also a credible threat for firms to set the NE price, firms have to weigh the discounted payoffs from collusion over time higher than the payoffs they get from competing and playing the NE, in order to uphold the tacit collusive agreement (Tadelis, 2013, ch. 10.4).

As will be discussed further in later sections, **there are 2 types of collusive strategies that are relevant to our paper; fixed-price strategies where firms cooperate at a fixed price, and price cycling strategies, where firms choose to cycle their prices at supra-competitive levels.**

2.2 Reinforcement-learning

In the following section, we will introduce Q-learning and reinforcement learning, which are important for understanding our algorithmic approach. The theory in the following section is based on Sutton and Barto (2015, ch. 1.1).

Reinforcement learning both describes a problem type, as well as the class of methods for solving these types of problems. To condense a reinforcement learning problem to its purest form, three essential aspects must be present. First, a sense of environment. There has to be a closed-loop environment where the agent can take action in the current situation or “state” of the environment and receive feedback from actions.

Second, the ability to take actions that affect the state, which is a representation of the environment that the agent is currently in. Based on the reward from the state of the environment, the agent is able to take action that can influence the state of the environment.

Third, a goal or reward relating to the past or present state of the environment needs to be implemented. The agent will then “learn” to take actions that maximize the reward over time. If a method can solve the type of problem described above, it can be characterized as a reinforcement learning method. This is exactly the case with the Q-learning method.

2.2.1 The value-function

When working with reinforcement learning, one needs to define the problem. The problem we want to solve with our particular Q-learning approach is the so-called value-function condition problem. Recall the Markov assumption from section 2.1.2. For two agents, that means they cannot condition on any other information than the previous competitor price $p_{j,t-1}$. Thus, there can be no communication between the two agents with the exception that they can observe each other’s previous price.

We start with a general framework of the situation with two firms. The firms can react to each other’s prices with reaction functions $R_i(p_{j,t-1})$, where the firms set their prices in their own turn: $p_{it} = R_i(p_{j,t-1})$ (Klein, 2021, p. 544). $R_i(\cdot)$ is closely related to $V_i(\cdot)$. It is the best response to the opponent price(s) such that we maximize the value function $V_i(\cdot)$. Hence $R_i(\cdot)$ is the argument p which maximizes the value-function. We follow the argument from Klein (2021, p. 544) in that for all prices along the equilibrium path, if the pair of strategies (R_1, R_2) is to be a Nash equilibrium, the value-function condition below must hold.

$$V_i(p_{jt}) = \max_p \left[\pi_i(p, p_{jt}) + \mathbb{E}_{p_{j,t+1}} \left[\delta \pi_i(p, p_{j,t+1}) + \delta^2 V_i(p_{j,t+1}) \right] \right] \quad (2.8)$$

Note that we do not actually know the competitor response $p_{j,t+1}$ in period t . The value-function therefore takes the expectation $\mathbb{E}_{p_{j,t+1}}$, which is firm i 's expectation of the competitor response $p_{j,t+1}$, conditioning on firm j 's response to our price, $R_j(p)$ (Klein, 2021, p. 544). ???

This framework can be expanded to the situation with more than two players. In a situation with three players, the strategy triple (R_1, R_2, R_3) is a Nash equilibrium if the value-function condition (2.9) below holds for all prices along the equilibrium path.

$$V_i(p_{jt}, p_{kt}) = \max_p \left[\pi_i(p, p_{jt}, p_{kt}) + \mathbb{E}_{p_{j,t+1}, p_{k,t+2}} \left[\delta \pi_i(p, p_{j,t+1}, p_{kt}) + \delta^2 \pi_i(p, p_{j,t+1}, p_{k,t+2}) + \delta^3 V_i(p_{j,t+1}, p_{k,t+2}) \right] \right] \quad (2.9)$$

Like the two-player value-function, this value-function takes the conditional expectations to upcoming opponent prices, $\mathbb{E}_{p_{j,t+1}, p_{k,t+2}}$, this time with respect to the ~~distributions of~~ reaction functions $R_k(p, p_{j,t+1})$ and $R_j(p, p_{k,t+2})$. Again, here $R_i(p_{j,t+1}, p_{k,t+2})$ gives firm i 's maximizing choice of p such that $V_i(\cdot)$ is maximized.

There is a clear parallel between the value-function conditions and the Q-functions that we will introduce in the following sections. From these value-function conditions, it is not yet clear how we compute the expectations for $p_{j,t+1}$ and $p_{j,t+1}, p_{k,t+2}$, respectively. The approximations to the distributions of these competitor prices (states) are shown later in section 2.2.2.2. As the Q-learning algorithm converges toward a solution to the value-function condition problem, the expectations of the states become more accurate. Recall the definition of an MPE from section 2.1.2. A strategy pair or triple is an MPE if condition (2.8) or (2.9) holds, respectively, for all firms and all prices. Maskin and Tirole (1988) identify two sets of Markov Perfect Equilibria: fixed-price strategy equilibria (also called focal price equilibria) and price cycle equilibria in the case of a repeated Bertrand game. These two types of equilibria will be explored further in the results section.

As discussed in section 2.1.4, the game has a static Nash equilibrium with positive payoffs, which we will call the non-zero static Nash equilibrium. We demonstrated an example where $1/6$ was a static NE for $k = 6$. The non-zero static NE $1/k$ can be shown to be general for all k . The following is based on the argument in Klein (2021, p. 13) that shows that $1/k$ is a static NE for two players. We adapted it to fit our three-player setup. To show that the price $1/k$ is a Nash equilibrium for all k we start by assuming that $R_2(p_1, p_3) = 1/k$ and $R_3(p_1, p_2) = 1/k$. That means both firm 2 and 3 will always set a price of one price point higher than zero marginal cost. Inserting the fixed prices into the value function seen in (2.9), we get:

$$V_1\left(\frac{1}{k}, \frac{1}{k}\right) = \max_p \left[\pi_1\left(p, \frac{1}{k}, \frac{1}{k}\right) + \mathbb{E}_{p_{j,t+1}, p_{k,t+2}} \left[\delta \pi_1\left(p, \frac{1}{k}, \frac{1}{k}\right) + \delta^2 \pi_1\left(p, \frac{1}{k}, \frac{1}{k}\right) + \delta^3 V_1\left(\frac{1}{k}, \frac{1}{k}\right) \right] \right] \quad (2.10)$$

Since $p_{j,t+1}$ and $p_{k,t+2}$ are fixed at $1/k$, the equation simplifies to:

$$V_1\left(\frac{1}{k}, \frac{1}{k}\right) = \max_p \left[\pi_1\left(p, \frac{1}{k}, \frac{1}{k}\right) + \delta \pi_1\left(p, \frac{1}{k}, \frac{1}{k}\right) + \delta^2 \pi_1\left(p, \frac{1}{k}, \frac{1}{k}\right) + \delta^3 V_1\left(\frac{1}{k}, \frac{1}{k}\right) \right] \quad (2.11)$$

continuation value doesn't depend on p...

Gathering the common terms and isolating for V_1 :

$$V_1\left(\frac{1}{k}, \frac{1}{k}\right) = \max_p \left[(1 + \delta + \delta^2) \pi_1\left(p, \frac{1}{k}, \frac{1}{k}\right) + \delta^3 V_1\left(\frac{1}{k}, \frac{1}{k}\right) \right] \quad (2.12)$$

$$\iff V_1\left(\frac{1}{k}, \frac{1}{k}\right) = \frac{(1 + \delta + \delta^2)}{(1 - \delta^3)} \max_p \left[\pi_1\left(p, \frac{1}{k}, \frac{1}{k}\right) \right] \quad (2.13)$$

The maximizing choice of firm 1 is then similarly $R_1(p_2, p_3) = \frac{1}{k}$, which is by symmetry a Nash equilibrium.

2.2.2 Q-learning introduction

Q-learning is a reinforcement-learning algorithm that was developed by Chris Watkins in 1989 (Sutton and Barto, 2015, p. 21). Being a reinforcement-learning algorithm means that the algorithm is not told what to do but instead, given the specific situation, must find the actions that give the biggest reward. This is done by simulating the problem many times with different actions, and learning what actions give the greatest reward. Hence our algorithm will not be told to collude, but instead told to produce the highest profit over time.

In order for the agent to take the best actions it will first need to explore what actions give the greatest reward. At some point, the agent should stop exploring and start exploiting the most rewarding actions. The dilemma of when to switch from exploration to exploitation is an important one to consider, as too little exploration can result in the Q-learner not learning what actions fulfill the reward potential (Assad *et al.*, 2020, p. 10). Relative to exploitation, too much exploration can lead to redundant exploration; the Q-learner might repeatedly keep exploring states which yield little to no reward, while it could have increased its rewards by exploiting instead. Realistically, if our algorithms were deployed in a practical setting, they would be trained first with a large number of periods before seeing practical use.

2.2.2.1 Q-learning setup

We have based the setup and parameter values on the work in Klein (2021, ch. 3). It is however uncertain whether these values are also optimal for the three-player game. As we have a sequential game we obtain profit from every round whenever a firm sets a price. When calculating the new estimate of ^{not defined}the Q-value, the profits of future periods are used in the calculation. The future profits are however more uncertain because we draw from the opponents' price distribution to "guess" their price for future rounds. The firms discount the importance of future profits using parameter $\delta \in [0, 1)$. The discount factor represents the degree to which firms value future payoffs compared

to immediate payoffs. A higher value of δ heightens the importance of future profits. When implementing multiple players this becomes important as distant periods are harder to predict than closer ones. We use $\delta = 0.95$.

We also use a parameter called α which represents the learning rate. It is the parameter that determines how important the new estimate is versus the original Q-value estimate. The Q-value is updated as follows.

$$Q \leftarrow (1 - \alpha) \cdot \text{previous estimate} + \alpha \cdot \text{new estimate} \quad (2.14)$$

Hence $\alpha \in (0, 1)$ is a parameter that determines how fast old information should be replaced with new. We use $\alpha = 0.3$. because?

During the simulation of the game, the players should explore less and exploit more over time. As they have already explored random possibilities, they should choose more strategies that maximize their Q-values. Therefore we implement ε so that exploration decreases from 100% at the first iteration to 0.1% at half the iterations, to 0.00001% at termination. Hence ε can be written as:

$$\varepsilon = (1 - \theta)^t \quad (2.15)$$

Where t is the iteration and where the decay parameter θ is expressed as:

$$\theta = - \left(\frac{1}{1000000} \right)^{\frac{1}{T}} + 1 \quad (2.16)$$

Where T is the total number of iterations.

already introduced

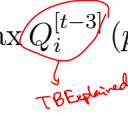
We are denoting k as the fineness of our price grid. The price range between 0 and 1 is divided into k -spaced intervals. For instance, when $k = 6$, the resulting price intervals would be $\{0, 1/6, 2/6, \dots, 1\}$. We use different values for k to showcase different outcomes with both 2 and 3 players. Since firms need to be able to set the monopoly price $p = 0.5$, k must be an even number.

2.2.2.2 Learning module

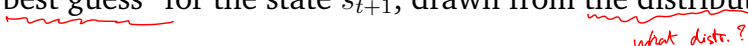
In order to discover the best possible actions, Q-learning has a learning module that updates values in the Q-table given a function. The Q-table is an n -dimensional matrix where n represents the number of players in the game. Hence for each combination of prices that competing firms set, we can look up a Q-value for each of our own prices.

The function that updates values in the Q-table is called the Q-function. Over time, we want the approximations from this function to converge to the optimal value-functions shown in (2.8) and (2.9). Depending on the environment, the Q-function is modified to describe the specific problem the algorithm needs to solve. We have used the implementation in Klein (2021, p. 545) of the Q-function for two players as a foundation for our three-player implementation shown below.

$$Q_i^{[t]}(p_{it}, r_t, s_t) \leftarrow (1 - \alpha) \cdot Q_i^{[t-3]}(p_{it}, r_t, s_t) + \alpha \cdot (\pi(p_{it}, r_t, s_t) + \delta \pi(p_{it}, s_{t+1}, r_t) + \delta^2 \pi(p_{it}, s_{t+1}, r_{t+2}) + \delta^3 \max_p Q_i^{[t-3]}(p, s_{t+1}, r_{t+2})) \quad (2.17)$$



We see that the Q-value being updated is the Q-value that corresponds to the price we set in period t and the states, which are the other players' prices in period t . The new Q-value is calculated using (2.14). The new estimate is calculated by adding four components; first the profit in period t given the price set by the player and the competitors' last prices. Second the profit in period $t + 1$ holding p_{it} and r_t constant and making the "best guess" for the state s_{t+1} , drawn from the distribution. Third the



profit in period $t + 2$ holding p_{it} constant, using the same draw of s_{t+1} , and drawing the state r_{t+2} from the distribution. The draws are from the following distributions:

$$\begin{aligned} \mathbf{s}_{t+1} & \begin{cases} \sim U\{P\} & \text{with probability } \varepsilon_t \\ = \operatorname{argmax}_s Q_j^{[t-2]}(s, r_{t-1}, p_t) & \text{with probability } 1 - \varepsilon_t \end{cases} \\ \mathbf{r}_{t+2} & \begin{cases} \sim U\{P\} & \text{with probability } \varepsilon_t \\ = \operatorname{argmax}_r Q_k^{[t-1]}(r, p_t, s_{t+1}) & \text{with probability } 1 - \varepsilon_t \end{cases} \end{aligned} \quad (2.18)$$

Therefore, player i assumes it knows its opponents' Q-tables and imitates how it would set its price like in the action module, shown in section 2.2.2.3. This way player i can more optimally grade the decision it is about to take, by calculating a Q-value that takes the current draw of the future states into account. The fourth element of the new estimate is the maximum Q-value for player i at s_{t+1} and r_{t+2} .

	t				
Firm i	p_{t-2}	p_{t-1}	p_t	p_{t+1}	p_{t+2}
Firm j	s_{t-2}	s_{t-1}	s_t	s_{t+1}	s_{t+2}
Firm k	r_{t-2}	r_{t-1}	r_t	r_{t+1}	r_{t+2}

Figure 2.2: Sequence of own prices and states from firm i 's perspective

To illustrate this state-drawing method more clearly, we here show figure 2.2 for the sequence of prices and states for any firm i in any range of periods $t > 3$. Recall that each firm can only change their price once every three periods. This means that for example, r_t and r_{t+1} are identical, as are s_{t-1} and s_t . In this case, r is always the state given by firm k 's response, and s is always the state given by firm j 's response. p_t is firm i 's price in period t . Firm i will make a draw on the states marked in bold in conjunction with its learning process as shown in (2.18).

2.2.2.3 Action module

In the action module, we need to decide whether to exploit or explore. Klein (2021, p. 545) handles this balance of exploration versus exploitation by picking actions in the following way:

$$p_{it} \begin{cases} \sim U\{P\} & \text{with probability } \varepsilon_t \\ = \operatorname{argmax}_p Q_i^{[t]}(p, s_t, r_t) & \text{with probability } 1 - \varepsilon_t \end{cases} \quad (2.19)$$

same as above

Where $\varepsilon_t \in [0, 1]$ is calculated as explained in 2.2.2.1. Here $U\{P\}$ is a discrete uniform distribution of the possible prices. Hence the first case picks a random price from the possible prices with probability ε_t . The random price choice means the algorithm will explore different prices and save the outcome of choosing this price as a Q-value in the Q-table. In the other case the algorithm will exploit, hence choosing the price that corresponds to the maximum Q-value at the specific state. Through exploration, the Q-learners' goal is to have learned optimal responses to the opponents' prices.

2.2.3 Q-learning with more players

mesh Mathematically the algorithm could easily be expanded to include more players. The states of the added players will need to be drawn in the same fashion as (2.18) and the Q-function (2.17) should be extended with the profits of the future rounds where the new players set their prices. However, the algorithm could quickly suffer from the curse of dimensionality, as the inclusion of every new player adds another dimension to the Q-table which we discuss in section 3.1.

2.2.4 Q-learning pricing algorithm setup

We have based the algorithmic setup of our three player sequential Q-learning on the pseudocode of the two player implementation in Klein (2021, p. 545). The pseudocode

provided shows how our simulation works. Here p_{1t}, p_{2t} and p_{3t} represent the price arrays that will keep track of each firm's prices throughout the simulation.

Pseudocode Sequential 3 player Q-Learning (Simulation)	
1	Set demand and learning parameters α, δ ; Initialize Q-functions
2	Initialize $\{p_{1t}, p_{2t}, p_{3t}\}$ for $t = \{1, 2, 3\}$ randomly
3	Initialize $t = 3, i = 1, j = 2$ and $k = 3$
4	Loop over each period
5	Update $Q_i(p_{i,t-3}, p_{j,t-3}, p_{k,t-3})$ according to (2.17)
6	Set p_{it} according to (2.19) and set $p_{jt} = p_{j,t-1}, p_{kt} = p_{k,t-1}$
7	Update $t \leftarrow t + 1$ and $\{i \leftarrow j, j \leftarrow k, k \leftarrow i\}$
8	Until $t = T$ (specified number of periods)

2.3 Performance metrics

The following section will introduce the performance metrics we have used in this paper to measure the performance of our algorithms.

2.3.1 Profitability

Profitability is a measure of profit for each firm. The profitability is the joint average profit of the firms in the sequential game. We take the average profit of the last 1,000 profits that each firm has set and calculate the profitability from these values. We can compute this with a maximum fineness of once every 1,000 periods, however we can also compute it less often. For example, we could do this every 50,000 periods hence giving us 10 measures of profitability if we run 500,000 iterations. As more companies are added to the game, the per-firm profitability will fall as profit is shared among more firms. The profitability measure from Klein (2021, p. 546) can be generalized for n players to:

$$\text{Profitability: } \Pi_i \doteq \frac{1}{1,000} \sum_{t=T-1,000}^T \pi_i(p_{it}, p_{jt}, \dots, p_{nt}) \quad (2.20)$$

2.3.2 Average profit gain

To determine how much of the monopoly profit the firms gain we use a measure called the average profit gain. Like Calvano *et al.* (2018) we derive this measure by taking the optimal joint profit that the firms could make if colluding and dividing it with the actual joint profit. This will give us a value in $[0; 1]$ where 0 is perfect competition and 1 is full collusion at the monopoly price. The measure is taken from Calvano *et al.* (2018, p. 19), and is denoted by:

$$\Delta \equiv \frac{\bar{\pi} - \pi^N}{\pi^M - \pi^N} \quad (2.21)$$

$\bar{\pi}$ is the actual per firm joint profitability, π^M is the optimal per firm joint profit that the firms could make if perfectly colluding (they set the monopoly price), and π^N is the per firm profit in the Bertrand-Nash static equilibrium (Calvano *et al.*, 2018, p. 19). Like Karlsen and Wismann (2022, p. 17) we choose this to be the static Bertrand-Nash equilibrium for perfect competition, such that $\pi^N = 0$ since marginal costs are zero. We can therefore simplify Δ to:

$$\Delta \equiv \frac{\bar{\pi}}{\pi^M} \quad (2.22)$$

2.3.3 Benchmarks

In this paper we have used different benchmarks to evaluate the performance of our algorithms. We have chosen to evaluate our algorithms with three benchmarks. The “Edgeworth cycle benchmark”, the “static non-zero Nash Equilibrium benchmark” and the “joint profit maximizing benchmark”.

The Edgeworth cycle benchmark for the two-player game is taken from Klein (2021, p. 546). Although he denotes it as a “competitive benchmark”, we don’t think the name is suitable as this benchmark is actually an Edgeworth price cycle where firms undercut each other by one price point down to marginal cost, after which they immediately

cycle up to one price point higher than the monopoly price. Hence we will call this benchmark the “EC benchmark”. For n players they cycle up to $n - 1$ price points higher than the monopoly price. The generalized grid point that the upcycling firm cycles up to is then:

$$\frac{\frac{k}{2} + n - 1}{k} \quad (2.23)$$

Where k denotes the number of intervals in the price grid. The benchmark used is the per firm, per period (t) profit in the Edgeworth Cycle. We calculate this benchmark by taking the average profit for each firm over the cycle. With the generalized expression for the upcycle price point shown in (2.23), we can explain what happens in the game with three players; the upcycling firm sets the price to two price points higher than the monopoly price, since otherwise, if it cycled up to one point above the monopoly price, the monopoly profit is not obtained by any of the firms, since one firm would still take the whole market at $p = MC$ as per the timing of the firms’ turns.

Another benchmark that makes sense to use is the lowest above-zero price point Nash-equilibrium found in section 2.1.3. In order to calculate the benchmark we compute the per-firm profit for this price. The benchmark makes sense to implement as it is the lowest Nash equilibrium where they still make a profit. We will call this benchmark the “static non-zero NE”.

At last, we define the joint profit maximization benchmark as the maximum profit the firms would be able to make. One of the ways whereby they could obtain this profit would be for the firms to set the monopoly price each round. We calculate the benchmark by computing the per-firm profit at this price.

Implementation and optimization

This section will discuss the implementation and optimization of our simulations. Furthermore, we will discuss the practical limitations of simulating a large amount of runs with varying computational complexity. We will examine the running time implications of adding another dimension to the Q-learner. Lastly, we will comment on the computational speedup we experienced by using a Just-In-Time (JIT) compiler.

3.1 Practical limitations

When implementing the algorithms presented in section 2.2.4, we made Python modules `twoplayers.py` and `threeplayers.py`, which are utilized in a Jupyter Notebook file called `main.ipynb`. We quickly realized that running the simulations was very time-consuming.

The more dimensions are added, the more operations are needed and the worse the running time gets. As Koenig and Simmons (1993) show, the asymptotic running time of Q-learning algorithms can range from being exponential in the state space n to running in $O(n^3)$, depending on the implementation. This is to say that Q-learning algorithms generally scale poorly, and thus running them for many periods in many different simulations is sure to take a long time on all but the most powerful hardware, especially as the state space grows with the addition of more players.

Handwritten note: n is n or k the most relevant?

We have been limited by our hardware and the fact that everything has been run on our own machines, with the best one having 16 GB of RAM and a Ryzen 3700U CPU.

When running multiple independent simulations, one might consider parallelization. This involves leveraging multiple threads of a CPU to execute calculations concurrently and independently of each other. From our experience, parallelization was hard to implement in Python. One of the issues that we realized was that it is only possible to parallelize over N , the number of runs, in our implementation. We concluded that it would not be worth our efforts, considering the time it would take to parallelize our code, although it would be a natural next step in the process.

→ But $N=1000$?

3.2 Optimization

We have used the C-written Python module called Numba (2023). Numba produces machine optimized code from our Python code. The module works best with code that consists of Numpy arrays, loops and functions, meaning that it is a great fit for our code. The functions that we deem slow or time demanding we decorated with the `@jit` decorator, which utilizes the Numba JIT-compiler in “nopython” mode, guaranteeing that no Python code will be run (Numba, 2023). When a call is made to a function decorated with `@jit`, the function is compiled into machine code “just-in-time” (JIT) for execution. As a result, a portion of our code can subsequently run at the speed of native machine code, significantly enhancing its performance.

3.2.1 Numba speedup

With the implementation of Numba we experienced a significant speedup. The speedup was especially noticeable in our three-player implementation. Running 1,000 simulations with 500,000 periods the three-player implementation without JIT compilation took 12+ hours to run, which is far from ideal. With the JIT compilation the running time fell to approximately 30 minutes, which is still a considerable amount of time. After the code was run the first time, the running time was even lower at approximately 25 minutes, since certain portions of the JIT-compiled code are saved in a cache (Numba, 2023). For our two-player implementation the difference wasn’t as significant

but definitely still noticeable. The two-player implementation without JIT compilation went from running approximately 280 minutes to 12 minutes with JIT compilation for $N = 1,000$ runs. See table 3.1 for the exact running times on our simulations (these can differ from machine to machine). *What machine?*

N = 1000	With Numba	Without Numba
2 players	11m. & 52 s.	280 m. & 29 s.
3 players	24m. & 35s.	1068 1068 m.

Table 3.1: Running time for sequential price-setting game 2 and 3 players

Results

In this section, we will present the key findings from our work with Q-learners. Our objective has been to investigate Q-learning behavior in the context of the sequential price-setting game for two and three players. We have run various simulations of this game, providing insights into the performance and behavior of the Q-learners. With the use of graphical illustrations, we try to shed light on Q-learners' collusive capabilities and responses to deviations. Additionally, we investigate the robustness of our results.

4.1 Performance

The following section will introduce the results of our performance metrics, average profitability and average profit gain over time presented in section 2.3.

4.1.1 Profitability

The first performance metric that we investigate is profitability, which serves as a key illustration of whether collusive behavior is present or not. From figure 4.1 and 4.2 we see how the profitabilities for the two-player game are robust with respect to k , meaning that the size of k does not affect the profitability outcome.

However the same does not show in the profitabilities for the three-player game. For three players we see in figure 4.3 and 4.4 how the average profitability falls dramatically for $k = 48$ compared to $k = 6$. From our tests, it seems that the Q-learners at $k = 6$ would set a lower price if there were more price points to choose from.

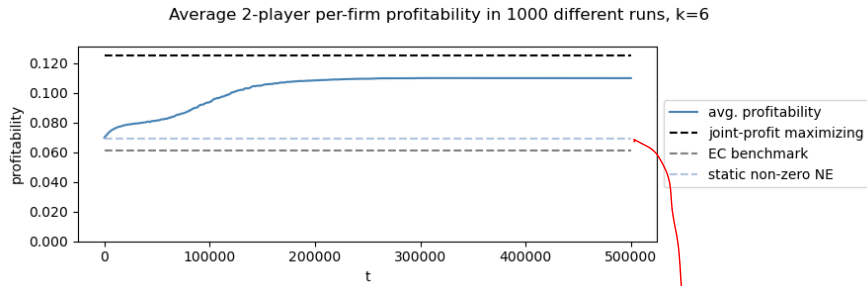


Figure 4.1: Per-firm profitability for the two-player game, $k=6$

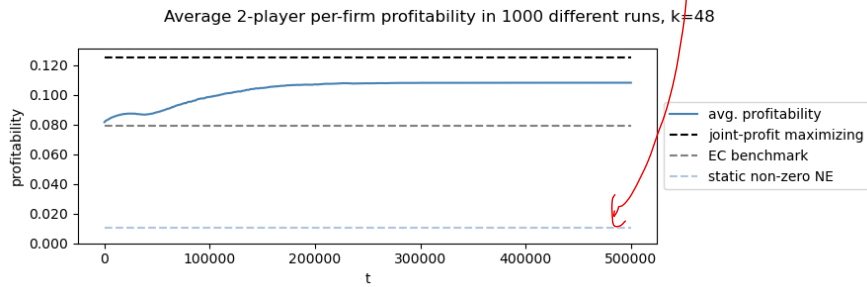


Figure 4.2: Per-firm profitability for the two-player game, $k=48$

Hence the average profitability at $k = 6$ seems to be held up “artificially” by this restriction. When more price points are introduced at $k = 48$, the Q-learners are able to set lower prices thus resulting in a loss of average joint-profitability. The

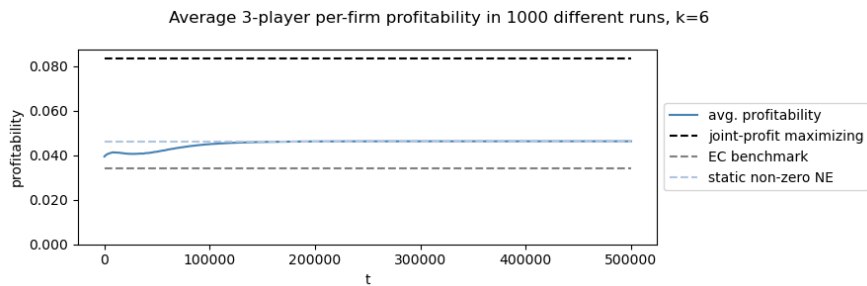


Figure 4.3: Per-firm profitability for the three-player game, $k=6$

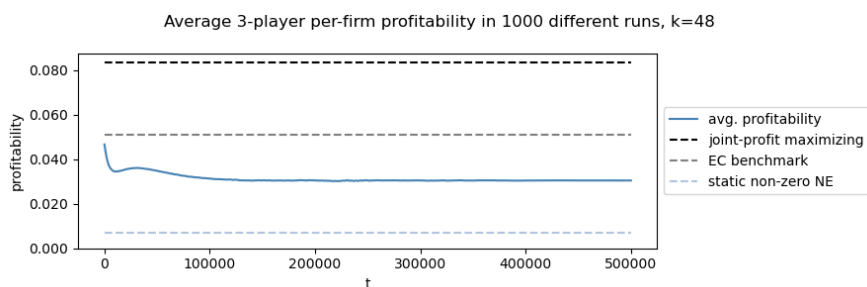


Figure 4.4: Per-firm profitability for the three-player game, $k=48$

drop in profit could be explained by the algorithm now having to take two opponents into account when drawing from opponent price distributions, hence making it more difficult to set optimal prices. While the two-player average profitability surpasses the EC benchmark for both $k = 6$ and $k = 48$, the three-player average profitability only surpasses the static non-zero NE benchmark for $k = 48$, while it converges to this benchmark at $k = 6$. This means that the agents in the two-player game come closer to the joint-maximizing profit compared to the three-player game. The supra-competitive profits in the games could indicate some level of autonomous algorithmic collusion. This idea will be further investigated in the section 4.3 on forced deviation, since a key element of collusion is some sort of reward-punishment strategy (Calvano *et al.*, 2018, p. 27).

4.1.2 Average profit gain

The next performance metric we want to highlight is average profit gain. This metric makes it easier to compare the results from two- and three-player games, since it helps illustrate the profits as percentages of the monopoly profit. The average profit gain distribution of the three-player game in figure 4.5 can, as discussed in section 4.1.1, be explained by the very few available price points, which restricts the Q-learners' price-setting options.

We observe from figures 4.5 and 4.6 that the distribution shifts to higher intervals for the agents in the two-player game while it is shifted lower for the agents in the three-player game. While the result highlights that agents in the two-player game reach a greater level of average profit gain than the agents in the three-player game, it also shows that both games leave agents with supra-competitive profits.

Different values for k affect the profitability and by extension the average profit gain. Therefore we illustrate how the average profit gain changes with respect to k in figure 4.7.



Figure 4.5: Average profit gain over time, $k = 6$

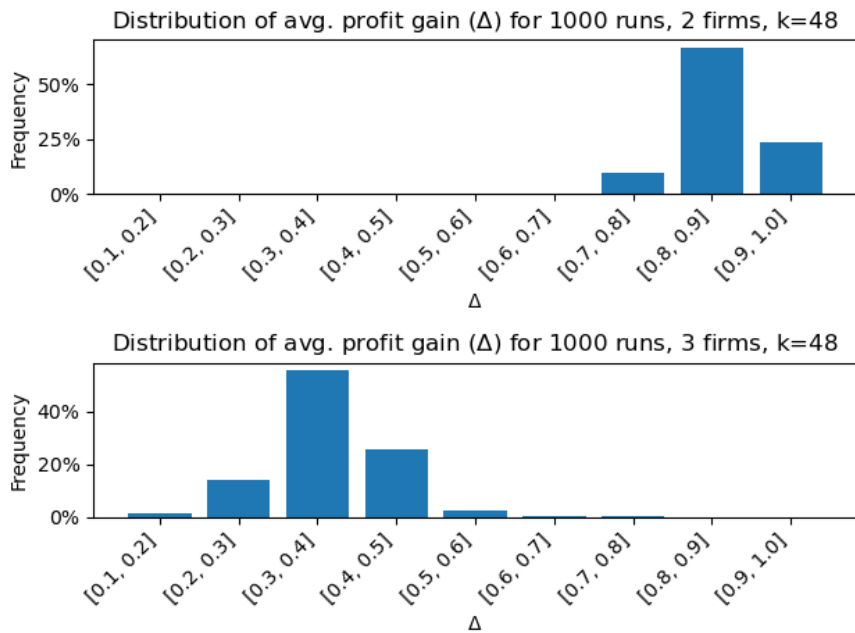


Figure 4.6: Average profit gain over time, $k = 48$

Overall, in the two-player game the average profit gain seems to rise and stabilize as k increases. The opposite seems to be true for the three-player game. In this case, as k increases, so falls the average profit gain. Interestingly, there are noticeable dips in the average profit gain at $k = 12$. We offer a possible explanation for this, at least in the case of the two-player game. Recall the two types of MPE strategies; fixed-price

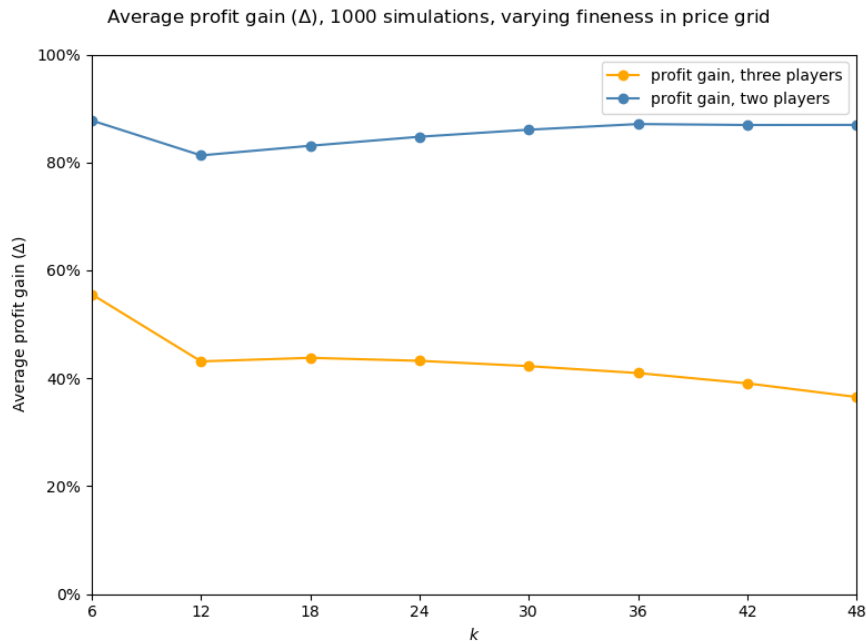


Figure 4.7: Average profit gain over varying k

strategies and price cycling strategies. As the firms move from choosing fixed-price strategies to price cycling strategies (see section 4.2), there is a loss in profit as firms are only able to undercut each other's prices in relatively large steps. In the case of the two-player game, as the price grid becomes finer, they are able to undercut each other with less, thus increasing the length of the more profitable section of a period in the price cycle, resulting in higher profits for both firms. For the three player game, this same intuition does not seem to apply, as there is a consistent fall in average profit gain across all the values for k we tested.

4.2 Strategy outcomes

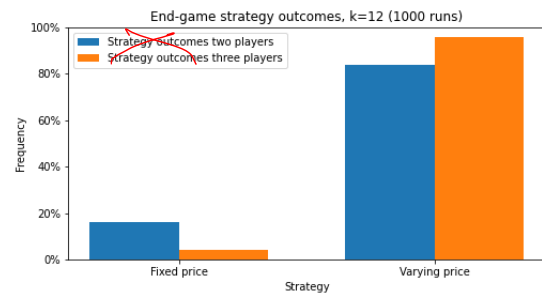
As discussed in section 2.2.1, there are two sets of Markov Perfect Equilibria: fixed-price strategies and price cycles. This section aims to illustrate the frequency of each type and how it changes with certain parameters. Additionally, we provide examples of price cycling strategies.

By computing the variances of the prices in the last 1,000 iterations of the simulation, we can determine whether the firms employ fixed price strategies; if the variances are very close to zero, we know the firms' prices have not changed over time. Otherwise, we know the prices have varied. Afterwards, we can inspect the price histories to determine whether the firms set the prices in a cycling pattern.

The end-game strategy outcome plots seen in figures 4.8 and 4.9 show that coarser price grids in general lead to more fixed-price strategy outcomes. From $k = 6$ to $k = 12$ we see a great decline in the frequency of fixed-price strategy outcomes. The distribution of strategy choices seems to stabilize for values of k between 24 and 48. This suggests that $k = 6$ drives the Q-learners to play fixed-price strategies simply because they don't have enough price options to cycle between. However, the stabilization in strategy choice that we see for $k = 24$ to $k = 48$ shows the observed frequency of strategies the Q-learners will play if they have more than $k = 6$ pricing options. Here we see that for both two and three-player games a varying price strategy is played much more often than a fixed price strategy.

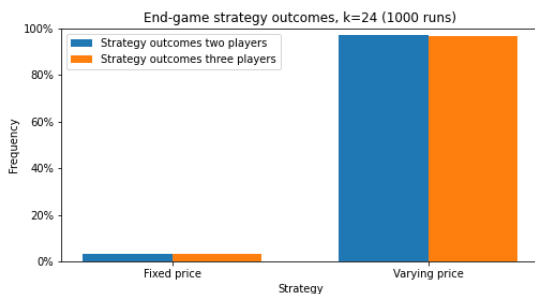


(a) End-game strategy outcomes, $k = 6$

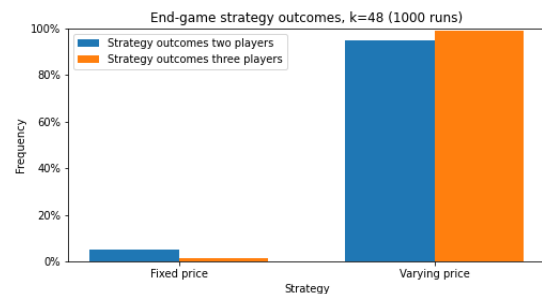


(b) End-game strategy outcomes, $k = 12$

Figure 4.8: Comparison of end-game strategy outcomes for different k values



(a) End-game strategy outcomes, $k = 24$



(b) End-game strategy outcomes, $k = 48$

Figure 4.9: Comparison of end-game strategy outcomes for different k values

The graphs in figures 4.8 and 4.9 also suggest that the two-player game results in more fixed price strategies than for three players. This indicates that it is easier for two players to coordinate prices at a fixed level than it is for three players, at least for the values of k other than $k = 6$.

We have shown that the firms vary their prices in at least some runs. We want to examine these variations more closely to determine whether any form of coordination is taking place. Figure 4.10 and 4.11 show examples of these variations. Note that these are taken from a single run, and there are many variations of these cycles. The following examples cannot be said to be fully representative of all runs.

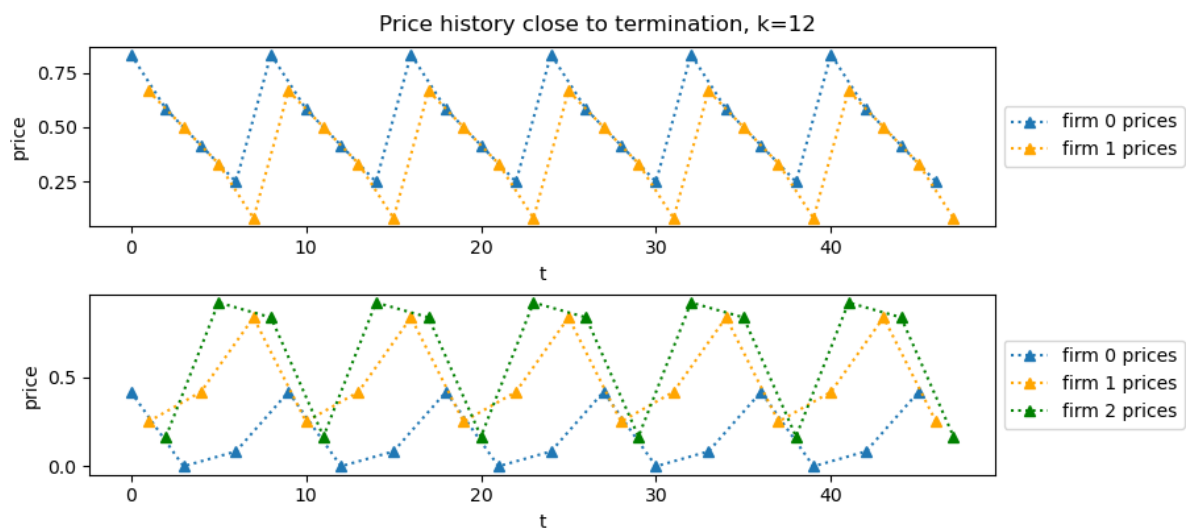


Figure 4.10: Example of price cycling runs, $k=12$

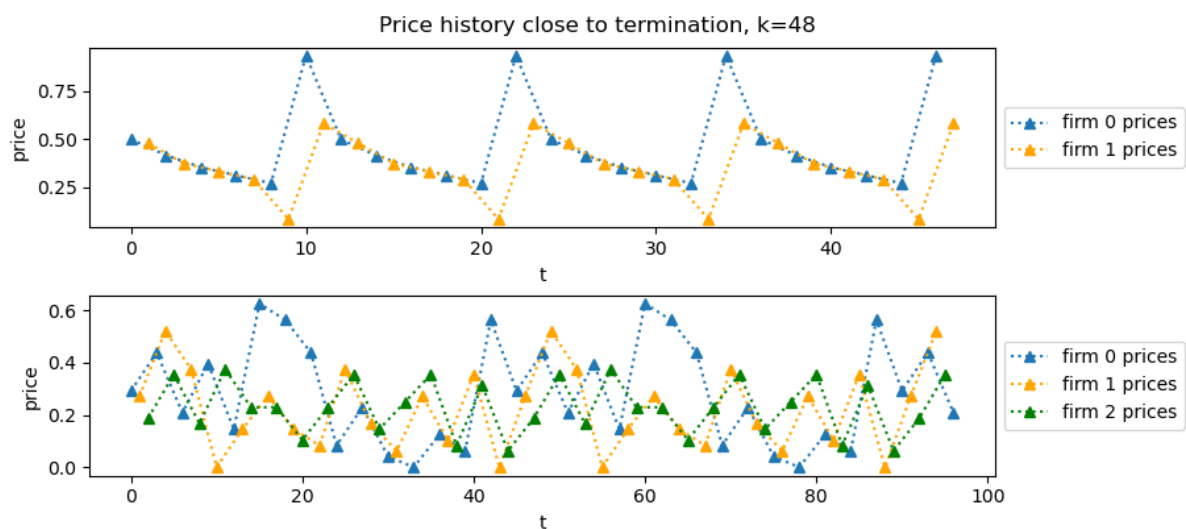


Figure 4.11: Example of price cycling runs, $k=48$

The two-player games from figures 4.10 and 4.11 exhibit clear Edgeworth price cycles; firms continue to undercut each other's prices until they reach a threshold, after which one firm starts the cycle anew by setting a high price. Curiously, it seems to be the same firm always resetting the price cycle in a run. This means that **it is the same firm that has to take the loss of resetting the cycle while the other firm reaps all the benefits.** While intuitively **this might seem unfair,** this is perfectly in line with the Folk Theorem (see section 2.1.2). As long as the payoffs across periods are strictly positive for both firms, they can still be in an MPE.

Let us move on to the three player games. The interesting thing here is that although the three player game seems to exhibit price cycles, they do not seem to result in the same degree of profitability and average profit gain, as seen in the previous results for profitability and average profit gain in section 4.1. Moreover, as k increases, the pattern in the three-player price cycling behaviour becomes increasingly complex. As shown in figure 4.11, we had to inspect 100 periods total to see that prices did not just vary at random; the price histories show periodicity.

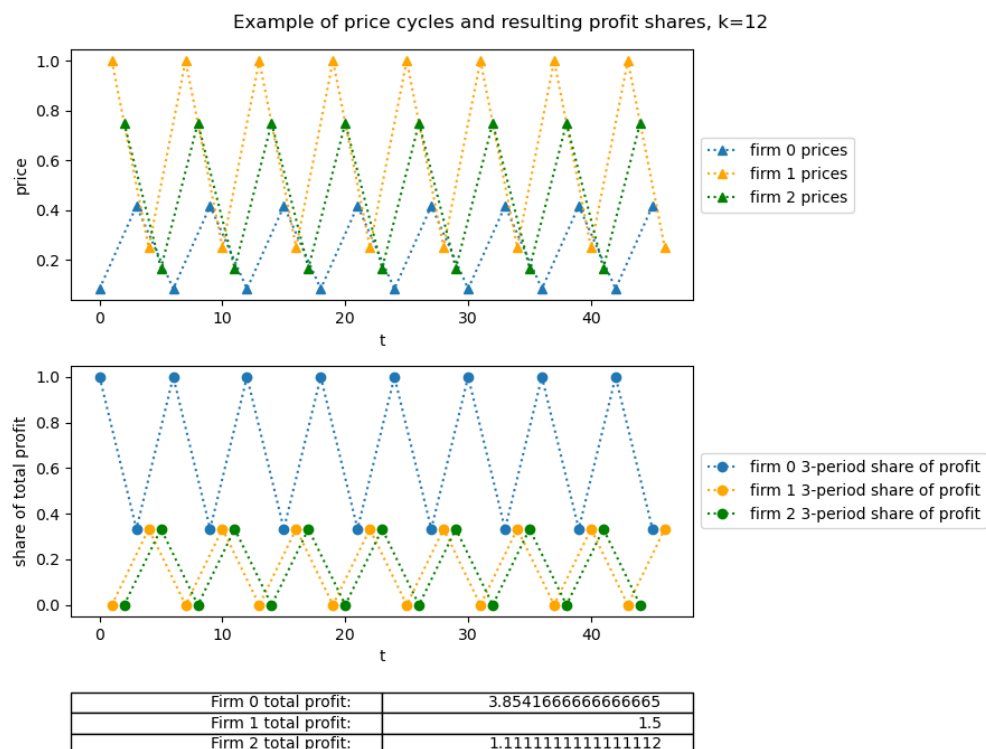


Figure 4.12: Price cycles and the firms' shares of the total profit, $k=12$

The averaging of profitabilities in section 4.1.1 and 4.1.2 shows overall trends, but obscures certain details from individual runs. One of these details is the way the firms divide their profits, which we will explore now. The following graphs show an examples of three-player price cycle strategies and their respective shares of the profit. In the run that figure 4.12 shows, the firms did not share their profits equally. This result is again in line with the Folk Theorem.

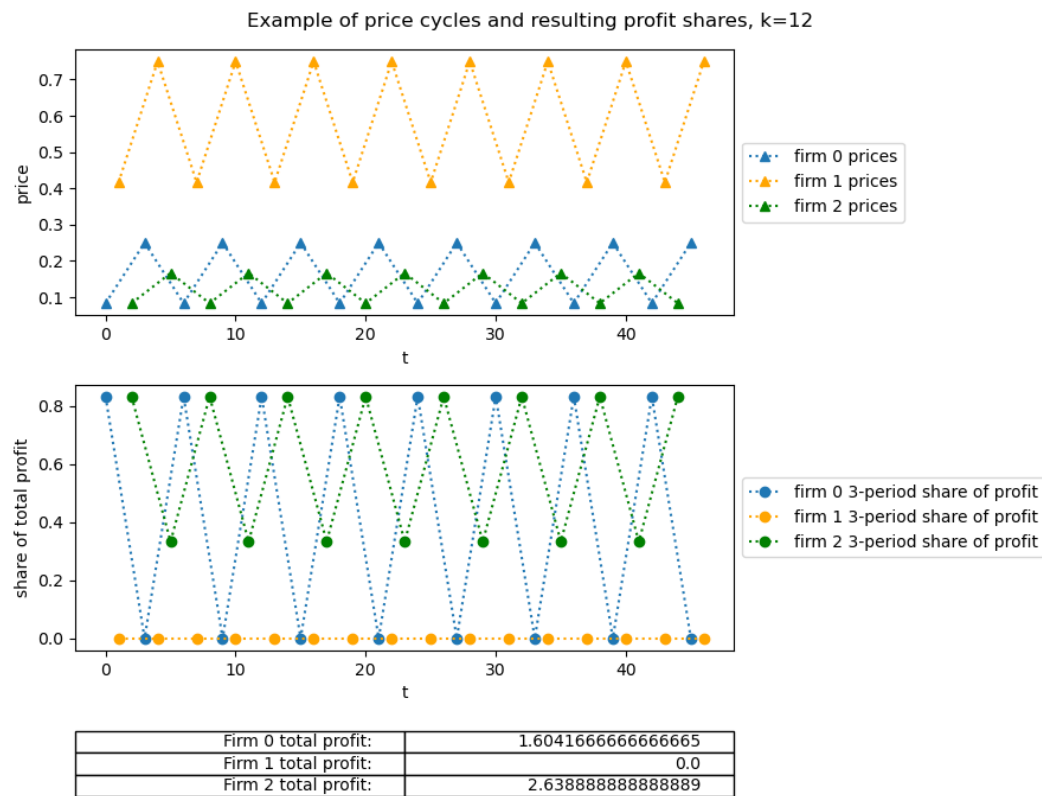


Figure 4.13: Price cycles and the firms' shares of the total profit, $k=12$

We show another example of a run in figure 4.13. Interestingly, one of the firms ends up with zero profit. This would mean that the firms are not in an MPE, as the zero-profit firm could change its strategy (for example setting $p = 1/k$) and gain an above-zero payoff. We wanted to explore this further. Therefore, we show here the number of occurrences of this phenomenon for different values for T .

In figure 4.14 we see that 4.5% of the runs at $T = 500,000$ resulted in one of the firms getting zero profit. This frequency decreases as T gets larger, ending up at 1.9% for $T = 10,000,000$. This could suggest that when the Q-learners are granted more time to learn, they have an easier time converging to a solution.

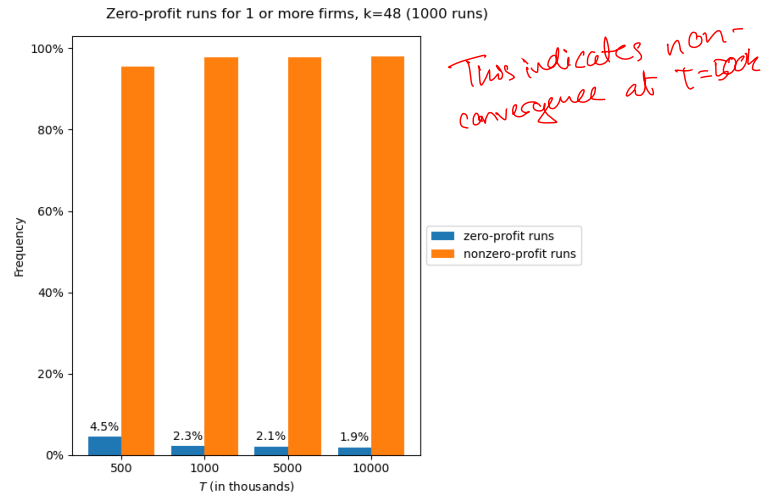


Figure 4.14: Occurrences of zero profit runs, $k=48$

4.3 Forced deviation

The following section will cover what happens when we force one Q-learner to deviate from the focal price and undercut the competitor. The section is based on the work of Calvano *et al.* (2018) and Klein (2021). A point of investigation is to check whether the deviating firm actually increased its sum of profits by deviating. We will investigate what happens to both the price and the profit of each firm in a two- and three-player environment. k has a dramatic effect on the amount of price cycling strategies and fixed price strategies, which we observed in section 4.2. We will only focus on the average of fixed price strategy runs, since averaging over price cycles makes it very difficult to see whether there was a reaction to the deviation. Changes in price might not be a reaction to a deviation; it could be part of the average of the price cycle as it is.

We take the 30-period profit, averaged across runs[?], for the deviating firm pre-deviation and do the same in the interval of t shown in figure 4.15. For this set of runs, the

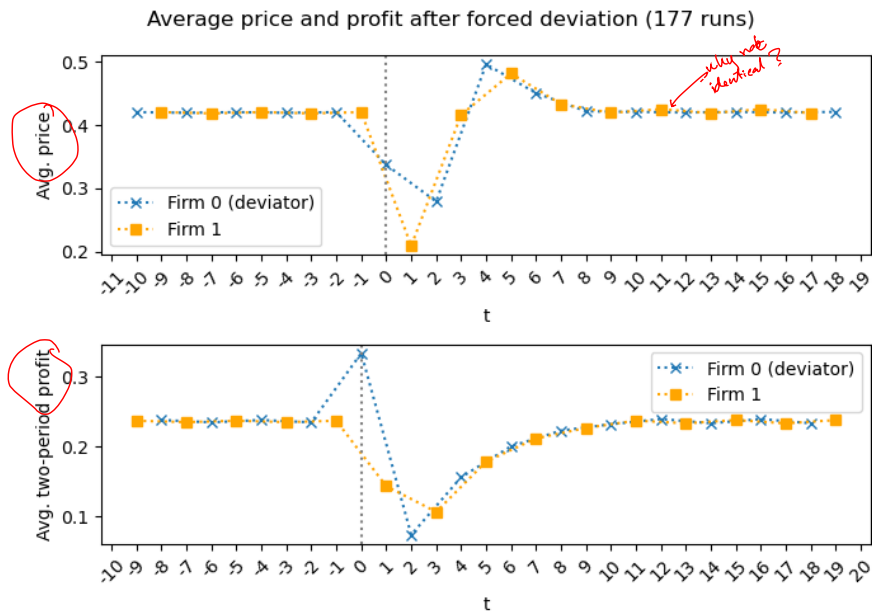


Figure 4.15: Forced deviation in the 2-player game, $k=12$

average 30-period profit pre-deviation is 3.5719. The 30-period profit spanned in the bottom graph seen in figure 4.15 is 3.1112. We therefore see that in this case, playing a collusive strategy provides better total payoffs than deviating from said strategy. Furthermore it can be seen from the price plot in figure 4.15 that the deviating firm gets punished by the non-deviating firm with a price undercut. The punishment is temporary, and the firms return to the fixed-price strategy after the punishment.

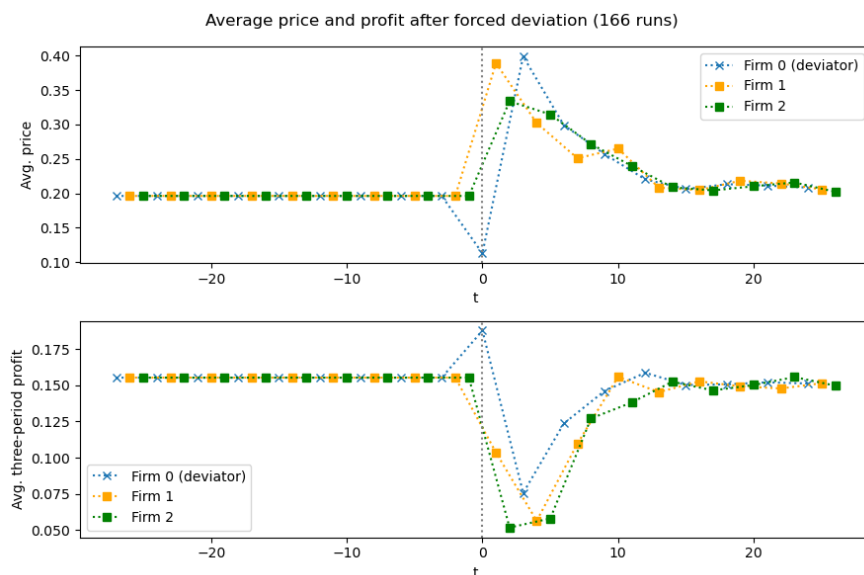


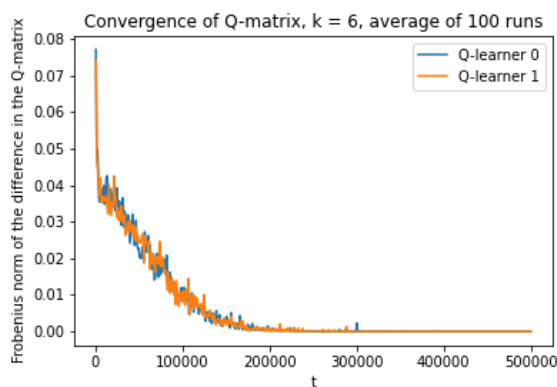
Figure 4.16: Forced deviation in the 3-player game, $k=12$

The deviation in the three-player game is shown in figure 4.16. There is no direct punishment to the deviating firm, however the firms react to their loss in profit by attempting to set higher prices. We compute the pre- and post-deviation profits for the deviating firm differently than for the two-player game. We take the mean of profits for $t = -27, \dots, -3$ to get the pre-deviation per-period profit and do the same for $t = 0, \dots, 27$ to get the per-period post deviation profits. For the set of runs shown in figure 4.16, the pre-deviation per-period profit was 0.1553 and the post-deviation per-period profit was 0.1440 for the deviator.

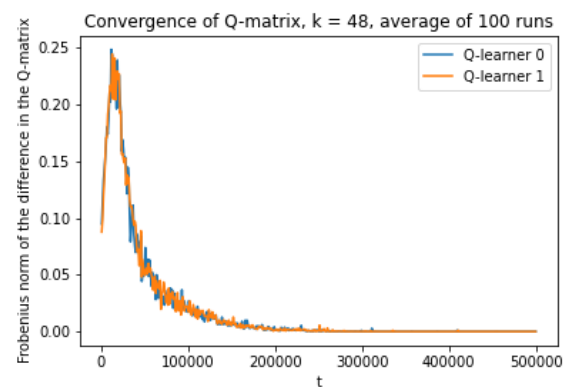
4.4 Convergence

In this section, we aim to ascertain the algorithm's convergence and its ability to attain a Q-table with a final strategy. To determine the convergence of the Q-table, we can

$\|\Delta\|$ or $\Delta\|\cdot\|$? conduct a straightforward examination. We observe whether there are any changes in the Frobenius norm of the Q-matrices. If no changes are detected, we can conclude that the table has converged.



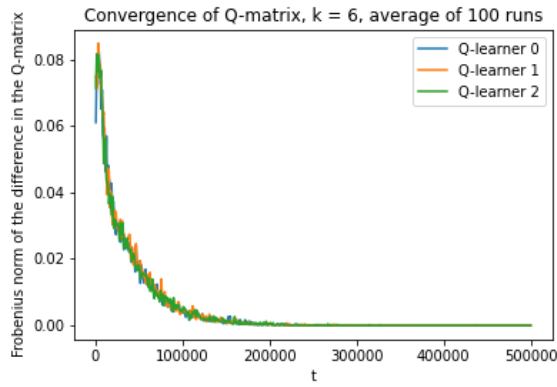
(a) Convergence for $k = 6$



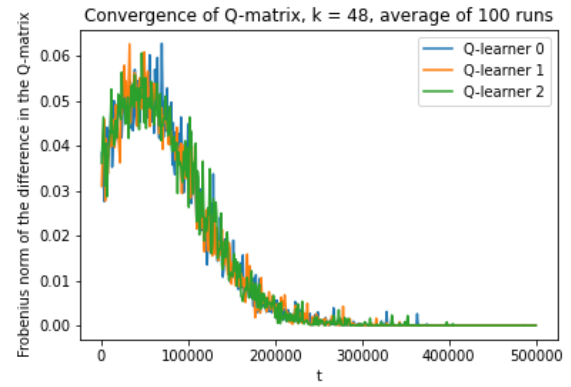
(b) Convergence for $k = 48$

Figure 4.17: Convergence 2 players

As expected figure 4.17 and 4.18 show convergence in the Q-matrices. It also looks like the convergence is more rapid for $k = 6$ than $k = 48$, as the norm falls quicker, hence the rate of convergence seems to be higher for $k = 6$. This tells us that more price options require more exploration, as more scenarios need to be explored.



(a) Convergence for $k = 6$



(b) Convergence for $k = 48$

Figure 4.18: Convergence 3 players

For $k = 48$ it is clear to see that the three-player simulation takes longer to converge which is expected, as more players means more scenarios to be explored.

Discussion

In this section, we discuss whether algorithmic collusion occurs in two- and three-player games. Furthermore, it is discussed what impact more players have on the presence of collusion. We discuss the results of our study and the assumptions made. We also suggest potential improvements.

5.1 Results summary

Throughout this paper, we have investigated Q-learning algorithms and their performance in a sequential price-setting environment. This has yielded a number of results, which we will discuss the meaning of in the following section.

5.1.1 Performance

From section 4.1 we saw how the profitability and average profit gain (Δ) fell when introducing another player to the sequential price-setting game. The profitability fell to levels above the static non-zero NE but under the EC benchmark. This tells us that the Q-learners fail to reach the same supra-competitive prices and profits when we introduce another player to the game. This could suggest that adding more firms to a given market makes algorithmic price coordination less profitable.

There is a parallel between this observation and how cartels work in theory. As Calvano *et al.* (2018, p. 4) argues, collusive agreements between agents become harder to uphold when more agents are included in the agreement. The same seems to be the case when the Q-learners have to deal with more competitors. The reasons for this

can be multifaceted. A remarkable similarity between our paper and Calvano *et al.* (2018), is that we demonstrate the exact same fall in profit gain by adding a third player as Calvano *et al.* (2018) does when increasing the memory of the Q-learners by one period. When we introduce another player into the game the firms now have to take two periods into account instead of one when drawing from the distributions of the states. As can be seen in figure 4.7, this results in the profit gain falling from approximately 88% to 56%. The average profit gain that Calvano *et al.* (2018, p. 36) demonstrates shows a decrease from 87% to 57%, ^{when they do what?} which the authors even argue is the same effect that would occur if a third player was added to their simultaneous game. Note that the nature of the games are different, as Calvano *et al.* (2018, p. 14) uses a simultaneous game, hence these results could be coincidental. ?

Calvano *et al.* (2018, p. 36) argues that this is due to the Q-learners failing to converge, as the Q-matrices grow too large for them to learn effectively. This could suggest that in our case, the reason for the fall in the profit gain is that the Q-learners cannot handle the increase in the information space well enough to learn collusive strategies.

An oversight we made with regards to the average profit gain is that we chose the static Bertrand-Nash equilibrium that gave zero profits, thus simplifying Δ to $\Delta = \bar{\pi}/\pi^M$. We therefore only have a measure of the profit gain above marginal cost. A potentially interesting point of investigation is the profit gain above the non-zero NE where prices are $1/k$. Δ would then have been the same measure as used in Calvano *et al.* (2018, p. 19). This could have altered our results and the interpretations thereof, but this path of investigation is left open due to time constraints. Why does this change and what?

An interesting case that we did not investigate is the question of what happens when the Q-learners' memory is restricted to one period even in the case of the three-player game. The Q-learners would then only be able to react to one of the opposing firms' prices. Their actions would have effects on their profits in the same way as before, however, they cannot observe the state of one of the firms directly. They would have to infer this information by proxy by observing their own profits. This would reduce the amount of information that they would process each period, which in turn might make

learning easier. The question is if the trade-off of losing information about one of the firms' prices leads to an increased ease of learning. We leave this as an open question for further investigation.

5.1.2 Strategy outcomes

In section 4.2 we show that firms change their strategies as k increases. These findings are similar to the ones in Klein (2020, p. 53). What's interesting is the way that firms in the three-player game skew towards the extremes; for $k = 6$ they play exclusively focal price strategies, whereas they almost exclusively exhibit price-cycling strategies when $k = 12$ and above, with the number of focal price runs being lower than for the two-player game. While the two-player price cycles support close to joint-maximizing levels of profitability, the same does not seem to hold for the three-player cycling strategies. Moreover, we display an example that shows the three players dividing the total profit unequally. These types of strategies could still be MPEs as per the Folk Theorem. Payoffs need only be strictly larger than zero for all firms to not conflict with the definition of a Nash equilibrium. More curiously, some of the runs show that one of the firms gains zero profit. For $T = 500,000$, this seems to be the case 4.5% of the time, but this frequency decreases to 1.9% as T increases, as we show in figure 4.14.

This shows that a non-negligible number of runs contain price cycling strategies that are completely unprofitable for at least one firm while other firms gain strictly positive profits. This conflicts with the definition of a Nash equilibrium, as the zero-profit firm can gain profit by changing its strategy e.g. by selecting the lowest non-zero gridpoint. This means that in these runs, at least one of the Q-learners failed to converge to an MPE. There can be a number of reasons for this. One of these reasons could be that the number of periods T is too small for the Q-learners to converge effectively or that the parameters α , δ , and θ are set sub-optimally for the three-player game. Another reason could be something similar to a saddle point problem. In the context of machine learning, this means that an algorithm fails and gets stuck in a saddle point

*You've shown that
 T is too low*


as it converges (Dauphin *et al.*, 2014, p. 1). Something similar could be happening to one of our Q-learners, resulting in the Q-learner not converging to an MPE.

5.1.3 Forced deviation *Repetition*

From section 4.3 we conclude that in the two-player game, there is a punishment for deviating and the overall profits are lower after a deviation. Following the argument from Calvano *et al.* (2018, p. 27), this is a key characteristic of algorithmic collusion. The firms in the three-player game do not punish the deviating firm in the same way, and there seems to only be a small difference in the average per-period profit pre- and post-deviation. Therefore, introducing a forced deviation to the three-player game does not seem to produce the same pattern of a punishment-reward strategy characteristic to algorithmic collusion.

5.1.4 Convergence

In section 4.4 we also found that the Q-learners converged to stable Q-matrices for both two and three players. It is important to note that we will see this convergence even if the Q-learners fail to approximate a good solution to the value-function problem. This is because we have defined the parameters ε and θ such that the Q-learners explore less over time. For this reason, they can end in a situation where the Q-matrices no longer update as much, although they have not optimized for their own profits sufficiently, hence the zero-profit runs shown in section 4.2.



5.2 Assumptions

Throughout the thesis, many assumptions have been made about the construction of the algorithms and competitive environment. In this section, we will try to discuss to what extent some of these assumptions uphold in the real world and what limitations

they have on our results. One of the main assumptions of the Q-learners is that they have complete knowledge of the opposing Q-tables. It is an assumption that is made such that the Q-learners have a training environment in which they can anticipate opponents moves, as otherwise, they would not be able to draw future states from the distributions. This assumption is less strict than it seems, as the only information necessary to replicate the opposing firms' Q-tables is the price and their learning parameters which we assume they know. An agent could thus reconstruct an approximation of the opponent's Q-tables and act based on those. It also seems fair to assume that firms know each others' prices in an experimental setting (Assad *et al.*, 2020, p. 6-7).

In continuation hereof Calvano *et al.* (2018, p. 6) argues that it can be difficult for firms to guess what specific algorithm the competitor is using, let alone have complete knowledge of the specific parameters. In this thesis, we have restricted the learning environment to the use of Q-learners, but in reality, there exist many different forms of reinforcement learning and different forms of Q-learners (Sutton and Barto, 2015, ch. 6).

Q-learning is one of the simplest reinforcement learning approaches and can be clearly characterized with certain parameters, of which the economic implications are clear (Calvano *et al.*, 2018, p. 5). More advanced reinforcement learning algorithms require a number of arguably arbitrary modeling choices, which makes them less suitable for economic interpretation, even if they might speed up convergence or solve the problem more efficiently (Calvano *et al.*, 2018, p. 5). The above points lead to some further questions regarding the algorithm's performance and applicability.

5.3 Further improvements

During our work on the algorithms, we thought of many ways in which we could improve speed and simplicity. Some of these possible improvements will be presented in this section.

First and foremost the code implementation should be written in a quicker programming language such as C, where parallelization is also more widely accessible (Womack *et al.*, 2022, ch. 1.1). **This could improve the running time drastically**, especially if more players were to be added, but even without adding more players, it would be an improvement to have the code in C as it would run a lot faster. *Do you know? Number can get close?* *Why didn't you do it then?*

A definite computational improvement that we realized too late was to implement the algorithms recursively since Q-learning itself consists of a recursive relation. This would make it more feasible to add more players dynamically. In our current implementation, we must add the extra players and dimensions manually, requiring a rewrite of the code. This takes a lot of man-hours and can introduce errors along the way. A recursive implementation written in C would be a massive step in the right direction. Unfortunately, this was realized too late, and we assessed that our time would be better spent on our current code setup.

A very obvious change or supplement to our setup would be to alter the demand function so that we would move away from classical Bertrand competition. For example, you could introduce differentiated products instead of homogeneous products or add marginal cost back into the game. This could make for a more realistic competitive environment.

Conclusion

Throughout the thesis, we have investigated what consequences adding more players to the sequential game has for the presence of algorithmic collusion. We can conclude that in our sequential game, both profits and average profit gains were reduced by a large margin with the addition of an extra player. Moreover, we show that Edgeworth price cycling strategies occur for two players.

The three-player game also shows price cycling strategies, although these become more erratic as the price grid becomes finer. These cycles still exhibit supra-competitive prices and average profit gains but these are markedly lower compared to the price coordination in the two-player game. It was shown that the Q-learners in some instances failed to reach a solution, in the sense that some of the three-player runs had one or more firms receive zero profits, even though there were profitable deviations from the zero profit strategy. As T increases, the frequency of these zero profit runs falls from approximately 4.5% to 1.9%.

We also found that the Q-learners lose the tendency to punish each other for deviating from a potentially collusive strategy when going from two to three Q-learners in a competitive environment. Hence, according to the definition of collusion in Calvano *et al.* (2018) the coordination in the three-player game that we have described cannot be said to be collusion.

Furthermore, we saw that the pricing strategies that the Q-learners chose shifted from predominantly being fixed-price strategies at $k = 6$, to almost exclusively being price-cycles at $k = 48$, and that the agents in the two-player game favored fixed-price strategies more than the agents in the three-player game for $k = 6$ to $k = 48$. The

changes in behavior with respect to k in the two-player game are similar to the findings in Klein (2020, p. 53).

These changes in strategy also had a small impact on the average profit gain, although the effects were different for the two-player game and the three-player game. As firms change from fixed price strategies to price cycling strategies from $k = 6$ to $k = 12$, there is a decrease in the average profit gain. As k then increases for the two-player game, the average profit gain approaches the joint-maximizing levels again, while it keeps falling for the three-player game.

Our research suggests that adding more agents to a market makes algorithmic collusion between Q-learners less likely. Hence policymakers should keep in mind that furthering competition could lower the risk of algorithmic collusion in a competitive Q-learning pricing environment.

Bibliography

- Asker, John, Chaim Fershtman, and Ariel Pakes (Mar. 2021). *Artificial Intelligence and Pricing: The Impact of Algorithm Design*. Working Paper 28535. National Bureau of Economic Research.
- Assad, Stephanie, Robert Clark, Daniel Ershov, and Lei Xu (2020). „Algorithmic Pricing and Competition: Empirical Evidence from the German Retail Gasoline Market“. In: *CESifo Working Paper Series* 8521.
- Calvano, Emilio, Giacomo Calzolari, Vincenzo Denicolo, and Sergio Pastorello (2018). „Artificial Intelligence, algorithmic pricing and collusion“. In: *SSRN Electronic Journal*.
- Danish Medicine Agency (2019). *Prices of medicine*. Accessed on June 8, 2023. Danish Medicine Agency. URL: <https://laegemiddelstyrelsen.dk/en/reimbursement/prices/>.
- Dauphin, Yann N., Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio (2014). „Identifying and Attacking the Saddle Point Problem in High-Dimensional Non-Convex Optimization“. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, pp. 2933–2941.
- DCCA (2020). *Cartels*. Accessed on Jun 7th, 2023. URL: <https://www.kfst.dk/konkurrenceforhold/karteller/>.
- DCCA (2021). „PRISALGORITMER OG DERES BETYDNING FOR KONKURRENCEN“. In: *Velfungerende Markeder* 46.
- Engel, Christoph (2011). „Dictator games: a meta study“. In: *Experimental Economics* 14.4, pp. 583–610.
- Ivaldi, Marc, Bruno Jullien, Patrick Rey, Paul Seabright, and Jean Tirole (2003). *The Economics of Tacit Collusion*. Final Report for DG Competition, European Commission. IDEI, Toulouse.

- Karlsen, Morten Vestergaard and Johanne Emilie Wismann (2022). *Price setting with artificial intelligence -A simulation study of collusive behaviour and asymmetric pricing technology*.
- Klein, Timo (2020). *Essays in Competition Economics*. Tinbergen Institute research series. Universiteit van Amsterdam.
- Klein, Timo (2021). „Autonomous algorithmic collusion: Q-learning under sequential pricing“. In: *The RAND Journal of Economics* 52.3, pp. 538–558. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/1756-2171.12383>.
- Koenig, Sven and Reid G. Simmons (1993). „Complexity Analysis of Real-Time Reinforcement Learning“. In: *Proceedings of the Eleventh National Conference on Artificial Intelligence*. AAAI'93. Washington, D.C.: AAAI Press, pp. 99–105.
- Maskin, Eric and Jean Tirole (1988). „A Theory of Dynamic Oligopoly, II: Price Competition, Kinked Demand Curves, and Edgeworth Cycles“. In: *Econometrica* 56.3, pp. 571–599.
- Munk-Nielsen, Anders (2022a). *Dynamiske Spil: Lecture Slides Microeconomics B*. 1st ed.
- Munk-Nielsen, Anders (2022b). *Nash-ligevægten: Lecture Slides Microeconomics B*. 1st ed.
- Numba (2023). *Numba documentation*. <https://numba.readthedocs.io/en/stable/>. [Online; accessed 21-May-2023].
- Porter, Robert H. (2005). „Detecting Collusion“. In: *Review of Industrial Organization* 26.2, pp. 147–167.
- Sutton, Richard S. and Andrew G. Barto (2015). *Reinforcement Learning: An Introduction*. 2. ed.
- Tadelis, Steven (2013). *Game Theory: An Introduction*. Princeton University Press.
- Varian, Hal R. (2014). *Intermediate Microeconomics With Calculus - A Modern Approach: An Introduction*. 1. ed.
- Womack, Michael, Anjia Wang, Patrick Flynn, Xinyao Yi, and Yonghong Yan (2022). *OpenMP Programming Book*. Accessed on June 9, 2023. University of North Carolina at Charlotte. URL: https://passlab.github.io/OpenMPProgrammingBook/openmp_c/1_IntroductionOfOpenMP.html.