Midterm Report - Algorithmic Collusion (MIR)

Côme Campagnolo[*]

March 14th 2024

1 Introduction

A literature review has been done to present the main problems about algorithmic collusion and how they have been discussed until now, mainly in the economic literature, but also in the computer science and finance literatures. After this presentation, the main focus of the research project will be to better understand **which specific market designs are favorable to collusive behaviors between AI agents** and to what extent these behaviors can be anticipated. This question underlines the decisive aspect of the algorithm design and also relates, from a policymaker's perspective, to the issue of "compliance by design". In practice, the number of agents (or players, or firms), the tick size (i.e. the minimal spread between two prices) and the delay of actions between agents are the key features that will be tested.

2 Building a Q-learning algorithm in an economic setting based on a Bertrand Price Model

Considering any finite Markov decision process, the aim of a Q-learning algorithm is to find an optimal action-selection policy, that is to say one that maximizes the expected value of the total reward, which is a weighted sum of expected values of all the future rewards, starting from the current step. The "Q"-function is the key element, the function which gives the expected rewards for every possible action in a given state. In comparison with other types of algorithms, Q-learning is relatively simple and transparent. The algorithm has **zero prior knowledge**, which means it only knows its set of available actions (the price grid) and observes the payoff from these actions afterwards in order to learn, that is to say to update its Q-values.

Starting from a Q-table filled with random values chosen between 10 and 20, so that their perceived value always exceeds the maximum price, the agent will update its Q-values after each iteration in order to have a better approximation of the reward-matrix. For this, it needs to gather information about the rewards of its possible actions in various states, which implies that it needs to experiment new actions, especially in the early stage of the simulation. However, its main goal remains to maximize the total reward, therefore it also wants to exploit the action which has the highest expected value in a given state.

Main variables and explanation of the setting

- **Q-table:** The Q-values represent the expected future rewards for taking specific actions in particular states.

[*]Master of Economics (M1), ENS Paris-Saclay, 4 av. des Sciences, 91190 Gif-sur-Yvette

- `epsilon` (between 0 and 1): The **exploration rate** determines the probability of choosing a random action instead of the one suggested by the policy:

Exploration strategy: **Exploitation** refers to the agent's strategy of setting prices that it perceives as maximizing its expected reward (profit) based on its current knowledge (Q-values). **Exploration** refers to the strategy of testing new actions, deviating from the agent's current best estimate of optimal prices to experiment with different pricing strategies. This allows the agent to gather more information and potentially uncover better solutions, but at the cost of not exploiting the action with the highest-known reward. This trade-off, also known as exploitation-exploration dilemma, leads to different rules for choosing between exploitation and exploration. The most basic one is the **ϵ-greedy exploration**:

$$p_a \begin{cases} \sim U\{P\} & \text{with probability} \varepsilon_i \\ = \text{argmax}_p Q_i(p, s_i) & \text{with probability} 1 - \varepsilon_i, \end{cases} \tag{1}$$

In this strategy, exploration corresponds to choosing any random price. More sophisticated decision rules allow to explore more specifically and to fix targets. ϵ depends on t, as it is generally supposed to decrease over time, because the opportunity cost of exploration increases over time[1]. The decay rate determines how quickly the exploration rate decreases. A faster decay rate increases the probability of converging to sub-optimal solutions.

- `alpha`: The **learning rate** controls the weight given to new information when updating the Q-values. A higher alpha means that the agent relies more on the most recent experiences, leading to faster learning and adaptation to changes in the environment. Conversely, a lower alpha results in slower learning, as the agent gives more weight to past experiences. We could think that a higher value of alpha is useful to accelerate learning, but it is not necessarily the case, as it can lead the model to **overfit** to recent experiences and the model can be less generalizable. This is especially true in a 'noisy' environment (e.g. with high variance).

Overfitting problem: The agent learns the Q-values too closely to the specific experiences encountered during training, rather than generalizing well to new, unseen states and actions. In the case of our Bertrand Price model, this would mean that the model doesn't generalize well to other settings and parameters. For example, the agent may learn to set prices optimally for low-demand scenarios but perform poorly when faced with high-demand situations. Overfitting also increases the risk of converging to sub-optimal policies.

- `decay_rate`: The decay rate determines the rate at which the exploration rate (epsilon) decreases over time. A value close to 1 means a slow decay, while a value close to 0 means a rapid decay.
- `D_mean`: The market demand is fixed in the first case. We can add variance by introducing `D_std`. We start with a standard deviation of 0, i.e. with a constant demand.
- `c`: The cost parameter, which represents the marginal cost of production for each firm, is constant and equal for both firms.
- `pL` and `pH`: These parameters set the lower (`pL`) and upper (`pL`) bounds of the price range.
- `dim`: The dimension parameter represents the number of discrete price points within the price range (from `pL` to `pH`) because we use a discrete action space.

[1]although this assumption could be discussed, cf. Compte

Tick size: As price takes values between 0 and 10 and `dim` $= 100$, we have a tick size of 0.1. Not only does the tick size affect the computation time, but it can also influence the convergence behavior and stability of the learning process. The importance of the tick size is a problem that has been discussed in the finance literature (cf. Cordella-Foucault[2], especially in order to determine the speed of convergence and the expected trading cost.

- `synchronous = 0`: This sets the updating rule parameter (`synchronous`) to 0, indicating an asynchronous updating rule.

Update rule: The Q-values are iteratively updated based on the agent's experiences.

$$Q(s,a) \leftarrow (1-\alpha)\cdot Q(s,a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s',a')) \tag{2}$$

- $Q(s,a)$ is the Q-value for state s and action a.
- α is the learning rate, determining the weight given to new information (reward).
- r is the reward received for taking action a in state s.
- γ is the discount factor, representing the importance of future rewards.
- s' is the next state after taking action a.
- a' is the next action chosen in state s'.
- $max_{a'}Q(s',a')$ represents the maximum Q-value for the next state s', reflecting the agent's estimate of the maximum expected future reward.

In **synchronous updates**, all agents simultaneously update all their Q-values based on their current estimates. This means that each agent considers the same set of Q-values for the current state and synchronously updates their own Q-value based on these values. Synchronous updates are used when agents have access to a shared global state, which is not the case here.

In **asynchronous updates**, each agent updates its Q-values independently of the others and updates only the Q-value of the action (price) chosen at this period ($W_i^k(p|s_i)$ is updated only for $p = p_i^k$ at $s_i = s_i^k$). If the Q-value is changed during the update, the updating rule is as follows:

$$W_i^{k+1}(p|s_i^k) = \alpha(k)\left[\pi_i(s_i^k,p) + \beta\left(\max_{y\in\mathcal{P}}\{W_i^k(y|s_i^{k+1})\}\right)\right] + (1-\alpha(k))W_i^k(p|s_i^k). \tag{3}$$

Otherwise, it remains as in the previous iteration:

$$W_i^{k+1}(p|s_i^k) = W_i^k(p|s_i^k). \tag{4}$$

The update rule calculates $W_i^{k+1}(p|s_i^k)$, the new perceived value for action p in state s_i^k, based on the observed profits $\pi_i(s_i,p)$ and the maximum perceived value in the next state s_i^{k+1}. Between the synchronous and asynchronous updating rules, other updating rules can be designed so as to have more or less information about the economic model. Using more sophisticated updating rules would be useful if we model more complex economic environments.

[2]CORDELLA, T., & FOUCAULT, T. (1999). Minimum Price Variations, Time Priority, and Quote Dynamics. *Journal of Financial Intermediation*, *8*(2), 141–173.

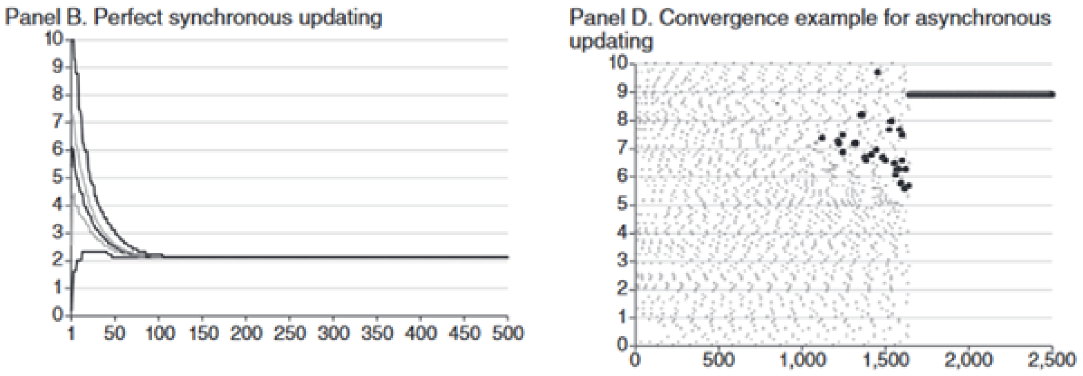


FIGURE 1. PRICE OUTCOMES WITH DIFFERENT ALGORITHM DESIGNS

- Q_i^0 = initial set of perceptions = random values between 10 and 20 (it needs to be over the max price `pH`).
- `itermax`: This sets the maximum number of iterations (`itermax`). The algorithm will terminate if convergence is not reached within this number of iterations.
- `nfirms`: This sets the number of firms (`nfirms`) participating in the market.

The number of firms. Collusion is expected to become harder as there are more agents. We find indeed a substantial decrease of convergence prices depending on the number of firms, especially between 2 and 3 firms, also between 3 and 4 firms, and less significantly for more firms. Increasing the number of agents leads to a significant growth of computation time because of the size of the Q-table.

- `n_simulations` (the seed values ensure reproducibility of the results across simulations)

"Convergence" or how to end the simulation: In the first model, we say that prices have converged when they have remained stable for a sufficient number of consecutive iterations (e.g. 200). In a case with exploration, this won't work, so we say that convergence is reached if Q-values have remained relatively stable for a sufficient number of consecutive iterations. Then, we use a sort of "q-stability threshold", which corresponds to the fact that the maximum value in the absolute difference matrix, i.e. the largest difference between any two corresponding elements in the current and previous iterations of the Q-table is lower than a given threshold.

3 Simulation example in a Bertrand Price Duopoly setting

Model from Asker et al. (perfect synchronous vs. asynchronous updating rule): Starting from a simple Bertrand Price Model, with an equilibrium price p*=2, we reproduce the experiment from Asker et al. [3] and find the same results [4] for synchronous and asynchronous updating rules, as shown in Figures 1 to 4 in the Appendix.

[3]1. ASKER, J., FERSHTMAN, C., PAKES, A. (2021). Artificial Intelligence and Pricing: The Impact of Algorithm Design. *NBER Working Paper No. w28535.*

[4]BPM V0

3.1 Synchronous update

See Figures 1 & 2 in the Appendix

In the perfect synchronous case, the convergence price is reached after less than 100 iterations and correspond to the Nash equilibrium price (p=2). This simulation is possible because this algorithm has a demand and cost model.

3.2 Asynchronous update

See Figures 3 & 4 in the Appendix

In the asynchronous case, convergence takes more time and reaches a supra-competitive price: $p = 7.4$ after more than 2000 iterations in this example.

4 Multiple Simulations in a Bertrand Price Duopoly setting

See Figures 5 & 6 in the Appendix

Running multiple simulations with a similar setting[5], varying only the **number of firms** (represented by different colors), we found out a pattern in which convergence takes more time to be reached, from around 2-3 000 for 2 firms up to 9-10 000 for 5 firms, and equilibrium prices converge towards lower values as the number of firms increase, becoming on average very close to the Nash equilibrium prices for 4 or 5 firms.

5 Further Problems to Solve

1. Focusing on reward-punishment strategies / patterns in collusive outcomes

   - Comparison with the case of forced deviation from the collusive equilibrium: Price paths following an optimal deviation

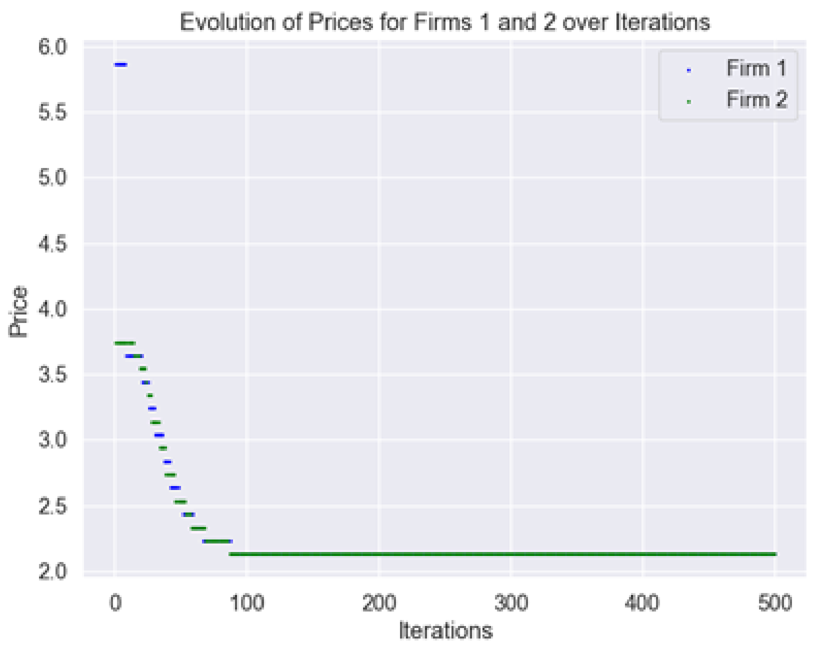
2. The tick size and its effects:

   - convergence prices
   - speed of convergence
   - convergence patterns

3. Introducing delay in agent's actions (how to do it is still not clear)
4. Going further in the definition of "convergence" in our context

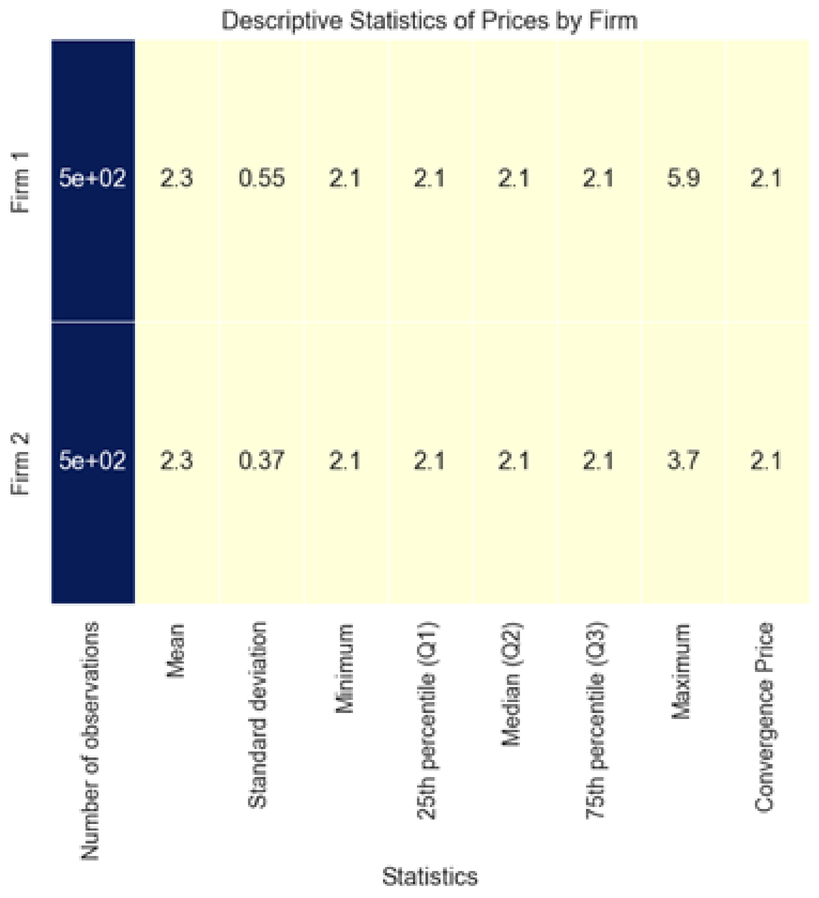
6 Appendix

[5]see code BPM V2



t

Figure 1: Synchronous Update Price Evolution



Descriptive Statistics of Prices by Firm

	Number of observations	Mean	Standard deviation	Minimum	25th percentile (Q1)	Median (Q2)	75th percentile (Q3)	Maximum	Convergence Price
Firm 1	5e+02	2.3	0.55	2.1	2.1	2.1	2.1	5.9	2.1
Firm 2	5e+02	2.3	0.37	2.1	2.1	2.1	2.1	3.7	2.1

Statistics

b

Figure 2: Synchronous Update Descriptive Statistics

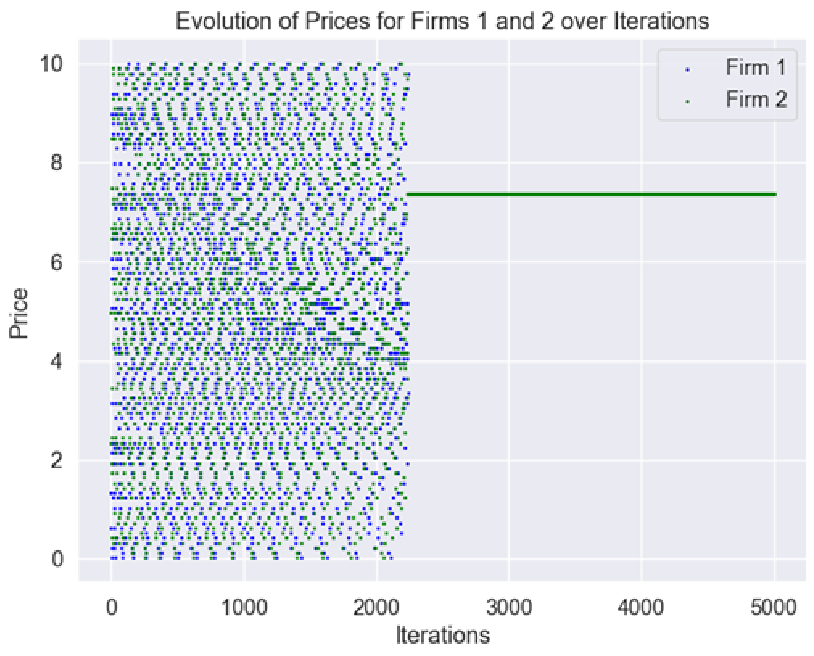
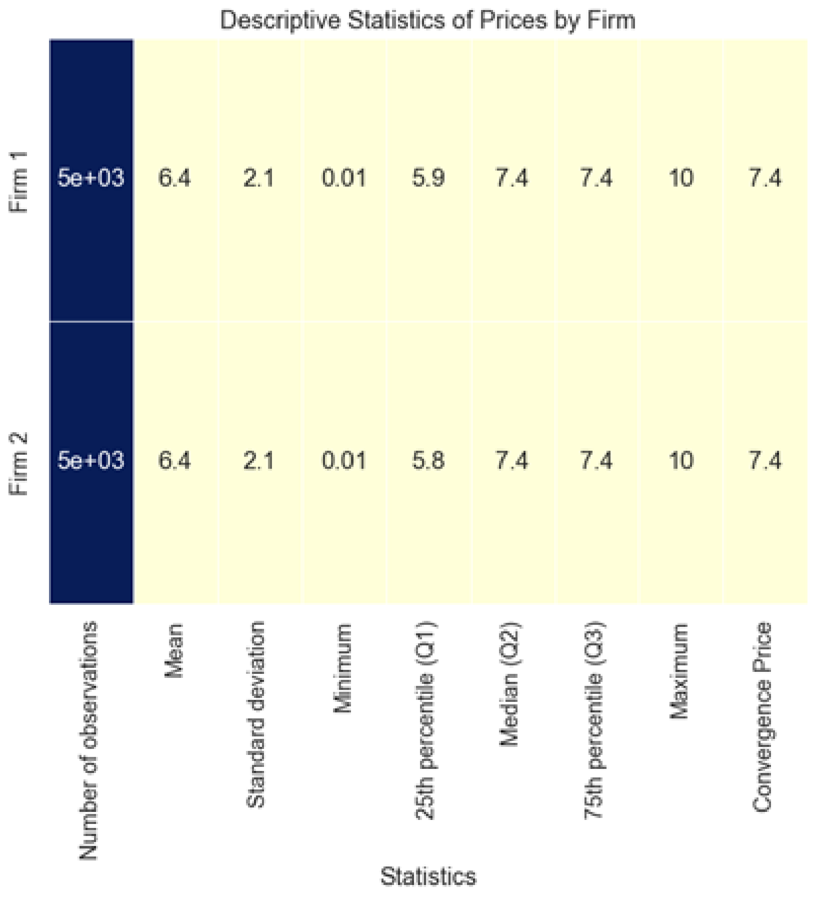


Figure 3: Asynchronous Update Price Evolution



Descriptive Statistics of Prices by Firm

	Number of observations	Mean	Standard deviation	Minimum	25th percentile (Q1)	Median (Q2)	75th percentile (Q3)	Maximum	Convergence Price
Firm 1	5e+03	6.4	2.1	0.01	5.9	7.4	7.4	10	7.4
Firm 2	5e+03	6.4	2.1	0.01	5.8	7.4	7.4	10	7.4

Statistics

Figure 4: Asynchronous Update Descriptive Statistics

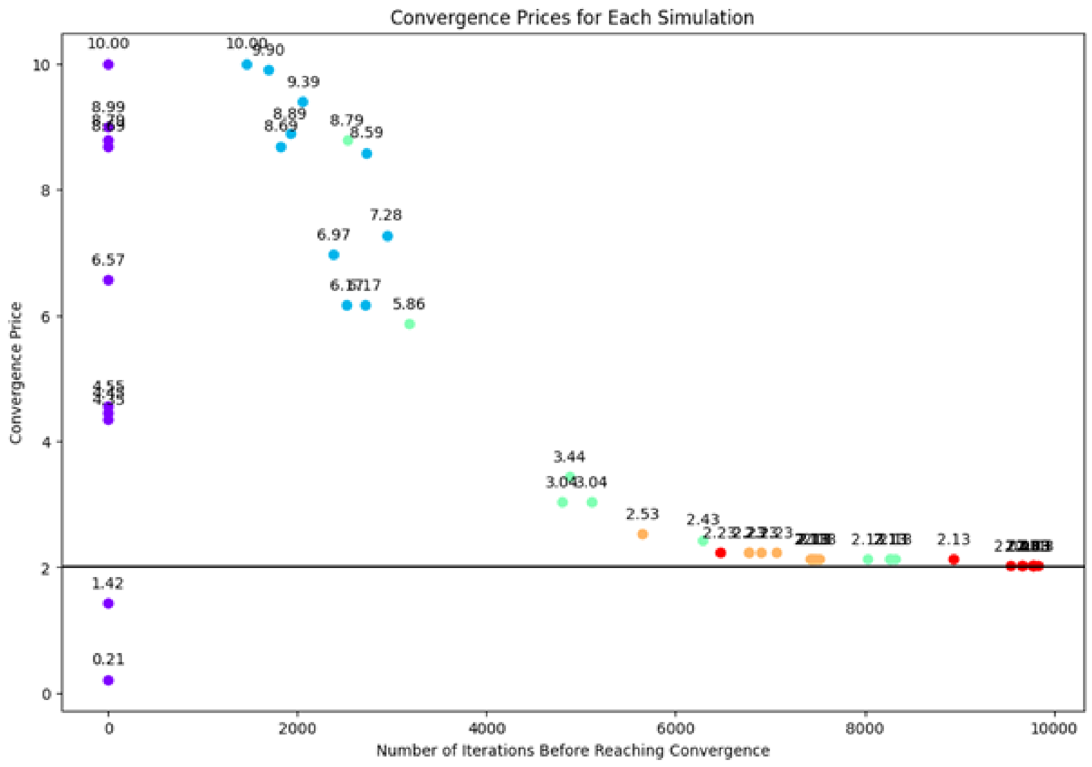


Figure 5: Convergence Prices over Multiple Simulations Depending on the Number of Firms

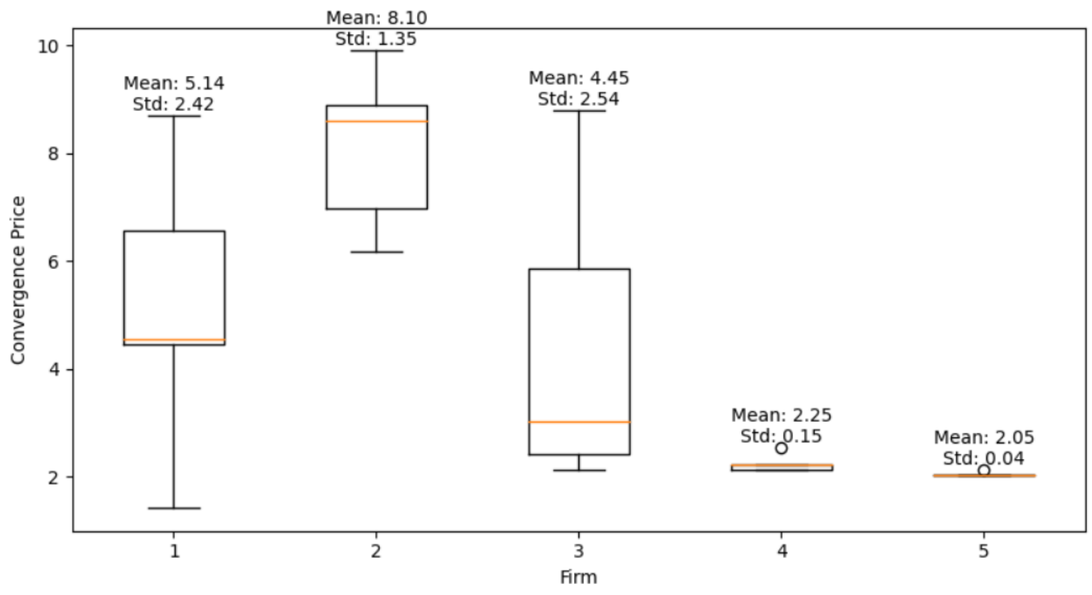


Figure 6: Descriptive Statistics on Convergence Prices Depending on the Number of Firms