Simulation of Unintentional Collusion Caused by Auto Pricing in Supply Chain Markets

Preprint · October 2020

DOI: 10.13140/RG.2.2.11305.21607

4 authors:

Masanori Hirano
Preferred Networks Inc.
69 PUBLICATIONS **103** CITATIONS
SEE PROFILE

Hiroyasu Matsushima
Shiga University
115 PUBLICATIONS **271** CITATIONS
SEE PROFILE

Kiyoshi Izumi
The University of Tokyo
259 PUBLICATIONS **1,355** CITATIONS
SEE PROFILE

Taisei Mukai
Waseda University
10 PUBLICATIONS **13** CITATIONS
SEE PROFILE

Simulation of Unintentional Collusion Caused by Auto Pricing in Supply Chain Markets

Masanori HIRANO[1][0000−0001−5883−8250], Hiroyasu MATSUSHIMA[2][0000−0001−7301−1956], Kiyoshi IZUMI[1], and Taisei MUKAI

[1] School of Engineering, The University of Tokyo, Japan
{hirano@g.ecc, izumi@sys.t}.u-tokyo.ac.jp https://mhirano.jp/
[2] Center for Data Science Education and Research, Shiga University, Japan
hiroyasu-matsushima@biwako.shiga-u.ac.jp

Abstract. In this paper, we address the problem of unintentional price collusion, which happens due to auto pricing, such as systems using reinforcement learning. Previous work has pointed out the risk of unintentional collusion caused by auto pricing using Q-learning, one of the basic models of reinforcement learning, using a market simulation. This study expands the previous study in 2 points, more sophisticated learning models, and the effect of public information. Firstly, sarsa and deep Q-Learning models were used for auto pricing to test whether they cause collusion like Q-learning. To test them, we performed multi-agent simulations of a competitive market with a pre-defined demand function. In each simulation, the agents learn their pricing strategies using reinforcement learning. And we defined and calculated the new collusion metric representing how agents collude. Secondly, we tested cases with open and shield bidding with multiple numbers of agents. In our result, we observe that Deep Q-Learning demonstrates the highest collusion metric. Also, contrary to expectations, we found that shield bidding has no significant effect on collusion levels when agents employ outperforming reinforcement learning, such as deep Q-learning. Moreover, the number of agents also contribute to less collusion levels.

Keywords: Auto Pricing · Unintentional Collusion · Reinforcement Learning · Deep Learning · Market.

1 Introduction

Generally, many automated processes are being introduced today. In some shops, the prices are changed based on automatic algorithms using supply-demand and sales information. Amazon, which is the biggest online shop, is partially introducing the auto pricing system based on recent sales or competitors' prices. These auto pricing systems are favorable to sellers. However, the current systems are a rule-based system whose performance is limited.

Moreover, recently, there has been a trial to automate the negotiations in buying and selling among products' supply chains. Last year, the new league called "supply chain management league" was made in the 10th International

Automated Negotiating Agents Competition [17]. It represents the rise of motivation towards automatical pricing, negotiation throughout the supply chain.

On the other hand, recently, many works relating to reinforcement learning, game theory, and mechanism design have been proposed with some implemented in real-life scenarios [11]. In terms of the auto pricing system, reinforcement learning can be applied to it. Moreover, for making the best of the auto pricing strategy, it is crucial to think of pricing competition in game theoretic terms.

Although applying these technologies to the auto pricing system makes it more beneficial and efficient, the new technologies poss some risks. The risks of auto pricing were pointed out in [5]. This indicates that the auto pricing system can learn unintentional price collusion without direct interaction between competitors. In [5], Q-Learning for auto pricing was tested.

Generally, Q-Learning, as a simple learning model, is easy to implement in real-life. Thus, it is widely used for real-life applications. So, if even Q-learning causes unintentional collusion, unintentional collusion in auto pricing is not only thought in theory but also becomes realistic now.

Moreover, the risks seem to be increasing using cutting-edge technologies. The performances of Q-Learning are limited compared with other recent technologies such as Deep Q-Learning. Used more outperforming reinforcement learning, the risks of unintentional collusion is assumed to be more increasing. It is because we could assume that a more outperforming learning method can learn strategies more efficiently.

The pricing problem can also be thought of in terms of game theory. The pricing problem (we assumed the model based on Bertrand competition [3], in which the seller who submits lower price could sell more products) usually seems to be a non-cooperative game. However, if we assumed it is a cooperative game, the situation of the game would change completely. As a non-cooperative game, basically, a lower price near the Nash equilibrium is a better strategy for each seller. On the other hand, as a cooperative game, a higher price near the cooperative equilibrium has higher profits than the Nash equilibrium. So, the dynamics in this situation are very difficult to learn, and it is uncertain how well reinforcement learning with higher performance such as Deep Q-Learning would perform.

In this paper, we test the risk of some reinforcement learning models for unintentional price collusion using our new collusion metric. Furthermore, to avoid unintentional price collusion, we test some situations, such as the shield-bidding market with multiple competitors.

2 Related Works

First, some numbers of research works related to the supply chain have been reported. The model employed in our study is just a minimum part of the supply chain because a simpler model is easy for us to understand what's happening and analyze deeper. As a complicated supply chain model, Yasser *et al.* showed their supply chain management world model [16]. Yasser also released his platform

for supply chain management called "*negmas*" [15]. This platform was also used for the supply chain management league (SCML) in the 10th International Automated Negotiating Agents Competition (ANAC2019) [17]. These works and competition aimed to negotiate between sellers and buyers automatically for optimization in supply chain systems.

Second, there are a lot of works trying to apply game theory to real-world tasks. The old and well-known models for pricing competition are Bertrand's duopoly model [3] and the Cournot competition model. Both models tried to analyze the duopoly market using game theory. Hence, we adopt the former as the basis of our paper. As another work applying game theory to the real-world task, Tambe showed the application of the Stackelberg competition for aviation security [24]. Recently, Castiglioni *et al.* extended the Stackelberg competition, which assumes that there are one leader and a follower, and enables it to handle the game with multiple leaders and followers [6].

In the context of reinforcement learning, many works and developments have been made throughout a long history. The Q-Learning is a famous reinforcement learning, which is an off-policy reinforcement learning based on Q-table [28]. The learning theory, i.e., the temporal difference, was proposed in [22]. Tesauro proposed the method of applying temporal difference learning to backgammon [25]. Sarsa, State–action–reward–state–action, is another example of simple reinforcement learning proposed in [18]. Although it is very similar to Q-Learning, it is an on-policy learning method. These two learning strategies are used in our study. But, there are also a lot of models for reinforcement learning. Generally, the most significant discovery in reinforcement learning is its combination with a neural network. After deep reinforcement learning becomes available, Mnih *et al.* [14] showed Deep Q-learning (Deep Q-Network; DQN) could outperform humans using the Atari Learning Environment (ALE) [2]. These improvements are possible due to the invention of neural networks, deep learning, such as a convolutional neural network for image classifications [13]. Consequently, many reinforcement learning methods using deep learning began to appear. As the extension of DQN, Double DQN (DDQN), was proposed in [26], and it performs better compared with DQN using two networks. Moreover, Dueling DDQN was also proposed using a new state value function to improve learning performance [27]. As the extensions for Q-Learning, Asynchronous Advantage Actor-Critic (A3C) [7] is a method using deep learning asynchronously. This is based on the actor-critic method proposed in [23]. However, afterward, because asynchronous is not important, Advantage Actor-Critic (A2C), was proposed [1]. As a method combining many methods mentioned above, Hessel proposed RainBow in [8]. As more outperforming methods, Ape-X DQN [10], which is using prioritized experience replay, and R2D2 [12], which is also using long short-term memory (LSTM) [9], were also proposed. Silver *et al.* proposed a reinforcement learning algorithm through self-playing, and they showed great performance in some games, Chess, Shogi, and Go [20]. This algorithm is based on their previous works, which is well known as "Alpha Go Zero." [21].

As works related to auto pricing, the dynamic pricing model is also studied. Generally, dynamic pricing is the way to set services or product prices according to the estimated demand. Dynamic pricing has been studied extensively by [4]. Ye *et al.* showed a dynamic pricing model that is genuinely used for Airbnb and its performances [29]. Shukla *et al.* also showed a dynamic pricing model used in the Airline booking price system and its performance [19]. On the other hand, there are some works related to auto pricing except for dynamic pricing, i.e., auto pricing under almost stable demands. Calvano *et al.* suggested that price collusion by auto pricing using reinforcement learning can occuer [5]. Hence, in this paper, we address this problem.

3 Market Settings

Here, we explain the market settings we employed. The market is based on Bertrand competition model [3]. It allows supply prices to be fixed by each production seller, and the demand quantities for each seller are calculated based on the prices. Figure 1 explains this concept. Because Bertrand competition

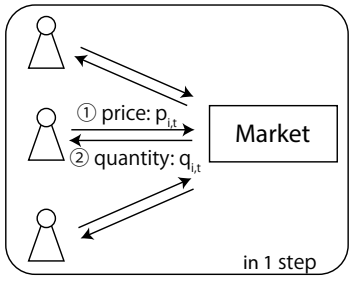

Fig. 1. Our market model. In every step, all agents submit their bidding price. Then, the quantities for them are calculated by the market. There is only one market in our simulation, and its demands are defined by the equation using the bidding prices of all agents.

model is a winner-take-all model (i.e., a seller who submits the lowest sell price gets all demands), the model's demand function is not C^0 function (i.e., the function is a non-smooth function). So, the model's demand function is difficult to analyze using numerical analysis. Thus, we made another demand function as the following.

$\mathscr{N}$ is the set of agents (sellers), and if the number of agents is N, $\mathscr{N} = \{1, 2, 3, \cdots, N\}$. Agent (products seller) i ($\in \mathscr{N}$) set their sell price at time t to $p_{i,t}$ (t means steps in simulations, i.e., $t = 1, 2, 3, \cdots$). Here, we noted the possible price space $\mathscr{P}$ (We assumed all agents set their sell price in the same range), i.e., $\forall i \in \mathscr{N}, \forall t, p_{i,t} \in \mathscr{P}$. $\mathscr{P}$ is usually the continuous non-negative values, i.e., $[0, \infty)$, in the real world, but this can be set of discrete non-negative values, such

as $\{1, 2, 3, \cdots, 10\}$, in our simulations. We also defined P_t as the set of all agents' price at time t, i.e., $P_t = (p_{1,t}, p_{2,t}, \cdots, p_{N,t})$.

At the step t, all agents publish their sell prices $p_{i,t} (i \in \mathscr{N})$. Then, a sell quantity of agent $i \in \mathscr{N}$ is calculated as:

$$q_{i,t} = \frac{\exp\left(-\frac{1}{\mu} p_{i,t}\right)}{\sum_{j \in \mathscr{N}} \exp\left(-\frac{1}{\mu} p_{j,t}\right) + \exp\left(-\frac{1}{\mu} a\right)} \tag{1}$$

where μ and a are given fixed parameters and $\mu = 5.0$ and $a = 5.0$ in our simulation. Here, μ is a kind of scaling factor, i.e., $p_{j,t}$ depends on μ, and if μ and a become twice, $p_{j,t}$ will be twice to get the same $q_{i,t}$. But, a can affect the market dynamics because the term $\exp\left(-\frac{1}{\mu} a\right)$ has effects on the total amount of sold quantities. So, we set a empirically. This equation is based on [5], but we have reduced some variables to facilitate holomorphic analytics. One of the benefits of this demand function is that the function is C^2 function (a function which can be differentiated at least twice) at least. The other is that, when two agents publish a very similar sell price, the response is less dramatic than Bertrand ones. It is because Bertrand's demand function is not C^0 function as:

$$q_{i,t} = \begin{cases} 1 & (\text{when } \forall j \in \mathscr{N} \backslash \{i\}, p_{i,t} > p_{j,t}) \\ 0 & (\text{when } \exists j \in \mathscr{N} \backslash \{i\}, p_{j,t} > p_{i,t}) \\ 1/n & (n = |\{j \in \mathscr{N} | p_{j,t} = min_{k \in \mathscr{N}} \{p_{k,t}\}\}| \text{ when } p_{i,t} = min_{j \in \mathscr{N}} \{p_{j,t}\}) \end{cases},$$

which means a winner takes all demand, but if there are multiple winners, they share a pie. So, we employed equation 1.

Then, in our simulation, we ignored the production cost because it does not affect the dynamics of markets. Thus, the profits of each agent are calculated as:

$$\Pi_{i,t}(P_t) = p_{i,t} \times q_{i,t} = \frac{p_{i,t} \exp\left(-\frac{1}{\mu} p_{i,t}\right)}{\sum_{j \in \mathscr{N}} \exp\left(-\frac{1}{\mu} p_{j,t}\right) + \exp\left(-\frac{1}{\mu} a\right)}. \tag{2}$$

This setting is very useful for analytics because it has one Nash equilibrium and one cooperative equilibrium, and both equilibria have the same value of $p_{i,t}$ for all agent $i \in \mathscr{N}$. The definitions for Nash equilibrium and cooperative equilibrium are the following:

- If $P_t^* = \left(p_{1,t}^*, p_{2,t}^*, \cdots, p_{N,t}^*\right)$ is Nash equilibrium,

$$\forall i \in \mathscr{N}, \forall p_{i,t} \in \mathscr{P} : \Pi_{i,t}\left(p_{i,t}^*, P_{-i,t}^*\right) \geq \Pi_{i,t}\left(p_{i,t}, P_{-i,t}^*\right),$$

  where $P_{-i,t}^*$ is a tuple from P_t^* except for $p_{i,t}$, is satisfied.

– If $P_t^{**} = \left(p_{1,t}^{**}, p_{2,t}^{**}, \cdots, p_{N,t}^{**}\right)$ is cooperative equilibrium,

$$\forall P_t : \sum_{i \in \mathscr{N}} \Pi_{i,t}\left(P_t^{**}\right) \geq \sum_{i \in \mathscr{N}} \Pi_{i,t}\left(P_t\right),$$

is satisfied.

4 Collusion Metric

Here, we define a metric for unintentional collusion. It is essential to estimate the performance or the risk of unintentional collusion. Therefore, we defined our collusion metric as:

$$C_t = \frac{1}{|\mathscr{N}|} \sum_{i \in \mathscr{N}} \frac{\Pi_{i,t}\left(P_t\right) - \Pi_{i,t}^{*}}{\Pi_{i,t}^{**} - \Pi_{i,t}^{*}} \tag{3}$$

where $\Pi_{i,t}^{*}$ and $\Pi_{i,t}^{**}$ mean the agent i's profits for Nash and Cooperative Equilibrium, respectively.

Price $p_{i,t}$ could be used instead of profit $\Pi_{i,t}$, but we did not employ prices for the metric. In the following, we assumed that there are two agents, i.e., $\mathscr{N} = \{1, 2\}$. Then we think of this condition:

$$\begin{cases} p_{1,t} = p_{1,t}^{*} \\ p_{2,t} = 2\left(p_{2,t}^{**} - p_{2,t}^{*}\right) + p_{2,t}^{*} \end{cases} \tag{4}$$

where $p_{i,t}^{*}$, $p_{i,t}^{**}$ mean agent i's prices for Nash and cooperative equilibrium, respectively. Under the condition, C_t would be 1 if price $p_{i,t}$ were used instead of profit $\Pi_{i,t}$. However, this condition is not price collusion because agent 1 makes the best response.

Usually, collusion happens when they pursue their profits, so the definition in equation 3 is more natural.

5 Simulations

To test the collusion level under some conditions, we performed simulations. Each simulation has 2 or 3 agents with 1,000,000 steps. In the simulation, there was one market handling only one type of goods. Furthermore, there are two types of markets. Both types have the same calculation process, as mentioned before. In one type of market, all agents can access other agents' bidding prices and quantities. In another type of market, all agents cannot access other agents' bidding prices and quantities. It means that the biddings are shielded, and the price would not be revealed even afterword. We call these two types of agents "open market" and "closed market".

Then, at the end of each simulation, we calculated the final collusion metric as:

$$C = \frac{1}{10,000} \sum_{i=0}^{9,999} C_{(1,000,000-i)}. \tag{5}$$

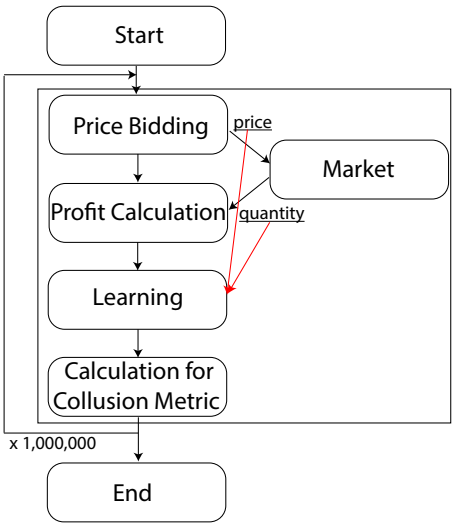

Fig. 2. Simulation flow.

Figure 2 shows the flow of our simulation. After the initialization of the simulation environment and agents, we performed 1,000,000 steps. In every step, agents bid their price to the market, and quantities for their bidding are given by the market. Then, all agents calculate their profits (rewards) and performed reinforcement learning, such as Q-Learning. At the same time, according to the bidding prices, calculations for the collusion metric also happened. Then, after the repeating steps, at the end of the simulation, we calculated the final collusion metric defined in equation 5.

6 Agents

We tested three types of agents, Q-Learning, Sarsa, and Deep Q-Learning agents.

All of them are using one historical price as a state. That is, if the market is an open market, they refer to previous prices of all agents' bids, and if the market is closed, they refer to only the previous prices of themselves. That means if the market is an open market, agent i uses $\{p_{j,t-1}|\forall j \in \mathscr{N}\}$, and if the market is a closed market, agent i uses only $p_{i,(t-1)}$. The reason why we employ only one historical price is the limitation of learning. That is, if we employed multiple historical prices, the state space would be too large to learn and will not converge.

As an action space, i.e., $\mathscr{P}$, we assumed discrete action space $\{p^*_{i.t} + (p^{**}_{i,t} - p^*_{i.t}) \times \frac{i}{10} | i = -1, 0, 1, \cdots, 10, 11\}$ where $p^*_{i,t}$ and $p^{**}_{i,t}$ are Nash and Cooperative equilibrium, respectively. That is -10%, 0%, $\cdots$, 100%, 110% points between Nash and Cooperative equilibrium prices.

We also employed ϵ-greedy and agents were made to choose their action at random using the following probability:

$$\epsilon = \epsilon_{end} + (\epsilon_{start} - \epsilon_{end}) \times r^{\frac{t}{decay}} \tag{6}$$

where t means steps. In our simulations, $\epsilon_{start} = 0.95$, $\epsilon_{end} = 0.0$, $r = 0.95$, and $decay = 1000$. These values are set empirically.

As a reward, we employed a profit coming from the action, i.e., immediate reward. (TD(0))

Other details were varied with each agent as the following. In the following, we note

- $s_{i,t}$: agnet i's state at step t
- $\mathscr{S}_i$: agent i's state space
- $a_{i,t}$: agnet i's action at step t
- $\mathscr{A}_i$: agnet i's action space
- $r_{i,t}$: agent i's reward at stpe t, i.e., the reward at step t coming from $a_{i,t}$ (It differ from the usual notation of reinforcement learning.)

6.1 Q-Learning Agent

Q-Learning is the strategy based on Q-table [28]. Q-table is a table of states vs. actions. Action is decided by:

$$a_{i,t} = \arg\max_{a' \in \mathscr{A}_i} Q_i\left(s_{i,t}, a'\right), \tag{7}$$

where $Q_i(s, a)$ means Q-value of agnet i for state s and action a. Q-values are updated at each step by:

$$\Delta Q = r_{i,t} + \gamma \max_{a \in \mathscr{A}} Q_i\left(s_{i,(t+1)}, a\right) - Q_i\left(s_{i,t}, a_{i,t}\right), \tag{8}$$

$$Q_i\left(s_{i,t}, a_{i,t}\right) \leftarrow Q_i\left(s_{i,t}, a_{i,t}\right) + \alpha \Delta Q. \tag{9}$$

In our simulation, we set $\alpha = 0.10$, $\gamma = 0.95$. α is the learning rate, and γ is the discount rate in reinforcement learning. These values were set empirically.

6.2 Sarsa Agent

Sarsa is State–action–reward–state–action proposed in [18]. This is very similar to Q-Learning, though on-policy learning method. The action is decided by:

$$a_{i,t} = \arg\max_{a' \in \mathscr{A}_i} Q_i\left(s_{i,t}, a'\right), \tag{10}$$

where $Q_i(s, a)$ means Q-value of agnet i for state s and action a. Q-values were updated at each step using:

$$\Delta Q = r_{i,t} + \gamma Q_i\left(s_{i,(t+1)}, a_{i,(t+1)}\right) - Q_i\left(s_{i,t}, a_{i,t}\right), \tag{11}$$

$$Q_i\left(s_{i,t}, a_{i,t}\right) \leftarrow Q_i\left(s_{i,t}, a_{i,t}\right) + \alpha \Delta Q. \tag{12}$$

In our simulation, we set $\alpha = 0.10$, $\gamma = 0.95$. α is the learning rate, and γ is the discount rate in reinforcement learning. These values are set empirically.

6.3 Deep Q-Learning Agent

Deep Q-Learning was proposed in [14]. It is an extended method using a deep neural network. The Q-table in Q-Learning is replaced by a deep neural network such as a multi-layered perceptron (MLP), and this network is called Deep Q-Network (DQN). Action is decided by the same equation as equation 7. For updating Q-values, DQN learning was performed. The loss for learning is defined as:

$$L_{i,t} = \begin{cases} 0.5\left(\Delta Q\right)^2 & (if\ |\Delta Q| < 1) \\ |\Delta Q| - 0.5 & (otherwise) \end{cases}. \tag{13}$$

For learning, we used the batch size of 10,000, i.e., learnings of DQN occurred every 10,000 steps, and the learning results applied current DQN every 100,000 steps.

The details of the MLP we used are:

- Activation function: ReLU.
- The number of layers: 3, 4, or 5 (at random).
- The number of nodes in each layer: 10, 11, $\cdots$, 20 (at random).
- Drop out: 20%.
- Loss function: Smooth L1 loss (as we stated in equation 13).
- Optimizer: RMSprop.
- Learning rate: 10^{-4}.

7 Experiments

In our experiments, we performed 12 patterns of simulations. We change in

- Agent type: Q-Learning, Sarsa, Deep Q-Learning.
- The number of Agents: 2, or 3.
- Market type: Open market, or Closed market.

Then, we calculated C for each pattern of simulation. The number of runs for each pattern is 1,000. Through these experiments, we assess the risk of unintentional collusion. For the assessments, we employed the collusion metric, which is explained before.

8 Results

Tables 1, 2, 3, and figures 3 and 4 show all the results.

Table 1 show the results from Q-learning agents. According to this result, when the agents employ Q-learning as a learning strategy, agents are affected by the number of agents more than whether the market is open or closed. The effect of the closed market is very slight. On the other hand, if the number of agents increases, the collusion metric was drastically dropped down. It means

Table 1. Results for Q-Learning Agents. The values represent collusion metric defined in equation 5 (appears in percentages). Each result written in bold is statistically different from others ($p < 0.01$).

	Open Market	Closed Market
2 Agents	$\mathbf{52.49\% \pm 15.40\%}$	$\mathbf{50.47\% \pm 22.17\%}$
3 Agnets	$39.01\% \pm 10.56\%$	$38.80\% \pm 17.73\%$

Table 2. Results for Sarsa Agents. The values represent collusion metric defined in equation 5 (appears in percentages). Each result is statistically different from others ($p < 0.01$).

	Open Market	Closed Market
2 Agents	$\mathbf{53.72\% \pm 15.18\%}$	$\mathbf{38.25\% \pm 28.86\%}$
3 Agnets	$\mathbf{47.32\% \pm 8.22\%}$	$\mathbf{21.31\% \pm 19.55\%}$

that if the agents employ Q-learning, the number of agents of the competitive markets are very important to avoid unintentional collusion.

Table 2 show the results from Sarsa Agents. The significant difference from the case of Q-Learning agents is the effect of closed markets. When the agents employ sarsa as a learning strategy, the difference between cases of open market and closed markets is very significant. The differences in the collusion metric, which the market type made, was supposed to contribute are around 20%. This difference is bigger than the effect of the number of agents. However, the number of agents is also contributing to the collusion levels. In this result, all patterns are significantly different from others.

Table 3. Results for Deep Q-Learning Agents. The values represent collusion metric defined in equation 5 (appears in percentages). Each result is not statistically different from others ($p < 0.01$).

	Open Market	Closed Market
2 Agents	$58.96\% \pm 29.87\%$	$58.66\% \pm 28.99\%$
3 Agnets	$57.15\% \pm 24.96\%$	$55.69\% \pm 25.02\%$
(4 Agents)	$53.90\% \pm 22.61\%$	$53.80\% \pm 21.88\%$

Table 3 shows the results from deep Q-learning agents. In this table, we also put the results from the case with four agents on the markets. It is because deep Q-learning agents can deal with the more dimensional state. Different from table-based reinforcement learning such as Q-learning or sarsa, deep Q-learning uses the neural network to estimate the Q-value for each action based on the current state. This enables the processing of more dimensional data because there are fewer parameters in learning models. The results are interesting. Both effects of a closed market and the number of agents are comparatively small. Basically, closed markets reduce the collusion metric, and the more agents there

are, the less collusion metric is. However, the differences among each pattern are very slight. In this result, the case of open market and closed market is not statistically different regardless of the number of agents. Moreover, some other pairs of results are also not statistically different.

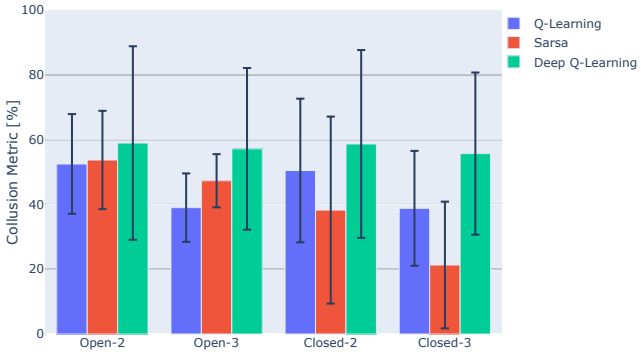

Fig. 3. Bar graph of all results sorted by market type and the number of agents. Blue, red, and green bars correspond to the results from Q-Learning, Sarsa, and Deep Q-Learning in tables 1, 2, and 3, respectively. Open-2, Closed-2, Open-3, and Closed-3 represent the results from when the market is an open market and has two agents, that from when the market is closed market and has two agents, that from when the market is an open market and has two agents, and that from when the market is closed market and has three agents, respectively. Black ranges represent the standard deviation of each result.

Figure 3 show the sorted results, and figure 4 shows the box plot of all results. According to these figures, deep Q-learning agents show comparatively significant results on the collusion metric regardless of the market settings or the number of agents. On the other hand, the performance of Q-learning agents and sarsa agents is not as high as deep Q-learning agents. This clearly shows that outperforming learning methods like deep Q-learning have significant risks on unintentional collusion.

Moreover, it is interesting that sarsa was weak for closed markets to collude. Sarsa is one of the on-policy reinforcement learning and a more realistic strategy than off-policy reinforcement learning. It is because the Q-value of sarsa is calculated by using realized actions. Thus, we assumed that it affects the results in closed markets. That is, unlike off-policy strategies such as Q-learning, sarsa doesn't always pursue the best results. Under the closed markets, it is supposed to be essential that multiple agents pursue the best outcome because they cannot know each others' actions strategy.

It cannot be appropriate to compare the case when the number of agents is 2 and 3 because Nash and Cooperative equilibrium are not the same for each case. However, basically, more agents are on the market, less collusion level agents

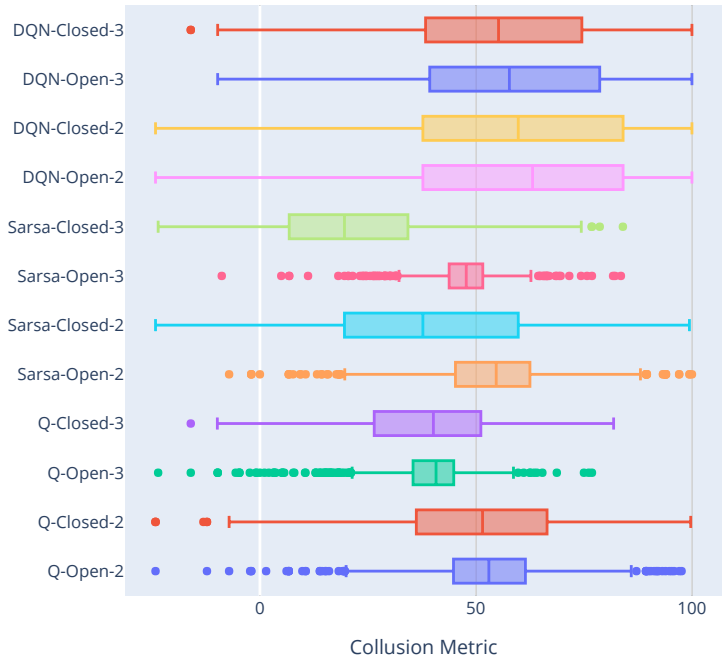

Fig. 4. Box plot of all results.

make. Especially when the performance for unintentional collusion is limited, i.e., Q-Learning and Sarsa cases, the effect of the number of agents can be bigger.

Interestingly, whether the market is an open market or a closed market did not make as big differences in the case of deep Q-learning as Q-learning or sarsa agents. Certainly, there is no big difference between the case of the open and closed market when there are only two agents on the market because agents can estimate prices of each other through their corresponding price and quantity. However, when there are three or more agents on the market, in our consideration, there should be some difference between the case of open market and closed market because they cannot estimate others' bidding price. So, we assumed that there is a good chance that these three types of agents cannot learn correctly because information of profit for each bidding price is included in the learning model only via rewards. Then, to make it clear, we tested another Deep Q-Learning model, which included the historical profit as the state for inputting into the DQN. And if the market is an open market, the historical profit, including others' profit as well as others' historical bidding prices, are used in the state.

Table 4 shows the new Deep Q-Learning model results. These results indicate there is no significant effect of whether the market is an open market or a closed market. This result suggests that our original deep Q-learning agents learned rewards correctly.

Table 4. Results for Deep Q-Learning Agents with Additional Features. The values mean the collusion metric defined in the equation 5 (appears in percentages). Each result is not statistically different from others ($p < 0.01$).

	Open Market	Closed Market
2 Agents	$58.91\% \pm 29.48\%$	$59.54\% \pm 28.98\%$
3 Agnets	$55.93\% \pm 25.21\%$	$55.17\% \pm 24.78\%$

9 Discussion

In this section, we present general discussions of our results. The results in the previous section demonstrate some interesting suggestions as follows.

First, the result that deep Q-learning agents outperformed the other agents is easy to be expected. The fact that Deep Q-Learning outperforms the Q-Learning is according to some previous works. Moreover, it is interesting that sarsa was weak for closed markets to collude. Sarsa is one of the on-policy reinforcement learning and a more realistic strategy than off-policy reinforcement learning. It is because the Q-value of sarsa is calculated by using realized actions. In equation 8, $\max_{a \in \mathscr{A}} Q_i\left(s_{i,(t+1)}, a\right)$ is adopted for evaluation of future state in Q-learning. However, in equation 11, $Q_i\left(s_{i,(t+1)}, a_{i,(t+1)}\right)$ is adopted in Sarsa. The latter is a more realistic strategy. Thus, it affects the results in closed markets. That is, unlike off-policy strategies such as Q-learning, sarsa does not always pursue the best results. Under closed markets, we believe it is essential that multiple agents pursue the best outcome to collude because they cannot know each others' actions. In terms of learning strategy, as future work, we should try some other architecture like A2C. Moreover, in this study, we employed homogeneous agents in each simulation. However, this assumption does not accord to the actual situation of supply chains. Thus, as future work, we should address this problem by employing heterogeneous agents in simulations or other.

Second, we discuss that the more agents on the market, the less collusion level the market would have. Although this tendency cannot be found statistically when the agents employ deep Q-learning, this result seemed to be normal and easy to be expected. So the number of agents is important to avoid unintentional collusion in the real market.

Thirdly, interestingly, the closed market has no significant effect on the collusion level when agents employ Q-learning or deep Q-learning. One reason we can think of here is that even in an open market, it is still difficult to estimate realized rewards (profits) due to existing uncertainty from other agents' behavior. In our settings, rewards depend not only on each agent's strategies but also on other agents' strategies. Hence, the situation is quite complicated, and it is difficult to estimate real rewards in advance. So, this situation makes it difficult not only to learn rewards correctly but also to collude with others unintentionally.

Moreover, all patterns in our experiments had significant deviations. Of course, because there are lots of random processes, it is correct that the results have some fluctuations. However, the deviations in our experiments are very

high. For example, figure 4 shows the existence of cases whose collusion metric was less than 0. This means, in some cases, the agents get less profits than Nash Equilibrium due to fail to employee an appropriate pricing strategy. It means the unintentional collusion we observed in our experiments are not robust. Especially when agents employ deep Q-learning, the deviations are comparatively high. So, this means deep Q-learning shows high performance in unintentional collusion but is not robust. However, at least, there is some tendency that the unintentional collusion happen when agents employe some kind of reinforcement learning and deep Q-learning tends to show high collusion performances.

The point is how to avoid these unintentional collusions and reduce the collusion metric. According to our results, the way to avoid unintentional collusion we found in this paper is only to increase the number of agents in the market. However, in the actual supply chain, the suppliers of each part are limited and it is not a realistic solution to add more suppliers. Moreover, unfortunately, the closed market is not working to avoid collusion in some situations. As future work, we should try some other attempts to avoid unintentional collusion. For example, for the market's demand function in equation 1, adding one term with some randomized factors is one possible solution we guess. That is a in equation 1 should be replaced by a randomized parameter for each step and making it more difficult to estimate the market's demand function via price and quantities. However, this solution is not suitable for real markets. So, as future work, we also address to make the simulation more real than the current ones and estimate the actual economic loss in the real world.

10 Conclusion

In this paper, we addressed the problem of unintentional collusion by auto pricing in a market.

First, we defined a new metric for unintentional collusion using Nash and cooperative equilibria in pricing competition in a market. The market's demand function is defined by the sellers' bidding prices.

Second, we run some simulations to calculate each collusion level under some conditions. We changed in the number of seller agents, agent's strategy learning type, and market type, i.e., price opened or closed markets. According to the results, we found that the number of seller agents and agent's strategy learning demonstrates a significant effect on the collusion level in prices. The more agents are put on the market, the less collusion level happens in the market. Moreover, in some situation, the market type, i.e., open market or closed market, also affect the collusion level.

Furthermore, we found that outperforming reinforcement learning, such as deep Q-learning, shows a high risk of collusion. In the future, we hope to find ways to avoid unintentional collusion other than more agents putting on the market.

Acknowledgment

This work was supported by Council for Science, Technology and Innovation (CSTI), Cross-ministerial Strategic Innovation Promotion Program (SIP), "AI Collaboration for Improved Value Chain Efficiency and Flexibility" (Funding agency: NEDO).

References

1. AI, O.: OpenAI Baselines: ACKTR & A2C (2017), https://openai.com/blog/baselines-acktr-a2c/
2. Bellemare, M.G., Veness, J., Bowling, M.: The Arcade Learning Environment: An Evaluation Platform for General Agents. Journal of Artificial Intelligence Research **47**, 253–279 (2013)
3. Bertrand, J.L.F.: Théorie mathématique de la richesse sociale par Léon Walras: Recherches sur les principes mathématiques de la théorie des richesse par Augustin Cournot. Journal des savants **67**, 499–508 (1883)
4. den Boer, A.V.: Dynamic pricing and learning: Historical origins, current research, and new directions. Surveys in Operations Research and Management Science **20**(1), 1–18 (2015). https://doi.org/10.1016/j.sorms.2015.03.001
5. Calvano, E., Calzolari, G., Denicolo, V., Pastorello, S.: Artificial Intelligence, Algorithmic Pricing and Collusion. SSRN Electronic Journal (2019). https://doi.org/10.2139/ssrn.3304991
6. Castiglioni, M., Marchesi, A., Gatti, N.: Be a Leader or Become a Follower: The Strategy to Commit to with Multiple Leaders. In: IJCAI International Joint Conference on Artificial Intelligence. pp. 123–129. International Joint Conferences on Artificial Intelligence (2019). https://doi.org/10.24963/ijcai.2019/18
7. Fortunato, M., Azar, M.G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., Legg, S.: Noisy Networks for Exploration (2017), http://arxiv.org/abs/1706.10295
8. Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D.: Rainbow: Combining improvements in deep reinforcement learning. In: 32nd AAAI Conference on Artificial Intelligence, AAAI 2018. pp. 3215–3222 (2018)
9. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
10. Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., Silver, D.: Distributed Prioritized Experience Replay (2018), http://arxiv.org/abs/1803.00933
11. Jain, M., An, B., Tambe, M.: An overview of recent application trends at the AAMAS conference: Security, sustainability, and safety. In: AI Magazine. vol. 33, pp. 14–28 (2012). https://doi.org/10.1609/aimag.v33i3.2420
12. Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., Dabney, W.: Recurrent Experience Replay in Distributed Reinforcement Learning. International Conference on Learning Representations pp. 1–15 (2019)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. vol. 2, pp. 1097–1105 (2012)

14. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
15. Mohammad, Y.: yasserfarouk/negmas: Negotiation Multi-Agent System (A negotiation library designed for situated negotiations within business-like simulations) (2019), https://github.com/yasserfarouk/negmas
16. Mohammad, Y., Viqueira, E.A., Ayerza, N.A., Greenwald, A., Nakadai, S., Morinaga, S.: Supply Chain Management World: A Benchmark Environment for Situated Negotiations. In: Proceedings of the 22nd International Conference on Principles and Practice of Multi-Agent Systems (PRIMA2019). pp. 153–169 (2019). https://doi.org/10.1007/978-3-030-33792-6_10, http://link.springer.com/10.1007/978-3-030-33792-6_10
17. Organizers, A.: ANAC2019 - Tenth Automated Negotiating Agents Competition (2019), http://web.tuat.ac.jp/ katfuji/ANAC2019/
18. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. University of Cambridge, Department of Engineering Cambridge, England (1994)
19. Shukla, N., Kolbeinsson, A., Otwell, K., Marla, L., Yellepeddi, K.: Dynamic pricing for airline ancillaries with customer context. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 2174–2182. Association for Computing Machinery (ACM) (2019). https://doi.org/10.1145/3292500.3330746
20. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science **362**(6419), 1140–1144 (2018). https://doi.org/10.1126/science.aar6404
21. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Van Den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of Go without human knowledge. Nature **550**(7676), 354–359 (2017). https://doi.org/10.1038/nature24270
22. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine Learning **3**(1), 9–44 (1988). https://doi.org/10.1007/BF00115009
23. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
24. Tambe, M.: Security and game theory : algorithms, deployed systems, lessons learned. Cambridge university press (2011)
25. Tesauro, G.: Temporal Difference Learning and TD-Gammon. Communications of the ACM **38**(3) (1995)
26. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double Q-Learning. In: 30th AAAI Conference on Artificial Intelligence, AAAI 2016. pp. 2094–2100 (2016)
27. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Frcitas, N.: Dueling Network Architectures for Deep Reinforcement Learning. In: 33rd International Conference on Machine Learning, ICML 2016. vol. 4, pp. 2939–2947. International Machine Learning Society (IMLS) (2016)
28. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8**(3-4), 279–292 (1992). https://doi.org/10.1007/bf00992698

29. Ye, P., Wu, C.H., Qian, J., Zhou, Y., Chen, J., De Mars, S., Yang, F., Zhang, L.: Customized regression model for Airbnb dynamic pricing. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 932–940. Association for Computing Machinery (2018). https://doi.org/10.1145/3219819.3219830