Reinforcement Learning in double auction markets

Reinforced BEYBs

December 2019

Contents

Abstract

In this paper we consider the use of reinforcement learning in double auction markets. We investigate single-agent as well as multi-agent learning. For single-agent learning, we first examine the learning capabilities of reinforcement learning algorithms with a toy problem. We then compare the performances of two classical RL algorithms in a simple market setting. For multi-agent learning, we first investigate how RL agents reach an equilibrium in different information settings. At last, we look into the behavior when agents are grouped together and share information.

1 Introduction & Motivation

Double auction markets have existed throughout history and have been used as a tool to study the convergence of prices in ordinary markets. For us, they are attractive to study because it is relatively easier to reason about optimal strategies in double auction markets, especially if the participants follow a narrow set of policies. In addition to their usefulness for economic theories, double auction markets also form a good environment to develop and compare reinforcement learning algorithms. Using reinforcement learning, we can simulate the convergence of prices, and given the agents model the reality well enough, arrive at conclusions about real world markets.

Hence, the combination of double auction markets and reinforcement learning is attractive for two reasons. First, reinforcement learning allows us to train agents which display intelligent behavior, i.e., the kind of behavior we would expect from strategically thinking humans. This allows us to explain certain properties of double auction markets, such as the optimal strategy that should be followed. Secondly, we use the double auction market as a framework for developing agents which are guided by reinforcement learning algorithms. The rest of this paper serves both of these goals.

The rest of this report is organized as follows. In section 2, we give some necessary background information and explain the methodology behind our implementation. In section 3, which is divided into two subsections, we discuss our experiments and the results. The first subsection is on the single-agent learning setting, while the second subsection focuses on the multi-agent setting. In section 4, we discuss general results of the project and possible research directions for the future.

2 Setup

2.1 Double Auction

Double auction is an important trading mechanism. Agents, who take on one of two discrete roles as either (1): buyers $b \in B$ or (2): sellers $s \in S$, participate in a market environment. Each of the sellers are selling a single unit of a product, with a production cost[1] of r_s^-, and ask price $a_s \in \mathbb{R}_{\geq 0}$. Because they want to make a profit, it is reasonable to assume that $r_s^- \leq a_s$. The buyers, on the other hand, each buy a single product, with their available budget[2] of r_b^+, at a bid price $b_b \in \mathbb{R}_{\geq 0}$. For a rational buyer, we assume that $r_b^+ \geq b_b$.

In this setting, the agents submit their offers, i.e. bids b_b and asks a_s, at discrete uniform time steps t. A buyer and seller pair is matched when a buyer's bid is enough to cover a seller's ask $a_s \leq b_b$. Then, a deal price $d_{b,s} \in [a_s, b_b]$ is decided by a matching algorithm.

[1] Also called reservation price

[2] Also called reservation price

Once an agent has been matched with another and a deal price has been set, both of the matched agents stop participating in the game, i.e., submitting offers. The environment resets after a preset number of time steps T, or when there are no more possible matches. Once the environment is reset, all agents are able to submit offers again. A round denotes this time interval from none of the agents being matched to no more matches being possible. A game refers to a collection of K rounds. For the duration of a game, the buyer budgets r_b^+ and the seller costs r_s^- are assumed to be constant. Additionally, a single market setting is assumed to be applied in each game.

2.2 Market Setting

The market setting controls:

- the matching mechanism, which are the rules about matching a buyer and a seller;
- the information setting, which controls what information each agent has access to at each time step.

2.2.1 Matching Mechanism

The matching mechanism we use in our markets is as follows. A buyer and seller has the possibility to be matched only if the bid of the buyer exceeds the ask of the seller, i.e., $a_s \leq b_b$. Clearly, if there are multiple sellers or buyers satisfying this criterion, a specific matching rule has to be implemented. The way this goes is as follows: the highest bidding buyer is matched with the lowest asking seller, the second highest bidding buyer is matched with the second lowest seller and so on.

Once a buyer and seller are matched, the deal price is determined as either:

- the mid-price of the bid and the ask:
$$d_{b,s} := (a_s + b_b)/2 \in [a_s, b_b] \tag{1}$$
- from a uniform distribution between the bid and the ask:
$$d_{b,s} \sim Uniform(a_s, b_b) \tag{2}$$

Although in some auctions typically the buyer's price is chosen to be the final deal price, we pick the mid-price(1) due to the symmetry. We choose not to consider the uniform random distribution(2), due to the unpredictable nature of this deal price function. Therefore, we opt for the mid-price(1) for implementations that use the deal price in determining the reward for the learning agents. Note that using the matching algorithm described above, setting the deal price to be the bid price (or the ask price) would allow for exploits. Indeed, if the bid price is chosen to be the final price, a seller can gain the advantage to obtain the lowest possible price simply by offering to sell at 0, since the matcher matches the lowest seller with the highest bidder.

2.2.2 Information Settings

In order to make a decision for their offer at time step t, each agent $g \in B \cup S$ can make use of information $i_{g,t}$ provided by the information setting.

- Black box setting: Agents see only their own last offers.
- Layer N offer information: In this setting, all agents will be able to see the best (highest) N bids $b_{s,t-1}$ and best (lowest) N asks $a_{b,t-1}$ of the last market round.

$$i_{g,t} = \max_N \{b_{b,t-1}\}_{b \in B} \cup \min_N \{a_{s,t-1}\}_{s \in S}, \tag{3}$$

  where $\max_N$ denotes a subset of the highest N offers from the given set, and $\min_N$ is a subset f the lowest N offers from the given set. If N is equal to the number of agents from one role, this corresponds to the Full Information setting as presented by [1].
- Time information: This setting enhances the settings above by allowing agents to know the current time step t of the market.

2.3 Implementation and design

We implemented the above formal setting in code in the Python 3 programming language. To enable easy reuse of the implementation, we packaged our code in a modular Python package, `dmarket`[3]. This allows for easy replication of our results and use for future projects. Previously, work in this area has been done by [1]. Here, an implementation of the market setting can be found. Our implementation is based on this. However, our implementation is a strict improvement in the following areas:

- Performance: comparatively profiling the code of both implementations show that our code can run anywhere between three or four magnitudes faster.
- Extensive testing: our code has been written with extensive unit tests, with over a 94% code coverage.
- Many built-in agent classes with pre-defined market strategies.
- Built-in support for both state-of-the-art stable-baselines [2] single-agent and RLlib [3] multi-agent library implementations.

2.3.1 Reinforcement learning setup

In reinforcement learning, the learning algorithm attempts to train a model in a given environment. A popular method to structure environments for easy reuse and reproduction is through the OpenAI gym framework [4]. In this

[3] `https://github.com/zhy0/dmarket_rl`

environment, agents have two constants: an observation space and an action space. The action space determines what actions an agent can take, for instance, in the market setting, it is the prices the agent can offer, hence an interval on the positive real line. The observation space determines what the agents 'see' after they have taken an action. In our market setting, this is determined by the information setting we chose. For instance, if we use the 1-layer information setting, the agents would be able to observe the lowest ask and highest bid in the last round, therefore the observation space would be the positive quadrant $\mathbb{R}R^2$.

Although strictly speaking the action space in a real-world market could be a continuous interval, we choose to discretize the action space for our agents to a finite set. The reason is that most algorithms we use (Q-learning variants) require a discrete action space. This means that the action space can be represented as a set $\{0, 1, 2, \dots, n-1\}$, which is in fact what the Gym framework specifies using the `Discrete` class.

Another consideration we make is with regarding the observation space. Generally, if our purpose is to train agents that perform well under different market scenarios, then they should be trained in such a way that it does not matter what the price range is of the given market and perhaps even to the extent that it should not matter whether the agent is a buyer or seller. To achieve this, we perform normalization on both the observation space (inputs) and action space (output) of the agent based on two parameters: the reservation price and a multiplicative factor $0 < \eta < 1$. This factor η determines the minimum respectively maximum price the agent can offer: A seller with factor η and reservation price r_s^- would be able to offer in the range $[r_s^-, (1+\eta)r_s^-)$, whereas a buyer would be able to offer in the range $((1-\eta)r_b^+, r_b^+]$. Given this interval, we discretize this interval uniformly in n bins. Therefore, we map an action $i \in \{0, 1, \dots, n-1\}$ to the offer price $(\beta - \alpha)i/n$, where α and β are the endpoints of the offer interval determined above.

Normalization of the observation space is done using only the reservation price. Suppose an reinforcement learning agent sees a price p given to him by the information setting. Then we normalize the price as follows:

$$\hat{p} := \begin{cases} \frac{r_b^+ - p}{r_b^+} & \text{if agent is a buyer,} \\ \frac{p - r_s^-}{r_s^-} & \text{if agent is a seller.} \end{cases}$$

In this way, an agent trained on a particular reservation price is also able to operate in other markets where his reservation price might be different.

3 Experiments

In this section we discuss the experiments we have run. In general terms, we divide the experiments into single-agent training problems and multi-agent training problems. We start the section with simple single-agent experiments and then later consider more intricate multi-agent scenarios.

3.1 Single-agent

3.1.1 Verifying learning

In this first experiment, we pose a simple challenge to the reinforcement learning algorithm. The situation is as follows: we are in a 1 vs 1 environment where the buyer is a reinforcement learning agent and the seller has a specific strategy. The reservation price is 100 and 50 respectively for the buyer and seller. Both participants are able to see the offers of the opposite side in the previous game round (1-layer offer information setting, since there are only 2 participants). Furthermore, the seller will be able to see the current time step of the market, but the buyer will not. The seller is encoded to have the following strategy:

- If this is the first round, set a count variable c to zero and offer to sell at 100;
- if not, check what the offer is of the buyer: if his offer is higher than 90, increment c and offer to sell at $100 - c$.
- If at some point $c \geq 3$, the seller will throw a large discount and offer 50.

For a buyer to optimize his strategy, he should offer three times something higher than 90 (but lower than 97 to avoid matching prematurely) and then the fourth step offer close to 50 to gain the discount.

Training is done using Stable Baseline's implementation of the DQN algorithm [2]. This algorithm is picked for its simplicity. The parameters are mostly kept to mostly the default. We introduce a larger discount value $\gamma = 0.9$ to stimulate the agent to trigger the correct sequence earlier in the game. Furthermore, exploration is set to be anywhere between 30% and 50% of the time. The results are shown in Figure 1 and 2. In the latter figure, we clearly see that the agent has successfully found the strategy to obtain the reward.

3.1.2 Learning agent against simple heuristics

In this experiment, we make the market setting a bit closer to real world situations, as follows: As information setting, the black box setting is used. There are 5 sellers and 5 buyers, each trying to maximize its own gains as usual. One of the buyers is a reinforcement learning agent, while the remaining 9 agents all follow the same randomized heuristic, which we describe below.

We describe the heuristic for buyers only, as the heuristic for sellers is completely symmetric. A buyer b decides on its next offer o_{i+1} solely by looking at his last offer o_i. Let r_b denote the reservation price (i.e. budget) of buyer b. Given o_i, o_{i+1} is a discretization of the random variable o'_{i+1} which is defined as follows:

$$o'_{i+1} \sim Uniform(o_i, r_b)$$

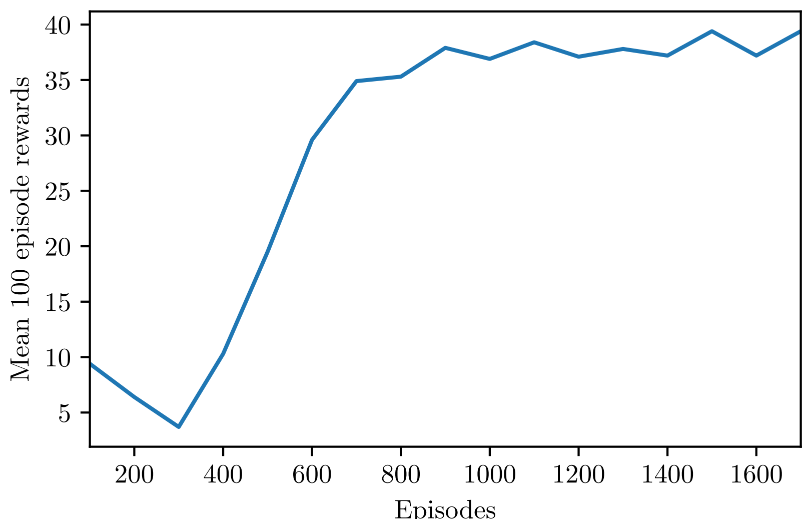


Figure 1: Training in the tricky seller situation. Note that exploration is set to at least 30% so the agent is not using its optimal policy.

This formulation represents a simple negotiation strategy: After every market step, each buyer will increase its bid, while each seller will decrease its asking price.
This experiment also serves as a benchmark to compare two different reinforcement algorithms:

1. Our implementation of the classical Q-learning algorithm
2. Our implementation of the Monte Carlo Control algorithm [5]

Next, we list the parameters of the experiment:

Q-learning:

- Reservation price (i.e. budget) for all buyers $r_b := 200$
- Reservation price (i.e. production cost) for all sellers $r_s := 100$
- Initial offer of all buyers $i_b := 100$
- Initial asking price of all sellers $i_s := 200$
- Time-steps per game $T = 10$

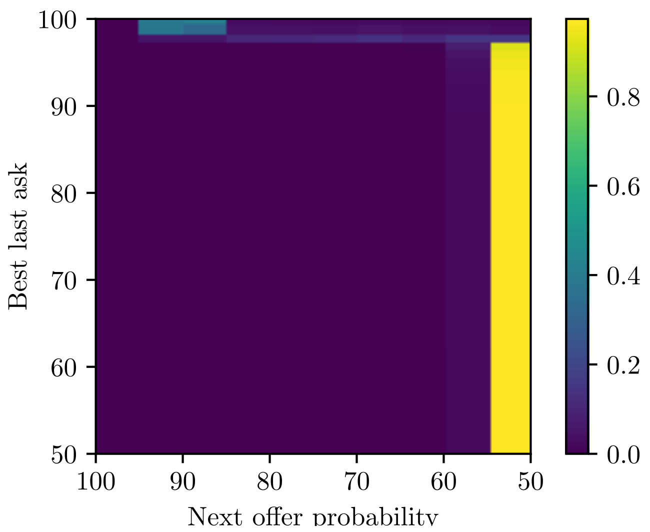


Figure 2: Q-table for learned strategy against the tricky seller. This table is normalized to show probabilities. Conditional on the best ask in the previous round (which is the price the tricky seller is asking since there is only one seller), the plot shows the probabilities of the agent offering the next price.

Since we are in the black box setting, we expect that the RL agent will decide on one value to offer and learn the other agents' behaviour. The state action space is as follows: The last offer of the RL agent is its state. For each state the agent can decide between three actions. It can either decrease it's offer by 25, by 5 or stay at the same price. The non-learning agents follow a variant of the heuristic strategy described above.

This agent is tested in two different settings. First one is a simple 5 sellers 5 buyers with one reinforcement learning agent as a seller.

As we can also observe from the table and the plot the agent decrements its offer by 5 three times and then stays at the offer value of 185 which earns him around 60 percent more rewards then its other seller competitors.

In the second setting there are only 4 buyers and 5 sellers, therefore, in order to get a reward the sellers have to react fast. From our results the reinforcement learning agent had around 20 percent more reward then other sellers. Our agent first decreases its offer by 5 twice and by 25 twice to reach 140 at the end.

Monte Carlo Control:

- Reservation price (i.e. budget) for all buyers $r_b := 100$
- Reservation price (i.e. production cost) for all sellers $r_s := 20$

- Initial offer of all buyers $i_b := 0$
- Initial asking price of all sellers $i_s := 100$

As the astute reader will notice, sellers are at a disadvantage in this setting. The maximum reward that a buyer can earn in a single episode is higher than that which can be earned by a seller. This imbalance will also be reflected in the results of the experiment.

The results for the Monte Carlo Control algorithm is displayed in Figure 3. The huge difference in the average performance is due to the following: Since there are 5 sellers and 5 buyers, even if 8 agents pair up, 1 seller still keeps reducing its asking bid until the end of the game. The Monte Carlo Control algorithm is exploiting this fact and always offering very low bids, knowing that the last agent will accept it.

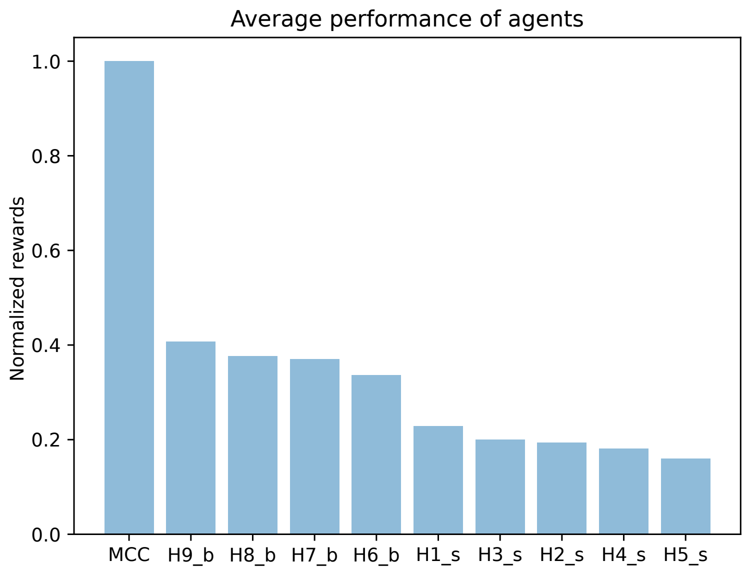


Figure 3: Results for Monte Carlo Control. MCC stands for the agent which is trained with the Monte Carlo Control algorithm. The other bars stand for the agents who follow the heuristic described above. b is for buyer and s is for seller. The disadvantage of sellers is also clear.

3.2 Multi-agent

The behavior of an agent trained in a single-agent environment is fundamentally dependent on the defined environment in which the agent is trained. As a result, how versatile or ‘useful’ the learned policy is depends entirely on the engineered environment. In the previous section we showed that under various different pre-defined market environments (determined by the other agents with fixed policies), a well-trained reinforcement learning agent is capable of exploiting the peculiarities of that market. However, in real world markets, the dynamics are usually never fixed. As shown by experiments with humans, agents in artificial markets learn from their experiences and change their strategies accordingly. It is therefore interesting to see how reinforcement learning agents operate in environments where other agents are also learning.

3.2.1 Reaching equilibrium

In our first multi-agent reinforcement learning experiment, we investigate whether training agents simultaneously will result in them converging to some kind of equilibrium. This numerical experiment is motivated by similar experiments with human players [6]. For this we first consider the following simple setup:

- 3 sellers with reservation price 90,
- 3 buyers with budget 110,
- all agents can offer 20 different offers,
- either black box or 1 layer offer information setting is used, both without time enhancement.

All six agents are trained simultaneously using different policies. There is no communication between the agents (except for that provided by the information setting) and the agents are not aware of the presence of other learning agents. We use RLlib’s multi-agent Deep-Q learning algorithm [3]. The results for offer information setting is displayed in Figure 4.

Clearly, in all four cases with the offer information setting, the agents’ outcome converge to some stable equilibrium, as can be seen from their respective rewards. There is still asymmetry in the plots as the rewards do not converge to a single value (say 10), as one would expect. Instead, we have that either the sellers dominate the buyers slightly or vice versa. Even so, in most cases there is a clear symmetry around the mid price at 10, which reflects the competitive, zero-sum nature of the game.

In the black box setting, the dynamics are entirely different, see Figure 5. Here we observe much more uncertainty. Whereas in the offer information setting, the agents are quick to reach an agreement, the black box setting is much less stable. Because there is no ‘anchor point’ provided by the market, the agents are completely in the dark and therefore only have their previous experiences to hold on to. In Figure 5, this is reflected in the volatile rewards over training time. We see that at first the agents obtain roughly rewards around the mid-price. However, at around 1500 episodes, the rewards drop dramatically. We speculate that at this stage, all agents simultaneously started to offer more aggressively, which resulted in many games where the agents did

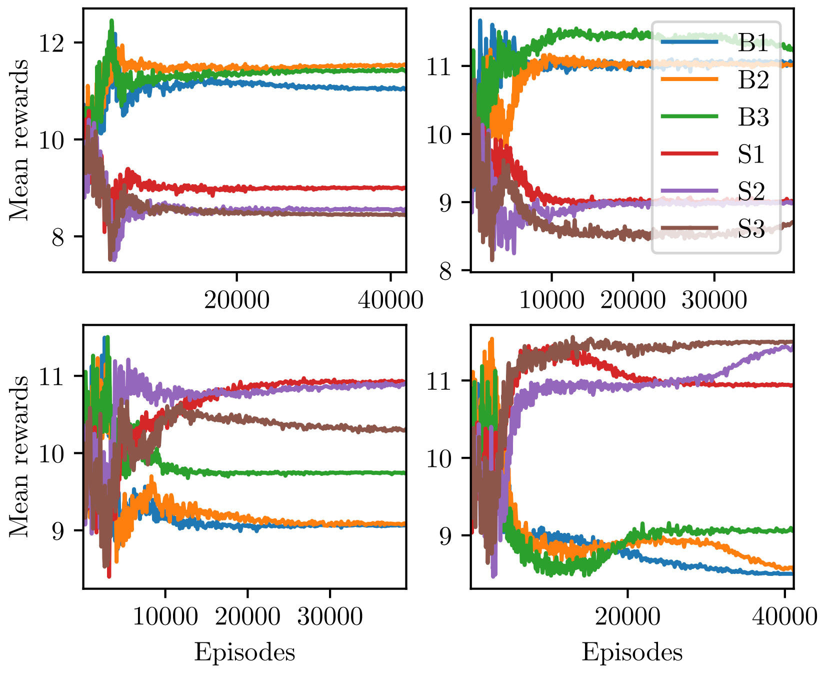


Figure 4: Multi-agent 3 vs 3 in 1-layer offer information setting.

not match within the maximum allowed time steps (30 in this case), hence the low rewards. After a while, agents learn to adjust their strategies to become less aggressive. This process seems to happen multiple times on smaller scales, as can be witnessed by the peaks in rewards and episode lengths between episodes 3000 and 4000.

3.2.2 Grouped agents

Related to the previous experiment, we consider the case when one side of the market is governed by a single agent. In real world markets, this could correspond to the scenario where the sell-side is dominated by a single corporation or organization. For example, one such instance could be the government of a particular country selling licenses to the use of certain radio frequency bands. In such case, all goods come from the government and the buyers act independently and compete to buy a single license.

We hypothesize intuitively that in such a scenario, given the sellers' monopoly situation, the coordinated sellers will be able to raise the prices arbitrarily and

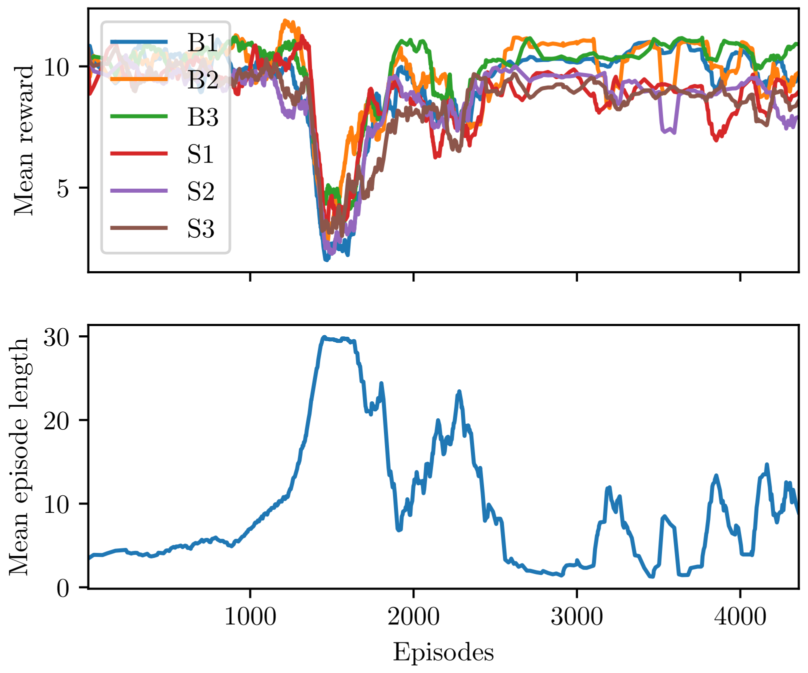


Figure 5: Training 3 buyers vs. 3 sellers under black box setting.

will be able to obtain better deals than the buyers. To test this hypothesis, we model with the following setup:

- 3 independent reinforcement learning buyers with reservation prices 110,
- 3 sellers with reservation price 90 operated by a single reinforcement learning agent,
- the reward for the grouped seller is the sum of the rewards of the individual sellers,
- 1-layer offer information setting.

The result of the experiment is plotted in Figure 7.

Interestingly, we observe to the contrary, that the buyers rather than the sellers dominate the market. This can be seen by their rewards: each individual buyer obtains a reward strictly more than 10, and the three sellers earn combined strictly less than 20 in all cases. We speculate that this happens due to two reasons. First, because the sellers are operated by one agent and this agent

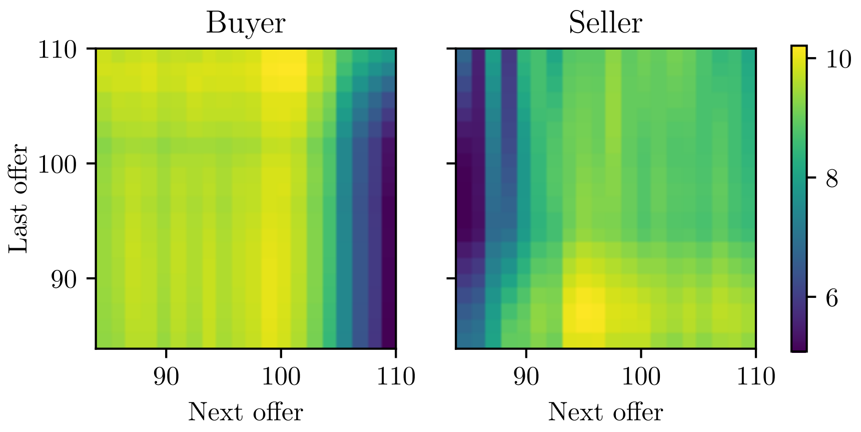


Figure 6: Deep Q-tables of a seller and buyer trained under the 3v3 black box setting.

has the combined action space and observation space of three individuals, the learning process for the learning algorithm could be slower due to the higher dimensionality. As such, individual agents start to 'mature' earlier and place higher demands early on, pushing the grouped seller to offer lower. This effect can be seen in the graphs: The buyers and sellers start off relatively even and then there is a strong decrease for the sellers. Then the graph stabilizes and after a while the sellers push back more, resulting in a slight increase in rewards for sellers. This can be most clearly seen in the top-left figure in Figure 7, where there is a minimum just before 5000 episodes, after which the sellers earn slightly more.

4 Conclusion

In general, this work is a confirmation that reinforcement learning agents are very effective when it comes to learning good policies in small double auction

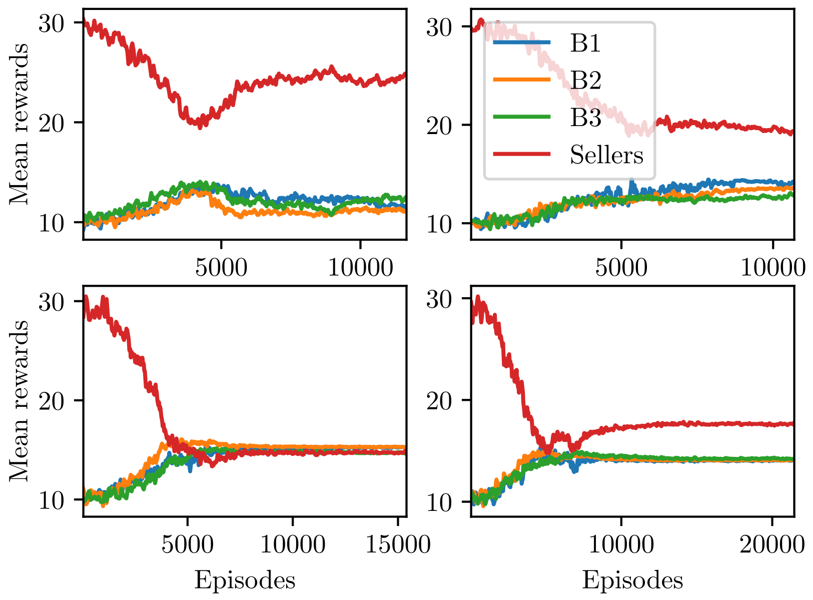


Figure 7: The mean reward per episode is plotted for different training scenarios with grouped sellers. In the graph B1, B2 and B3 are the individual buyers, and the rewards for the sellers is reported together.

market settings. Some future directions may include:

- Learning in larger markets, with total number of agents in the hundreds.
- Training agents in highly asymmetrical settings, such as 5 buyers vs 50 sellers.
- Making the agents more robust, i.e. testing their performance on a family of double auction market settings.

References

[1] T. Asikis, "Reinforcement learning for markets." `https://github.com/asikist-ethz/market_rl`, 2019.

[2] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines." `https://github.com/hill-a/stable-baselines`, 2018.

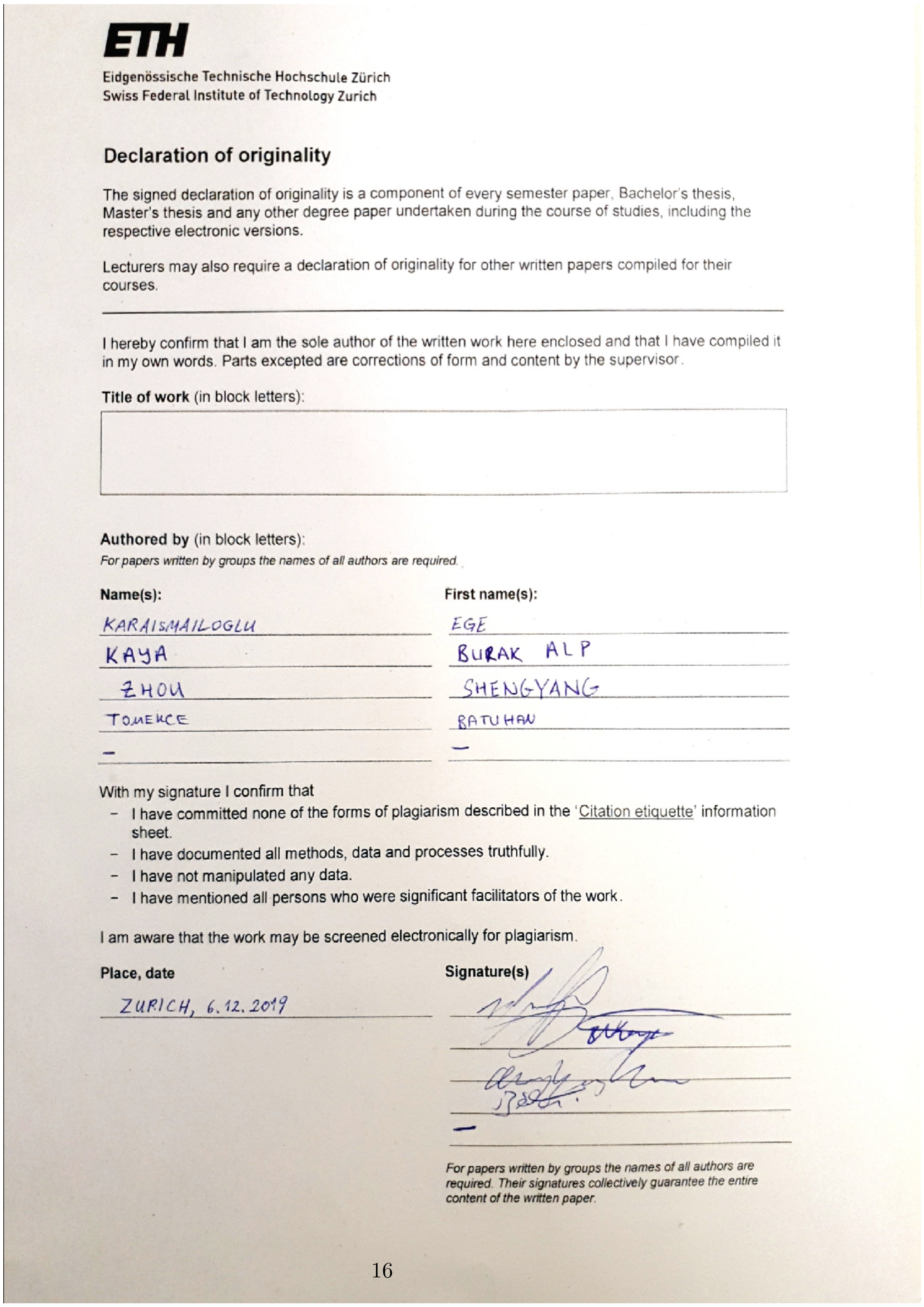
[3] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for Distributed Reinforcement Learning," *arXiv e-prints*, p. arXiv:1712.09381, Dec 2017.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[5] A. G. B. Richard S. Sutton, "Reinforcement learning, an introduction," Feb 2015.

[6] Nax, B. Pradelski, and D. N. Duran, "Feedback effects in the experimental double auction with private information," Feb 2019.

5 Appendix



ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
KARAISMAILOGLU	EGE
KAYA	BURAK ALP
ZHOU	SHENGYANG
TOMEKCE	BATUHAN
–	–

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
ZURICH, 6.12.2019	[illegible]
	[illegible]
	[illegible]
	[illegible]
	–

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.