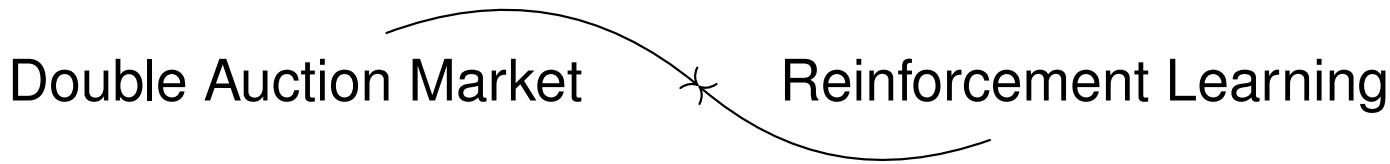
RL in double auction markets

Overview

Introduction



Double Auction

Agents: Buyers $b \in B$, Sellers $s \in S$.

- Each Seller sells a single product, with production cost r_s^- and ask price $a_s \in \mathbb{R}_{\geq 0}$.

$$r_s^- \leq a_s \tag{1}$$

- Each Buyer buys a single product, with budget r_b^+ and bid price $b_b \in \mathbb{R}_{\geq 0}$.

$$r_b^+ \geq b_b \tag{2}$$

- Agents submit their offers b_b and a_s. Deals are made when:

$$a_s \leq b_b \tag{3}$$

Matching Mechanism

Highest Bidding Buyer	Lowest Asking Seller
2^{nd} Highest Bidding Buyer	2^{nd} Lowest Asking Seller
3^{rd} Highest Bidding Buyer	3^{rd} Lowest Asking Seller
...	...

- Deal price = mid price

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: All agents know the best (highest) N bids $b_{s,t-1}$ and best (lowest) N asks $a_{b,t-1}$ of the last market round.

$$i_{g,t} = \max_N \{b_{b,t-1}\}_{b \in B} \cup \min_N \{a_{s,t-1}\}_{s \in S}, \tag{4}$$

  $\max_N$:subset of the highest N offers, $\min_N$:subset f the lowest N offers

- Time information: Agents also know current time step t

Implementation

- Python 3 - Programming language
- Packaged code in a modularly for easy replication
- Repo: `dmarket`[1]
- Repo: `RLFM`[2]
- qAgentPlayGrounds - An easy to use notebook to quickly run experiments

[1] `https://github.com/zhy0/dmarket_rl`
[2] `https://github.com/ekarais/RLFM`

Improvements

Based on `market_rl`[3] However, our implementation is a strict improvement in the following:

- performance: 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 [1] single-agent and RLlib [2] multi-agent library implementations.

[3] `https://github.com/asikist-ethz/market_rl`

Reinforcement Learning Setup

OpenAI gym framework [3] provides a way to structure environments for easy reuse and reproduction. In this environment, agents have two constants:

- Observation space: What agents 'see': (other) offers provided by the information setting
- Action space: What agent 'can do': offers the agents make

Single-Agent: MC Control vs. Heuristic Agents

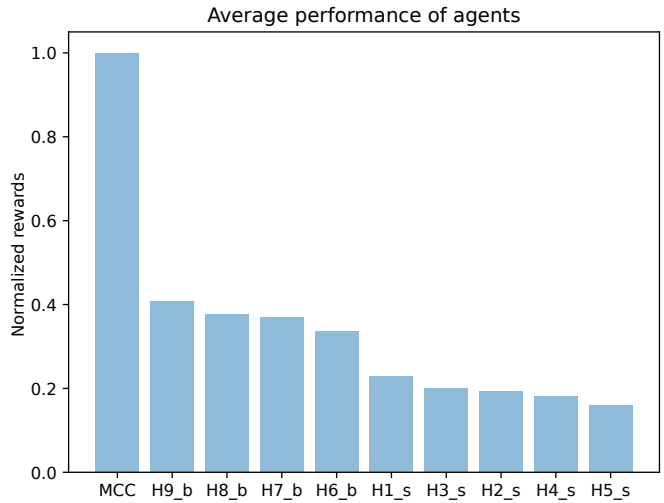


Figure: Results for Monte Carlo Control. MCC stands for the agent which is trained with the Monte Carlo Control algorithm. The other bars stand for the agents who follow the heuristic described above. b is for buyer and s is for seller. The disadvantage of sellers is also clear.

Single-Agent: Tricky Seller - Rewards

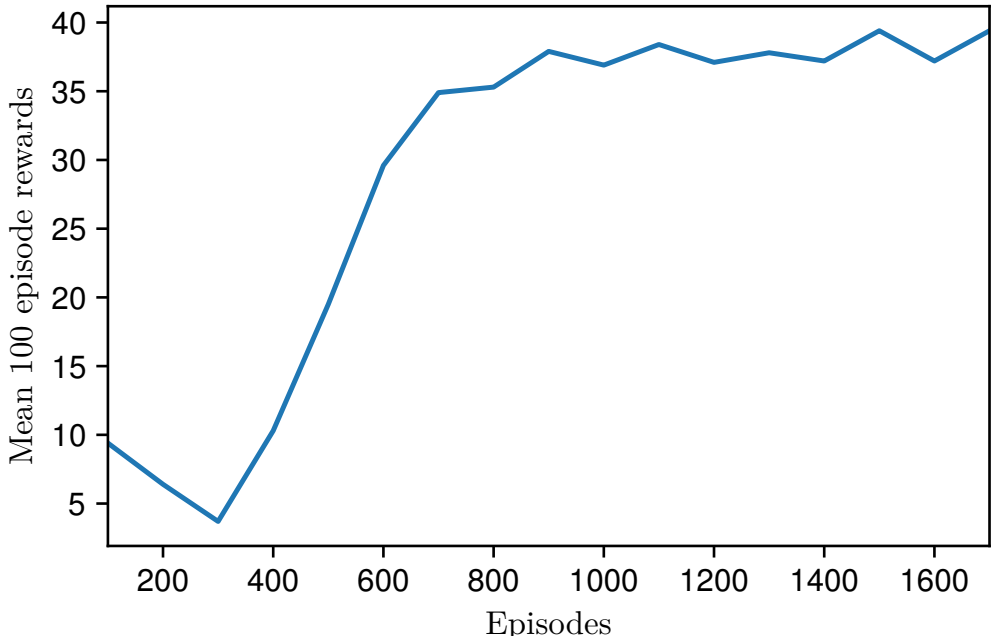


Figure: Exploration is set to at least 30% so the agent is not using its optimal policy.

Single-Agent: Tricky Seller - Q Table

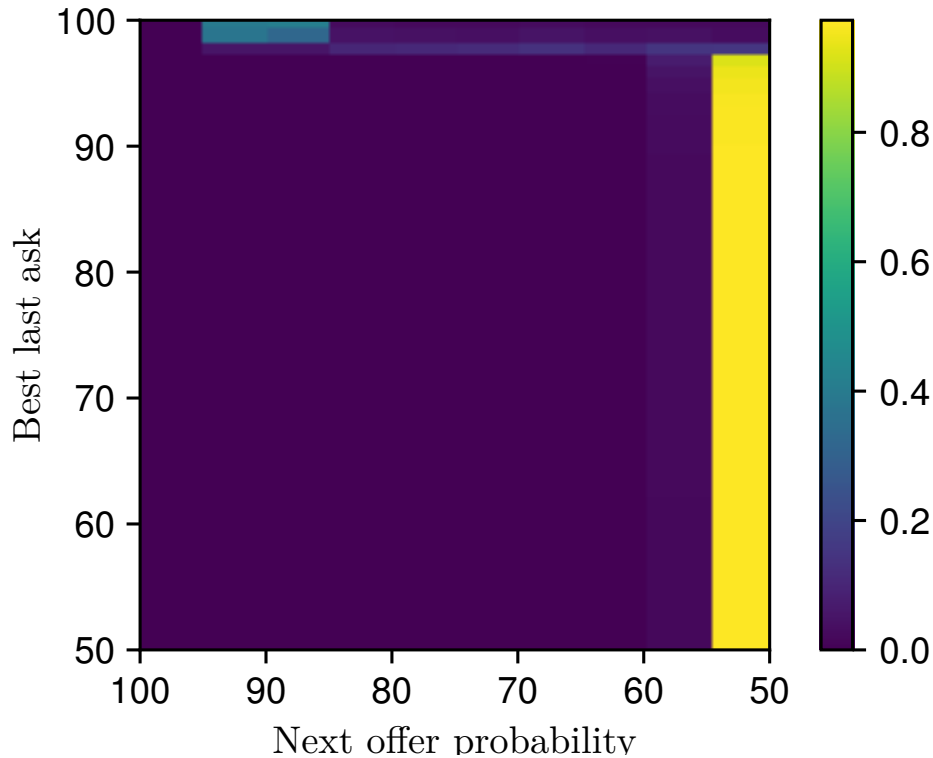


Figure: Q-table for learned strategy against the tricky seller.

Multi-Agent: Reaching Equilibrium

Multi-Agent: Reaching Equilibrium

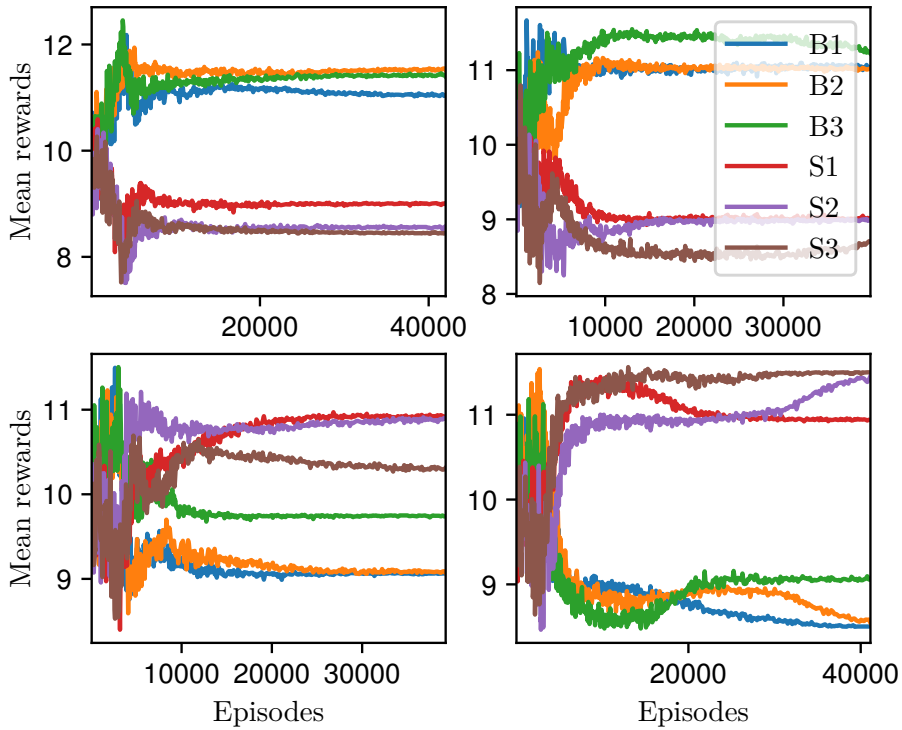


Figure: Multi-agent 3 vs 3 in 1-layer offer information setting.

Multi-Agent: 3v3 BlackBox

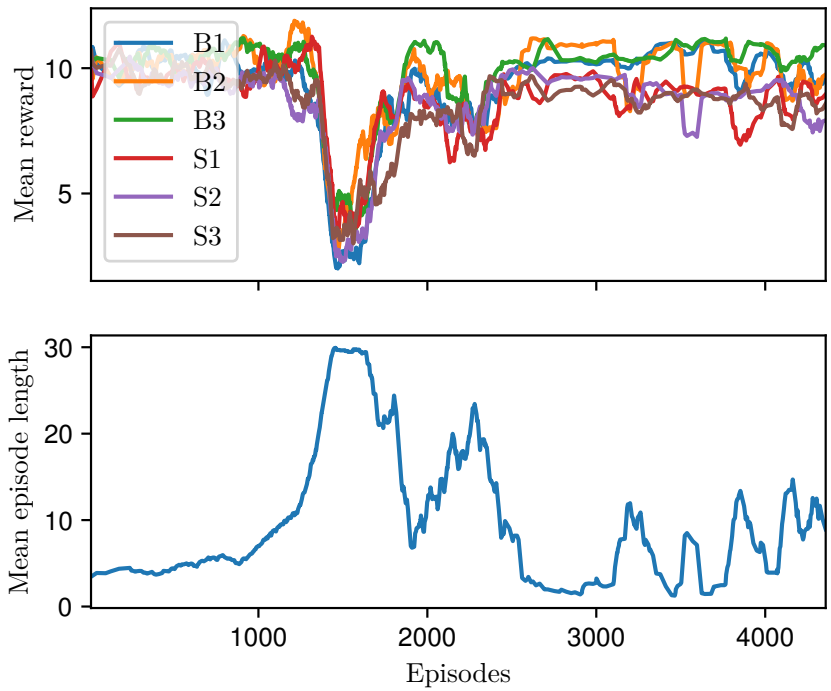


Figure: Training 3 buyers vs. 3 sellers under black box setting.

Multi-Agent: monopoly grouped sellers

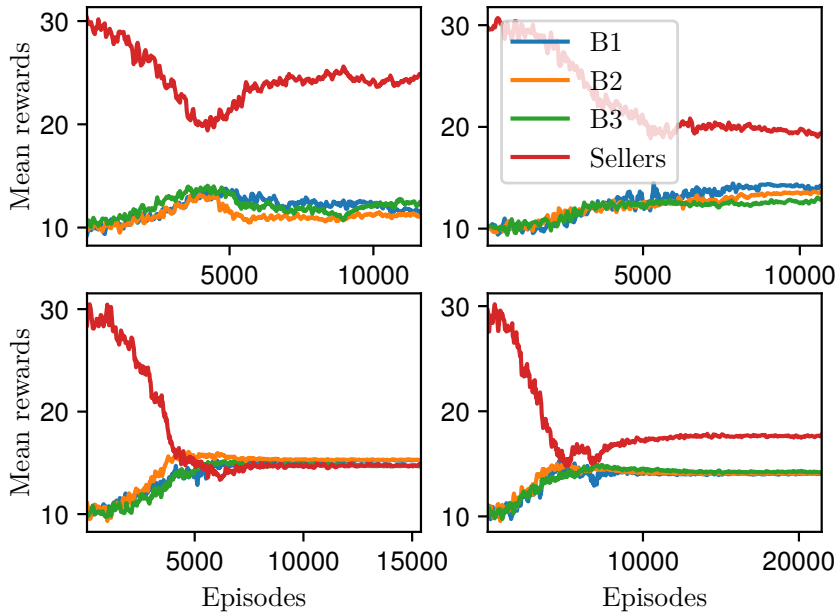


Figure: 3 sellers operated by one RL agent with summed rewards.

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 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

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.

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.

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