Reproduction of Multi-Agent Deep Reinforcement Learning for Liquidation Strategy Analysis

Janet Wang
Department of Computer Science
University of Virginia
Charlottesville, US

yw4fm@virginia.edu

Keywords

Stock liquidation, multi-agent reinforcement learning.

1. MOTIVATION

1.1. Definition of Liquidation

Stock liquidation takes place when an individual or a group of shareholders sell their shares in the open market for ready cash. These shareholders place sale orders through stockbrokers. If the sale price is higher than the decision price, the investors makes a profit. If the sale price is lower than the decision price, the liquidation results in a capital loss, also known as implementation shortfall. For most of the time, investors will experience implementation shortfalls because the market price of the stock drops as more shares become available in the market. The priority of stockbrokers is to minimize implementation loss and manage risk levels for investors. Therefore, effective trading is of great importance.

1.2. Definition of Reinforcement Learning

Researchers have adopted financial and mathematical models to develop liquidation strategies. They simplify the problem into a single-agent scenario with many limitations [1]. Later, researchers have adopted reinforcement learning in the gaming field to improve trading strategies [2]. Reinforcement learning is a type of non-supervised learning such that agents are bound to learn from its interaction with the environment. A reinforcement learning agent learns from consequences of actions and selects it actions based on its past experience. The agent aims to learn to select actions that maximizes the accumulated reward over time. This training approach has been applied by many researchers to the liquidation problem to investigate the optimal strategy for stockbrokers.

However, due to dynamic and interactive nature of the market, researchers soon realize that a single-agent environment doesn't give out the full picture of the stock market. It is more likely to have multiple agents and stockbrokers liquidating the stock at the same time. Every order of liquidation may impact on the market and add uncertainty to the market condition. Therefore, researchers of the original paper used past applications of multi-agent reinforcement learning in the fields of gaming and robotics [3] and analyzed the liquidation problem under a multi-agent reinforcement setting.

2. BACKGROUND

2.1 Optimal Liquidation Problem

To narrow down this broad problem into a simple scenario, we consider a stockbroker at risk aversion level λ aims to sell X shares within a T-day timeframe. The timeframe is usually decided between the shareholders and stockbrokers before the liquidation so that all shares will be liquidated in a certain amount of time. For simplicity, we say that the stockbroker may only place 1 trade each day. The expected implementation shortfall $E(X)$ happens on each trading day and is the difference between the initial stock price before the liquidation and the execution price of the trade on that day [4]. The goal of the liquidation problem is to find an optimal trading trajectory $\overrightarrow{x_t}$ so that the stockbroker is able to minimize the implementation shortfall. The trading trajectory $[x_1, x_2, \ldots, x_t]$ is a vector of remaining shares required to liquidate and x_t is the shares left on Day t.

2.2. Trading Environment Simulation

Based on the belief that trades done by stockbrokers will impact on the market, Bao and Liu trained agents in an environment simulated by the Almgren and Chriss market impact model [5]. The Almgren and Chriss model simulate the execution of portfolio transaction with the goal of minimizing a combination of volatility risk and transaction cost raised from permanent and temporary market impact.

For a simple linear cost model, Almgren and Chriss explicitly constructed the efficient frontier of stockbrokers in the space of time-dependent liquidation strategies. The setting of Almgren and Chriss market impact corresponds to most of the assumptions made in Bao and Liu's liquidation problem. The only difference is that we consider the volatility risk as a fixed constant and only minimize transaction cost raised from permanent and temporary market impacts. The simple linear cost function per share is as follows:

Price on Day K under permanent and temporary impact

$$P_k = P_{k-1} + \sigma\tau^{1/2}\xi_k - \tau g\left(\frac{n_k}{\tau}\right), k = 1, \ldots, N$$

$$P_k - P_{k-1} = \sigma\tau^{1/2}\xi_k - \tau g\left(\frac{n_k}{\tau}\right), k = 1, \ldots, N$$

where σ represents market volatility of the stock and ξ_k are random variables to simulate the discrete arithmetic random walk. $\sigma\tau^{1/2}\xi_k$ quantifies the randomness of the market. $g\left(\frac{n_k}{\tau}\right)$ is a function of the average rate of trading, n_k is the number of shares sold from market close on Day $K-1$ to Day K and $\tau = \frac{T}{N}$ is the time length of each trade. $\tau g\left(\frac{n_k}{\tau}\right)$ reflects the negative impact of liquidation on market price. $P_k - P_{k-1}$ is then the implementation shortfall on each share.

2.3. Liquidation as MDP

After setting up the price change, we model the liquidation as a Markov Decision Process to solve the problem. The Markov process defines that the future state only depends on the present state and does not depend on the past. It is a mathematical framework widely used in modeling decision-making problems where the outcomes are partly random and partly controllable [6].

This framework is especially applicable to the liquidation problem because the market volatility is random while the impact form agents is controllable.

To begin the process, we first define a single agent on Day K before extending to investigation of multi-agents:

- *State*: $s_k = [r, m, l]$ where r is the log-return on shares sold, m is the remaining number of trades normalized by the total number of trades, and l is remaining number of shares normalized by the total number of shares. r captures price information of the previous state at t_{k-1}. For a state to be Markov, the probability of next state should only depend on the current state and the past states
- *Action*: a_k represents the how much the agent sells as a fraction of remaining shares. a_k is between 0 and 1
- *Reward*: $R(s_k, a_k) = U_k(x_k) - U_{k+1}(x_{k+1})$ defines the reward function where x is the trading trajectory. The reward function is the marginal change in utility between t_{k-1} and t_k. Since Algren and Chriss model aims to construct the efficient frontier, the utility of the agent is used to rationalize the process behind picking an action. The utility function can be decomposed into:

  $$U(x) = E(x) + \lambda V(x)$$

  $E(x)$ factors in the impact of market volatility and the cost of trading of the stockbroker, which includes the expected the implementation shortfall. $V(x)$ is dependent on the liquidation amount and the market volatility. $U(x)$ aligns with the behavior of a rational market player. When the market is volatile, the agent liquidating a large amount on that day generate a large $\lambda V(x)$. A risk-neutral agent with high λ will have greater utility than a risk-averse agent.
- *Policy:* $\pi(s_k)$ is the probability distribution of selling percentage a_k of remaining shares at state s_k. It is a probability distribution assigned to a set of actions. Highly rewarding actions will have a higher probability and vice versa.

Obeying the Markov Property at each state, we want to generate an optimal sequence of actions $[a_1, a_2, \dots, a_T]$ or an optimal sequence of policy such that the agent's implementation shortfall is minimized. The goal of this Markov Decision Process is to minimize the implementation shortfall embedded in the utility functions under rewards. We can simply translate action vector $[a_1, a_2, \dots, a_T]$ by subtracting it from $[1,1, \dots, 1]$ to get the vector of remaining shares to solve the optimal liquidation problem.

3. RELATED WORK

The above training setting referenced work from multiple researchers. The reinforcement learning algorithms comes from the research of Yang [7]. The problem of optimal liquidation strategy is investigated by the impact model that agents liquidate assets completely in a given time frame by Almgren and Chriss. The paper also quoted other researches related to deep reinforcement learning algorithm, including Omindshafiei [8], Wang [9] and Lillicrap [10].

4. CLAIM / TARGET TASK

We have studied the liquidation problem from the perspective of a single agent following the Markov Decision Process in an Almgren-and-Chriss market.

To bring practicality and accurately simulate the market, we expand the single-agent setting to a multi-agent setting. Then, we will use several experiments to prove two theorems that conclude the necessity of using multi-agent reinforcement learning and justify our approach to the problem.

After justifying our improvement, we will look into the relationships between agents by redefining their relationships into cooperation and competition. We will analyze how implementation shortfalls vary in these two relationships. This will help investors to decide whether they should hire more than one stockbroker to help them liquidate the shares and whether multiple investors liquidating at the same time will make the process more costly.

Finally, we aim to derive an optimal liquidation strategy for one stockbroker in the multi-agent environment to minimize his implementation shortfall. This strategy will hopefully help this stockbroker to outperform its peers and the investor to make the best out of the liquidation.

5. "WHY" WITH INTUITIVE FIGURE

As shown in Figure 1, the intuition behind this paper is that the trading market of a particular stock will have multiple stockbrokers with similar objectives of liquidating their shares and minimizing their implementation shortfalls. One scenario could be that several investors are liquidating their shares and competing against each other for lower implementation shortfall. Another scenario would be that only one investor is liquidating shares, and that investor hires multiple stockbrokers to sell a certain percentage of his shares.

Both of these scenarios drive demand for researchers to apply multi-agent reinforcement learning to learn about behaviors of stockbrokers, reaction of the market and upgrade the optimal trading strategy to minimize the implementation shortfall.

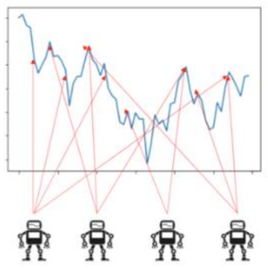


Figure 1 Multiple agents will impact the stock market and affect each other's selling cost

6. PROPOSED SOLUTION

Bao and Liu proposed a solution to further extend the single-agent reinforcement learning to a multi-agent setting so that we can incorporate high level complexities of the system.

There are several benefits of extending into a multi-agent reinforcement learning setting:

6.1. *Adapt to high market complexity*

The main benefit of using a multi-agent reinforcement learning for liquidation problem is that it avoids hard coding and mathematical models. The agents will learn the trading strategy on its own and adapt to the complex market conditions created by other agents liquidating at the same time. This brings greater optimality than human traders since these agents evaluate observations and pricing information unbiasedly.

6.2. *Derive a practical liquidation strategy*

Moreover, this setting leads to real-world application. In the past when we adopt a single-agent liquidation strategy, we barely simulate the trading environment of the stock market. Investors will usually hire multiple stockbrokers to liquidate different portions of their shares to dissipate the risk of one stockbroker leading the liquidation. If in extreme scenarios such that the target company shows strong signal of bankruptcy, multiple investors might file liquidation simultaneously and compete to minimize their liquidation shortfall. It would be risky and nearly impossible for stockbrokers to follow the trading trajectory generated from the single-agent trading environment since the model ignores the impact from other stockbrokers in the game. The trading trajectory is still highly dependent on human judgement.

After moving to a multi-agent level, the trading trajectory is closer to a replacement of human traders and brings the efficiency of applying reinforcement learning to solving liquidation problems. However, the multi-agent setting in this paper assumes that in incorporates all potential behaviors of other agents and that stock market has no noise on the orders generated by the players oversimplifies the real world scenario. But the paper still builds a foundation of a more complicated environment for future researchers to find the most accurate liquidation strategy in future studies.

7. IMPLEMENTATION

We take the single-agent environment as a starting point.

As shown in Figure 2, The multi-agent setting inherits all properties from the single-agent setting defined in Section 2.3 and makes two changes to the setting. The environment will return additional information, noted as “observation”, of other agents’ action at each time step and this limited information observed from the market is fed in to the states of the agents. Additionally, we redefine the reward functions while we compare cooperative and competitive relationships between two agents.

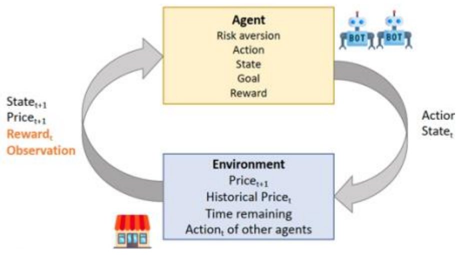


Figure 2 Visualization of Multi-agent Reinforcement Learning

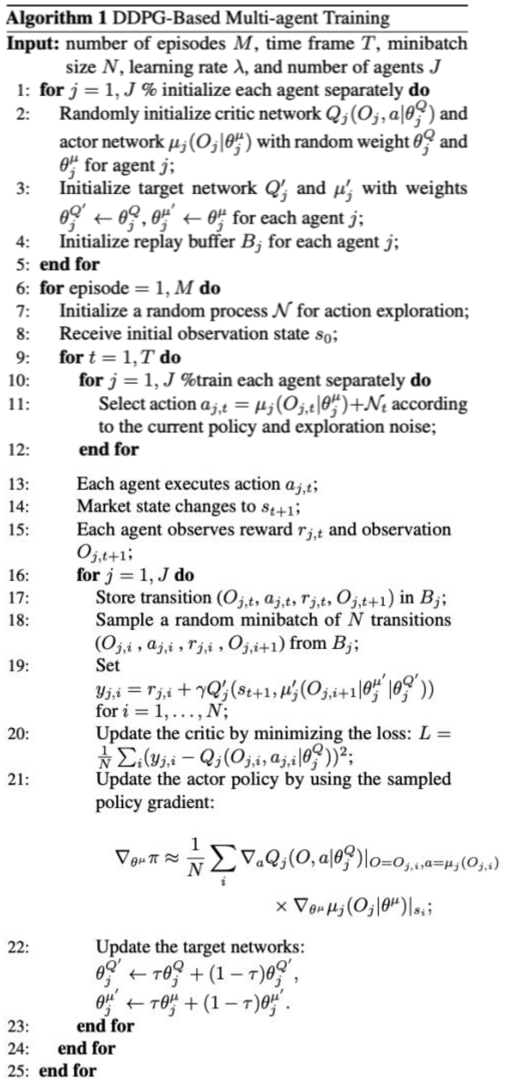
7.1. *Multi-agent Reinforcement Learning Setting*

As we feed states observed from the environment to each agent, each agent predicts actions and return actions and states to the environment. The multi-agent setting with J agents is specified as follows:

- *State*: $s_k = [r_{k-D}, \dots, r_{k-1}, r_{k-1}, m_k, l_1, \dots, l_{J,k}]$
- *Action*: a_k represents the selling fraction of remaining shares. a_k is between 0 and 1, same as before
- *Reward*: $R(s_k, a_k) = U_k(x_k) - U_{k+1}(x_{k+1})$ remains the same as well for general cases without considering the competitive and cooperative relationships.
- *Policy:* $\pi(s_k)$ is the probability distribution of selling percentage a_k of remaining shares at state s_k.
- *Observation: O* Each agent can see what others are acting as well as their sale price, which provides an information vector of $O_{j,k} = [r_{k-D}, \dots, r_{k-1}, r_{k-1}, m_k, l_{j,k}]$. However, each agent only observes limited state information, which means that they don’t know each other’s remaining shares.

7.2. *Implementation of Actor-Critic Model*

Next, Bao and Liu introduces the Actor-Critic model to implement reinforcement learning [11]. The Actor-Critic model splits the model into two: one for computing an action based on a state and one to produce the Q values of actions. These two models participate in a game where they both get better in their own roles as time passes. This Actor-Critic model merges the bright sides of both value-based and policy-based methods. The value-based methods are more efficient and steadier while the policy-based methods are faster for convergence [12].



```
Algorithm 1 DDPG-Based Multi-agent Training
Input: number of episodes $M$, time frame $T$, minibatch
       size $N$, learning rate $\lambda$, and number of agents $J$
 1: for $j = 1, J$ % initialize each agent separately do
 2:     Randomly initialize critic network $Q_j(O_j, a|\theta_j^Q)$ and
        actor network $\mu_j(O_j|\theta_j^\mu)$ with random weight $\theta_j^Q$ and
        $\theta_j^\mu$ for agent $j$;
 3:     Initialize target network $Q'_j$ and $\mu'_j$ with weights
        $\theta_j^{Q'} \leftarrow \theta_j^Q, \theta_j^{\mu'} \leftarrow \theta_j^\mu$ for each agent $j$;
 4:     Initialize replay buffer $B_j$ for each agent $j$;
 5: end for
 6: for episode $= 1, M$ do
 7:     Initialize a random process $\mathcal{N}$ for action exploration;
 8:     Receive initial observation state $s_0$;
 9:     for $t = 1, T$ do
10:         for $j = 1, J$ %train each agent separately do
11:             Select action $a_{j,t} = \mu_j(O_{j,t}|\theta_j^\mu) + \mathcal{N}_t$ according
                to the current policy and exploration noise;
12:         end for
13:         Each agent executes action $a_{j,t}$;
14:         Market state changes to $s_{t+1}$;
15:         Each agent observes reward $r_{j,t}$ and observation
            $O_{j,t+1}$;
16:         for $j = 1, J$ do
17:             Store transition $(O_{j,t}, a_{j,t}, r_{j,t}, O_{j,t+1})$ in $B_j$;
18:             Sample a random minibatch of $N$ transitions
                $(O_{j,i}, a_{j,i}, r_{j,i}, O_{j,i+1})$ from $B_j$;
19:             Set
                $y_{j,i} = r_{j,i} + \gamma Q'_j(s_{t+1}, \mu'_j(O_{j,i+1}|\theta_j^{\mu'}|\theta_j^{Q'}))$
                for $i = 1, \dots, N$;
20:             Update the critic by minimizing the loss: $L =$
                $\frac{1}{N}\sum_i (y_{j,i} - Q_j(O_{j,i}, a_{j,i}|\theta_j^Q))^2$;
21:             Update the actor policy by using the sampled
                policy gradient:
                $\nabla_{\theta^\mu}\pi \approx \frac{1}{N}\sum_i \nabla_a Q_j(O, a|\theta_j^Q)|_{O=O_{j,i}, a=\mu_j(O_{j,i})}$
                $\times \nabla_{\theta^\mu}\mu_j(O_j|\theta^\mu)|_{s_i}$;
22:             Update the target networks:
                $\theta_j^{Q'} \leftarrow \tau\theta_j^Q + (1-\tau)\theta_j^{Q'}$,
                $\theta_j^{\mu'} \leftarrow \tau\theta_j^\mu + (1-\tau)\theta_j^{\mu'}$.
23:         end for
24:     end for
25: end for
```

Figure 3 Algorithm of DDPG Training

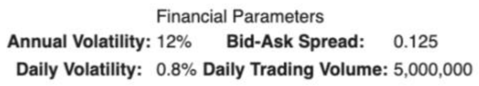
Deep Deterministic Policy Gradient algorithm, as shown in Figure 3, is an example of an Actor-Critic model. The critics will learn the Q-value and update policy parameters. The actors will bring the advantage of computing continuous actions without the need of Q-value functions. As DDPG learns continuous actions, we will use DDPG to generate an optimal strategy. With training skills given by DDPG, the agent will learn from trial and error and find the optimal trading trajectory that minimizes trading cost.

8. DATA SUMMARY

As we simulate in a multi-agent reinforcement learning setting, we set up the financial parameters in the stock market and the parameters of the agent. The financial parameters will remain the same throughout all experiments. We will alter the parameters of agents accordingly by each scenario we analyze.

8.1. Trading Market Setup

We assume that there exists a stock priced at $50 with daily trading volume at 5,000,000 shares. The two stockbrokers aim to liquidate 1,000,000 shares in total throughout the 60-day time frame. This is a reasonable and practical assumption because the total amount of liquidation accounts for 25% of the average daily trading volume, which adds liquidity to the stock but doesn't have a dominating pricing power on the share price. Investors are unlikely to liquidate multiples of average daily trading volume to accelerate the price drop and exacerbate the shortfall implementation.


Financial Parameters

Annual Volatility:	12%	**Bid-Ask Spread:**	0.125
Daily Volatility:	0.8%	**Daily Trading Volume:**	5,000,000

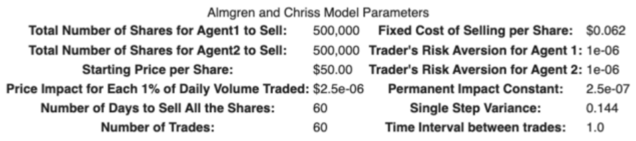
Figure 4 Market parameters summary

Random market noise and other players will still have an impact on the share price. The annual volatility of the stock is 12%. According to the S&P 500's annualized standard deviation from 1926 through 2017, the median standard deviation was 12.5% [13]. Thus, it is reasonable for us to assume a 12% volatility that is representative of any stock in the S&P 500. The bid-ask price is 0.125 and the daily volatility of the stock is 0.8%. The average daily movement in the stock market is between -1% and +1% [14], justifying our default parameter.

These market conditions work as a baseline and will remain constant for all future experiments. While these market parameters are mostly average parameters from the S&P500, the conclusions from our experiments are able to generalize to most liquidation problems, fulfilling the target task we defined in Section 4. Figure 4 is a summary of the market conditions discussed above.

8.2. Trading Market Setup

After setting up the market environment, we look into the agents. The parameters of the agents, as summarized in Figure 6, can be split into two parts – controlled and independent variables.


Almgren and Chriss Model Parameters

Total Number of Shares for Agent1 to Sell:	500,000	**Fixed Cost of Selling per Share:**	$0.062
Total Number of Shares for Agent2 to Sell:	500,000	**Trader's Risk Aversion for Agent 1:**	1e-06
Starting Price per Share:	$50.00	**Trader's Risk Aversion for Agent 2:**	1e-06
Price Impact for Each 1% of Daily Volume Traded:	$2.5e-06	**Permanent Impact Constant:**	2.5e-07
Number of Days to Sell All the Shares:	60	**Single Step Variance:**	0.144
Number of Trades:	60	**Time Interval between trades:**	1.0

Figure 5 Agents parameters summary

- The first part are default parameters that we keep constant in all scenarios. Fixed cost of selling per share is $0.062. Referring back to the Almgren and Chriss model, the permanent market impact constant is fixed at $2.5e^{-0.7}$ and the temporary impact for each 1% of Daily Volume traded is $2.5e^{-0.6}$. A single step variance is 0.144. Every agent will only place 1 trade order per day within the time frame
- The independent variables are total number of shares for Agent 1 and Agent 2 to sell, and risk aversion levels for each agent. The total number of shares of Agent 1 and 2 will always sum up to 1,000,000 shares.

We will redefine the independent variables accordingly as we run each agent or agent pair in the experiments to get one or multiple arrays of implementation shortfalls and trading trajectories. Figure 5 is a summary of the agents' parameters discussed above.

9. EXPERIMENT RESULTS

We conducted four experiments to complete our target tasks - verify the necessity of the multi-agent reinforcement learning and derive an optimal trading trajectory.

The first two experiments proved mathematical conclusions in the reinforcement setting to demonstrate necessity of multi-agents. The latter two experiments explored agent relationships and derived optimal trajectories for agents.

*9.1. Experiment 1: Optimal shortfall (**Theorem 1**)*

The first theorem Bao and Liu brought up is in a multi-agent environment with J agents, where each agent has X_j shares to sell with in a given time frame T, the total expected shortfall is larger than or equal to the sum of the expected shortfall that these agents would obtain if they are in a single-environment.

$$\sum_{j=1}^{J} E(X_j) \leq E(\sum_{j=1}^{J} X_j)$$

We will demonstrate the above inequality by training three agents A, B_1 and B_2 in our multi-agent model. We set the risk level of all three agents at $\lambda_A = \lambda_{B_1} = \lambda_{B_2} = 1e^{-6}$ to eliminate the impact of risk level on implementation shortfalls. We first train Agent A in a single-agent environment liquidating 1,000,000 shares all by itself and get an array of expected implementation shortfalls. We then reset the agent parameters and train Agent B_1 and B_2 in a multi-agent environment with $X_{B_1} = 300{,}000$ and $X_2 = 700{,}000$. We get two arrays of expected implementation shortfalls of Agent B_1 and B_2 and through addition a sum of implementation shortfalls of two agents.

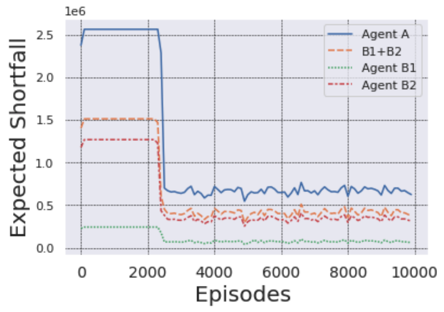


Figure 6 Expected shortfalls of A, B_1 and B_2

We graph the fourth shortfalls in Figure 6 and observe that the expected implementation shortfall of Agent A is consistently higher than the sum of expected implementation shortfall of Agent B_1 and B_2 at every episode. Therefore, Theorem 1 is proved in the reinforcement setting as well.

9.2. *Experiment 2: Agent Interaction (**Theorem 2**)*

After proving that the expected implementation shortfalls are greater in a single-agent environment, Bao and Liu brought up Theorem 2 claiming that the agent interactions will distort the optimal trading trajectory of two agents. Theorem 2 states that in a two-agent environment where Agent 1 has risk aversion level λ_1 and Agent 2 has risk aversion level λ_2, where $\lambda_1 \neq \lambda_2$, and each of them has the same number of stocks to liquidate, the biased trajectories $x(\lambda_1)$ and $x(\lambda_2)$ and would satisfy that $x^*(\lambda_1) \neq x(\lambda_1), x^*(\lambda_2) \neq x(\lambda_2)$.

We first train two agents Agent A_1 and A_2 separately in two single-agent environments at risk aversion level $\lambda_{A_1} = 1e^{-4}$ and $\lambda_{A_2} = 1e^{-9}$ to set the benchmark for comparison, $x(\lambda_{A_1})$ and $x(\lambda_{A_2})$. We then train Agent B_1 and B_2 each liquidating 500,000 shares at risk aversion level $\lambda_{B_1} = 1e^{-4}$ and $\lambda_{B_2} = 1e^{-9}$ in a multi-agent environment. We set the amount of shares for Agent B_1 and B_2 equal so that amount of shares doesn't impact on the optimal trading trajectories of agents.

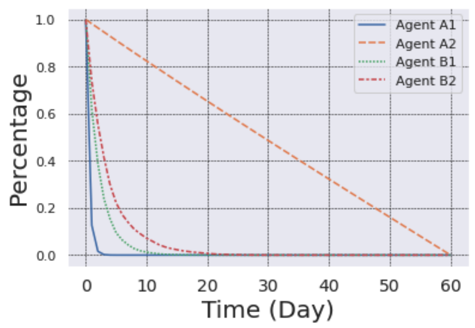


Figure 7 Optimal trading trajectories of A_1, A_2, B_1 and B_2

Figure 7 shows that the less risk-averse Agent A_2 liquidates a fix amount everyday consistently throughout the time frame. The more risk-averse Agent A_1 liquidates quickly in the first few days to avoid future market risk. On the other end, Agent B_1 and B_2 liquidate in two different trajectories from Agent A_1 and A_2. For risk level $\lambda_{A_2} = \lambda_{B_2} = 1e^{-9}$, Agent A_2 liquidates slower than Agent B_2. For risk level $\lambda_{A_1} = \lambda_{B_1} = 1e^{-9}$, Agent A_1 liquidates faster than Agent B_1. Theorem 2 is proven by our experiment.

Experiment 1 and 2 prove the first contribution in Section 6.1 that multi-agent reinforcement learning adapts better to the market complexity. The trading trajectories and expected implementations of agents are closer to reality comparing to results from agents trained in single-agent settings.

9.3. *Experiment 3: Relationship Analysis*

After proving the necessity of using multi-agent reinforcement learning in Experiment 1 and 2, we dive deeper into the relationships between two agents. We will redefine the reward function to embed competitive and cooperative relationship and compare the sum of expected shortfalls with expected shortfalls trained in single-agent environment to evaluate relationships.

- In a cooperation relationship, both agents will be rewarded equally by $R_{1,t} = R_{2,t} = \frac{R_{1,t}+R_{2,t}}{2}$. Agents are motivated to work together to maximize the sum of rewards instead of themselves.
- In a competitive relationship, the agent with higher reward will be rewarded and the other agent will be punished by their difference in reward. If $R_{1,t} > R_{2,t}$, $R'_{1,t} = R_{1,t}$, $R'_{2,t} = R_{2,t} - R_{1,t}$ and vice versa. The reward for Agent 2 is negative, making it a punishment.

We train Agent A_1 to liquidate 1,000,000 shares at risk aversion level $\lambda_{A_1} = 1e^{-6}$ in a single-agent environment; Agent A_2 to liquidate 1,000,000 shares at risk aversion level $\lambda_{A_2} = 1e^{-9}$ as well. We then train Agent B_1 and Agent B_2 to each liquidate 500,000 shares at $\lambda_{B_1} = \lambda_{B_2} = 1e^{-6}$ with competitive reward functions. Finally, we train Agent C_1 and Agent C_2 with cooperative reward functions under same setting as Agent B_1 and Agent B_2.

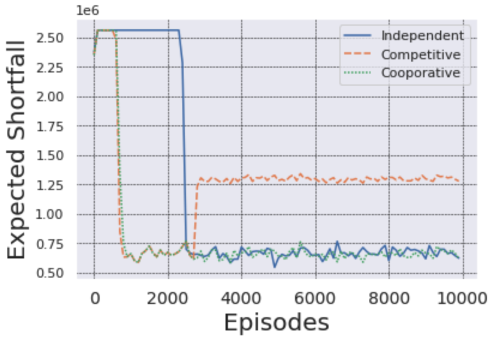


Figure 8 Sum of expected shortfalls of groups A, B and C

From output in Figure 8, expected shortfalls under cooperative relationship decreases sharply after the first 800 episodes and merges with the expected shortfalls of independent agents after 2,200 episodes. The two training trajectories are much in line with each other. We conclude that the cooperative agents are not necessarily better than independent agents trained with independent rewards.

The expected shortfalls under competitive relationship are similar to the shortfalls of the cooperative relationship before the 2,000 episodes. After 2,500 episodes, the expected shortfalls under competitive relationship rises drastically and remains steadily at 1.25 in the later episodes. Two agents learn to minimize expected shortfall faster than other types of agents after learning their competitive nature, and then malignant competition leads to significant increase in shortfall increment.

Experient 3 addresses the second contribution in Section 6.2 that whether investors should reconsider whether it is necessary to hire multiple stockbrokers to liquidate all the shares. This also proves that liquidating simultaneously with other investors will rise of the cost of liquidation.

9.4. *Experiment 4: Strategy Development*

Finally, in the last experiment, we will find a trading trajectory that optimizes the expected shortfall given there are competitors in the environment.

We train Agent A_1 and Agent A_2 to each liquidate 500,000 shares at $\lambda_{A_1} = 1e^{-6}$ and $\lambda_{A_2} = 1e^{-9}$ under a competitive reward function since different investors have different risk aversion levels. This intends to simulate a real-world trading environment where different investors are likely to liquidate at the same time. We will reuse Agent A from Experiment 1 to compare the trajectories and expected implementation shortfalls.

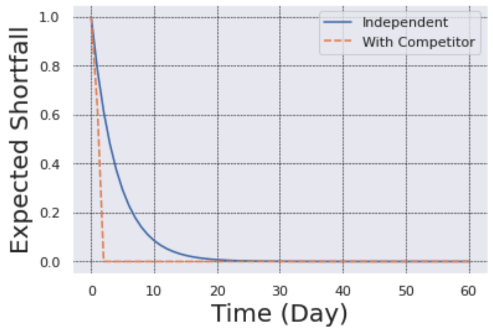


Figure 9 Expected shortfalls of independent agents and agents with competitors

In figure 9, we sum up the expected shortfalls of agents under competitor relationship and compare them with the expected shortfall agent in single-agent environment. Without competitors in an independent environment, Agent A completes the liquidation in 20 days. With competitors, Agent A_1 and A_2 normally sell in 2 days. Agents learn to avoid taking unnecessary risk by selling all shares in quite a short time.

Experiment 4 refers to the contribution mentioned in Section 6.2 of deriving an optimal trading strategy for agent in a multi-agent environment. We conclude that the agent will adopt a faster liquidation strategy and a more aggressive trading trajectory to avoid long-risk in staying in the market.

10. EXPERIMENT ANALYSIS

9.1. *Necessity of using multi-agent*

From Experiment 1 and 2 addressed in Section 9.1 and 9.2, we have demonstrated the necessity of using multi-agent reinforcement learning. For the same amount of shares, implementation shortfall is greater if it is liquidated by one agent than multiple agents. More agents have a negative impact on the liquidation process for investors. It would be more preferrable for stockbrokers to liquidate when there are no other stockbrokers playing in the market.

Agents in multi-agent environment factors in extra information from observing other agents in the environment when deciding their trading trajectory. This proves that observation will alter the trading trajectory and using a multi-agent training environment will make a difference in projecting the optimal trading trajectory. If agents are all trained individually in a single-agent training environment, the optimal trading trajectory might not produce an optimal implementation shortfall when applied in real-world scenarios. The interactive nature of the liquidation process emphasizes the necessity of using multi-agents reinforcement learning to derive trading strategy.

9.2. *Analysis of competitive relationships*

We learn from Experiment 3 that competitive relationships raise implementation shortfall for agents in the market. We will then analyze the competitive relationship from the lends of trading trajectories.

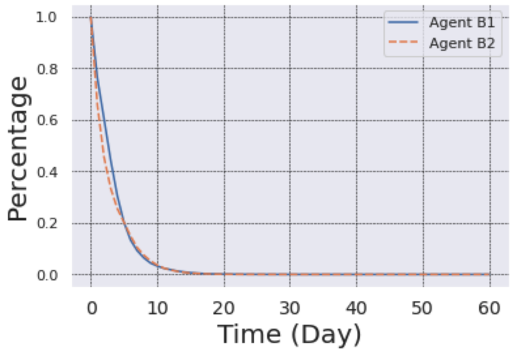


Figure 10 Trading Trajectory of Agent B_1 and B_2 in competitive relationship

From Figure 10, we can tell that two competitive agents learn to sell all shares at similar pace since their trading trajectories mostly overlap.

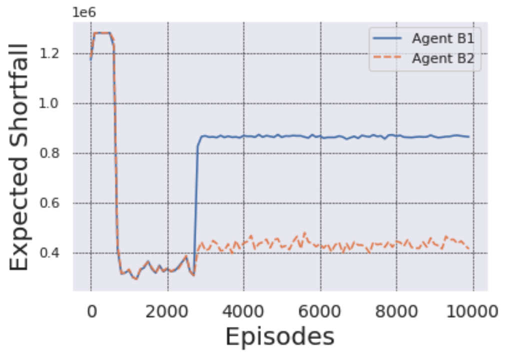


Figure 11 Expected shortfall Agent B_1 and B_2 in competitive relationship

During the first 2,500 episodes, both agents perform well and roughly have same expected shortfalls but Agent B_1 starts to outperform the other significantly at the cost of the Agent B_2. But since the expected implementation shortfalls of both agent increase, none of the agents is winning in the game.

9.3. *Analysis of strategy development*

After learning the agent's behavior in a multi-agent environment, we observe that the agent liquidates at a much faster pace to avoid unnecessary market risk. Since more stockbrokers participate in the trading market to liquidate their shares, more risk arises from drastic price drop. It would be reasonable for the agent to adopt faster trading trajectory so that less shares will be liquidated at lower share price.

11. CONCLUSIONS AND FUTURE WORK

We have shown that multi-agent reinforcement learning presents the dynamic and interactive nature of the problem.

11.1. *Adapt to high market complexity*

We extended the Almgren-Chriss model with reinforcement learning to set up the basis of using multi-agent trading environment to have a better analysis of expected shortfalls. Through Experiment 1 and 2, we learned that the expected shortfalls and trading trajectories of agents in multi-agent environment is different from the ones of agents in single-agent environment. The single-agent environment over-simplifies the dynamic as well as the interactive nature of the stock market. This implies that future researchers should analyze the liquidation problem under the context of multi-agent environment instead of single-agent to reference the real-world scenario.

In the future, we should look into more dynamic factors to simulate our trading environment and specify other factors driving volatility of share price. Possible factors may include news, general strategy and legal complaints. In the price function quantifying the randomness of the price level, we could possibly include economic factors, such as interest rate changes, global financial outlook and inflations, to accurately take account of the price change [15].

11.2. *Derive a practical liquidation strategy*

A cooperative relationship is not necessarily optimal than an independent one. Investors don't necessarily have to employ more than one stockbroker to liquidate their shares in hand since more employment means greater broker's cost.

Other the other hand, a competitive relationship would hurt overall and individual performances of all agents trading in the multi-agent environment. Investors should avoid liquidating at the same time with other investors. If a competitive liquidation is inevitable, stockbrokers should follow the optimal trading trajectory and liquidate faster to avoid market risk from drastic price drop.

Although this liquidation strategy is heavily dependent on the observation from the market and actions of other agents, the market environment itself will make a huge difference on the liquidation pace. In future analysis, we should simulate two cases, the bull and bear, to predict the stock price movements under these two scenarios. Since it would be unlikely for stockbrokers to fully adopt trading trajectories from reinforcement learning, it would helpful to provide human traders with a range of how many shares to sell based on the bull and bear market conditions.

12. REFERENCE

[1] J. Kang, "A Brief History Of Implementation Shortfall: Execution Algorithm: Implementation Cost," Quantitative Brokers, 28-Mar-2018. [Online]. Available: https://quantitativebrokers.com/blog/a-brief-history-of-implementation-shortfall?utm_campaign=QB+General. [Accessed: 11-Dec-2020].

[2] Xiong, Z., Liu, X.-Y., Zhong, S., Walid, A., et al. Practical deep reinforcement learning approach for stock trading. NeurlIPS Workshop on Challenges and Opportunities for AI in Financial Services, 2018.

[3] Almgren, R. and Chriss, N. Optimal execution of portfolio transactions. Journal of Risk, 3:5–40, 2001.

[4] Franciszek Grabski, in Semi-Markov Processes: Applications in System Reliability and Maintenance, 2015

[5] Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., and Whiteson, S. Stabilising experience replay for deep multi-agent reinforcement learning. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 1146–1155. JMLR. org, 2017.

[6] Buehler, H., Gonon, L., Teichmann, J., and Wood, B. Deep hedging. Quantitative Finance, pp. 1–21, 2019.

[7] Yang, H., Liu, X.-Y., and Wu, Q. A practical machine learning approach for dynamic stock recommendation. In IEEE International Conference On Trust, Security And Privacy (TrustCom), pp. 1693–1697. IEEE, 2018

[8] Omidshafiei, S., Pazis, J., Amato, C., How, J. P., and Vian, J. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 2681–2690. JMLR. org, 2017.

[9] Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. Sample efficient actor-critic with experience replay. arXiv preprint arXiv:1611.01224, 2016. Lillicrap

[10] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. ICLR, 2016.

[11] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529, 2015.s

[12] Sergios Karagiannakos, "The idea behind Actor-Critics and how A2C and A3C improve them," AI Summer, 16-Nov-2018. [Online]. Available: https://theaisummer.com/Actor_critics/. [Accessed: 11-Dec-2020].

[13] Person, "Is the Stock Market More Volatile Now Than Ever Before?," Reuters, 26-Feb-2019. [Online]. Available: https://www.reuters.com/article/sponsored/stock-market-more-volatile. [Accessed: 11-Dec-2020].

[14] Financial Samurai, Victoria, TheEngineer, Liam, G. Finja, P. Tiger, Drew, Frankie, S. M. Man, Joe, J. Stojanovski, F. F. Countdown, S. Wealth, S. of Money, Xrayvsn, Bill, F. F. Fanatic, L. from Europe, M. Fireby2023, Snazster, BuddMann, Kevin, Nate, C. D. Kanaya, and Untemplater, "Average Daily Percent Move Of The Stock Market: S&P Volatility Returns," Financial Samurai, 24-Jul-2020. [Online]. Available: https://www.financialsamurai.com/average-daily-percent-move-of-the-stock-market/. [Accessed: 11-Dec-2020].

[15] "What causes share prices to change?," IG. [Online]. Available: https://www.ig.com/en/trading-strategies/what-causes-share-prices-to-change--190128. [Accessed: 11-Dec-2020].