Optimal execution strategy using Deep Q-Network with heuristics policy

1st Tatsuyoshi Ogawa
Division of Transdisciplinary Sciences
JAIST
Ishikawa, Japan
ogawa.tatsuyoshi@jaist.ac.jp

2nd Kei Nakagawa
Innovation Lab
Nomura Asset Management Co., Ltd.
Osaka Metropolitan University
Tokyo, Japan
kei.nak.0315@gmail.com

3td Kokolo Ikeda
Division of Transdisciplinary Sciences
JAIST
Ishikawa, Japan
kokolo@jaist.ac.jp

***Abstract*—The optimal execution problem involves planning a stock execution strategy that minimizes trading costs for a specific quantity of stock over a certain timeframe. To tackle this problem, advanced techniques like Deep Reinforcement Learning (DRL), especially the Deep Q-Network (DQN) which employs deep learning to approximate the Q value function, have been introduced to identify the most efficient execution strategies. However, DRL methods face challenges such as learning instability and the extensive data requirements. Therefore, we propose to use prioritized experience replay and to incorporate a strategy derived from the insights of the financial field into the DQN during learning process. Particularly, we introduce a time-weighted average price (TWAP) strategy that has been proven to be optimal under specific conditions as a heuristic policy. This approach is expected to be able to enhance the stability and performance of policy learning. We have conducted numerical experiments in various noise-prone environments to assess the effectiveness of our approach. The findings indicate that our proposed method consistently outperforms conventional benchmarks by reducing costs in all tested environments.**

***Index Terms*—optimal execution problem, DQN, DDQN, TWAP**

I. Introduction

In practical asset management, the essential process of making investment decisions and executing trades [1], known as optimal execution, is crucial for the efficiency and success of investment operations. Portfolio managers in asset management companies decide on asset allocations, which traders then execute in the market with precision to minimize market impact and execution costs[1]. Given the large volumes of assets these companies often manage [2], executing large orders without significantly affecting market prices is a significant challenge, as large trades can lead to adverse price changes and increased costs [3], [4]. Optimizing trading strategies to minimize transaction costs is increasingly recognized for its importance, with rising trading volumes highlighting the significant impact of these costs on the overall performance of investment portfolios [5], [6]. The concept of the *optimal execution problem* (OEP) aims to identify execution strategies that enable traders to purchase or sell a predetermined quantity of stocks within a specific timeframe, with the goal of minimizing associated transaction costs. This area of study was first addressed in [3], which applied stochastic dynamic programming to derive analytical solutions for OEP under certain conditions, such as assuming that price movements follow a random walk process. Since this study, there has been significant advancement in extending analytical solutions beyond the initial constraints set by [3], to incorporate more complex and realistic models of market price dynamics. Despite these advances, the challenge of deriving optimal execution strategies that are universally applicable remains difficult. The use of stochastic dynamic programming, while powerful, often face limitations when applied to more generalized conditions, as noted in subsequent studies by [4], [7]. These studies emphasize the complex nature of financial markets and the difficulties inherent in capturing the lots of factors that influence stock price movements [8] within analytical models.

Considering difficulties associated with deriving analytical solutions for the OEP, alternative approaches using reinforcement learning have been proposed. These methods, particularly beneficial due to their minimal assumptions about market price dynamics, represent a significant shift from traditional analytical strategies. Notably, [9], [10] employed Q-learning, a foundational reinforcement learning algorithm [11], in their empirical investigations of OEP, demonstrating its applicability in this context. While reinforcement learning has enabled flexible configurations, Q-learning algorithms have other challenges, primarily due to the "curse of dimensionality." This challenge arises from the algorithm's need to maintain and update a comprehensive list of action values (Q-values) for every possible state-action pair, leading to exponential increases in computational and memory demands as the number and complexity of states and actions increase. In response to these challenges, Deep Q-Network (DQN [12]) and its variants, such as the Double Deep Q-Network (DDQN [13]) have been proposed. DQN utilizes deep learning techniques to approximate the Q-function, significantly reducing the computational burden associated with traditional Q-learning. DDQN enhances the accuracy of policy estimation by employing a dual-network architecture, which includes both a Q-network and a target Q-network. This approach effectively mitigates

[1]see https://www.am.mufg.jp/english/service/investment.html for an example of the investment process at an actual asset management company.

the overestimation bias often encountered in DQN, thereby refining the execution strategy development process for OEP [14], [15].

However, to apply DQN and DDQN to the OEP, it is important to recognize the challenges inherent in their learning processes. Both DQN and DDQN need a lot of amount of datasets to achieve acceptable levels of performance. This requirement can be a significant hurdle, especially in financial markets where data quality and availability might vary. As a result, it has been observed that, in practice, the performance of DQN and DDQN can be disappointing and prone to instability [16]. This underscores the need for refinement in these models to enhance their robustness and reliability in OEP.

Therefore, this study aims enhancing the stability and efficiency of DDQN's learning process. Specifically, we propose a new approach that integrates prioritized experience replay and probabilistic action selection, guided by a heuristic policy, during the training phase of the DDQN algorithm. This method is designed to optimize the balance between exploration of new strategies and exploitation of known rewarding actions, aiming to diminish the volume of data required for effective training and speed up the learning convergence. A key element of our approach is the incorporation of the Time-weighted Average Price (TWAP) strategy as the heuristic policy. The TWAP strategy, which has been identified as optimal under certain market conditions [3], serves as a foundational guideline for action selection in the OEP. By embedding this strategy into the training process, we seek to provide a structured framework that aids the DDQN in navigating the complexities of the financial markets, thereby enhancing the stability and reliability of the optimal execution strategies it generates.

The results of the numerical experiments show that our proposed method can consistently reduce the execution costs compared to existing methods across all experimental settings. Furthermore, in experiment with more complex settings, our proposed method can reduce the standard deviation of the execution costs in test samples with 5 trials with different seeds, indicating the stability of the learning process. Additionally, by observing the average reward during learning for these experimental settings, we find that our proposed method exhibit greater stability throughout the learning process.

The structure of this paper is as follows: We begin by reviewing the existing literature on the application of machine learning methods within the domain of asset management. Next, we introduce the problem formulation and propose our methodology. Subsequently, we present the findings derived from our experiments. Finally, we conclude.

II. Related Work

With the development of machine learning techniques, these technologies have gained significant attention for their application in the actual investment process [17], [18].Their ability to analyze vast datasets, identify patterns, and make predictions has opened new avenues for optimizing investment strategies, enhancing investment decision-making, and improving the efficiency of trade executions in asset management [19]. This integration of machine learning is transforming the landscape of asset management business, offering more sophisticated methods for managing assets and navigating complex market dynamics [8], [20].

Machine learning methods offer a significant advantage over traditional time series analysis when dealing with the stock market's nonlinear, noisy, and complex data [8], [20]. These advanced techniques are capable of capturing intricate patterns within the financial data that traditional models might overlook, leading to more effective and reliable market predictions. Promisingly, deep learning models, guided by financial market heuristics and designed with specialized network structures and learning methods, have been proposed to further refine these predictions and enhance investment strategies [21], [22].

For example, the RIC-NN framework [21] is designed to improve stock return predictions by combining heuristics with learning techniques. It employs a nonlinear approach to incorporate multiple predictive factors, uses a ranked information coefficient to prevent overfitting by determining optimal training stop points, and leverages deep transfer learning to apply insights across different markets. [22] enhances deep learning-aided stock prediction by focusing on the prediction of residual factors distribution, vital for risk hedging. It introduces a computationally efficient way to extract residual information for use with prediction models and a new neural network architecture that embeds key financial heuristics like amplitude and time-scale invariance [23].

Building on these studies, this research constructs a deep reinforcement learning method that incorporates heuristics in financial markets, while also focusing on another critical process in asset management: optimal execution.

III. Problem Formulation

Let $P_t \in \mathbb{R}^+$ denote the stock price at discrete time $t \in [1, T+1]$. The optimal execution problem (OEP) seeks to find a strategy for purchasing stocks at each time t such that the total execution cost is minimized when the objective is to purchase $\bar{N} > 0$ stocks by time T. Formally, the OEP can be formulated as follows:

Problem 1 (Optimal Execution Problem (OEP)):

$$\min_{N_t} \ \mathbb{E}[\sum_{t=1}^{T} N_t P_{t+1}] \tag{1}$$

$$s.t., \sum_{t=1}^{T} N_t = \bar{N} \tag{2}$$

The solution N_t^* to this problem is referred to as the optimal execution strategy.

There are cases where the optimal execution strategy can be analytically determined when specific constraints are imposed on the OEP [3]. For instance, if the price dynamics of the stock P_t follow the random walk, and the conditional expectation of the disturbance term ε_t satisfies the following:

$$P_{t+1} = P_t + \theta N_t + \varepsilon_t \tag{3}$$

$$\mathbb{E}[\varepsilon_t | P_t, N_t] = 0 \tag{4}$$

In this case, the optimal execution strategy is obtained as the Time-Weighted Average Price (TWAP) as shown in [3].

Definition 1 (Time-Weighted Average Price (TWAP)): The Time-Weighted Average Price (TWAP) is an execution strategy which uniformly distributes purchases over time and does not depend on time.

$$N_t^* = \frac{\bar{N}}{T} \tag{5}$$

Proposition 1 (Optimlity of TWAP): TWAP is the optimal execution strategy if equation (3) and (4) are satisfied.

Proof 1: see [3] for proof.

Based on the proposition, we use TWAP as a heuristic strategy in the proposed method described in the next section.

IV. Proposed Method

In this paper, we employ prioritized experience replay, an enhancement over the conventional experience replay used in DDQN [13] as described in previous research [14]. Additionally, we incorporate demonstration data guided by a heuristic policy into the learning process. This approach aims to stabilize the learning trajectory and decrease the volume of data needed for effective training.

Below, we will first introduce the foundational concepts of reinforcement learning. In the framework of reinforcement learning, an agent is designed to learn how to take optimal actions within a given environment by receiving and interpreting rewards. Let us define $\mathcal{S} \ni s, s'$ as the set of states, $\mathcal{A} \ni a$ as the set of actions, and r as the reward. The decision-making strategy of the agent, known as the policy, is represented by the probability of choosing a particular action a when in state s. This policy can be mathematically denoted by $\pi(a|s) = \Pr(A_t = a|S_t = s)$, where S_t and A_t are the random variables representing the state and action at time step t, respectively. Following this policy, the agent performs an action, subsequently receiving a reward r as a consequence of its interaction with the environment.

Reinforcement learning is fundamentally concerned with discovering a policy that optimizes the expected (discounted) cumulative reward C_0 beginning from time step $t = 0$. The mathematical formulation of this objective is given by:

$$\max_{\pi} C_0 = \mathbb{E}_{\pi}\left[\lim_{T\to\infty}\sum_{t=1}^{T}\gamma^t r_t\right] \tag{6}$$

In this expression, $\gamma \in [0, 1)$ represents the discount factor, a parameter that modulates the influence of future rewards on the current decision-making process. The discount factor essentially dictates the degree of importance the agent assigns to immediate versus long-term rewards, thereby guiding its strategy in navigating the environment and optimizing its actions for maximal cumulative reward.

In the domain of reinforcement learning, various algorithms have been employed to optimize decision-making processes. Among these, methods leveraging the action value function, denoted as $Q^\pi(s, a)$, have received considerable attention and have been the subject of extensive research [24]. The function

Algorithm 1 DDQN with heuristics policy (with ϵ-greedy)

Input: k: mini-batch size, M: replay buffer size, ω, ω_{fixed}: weights for initial deep network (random), τ: frequency at which to update ω_{fixed}, ϵ, ϵ_h: thresholds for action selection, α: degree of priority
Initialize replay buffer $\mathcal{H}$
for episode $\in \{1, 2, ...\}$ **do**
  for $t \in \{1...T\}$ **do**
    if $p_1 \sim \mathcal{U}(0, 1) < \epsilon$ **then**
      $a_t \sim \pi_{\text{random}}$
    else if $p_2 \sim \mathcal{U}(0, 1) < \epsilon_h$ **then**
      $a_t \sim \pi_{\text{heuristic}}$
    else
      $a_t = \arg\max_a \hat{Q}_\omega(s_t, a)$
    end if
    Play action a_t and observe (s', r)
    Store (s, a, r, s') into $\mathcal{H}$ with maximal priority, overwriting oldest transition if over capacity
    for $j = 1$ to k **do**
      Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
      Compute error
      $\delta_j = \hat{Q}_\omega(s_m, a_m) - (r_j + \gamma \max_{a' \in \mathcal{A}} \hat{Q}_\omega)$
      Update transition priority $p_j \leftarrow |\delta_j|$
    end for
    Calculate loss $l(\omega)$
    Perform a gradient descent step to update ω
    if t **mod** $\tau = 0$ **then**
      $\omega_{\text{fixed}} \leftarrow \omega$
    end if
    $s \leftarrow s'$
  end for
end for

$Q^\pi(s, a)$ quantifies the expected discounted cumulative reward of taking an action a in a state s, under a given policy π. It is expressed as:

$$Q^\pi(s, a) = \mathbb{E}^\pi[C_t|S_t = s, A_t = a] \tag{7}$$

where C_t represents the cumulative reward from time step t. The objective in this approach is to identify the optimal policy π^* that maximizes $Q^\pi(s, a)$ for all state-action pairs. A notable advancement in this area is the development of DQN, which utilize deep neural networks to approximate the action value function $Q^\pi(s, a)$. DQN enhances the stability of learning by employing a technique known as experience replay, allowing for the efficient reuse of past experiences. This approach has been successfully implemented across a diverse range of applications, demonstrating the versatility and effectiveness of DQN in solving complex decision-making tasks [25].

In this research, we utilize DDQN that enhances the estimation of optimal action values through deep learning techniques. DQN begins by initializing the parameters ω_0 of a model $\hat{Q}_\omega$, which serves as an estimator for action values. At each iteration $k = 1, 2, \ldots$, the model undertakes the following steps.

For a subset of experience data (s_m, a_m, r_m, s'_m), randomly selected from the historical dataset $\mathcal{D}$ where $m \in \{1, \ldots, M\}$, the target value q_m for each data point is computed as:

$$q_m \coloneqq r_m + \gamma \max_{a' \in \mathcal{A}} \hat{Q}_\omega(s'_m, a'). \tag{8}$$

Subsequently, the parameters ω are refined through stochastic gradient descent to minimize the loss function, which quantifies the approximation error as follows:

$$l(\omega) = \frac{1}{2M} \sum_{m=1}^{M} (\hat{Q}_\omega(s_m, a_m) - q_m)^2. \tag{9}$$

A distinctive attribute of DQN, and by extension DDQN, is the implementation of experience replay. This technique involves randomly sampling from the historical data $\mathcal{D}$ to update the model, as opposed to using the data in the sequence it was observed. The rationale behind experience replay is to mitigate the strong correlations that often exist between sequentially close data points, thereby enhancing the robustness and effectiveness of the learning process.

One of the challenges with DQN is its tendency to overestimate the value of the objective variable with a high probability. DDQN addresses this issue to a certain extent by altering the estimation process. In the context of DDQN, the evaluation mechanism, originally depicted in equation (8), is redefined as follows:

$$\text{(action selection) } a^* \coloneqq \arg\max_{a' \in \mathcal{A}} \hat{Q}(s'_m, a'), \tag{10}$$

$$\text{(value evaluation) } q_m \coloneqq r_m + \gamma \hat{Q}(s'_m, a^*). \tag{11}$$

This method involves a bifurcation of the estimation process into "action selection" and "value evaluation" phases. To mitigate the overestimation bias, DDQN employs two distinct function approximators: $\hat{Q}_\omega$ for action selection and $\hat{Q}_{\omega_{\text{fixed}}}$ for value evaluation. Here, $\hat{Q}_{\omega_{\text{fixed}}}$ refers to the action value estimate derived from a previously fixed parameter set ω_{fixed}. Despite these improvements, it is important to note that DDQN, much like its predecessor, can still exhibit instability in learning and often necessitates a substantial volume of data to achieve satisfactory performance, as indicated by recent studies [16].

To enhance the efficiency of the learning process, this study advocates for the implementation of prioritized experience replay, as introduced by [26]. Unlike conventional experience replay, where experiences are sampled uniformly at random from the replay buffer without assessing their learning value, prioritized experience replay optimizes the learning process by focusing more on experiences that are regard as crucial for learning and less on those that are not. In this approach, the selection probability of each experience data is determined by:

$$p_v(m) \coloneqq \frac{v_m}{\sum_{m=1}^{M} v_m} \tag{12}$$

where v_m represents the priority assigned to the m-th experience, ensuring that experiences with higher learning value are sampled more frequently. The priority value v_m is defined as:

$$v_m \coloneqq (|\delta_m| + \epsilon)^\alpha \tag{13}$$

based on the magnitude of the approximation error δ_m, which is calculated as:

$$\delta_m \coloneqq \hat{Q}_\omega(s_m, a_m) - q_m. \tag{14}$$

In this formulation, $\epsilon > 0$ is a small hyperparameter introduced to prevent any experience from having a nearly zero selection probability, while α is another hyperparameter that modulates the influence of the priority value. Setting $\alpha = 0$ reverts the selection probability $p_v(m)$ to a uniform distribution, aligning it with the traditional experience replay mechanism. This prioritization strategy aims to make the learning process more targeted and efficient by emphasizing experiences that offer significant learning opportunities.

We assume that the utilization of a near-optimal policy, when available, as a benchmark for exploring the adjacent decision space can significantly stabilize the learning process and diminish the volume of data necessary for effective training. This strategy enables the agent to augment its historical dataset $\mathcal{D}$ with experience tuples (s_m, a_h, r_m, s'_m), where the actions a_h are derived from adhering to a heuristic policy π_h. By incorporating experiences guided by π_h into the learning dataset, the agent can more effectively navigate towards optimal decision-making, leveraging the insights provided by the heuristic policy to expedite its learning trajectory and enhance the overall efficiency of the learning process.

In summary, if ϵ-greedy is adopted as the policy of the agent during learning, the whole algorithm of our proposed method can be written as Algorithm 1.

V. Experiment

In this section, we conduct numerical experiments to evaluate the effectiveness of the proposed method.

A. Experimental settings

Aligned with the framework outlined in Problem 1, we conduct experiments under a unified set of parameters: $T = 10$, $\bar{N} = 20$, $P_1 = 100$, and $\theta = 10$. We also investigate the impact of distinct noise terms ε_t through the following experimental settings:

Setting 1: $\varepsilon_t = 0$.
Setting 2: $\varepsilon_t \sim \mathcal{N}(0, 3^2)$.
Setting 3: $\varepsilon_t \sim \mathcal{N}(3, 3^2)$.
Setting 4: $\varepsilon_t \sim \mathcal{N}(-3, 3^2)$.

Within these experimental frameworks, Settings 1 and 2, characterized by $\mathbb{E}[\varepsilon_t | P_t, N_t] = 0$, suggest that the TWAP strategy is the optimal execution strategy, as supported by proposition 1. Conversely, Settings 3 and 4, where $\mathbb{E}[\varepsilon_t | P_t, N_t] \neq 0$, imply the potential for strategies that may surpass the efficacy of TWAP.

B. Training

The configuration of states, actions, and rewards within experiments was defined as follows:

State:

The state at time t, denoted by s_t, is characterized by

TABLE I: Mean and standard deviation between trials of the reward at test in each setting. 10,000 episodes were tested for each of 5 trials of different seeding.

	TWAP	DDQN	Proposed method
Setting 1	-4200	-4256 $\pm$ 15.1	-4252 $\pm$ 24.6
Setting 2	-4200	-4292 $\pm$ 65.4	-4256 $\pm$ 44.2
Setting 3	-4530	-4667 $\pm$ 90.0	-4588 $\pm$ 78.1
Setting 4	-3870	-3968 $\pm$ 122.9	-3915 $\pm$ 48.2

a triplet including the current time step t, the price P_t, and the remaining shares W_t to be executed.

Action:

The action at time t, represented by a_t, involves selecting the number of shares N_t to be executed at time t, where $N_t \in [0, W_t]$.

Reward:

The reward at time t, r_t is defined as the negative product of the number of shares executed at time t and the price at the next time step, $-N_t P_{t+1}$, reflecting the actual cost.

Regarding the neural network architecture, a fully connected network was employed, consisting of 6 layers with 16 nodes each. The Adam optimizer [27] was used for optimization purposes, with a learning rate set to 0.0001. To ensure the stability of the learning process, the discount factor γ was fixed at 0.99, a decision informed by insights gleaned from preliminary experimental evaluations.

The agent employing the conventional method employs an ϵ-greedy approach with $\epsilon = 0.1$ for action selection during the learning phase. In contrast, the agent in our proposed method opts for an action derived from the TWAP strategy with a 10% probability, as an alternative to the standard greedy action. This is formalized as follows for the action a_t at time t:

$$a_t = \begin{cases} \frac{W_t}{T-t+1}, & \text{with probability } 0.1, \\ \arg\max_{a \in \mathcal{A}} \hat{Q}(s_t, a), & \text{otherwise.} \end{cases}$$

Here, $\hat{Q}(s_t, a)$ represents the estimated value of taking action a in state s_t at time step t. The training regimen spanned over 10,000 episodes, which equates to a total of 100,000 steps, ensuring a comprehensive learning experience for the agent.

C. Results and discussion

Table I presents a comparison of the average rewards obtained at test time for the TWAP strategy, the existing DDQN method, and the proposed method across various experimental settings.

We can confirm that the proposed method was effective in reducing the execution costs associated with the OEP, outperforming the existing DDQN-based approach. Notably, in experimental settings 3 and 4, where TWAP may not represent the optimal strategy, the proposed method demonstrated lower standard deviations in cost, indicating more stable learning outcomes. This stability suggests a consistent performance across different trials, reinforcing the robustness of the proposed method.

Moreover, the proposed method was found to devise strategies that surpassed the performance of TWAP in these settings. Evidence of this was seen in several trials where the proposed method achieved higher average rewards at test time over 10,000 episodes, surpassing the benchmarks set by TWAP. This emphasizes the proposed method's capability not just to replicate heuristic policies effectively but to explore and identify superior strategies that align closely with such heuristic guidelines.

In the subsequent analysis, we move onto experimental settings 3 and 4, where the TWAP strategy might not represent the optimal approach for execution. Figure 1 illustrates the reward trajectories for both the existing DDQN and proposed methods throughout the training phase. To smooth out fluctuations and provide a clearer view of the trends, moving averages with a window size of $n = 100$ were calculated. The average outcomes across five distinct seed values are depicted by solid lines, while the variability in these results is captured by the inclusion of error bars representing standard deviations.

Figure 1a shows the rewards of the existing method and the proposed method in the experimental setting 3. Comparing the two method might appear difficult at first glance; however, a focused examination of the error bars provides valuable insights. These error bars illustrate that the proposed method demonstrates enhanced stability in the learning process compared to its existing counterparts, as evidenced by the reduced variability in rewards across training sessions.

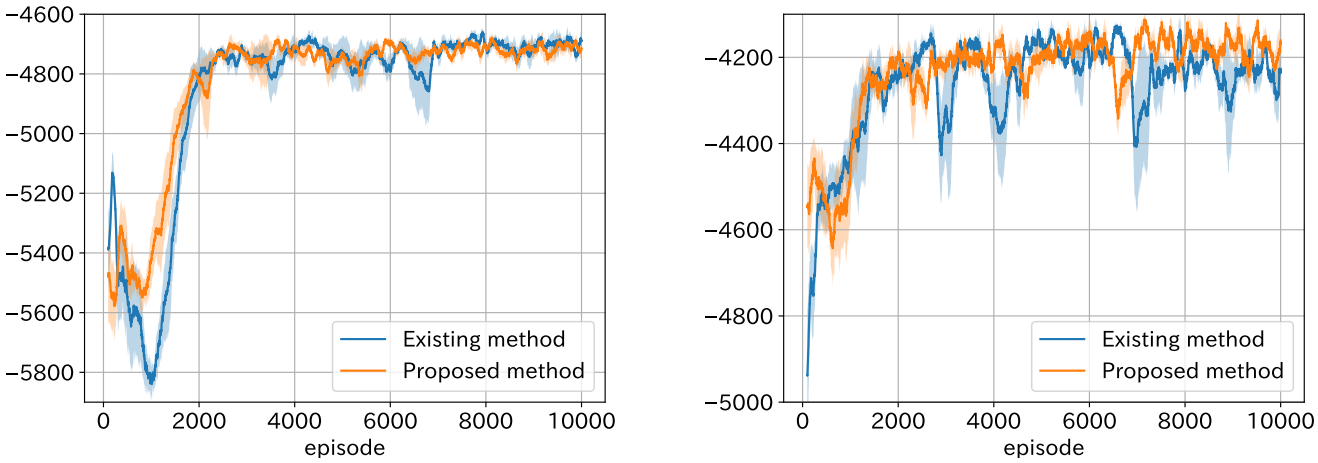
Furthermore, Figure 1a also shows the performance of the proposed and existing methods in experimental setting 4. The solid lines represent the average rewards over time, with the proposed method generally keeping higher rewards than the existing method. This superiority is complemented by the error bars, which further clarify to the proposed method's increased learning stability. The smaller range of the error bars associated with the proposed method suggests a more consistent learning outcome, emphasizing its effectiveness and reliability in these experimental scenarios.

VI. Conclusion

The contributions of this study are summarized as follows:

- We have developed a variant of the DDQN that integrates prioritized experience replay alongside a heuristic policy, specifically the TWAP, for addressing the OPE—a significant challenge within the financial domain.
- Through numerical experiments, it was demonstrated that our proposed method not only enhances the stability of the learning process but also reduces the volume of data required for effective training.
- Furthermore, our findings indicate that the proposed method is capable of identifying superior strategies, even in scenarios where the heuristic policy (TWAP) may not be inherently optimal.

For future research, we plan to use real-world data to estimate environments and apply the proposed method. Additionally, we will compare our method with similar approaches, such as Deep Q-learning from Demonstrations (DQfD [16]),



(a) Setting 3

(b) Setting 4

Fig. 1: Comparison of rewards during learning between the existing and proposed methods in (a) experimental setup 3 and (b) experimental setup 4. Moving averages ($n = 100$) were computed, with the mean of the results for five different seed values shown as a solid line and the standard deviation as an error bar.

and explore new strategies that integrate heuristic policies within policy-based reinforcement learning frameworks.

References

[1] R. C. Grinold and R. N. Kahn, "Active portfolio management," 2000.

[2] B. Braun and B. Christophers, "Asset manager capitalism: An introduction to its political economy and economic geography," *Environment and Planning A: Economy and Space*, vol. 56, no. 2, pp. 546–557, 2024.

[3] D. Bertsimas and A. W. Lo, "Optimal control of execution costs," *Journal of financial markets*, vol. 1, no. 1, pp. 1–50, 1998.

[4] K. Kubo, K. Nakagawa, D. Mizukami, and D. Acharya, "Optimal liquidation strategy for cryptocurrency marketplaces using stochastic control," *Finance Research Letters*, vol. 53, p. 103639, 2023.

[5] J. A. Busse, T. Chordia, L. Jiang, and Y. Tang, "Transaction costs, portfolio characteristics, and mutual fund performance," *Management Science*, vol. 67, no. 2, pp. 1227–1248, 2021.

[6] K. Nakagawa, S. Noma, and M. Abe, "Rm-cvar: regularized multiple β-cvar portfolio," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 4562–4568.

[7] O. Guéant, *The Financial Mathematics of Market Liquidity: From optimal execution to market making*. CRC Press, 2016, vol. 33.

[8] K. Ito, K. Minami, K. Imajo, and K. Nakagawa, "Trader-company method: A metaheuristics for interpretable stock price prediction," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 2021, pp. 656–664.

[9] Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement learning for optimized trade execution," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 673–680.

[10] D. Hendricks and D. Wilcox, "A reinforcement learning extension to the almgren-chriss framework for optimal trade execution," in *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*. IEEE, 2014, pp. 457–464.

[11] R. S. Sutton and A. G. Barto, "The reinforcement learning problem," *Reinforcement learning: An introduction*, pp. 51–85, 1998.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[14] B. Ning, F. H. T. Lin, and S. Jaimungal, "Double deep q-learning for optimal execution," *Applied Mathematical Finance*, vol. 28, no. 4, pp. 361–380, 2021.

[15] S. Lin and P. A. Beling, "An end-to-end optimal trade execution framework based on proximal policy optimization," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 4548–4554.

[16] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[17] G. S. Atsalakis and K. P. Valavanis, "Surveying stock market forecasting techniques–part ii: Soft computing methods," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5932–5941, 2009.

[18] R. C. Cavalcante, R. C. Brasileiro, V. L. Souza, J. P. Nobrega, and A. L. Oliveira, "Computational intelligence and financial markets: A survey and future directions," *Expert Systems with Applications*, vol. 55, pp. 194–211, 2016.

[19] M. M. L. de Prado, *Machine learning for asset managers*. Cambridge University Press, 2020.

[20] Y. Fujimoto, K. Nakagawa, K. Imajo, and K. Minami, "Uncertainty aware trader-company method: Interpretable stock price prediction capturing uncertainty," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 1238–1245.

[21] K. Nakagawa, M. Abe, and J. Komiyama, "Ric-nn: A robust transferable deep learning framework for cross-sectional investment strategy," in *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2020, pp. 370–379.

[22] K. Imajo, K. Minami, K. Ito, and K. Nakagawa, "Deep portfolio optimization via distributional prediction of residual factors," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 213–222.

[23] E. E. Peters, *Fractal market analysis: applying chaos theory to investment and economics*. John Wiley & Sons, 1994, vol. 24.

[24] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[25] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[26] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.