Dynamic Action Space Reinforcement Learning for Optimal Trading Execution

Pangjing Wu*
College of Computer Science and Software Engineering,
Hohai University
Nanjing, China
Department of Computing,
The Hong Kong Polytechnic University
Hong Kong, China
pang-jing.wu@connect.polyu.hk

Xiaodong Li†
College of Computer Science and Software Engineering,
Hohai University
Nanjing, China
xiaodong.li@hhu.edu.cn

ABSTRACT

Optimal trading execution (OTE) involves dividing large parent orders into smaller child orders to minimize transaction costs and market impact. While reinforcement learning (RL) has demonstrated its effectiveness in this domain, traditional RL methods for OTE are often constrained by a small-scale action space, limiting their performance. Simply increasing the action space results in exponential growth in trial-and-error. Dynamically pruning action space based on context offers a promising balance between performance and efficiency. However, existing RL models are designed for a fixed-action space, making them impractical for dynamic action spaces. Balancing exploration and exploitation in the dynamic action space also remains an unsolved challenge. To address these issues, we propose Dynamic Action Space Reinforcement Learning (DASRL). It iteratively leverages adaptive exploration and dynamic action space search to improve learning efficiency in trading tasks with large action spaces. Theoretical analysis shows that the DASRL reduces cumulative regret while ensuring the introduced optimality bias decreases over time. Experimental results on eight stocks from different sectors demonstrate that the DASRL significantly outperforms traditional strategies and RL baselines, reducing transaction costs by up to 16.2% and improving learning efficiency by up to 14.0%. It highlights the effectiveness of the DASRL in the OTE task with large-scale action spaces.

KEYWORDS

Reinforcement Learning; Algorithmic Trading; Deep Learning; Quantitative Finance

ACM Reference Format:
Pangjing Wu and Xiaodong Li. 2026. Dynamic Action Space Reinforcement Learning for Optimal Trading Execution. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/GFLB4392

*This work was done when the author studied at Hohai University.
†Corresponding Author.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/GFLB4392

1 INTRODUCTION

Optimal trading execution (OTE) is a critical challenge in algorithmic trading, attracting increasing attention in modern finance and artificial intelligence research. The primary objective of OTE is to minimize market impact and transaction costs by strategically splitting large parent orders into smaller child orders for execution, since directly placing large orders into the market often leads to significant price swings and unfavorable costs, *a.k.a.* transaction costs. Traditional approaches, such as Time-Weighted Average Price (TWAP) and Volume-Weighted Average Price (VWAP) strategies, have been widely adopted to address this challenge [13]. However, these essential strategies are based on predetermined rules, thus limiting their performance in dynamic financial markets.

Reinforcement learning (RL) has emerged as a promising alternative, offering the flexibility to adapt dynamically to complex, non-stationary environments. Its sequential decision-making nature consistently aligns with algorithmic trading, enabling the development of strategies that outperform static rule-based methods. Early applications have shown the effectiveness of RL on small-scale OTE tasks [10, 23, 28]. Recent advancements in deep RL (DRL) have extended the applicability of RL to large-scale trading environments. DRL-based approaches have demonstrated their effectiveness in tackling challenges such as extended trading durations and larger order sizes [18], showcasing significant potential in addressing the demands of modern financial markets [21, 24, 35]. For instance, hybrid discrete–continuous action RL has been explored for execution tasks [25], multi-agent RL has been tested in realistic limit order book simulations [15], and deep RL has been applied to optimal limit order placement in financial markets [27].

Despite these advancements, RL-based methods remain constrained by a small-scale action space. It is inadequate to handle the real-world OTE tasks with large-scale action spaces, as illustrated in Table 1. Expanding the action space to include additional prices and sizes is a straightforward approach. However, it is computationally prohibitive due to the exponential growth in the number of trials required for trial-and-error. Dynamically adding new high-value actions during training introduces substantial challenges, as current RL methods are designed for a fixed action space. It requires retraining RL agents from scratch, further increasing computational costs. Moreover, recent model-based approaches to a parameterized action space [36] and domain-specific applications [8, 33] illustrate that

Table 1: Existing Studies of RL in Algorithmic Trading

Method	Price Action	Size Action	Time Length	Total Order Size
Nevmyvaka *et al.* [23]	20	1	2∼8 min	5K; 10K
Hendricks and Wilcox [10]	1	8	20 ∼ 60 min	10K; 1M
Shen *et al.* [28]	20	1	10 min	20K
Ning *et al.* [24]	1	1	60 min	2K
Ye *et al.* [35]	10	1	2 min	5K; 10K
Lin *et al.* [21]	1	[0,1]	2 min	300∼7K
Fang *et al.* [9]	1	[0,1]	30 min	<1K
Wang *et al.* [32]	1	1	20 min	10K∼40K
Li *et al.* [18]	20	1	240 min	10K∼100K
Ours	**20**	**50**	**30 min**	**1M∼5M**

action space complexity is increasingly recognized as a bottleneck across different RL domains.

A promising approach is to **start with a large-scale action space and dynamically prune confident low-value actions during training**. It allows the agent to focus on high-value actions, enhancing learning efficiency. While existing studies on a pruning *invalid* action space [4, 31] and dynamic action adaptation [17] have demonstrated effectiveness, they primarily target masking invalid or repeated actions. Other recent studies propose dynamic reduction strategies in autonomous driving [8] and multi-agent control systems [33], further demonstrating the cross-domain potential of dynamic action adaptation. However, existing approaches have not been systematically adapted to address OTE tasks, where high-frequency decisions and market feedback require efficient updates to RL policies within a dynamic action space and accurate evaluation of action values.

To address these challenges, we propose a Dynamic Action Space Reinforcement Learning (DASRL) framework, specifically designed to optimize RL policies for the OTE tasks with a large-scale action space. It iteratively prunes the action space to focus on high-value actions, efficiently optimizing policies over the dynamic action space. Our framework enhances RL policy optimization through three modules: 1) *Action Value Evaluation Module*, which integrates environmental feedback and domain-specific indicators to quantify each action's value; 2) *Action Subspace Search Module*, which identifies high-value action subspaces by balancing exploration and exploitation, allowing the agent to focus on the most promising actions; and 3) *Action Subspace-Based Policy Optimization Module*, which efficiently optimizes the agent's policy within the identified subspace to maximize performance. These modules operate synergistically to accelerate learning and enhance performance for real-world OTE tasks.

We provide theoretical and empirical validations to evaluate the effectiveness of the DASRL in the OTE. The theoretical analysis demonstrates that the DASRL reduces cumulative regret by scaling the whole action space to the identified subspace. Furthermore, the introduced optimality bias diminishes over time, ensuring convergence to the optimal policy. Empirical experiments on eight representative stocks from the Shanghai Stock Exchange (SSE) further validate DASRL's efficacy. The results show that the DASRL reduces transaction costs by up to 16.2% and improves learning efficiency by 14.0% compared to baseline methods. These improvements highlight the effectiveness of the DASRL in accelerating convergence and enhancing performance in complex trading scenarios.

Our contributions are three-fold:

- We propose **Dynamic Action Space Reinforcement Learning (DASRL)**, a framework that dynamically prunes action space to improve learning efficiency in large-scale OTE tasks.
- We provide theoretical guarantees showing that DASRL reduces cumulative regret by pruning the action subspace, while the induced optimality bias diminishes over time.
- Extensive experiments on eight SSE stocks demonstrate that DASRL effectively prunes low-value actions, reducing transaction costs by 16.2% and accelerating learning efficiency by 14.0% compared to strong baselines.

2 RELATED WORK

2.1 Optimal Trading Execution

Early OTE strategies were mainly based on ideal assumptions, focusing on modeling factors that could impact transaction costs, such as market information [5], price impact [1], and liquidity risk preferences [12]. However, these strategies relied on strict market pattern assumptions, which limited their adaptability and effectiveness in real-world markets. To overcome these limitations, Berkowitz *et al.* [3] proposed the VWAP strategy, which reduced transaction costs by splitting orders based on market liquidity. While promising, this strategy relied on fixed execution rules and lacked flexibility, resulting in poor performance in dynamic markets. Researchers later worked to improve the VWAP strategy by refining volume predictions [6, 16] and incorporating additional market data [2, 14]. Despite these efforts, its performance was constrained by the models' limited predictive power, which struggled to adapt to rapidly changing market conditions.

Since the sequential decision mechanism of RL aligns naturally with order execution, researchers have investigated its effectiveness in small-scale trading tasks [10, 23, 28]. The advent of deep Q-learning networks (DQN) [22] empowered RL agents to learn trading strategies directly from raw market data [21, 24, 35]. Subsequent works extended DRL to more realistic settings, such as hybrid discrete–continuous control [25], multi-agent limit order book environments [15], and practical order placement strategies [27]. Recent studies have further advanced structural efficiency in trading RL: hierarchical decision frameworks [26], maskable action representations [37], and logic-guided Q-learning [19] enhance adaptability and policy stability. Alongside parameterized action models [36], these works collectively highlight the growing importance of managing action space complexity in OTE.

2.2 Dynamic Action Space

Although researchers have investigated RL for over half a century, most studies have modeled the environment with a fixed action space comprising all available actions across states. However, as the action space grows, not all actions are valid, significantly reducing learning efficiency by spending time exploring invalid actions. An intuitive approach to avoid pointless sampling was to mask invalid actions, commonly used in optimizing game environments with millions of actions [4, 31]. Boutilier *et al.* [7] explored the effective implementation of Q-learning with the action mask, while Huang *et al.* [11] provided theoretical proof of effectiveness in policy gradient methods. Nevertheless, many actions remain valid in

open-world tasks but are inefficient, leading RL to fail to achieve optimal efficiency solely by using the action mask.

Recent advances extend beyond invalid-action masking to broader forms of action adaptation. Dynamic action repetition has been proposed to compress action usage in temporally extended tasks [17]. Domain-specific studies have introduced structured action reduction, such as in autonomous driving [8] and multi-agent control systems [33]. Model-based approaches for a parameterized action space [36] also advance this direction by jointly learning the environment dynamics and action representations. These methods collectively illustrate the growing recognition of dynamic action adaptation across domains. However, they do not explicitly address pruning low-value actions in trading environments, where actions are tightly integrated with financial feedback. Our work fills this gap by introducing a systematic framework for value-based pruning in OTE tasks.

3 PRELIMINARY

3.1 Reinforcement Learning

RL provides a framework for solving sequential decision-making problems where an agent learns to interact with an environment to maximize cumulative rewards. This learning process is typically modeled as a Markov Decision Process (MDP) [29], defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{F}, R, \gamma)$, where $\mathcal{S}$ represents a finite set of states describing the environment; $\mathcal{A}$ denotes a finite set of actions available to the agent; $\mathcal{F} : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1] = \Pr(s' \mid s, a)$ defines the state transition probability of transitioning from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$; $R(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function specifies the immediate reward for executing action a in state s; and $\gamma \in [0,1)$ is the discount factor balancing the importance of immediate versus future rewards.

RL aims to derive an optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative reward, also known as the return. The return from time step t is defined as,

$$G_t = \sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}), \tag{1}$$

where γ determines the extent to which future rewards are considered. The policy $\pi(a \mid s)$ specifies the probability of conducting action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. Action-value functions quantify the expected return of a policy π,

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[G_t \mid s_t = s, a_t = a\right]. \tag{2}$$

The optimal policy π^* maximizes the value functions, satisfying the Bellman equations,

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \Pr(s' \mid s, a) \max_{a' \in \mathcal{A}} Q^*(s', a'). \tag{3}$$

The optimal policy is derived as,

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a). \tag{4}$$

3.2 Optimal Trading Execution

OTE is a key challenge in algorithmic trading, aiming to minimize transaction costs and market impact when executing large orders. Let the total order size be Q, and the execution period is divided into T intervals. The objective is to execute the order while minimizing the total cost,

$$\text{Minimize: } C = \text{side} \cdot \sum_{t=1}^{T} q_t p_t, \tag{5}$$

where *side* represents the trading direction, with 1 for buy orders and -1 for sell orders; q_t is the volume executed by a given trading strategy within interval t, satisfying $\sum_t q_t = Q$; and p_t is the execution price in the same interval. The problem is to find the optimal allocation of $\{q_t\}_{t=1}^{T}$ that minimizes market impact and transaction costs.

Traditional strategies, such as the VWAP strategy, address this problem using predefined allocation rules. The VWAP strategy allocates the order size proportionally to the expected market volume $\hat{v}_t$ in each interval, aligning execution with market liquidity under the assumption of accurate volume predictions,

$$q_t = Q \cdot \frac{\hat{v}_t}{\sum_{t=1}^{T} \hat{v}_t}, \quad \forall t \in \{1, \ldots, T\}. \tag{6}$$

The VWAP of the strategy is calculated by,

$$P_{\text{VWAP}} = \frac{\sum_{t=1}^{T} q_t p_t}{\sum_{t=1}^{T} q_t}. \tag{7}$$

The performance of algorithmic trading strategies is generally benchmarked against the *market VWAP*, defined as,

$$\bar{P}_{\text{VWAP}} = \frac{\sum_{t=1}^{T} v_t \bar{p}_t}{\sum_{t=1}^{T} v_t}, \tag{8}$$

where v_t is the market volume at interval t, $\bar{p}_t$ is the market average trading price in the same interval. A key metric derived from this is *VWAP slippage*, which quantifies the execution performance as,

$$\text{VWAP Slippage} = \text{side} \cdot \frac{P_{\text{VWAP}} - \bar{P}_{\text{VWAP}}}{\bar{P}_{\text{VWAP}}}. \tag{9}$$

It indicates how closely the strategy aligns with the VWAP of the ideal market. Larger slippage indicates inefficiencies in execution, often caused by poor order placement or suboptimal strategy performance. For consistency in RL optimization and result presentation, we *reverse* the polarity of VWAP slippage in the following parts, where a positive value indicates optimal transaction costs.

4 ENVIRONMENT MODELING

To optimize algorithmic trading strategies using reinforcement learning (RL), we first establish simulation rules for order execution. To capture the characteristics of the OTE task and enable effective RL policy optimization under a dynamic action subspace, we generalize the standard Markov Decision Process (MDP) into an action-subspace Markov Decision Process (ASMDP), denoted as $(\mathcal{S}, \mathcal{A}, \Omega, R, \mu, \mathcal{F}, \gamma)$. The formal definition of each component is given below.

State. $\mathcal{S}$ consists of the LOB sequences and the order execution progress. The LOB sequence contains the latest 20 snapshots. Each snapshot is taken every 3 seconds and includes 10 quote price levels and corresponding buy and sell order sizes. The order execution progress consists of a *remaining interval ratio* and an *executed order ratio*, providing agents with information to optimize the execution of remaining orders.

Action. $\mathcal{A}$ defines the agent's trading actions, including buying (or selling) a specified stock volume at a given price level. Each action is defined by an order price a_p and size a_v, *i.e.*, $a = (a_p, a_v)$. The order price spans 20 levels in the level-2 LOB, from bid_{10} to ask_{10}, while the order size ranges from 1 to L lots. It results in $20 \times L$ feasible actions and a default *do nothing* action, $a_\emptyset$. Our action space is significantly more complex than those considered in previous studies, as summarized in Table 1.

Action Subspace. Ω is a subset of $\mathcal{A}$, defined as $\Omega \subseteq \mathcal{A}$, and includes only a selected portion of order prices and sizes. To maintain the completeness of order execution, all action subspaces include the default $a_\emptyset$, allowing the agent to wait for favorable order execution conditions.

Action Values. μ is a vector of domain-specific indicators used to evaluate the value of actions for action subspace pruning. The action values consist of instant trading costs $\mu^{(r)}$ and execution intervals $\mu^{(\tau)}$, expressed as $\mu(a) = [\mu^{(r)}(a); \mu^{(\tau)}(a)]$. The $\mu^{(r)}$ represents the current VWAP slippage, set to 0 if no orders are executed. The $\mu^{(\tau)}$ measures the intervals taken to complete the action. It reflects time risk, as delayed execution exposes the order to adverse market movements.

Reward. R guides the agent in learning the best trading policy. A sparse reward is used in this task. It feeds back "0" at each trading interval and *VWAP slippage* at the end of trading. To reflect the practical requirement that brokers must complete orders within a specified time, the agent receives a penalty if the order remains unexecuted. Additionally, the commission fee of the SSE is ignored as it remains relatively fixed.

Transition Probability. Since the environment is simulated using historical data that cannot reflect the agent's current actions, we follow related studies of [18] and [20], which assume that the agent's behavior does not influence other market participants. Consequently, the market state transition $\mathcal{F}$ remains consistent with historical records.

5 DYNAMIC ACTION SPACE REINFORCEMENT LEARNING

Given the large action space of the OTE task, existing RL approaches face computational challenges due to the exponential growth of trial-and-error. To address this challenge, we introduce the DASRL, which starts with a large-scale action space and dynamically prunes confident low-value actions during training to enhance the learning efficiency, as depicted in Figure 1. The subsequent sections provide a detailed exposition of each module within the DASRL framework.

5.1 Action Value Evaluation

The action value evaluation module evaluates the domain-specific indicators of each action, serving as a reference for action subspace pruning. We replace the original indicator values with their ranks before normalization to account for scale differences and reduce the impact of outliers. Since the indicators have different polarities, we apply appropriate sorting methods to ensure that higher ranks consistently reflect better values. For example, we sort $\mu^{(r)}$

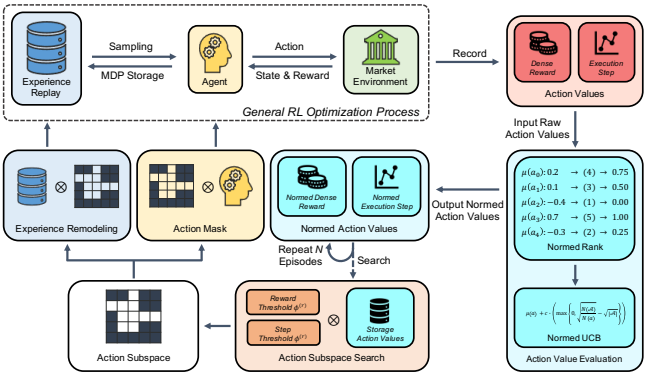


Figure 1: The scheme of general RL and the proposed DASRL. Besides the general RL process, it consists of three modules: 1) action value evaluation; 2) action subspace search; 3) action subspace-based policy optimization.

(instant income) in ascending order so that a larger value represents lower transaction costs. In contrast, $\mu^{(\tau)}$ (execution interval) in descending order, as larger values indicate larger liquidity risk. Finally, we normalize the ranks using min-max scaling to scale the action values to $[0, 1]$, ensuring consistency and comparability across indicators.

Due to the non-stationary nature of the environment, actions should be explored multiple times to calculate their values, especially in the early stages of training. Since our action subspace search mechanism primarily accelerates the initial training phase, balancing **exploration** and **exploitation** becomes critical. To address this issue, we adopt the Upper Confidence Bound (UCB) score [29] to evaluate action values and balance exploration and exploitation. Specifically, the UCB score is defined as,

$$\text{UCB}(a) = \mu(a) + c \cdot \sqrt{\frac{\ln N(\mathcal{A})}{N(a)}}, \tag{10}$$

where the first term is the exploitation term, denoting the action values; the second term is the exploration term, representing the uncertainty or variance of action value estimation; $\mu(a)$ is the normalized action value; $N(a)$ is the visit count of a; $N(\mathcal{A})$ is the visit count of all actions, and c is a temperature coefficient. It ensures that actions with high estimated rewards are prioritized while encouraging the exploration of less frequently selected actions to improve policy learning efficiency.

However, the original UCB score is designed to balance exploration and exploitation during agent-environment interactions, not for pruning the optimal action subspace. In particular, under an initial ϵ-greedy strategy where $\epsilon = 1.0$ and all actions are chosen uniformly, the expectation of the exploration term in equation (10) is not 0. As a result, evaluating the sufficiency of exploration is challenging because the term is biased and lacks a known value.

To address this issue and adapt the UCB score for action value evaluation and further action subspace search, we modify its exploration term by subtracting a baseline equal to its expectation under uniform action pruning. This adjustment ensures that the expectation of the exploration term becomes 0, providing a more balanced

measure of exploration sufficiency. However, the logarithm in the original UCB exploration term complicates the calculation, as it is hard to express in terms of known constants. Since the logarithm primarily scales the exploration term and preserves its monotonicity, we can remove it without changing the relative ranking of actions. After removing the logarithm, the expectation of the exploration term under uniform action pruning becomes,

$$\mathbb{E}_{a\sim\pi^{\epsilon=1.0}}\left[c\cdot\sqrt{\frac{N(\mathcal{A})}{N(a)}}\right]=c\cdot\sqrt{|\mathcal{A}|}, \tag{11}$$

where $|\mathcal{A}|$ is the dimension of action space or action subspace. We subtract the baseline from the exploration term, then the expectation of the exploration term under the uniform distribution is 0, that is,

$$\mathbb{E}_{a\sim\pi^{\epsilon=1.0}}\left[c\cdot\left(\sqrt{\frac{N(\mathcal{A})}{N(a)}}-\sqrt{|\mathcal{A}|}\right)\right]=0. \tag{12}$$

Furthermore, since the variance of $\sqrt{\frac{N(\mathcal{A})}{N(a)}}$ depends on the policy distribution and is difficult to estimate during training, we set the minimum exploration term to 0. Subtracting the baseline prevents negative values, ensuring stability during exploration. Finally, the standardized UCB score for action subspace search is,

$$\mathrm{UCB}_N(a)=\mu(a)+c\cdot\left(\max\left\{0,\sqrt{\frac{N(\mathcal{A})}{N(a)}}-\sqrt{|\mathcal{A}|}\right\}\right). \tag{13}$$

5.2 Action Subspace Search

The action subspace search module periodically identifies a new subspace from the current action space (or action subspace) based on action values and predefined pruning conditions. Specifically, given a preset search period, the module selects an action subspace using thresholds $\Phi=\left\{\left(\phi_l^{(i)},\phi_u^{(i)}\right)|i=1,\ldots,|\mu|\right\}$, where the $\phi_l^{(i)}$ is the lower threshold of $\mu^{(i)}$, and the $\phi_u^{(i)}$ the upper threshold of $\mu^{(i)}$. The search process iterates until the resulting subspace meets the specified condition. At each search step, multiple candidate action subspaces $\left\{\Omega_{\mu_t^{(1)}},\Omega_{\mu_t^{(2)}},\ldots,\Omega_{\mu_t^{(|\mu|)}}\right\}$ may be selected because the action values $\mu=\{\mu^{(1)},\mu^{(2)},\ldots,\mu^{(|\mu|)}\}$ consist of several indicators. We take the intersection of these candidate subspaces to ensure all pruning conditions are satisfied. Therefore, the action subspace at learning step t is represented as,

$$\Omega_{\mu_t}=\bigcap_{i=1}^{|\mu|}\Omega_{\mu_t^{(i)}}. \tag{14}$$

5.3 Action Subspace-Based Policy Optimization

The dynamic action space poses challenges for typical RL training schemes, as RL agents require retraining whenever the action space changes, leading to a loss of previously learned knowledge. DASRL addresses this issue through an action subspace strategy optimization module that combines action auto-masking and experience remodeling to enable continuous policy optimization over a changing action space while retaining learned experience.

Given an action subspace Ω_e obtained from the action subspace search module at the e-th episode, the agent selects the action with the highest expected reward within Ω_e, based on the policy π_{Ω_e} corresponding to Ω_e, that is,

$$a^*=\arg\max_{a\in\Omega_e}Q(s,a;\theta), \tag{15}$$

where θ is the parameter of the policy.

To ensure the RL agent focuses on the current action subspace, an automated action masking approach is applied before policy optimization. It prevents the inclusion of previously learned values for actions outside the current action subspace Ω_e. Specifically, we introduce the action subspace as a constraint in $Q(s,a;\theta)$, eliminating actions belonging to the complement of Ω_e, denoted as Ω_e^c,

$$Q(s,a;\theta,\Omega_e)=\begin{cases}Q(s,a;\theta), & a\in\Omega_e;\\ -\infty, & a\in\Omega_e^c.\end{cases} \tag{16}$$

After that, the a^* selected by equation (15) is in the Ω_e.

In off-policy RL models, experience replay is a widely used technique to improve sampling efficiency during policy optimization. It involves maintaining a replay buffer of MDP (or its extended) tuples [22], from which random samples are drawn to optimize the policy. However, because the replay buffer is circularly updated, it may contain samples with actions outside the current action subspace Ω_e. These irrelevant samples can interfere with parameter optimization and negatively impact policy learning.

To address this issue, we introduce experience remodeling, which filters out samples associated with actions outside the current action subspace Ω_e. Specifically, after obtaining Ω_e, the dataset $\mathcal{D}$ is cleaned by removing tuples where the action a belongs to the complement of Ω_e, denoted as,

$$\mathcal{D}\leftarrow\mathcal{D}\setminus\left\{(s,a,r,s')\mid(s,a,r,s')\in\mathcal{D},a\in\Omega_e^c\right\}. \tag{17}$$

By removing these irrelevant samples, experience remodeling ensures that the replay buffer aligns with the current action subspace, thereby improving the stability and efficiency of policy optimization in dynamic action space environments.

6 THEORETICAL ANALYSIS

In this section, we conduct a theoretical analysis of the DASRL's convergence rate and optimality bias compared to the general Q-learning method.

6.1 Convergence Rate Analysis

In the DASRL, action subspaces are periodically selected to reduce the exploration range, thereby reducing the number of suboptimal actions and accelerating convergence. We first provide the regret bound after the DASRL searches the action subspace.

Property 1 (Cumulative Regret Bound). *Let $|\mathcal{A}|$ denote the size of the action space and $|\Omega_e|$ the size of the action subspace after the e-th search. The cumulative regret $R(T)$ of the DASRL satisfies the following bound,*

$$R(T)\le O(|\mathcal{S}|\cdot|\Omega_e|\cdot\ln T), \tag{18}$$

where $|\mathcal{S}|$ is the state space size, and T is the number of time steps.

Corollary 1 (Impact of Action Space Size). *The effect on convergence rate is limited when the action space is small ($|\mathcal{A}|\le 10$). When the action space is large ($|\mathcal{A}|\gg 10$), the DASRL significantly*

accelerates convergence, reducing the cumulative regret growth rate by a factor of $O(\sqrt{|\mathcal{A}|})$.

6.2 Optimality Bias Analysis

The action subspace search in the DASRL may introduce optimality bias, *i.e.*, losing the optimal action a^* during subspace pruning could degrade policy performance.

Definition 1 (Optimality Bias). *Let $\Omega_e \subseteq \mathcal{A}$ denote the action subspace selected at the e-th search. If the optimal action $a^* \notin \Omega_e$, the optimality bias is defined as,*

$$\delta_{opt} = \gamma \cdot \left| \max_{a \in \mathcal{A}} Q^*(s,a) - \max_{a \in \Omega_e} Q^*_{\Omega_e}(s,a) \right|, \tag{19}$$

where γ is the discount factor.

Property 2 (Bound on Optimality Bias). *Let p_e denote the probability that the optimal action a^* is not included in the subspace Ω_e. The optimality bias of the DASRL satisfies,*

$$\delta_{opt} \leq \gamma \cdot C_Q \cdot p_e, \tag{20}$$

where $C_Q = \max_{s,a} |Q^*(s,a)|$.

Corollary 2 (Bias Convergence Condition). *If the DASRL ensures $p_e \to 0$, the final policy bias satisfies:*

$$\lim_{e \to \infty} \delta_{opt} = 0.$$

Our theoretical analysis demonstrates that DASRL reduces cumulative regret from $O(|\mathcal{A}| \cdot \ln T)$ in the whole action space to $O(|\Omega_e| \cdot \ln T)$ in the optimal subspace. Moreover, the subspace pruning ensures that the introduced bias decreases over time, guaranteeing convergence to the optimal policy.

7 EXPERIMENTS AND DISCUSSION

To verify the efficiency of the DASRL, we conduct extensive OTE experiments and compare it with baselines on transaction costs and training efficiency. In addition, we empirically analyze the impact of different modules on the DASRL's performance to verify each module's validity.

7.1 Datasets

From the SSE 50 Index's constituent stocks in May 2014, we select eight stocks from different sectors. We perform experiments on their level-2 data from June 2014 to March 2015. The level-2 data includes second-by-second LOBs and trade data, with LOBs quoted at 10 price levels on both sides, and trade data containing each trading record. The **LOBs contain 1M snapshots of quotes**, while the **trade data contains about 15M trading records**. Due to insufficient liquidity, data from days when the market price reaches the price limit are excluded. The first 80% of the data is used as the training set, while the last 20% is the test set.

7.2 Experimental Setup

Baselines. We compare the DASRL model with two classical algorithmic trading strategies and three widely used DRL methods: **TWAP** [13]: Orders are evenly executed within a given time slice and unaffected by other factors. [21] used it as a baseline. **Arrival Price (AP)**: Order's price is determined by the filled ratio, *i.e.,* if the filled ratio is lower than the time ratio, execute as a market order; otherwise, execute as a limit order. When the filled ratio exceeds 10% of the time ratio, it stops issuing new orders. **DQN** [22]: A TD-based DRL model. [23], [1], and [28] used it to optimize algorithmic trading strategies. **DDQN** [30]: An improved DRL model based on DQN. [24] used it to optimize algorithmic trading strategies. **Advantage Actor-Critic (A2C)** [34]: An actor-critic-based DRL model, which consists of an actor-network and a critic-network.

Metrics. We use the base point (bp) of VWAP slippage (Equation 9) as the metric of transaction costs and propose an area under test curve (AUTC) to measure the efficiency of RL policy learning. The AUTC is the area enclosed by the coordinate axis and the test curve, as the model with higher learning efficiency has a faster-growing curve and a larger area under the curve. In practice, the AUTC is calculated using the trapezoidal method. We select the maximum m^+ and minimum m^- from the test curves and use the rectangular space, Z, enclosed by m^+, m^-, and x axis as the baseline to normalize the AUTC. The normalized AUTC is represented as,

$$\text{AUTC}\,(\boldsymbol{m}) = \frac{1}{Z} \cdot \sum_{m_i \in \boldsymbol{m}} \frac{\Delta m}{2} \cdot [m_i + m_{i+1}], \tag{21}$$

where $\boldsymbol{m}$ is the checkpoint on the test curve, and Δm is the interval.

Hyperparameters. In the experiment, RL agents are required to execute a parent sell order ranging from 100,000 to 500,000 lots, starting with an action space of 1,000 initial actions. The action subspace is updated every $\phi = 50$ episodes, targeting a reduced scale of $|\tilde{\Omega}| = 0.2|\mathcal{A}|$. Reward thresholds are set to $(\phi_l^{(r)}, \phi_u^{(r)}) = (0.15, 1.0)$, while step thresholds are $(\phi_l^{(\tau)}, \phi_u^{(\tau)}) = (0.1, 1.0)$.

7.3 Results

Table 2 shows the transaction costs of the DASRL and the baselines over eight stocks, where the best results are highlighted in bold. It demonstrates that the DASRL achieves the lowest transaction costs across all eight stocks, consistently outperforming classical algorithmic strategies (*i.e.*, TWAP and AP) and DRL-based baselines (*i.e.*, DQN, DDQN, and A2C). These results indicate that the DASRL can effectively identify high-value actions, enabling optimal cost minimization across diverse stocks.

Figure 2 shows the learning curves of the DASRL and the baselines over eight stocks. The solid lines in the figure represent the mean transaction costs, while the shadows denote the respective standard deviations. It shows that the DASRL curves grow considerably faster than the baselines during the early training stage. The upward jumps in specific episodes are due to the action subspace search in the DASRL. Furthermore, Table 3 demonstrates the efficiency of the DASRL via the AUTC, where the best and second-best results are highlighted by bold and underlined. It shows that the DASRL has the highest learning efficiency among all stocks. DDQN and DQN closely follow, while A2C performs the worst. The results demonstrate that the DASRL enables efficient policy optimization for OTE tasks with a large-scale action space.

Table 2: VWAP Slippage of the DASRL and Baselines (bp)

	600000	600010	600018	600028	600030	600048	600050	600104
TWAP	-3.05	-13.90	-9.02	-8.11	-2.48	-5.08	-13.36	-4.56
AP	-2.39	-11.03	-8.92	-4.85	-2.49	-4.78	-10.29	-4.68
DQN	8.05	11.65	7.56	5.62	6.84	26.15	27.22	9.72
w/ DASRL	8.47	13.28	10.40	6.66	7.05	29.88	31.20	13.17
Improvement (%)	+5.22%	+13.99%	+37.57%	+18.50%	+3.07%	+14.27%	+14.64%	+35.50%
DDQN	8.36	12.44	9.73	6.28	6.88	31.47	29.26	14.39
w/ DASRL	8.75	14.84	11.48	7.02	7.11	36.32	34.16	16.11
Improvement (%)	+4.67%	+19.30%	+17.98%	+11.78%	+3.34%	+15.41%	+16.74%	+11.95%
A2C	6.12	9.85	9.57	6.34	6.25	27.46	23.67	11.97
w/ DASRL	7.34	10.27	10.44	7.02	6.29	31.55	28.83	12.88
Improvement (%)	+19.93%	+4.27%	+9.09%	+10.73%	+0.64%	+14.90%	+21.80%	+7.60%
DASRL (Best)	**8.75**	**14.84**	**11.48**	**7.02**	**7.11**	**36.32**	**34.16**	**16.11**

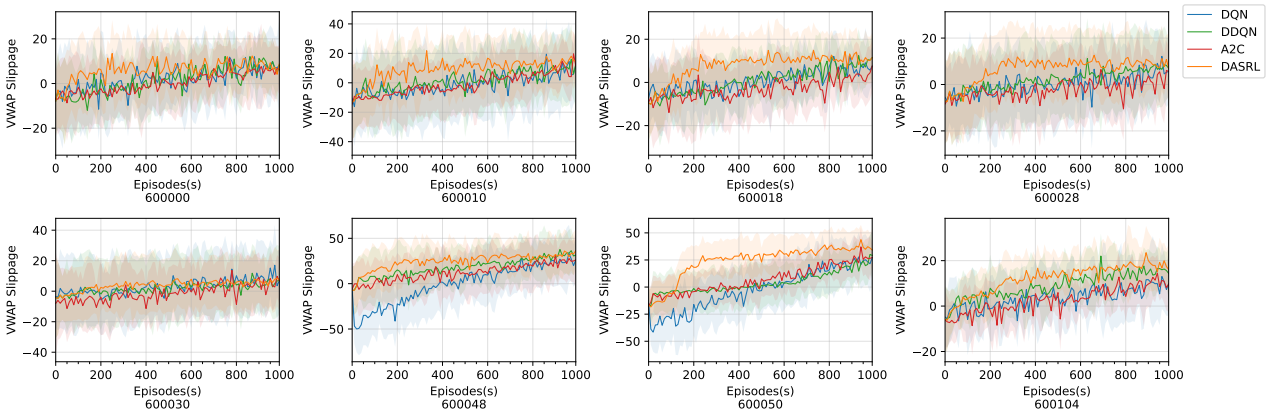


Figure 2: Transaction costs of the DASRL and baselines on eight stocks.

Table 3: AUTC of the DASRL and Baselines

	600000	600010	600018	600028	600030	600048	600050	600104
DQN	0.674	0.654	0.666	0.661	0.679	0.656	0.645	0.684
DDQN	0.669	0.671	0.665	0.672	0.672	0.779	0.685	0.719
A2C	0.662	0.653	0.640	0.646	0.651	0.739	0.705	0.678
DASRL	**0.701**	**0.722**	**0.713**	**0.705**	**0.686**	**0.822**	**0.820**	**0.740**

7.4 Discussions

To investigate how different thresholds of action subspace search affect algorithmic trading strategy optimization, we varied the thresholds for reward ($\phi_l^{(r)}, \phi_u^{(r)}$) and step ($\phi_l^{(\tau)}, \phi_u^{(\tau)}$) in the DASRL. Table 4 depicts the VWAP slippages corresponding to four threshold combinations. The average decline of transaction costs from #1 to #2 is 2.01, significantly higher than that from #3 to #4, which is −0.73. It shows that the transaction costs of each stock decrease gradually as the $\phi^{(r)}$ and $\phi^{(\tau)}$ increase. The benefits of increasing the thresholds are limited or even opposite (*e.g.*, #3 vs. #4) after the thresholds reach a certain level (*i.e.*, #3). Figure 3 shows the test curves of the DASRL under different threshold combinations. As the $\phi^{(r)}$ and $\phi^{(\tau)}$ increase, the learning speed of the DASRL also increases, while the overall trend remains consistent. Additionally, the test curve standard deviation decreases significantly as thresholds increase, indicating that higher thresholds can reduce standard deviation and enhance the robustness of algorithmic trading strategies.

To reveal the mechanism of the action subspace search module in DASRL, we visualize the selected action subspace at each search period. Figure 4 shows the action subspace of each search period,

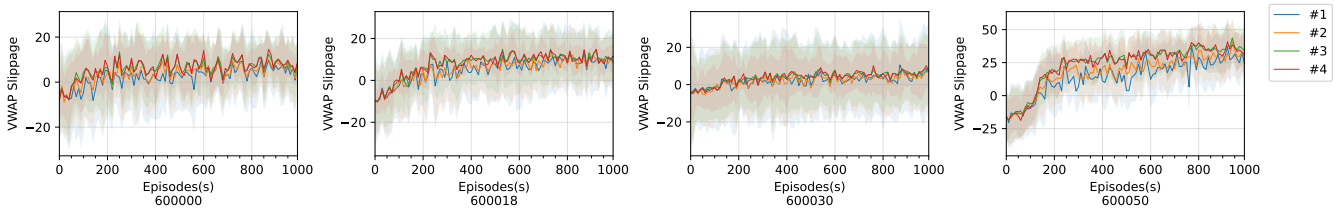


Figure 3: Transaction costs of the DASRL based on different search thresholds.

Table 4: VWAP Slippage of the DASRL on Different Search Thresholds (bp)

	$\phi^{(r)}(\geq)$	$\phi^{(\tau)}(\geq)$	600000	600018	600030	600050
#1	0.100	0.005	7.35	8.55	6.42	24.25
#2	0.125	0.075	8.33	10.02	6.76	29.48
#3	0.150	0.100	**8.75**	**11.48**	7.11	**34.16**
#4	0.200	0.150	8.64	11.62	**7.14**	31.19

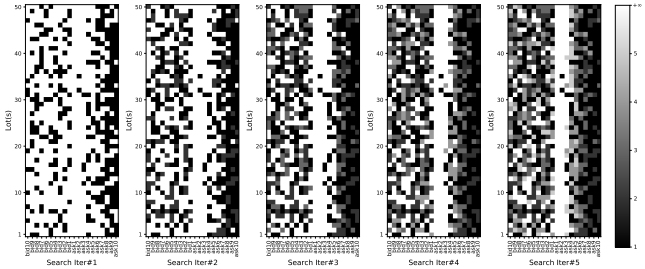
Table 5: Statistics of Stock Price Volatility

	600000	600018	600030
Mean	1.41	1.93	2.13
Variance	1.18	2.53	3.64

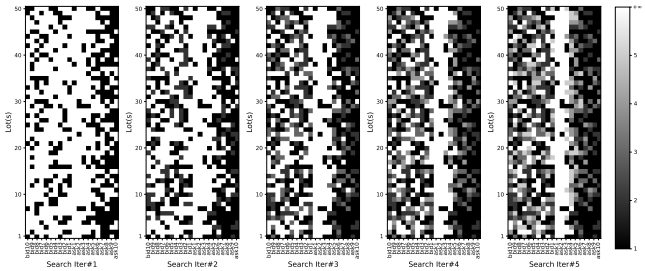
where white blocks represent the action subspace of the current period, and gray blocks indicate the actions discarded in the corresponding period. The initially discarded actions are mostly located at both sides of the price axis, while the final action subspace is generally between "bid_1" and "ask_3". The results demonstrate that the price of orders significantly impacts transaction costs. In addition, the action subspace of 600018 has the narrowest distribution along the price axis, and the distribution of 600000 is similar to that of 600018, while the distribution of 600030 is the widest. We also calculate the price volatility of these stocks and present the results in Table 5. It demonstrates that low-priced volatility stocks exhibit a narrow distribution along the price axis, and vise versa.

8 CONCLUSION

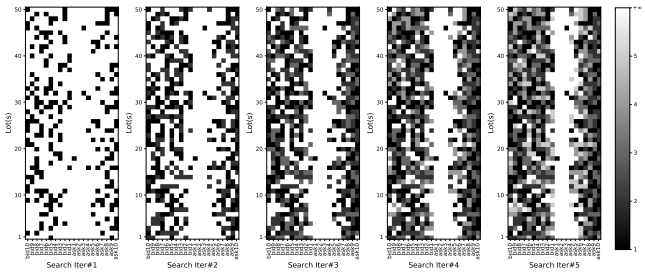
In this paper, we propose DASRL, a dynamic action space-based DRL model designed for OTE tasks with a large-scale action space. The DASRL addresses the challenge of a large action space by dynamically pruning the action subspace and optimizing the policy within it, thereby substantially improving learning efficiency. Theoretical analysis demonstrates that the DASRL reduces cumulative regret while ensuring the introduced optimality bias decreases over time. Comprehensive experiments across eight SSE-listed stocks show that our approach outperforms existing methods in reducing transaction costs and improving learning efficiency. Specifically, the DASRL surpasses the most powerful baselines by 16.2% in reducing VWAP slippage and by 14.0% in improving AUTC.



(a) *Shanghai Pudong Development Bank* (600000)



(b) *Shanghai International Port* (600018)



(c) *Citic Securities Company Limited* (600030)

Figure 4: Search process of action subspace.

The limitation of this work is that the model performs a subspace search on a fixed period. In future research, we aim to introduce a dynamic search period based on action value estimation. The dynamic period can skip converged evaluation processes and avoid conducting subspace searches when non-convergence occurs, thereby further improving efficiency.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China, No. 2024YFC3210800.

REFERENCES

[1] Robert Almgren and Neil Chriss. 2001. Optimal execution of portfolio transactions. *Journal of Risk* 3 (2001), 5–40.

[2] Robert Almgren and Julian Lorenz. 2006. Bayesian adaptive trading with a daily cycle. *The Journal of Trading* 1, 4 (2006), 38–46.

[3] Stephen A Berkowitz, Dennis E Logue, and Eugene A Noser Jr. 1988. The total cost of transactions on the NYSE. *The Journal of Finance* 43, 1 (1988), 97–112.

[4] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680* (2019).

[5] Dimitris Bertsimas and Andrew W Lo. 1998. Optimal control of execution costs. *Journal of Financial Markets* 1, 1 (1998), 1–50.

[6] Jędrzej Białkowski, Serge Darolles, and Gaëlle Le Fol. 2008. Improving VWAP strategies: A dynamic volume approach. *Journal of Banking & Finance* 32, 9 (2008), 1709–1722.

[7] Craig Boutilier, Alon Cohen, Avinatan Hassidim, Yishay Mansour, Ofer Meshi, Martin Mladenov, and Dale Schuurmans. 2018. Planning and learning with stochastic action sets. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 4674–4682.

[8] Elahe Delavari, Feeza Khan Khanzada, and Jaerock Kwon. 2025. Action Space Reduction Strategies for Reinforcement Learning in Autonomous Driving. *arXiv preprint arXiv:2507.05251* (2025).

[9] Yuchen Fang, Kan Ren, Weiqing Liu, Dong Zhou, Weinan Zhang, Jiang Bian, Yong Yu, and Tie-Yan Liu. 2021. Universal Trading for Order Execution with Oracle Policy Distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 107–115.

[10] Dieter Hendricks and Diane Wilcox. 2014. 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, 457–464.

[11] Shengyi Huang and Santiago Ontañón. 2020. A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171* (2020).

[12] Gur Huberman and Werner Stanzl. 2005. Optimal liquidity trading. *Review of finance* 9, 2 (2005), 165–200.

[13] Barry Johnson. 2018. *Algorithmic trading and DMA*. 4Myeloma Press.

[14] Sham M Kakade, Michael Kearns, Yishay Mansour, and Luis E Ortiz. 2004. Competitive algorithms for VWAP and limit order trading. In *Proceedings of the 5th ACM conference on Electronic commerce*. 189–198.

[15] Michaël Karpe, Jin Fang, Zhongyao Ma, and Chen Wang. 2020. Multi-agent reinforcement learning in a realistic limit order book market simulation. In *Proceedings of the first ACM international conference on AI in finance*. 1–7.

[16] Hizuru Konishi. 2002. Optimal slice of a VWAP trade. *Journal of Financial Markets* 5, 2 (2002), 197–221.

[17] Aravind Lakshminarayanan, Sahil Sharma, and Balaraman Ravindran. 2017. Dynamic action repetition for deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

[18] Xiaodong Li, Pangjing Wu, Chenxin Zou, and Qing Li. 2023. Hierarchical deep reinforcement learning for vwap strategy optimization. *IEEE Transactions on Big Data* 10, 3 (2023), 288–300.

[19] Zhiming Li, Junzhe Jiang, Yushi Cao, Aixin Cui, Bozhi Wu, Bo Li, Yang Liu, and Danny Dongning Sun. 2025. Logic-Q: Improving Deep Reinforcement Learning-based Quantitative Trading via Program Sketch-based Tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 18584–18592.

[20] Siyu Lin and Peter A. Beling. 2020. An End-to-End Optimal Trade Execution Framework based on Proximal Policy Optimization. In *Twenty-Ninth International Joint Conference on Artificial Intelligence IJCAI -20*. 4548—4554.

[21] Siyu Lin and Peter A Beling. 2021. An end-to-end optimal trade execution framework based on proximal policy optimization. In *Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence*. 4548–4554.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[23] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. 2006. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*. 673–680.

[24] Brian Ning, Franco Ho Ting Lin, and Sebastian Jaimungal. 2018. Double deep q-learning for optimal execution. *arXiv preprint arXiv:1812.06600* (2018).

[25] Feiyang Pan, Tongzhe Zhang, Ling Luo, Jia He, and Shuoling Liu. 2022. Learn Continuously, Act Discretely: Hybrid Action-Space Reinforcement Learning For Optimal Execution. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 3912–3918. https://doi.org/10.24963/ijcai.2022/543 Main Track.

[26] Molei Qin, Shuo Sun, Wentao Zhang, Haochong Xia, Xinrun Wang, and Bo An. 2024. Earnhft: Efficient hierarchical reinforcement learning for high frequency trading. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 14669–14676.

[27] Matthias Schnaubelt. 2022. Deep reinforcement learning for the optimal placement of cryptocurrency limit orders. *European Journal of Operational Research* 296, 3 (2022), 993–1006.

[28] Yun Shen, Ruihong Huang, Chang Yan, and Klaus Obermayer. 2014. Risk-averse reinforcement learning for algorithmic trading. In *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*. IEEE, 391–398.

[29] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[30] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[31] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. 2017. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* (2017).

[32] Rundong Wang, Hongxin Wei, Bo An, Zhouyan Feng, and Jun Yao. 2021. Commission Fee is not Enough: A Hierarchical Reinforced Framework for Portfolio Management. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 626–633.

[33] Tong Wu, Pan Zhou, Kai Liu, Yali Yuan, Xiumin Wang, Huawei Huang, and Dapeng Oliver Wu. 2020. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. *IEEE Transactions on Vehicular Technology* 69, 8 (2020), 8243–8256.

[34] Yuhuai Wu, Elman Mansimov, Shun Liao, Alec Radford, and John Schulman. 2017. *Openai baselines: Acktr & a2c*. url:https://openai.com/blog/baselines-acktr-a2c

[35] Zekun Ye, Weijie Deng, Shuigeng Zhou, Yi Xu, and Jihong Guan. 2020. Optimal trade execution based on deep deterministic policy gradient. In *International Conference on Database Systems for Advanced Applications*. Springer, 638–654.

[36] Renhao Zhang, Haotian Fu, Yilin Miao, and George Konidaris. 2024. Model-based Reinforcement Learning for Parameterized Action Spaces. In *International Conference on Machine Learning*. PMLR, 58935–58954.

[37] Wentao Zhang, Yilei Zhao, Shuo Sun, Jie Ying, Yonggang Xie, Zitao Song, Xinrun Wang, and Bo An. 2024. Reinforcement learning with maskable stock representation for portfolio management in customizable stock pools. In *Proceedings of the ACM Web Conference 2024*. 187–198.