mathematics

MDPI

Article

Joint Learning of Volume Scheduling and Order Placement Policies for Optimal Order Execution

Siyuan Li [1,*], Hui Niu [2], Jiani Lu [1] and Peng Liu [1]

[1] Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China
[2] Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China
* Correspondence: siyuanli@hit.edu.cn

Abstract: Order execution is an extremely important problem in the financial domain, and recently, more and more researchers have tried to employ reinforcement learning (RL) techniques to solve this challenging problem. There are a lot of difficulties for conventional RL methods to tackle the order execution problem, such as the large action space including price and quantity, and the long-horizon property. As naturally order execution is composed of a low-frequency volume scheduling stage and a high-frequency order placement stage, most existing RL-based order execution methods treat these stages as two distinct tasks and offer a partial solution by addressing either one individually. However, the current literature fails to model the non-negligible mutual influence between these two tasks, leading to impractical order execution solutions. To address these limitations, we propose a novel automatic order execution approach based on the hierarchical RL framework (OEHRL), which jointly learns the policies for volume scheduling and order placement. OEHRL first extracts the state embeddings at both the macro and micro levels with a sequential variational auto-encoder model. Based on the effective embeddings, OEHRL generates a hindsight expert dataset, which is used to train a hierarchical order execution policy. In the hierarchical structure, the high-level policy is in charge of the target volume and the low-level learns to determine the prices for a series of the allocated sub-orders from the high level. These two levels collaborate seamlessly and contribute to the optimal order execution policy. Extensive experiment results on 200 stocks across the US and China A-share markets validate the effectiveness of the proposed approach.

Keywords: order execution; hierarchical reinforcement learning; imitation learning representation learning

MSC: 37M10

check for updates

Citation: Li, S.; Niu, H.; Lu, J.; Liu, P. Joint Learning of Volume Scheduling and Order Placement Policies for Optimal Order Execution. *Mathematics* **2024**, *12*, 3440. https://doi.org/10.3390/math12213440

Academic Editors: Luca Di Persio and Matteo Garbelli

Received: 7 September 2024
Revised: 18 October 2024
Accepted: 2 November 2024
Published: 4 November 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In stock markets, financial institutions usually adjust the size of their holdings to manage their portfolios based on their risk tolerance. However, accomplishing such adjustments in a single large trade is excessively costly and even unfeasible due to market liquidity constraints. Therefore, market participants need to divide a large order into smaller sub-orders and execute them sequentially over an extended period, typically hours or one day, which is typically called order execution [1,2]. During the order execution process, agency brokers aim to complete the order fulfillment while simultaneously achieving a favorable trading price. For example, a trader aimed at selling a specific number of shares will execute more volumes at higher prices within the trading period. To achieve this goal, the traders frequently adjust their quotes to decide the prices, sizes, and types of their orders based on the dynamic market conditions. The order execution problem [3] is fundamentally regarded as a sequential decision-making problem, where deep reinforcement learning (RL) techniques have shown great promise.

Directly applying existing RL algorithms to the order execution problem is challenging, such as the high-dimensional action space and the long-horizon problems. Since the

duration of a trading period corresponds to a long episode length in the high-frequency trading formulation, such a long-horizon complex decision-making process can be unfolded into two stages [4]: (i) **volume scheduling stage**: the volume assigned for a specific asset is divided into smaller slices to be executed within a sub-period of several minutes; and (ii) **order placement stage**: the segmented volumes are transformed into a series of orders submitted to the market, with further order details such as price and quantity specified. As a compromise solution, current RL-based order execution approaches typically treat these two stages as distinct tasks, concentrating on learning the policy for either task individually. A rich line of research utilizes macro-level market information to address coarse-grained volume scheduling [5–7], assuming that the allocated volume can be entirely executed at a specific price. This assumption leads to a non-negligible deviation between the expected and the actual execution price, which causes potential losses. Other methods aim at minimizing trading costs during short-term order placement [8–10]. Nevertheless, the target volume for these tasks is usually fixed and pre-defined, which is not aligned with real-world trading scenarios. While modeling order execution with these two separate tasks offers an easier solution to policy learning, it is less realistic.

In the order execution problem, volume scheduling and order placement significantly influence each other. On the one hand, the allocated volume from the volume scheduling stage directly influences the trading pattern of the order placement strategy. On the other hand, the true execution outcomes of the allocated volume substantially influence the decision-making process of subsequent steps in volume scheduling. Inspired by the human trader workflow, we propose a novel order execution approach, called OEHRL, which can jointly learn the policies for volume scheduling and order placement. Specifically, we model the order execution problem with a hierarchical RL framework, where a high-level policy is learned to schedule volumes, and a low-level policy is learned to place orders based on the allocated volumes. As volume scheduling and order placement need to consider different aspects in the markets, we propose a novel representation learning module to effectively extract multi-granularity state embeddings for the hierarchical decision-making process. The primary contributions of this paper are summarized as follows:

- To distill multi-granularity embeddings from noisy market data, we develop a new representation learning module with the variational recurrent neural network [11].
- We propose a hierarchical RL-based order execution approach, which can jointly learn the policies for volume scheduling and order placement. The hierarchical decision-making is built upon the state embeddings extracted by the representation learning module.
- Extensive experiment results demonstrate that the proposed approach outperforms the state-of-the-art baselines in order execution. Ablation studies validate the effectiveness of the various components in the proposed approach.

In the rest of this paper, we first review the related works and then introduce the preliminary knowledge about order execution. Next, we present the proposed approach, which is composed of multi-granularity representation learning and hierarchical policy learning. After that, we conduct experiments in both the US market and the China A-share market. The results and discussion are presented in the following section. Finally, we conclude and point out some future directions. We note that a shorter conference version of this paper appeared in [12]. We have improved this approach and enriched the experiments. The difference between this paper and the conference paper is larger than 70%.

2. Related Work

In this section, we first introduce the human trader workflow for order execution, and then review the related work on order execution from two perspectives, the traditional approaches with data analysis, and recent methods based on RL.

2.1. Human Trader Workflow for Order Exeuction

As shown in Figure 1, human traders deal with the influence between volume scheduling and order placement in a two-level hierarchical way. As volume scheduling is related to long-term profits and order placement is about the best execution within the short term, expert agency brokers first analyze the short-term and long-term trends based on the market conditions. Using these analyses, they schedule the volumes as high-level decisions, and then, based on the target volumes, they place the orders in the trading environment as low-level actions.

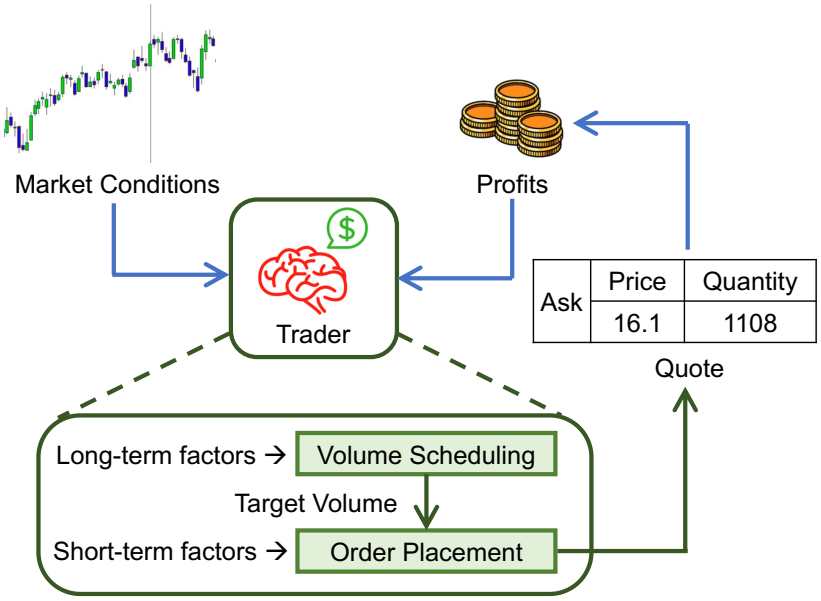


Figure 1. The workflow of human order execution traders.

2.2. Traditional Approaches for Order Execution

Traditional finance approaches for order execution are typically model-based. These methods assume that market price movements follow stochastic processes like Brownian motion, and then apply stochastic optimal control methods to analytically derive the volume trajectory [13–16]. Since these assumptions are not applicable to real-world scenarios, these methods might not be effective in actual trading markets.

In real-world trading agencies, rule-based strategies such as time-weighted average price (TWAP) strategy [15] and volume-weighted average price (VWAP) strategy [17] are quite popular [9]. For instance, the TWAP strategy divides a large order into equal-sized sub-orders and executes each within uniformly distributed time intervals. The VWAP strategy initially estimates the average volume traded for each time interval based on historical data and then divides the order according to these estimates. Despite their simplicity, TWAP and VWAP remain widely favored today because their execution costs consistently align with the market.

2.3. Reinforcement Learning for Order Execution

In addition to the aforementioned analytical methods, RL provides an alternative perspective for optimizing the order execution problem. Previous RL-based approaches extend the model-based assumptions mentioned above to evaluate how much RL algorithms can enhance analytical solutions [5,18]. They also assess whether the Markov Decision Process (MDP) remains viable within the specified market assumptions [19]. However, these RL-based approaches hinge on financial model assumptions about market dynamics, lacking practical applicability.

Another branch of RL methods abandons these assumptions and leverages model-free RL to optimize execution strategies. Several studies [6,20] propose variations of

DQN to choose discrete volumes. These adaptations aim to address the high-dimensional nature and complexity of financial markets using deep neural networks. In contrast to the manually crafted attributes, a PPO-based optimal execution framework is designed to make decisions based on raw level-2 market data [21]. In addition, a policy distillation paradigm has been deployed to achieve order execution [7], where a distilled PPO agent is used to determine optimal order-splitting volumes. The above methods simply use market orders and focus on the problem of volume-splitting.

Other works deal with the stage of optimal order placement, leveraging limit orders to provide more accurate trading decisions. Nevmyvaka et al. [8] pioneered the application of model-free RL for order placement, employing Q-Learning [22] to guide the agent in selecting the limit price. However, their approach is constrained as it only accounted for a limited set of discrete actions, primarily focusing on a few bid and ask prices. Akbarzadeh et al. [23] proffered an online learning algorithm based on the Almgren–Chriss model, learning the optimal policy by pure exploitation. Recently, Pan et al. [9] introduced a hybrid framework, where the agent first scopes an action subset and then chooses a specific limit price. Additionally, Chen et al. [10] proposed a cost-efficient RL approach under a dynamic market environment. Despite these advances, existing methods suffer from losses and inefficiency due to the complex action space and inadequate market representations.

Moreover, all these existing methods focus on a single task, either volume splitting or order placement, providing a partial solution to the optimal order execution problem. In contrast, the proposed method simultaneously tackles both tasks within a hierarchical framework.

3. Preliminaries

In this section, we introduce preliminary knowledge about this work, including the observable data in the market, OHLCV, basic elements in the trading procedures, and the environment formulation in RL.

3.1. OHLCV Data

An OHLCV vector is a type of bar chart derived directly from the financial market, showing the market dynamics. OHLCV data is often used to summarize market conditions over a period of time and to forecast market trends over a correspondingly long period of time in the future. We denote the OHLCV vector at timestep t as $\boldsymbol{x}_t$, which includes the opening price, highest price, lowest price, closing price, and volume.

3.2. Market Orders and Limit Orders

A market order is an order to buy or sell a specific quantity of a financial instrument, executed immediately at the best available price in the market. Comparatively, A limit order is an order to buy or sell a financial instrument, setting the highest or lowest acceptable price, defined as $l = (p_{limit}, \pm q)$, where p_{limit} represents the submitted target price, q represents the submitted target quantity, and $\pm$ represents the trading direction (long/short).

3.3. Limit Order Book

A limit order book (LOB) is a collection of publicly available, aggregated information about limit orders placed by all participants in the market [24]. Limit orders will remain in the order book until executed, canceled, or expired. As depicted in Figure 2, a m level LOB at timestep t is denoted as $\{(p_{-m}, q_{-m}, t_{-m}); \ldots; (p_{-1}, q_{-1}, t_{-1}); (p_1, q_1, t_1); \ldots; (p_m, q_m, t_m)\}$, where p_{-i} and q_{-i} represents the price and quantity at level i on the bid side, and p_i and q_i are the corresponding variables on the ask side. p_{-1} is the best bid price, and p_1 is the best ask price. A LOB is a price–time-priority queue pair, i.e., lower sell price orders for the same asset are the most probable to be filled, as are higher bid price orders. If there are two orders with the same price to sell or buy, the first submitted one will be traded first.

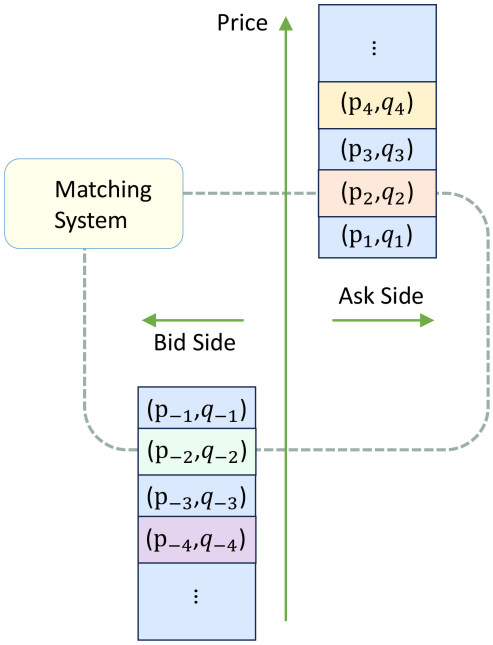


Figure 2. An example limit order book at timestep t.

3.4. Matching System

If the best bid price is higher than the best ask price, the matching system will quickly match and trade the two orders. A matching system, often referred to as an order matching system or order matching engine, is a computerized system that matches buy and sell orders in a financial market. It uses specific rules to determine the trades that will be executed based on the orders in the LOB.

3.5. Markov Decision Processes

In RL, an environment is formulated as a Markov Decision Process (MDP) [25], denoted as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma, T)$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition function specifying the conditional transition probabilities between states, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the bounded reward function, $\gamma \in (0, 1]$ is the discount factor and T is a time horizon. After taking action a_t at state s_t, an agent switches to the next state s_{t+1} according to the transition function $P(s_{t+1}|s_t, a_t)$ and obtains a reward $r_t = R(s_t, a_t, s_{t+1})$. The transition function in the order execution problem is simulated by the matching system. To maximize the cumulative reward $\mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t r_t]$, we need to optimize a policy $\pi(a_t|s_t)$ that outputs an action a_t for a given state s_t, where T denotes the ending time of the whole trading process.

4. Approach

In daily order execution, traders analyze market conditions and formulate their trading strategies. Subsequently, they submit their orders specifying desired prices and quantities to the matching system. The matching system executes these orders at the most favorable available prices since the primary goal of optimal execution is to achieve the target fulfillment while trading at a more advantageous price. Then, it will provide execution results to the traders. The whole process costs lots of human labor and is heavily influenced by the subjective decisions of the traders.

To accomplish better order execution, we propose a novel automatic order execution approach based on hierarchical RL, called OEHRL. An overview of the proposed approach is shown in Figure 3. The proposed approach is composed of two components: representation learning and hierarchical policy learning. As financial data is noisy and stochastic, we need to extract the prominent features from the sequential market data to support effective decision making. Furthermore, since the order execution problem involves two

parts, volume scheduling and order placement, and these two parts heavily affect each other, we develop a hierarchical structure to model this influence and learn the order execution policy.

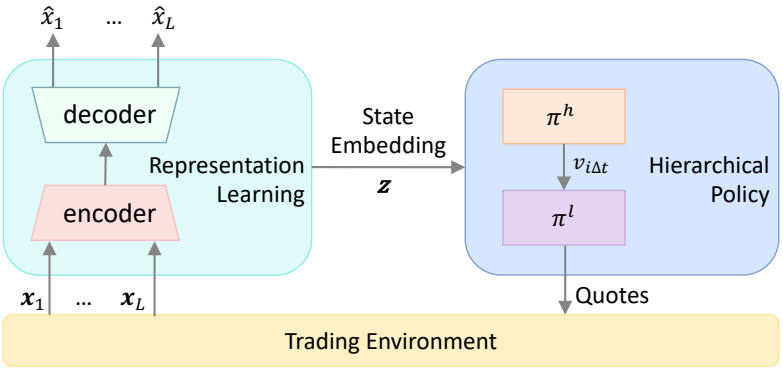


Figure 3. An overview of the proposed OEHRL framework.

In the following, Section 4.1 elaborates on the multi-granularity representation module. Section 4.2 introduces how to utilize these representations to formulate the hierarchical MDP, and how to learn an effective hierarchical order execution policy.

4.1. Multi-Granularity Representation Learning

As the decision-making process of volume scheduling necessitates long-term market representation while the order placement task relies more on short-term market fluctuation and liquidity, we propose a novel multi-granularity representation learning approach based on the variational recurrent neural network (VRNN) [11] structure.

The temporal dimension in the financial data contains much information. To achieve effective order execution, we need to extract the underlying relationship from the sequential data. It is well known that recurrent neural networks (RNNs) [26] excel at dealing with data sequences. As important as the network structure, the learning objective also plays a critical role in the learned embeddings. Prediction-based learning objectives may be helpful in representation learning from noisy financial data. However, it is hard to select an appropriate prediction target, i.e., with the observation $\mathbf{x_{1:t}}$ (Here the notation $\{1:t\}$ denotes a data sequence from timestep 1 to timestep t), it is challenging to figure out whether predicting $\mathbf{x_{t+i}}$ or predicting $\mathbf{x_{t+j}}$ can produce better results, where $i \neq j$. Aiming at learning multi-granularity representations including long-term and short-term factors, the difficulty of selecting the proper prediction target increases. Inspired by the VRNN approach, we propose to utilize the unsupervised variational auto-encoder (VAE) [27] paradigm to conduct representation learning. The loss function for the representation learning network is as follows:

$$\begin{aligned}\mathcal{L} =& \mathbb{E}_{q(\mathbf{s}_{\leq L}|\mathbf{x}_{\leq L})}[\sum_{t=1}^{L}(\beta KL(q_{pos}(\mathbf{s}_t|\mathbf{x}_{\leq t},\mathbf{s}_{<t})||p_{prior}(\mathbf{s}_t|\mathbf{x}_{<t},\mathbf{s}_{<t}))\\ &-\log p_{like}(\mathbf{x}_t|\mathbf{s}_{\leq t},\mathbf{x}_{<t}))],\end{aligned} \tag{1}$$

where q_{pos} denotes the approximated posterior distribution modeled by the encoder network ψ_{enc}, p_{like} denotes the conditional likelihood distribution modeled by the decoder network ψ_{dec}, p_{prior} is the prior distribution, L denotes sequence length, and β is a scaling factor following the β-VAE method. Different from the standard VAE, which only considers learning the embedding $\mathbf{s_t}$ from observation $\mathbf{x_t}$, we aim to model the relationship in sequential data, so we take the effect of $(\mathbf{x_{<t}}, \mathbf{s_{<t}})$ into consideration, and convert all the distributions in the standard VAE into the distributions conditioned on the sequence $(\mathbf{x_{<t}}, \mathbf{s_{<t}})$ as shown in Equation (1). Then, the problem is how to model the past data

sequence. The RNN structure provides a good solution, which uses a hidden variable $\mathbf{h_{t-1}}$ to model $(\mathbf{x_{<t}}, \mathbf{s_{<t}})$ with the following recurrence equation:

$$h_{t-1} = f_\theta(\psi_x(x_{t-1}), \psi_s(s_{t-1}), h_{t-2}), \tag{2}$$

where ψ_x denotes the encoder for the observation $\mathbf{x}$ (For data at different timesteps, we use the same encoder), ψ_s denotes the encoder for the state embedding $\mathbf{s}$, and f_θ is a deterministic non-linear transition function, which is implemented with a gated recurrent unit (GRU) [28]. With Equation (2), the distributions in Equation (1) can be derived as follows:

$$\begin{aligned} q_{pos}(\mathbf{s}_t|\mathbf{x}_{\leq t}, \mathbf{s}_{<t}) &= \psi_{enc}(\psi_x(\mathbf{x_t}), \mathbf{h_{t-1}}), \\ p_{prior}(\mathbf{s}_t|\mathbf{x}_{<t}, \mathbf{s}_{<t}) &= \psi_{prior}(\mathbf{h_{t-1}}), \\ p_{like}(\mathbf{x}_t|\mathbf{s}_{\leq t}, \mathbf{x}_{<t}) &= \psi_{dec}(\psi_s(\mathbf{s_t}), \mathbf{h_{t-1}}). \end{aligned} \tag{3}$$

All the neural networks mentioned above are optimized with the loss function in Equation (1), including those denoted with ψ and θ. The outputs of the networks in Equation (3) are the means and variances of the corresponding distributions.

Although the GRU has a strong ability to tackle sequential data, it still has a forgetting problem due to vanishing gradients for long and complex sequences. In contrast, for a shorter sequence, the forgetting problem is less obvious. Therefore, we propose to segment the long sequence of $\mathbf{s}$ with length L to shorter sequences with length k and extract higher-level representation $\mathbf{m}$ to model a longer-term temporal dependency. As shown in Figure 4, we build a three-level hierarchical representation learning framework, where $\mathbf{m}$ is learned from $\mathbf{s}$ with a VRNN encoder, and $\mathbf{l}$ is learned from $\mathbf{m}$ with another VRNN encoder. In this way, the highest-level representation $\mathbf{l}$ can model a longer-term temporal relationship. The representations $\mathbf{s}, \mathbf{m}$, and $\mathbf{l}$ compose multi-granularity state embeddings, and are pre-trained before policy learning in an unsupervised manner. These embeddings are concatenated as a total state embedding $\mathbf{z_t} = \mathbf{s_t}||\mathbf{m}_{\lfloor \mathbf{t/k} \rfloor}||\mathbf{l}_{\lfloor \mathbf{t/k^2} \rfloor}$ and passed to the hierarchical order execution as an observation. The next subsection defines the hierarchical order execution policy and presents how this policy is optimized.

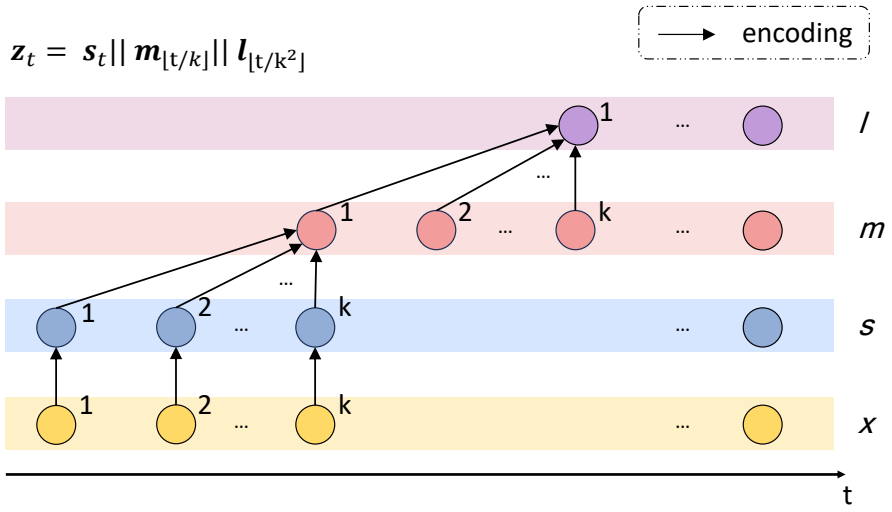


Figure 4. Multi-granularity representation learning.

4.2. Hierarchical Order Execution Policy Learning

This section proposes a hierarchical order execution framework that jointly learns the policies for volume scheduling and order placement. The volume scheduling process focuses on long-term profitability, while order placement focuses on best execution in the short term, but not necessarily long-term profitability. As aforementioned, a major challenge to intraday order execution is to balance the multifaceted objectives of different decision processes.

These two well-connected decision processes, coarse-grained volume scheduling and fine-grained order placement, are formulated as MDPs as follows (see Figure 5):

$$MDP^h = (\mathcal{S}^h, \mathcal{A}^h, \mathcal{P}^h, R^h, \gamma, T^h),$$

$$MDP^l = (\mathcal{S}^l, \mathcal{A}^l, \mathcal{P}^l, R^l, \gamma, T^l),$$

which aligns with a two-tier HRL structure, of which high-level RL policy is π^h and low-level RL policy is π^l. First, we elaborate on how we design the two-level MDP and then present the learning methods for the corresponding policies.

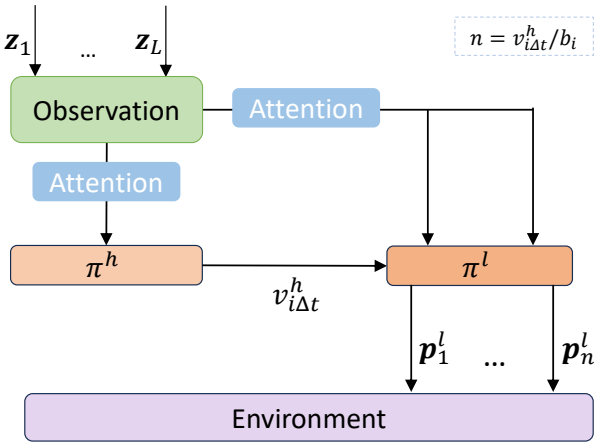


Figure 5. The hierarchical order execution policy.

4.2.1. The High-Level MDP for Volume Scheduling

In the high-level MDP, the whole trading period T is equally segmented into i sub-periods, with the start points denoted as $t \in \{0, 1 \cdot \Delta t, \ldots, (i-1) \cdot \Delta t\}$ where $\Delta t = \lfloor T/i \rfloor$. At the beginning of each sub-period, the high-level policy determines the desired volume to be executed in this sub-period. At the end of the last sub-period, the remaining volume is traded with a market order. The state space, action space, and the reward function for the high-level MDP are designed as follows.

State. The state vector observed by the high-level agent is defined as $\boldsymbol{s}^h_t = (\boldsymbol{m}^h_t, \boldsymbol{p}^h_t)$, where $\boldsymbol{m}^h_t = (\boldsymbol{z}_{t-L}, \ldots, \boldsymbol{z}_{t-1}, \ldots, \boldsymbol{z}_t)$ denotes the market state containing the effective multi-granularity market representations learned in Section 4.1. The private state $\boldsymbol{p}^h_t$ includes the proportion of the remaining volume $\alpha_t = I^h_t / \mathcal{Q}$ and remaining time $1 - t/T$, where I^h_t is the remaining volume at timestep t, and $\mathcal{Q}$ is total volume to execute. With ample time and less remaining volume, the agent will use a conservative strategy, awaiting superior trading opportunities. Conversely, in scenarios with less time and substantial remaining volume, the agent selects more aggressive strategies to achieve a faster turnover.

Action. The action in the high-level MDP is the proportion of the total volume to be executed in this sub-period, and the high-level action space is defined as $[0, \min(c, \alpha_t)]$, where $0 < c < 1$ denotes the proportion of the maximum allowable executed volume in a sub-period. The value of c is prefixed based on the actual task scenario and customer preference. In this paper, taking the number of sub-periods $i = 30$, we set $c = 0.1$. That is, the allowable execution ratio for each sub-period does not exceed 10%. Note that while the action is a continuous variable, the executed volume must be a discrete integer. Therefore, we round down the volume when calculating it. Furthermore, we impose a constraint: while $a^h_t < 0.01$, a^h_t is set to 0. This constraint helps prevent excessively small target volumes.

Reward. To receive long-term profits, we formulate the reward as a volume-weighted price advantage over VWAP for high-level MDP: $r^h_{i\Delta t} = \sum_{j \in \mathcal{O}_i} \frac{q_j}{\mathcal{Q}} \left(\frac{p_j - \tilde{p}}{\tilde{p}} \right)$, where $\mathcal{O}_i$ is the set

of orders traded in the i-th sub-period $[i\Delta t, (i+1)\Delta t)$, p_j and q_j are the price and quantity of the j-th sub-order in $\mathcal{O}_i$ with $\sum q_j \leq a_i^h$, and $\tilde{p}$ is the global VWAP.

4.2.2. The Low-Level MDP for Order Placement

A more realistic way of modeling order placement is a continuous-time decision process rather than adhering to fixed discrete time points for order placement. Inspired by the practices of financial institutions that employ iceberg orders [29], we divide the target order into sub-orders of a fixed basic size b_i, calculated as the average size of orders sent to the market in the last sub-period. As a result, the low-level policy only needs to set the limit price for each sub-order. Whenever the current sub-order is completed or canceled due to timeout (In this work, we set the timeout limit as 5 s.), the agent decides the price of the next sub-order and sends it to the market. By this design, we achieve a continuous-time order placing. The state space, action space, and reward function in the low-level MDP are set as follows.

State. We maintain the multi-granularity representation z_t as the market states for low-level policy. Correspondingly, the remaining time and volume, the private states of the low-level MDP, in the i-th sub-period can be defined as $(t - i\Delta t)/\Delta t$ and $I_t^l / a_{i\Delta t}^h$, respectively. Different from the high-level MDP, where I_t^h is the residual transaction volume of the volume allocated for the entire order execution, here, I_t^l is the residual transaction volume of the volume allocated by the high-level MDP agent for the current sub-period. Notably, we add a new variable $u_t = a_{i\Delta t}^h / b_i$ which implies the task urgency w.r.t. the target volume for the i-th period for low-level policy.

Action. Each low-level action corresponds to a limit order decision with the target price $a_t^l = p_t^{limit}$. It is a n^p-dimension variable $a_t^l \in \{-\frac{n^p-1}{2}, \ldots, -1, 0, 1, \ldots, \frac{n^p-1}{2}\}$ which represents a price with a_t^l ticks below the best ask price.

Reward. The reward function of the low-level policy is a volume-weighted price advantage over sub-period VWAP $\tilde{p}^i$: $r_t^l = \frac{b_i}{a_{i\Delta t}^h}\left(\frac{p_t - \tilde{p}^i}{\tilde{p}^i}\right)$, where p_t is the actual trading price of this executed order.

4.2.3. High-Level Policy Learning

We start by describing the neural network structure utilized by the high-level agent. Then, we introduce a hindsight expert dataset and optimize the high-level policy by RL combined with imitation learning.

Network Structure. As illustrated in Figure 5, to extract useful features from multi-granularity representations for a global view of price advantage, we first apply a temporal attention mechanism on the market representation $\boldsymbol{m}_t = (\boldsymbol{z}_{t-L}, \ldots, \boldsymbol{z}_{t-1}, \boldsymbol{z}_t)$ for the high-level policy:

$$\begin{aligned} e_{t-l}^h &= \boldsymbol{V}_e \tanh(\boldsymbol{W}_1[\boldsymbol{z}_{t-l}; \boldsymbol{z}_t]), \forall l \in \{1:L\}, \\ att_{t-l}^h &= \frac{\exp(e_{t-l}^h)}{\sum_{j=0}^{L} \exp(e_{t-j}^h)}, \\ a_t^h &= c \cdot \text{sigmoid}\left(\boldsymbol{W}_2 \cdot \text{concat}\Big(\sum_{l=0}^{L} att_{t-l}^h \cdot \boldsymbol{z}_{t-l}, \boldsymbol{p}_t^h\Big) + \boldsymbol{b}_t^h\right), \end{aligned} \tag{4}$$

where $\boldsymbol{V}_e$, $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ and $\boldsymbol{b}_t$ are free parameters to optimize, att_t is the normalized attention weight output by the temporal attention module, $\boldsymbol{p}_t^h$ is the high-level private state, and c is the pre-defined action limitation.

Hindsight Expert Datasets. To set an auxiliary goal for high-level policy, we introduce intraday greedy actions as the expert actions. In hindsight, we create a trading expert who always assigns more sell volumes at higher prices and, therefore, can necessarily make profits. The trading volume is first allocated to the sub-period with the highest price and then the remaining volume is allocated to other sub-periods according to the VWAP of these

sub-periods. This is a suboptimal policy since the actual trading price executed by the low-level agent might not be the true VWAP. As the expert dataset contains reasonable volume scheduling behaviors leveraging future information, the agent benefits from imitation learning techniques and the guidance of the expert. Thus, the proposed method can enable more efficient exploration and policy learning in the highly stochastic market environment compared to the RL methods without imitation learning. The expert dataset generated in the hindsight manner is denoted as D_E.

RL Incorporated with Imitation Learning. Since the stochastic trading environment induces a hard exploration problem, learning with a pure RL objective is extremely difficult. To promote policy learning in such a complex trading environment, we propose to augment the RL method to imitate the quoting behavior in the expert dataset D_E, and the policy π^h is updated with the deterministic policy gradient [30] as:

$$\begin{aligned}\pi^h = \arg\max_{\pi^h} \Big(& \mathbb{E}_{(\boldsymbol{s}^h, a^h)\sim D^h}\left[Q^h(\boldsymbol{s}^h, \pi^h(\boldsymbol{s}^h))\right] \\ & -\lambda \mathbb{E}_{(\boldsymbol{s}^h, \hat{a}^h)\sim D_E}\left[(\pi(\boldsymbol{s}^h) - \hat{a}^h)^2\right]\Big),\end{aligned} \tag{5}$$

where $\hat{a}^h$ is the expert action, and λ is a balancing coefficient between the Q loss and the behavior cloning (BC) loss. The hyper-parameter λ decreases with the training steps. D^h is the replay buffer for the high-level policy π^h. Q^h is the high-level value function approximating the expected cumulative reward, $Q^h(\boldsymbol{s}_t^h, \boldsymbol{a}_t^h) = \mathbb{E}[\sum_{i=t}^{T} \gamma^{i-t} r_i^h | \boldsymbol{s}_t^h, \boldsymbol{a}_t^h]$.

4.2.4. Low-Level Policy Learning

The low-level order placement is treated as a discrete control problem with 7 relative price levels, $|p^l| = n^p = 7$. Similar to the high-level policy learning, we employ a temporal attention layer to learn effective features for low-level policy. Here we utilize the Dueling Q-Network [31] to decouple state and action to some extent. Formally, The Q function $Q^l(s_t^l, a_t^l)$ is expressed in terms of the state value $V^l(s_t^l)$ and the corresponding advantage function $Adv(s_t^l, a_t^l)$:

$$Q^l(s_t^l, a_t^l) = V^l(s_t^l) + \left(Adv(s_t^l, a_t^l) - \frac{1}{n^p} \sum_{\tilde{a}\in\mathcal{A}^l} Adv(s_t^l, \tilde{a}) \right). \tag{6}$$

The Q function in Equation (6) is trained based on the one-step temporal-difference error as follows:

$$\hat{Q}^l(s_t^l, a_t^l) = r_t^l + \gamma Q^-\left(s_{t+1}^l, \arg\max_{\tilde{a}} Q(s_{t+1}^l, \tilde{a})\right), \tag{7}$$

$$\mathcal{L}_Q = \mathbb{E}_{(s_t^l, a_t^l, s_{t+1}^l, r_t^l)\sim D^l}\left[\left(\hat{Q}^l(s_t^l, a_t^l) - Q^l(s_t^l, a_t^l)\right)^2\right], \tag{8}$$

where D^l denotes the low-level replay buffer, and Q^- denotes the target Q network.

4.2.5. Training Scheme

The financial data including OHLCV data, multiple factors, and LOB data, is complex and high-dimensional. To efficiently use the available data, we utilize a pre-training scheme and iterative training. In the general setting of HRL, the high-level policy and the low-level policy are trained together in a single environment, which suffers from the non-stationary issue. To address this issue, we pre-train the low-level policy in the LOB environment of each asset with various target volumes, deriving multiple low-level policy parameters. By introducing the pre-training scheme, the low-level policy is trained with more diverse interactions, thereby achieving better generalization ability and robustness. Following the iterative training scheme in [8,32], we augment the private state repeatedly in the low-level pre-training. Specifically, we traverse the target quantity from (0, 0) to a max target quantity q^{max} and the remaining time from 0 to a max trading time window T^{max}. By diversifying

the training scenarios, the low-level policy is trained with the augmented private states, thus can generalize to different subtasks assigned by the high-level policy.

5. Experiments

The experiments are conducted to answer the following questions.

- **Q1:** Can the proposed approach OEHRL successfully learn a proper order execution strategy that beats markets? (Section 6)
- **Q2:** Does the high-level agent in the hierarchical framework achieve effective price discovery? (Section 6.1)
- **Q3:** Does the low-level agent aid in further improving the performance? (Section 6.2)
- **Q4:** Is the proposed representation learning module effective? (Section 6.3)
- **Q5:** Can the proposed hindsight expert dataset improve high-level policy learning? (Section 6.4)

In the following, we start by presenting the experiment setup and then demonstrate the experiment results.

5.1. Datasets for the Experiments

To extensively evaluate OEHRL, we have collected data from 200 constituent stocks within two real-world stock indexes spanning the Chinese and US markets to construct our datasets, as summarized in Table 1. The CSI100 dataset comprises 100 component stocks, providing insights into the primary state of the China A-share market. The NASDAQ100 dataset includes the 100 largest non-financial international companies listed on NASDAQ. In the experiments, 80% of the trading days are utilized as the training dataset, while the remaining 20% constitute the test dataset. The trading task is to sell the shares of 5% of the entire market trading volume of each asset during 4 h following the market's opening every day, i.e., $T = 240$ min. The episode length of the high-level policy is 240, and the execution time for the low-level policy is $\Delta t = 1$ min. We assume that the trading agent is available to the level-II LOB, which contains the full depth of orders on both the ask and bid sides.

Table 1. Experiment datasets.

Dataset	LOB	Num of Stocks	From	To
CSI100	Level-II	100	22/09/01	23/09/01
NASDAQ100	Level-II	100	22/06/08	23/06/08

5.2. Baselines

OEHRL is compared with two categories of order execution methods as follows.

Rule-Based and Traditional Methods

- **TWAP** strategy evenly divides an order into T segments and executes an equal share quantity at each time step [15].
- **VWAP** strategy aims to closely align the execution price with the true market average price, allocating orders based on the empirically estimated market transaction volume from the previous 10 trading days [17].
- **AC** (Almgren–Chriss) model analytically determines the efficient frontier for optimal execution. We only focus on the temporary price impact to ensure a fair comparison [13].

RL-Based Methods

- **DDQN** (Double Deep Q-network) is a value-based RL method that adopts state engineering [6].
- **PPO** is a policy-based RL method [21] that utilizes a PPO algorithm with a sparse reward to train an agent with a recurrent neural network for feature extraction.

- **OPD** leverages RL with policy distillation to determine the size of each sub-order (volume scheduling) and place market orders [7].
- **HALOP** first, automatically scopes an action subset according to the market status and subsequently, chooses a specific discrete limit price from the action subset using a discrete-control agent [9].

5.3. Evaluation Metrics

All the order execution policies have been evaluated based on the following metrics considering both profits and risks:

- **Price Advantage over VWAP (PA).** We use the average excess return over VWAP-with-market-order as the return metric, displayed in basis points (bps, 1 bp = 0.01%). Note that here, VWAP is calculated with true trading volume instead of its estimation adopted in the VWAP baseline.
- **Win Ratio (WR).** WR measures the ratio of the days that the agent beats VWAP, defined as:

$$WR_T = \frac{\sum_{i \in \mathcal{O}_\mathcal{T}} \mathbb{I}(PA_i > 0)}{\sum_{i \in \mathcal{O}_\mathcal{T}} \left(\mathbb{I}(PA_i > 0) + \mathbb{I}(PA_i \leq 0)\right)}. \tag{9}$$

  where $\mathbb{I}()$ is an indicator function.
- **Gain–Loss Ratio (GLR).** GLR reports the gain–loss ratio, defined as:

$$GLR_T = \frac{\mathbb{E}(PA|PA > 0)}{\mathbb{E}(PA|PA < 0)}. \tag{10}$$

- **Averaged Final Inventory (AFI).** AFI refers to the averaged remaining inventory at the end of the trading period, evaluating the non-fulfillment risk.

5.4. Hyperparameter Setup

The values of the hyperparameters in the proposed approach are listed in Table 2. For a fair comparison, we tune the hyperparameters of all the baseline methods for the maximum PA value on the validation dataset.

Table 2. Hyperparameters of OEHRL.

Parameter	Value
Length L	128
Episode length T	240
Execution time Δt	1
Price level n^p	7
Imitation coefficient λ	1.0
Minibatch size	4
VAE coefficient β	0.5
Learning rate	3×10^{-4}

6. Results and Discussion

To answer Question **Q1**, we compare OEHRL with seven baselines on five meaningful financial metrics introduced in the previous section. The comparison results on the China A-share market and the US market are listed in Tables 3 and 4, respectively. As demonstrated in Table 3, on the CSI100 dataset, the proposed method significantly outperforms the seven state-of-the-art baselines, including the strongest RL-based baseline HALOP. OEHRL achieves the highest price advantage (3.31 bps), and also a high winning ratio (0.72), albeit with the smallest AFI (0.02). In contrast to the three flat RL benchmarks, the strongest baseline, HALOP, employs a continuous policy to initially determine the range of price levels and subsequently employs a discrete agent to select the price level. Benefiting from the price trend prediction module, HALOP achieves significant improvements in these

metrics. Different from HALOP, where the two agents follow the same decision frequency, OEHRL operates on multiple levels of temporal granularity and takes advantage of the multi-granularity representations. Therefore, OEHRL is better able to identify favorable trading opportunities from a longer-term perspective. Similarly, on the NASDAQ100 dataset in Table 4, OEHRL significantly improves the performance on the PA, WR, and AFI metrics (14%, 10%, and 70%, respectively) compared to the strongest baseline, HALOP. The comparison results across these 200 stocks with various baselines validate the effectiveness of the proposed multi-granularity HRL framework.

Table 3. The comparison results of the proposed method and the benchmarks on CSI100.

Methods	PA (bps)↑	PA-Std (bps)↓	WR↑	GLR↑	AFI↓
TWAP	−0.12	**3.12**	0.49	0.97	**0**
VWAP	−3.29	4.87	0.39	0.78	0.04
AC	−1.24	4.21	0.45	1.02	0
DDQN	−1.21	7.09	0.47	0.99	0.05
PPO	−0.98	7.01	0.52	0.92	0.03
OPD	0.32	6.78	0.53	1.07	0.10
HALOP	2.89	6.12	0.65	1.12	0.07
OEHRL	**3.31**	6.23	**0.72**	**1.37**	0.02

Table 4. The comparison results of the proposed method and the benchmarks on NASDAQ100.

Methods	PA (bps)↑	PA-Std (bps)↓	WR↑	GLR↑	AFI↓
TWAP	−0.19	4.31	0.49	1.01	**0**
VWAP	−2.89	4.72	0.35	0.79	0.02
AC	−2.31	4.39	0.39	0.96	0
DDQN	−1.98	6.89	0.45	0.97	0.03
PPO	−1.20	6.79	0.46	0.96	0.04
OPD	1.03	6.82	0.54	1.05	0.12
HALOP	2.75	5.87	0.63	1.13	0.05
OEHRL	**3.14**	5.69	**0.74**	**1.28**	0.01

6.1. Analysis of the High-Level Policy

Figure 6 demonstrates the relationship between the high-level action and the closing price trend. The results indicate that these two terms are highly correlated. As shown in Figure 6a, when the closing price is low, in the beginning, the closing price is high, and the high-level policy allocates most of the volume. In contrast, with the drop in the price, the high-level actions are nearly 0. This phenomenon implies that the high-level agent has a price prediction ability to some extent, and can automatically adapt the volume scheduling. Similarly, in Figure 6b,c, we observe that the high-level actions align with the price trend, which gives a positive answer to Question **Q2**.

6.2. Ablation Study on RL-Based Low-Level Policy

To investigate whether the RL-based low-level policy is effective, we conduct an ablation study by replacing the policy learned by the RL method in OEHRL with a TWAP-with-market-order strategy in the order placement MDP. This experiment is conducted on the CSI100 dataset, and the results are shown in Table 5. These results indicate that the low-level policy learned by the RL method can enhance the profitability of the high-level volume scheduling policy, answering Question **Q3**.

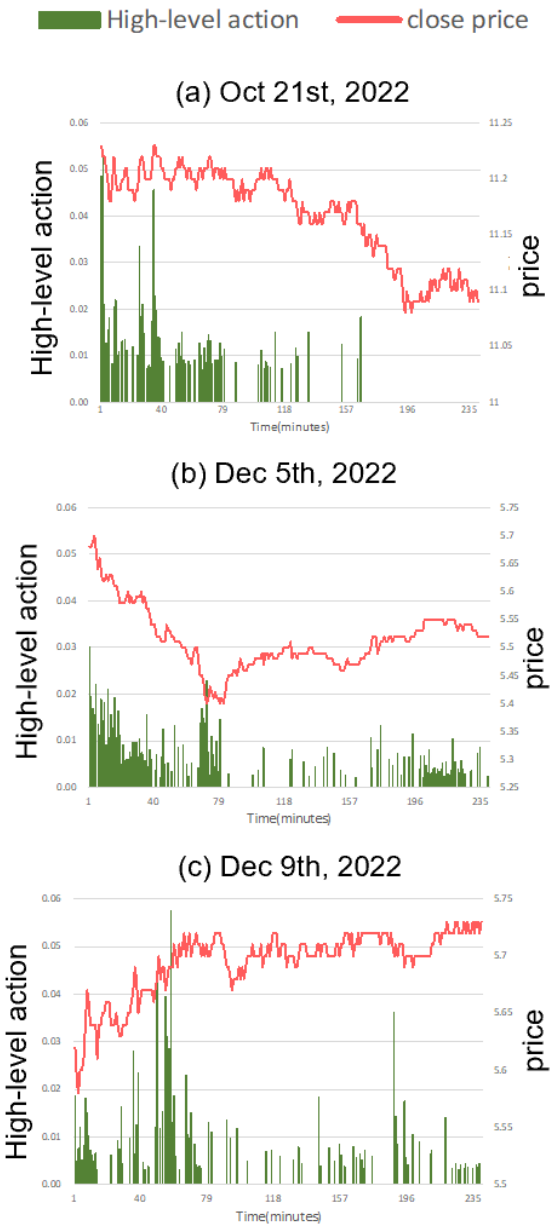


Figure 6. The high-level actions and the closing price trends.

Table 5. Ablation study on low-level policy learning.

Methods	PA (bps) ↑	WR ↑
w/o low-level learning	2.29	0.65
w/ low-level learning	**3.31**	**0.72**

6.3. Ablation Study on Multi-Granularity Representations

In this subsection, we conduct an ablation study on the multi-granularity representation learning module in OEHRL. On the CSI100 dataset, we compare the performance of OEHRL with a variant version, which takes the raw observations of OHLCV data as the input of the hierarchical policy. The results are shown in Table 6. There are significant improvements (+8.57 bps on PA and 0.37 on WR) brought by the multi-granularity representation module. As a result, to answer Question **Q4**, the multi-granularity representation module plays a key role in the OEHRL framework by providing effective state embeddings for the two-level MDPs.

Table 6. Ablation study on Multi-Granularity Representations.

Methods	PA (bps) ↑	WR ↑
w/o representation learning	−5.26	0.35
w/ representation learning	**3.31**	**0.72**

6.4. Ablation Study on Imitation Learning

In the proposed framework, the high-level policy is optimized with the method combining RL and imitation learning, which relies on a novel method to construct a hindsight expert dataset. In this subsection, we conduct an ablation study on imitation learning by removing the behavior cloning loss during high-level policy learning. The results demonstrated in Table 7 imply that the incorporation of imitation learning techniques greatly empowers the price discovery ability of the high-level policy.

Table 7. Ablation study on imitation learning.

Methods	PA (bps) ↑	WR ↑
w/o imitation learning	1.36	0.43
w/ imitation learning	**3.31**	**0.72**

To further evaluate the proposed hindsight expert dataset, we depict the parameter sensitivity of the imitation learning coefficient λ in Figure 7. Compared to values significantly below 0.1 (near no imitation) or above 3 (imitation-dominant), a value around $\lambda = 1$ shows more favorable performance, so we set λ as 1 in all the experiments above. This phenomenon implies that due to the mutual influence between volume scheduling and order placement, directly imitating the hindsight expert in high-policy learning may lead to sub-optimal trade decisions.

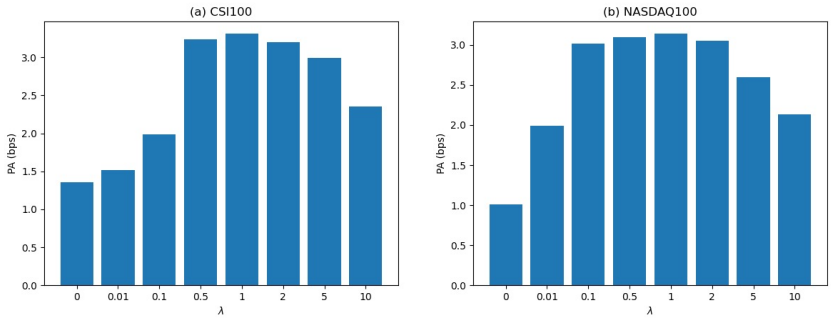


Figure 7. Experiment results with different value selection of λ.

7. Conclusions

To solve the optimal order execution problem, we propose a novel order execution framework based on hierarchical RL. The proposed framework, called OEHRL, is composed of a double-stage framework. OEHRL first employs a multi-layered variational recurrent network to distill multi-granularity market representations. Subsequently, the effective representations are utilized in a hierarchical MDP framework. To stabilize the learning process, OEHRL first pre-trains a low-level agent to determine the price for order placement and then trains a high-level policy to split volumes in coarse granularity on top of the low-level policy. We have conducted extensive experiments on two types of market data to evaluate the proposed framework. The results demonstrate that OEHRL significantly outperforms the state-of-the-art baselines. The ablation studies validate the effectiveness of the various components in the proposed framework. As, in this paper, we have not dealt with the influence of multiple intelligent agents, a promising future direction is to consider the influence of multiple traders on the market and build a more realistic order-matching engine.

Author Contributions: Conceptualization, S.L. and H.N.; methodology, S.L. and H.N.; validation, S.L. and H.N.; formal analysis, S.L. and H.N.; writing—original draft preparation, S.L. and H.N.; writing—review and editing, J.L. and P.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 62306088) and Songjiang Lab (Grant No. SL20230309).

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cartea, Á.; Sánchez-Betancourt, L. Optimal execution with stochastic delay. *Financ. Stoch.* **2023**, *27*, 1–47. [CrossRef]
2. Carmona, R.; Leal, L. Optimal execution with quadratic variation inventories. *SIAM J. Financ. Math.* **2023**, *14*, 751–776. [CrossRef]
3. Kath, C.; Ziel, F. Optimal order execution in intraday markets: Minimizing costs in trade trajectories. *arXiv* **2020**, arXiv:2009.07892.
4. Cont, R.; Kukanov, A. Optimal order placement in limit order markets. *Quant. Financ.* **2017**, *17*, 21–39. [CrossRef]
5. Hendricks, D.; Wilcox, D. A reinforcement learning extension to the Almgren-Chriss framework for optimal trade execution. In Proceedings of the 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), London, UK, 27–28 March 2014; pp. 457–464.
6. Ning, B.; Ling, F.H.T.; Jaimungal, S. Double Deep Q-Learning for Optimal Execution. *Appl. Math. Financ.* **2018**, *28*, 361–380. [CrossRef]
7. Fang, Y.; Ren, K.; Liu, W.; Zhou, D.; Zhang, W.; Bian, J.; Yu, Y.; Liu, T.Y. Universal Trading for Order Execution with Oracle Policy Distillation. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021.
8. Nevmyvaka, Y.; Feng, Y.; Kearns, M. Reinforcement learning for optimized trade execution. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006.
9. Pan, F.; Zhang, T.; Luo, L.; He, J.; Liu, S. Learn Continuously, Act Discretely: Hybrid Action-Space Reinforcement Learning For Optimal Execution. In Proceedings of the International Joint Conference on Artificial Intelligence, Vienna, Austria, 23–29 July 2022.
10. Chen, D.; Zhu, Y.; Liu, M.; Li, J. Cost-Efficient Reinforcement Learning for Optimal Trade Execution on Dynamic Market Environment. In Proceedings of the Third ACM International Conference on AI in Finance, New York, NY, USA, 2–4 November 2022; pp. 386–393.
11. Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A.C.; Bengio, Y. A recurrent latent variable model for sequential data. *arXiv* **2015**, arXiv:1506.02216.
12. Niu, H.; Li, S.; Li, J. MacMic: Executing Iceberg Orders via Hierarchical Reinforcement Learning. In Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24, Jeju, Republic of Korea, 3–9 August 2024; Larson, K., Ed.; pp. 6008–6016, Main Track. [CrossRef]
13. Almgren, R.; Chriss, N. Optimal execution of portfolio transactions. *J. Risk* **2001**, *3*, 5–40. [CrossRef]
14. Huberman, G.; Stanzl, W. Optimal liquidity trading. *Rev. Financ.* **2005**, *9*, 165–200. [CrossRef]
15. Bertsimas, D.; Lo, A.W. Optimal control of execution costs. *J. Financ. Mark.* **1998**, *1*, 1–50. [CrossRef]
16. Wang, J.; Zhang, C. Dynamic Focus Strategies for Electronic Trade Execution in Limit Order Markets. In Proceedings of the 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06), San Francisco, CA, USA, 26–29 June 2006; p. 26. [CrossRef]
17. Kakade, S.M.; Kearns, M.; Mansour, Y.; Ortiz, L.E. Competitive algorithms for VWAP and limit order trading. In Proceedings of the 5th ACM conference on Electronic Commerce, New York, NY, USA, 17–20 May 2004; pp. 189–198.
18. Dabérius, K.; Granat, E.; Karlsson, P. Deep Execution-Value and Policy Based Reinforcement Learning for Trading and Beating Market Benchmarks. 2019. Available online: https://ssrn.com/abstract=3374766 (accessed on 1 November 2024).
19. Hu, R. Optimal Order Execution Using Stochastic Control and Reinforcement Learning. Master's Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2016.
20. Lin, S.; Beling, P.A. Optimal liquidation with deep reinforcement learning. In Proceedings of the 33rd Conference on Neural Information Processing Systems, Deep Reinforcement Learning Workshop, Vancouver, BC, Canada, 8–14 December 2019.
21. Beling, P.A.; Lin, S. An End-to-End Optimal Trade Execution Framework based on Proximal Policy Optimization. In Proceedings of the International Joint Conference on Artificial Intelligence, Virtual, 7–15 January 2020.
22. Watkins, C.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
23. Akbarzadeh, N.; Tekin, C.; van der Schaar, M. Online Learning in Limit Order Book Trade Execution. *IEEE Trans. Signal Process.* **2018**, *66*, 4626–4641. [CrossRef]
24. Gould, M.D.; Porter, M.A.; Williams, S.; McDonald, M.; Fenn, D.J.; Howison, S.D. Limit order books. *Quant. Financ.* **2013**, *13*, 1709–1742. [CrossRef]
25. Puterman, M.L. Markov decision processes. In *Handbooks in Operations Research and Management Science*; Elsevier: Amsterdam, The Netherlands, 1990; Volume 2, pp. 331–434.

26. Salehinejad, H.; Sankar, S.; Barfett, J.; Colak, E.; Valaee, S. Recent advances in recurrent neural networks. *arXiv* **2017**, arXiv:1801.01078.
27. Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.P.; Glorot, X.; Botvinick, M.M.; Mohamed, S.; Lerchner, A. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
28. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
29. Esser, A.; Mönch, B. The navigation of an iceberg: The optimal use of hidden orders. *Financ. Res. Lett.* **2007**, *4*, 68–81. [CrossRef]
30. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the International Conference on Machine Learning (PMLR), Beijing, China, 22–24 June 2014; pp. 387–395.
31. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.V.; Lanctot, M.; de Freitas, N. Dueling Network Architectures for Deep Reinforcement Learning. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015.
32. Li, S.; Zheng, L.; Wang, J.; Zhang, C. Learning subgoal representations with slow dynamics. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 3–7 May 2021.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.