Imitation from Learning-based Oracle for Universal Order Execution in Quantitative Finance

Yuchen Fang
arthur_fyc@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Kan Ren
kan.ren@microsoft.com
Microsoft Research
Shanghai, China

Weiqing Liu
weiqing.liu@microsoft.com
Microsoft Research
Beijing, China

Dong Zhou
zhou.dong@microsoft.com
Microsoft Research
Beijing, China

Weinan Zhang
wnzhang@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Yong Yu
yyu@apex.sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

Order execution aims at fulfilling a specific liquidation or acquirement order for a given instrument over a period of time. As a fundamental problem in algorithmic trading, order execution has attracted much effort to find effective execution strategy. Recent years have witnessed the shift from the analytical view with model-based market assumptions to model-free perspective i.e., reinforcement learning, due to its nature of sequential decision optimization. However, common model-free reinforcement learning algorithms suffer from low sample efficiency due to the noisy and imperfect financial market data. In this paper, we propose a novel universal trading policy optimization framework to bridge the gap between the noisy yet imperfect market states and the optimal action sequences for order execution. Particularly, we utilize imitation learning method that can better guide the learning of the common policy towards practically optimal execution. To overcome the difficulty of obtaining expert behavior for order execution, we utilize a learning based oracle with perfect information to approximate the optimal trading strategy. The extensive experiments have shown significant improvements of our method over various strong baselines, with reasonable trading actions.

CCS CONCEPTS

• **Theory of computation → Sequential decision making**; • **Computing methodologies →** *Learning from demonstrations.*

KEYWORDS

reinforcement learning, imitation learning, order execution

ACM Reference Format:
Yuchen Fang, Kan Ren, Weiqing Liu, Dong Zhou, Weinan Zhang, and Yong Yu. 2021. Imitation from Learning-based Oracle for Universal Order Execution in Quantitative Finance. In *DRL4IR, July 15, 2021, Online.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Financial investment, aiming to pursue long-term maximized profits, is usually behaved in the form of a sequential process of continuously adjusting the asset portfolio. One indispensable link of this process is *order execution*, consisting of the actions to adjust the portfolio. Take stock investment as an example, the investors in this market construct their portfolios of a variety of instruments. As illustrated in Figure 1, to adjust the position of the held instruments, the investor needs to sell (or buy) a number of shares through executing an order of liquidation (or acquirement) for different instruments. Essentially, the goal of order execution is two-fold: it does not only requires to fulfill the whole order but also targets a more economical execution with maximizing profit gain (or minimizing capital loss).

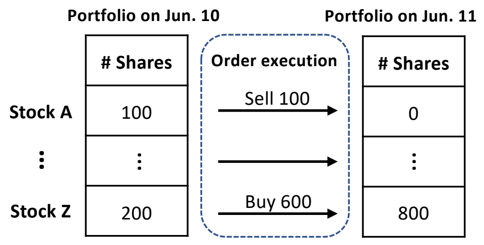

Figure 1: An example of portfolio adjustment which requires order execution.

As discussed in [7], the main challenge of order execution lies in a trade-off between avoiding harmful "*market impact*" caused by large transactions in a short period and restraining "*price risk*", i.e., missing good trading opportunities, due to slow execution. Previously, traditional analytical solutions often adopt some stringent assumptions of market liquidity, i.e., price movements and volume variance, then derive some closed-form trading strategies based on stochastic control theory with dynamic programming principle [1, 2, 5–7]. Nonetheless, these model-based[1] methods are not likely to be practically effective because of the inconsistency between market assumption and reality.

According to the order execution's trait of sequential decision making, reinforcement learning (RL) solutions [24, 30, 31] have

[1]Here 'model' corresponds to some market price assumptions, to distinguish the environment model in reinforcement learning.

been proposed from a model-free perspective and been increasingly leveraged to optimize execution strategy through interacting with the market environment purely based on the market data. As data-driven methods, RL solutions are not necessarily confined by unpractical assumptions and are more capable of capturing the market's microstructure for better execution.

Nevertheless, RL solutions may suffer from a vital issue, i.e., the noisy and imperfect information. For one thing, the noisy data [42] could lead to a quite low sample efficiency of RL methods and thus results in poor effectiveness of the learned order execution policy.

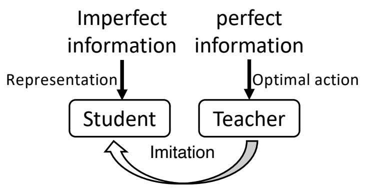

Figure 2: Brief illustration of the imitation learning based on oracle teacher.

More importantly, the only information that can be utilized when taking actions is the historical market information, without any obvious clues or accurate forecasts of the future trends of the market price[2] or trading activities. Such issues indeed place a massive barrier for obtaining a policy to achieve the optimal trading strategy with profitable and reasonable trading actions.

To break this barrier, in this paper, we propose a universal policy optimization framework for order execution. Specifically, this framework introduces imitation learning approach in order for bridging the gap between the noisy yet imperfect market information and the optimal order execution policy. Because reliable human behavior data is not available in such complex and data-rich scenarios, we utilize a learning-based oracle teacher for our policy to imitate from. More concretely, as shown in Figure 2, the teacher, with access to perfect information, is trained to be an oracle to figure out the optimal trading strategy, and the student is learned by imitating the teacher's optimal behavior patterns. For practical usage, only the common student policy with imperfect information, would be utilized *without* teacher or any future information leakage. Also note that, since our RL-based model can be learned using all the data from various instruments, it can largely alleviate the problem of model over-fitting.

The main contributions of our paper include:

- We show the power of learning-based yet model-free method for optimal execution, which not only surpasses the traditional methods, but also illustrates reasonable and profitable trading actions.
- The imitation learning from learning-based teacher paradigm may beneficially motivate the community, to alleviate the problem between imperfect information and the optimal decision making and make up for the lack of data from human expert.
- To the best of our knowledge, it is the first paper exploring to learn from the data of various instruments, and derive a universal trading strategy for optimal execution.

[2]We define 'market price' as the averaged transaction price of the whole market at one time which has been widely used in literature [30, 31].

2 RELATED WORKS

2.1 Order Execution

Optimal order execution is originally proposed in [2] and the main stream of the solutions are model-based. [2] assumed a market model where the market price follows an arithmetic random walk in the presence of linear price impact function of the trading behavior. Later in a seminal work [1], the Almgren-Chriss model has been proposed which extended the above solution and incorporated both temporary and permanent price impact functions with market price following a Brownian motion process. Both of the above methods tackled order execution as a financial model and solve it through stochastic control theory with dynamic programming principle.

Several extensions to these pioneering works have been proposed in the past decades [6, 8, 13, 14] with modeling of a variety of market features. From the mentioned literature, several closed-form solutions have been derived based on stringent market assumptions. However, they might not be practical in real-world situations thus it appeals to non-parametric solutions [31].

2.2 Reinforcement Learning

RL attempts to optimize an accumulative numerical reward signal by directly interacting with the environment [39] under few model assumptions such as Markov Decision Process (MDP). RL methods have already shown superhuman abilities in many applications, such as game playing [22, 26, 38], resource dispatching [23, 40], financial trading [27–29] and portfolio management [19, 43].

Besides the analytical methods mentioned above, RL gives another view for sequential decision optimization in order execution problem. Some RL solutions [10, 15, 17] solely extend the model-based assumptions mentioned above to either evaluate how RL algorithm might improve over the analytical solutions [10, 15], or test whether the MDP is still viable under the imposed market assumptions [17]. However, these RL-based methods still rely on financial model assumption of the market dynamics thus lack of practical value.

Another stream of RL methods abandon these assumptions and utilize model-free RL to optimize execution strategy [24, 30, 31]. Nevertheless, these works faces challenges of utilizing imperfect and noisy market information. And they even train separate strategies for each of several manually chosen instruments, which is not efficient and may result in over-fitting. We propose to handle the issue of market information utilization for optimal policy optimization, while training a universal strategy for all instruments with general pattern mining.

2.3 Imitation Learning

Imitation learning aims at mimic the behaviors of human expert. Demonstrations provided in imitation learning tends to provide more direct guidance than reward functions, especially when the reward signal is sparse or the environment is extremely noisy [18]. [33] first utilize imitation learning to build a autonomous vehicle by cloning the behavior of human drivers. However, the compounding errors in behavior cloning methods can cause critical failures in scenarios that requires long-term planing. Data augmentation [4] and dataset aggregation [35] methods aim at solving these problems by collect more data from human experts to cover as much sampling space as possible. However, all these methods face two problems. First, it is hard and expensive to collect enough human expert's behavior for training of imitation policies. Second, for complex missions such as order execution, no human expert's behaviors can be offered. Our proposed method solves the problem by utilizing a learning-based oracle teacher to offer the behaviors to be imitated.

2.4 Policy Distillation

This work is also related to policy distillation [36], which incorporates knowledge distillation [16] in RL policy training and has attracted many researches studying this problem [12, 21, 32, 41, 44]. However, these works mainly focus on multi-task learning or model compression, which aims at deriving a comprehensive or tiny policy to behave normally in different game environments. Our work dedicates to bridge the gap between the imperfect environment states and the optimal action sequences, which has not been properly studied before.

3 METHODOLOGY

In this section, we first formulate the problem and present the assumptions of our method including MDP settings and market impact. Then we introduce our oracle teacher based imitation learning framework and the corresponding policy optimization algorithm in details. Without loss of generality, we take *liquidation*, i.e., to sell a specific number of shares, as an on-the-fly example in the paper, and the solution to acquirement can be derived in the same way.

3.1 Formulation of Order Execution

Generally, order execution is formulated under the scenario of trading within a predefined time horizon, e.g., one day. We follow a widely applied simplification to the reality [7, 31] by which trading is based on discrete time.

Under this configuration, we assume there are T timesteps $\{0, 1, \ldots, T-1\}$, each of which is associated with a price for trading $\{p_0, p_1, \ldots, p_{T-1}\}$. At each timestep $t \in \{0, 1, \ldots, T-1\}$, the trader will propose to trade a volume of $q_{t+1} \geq 0$ shares, the trading order of which will then be actually executed with the execution price p_{t+1}. With the goal of maximizing the revenue with completed liquidation, the objective of optimal order execution, assuming totally Q shares to be liquidated during the whole time horizon, can be formulated as

$$\arg\max_{q_1, q_2, \ldots, q_T} \sum_{t=0}^{T-1} (q_{t+1} \cdot p_{t+1}), \ s.t. \sum_{t=0}^{T-1} q_{t+1} = Q \, . \tag{1}$$

The average execution price (AEP) is calculated as $\bar{P} = \frac{\sum_{t=0}^{T-1}(q_{t+1} \cdot p_{t+1})}{\sum_{t=0}^{T-1} q_{t+1}} = \sum_{t=0}^{T-1} \frac{q_{t+1}}{Q} \cdot p_{t+1}$. The equations above expressed that, for liquidation, the trader needs to maximize the average execution price $\bar{P}$ so that to gain as more revenue as possible and for acquirement, the objective is reversed.

3.2 Order Execution as a Markov Decision Process

Given the trait of sequential decision-making, order execution can be defined as a Markov Decision Process. To solve this problem, RL solutions are used to learn a policy $\pi \in \Pi$ to trade through proposing an action $a \in \mathcal{A}$ given an observed state $\boldsymbol{s} \in \mathcal{S}$, while maximizing the total expected rewards based on the reward function $R(\boldsymbol{s}, a)$.

State. The observed state $\boldsymbol{s}_t$ at the timestep t describes the status information of the whole system including both the trader and the market variables. Two types of widely-used information [24, 30] for the trading agent are private variable of the trader and public variable of the market. Private variable includes the elapsed time t and the remained inventory $(Q - \sum_{i=1}^{t} q_i)$ to be executed. As for the public variable, it is a bit tricky since the trader only observes *imperfect* market information. Specifically, one can only take the past market history, which have been observed *at or before* the time t, into consideration to make the trading decision for the next timestep. The market information includes price and volume information of each timestep.

Action. Given the observed state $\boldsymbol{s}_t$, the agent proposes the decision $a_t = \pi(\boldsymbol{s}_t)$ according to its trading policy π, where a_t is discrete and corresponds to the proportion of the target order Q. Thus, the trading volume to be executed at the next time can be easily derived as $q_{t+1} = a_t \cdot Q$, and each action a_t is the standardized trading volume for the trader.

According to Eq. (1), we follow the prior works [24, 31] and assume that the summation of all the agent actions during the whole time horizon satisfies the constraint of order fulfillment that $\sum_{t=0}^{T-1} a_t = 100\%$. So that the agent will have to propose $a_{T-1} = \max\{1 - \sum_{i=0}^{T-2} a_i, \pi(\boldsymbol{s}_{T-1})\}$ to fulfill the order target, at the last decision making time of $(T-1)$. However, leaving too much volume to trade at the last trading timestep is of high price risk, which requires the trading agent to consider future market dynamics at each time for better execution performance.

Reward. The reward of order execution in fact consists of two practically conflicting aspects, trading profitability and market impact penalty. As shown in Eq. (1), we can partition and define an immediate reward after decision making at each time.

To reflect the trading profitability caused by the action, we formulate a positive part of the reward as volume weighted *price advantage*:

$$\hat{R}_t^+(\boldsymbol{s}_t, a_t) = \frac{q_{t+1}}{Q} \cdot \overbrace{\left(\frac{p_{t+1} - \tilde{p}}{\tilde{p}}\right)}^{\text{price normalization}} = a_t \left(\frac{p_{t+1}}{\tilde{p}} - 1\right) , \tag{2}$$

where $\tilde{p} = \frac{1}{T} \sum_{i=0}^{T-1} p_{i+1}$ is the averaged original market price of the whole time horizon. Here we apply normalization onto the absolute revenue gain in Eq. (1) to eliminate the variance of different instrument prices and order volume, to optimize a universal trading strategy for all instruments.

Note that although we utilize the future price of the instrument to calculate $\hat{R}^+$, the reward is not included in the state thus would not influence the actions of our agent or cause any information leakage.

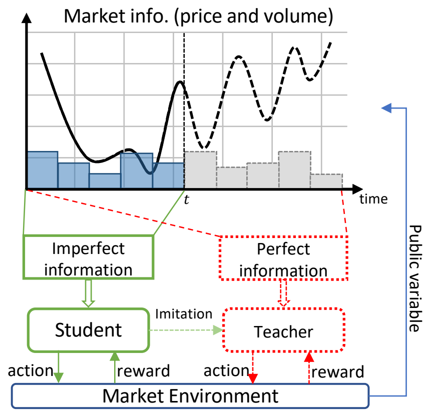

Figure 3: The framework of learning-based teacher and imitation learning. All the modules with dotted lines, i.e. teacher and imitation learning procedure, are only used during the *training* phase, and would be removed during the *test* phase or in practice.

It would only take effect in back-propagation during training. As $\tilde{p}$ varies within different episodes, this MDP might be non-stationary. There are some works on solving this problem [11], however this is not the main focus of this paper and would not influence the conclusion.

To account for the potential that the trading activity may have some impacts onto the current market status, following [24, 31], we also incorporate a quadratic penalty

$$\hat{R}_t^- = -\alpha(a_t)^2 \tag{3}$$

on the number of shares, i.e., proportion a_t of the target order to trade at each decision time. α controls the impact degree of the trading activities. Such that the final reward is

$$\begin{aligned} R_t(s_t, a_t) &= \hat{R}_t^+(s_t, a_t) + \hat{R}_t^-(s_t, a_t) \\ &= \left(\frac{p_{t+1}}{\tilde{p}} - 1\right) a_t - \alpha\,(a_t)^2 \ . \end{aligned} \tag{4}$$

The overall value, i.e., expected cumulative discounted rewards of execution for each order following policy π, is $\mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t R_t]$ and the final goal of reinforcement learning is to solve the optimization problem as

$$\arg\max_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t R_t(s_t, a_t) \right] . \tag{5}$$

3.2.1 Assumptions. Note that there are two main assumptions adopted in this paper. Similar to [24], (i) the temporary market impact has been adopted as a reward penalty and we assume that the market is resilient and will bounce back to the equilibrium at the next timestep. (ii) We either ignore the commissions and exchange fees as the these expense is relatively small fractions for the institutional investors that we are mainly aimed at.

3.3 Imitation Learning and Optimization

In this section, we explain the underlying intuition behind our imitation learning from learning-based policy and illustrate the detailed optimization framework.

3.3.1 Imitation Learning. As aforementioned, our goal is to bridge the gap between the imperfect information and the optimal trading action sequences. An end-to-end trained RL policy may not effectively capture the representative patterns from the imperfect yet noisy market information. Thus it may result in much lower sample efficiency, especially for RL algorithms. To tackle with this, we propose a two stage learning paradigm with teacher-student imitation learning.

We first explain the imperfect and perfect information. In additional to the private variable including left time and the unexecuted order volume, at time t of one episode, the agent will also receive the state of the public variable, i.e., the market information of the specific instrument price and the overall transaction volume within the whole market. However, the actual trader only has the history market information which is collected before the decision making time t. We define the history market information as *imperfect* information and the state with that as the common state $\boldsymbol{s}_t$. On the contrary, assuming that one oracle has the clue of the future market trends, she may derive the optimal decision sequence with this *perfect* information, as notated as $\tilde{\boldsymbol{s}}_t$.

As illustrated in Figure 3, we incorporate a teacher-student learning paradigm.

- **Teacher** plays a role as an oracle whose goal is to achieve the optimal trading policy $\tilde{\pi}_{\boldsymbol{\phi}}(\cdot|\tilde{\boldsymbol{s}}_t)$ through interacting with the environment given the perfect information $\tilde{\boldsymbol{s}}_t$, where $\boldsymbol{\phi}$ is the parameter of the teacher policy.
- **Student** itself learns by interacting with the environment to optimize a common policy $\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}_t)$ with the parameter $\boldsymbol{\theta}$ given the imperfect information $\boldsymbol{s}_t$.

To build a connection between the imperfect information to the optimal trading strategy, we implement a imitation loss L_d for the student. A proper form is the negative log-likelihood loss measuring how well the student's decision matching teacher's action as

$$L_d = -\mathbb{E}_t \left[\log \Pr(a_t = \tilde{a}_t | \pi_{\boldsymbol{\theta}}, \boldsymbol{s}_t; \pi_{\boldsymbol{\phi}}, \tilde{\boldsymbol{s}}_t) \right] , \tag{6}$$

where a_t and $\tilde{a}_t$ are the actions taken from the policy $\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}_t)$ of student and $\pi_{\boldsymbol{\phi}}(\cdot|\tilde{\boldsymbol{s}}_t)$ of teacher, respectively. $\mathbb{E}_t$ denotes the empirical expectation over timesteps.

Note that, the teacher is utilized to achieve the optimal action sequences which will then be used to guide the learning of the common policy (student). Although, with perfect information, it is possible to find the optimal action sequence using a searching-based method, this may require human knowledge and is quite inefficient with extremely high computational complexity. Moreover, another key advantage of the learning-based teacher lies in the universality, enabling to transfer this method to the other tasks, especially when it is very difficult to either build experts or leverage human knowledge.

3.3.2 Policy Optimization. Now we move to the learning algorithm of both teacher and student. Each policy of teacher and student is optimized separately using the same algorithm, thus we describe the optimization algorithm with the student notations, and that of teacher can be similarly derived by exchanging the state and policy notations.

With the MDP formulation described above, we utilize Proximal Policy Optimization (PPO) algorithm [37] in actor-critic style

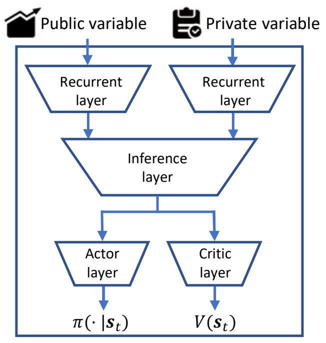

Figure 4: The overall structure of the policy network.

to optimize a policy for directly maximizing the expected reward achieved in an episode. PPO is an on-policy RL algorithm which seeks towards the optimal policy within the trust region by minimizing the objective function of policy as

$$\begin{aligned} L_p(\boldsymbol{\theta}) = & -\mathbb{E}_t\left[\frac{\pi_{\boldsymbol{\theta}}(a_t|\boldsymbol{s}_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(a_t|\boldsymbol{s}_t)}\hat{A}(\boldsymbol{s}_t, a_t)\right] \\ & + \mathbb{E}_t\left[\beta \text{KL}\left[\pi_{\boldsymbol{\theta}_{\text{old}}}(\cdot|\boldsymbol{s}_t), \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}_t)\right]\right] . \end{aligned} \tag{7}$$

Here $\boldsymbol{\theta}$ is the current parameter of the policy network, $\boldsymbol{\theta}_{\text{old}}$ is the previous parameter before the update operation. $\hat{A}(\boldsymbol{s}_t, a_t)$ is the estimated advantage calculated as

$$\hat{A}(\boldsymbol{s}_t, a_t) = R_t(\boldsymbol{s}_t, a_t) + \gamma V_{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}) - V_{\boldsymbol{\theta}}(\boldsymbol{s}_t) . \tag{8}$$

It is a little different to that in [37] which utilizes generalized advantage estimator (GAE) for advantage estimation as our time horizon is relatively short and GAE does not bring better performance in our experiments. $V_{\boldsymbol{\theta}}(\cdot)$ is the value function approximated by the critic network, which is optimized through a value function loss

$$L_v(\boldsymbol{\theta}) = \mathbb{E}_t\left[\|V_{\boldsymbol{\theta}}(\boldsymbol{s}_t) - V_t\|_2\right] . \tag{9}$$

V_t is the empirical value of cumulative future rewards

$$V_t = \sum_{t'=t}^{T-1} \mathbb{E}\left[\gamma^{T-t'-1} R_{t'}(\boldsymbol{s}_{t'}, a_{t'})\right] . \tag{10}$$

The penalty term of KL-divergence in Eq. (7) controls the change within an update iteration, whose goal is to stabilize the parameter update. And the adaptive penalty parameter β is tuned according to the result of KL-divergence term following [37].

As a result, the overall objective function of the student includes the policy loss L_p, the value function loss L_v and the imitation loss L_d as

$$L(\boldsymbol{\theta}) = \overbrace{L_p + \lambda L_v}^{\text{policy optimization}} + \overbrace{\mu L_d}^{\text{imitation}} , \tag{11}$$

where λ and μ are the weights of value function loss and the imitation loss. The policy network is optimized to minimize the overall loss with the gradient descent method. Please also be noted that the overall loss function of the teacher $L(\boldsymbol{\phi})$ does not have the imitation loss.

3.3.3 Policy Network Structure. As is shown in Figure 4, there are several components in the policy network where the parameter $\boldsymbol{\theta}$ is shared by both actor and critic. Two recurrent neural networks (RNNs) have been used to conduct high-level representations from the public and private variables, separately. The reason to use RNN is to capture the temporal patterns for the sequential evolving states within an episode. Then the two obtained representations are combined and fed into an inference layer to calculate a comprehensive representation for the actor layer and and the critic layer. The actor layer will propose the final action distribution under the current state. For training, following the original work of PPO [37], the final action are sampled w.r.t. the produced action distribution $\pi(\cdot|\boldsymbol{s}_t)$. As for evaluation and testing, on the contrary, the policy migrates to fully exploitation thus the actions are directly taken by $a_t = \arg\max_a \pi(\cdot|\boldsymbol{s}_t)$ without exploration.

3.3.4 Discussion. Compared to previous related studies [24, 31], the main novelty of our method lies in that we incorporate a novel imitation paradigm to help common policy to approximate the optimal trading strategy more effectively. In addition, other than using a different policy network architecture, we also proposed a normalized reward function for universal trading for all instruments which is more efficient than training over single instrument separately. Moreover, we find that the general patterns learned from the other instrument data can improve trading performance and we put the experiment and discussions in 4.

4 EXPERIMENTS

In this section, we present the details of the experiments.

We first raise three research questions (RQs) to lead our discussion in this section. **RQ1**: Does our method succeed in finding a proper order execution strategy which applies universally on all instruments? **RQ2**: Does the imitation learning method help our agent to improve the overall trading performance? **RQ3**: What typical execution patterns does each compared method illustrate?

4.1 Experiment Settings

4.1.1 Compared methods. We compare our method and its variants with the following baseline order execution methods.

TWAP (Time-weighted Average Price) is a model-based strategy which equally splits the order into T pieces and evenly execute the same amount of shares at each timestep. It has been proven to be the optimal strategy under the assumption that the market price follows Brownian motion [2].

AC (Almgren-Chriss) is a model-based method [1], which analytically finds the efficient frontier of optimal execution. We only consider the temporary price impact for fair comparison.

VWAP (Volume-weighted Average Price) is another model-based strategy which distributes orders in proportion to the (empirically estimated) market transaction volume in order to keep the execution price closely tracking the market average price ground truth [3, 20].

DDQN (Double Deep Q-network) is a value-based RL method [31] and adopts state engineering optimizing for individual instruments.

PPO is a policy-based RL method [24] which utilizes PPO algorithm with a sparse reward to train an agent with a recurrent neural network for state feature extraction. The reward function and the network architecture are different to our method.

ILO is our proposed method described above, which has two other variants for ablation study: $\mathbf{ILO^S}$ is the pure student trained *without* imitation loss and $\mathbf{ILO^T}$ is the teacher trained with perfect market information.

Note that, for fair comparison, all the learning-based methods, i.e., DDQN, PPO and ILO, have been trained over all the instrument data, rather than that training over the data of individual instrument as shown in the related works [24, 31].

4.1.2 Datasets. All the compared methods are trained and evaluated with the historical transaction data of the stocks in the China A-shares market. The dataset contains (i) minute-level price-volume market information and (ii) the order amount of every trading day for each instrument from Jan. 1, 2017 to June 30, 2019. The market price information includes high, open, low, close and average prices of each minute. The volume information is the total number of the traded shares of that instrument on the market at each minute. Without loss of generality, we focus on intraday order execution, while our method can be easily adapted to a more high-frequency trading scenario. The order is generated following the method described in [34] and includes the order type (liquidation or acquirement) and the target number of the instrument shares that needs to be executed during the time horizon of one trading day of 240 minutes.

The dataset is divided into training, validation and test datasets according to the trading time. The detailed statistics are listed in Table 4. We keep only the instruments of CSI 800 in validation and test datasets due to the limit of our computing power. Note that as CSI 800 Index is designed to reflect the overall performance of A-shares [9], this is a fair setting for evaluating all the compared methods.

For all the model-based methods, the parameters are set based on the training dataset and the performances are directly evaluated on the test dataset. For all the learning-based RL methods, the policies are trained on the training dataset and the hyper-parameters are tuned on the validation dataset. The final performances of RL policiesd are calculated over the test dataset averaging the results from six policies trained with the same hyper-parameter but different random seeds.

	Training	Validation	Test
instruments	3,566	855	855
order	1,654,385	35,543	33,176
Time	201701 - 201902	201903 - 201904	201905 - 201906

Table 1: The dataset statistics.

4.2 Evaluation Workflow

Every trading strategy has been evaluated following the procedure in Figure 5. Given the trading strategy, we need to go through the evaluation data to assess the performance of order execution. After (1) the evaluation starts, if (2) there still exists unexecuted orders, (3) the next order record would be used for intraday order execution evaluation. Otherwise, (4) the evaluation is over. The execution of one order would end once the time horizon ends or the order has been fulfilled.

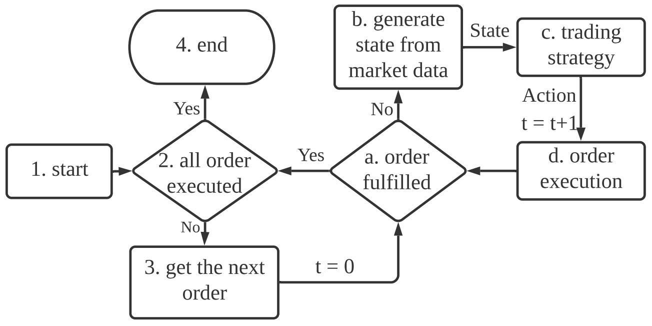

Figure 5: Evaluation workflow.

The detailed evaluation protocol is presented in Figure 6. The trading day is divided into T periods by the T timesteps, at the beginning of period t, i.e., at timestep t, the state $\mathbf{s}_t$ is generated by the environment and action a_t is proposed by the strategy. The proposed volume to trade $q_{t+1} = a_t Q$ is then distribute to every minute in the coming period following a predefined base strategy.

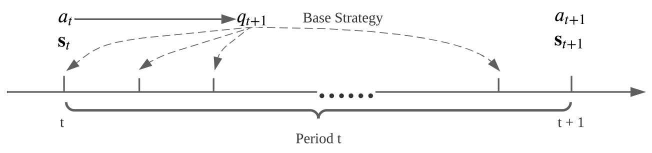

Figure 6: The detailed evaluation protocol of each timestep.

Specifically, in all of our experiments, we set $T = 8$ in our experiment and each period has a length of 30 minutes. Without loss of generality, following [31], we use TWAP to conduct the actual execution within each period, in other words, we equally allocate some volume on every minute in period t from q_{t+1}. Note that, one can also replace TWAP with any other base strategies.

Evaluation Metrics.

We use the obtained **reward** as the first metric for all the compared methods. Although some methods are model-based (i.e., TWAP, AC, VWAP) or utilize a different reward function (i.e., PPO), the reward of their trading trajectories would all be calculated as $\frac{1}{|\mathbb{D}|}\sum_{k=1}^{|\mathbb{D}|}\sum_{t=0}^{T-1} R_t^k(\mathbf{s}_t^k, a_t^k)$, where R_t^k is the reward of the k-th order at timestep t defined in Eq. (4) and $|\mathbb{D}|$ is the size of the dataset. The second is the *price advantage* (**PA**) $= \frac{10^4}{|\mathbb{D}|}\sum_{k=1}^{|\mathbb{D}|}\left(\frac{\bar{P}^k_{\text{strategy}}}{\tilde{p}^k} - 1\right)$, $\bar{P}^k_{\text{strategy}}$ is the corresponding AEP that our strategy has achieved on order k. This measures the relative gained revenue of a trading strategy compared to that from a baseline price $\tilde{p}^k$. Generally, $\tilde{p}^k$ is the averaged market price of the instrument on the specific trading day, which is constant for all the compared methods. PA is measured in basis points (BPs), where one basis point is equal to 0.01%. We conduct significance test [25] on PA and reward to verify the statistical significance of the performance improvement. We also report *gain-loss ratio* (**GLR**) $= -\frac{\mathbb{E}[\text{PA}|\text{PA}>0]}{\mathbb{E}[\text{PA}|\text{PA}<0]}$ result.

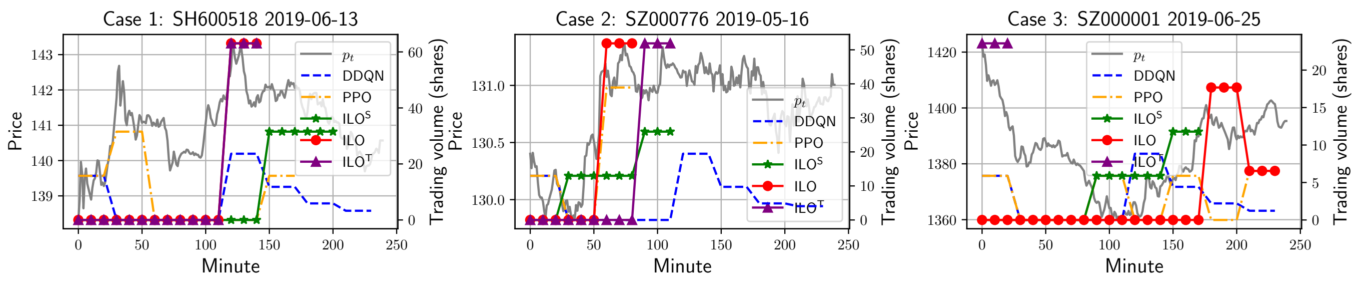

Figure 7: An illustration of execution details of different methods.

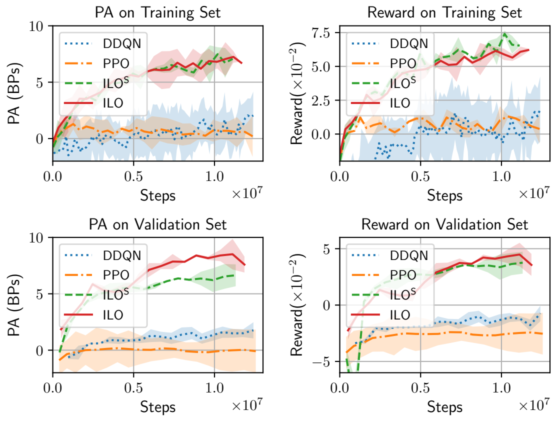

Figure 8: Learning curves (mean±std over 6 random seeds). Every step means one interaction with the environment.

4.3 Experiment Results

We analyze the experimental results from three aspects.

4.3.1 Overall Performance. The overall results are reported in Table 2 (* indicates p-value < 0.01 in significance test) and we have the following findings. (i) ILO has the best performance among all the compared methods including ILO^S without teacher guidance, which illustrates the effectiveness of imitation learning from oracle (**RQ2**). (ii) All the model-based methods perform worse than the learning-based methods as their adopted market assumptions might not be practical in the real world. Also, they fail to capture the microstructure of the market and can not adjust their strategy accordingly. As TWAP equally distribute the order to every timestep, the AEP of TWAP is always $\tilde{p}$. Thus the PA and GLR results of TWAP are all zero. (iii) Our proposed method outperforms other learning-based methods (**RQ1**). Speaking of DDQN, we may conclude that it is hard for the value-based method to learn a universal Q-function for all instruments. When comparing ILO^S with PPO, we conclude that the normalized instant reward function may largely contribute to the performance improvement.

4.3.2 Learning Analysis. We show learning curves over training and validation datasets in Figure 8. We have the following findings that (i) all these methods reach stable convergence after about 8 million steps of training. (ii) Among all the learning-based methods, our ILO method steadily achieve the highest PA and reward results on validation set after convergence, which shows the effectiveness of imitation learning. (iii) Though the performance gap between ILO^S and ILO is relatively small in the training dataset, ILO evidently outperforms ILO^S on validation dataset, which shows that oracle guidance can help to learn a more general and effective trading strategy.

Category	Strategy	Reward($\times 10^{-2}$)	PA	GLR
financial model-based	TWAP	-0.42	0	0
	AC	-1.45	2.33	0.89
	VWAP	-0.30	0.32	0.88
learning-based	DDQN	2.91	4.13	1.07
	PPO	1.32	2.52	0.62
	ILO^S	3.24	5.19	1.19
	ILO	**3.36***	**6.17***	**1.35**

Table 2: Performance comparison; the higher, the better.

4.3.3 Case Study. We further investigate the action patterns of different strategies (**RQ3**). In Figure 7, we present the execution details of all the learning-based methods. The colored lines exhibit the volume of shares traded by these agents at every minute and the grey line shows the trend of the market price p_t of the underlying trading asset on that day.

We can see that in all cases, the *teacher* ILO^T always captures the optimal trading opportunity. It is reasonable since ILO^T can fully observe the perfect information, including future trends. The execution situation of DDQN in all cases are exactly the same, which indicates that DDQN fails to adjust its execution strategy dynamically according to the market, thus not performing well. For PPO and ILO^S, either do they miss the best price for execution or waste their inventory at a bad market opportunity.

Our proposed ILO method outperforms all other methods in all three cases except the teacher ILO^T. Specifically, in Case 1, ILO captures the best opportunity and manages to find the optimal action sequence. Although ILO fails to capture the optimal trading time in Case 2, it still manages to find a relatively good opportunity and achieve a reasonable and profitable action sequence. However, as a result of observing no information at the beginning of the trading day, our method tends to miss some opportunities at the beginning of one day, as shown in Case 3. Though that, in such cases, ILO still manages to find a sub-optimal trading opportunity and illustrates some advantages to other methods.

4.3.4 Advantages of universal training. Recall that our proposed method ILOuses the data of all the instruments to train a universal policy for executing all instrument orders. However, the other learning-based methods (DDQN and PPO) were originally proposed

Algorithm	Policy	Reward $\times 10^{-2}$										
		Mean	SH600519	SH601318	SH600036	SH600276	SH601166	SH600030	SH600887	SH601328	SH600016	SH601288
DDQN	Single	-2.55	0.01	-3.94	0.32	-6.62	**-2.13**	-5.24	**-5.00**	**0.24**	-0.82	-2.33
PPO	Single	-20.62	-2.57	-12.28	-10.07	-29.19	-2.70	-3.62	-11.95	-48.70	-7.98	-77.12
ILO	Universal	0.34	7.79	**2.23**	1.39	-3.75	-2.28	3.28	**9.57**	-6.98	-4.45	-3.37
	Fine-tuned	**1.82**	**14.30**	-0.62	**9.46**	**4.46**	-7.24	**4.84**	2.55	-8.67	-1.13	**0.21**
Algorithm	**Policy**	**PA**										
		Mean	SH600519	SH601318	SH600036	SH600276	SH601166	SH600030	SH600887	SH601328	SH600016	SH601288
DDQN	Single	0.57	-2.82	1.31	3.61	-0.77	-0.36	-0.34	-2.19	5.41	1.60	0.28
PPO	Single	4.7916	12.12	**9.474**	-1.31	1.4	**1.70**	4.97	4.016	**7.213**	3.079	**5.257**
ILO	Universal	4.12	5.65	8.39	4.85	1.82	0.27	9.74	**12.66**	-1.79	-0.80	0.37
	Fine-tuned	**6.63**	**12.89**	6.42	**13.67**	**11.03**	-4.35	**11.95**	9.31	-2.66	**3.31**	4.78
Algorithm	**Policy**	**GLR**										
		Mean	SH600519	SH601318	SH600036	SH600276	SH601166	SH600030	SH600887	SH601328	SH600016	SH601288
DDQN	Single	1.02	0.66	0.84	1.07	0.90	0.93	0.91	0.53	**2.10**	1.10	1.04
PPO	Single	1.10	**1.39**	0.91	0.78	0.84	0.56	0.91	1.33	1.66	1.34	1.08
ILO	Universal	**1.29**	0.87	**1.19**	1.31	**1.79**	**1.07**	**1.49**	**2.12**	0.73	0.99	1.37
	Fine-tuned	1.11	1.18	0.89	**1.82**	0.93	0.76	0.88	1.18	0.64	**1.39**	**1.49**

Table 3: The test results over 10 selected instruments.

by learning over single instrument data and to derive an individual trading policy for each instrument. In this section, we compared the results of the two training paradigm.

For comparison, we choose 10 instruments with the highest market capitalization and liquidation in A-shares market in order to fully represent the situation of the whole market. For PPO and DDQN, we train and evaluate the policies with the data of each single instrument in the training and test set, respectively. For our ILO method, we first train a universal policy on all the training data of various instruments. Then for each instrument, the universal policy is fine-tuned with the corresponding instrument data in the training set. The fine-tuned policies and the universal policy are then evaluated over the test data of these selected 10 instruments.

The results are presented in Table 3. Recall that, our ILO method achieves the best overall performance when evaluated over all instruments. When executing the orders over single instrument, the universal policy of ILO achieves comparable results to that of PPO trained with only the specific single instrument data. After fine-tuning, ILO policies outperform all other baseline policies. Considering that there are thousands of instruments on market, it is costly to train different policies for each individual instrument, thus the universally trained ILO policy is preferable more efficient, from the view of either the overall performance or the single instrument experiment.

Moreover, we may easily notice that the Reward performance of PPO is quite bad. The reason is that, according to our experimental findings, most trained PPO policies tend to converge to a policy that does not trade any shares until the last timestep when trained on single instrument's data, which indicates that training over single instrument data may easily derive over-fitting. Though, on the selected instruments, such strategy can bring good PA results. They would suffer from large penalty of the market impact. Also, such policies can not generalize well on the data of other instruments, neither for the data within a different time range.

In summary, the universal trading policy trained on all the instrument data can achieve better results and generalization ability with lower cost.

4.4 Rolling window experiments

To make comprehensive comparison, we generate three datasets with a rolling window of eighteen-month length and six-month stride size. The detailed statistics of these datasets are listed in Table 4.

Rolling window	Phase	# order	Time period
1801-1908	Training	845,006	01/01/2018 - 31/12/2018
	Validation	132,098	01/01/2019 - 28/02/2019
	Test	455,332	01/03/2019 - 31/08/2019
1807-2002	Training	854,936	01/07/2018 - 30/06/2019
	Validation	163,140	01/07/2019 - 31/08/2019
	Test	428,846	01/09/2019 - 29/02/2020
1901-2008	Training	883,904	01/01/2019 - 31/12/2019
	Validation	132,678	01/01/2020 - 29/02/2020
	Test	465,014	01/03/2020 - 31/08/2020

Table 4: The dataset statistics.

The PA results of all compared methods on these datasets are shown in Table 5. We can tell from the results that our purposed ILO method steadily outperforms all compared methods on all datasets, indicating our method's efficiency for order execution.

Method	1801-1908	1807-2002	1901-2008
AC	2.06	1.13	1.24
VWAP	0.43	-0.79	-0.75
DDQN	4.26±0.15	2.48±0.24	3.03±0.43
PPO	1.29±0.36	-0.14±1.07	-0.76±0.21
ILO^S	6.34±0.20	5.28±0.16	3.93±0.57
ILO	**6.49±0.27**	**5.35±0.73**	**5.69±0.73**

Table 5: The PA results on three rolling window datasets.

5 CONCLUSION

In this paper, we presented a model-free reinforcement learning framework for optimal order execution. We incorporate a universal policy optimization method that learns and transfers knowledge data from various instruments for order execution over all the instruments. Also, we conducted an imitation learning from learning-based oracle teacher paradigm to improve the sample efficiency of RL methods and help derive a more reasonable trading strategy. It has been done through letting a student imitate the optimal behavior patterns from the optimal trading strategy derived by a teacher with perfect information. The experiment results over real-world instrument data have reflected the efficiency and effectiveness of the proposed learning algorithm.

In the future, we plan to incorporate imitation learning to learn a general trading strategy from the oracles conducted for each single asset.

REFERENCES

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

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

[3] 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.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[5] Alvaro Cartea and Sebastian Jaimungal. 2015. Optimal execution with limit and market orders. *Quantitative Finance* 15, 8 (2015), 1279–1291.

[6] Alvaro Cartea and Sebastian Jaimungal. 2016. Incorporating order-flow into optimal execution. *Mathematics and Financial Economics* 10, 3 (2016), 339–364.

[7] Álvaro Cartea, Sebastian Jaimungal, and José Penalva. 2015. *Algorithmic and high-frequency trading.* Cambridge University Press.

[8] Philippe Casgrain and Sebastian Jaimungal. 2019. Trading algorithms with learning in latent alpha models. *Mathematical Finance* 29, 3 (2019), 735–772.

[9] China Securities Index Co. 2020. CSI800 index. https://bit.ly/3btW6di. [Online; accessed: 2020-10-15].

[10] Kevin Dabérius, Elvin Granat, and Patrik Karlsson. 2019. Deep Execution-Value and Policy Based Reinforcement Learning for Trading and Beating Market Benchmarks. *Available at SSRN 3374766* (2019).

[11] Maor Gaon and Ronen Brafman. 2020. Reinforcement Learning with Non-Markovian Rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3980–3987.

[12] Sam Green, Craig M Vineyard, and Cetin Kaya Koç. 2019. Distillation Strategies for Proximal Policy Optimization. *arXiv preprint arXiv:1901.08128* (2019).

[13] Olivier Guéant. 2016. *The Financial Mathematics of Market Liquidity: From optimal execution to market making.* Vol. 33. CRC Press.

[14] Olivier Guéant and Charles-Albert Lehalle. 2015. General intensity shapes in optimal liquidation. *Mathematical Finance* 25, 3 (2015), 457–495.

[15] 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.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[17] Robert Hu. 2016. *Optimal Order Execution using Stochastic Control and Reinforcement Learning.* Master's thesis. KTH Royal Institute of Technology.

[18] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 1–35.

[19] Zhengyao Jiang and Jinjun Liang. 2017. Cryptocurrency portfolio management with deep reinforcement learning. In *2017 Intelligent Systems Conference (IntelliSys)*. IEEE, 905–913.

[20] 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.

[21] Kwei-Herng Lai, Daochen Zha, Yuening Li, and Xia Hu. 2020. Dual Policy Distillation. *arXiv preprint arXiv:2006.04061* (2020).

[22] Junjie Li, Sotetsu Koyamada, Qiwei Ye, Guoqing Liu, Chao Wang, Ruihan Yang, Li Zhao, Tao Qin, Tie-Yan Liu, and Hsiao-Wuen Hon. 2020. Suphx: Mastering Mahjong with Deep Reinforcement Learning. *arXiv preprint arXiv:2003.13590* (2020).

[23] Xihan Li, Jia Zhang, Jiang Bian, Yunhai Tong, and Tie-Yan Liu. 2019. A cooperative multi-agent reinforcement learning framework for resource balancing in complex logistics network. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 980–988.

[24] Siyu Lin and Peter A Beling. 2020. An End-to-End Optimal Trade Execution Framework based on Proximal Policy Optimization. In *IJCAI*.

[25] Simon J Mason and Nicholas E Graham. 2002. Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation. *Quarterly Journal of the Royal Meteorological Society* (2002).

[26] 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.

[27] John Moody and Matthew Saffell. 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks* 12, 4 (2001), 875–889.

[28] John Moody and Lizhong Wu. 1997. Optimization of trading systems and portfolios. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*. IEEE, 300–307.

[29] John E Moody, Matthew Saffell, Y Liao, and L Wu. 1998. Reinforcement Learning for Trading Systems and Portfolios.. In *KDD*. 279–283.

[30] 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.

[31] Brian Ning, Franco Ho Ting Ling, and Sebastian Jaimungal. 2018. Double Deep Q-Learning for Optimal Execution. *arXiv preprint arXiv:1812.06600* (2018).

[32] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2015. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342* (2015).

[33] Dean A Pomerleau. 1989. *Alvinn: An autonomous land vehicle in a neural network*. Technical Report. CARNEGIE-MELLON UNIV PITTSBURGH PA ARTIFICIAL INTELLIGENCE AND PSYCHOLOGY ....

[34] Edward E Qian, Ronald H Hua, and Eric H Sorensen. 2007. *Quantitative equity portfolio management: modern techniques and applications.* CRC Press.

[35] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 627–635.

[36] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. 2015. Policy distillation. *arXiv preprint arXiv:1511.06295* (2015).

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[38] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

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

[40] Xiaocheng Tang, Zhiwei Qin, Fan Zhang, Zhaodong Wang, Zhe Xu, Yintai Ma, Hongtu Zhu, and Jieping Ye. 2019. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1780–1790.

[41] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. 2017. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*. 4496–4506.

[42] Hanwei Wu, Ather Gattami, and Markus Flierl. 2020. Conditional Mutual information-based Contrastive Loss for Financial Time Series Forecasting. *arXiv preprint arXiv:2002.07638* (2020).

[43] Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Jun Xiao, Bo Li, et al. 2020. Reinforcement-learning based portfolio management with augmented asset movement prediction states. *arXiv preprint arXiv:2002.05780* (2020).

[44] Haiyan Yin and Sinno Jialin Pan. 2017. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Thirty-First AAAI Conference on Artificial Intelligence*.