

Universal Trading for Order Execution with Oracle Policy Distillation

Yuchen Fang,^{1*} Kan Ren,² Weiqing Liu,² Dong Zhou,²
Weinan Zhang,¹ Jiang Bian,² Yong Yu,¹ Tie-Yan Liu²

¹Shanghai Jiao Tong University, ²Microsoft Research
{arthur.fyc, wnzhang}@sjtu.edu.cn, yyu@apex.sjtu.edu.cn
{kan.ren, weiqing.liu, zhou.dong, jiang.bian, tyliu}@microsoft.com

Abstract

As a fundamental problem in algorithmic trading, order execution aims at fulfilling a specific trading order, either liquidation or acquirement, for a given instrument. Towards 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, the noisy and yet imperfect market information that can be leveraged by the policy has made it quite challenging to build up sample efficient reinforcement learning methods to achieve effective order execution. 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, this framework leverages a policy distillation method that can better guide the learning of the common policy towards practically optimal execution by an oracle teacher 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.

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).

As discussed in (Cartea, Jaimungal, and Penalva 2015), the main challenge of order execution lies in a trade-off between avoiding harmful “*market impact*” caused by large

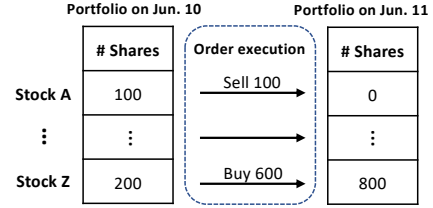


Figure 1: An example of portfolio adjustment which requires order execution.

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 (Bertsimas and Lo 1998; Almgren and Chriss 2001; Cartea and Jaimungal 2015, 2016; Cartea, Jaimungal, and Penalva 2015). Nonetheless, these model-based¹ 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 (Nevmyvaka, Feng, and Kearns 2006; Ning, Ling, and Jaimungal 2018; Lin and Beling 2020) have 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 largely noisy and imperfect information. For one thing, the noisy data (Wu, Gattami, and Flierl 2020) could lead to a quite low sample efficiency of RL methods and thus results in poor effectiveness of the learned order execution policy.

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

*The work was conducted when Yuchen Fang was doing internship at Microsoft Research Asia.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

¹Here ‘model’ corresponds to some market price assumptions, to distinguish the environment model in reinforcement learning.

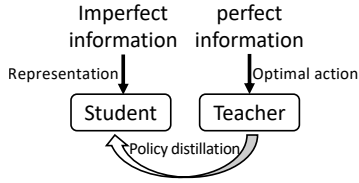


Figure 2: Brief illustration of the oracle policy distillation.

future trends of the market price² 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 a novel policy distillation approach in order for bridging the gap between the noisy yet imperfect market information and the optimal order execution policy. More concretely, as shown in Figure 2, this approach is essentially a teacher-student learning paradigm, in which 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 teacher-student learning paradigm for policy distillation may beneficially motivate the community, to alleviate the problem between imperfect information and the optimal decision making.
- 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.

Related Works

Order Execution

Optimal order execution is originally proposed in (Bertsimas and Lo 1998) and the main stream of the solutions are model-based. (Bertsimas and Lo 1998) 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 (Almgren and Chriss 2001), the Almgren-Chriss model has been proposed which extended the above solution and incorporated both

²We define ‘market price’ as the averaged transaction price of the whole market at one time which has been widely used in literature (Nevmyvaka, Feng, and Kearns 2006; Ning, Ling, and Jaimungal 2018).

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 (Cartea and Jaimungal 2016; Guéant and Lehalle 2015; Guéant 2016; Casgrain and Jaimungal 2019) 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 (Ning, Ling, and Jaimungal 2018).

Reinforcement Learning

RL attempts to optimize an accumulative numerical reward signal by directly interacting with the environment (Sutton and Barto 2018) under few model assumptions such as Markov Decision Process (MDP). RL methods have already shown superhuman abilities in many applications, such as game playing (Mnih et al. 2015; Li et al. 2020; Silver et al. 2016), resource dispatching (Li et al. 2019; Tang et al. 2019), financial trading (Moody and Wu 1997; Moody et al. 1998; Moody and Saffell 2001) and portfolio management (Jiang and Liang 2017; Ye et al. 2020).

Besides the analytical methods mentioned above, RL gives another view for sequential decision optimization in order execution problem. Some RL solutions (Hendricks and Wilcox 2014; Hu 2016; Dabérius, Granat, and Karlsson 2019) solely extend the model-based assumptions mentioned above to either evaluate how RL algorithm might improve over the analytical solutions (Dabérius, Granat, and Karlsson 2019; Hendricks and Wilcox 2014), or test whether the MDP is still viable under the imposed market assumptions (Hu 2016). 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 financial market assumptions and utilize model-free RL to optimize execution strategy (Nevmyvaka, Feng, and Kearns 2006; Ning, Ling, and Jaimungal 2018; Lin and Beling 2020). 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.

Policy Distillation

This work is also related to policy distillation (Rusu et al. 2015), which incorporates knowledge distillation (Hinton, Vinyals, and Dean 2015) in RL policy training and has attracted many researches studying this problem (Parisotto, Ba, and Salakhutdinov 2015; Teh et al. 2017; Yin and Pan 2017; Green, Vineyard, and Koç 2019; Lai et al. 2020).

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 through policy distillation, which has not been properly studied before.

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 policy distillation 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.

Formulation of Order Execution

Generally, order execution is formulated under the scenario of trading within a predefined time horizon, e.g., one hour or a day. We follow a widely applied simplification to the reality (Cartea, Jaimungal, and Penalva 2015; Ning, Ling, and Jaimungal 2018) by which trading is based on discrete time space.

Under this configuration, we assume there are T timesteps $\{0, 1, \dots, T-1\}$, each of which is associated with a respective price for trading $\{p_0, p_1, \dots, p_{T-1}\}$. At each timestep $t \in \{0, 1, \dots, T-1\}$, 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} , i.e., the market price at the timestep $(t+1)$. 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

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

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. Note also that, for acquirement, the objective is reversed and the trader is to minimize the execution price to reduce the acquirement cost.

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 control the whole trading process through proposing an action $a \in \mathcal{A}$ given the observed state $s \in \mathcal{S}$, while maximizing the total expected rewards based on the reward function $R(s, a)$.

State. The observed state 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 (Nevmyvaka, Feng, and Kearns 2006; Lin and Beling 2020) 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 open, high, low, close, average price and transaction volume of each timestep.

Action. Given the observed state s_t , the agent proposes the decision $a_t = \pi(s_t)$ 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 (Ning, Ling, and Jaimungal 2018; Lin and Beling 2020) 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 $a_{T-1} = \max\{1 - \sum_{i=0}^{T-2} a_i, \pi(s_{T-1})\}$ 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*:

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

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 target order volume, to optimize a universal trading strategy for various 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. 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 (Gaon and Brafman 2020), however this is not the main focus of this paper and

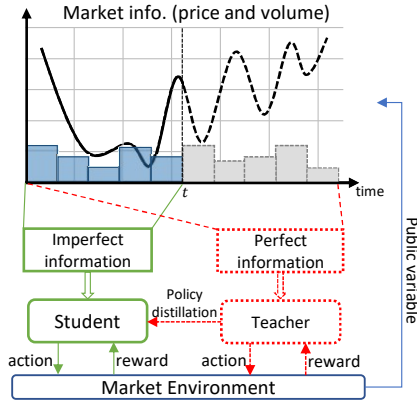


Figure 3: The framework of oracle policy distillation. All the modules with dotted lines, i.e. teacher and policy distillation procedure, are only used during the *training* phase, and would be removed during the *test* phase or in practice.

would not influence the conclusion.

To account for the potential that the trading activity may have some impacts onto the current market status, following (Ning, Ling, and Jaimungal 2018; Lin and Beling 2020), we also incorporate a quadratic penalty

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

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

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

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

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

As for the numerical details of the market variable and the corresponding MDP settings, please refer to the supplementary including the released code.

Assumptions Note that there are two main assumptions adopted in this paper. Similar to (Lin and Beling 2020), (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 these expense is relatively small fractions for the institutional investors that we are mainly aimed at.

Policy Distillation and Optimization

In this section, we explain the underlying intuition behind our policy distillation and illustrate the detailed optimization framework.

Policy Distillation 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 policy distillation.

We first explain the imperfect and perfect information. In addition to the private variable including left time and the left 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 as a common agent 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 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{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 $\tilde{\pi}_\phi(\cdot|\tilde{s}_t)$ through interacting with the environment given the perfect information \tilde{s}_t , where ϕ is the parameter of the teacher policy.
- **Student** itself learns by interacting with the environment to optimize a common policy $\pi_\theta(\cdot|s_t)$ with the parameter θ given the imperfect information s_t .

To build a connection between the imperfect information to the optimal trading strategy, we implement a policy distillation 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

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

where a_t and \tilde{a}_t are the actions taken from the policy $\pi_\theta(\cdot|s_t)$ of student and $\pi_\phi(\cdot|\tilde{s}_t)$ 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 as analyzed in our supplementary. 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.

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.

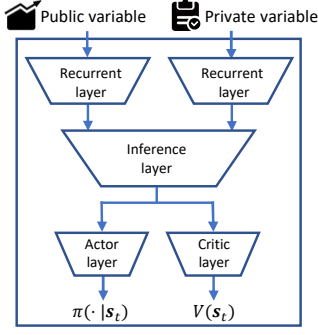


Figure 4: The overall structure of the policy network.

With the MDP formulation described above, we utilize Proximal Policy Optimization (PPO) algorithm (Schulman et al. 2017) in actor-critic style 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

$$L_p(\theta) = -\mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}(s_t, a_t) - \beta \text{KL} [\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right]. \quad (7)$$

Here θ is the current parameter of the policy network, θ_{old} is the previous parameter before the update operation. $\hat{A}(s_t, a_t)$ is the estimated advantage calculated as

$$\hat{A}(s_t, a_t) = R_t(s_t, a_t) + \gamma V_\theta(s_{t+1}) - V_\theta(s_t). \quad (8)$$

It is a little different to that in (Schulman et al. 2017) 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. $V_\theta(\cdot)$ is the value function approximated by the critic network, which is optimized through a value function loss

$$L_v(\theta) = \mathbb{E}_t [\|V_\theta(s_t) - V_t\|_2]. \quad (9)$$

V_t is the empirical value of cumulative future rewards calculated as below

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

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 (Schulman et al. 2017).

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

$$L(\theta) = \underbrace{L_p + \lambda L_v}_{\text{policy optimization}} + \underbrace{\mu L_d}_{\text{policy distillation}}, \quad (11)$$

where λ and μ are the weights of value function loss and the distillation 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 $L(\phi)$ does not have the policy distillation loss.

Policy Network Structure As is shown in Figure 4, there are several components in the policy network where the parameter θ 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 the critic layer. The actor layer will propose the final action distribution under the current state. For training, following the original work of PPO (Schulman et al. 2017), the final action are sampled w.r.t. the produced action distribution $\pi(\cdot|s_t)$. As for evaluation and testing, on the contrary, the policy migrates to fully exploitation thus the actions are directly taken by $a_t = \arg \max_a \pi(\cdot|s_t)$ without exploration. The detailed implementations are described in supplemental materials.

Discussion Compared to previous related studies (Ning, Ling, and Jaimungal 2018; Lin and Beling 2020), the main novelty of our method lies in that we incorporate a novel policy distillation 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 the supplementary.

Experiments

In this section, we present the details of the experiments. The open-source code and the supplementary are presented in this link³.

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 oracle policy distillation method help our agent to improve the overall trading performance? **RQ3**: What typical execution patterns does each compared method illustrate?

Experiment settings

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 (Bertsimas and Lo 1998).

AC (Almgren-Chriss) is a model-based method (Almgren and Chriss 2001), which analytically finds the efficient frontier of optimal execution. We only consider the temporary price impact for fair comparison.

³<https://seqml.github.io/opd/>

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 (Kakade et al. 2004; Białkowski, Darolles, and Le Fol 2008).

DDQN (Double Deep Q-network) is a value-based RL method (Ning, Ling, and Jaimungal 2018) and adopts state engineering optimizing for individual instruments.

PPO is a policy-based RL method (Lin and Beling 2020) 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.

OPD is our proposed method described above, which has two other variants for ablation study: **OPD^S** is the pure student trained *without* oracle guidance and **OPD^T** is the teacher trained with perfect market information.

Note that, for fair comparison, all the learning-based methods, i.e., DDQN, PPO and OPD, have been trained over all the instrument data, rather than that training over the data of individual instrument as shown in the related works (Ning, Ling, and Jaimungal 2018; Lin and Beling 2020).

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. 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 detailed description of the market information including data processing can be referred to the supplementary.

The dataset is divided into training, validation and test datasets according to the trading time. The detailed statistics are listed in Table 1. 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 (Co. 2020), 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. All these setting values are listed in the supplementary. The final performances of RL policies we present below are calculated over the test dataset averaging the results from six policies trained with the same hyper-parameter and six different random seeds.

| | Training | Validation | Test |
|---------------|----------------------|----------------------|----------------------|
| # instruments | 3,566 | 855 | 855 |
| # order | 1,654,385 | 35,543 | 33,176 |
| Time period | 1/1/2017 - 28/2/2019 | 1/3/2019 - 31/4/2019 | 1/5/2019 - 30/6/2019 |

Table 1: The dataset statistics.

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 flow is over. For each order record, the execution of the order would end once the time horizon ends or the order has been fulfilled. After all the orders in the dataset have been executed, the overall performance will be reported. More details are shown in the supplementary.

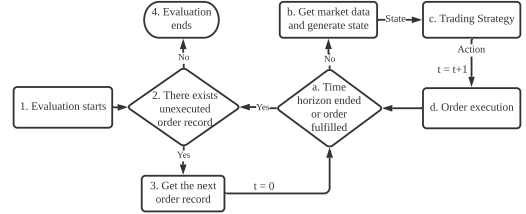


Figure 5: Evaluation workflow.

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 $\frac{1}{|\mathbb{D}|} \sum_{k=1}^{|\mathbb{D}|} \sum_{t=0}^{T-1} R_t^k(s_t^k, a_t^k)$, 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)** $= \frac{10^4}{|\mathbb{D}|} \sum_{k=1}^{|\mathbb{D}|} (\frac{\bar{P}_{\text{strategy}}^k}{\bar{p}^k} - 1)$, $\bar{P}_{\text{strategy}}^k$ 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 \bar{p}^k . Generally, \bar{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 (Mason and Graham 2002) 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.

Experiment Results

We analyze the experimental results from three aspects.

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) OPD has the best performance among all the compared methods including OPD^S without teacher guidance, which illustrates the effectiveness of oracle policy distillation (**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 \bar{p} . Thus

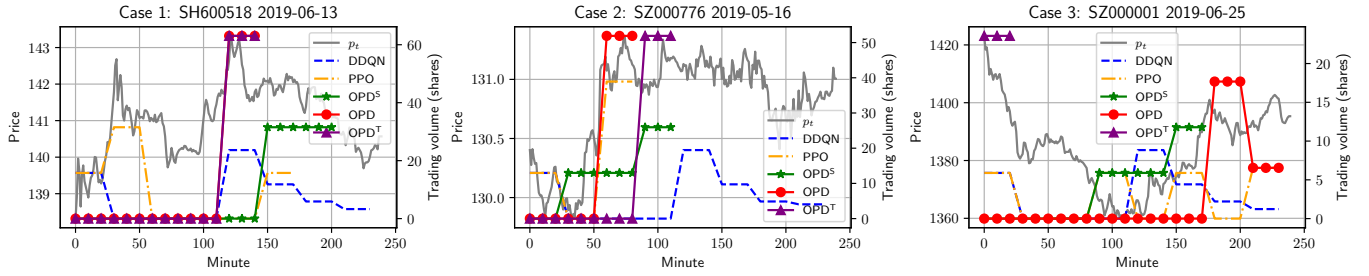


Figure 6: An illustration of execution details of different methods.



Figure 7: Learning curves (mean \pm std over 6 random seeds). Every step means one interaction with the environment.

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 OPD^S with PPO, we conclude that the normalized instant reward function may largely contribute to the performance improvement.

We also conduct our experiments on three datasets for a comprehensive comparison with a rolling window along time. The results are shown in the supplementary.

| Category | Strategy | Reward ($\times 10^{-2}$) | PA | GLR |
|-----------------------|------------------------------|-----------------------------|--------------|-------------|
| financial model-based | TWAP (Bertsimas et al. 1998) | -0.42 | 0 | 0 |
| | AC (Almgren et al. 2001) | -1.45 | 2.33 | 0.89 |
| | VWAP (Kakade et al. 2004) | -0.30 | 0.32 | 0.88 |
| | DDQN (Ning et al. 2018) | 2.91 | 4.13 | 1.07 |
| learning-based | PPO (Lin et al. 2020) | 1.32 | 2.52 | 0.62 |
| | OPD^S (pure student) | 3.24 | 5.19 | 1.19 |
| | OPD (our proposed) | 3.36* | 6.17* | 1.35 |

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

Learning analysis We show learning curves over training and validation datasets in Figure 7. 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 OPD method steadily achieve the highest PA and reward results on validation set after convergence, which shows the effectiveness of oracle policy distillation. (iii) Though the performance gap between OPD^S and OPD is relatively small in the training dataset, OPD evidently outperforms OPD^S on validation dataset,

which shows that oracle guidance can help to learn a more general and effective trading strategy.

Case study We further investigate the action patterns of different strategies (RQ3). In Figure 6, 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* OPD^T always captures the optimal trading opportunity. It is reasonable since OPD^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 OPD^S , either do they miss the best price for execution or waste their inventory at a bad market opportunity.

Our proposed OPD method outperforms all other methods in all three cases except the teacher OPD^T . Specifically, in Case 1, OPD captures the best opportunity and manages to find the optimal action sequence. Although OPD 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, OPD still manages to find a sub-optimal trading opportunity and illustrates some advantages to other methods.

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 in the data from various instruments for order execution over all the instruments. Also, we conducted an oracle policy distillation 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 and distillate 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 policy distillation to learn a general trading strategy from the oracles conducted for each single asset.

Acknowledgments

The corresponding authors are Kan Ren and Weinan Zhang. Weinan Zhang is supported by MSRA Joint Research Grant.

References

- Almgren, R.; and Chriss, N. 2001. Optimal execution of portfolio transactions. *Journal of Risk* 3: 5–40.
- Bertsimas, D.; and Lo, A. W. 1998. Optimal control of execution costs. *Journal of Financial Markets* 1(1): 1–50.
- Białkowski, J.; Darolles, S.; and Le Fol, G. 2008. Improving VWAP strategies: A dynamic volume approach. *Journal of Banking & Finance* 32(9): 1709–1722.
- Cartea, A.; and Jaimungal, S. 2015. Optimal execution with limit and market orders. *Quantitative Finance* 15(8): 1279–1291.
- Cartea, A.; and Jaimungal, S. 2016. Incorporating order-flow into optimal execution. *Mathematics and Financial Economics* 10(3): 339–364.
- Cartea, Á.; Jaimungal, S.; and Penalva, J. 2015. *Algorithmic and high-frequency trading*. Cambridge University Press.
- Casgrain, P.; and Jaimungal, S. 2019. Trading algorithms with learning in latent alpha models. *Mathematical Finance* 29(3): 735–772.
- Co., C. S. I. 2020. *CSI800 index*. <https://bit.ly/3btW6di>.
- Dabérius, K.; Granat, E.; and Karlsson, P. 2019. Deep Execution-Value and Policy Based Reinforcement Learning for Trading and Beating Market Benchmarks. *Available at SSRN 3374766*.
- Gaon, M.; and Brafman, R. 2020. Reinforcement Learning with Non-Markovian Rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3980–3987.
- Green, S.; Vineyard, C. M.; and Koç, C. K. 2019. Distillation Strategies for Proximal Policy Optimization. *arXiv preprint arXiv:1901.08128*.
- Guéant, O. 2016. *The Financial Mathematics of Market Liquidity: From optimal execution to market making*, volume 33. CRC Press.
- Guéant, O.; and Lehalle, C.-A. 2015. General intensity shapes in optimal liquidation. *Mathematical Finance* 25(3): 457–495.
- Hendricks, D.; and Wilcox, D. 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)*, 457–464. IEEE.
- Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Hu, R. 2016. Optimal Order Execution using Stochastic Control and Reinforcement Learning.
- Jiang, Z.; and Liang, J. 2017. Cryptocurrency portfolio management with deep reinforcement learning. In *2017 Intelligent Systems Conference (IntelliSys)*, 905–913. IEEE.
- Kakade, S. M.; Kearns, M.; Mansour, Y.; and Ortiz, L. E. 2004. Competitive algorithms for VWAP and limit order trading. In *Proceedings of the 5th ACM conference on Electronic commerce*, 189–198.
- Lai, K.-H.; Zha, D.; Li, Y.; and Hu, X. 2020. Dual Policy Distillation. *arXiv preprint arXiv:2006.04061*.
- Li, J.; Koyamada, S.; Ye, Q.; Liu, G.; Wang, C.; Yang, R.; Zhao, L.; Qin, T.; Liu, T.-Y.; and Hon, H.-W. 2020. Suphx: Mastering Mahjong with Deep Reinforcement Learning. *arXiv preprint arXiv:2003.13590*.
- Li, X.; Zhang, J.; Bian, J.; Tong, Y.; and Liu, T.-Y. 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*, 980–988. International Foundation for Autonomous Agents and Multiagent Systems.
- Lin, S.; and Beling, P. A. 2020. An End-to-End Optimal Trade Execution Framework based on Proximal Policy Optimization. In *IJCAI*.
- Mason, S. J.; and Graham, N. E. 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*.
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.
- Moody, J.; and Saffell, M. 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks* 12(4): 875–889.
- Moody, J.; and Wu, L. 1997. Optimization of trading systems and portfolios. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, 300–307. IEEE.
- Moody, J. E.; Saffell, M.; Liao, Y.; and Wu, L. 1998. Reinforcement Learning for Trading Systems and Portfolios. In *KDD*, 279–283.
- Nevmyvaka, Y.; Feng, Y.; and Kearns, M. 2006. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, 673–680.
- Ning, B.; Ling, F. H. T.; and Jaimungal, S. 2018. Double Deep Q-Learning for Optimal Execution. *arXiv preprint arXiv:1812.06600*.
- Parisotto, E.; Ba, J. L.; and Salakhutdinov, R. 2015. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*.

Rusu, A. A.; Colmenarejo, S. G.; Gulcehre, C.; Desjardins, G.; Kirkpatrick, J.; Pascanu, R.; Mnih, V.; Kavukcuoglu, K.; and Hadsell, R. 2015. Policy distillation. *arXiv preprint arXiv:1511.06295*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529(7587): 484.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tang, X.; Qin, Z.; Zhang, F.; Wang, Z.; Xu, Z.; Ma, Y.; Zhu, H.; and Ye, J. 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.

Teh, Y.; Bapst, V.; Czarnecki, W. M.; Quan, J.; Kirkpatrick, J.; Hadsell, R.; Heess, N.; and Pascanu, R. 2017. Distal: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, 4496–4506.

Wu, H.; Gattami, A.; and Flierl, M. 2020. Conditional Mutual information-based Contrastive Loss for Financial Time Series Forecasting. *arXiv preprint arXiv:2002.07638*.

Ye, Y.; Pei, H.; Wang, B.; Chen, P.-Y.; Zhu, Y.; Xiao, J.; Li, B.; et al. 2020. Reinforcement-learning based portfolio management with augmented asset movement prediction states. *arXiv preprint arXiv:2002.05780*.

Yin, H.; and Pan, S. J. 2017. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Thirty-First AAAI Conference on Artificial Intelligence*.