Universal Trading for Order Execution with Oracle Policy Distillation Supplemental Materials

Yuchen Fang, [1*] **Kan Ren,** [2] **Weiqing Liu,** [2] **Dong Zhou,** [2]
Weinan Zhang, [1] **Jiang Bian,** [2] **Yong Yu,** [1] **Tie-Yan Liu**[2]
[1]Shanghai Jiao Tong University
[2]Microsoft Research
{arthur_fyc, wnzhang}@sjtu.edu.cn, yyu@apex.sjtu.edu.cn
{kan.ren, weiqing.liu, zhou.dong, jiang.bian, tyliu}@microsoft.com

Implementation Details

In this section, we introduce the details of our implementation of the compared methods, including the learning algorithm for our method, the detailed network architecture and the hyper-parameter settings.

Learning Algorithm

We present the detailed learning process of our proposed method. As is discussed in our paper, the policy distillation has teacher-student paradigm thus the learning procedure is two-phased As shown in Algorithm 1, in the first phase, teacher is trained to convergence. Then the optimal policy π_{ϕ} is used to guide the student in the second phase following the method we proposed in the Methodology section.

Algorithm 1: Teacher-student learning algorithm

```
Random initialize $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ as the parameters for
  teacher and student respectively.;
$\boldsymbol{\phi}_{\text{old}} \longleftarrow \boldsymbol{\phi}$;
repeat
    Run policy $\pi_{\boldsymbol{\phi}_{\text{old}}}$ in environment for $M$
      episodes;
    Compute Advantages for each collected
      timestep.;
    Optimize loss $L_p + \lambda L_v$ w.r.t. $\boldsymbol{\phi}$, with $K$
      epochs and batch size $B$;
    $\boldsymbol{\phi}_{\text{old}} \longleftarrow \boldsymbol{\phi}$;
until Teacher is converged;
$\boldsymbol{\theta}_{\text{old}} \longleftarrow \boldsymbol{\theta}$;
repeat
    Run policy $\pi_{\boldsymbol{\theta}_{\text{old}}}$ in environment for $M$
      episodes;
    Compute Advantages and $L_d$ for each
      collected timestep.;
    Optimize loss $L = L_p + \lambda L_v + \mu L_d$ w.r.t. $\boldsymbol{\theta}$,
      with $K$ epochs and batch size $B$;
    $\boldsymbol{\theta}_{\text{old}} \longleftarrow \boldsymbol{\theta}$;
until Student is converged;
```

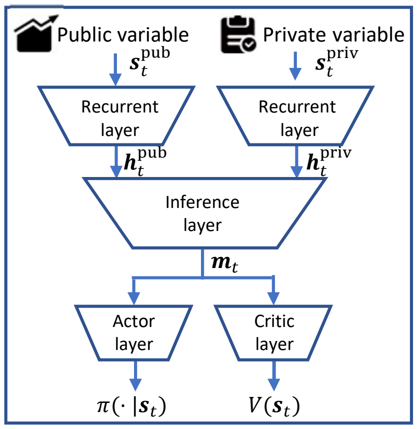

Figure 1: The architecture of our policy network.

Network Architecture

In this section, we describe the policy network's architecture of our proposed OPD method in detail. The overall architecture is shown in Figure 1. Recall that, at every timestep the input of policy network can be divided into two parts, the public variables and the private variables, denoted as $\mathbf{s}_t^{\text{pub}}$ and $\mathbf{s}_t^{\text{priv}}$ respectively. We utilize two recurrent neural networks to extract high level representation from public and private variables as bellow

$$\mathbf{h}_t^{\text{pub}} = f^{\text{pub}}(\mathbf{s}_t^{\text{pub}}) \,, \tag{1}$$

$$\mathbf{h}_t^{\text{priv}} = f^{\text{priv}}(\mathbf{s}_t^{\text{priv}}) \,. \tag{2}$$

Specifically, we implement Gated Recurrent Unit (GRU) (Cho et al. 2014) as the recurrent unit $f(\mathbf{s}_t|\mathbf{h}_{t-1})$, calculated as

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \cdot [\mathbf{h}_{t-1}, \mathbf{s}_t]) \tag{3}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \cdot [\mathbf{h}_{t-1}, \mathbf{s}_t]) \tag{4}$$

$$\tilde{\mathbf{h}}_t = tanh(\mathbf{W} \cdot [\mathbf{r}_t \odot \mathbf{h}_{t-1}, \mathbf{s}_t]) \tag{5}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{6}$$

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

After getting the representations $\mathbf{h}_t^{\text{pub}}$ and $\mathbf{h}_t^{\text{priv}}$, we concatenate them as the input of a three-layer multi-layer perception (MLP) with Rectifier (ReLU) activate function $\text{ReLU}(x) = \max(0, x)$ to get a comprehensive representation of the state as

$$\mathbf{m}_t = g([\mathbf{h}_t^{\text{pub}}, \mathbf{h}_t^{\text{priv}}]) \ . \tag{7}$$

Finally, the representation $\mathbf{m}$ is fed into two fully-connected layers to generate the output of critic $V(\mathbf{s}_t)$ and actor $\pi(\cdot|\mathbf{s}_t)$.

$$V(\mathbf{s}_t) = p_v(\mathbf{m}_t) \ , \tag{8}$$

$$\pi(\mathbf{s}_t) = p_\pi(\mathbf{m}_t) \ . \tag{9}$$

Hyper-parameters

In this section, we present all the hyper-parameters we use for result reproduction. There are two sets of hyper-parameters, the environment settings and the algorithm settings.

Environment settings All parameters for the environment and our MDP are listed below as the environment settings.

Penalty coefficient α controls the degree of the market impact from the trading activities have on market price. We set $\alpha = 100$ for all the experiments.

Discount factor γ is set to 1 as the overall profit over the whole trading horizon is the optimization goal as defined in Eq. (1) in the main paper.

Algorithm settings All the compared algorithms can be divided into financial model-based algorithms and learning-based algorithms. And for model-based algorithms, both TWAP and VWAP are non-parametric methods. As a result, we only present the parameter setting of AC. AC (Almgren and Chriss 2001) provides a unique optimal execution strategy for each value of risk aversion as

$$q_{t+1} = \frac{2sinh(\frac{1}{2}k)}{sinh(kT)} cosh(k(T - t + \frac{1}{2}))Q \tag{10}$$

where

$$k = cosh^{-1}(\frac{1}{2}\tilde{k}^2 + 1) \tag{11}$$

and

$$\tilde{k}^2 = \frac{\lambda\sigma^2}{\eta} \tag{12}$$

Here $\epsilon = 0.0625$ is the bid-ask spread. $\eta = 2.5 \times 10^{-6}$ is the temporal impact coefficient. σ is the volatility of the instrument which we estimate by taking the average volatility of last 30 days.

The hyper-parameters used by all learning-based methods are presented in Table 1, together with the searching ranges within which we fine-tuned the parameters. All learning-based methods are implemented based on Tianshou framework (Weng et al. 2020). Please refer to its main page[1] for more detailed explanations of the parameters.

[1] https://github.com/thu-ml/tianshou

Algorithm	Parameters	Search Range
General	learning rate: 1×10^{-4}	—-
	B: 512	{512, 1024}
	activation functions: ReLU	—-
	GRU cell size: 64	{64, 128}
	optimizer: Adam	—-
DDQN	target_update_freq: 200	{100, 200}
	M: 150	{150, 300, 1500}
PPO	M: 10000	{5000, 10000, 20000}
	λ: 1.0	{0.5, 1.0}
OPDS	MLP sizes: 128, 64, 32	—-
OPD	μ: 0.01	{0.001, 0.01, 0.1}

Table 1: Parameters and the search range of parameter-tuning for each Algorithm. Here target_update_freq is the frequency of updating the target network's parameters in DQN. And M means the number of episodes to sample during each step of training.

Additional Experiment Settings

We present some additional settings for our experiments in the following sections. All the experiments are conducted in a machine with an Intel Xeon E5-2673 CPU.

Dataset details

As we mentioned before, the datasets consist of each instrument's minute level price-volume information and the order information. 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. The order is generated following the method described in (Qian, Hua, and Sorensen 2007) 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.

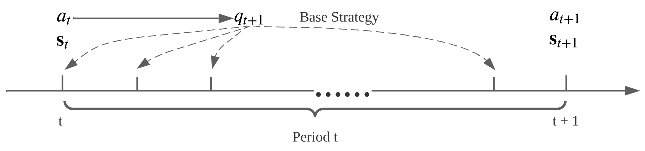

Figure 2: The detailed evaluation protocol of each timestep.

Evaluation protocol

The detailed trading procedure of each day in our experiment is described in this section. As we described before, every order record should be traded before the time horizon ends, i.e., within one trading day with 240 minutes. The detailed evaluation protocol is presented in Figure 2. 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.

Specifically, in all of our experiments, we set $T = 8$ in our experiment and each period has a length of 30 min-

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
OPD	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**
OPD	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
OPD	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 2: The test results over 10 selected instruments.

utes. Without loss of generality, following (Ning, Ling, and Jaimungal 2018), we use TWAP to conduct the actual execution within each period, i.e., we equally allocate q_{t+1} on every minute in period t. Note that, one can also replace TWAP with any other base strategy.

Further Investigation

Rolling window experiments

To make comprehensive comparison, we generate three datasets with a rolling window of eighteen-month length and six-month stride size and evaluate all compared methods on them. The detailed statistics of these datasets are listed in Table 3.

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 3: The dataset statistics.

The PA results of all compared methods on these datasets are shown in Table 4. We can tell from the results that our purposed OPD method steadily outperforms all other methods on all datasets, which further supports that oracle guidance improves the performance of order execution policy.

	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
OPD$^{\mathrm{S}}$	6.34 ± 0.20	5.28 ± 0.16	3.93 ± 0.57
OPD	$\mathbf{6.49\pm0.27}$	$\mathbf{5.35\pm0.73}$	$\mathbf{5.69\pm0.73}$

Table 4: The PA results on three rolling window datasets.

Advantages of universal training

Recall that our proposed method OPD uses the data of all the instruments to train a universal policy for executing all instruments' orders. However, the other learning-based methods (DDQN and PPO) were originally proposed 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 OPD 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 2. Recall that, in the main paper, our OPD method achieves the best overall performance when evaluated over all instruments. When executing the orders over single instrument, the universal policy of OPD achieves comparable results to that of PPO trained with only the specific single instrument data. After fine-tuning, OPD 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 OPD 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, nei-

Strategy	Reward($\times 10^{-2}$)	PA	GLR
OPDR (searching-based teacher)	-1.28	5.54	1.30
OPD (our proposed)	**-0.73**	**6.17**	**1.35**

Table 5: Performance comparison; the higher, the better. The performance has been aggregated as the mean value from the results with 6 random seeds.

ther 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.

Comparison with searching-based teacher

As we mentioned in the Method section, with perfect information, it is possible to find the optimal action sequence for the teacher using a searching-based method. As for the searching-based teacher with the perfect market information, specifically, we traverse all the trading orders of all the instrument in the dataset at all possible timesteps, and calculate the optimal action sequences for each order. We have described the learning-based teacher in the main paper. In this section, we compare the learning-based teacher and searching-based teacher from three aspects.

First, we compare the performance of the students guided by learning-based and searching-based teachers. The results are presented in Table 5, where OPD is the student guided by the learning-based teacher (proposed in our paper) and OPDR is the one guided by the searching-based teacher. As we can see, OPD achieves better performance comparing to OPDR. The reason might be that the learning-based teacher can generate actions that are more learnable for the student as it shares part of the state with the student.

Second, we analyze the time complexity of generating action sequences with learning-based and searching-based teachers. The time complexity for the learning-based teacher to generate optimal actions for a given order dataset is $\mathcal{O}(|\mathbb{D}|T)$, where $\mathbb{D}$ is the size of the dataset and T is the length of the time horizon. In contrast, as the searching-based teacher needs to traverse all possible actions at every timestep to decide the optimal action sequence, the time complexity for the searching-based teacher to generate optimal actions is $\mathcal{O}(|\mathbb{D}||\mathbb{A}|^T)$, where $|\mathbb{A}|$ is the size of all possible actions. Generally speaking, $|\mathbb{A}|^T \gg T$, as a result, it is time-consuming to generate actions in a searching-based way.

Third, as mentioned in our paper, searching-based teacher may require some human knowledge and, in many other environments, it is hard to implement even with perfect information. Our method based on learning and directly interacting with the environment, thus it can be transferred to other tasks more easily.

To conclude, the learning-based teacher can bring the student with better performance, higher efficiency and much better scalability, thus is preferred in our oracle policy distillation framework.

References

Almgren, R.; and Chriss, N. 2001. Optimal execution of portfolio transactions. *Journal of Risk* 3: 5–40.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *EMNLP*.

Ning, B.; Ling, F. H. T.; and Jaimungal, S. 2018. Double Deep Q-Learning for Optimal Execution. *arXiv preprint arXiv:1812.06600* .

Qian, E. E.; Hua, R. H.; and Sorensen, E. H. 2007. *Quantitative equity portfolio management: modern techniques and applications*. CRC Press.

Weng, J.; Zhang, M.; Duburcq, A.; You, K.; Yan, D.; Su, H.; and Zhu, J. 2020. Tianshou. https://github.com/thu-ml/tianshou.