

# Data Driven Optimal Execution with Deep Reinforcement Learning

Kang Li

Department of Statistics

University of Oxford

November 14, 2022

## Details of Algorithm

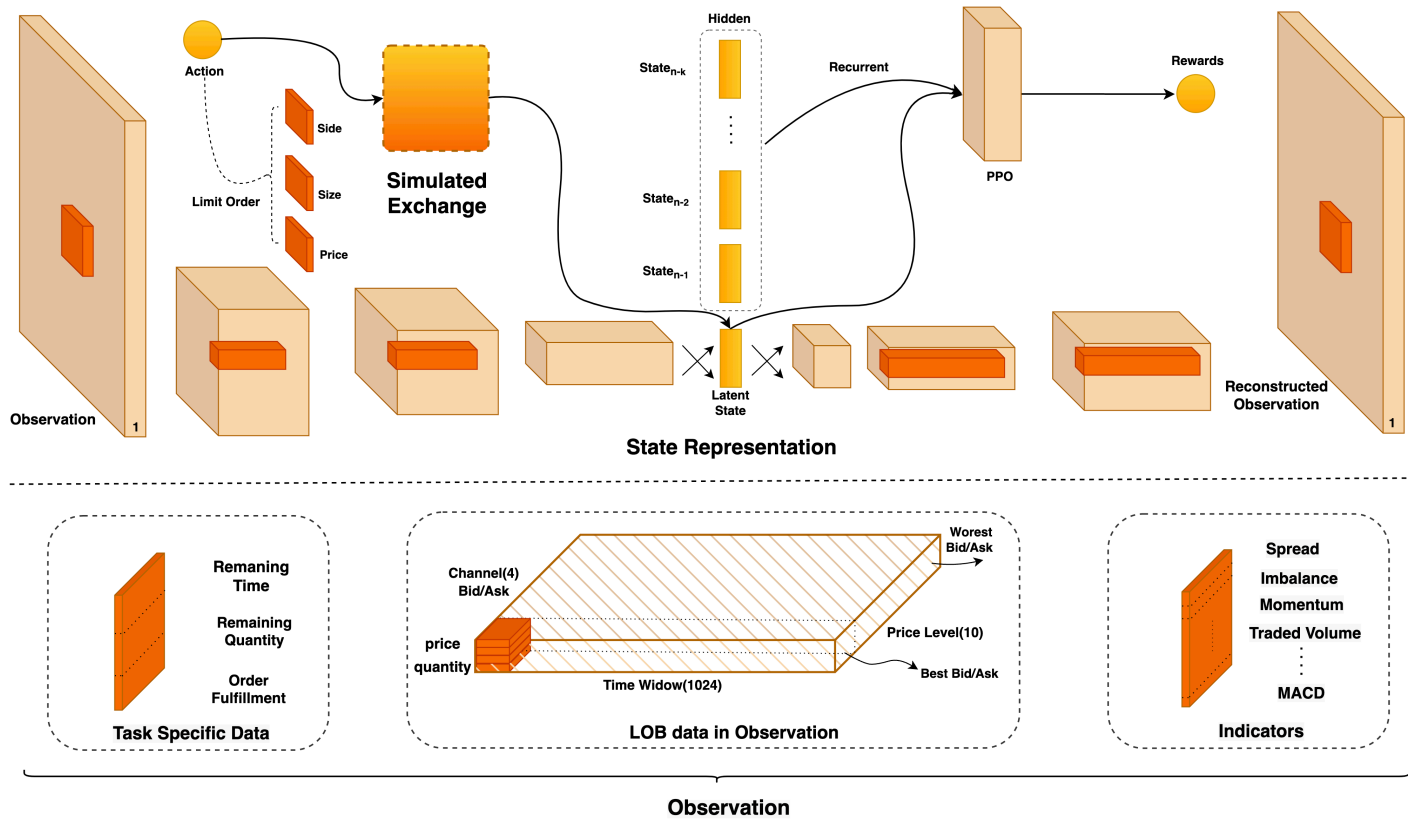


Figure 1: Model Diagram

The task of Agent is to complete the trading volume of sub-tasks within half an hour or one hour, and try to ensure that the final transaction price is higher (for the optimal liquidation task). Here

are the ten algorithmic details of the model.

First, traditional algorithms assume that Agent’s behavior has a linear or quadratic effect on market price. This is unreasonable. When we have historical data, we can directly get the impact of Agent orders on market through data simulation. For this purpose, we will build a stock trading simulator based on historical data.

Second, the problem of optimal execution requires us to consider the effects of behavior long ago on the present. For example, let’s say we’re in a hurry to sell all our shares at the beginning. Then at the end of the day, when we observe a better trade price, we miss the trade opportunity. Therefore, when designing reinforcement learning algorithms, we will consider adding time series. For example using the structure of a recursive neural network.

Third, we will separate the Optimal Liquidation task into sub-tasks with a smaller liquidation amount target and a shorter task duration based on the trading volume characteristics of the day. In our design, half an hour to one hour will be selected as the algorithm’s time window. The trading volume of each time window will be allocated in proportion to the trading volume of each period of the day according to the total trading volume.

Fourth, the observation of Agent is defined as the market information with the last minute as the time window. In our model, the observation will be divided into three parts. The first part is the task related observation: such as the remaining time, the number of tasks remaining, the degree of task completion and so on. The second part is the observation of order thin correlation: prices and quantities across the 10 price levels of all tick in a one-minute time window. The third part is the index related observation: such as bid-ask spread, momentum and order thin side imbalance index.

Fifth, we need to denoise and denoise observations. Because the Level2 order book data contains a lot of duplicate information, we also adopted the concept of representation learning to learn the Latent State extracted from the limit order book. The combination of AutoEncoder and RNN has been studied in previous papers[3] In addition, we will also try to use Sparse Transformer[2] to reduce dimension

Sixth, we will adopt the Agent with strategy gradient (such as PPO). The input of the strategy is observation, the output is action taken, and the goal is to maximize the total reward sum in the end. So how do we define the actions of the Agent? First, for observation, it is assumed that the Agent takes action at an interval of one second, and the specific action is to submit a limit order to the market. The optional parameters are: the price, quantity and direction of the order.



Seventh, for the design of training environment, we can also learn from the principle of Experience Replay, for example, [1]; Also can through the design of differential training environment, specific

reference cite DifferentiableAgentBasedEpidemiology. Such training environment can make the situation that is difficult to obtain good trading results be repeated for many times, so that the Agent of reinforcement learning can finally explore the appropriate strategy.

Eighth, for the Agent of reinforcement Learning, we do not start training from scratch, but refer to the idea of Residual Policy Learning[4]. After taking the optimal solution on the mathematical model as a reference for the input, the neural network will learn how to make reasonable adjustments based on this.

Ninth, the design of reward function. We refer to traditional Performance measures, such as Implementation Shortfall, PnL (with a penalty term of transaction cost), trading cost, profit, Sharpe ratio, Sortino ratio, and Return. Here, we define the reward function of Agent as follows, where  $\lambda \in [0, 1]$  serves as the weight adjustment index between the two terms.  $x_i$  is the number of orders traded at each price between two Agent actions.

$$Reward = \sum_{i \in \text{own trades}} x_i(p_i - p_t^{vwap}) + \lambda \cdot \sum_{i \in \text{own trades}} x_i(p_t^{vwap} - p_0) \quad (1)$$

- Advantage over Other Traders 
- Price Drift according to Initial Price 

Where,

$$p_t^{vwap} = \begin{cases} \frac{\sum_{j \in \text{others trades}} p_j x_j}{\sum x_j} & \text{if } volume_t > 0 \\ \frac{p_{t-1}^b + p_{t-1}^a}{2} & \text{if } volume_t = 0 \end{cases} \quad (2)$$

Tenth, traditional algorithms, such as the solution of Almgren Chriss model, Time Weighted Average Price, or Volume Weighted Average Price, will be used as the Benchmark for comparison.

## References

- [1] Di Chen, Yada Zhu, Miao Liu, and Jianbo Li. Cost-efficient reinforcement learning for optimal trade execution on dynamic market environment. In *Proceedings of the Third ACM International Conference on AI in Finance*, ICAIF '22, page 386–393, New York, NY, USA, 2022. Association for Computing Machinery.
- [2] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.
- [3] Otto Fabius and Joost R Van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.
- [4] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning, 2018.