
Journal of Scientific Computing and Machine Learning, vol. 1, pp. 9–15, 2025
Submission Date: January 3, 2025 Acceptance Date: January 26, 2025

Reinforcement Learning for Optimal Trade Execution

Lufan Wang[1] and Justin W.L. Wan[2]
[1,2]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario
l87wang@uwaterloo.ca

Abstract

In the financial industry, executing a large volume of shares (either buy or sell) is a common and yet challenging activity, since asset prices may fluctuate due to large movements in quantity. In this paper, we propose an enhanced reinforcement learning (RL) framework to solve the optimal trade execution problem by shaping the reward function such that it will account for both the expectation and variance of the costs. Our approach evaluates intermediate actions by comparing current performance against a baseline trade list. The proposed framework is flexible and works for both linear and nonlinear conditions. It can achieve performance comparable to the standard model [1] with errors within 3% across different cases. More importantly, our approach extends seamlessly to nonlinear scenarios.

Keywords Reinforcement learning · Reward shaping · Optimal trade execution · Almgren-Chriss model

1. Introduction

The optimal trade execution problem seeks strategies to execute large orders within a given time frame while minimizing the total costs. If the trade rate is too rapid, it may lead to negative market price change, which is often referred to as temporary price impact or permanent price impact [12, 18]. Temporary impact is the immediate, short-lived effect of a trade on execution price. Permanent impact refers to the lasting change in market price caused by the trade, affecting future valuations [9, 2, 5]. On the other hand, if the trade rate is too slow, the return may not be optimal due to price depreciation over time.

However, focusing solely on minimizing costs and market impacts can lead to significant uncertainty, resulting in high volatility of the returns. Obtaining a balance between return and uncertainty is therefore crucial. To measure this uncertainty, we use variance as a key metric. Modern approaches to the optimal trade execution problem aim to optimize a combination of the expectation of costs and their variance, ensuring a more balanced and stable solution. Researchers have explored various methods to identify optimal trading strategies, including dynamic programming [4, 17], modeling using partial differential equations [1, 8] and game theory [7]. However, they all face common challenges in re-solving or re-calibrating the model whenever the environment changes [4, 10, 7, 6], including adapting to shifts in market impact parameters (e.g., from linear to nonlinear) and extending to multiple assets.

Reinforcement learning (RL) offers a dynamic, model-free approach that adapts to changing conditions. However, prior RL applications [15, 11] have limitations. The model in [15] focuses only on final-period rewards, neglecting intermediate actions whereas the model in [11] ignores variance in reward shaping, leading to higher risks despite higher returns. These drawbacks can lead to a suboptimal action-selection strategy.

In this paper, we propose an enhanced RL framework for solving the optimal trade execution problem. The novelty of our approach is to incorporate the expectation and variance of cost via the reward function. Our model overcomes the drawbacks of prior models, achieves performance comparable to the classic Algren-Chriss model [1], and extends its applicability

to nonlinear conditions.

2. Reinforcement Learning Framework

2.1. MDP formulation

A fundamental concept in reinforcement learning is the Markov Decision Process (MDP)[3], which provides a mathematical framework for modeling decision-making. An MDP can be described by a 4-tuple, $(\boldsymbol{S}, \boldsymbol{A}, \mathbb{P}, \boldsymbol{R})$, where $\boldsymbol{S}$ = State space, $\boldsymbol{A}$ = Action space, $\mathbb{P}$ = State-transition probability and $\boldsymbol{R}$ = Reward [19].

For easy exposition, we only consider the selling case in this paper. In modeling trade execution using MDP, we define each state $s_t \in \boldsymbol{S}$ as a vector of three elements: the current timestep $t \in [0, N]$ where N is the total number of timesteps in the trading horizon; the current asset price $X_t \in \mathbb{R}$, and the remaining inventory $q_t \in [0, Q]$ where Q is the initial inventory of shared to be executed during the trading horizon.

Next, we let $\boldsymbol{A} = [0, 1]$ and define an action $u_t \in \boldsymbol{A}$ as the percentage of the current inventory of shares that needs to be sold. Thus, $u_t = 0$ indicates no shares are sold, $u_t = 1$ means the entire inventory is sold, and $u_t \in (0, 1)$ represents the proportion of shares on hand to be sold.

2.2. Reward Shaping

Reward shaping is a critical component in our RL model, which is also our main contribution. In modeling optimal trade execution, we minimize a combination of the expectation of total cost, measured as the implementation shortfall [16] x, and its variance. More precisely, at each state s_t, the agent takes an action u_t and sells $u_t \cdot q_{\text{average}}$ shares at the average selling price X_t^{average}. We define the cost as the difference between the average value and the actual money earned:

$$c_t = u_t \cdot q_{\text{average}} \cdot (X_0 - X_t^{\text{average}}). \tag{1}$$

The shortfall x is the sum of c_t over all timesteps:

$$x = \sum_{t=0}^{N-1} c_t. \tag{2}$$

If the agent has sold all the shares at timestep k, then for $t = k+1, \ldots, N-1$, we have $c_t = 0$. The objective of this mean-variance optimization problem is to minimize:

$$\min_x J(x) \equiv E(x) + \lambda Var(x), \tag{3}$$

where $\lambda \in [0, \infty)$ represents the weight given to variance. While $E(x)$ can be computed incrementally at each timestep, variance requires the complete trajectories and can only be calculated at the final timestep.

This dependency creates a critical challenge: how do we ensure that the agent's intermediate actions contribute to minimizing the final variance, even when variance cannot be directly observed during these steps? Traditional methods often overlook this, leading to unstable training and suboptimal results.

We need to establish a method for calculating variance at intermediate timesteps. To do this, we generate multiple price paths (W). For a given state s_t^w and $w \in [0, W-1]$, it is important to note that using the current cumulative cost $\sum_{k=0}^{t} c_k^w$ at timestep t to compute variance is inaccurate. This is because variance computation requires the shortfall x, which represents the total cost after all shares have been sold. As long as there are remaining shares at timestep t, the current cumulative cost $\sum_{k=0}^{t} c_k^w$ does not equate to the shortfall x of this price path and thus cannot be used to compute the mean-variance value. Hence, we assume that the agent executes all remaining shares at $t+1$ when needed. This ensures complete trajectories for variance computation and allows us to calculate the shortfall $x_{t+1}^w = \sum_{k=0}^{t+1} c_k^w = \sum_{k=0}^{N-1} c_k^w$. The mean-variance value at t is then defined as:

$$j_t^w \equiv J(x_{t+1}^w) = E(x_{t+1}^w) + \lambda \text{Var}(x_{t+1}^w). \tag{4}$$

If at a timestep t^*, the agent has already sold everything with $q_{t^*+1}^w = 0$, we assign $j_k^w = j_{t^*}^w$ for $k \in [t^*+1, N-1]$. We refer to this specific t^* as the finished index.

One may define the reward r_t^w as $-j_t^w$ so that maximizing r_t^w will be equivalent to minimizing j_t^w. However, this does not work well. Note that the ultimate objective is to minimize the mean-variance value at the final timestep, j_{N-1}^w. Focusing on minimizing j_t^w at each individual timestep could lead to suboptimal results, similar to a greedy algorithm. It is possible that an action may increase j_t^w temporar-

ily but result in a better next state s_{t+1}^w, ultimately leading to a more optimal j_{N-1}^w.

To address this, we introduce a reward structure that intuitively guides the agent. Specifically, at each time step t, we compare the agent's performance against a baseline trajectory. This baseline represents a reference strategy, derived from the trade list that achieves the lowest mean-variance value observed so far. For each baseline trajectory $w \in \{0, \ldots, W-1\}$, we compute its mean-variance value $j_t^{\text{baseline},w}$. We then average these values across all W trajectories to get the overall baseline mean-variance value:

$$j_t^{\text{baseline}} = \frac{\sum_{w=0}^{W-1} j_t^{\text{baseline},w}}{W}, \tag{5}$$

The agent earns a reward based on how much it outperforms the baseline; i.e. it is proportional to the difference between the baseline's mean-variance value and the agent's current value:

$$\tilde{r}_t^w = \frac{(j_t^{\text{baseline}} - j_t^w)}{j_t^{\text{baseline}}}. \tag{6}$$

We also include an ad hoc bonus to encourage the agent to minimize the final mean-variance value j_{N-1}. If the agent achieves a better final outcome than the baseline ($j_{N-1} < j_{N-1}^{\text{baseline}}$), it receives an extra reward of 10 at t^* and an additional reward of 1 for each intermediate timestep leading up to t^*. These bonuses incentivize the agent to make earlier decisions that may yield slightly lower immediate rewards but lead to a reduced j_{N-1}.

Finally, we can define our reward as:

$$r_t^w = \begin{cases} \tilde{r}_t^w + 1, & \text{if } j_{N-1} < j_{N-1}^{\text{baseline}} \text{ and } t < t^* \\ \tilde{r}_t^w + 10, & \text{if } j_{N-1} < j_{N-1}^{\text{baseline}} \text{ and } t = t^* \\ \tilde{r}_t^w, & \text{if } j_{N-1} \geq j_{N-1}^{\text{baseline}} \text{ and } t \leq t^* \\ 0, & \text{if } t > t^* \end{cases} \tag{7}$$

This reward formulation ensures that the agent focuses on outperforming the baseline, balances immediate and long-term goals, and stabilizes training by normalizing rewards and reducing sensitivity to fluctuations. The added bonus highlights the importance of the final outcome, encouraging the agent to optimize its strategy holistically.

2.3. The RL algorithm

The proposed reinforcement learning algorithm employs a standard actor-critic framework [14] to determine the optimal action-selection strategy given the MDP formulated in Section 2.1. It leverages the Bellman equation and Temporal Difference (TD) learning [20] to update both the policy network π and the value network V. Both networks are modeled using fully-connected, multi-layer feedforward artificial neural networks (ANN) with parameters ϕ and θ.

The training process alternates between two key steps: the Critic Update and the Actor Update. During the Critic Update, the value network is refined by simulating W_V trajectories, computing TD errors, and updating ϕ using an adam optimizer step[13]. The Actor Update improves the policy network by simulating W_{policy} trajectories and using the TD error to adjust θ. A baseline trajectory with the best mean-variance value j_{N-1} so far is maintained and updated if the agent achieves better performance. This baseline ensures stable training, while the iterative updates allow the agent to progressively refine its strategy and minimize the mean-variance cost.

3. Experiments

In this section, we first analyze the results of our RL model in comparison with the Almgren-Chriss model [1], which serves as a benchmark for the optimal trade execution problem. We then present the outcomes of our RL model incorporating a nonlinear price impact function.

3.1. Optimal Execution of Portfolio Transactions

We adopt the same parameter settings as in [1]. The initial portfolio consists of 1 million shares priced at $X_0 = 50$ \$/share. The stock has a bid-ask spread of $\frac{1}{8}$ \$/share, and a median daily trading volume of 1 million shares. We assume a liquidation time $T = 60$ days, divided into $N = 60$ daily trades ($\tau = 1$ day). A linear permanent impact function $g(v_t)$ in [1] is defined as:

$$g(v_t) = \gamma v_t, \tag{8}$$

where $\gamma = 2.5 \times 10^{-7}$ and v_t = trading velocity = $\frac{u_t}{\tau}$. The temporary impact function $h(v_t)$ is defined as:

$$h(v_t) = \epsilon \operatorname{sgn}(v_t) + \eta v_t, \tag{9}$$

where sgn is the sign function, $\eta = 2.5 \times 10^{-6}$, and $\epsilon = 0.0625$ is used as fixed costs of selling. The price evolution model in [1] assumes that the agent has no knowledge of future price movements, and thus, does not include a drift term. The price evolves according to:

$$X_t = X_{t-1} + \sigma\sqrt{\tau}\,\xi_t - \tau\, g\left(\frac{u_t}{\tau}\right), \quad t \in [1, N], \tag{10}$$

where $\sigma = 0.38$ represents the daily volatility, ξ_t are independent draws from a standard normal distribution. The average selling price is given by:

$$X_t^{\text{average}} = X_{t-1} - h\left(\frac{u_t}{\tau}\right), t \in [1, N]. \tag{11}$$

For our RL model, the value and policy networks are initialized with learning rates of $\alpha = 1 \times 10^{-3}$ and $\beta = 1 \times 10^{-3}$, respectively.

In our tests, we evaluate 13 different values of λ: 1×10^{-6} to 7×10^{-6} in increments of 0.5×10^{-6}. Each case leads to an optimal trading list $\{u_t\}_{t=0}^{N-1}$ and its corresponding inventory list $\{q_t\}_{t=0}^{N}$. Trades u_t are executed over 10,000 price trajectories, and the final value of $E(x) + \lambda \text{Var}(x)$ is calculated. Since price evolution involves randomness, the value of $E(x) + \lambda \text{Var}(x)$ can vary. To address this, we test each trade list 100 times on 10,000 trajectories, generating 100 outcomes for $E(x) + \lambda \text{Var}(x)$. The median of these outcomes is used as the final value for that trade list.

Figure 1 shows the efficient frontiers computed by our model (blue) and the benchmark (orange). The overlapping of the two curves demonstrates a strong agreement between the two models. To further illustrate the performance of the two models, the values of $E(x) + \lambda Var(x)$ are shown in Figure 2. Figure 3 plots the relative error as a percentage using $\left|\frac{\text{our result} - \text{benchmark}}{\text{benchmark}}\right|$. The relative error remains within 3% and decreases as λ increases.

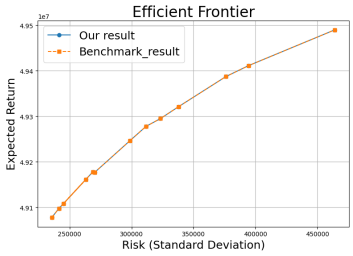


Figure 1: Efficient frontier curves for each λ value.

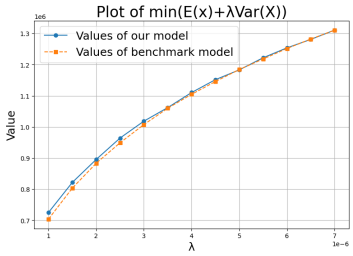


Figure 2: $\min(E(x) + \lambda \text{Var}(x))$ for each λ.

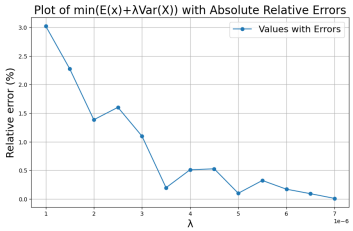
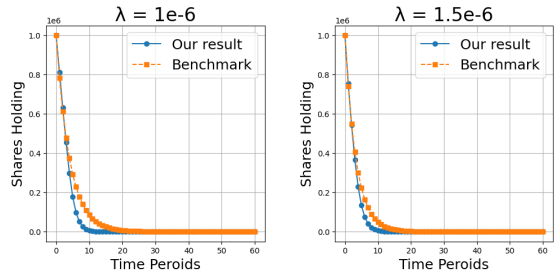
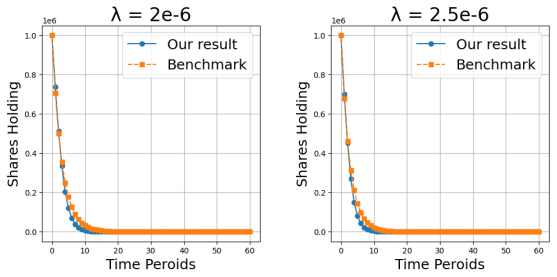


Figure 3: Relative error of $E(x) + \lambda \text{Var}(x)$ for each λ.

Moreover, We compare the inventory lists $\{q_t\}_{t=0}^{N}$, where $q_0 = Q$ and $q_N = 0$, for each λ. From Figure 4, our model (blue curves) consistently reaches zero faster than the benchmark (orange curves) for all λ values. Table 1 further confirms that our model completes trading earlier while achieving similar values of $J(x)$. This trend persists across all λ, with the gap widening as λ decreases.



(a) $\lambda = 1 \times 10^{-6}$ (b) $\lambda = 1.5 \times 10^{-6}$



(c) $\lambda = 2 \times 10^{-6}$ (d) $\lambda = 2.5 \times 10^{-6}$

Figure 4: Shares Holding vs Time Periods.

Table 1: Comparison of Time Periods to Stop for Our Results and Benchmark Results

λ **value** (1×10^{-6})	**Our Model stop period** $[J(x)]$	**Benchmark stop period** $[J(x)]$
1	25 $[7.25 \times 10^5]$	54 $[7.04 \times 10^5]$
1.5	22 $[8.22 \times 10^5]$	43 $[8.04 \times 10^5]$
2	23 $[9.00 \times 10^5]$	39 $[8.83 \times 10^5]$
2.5	21 $[9.64 \times 10^5]$	35 $[9.49 \times 10^5]$

Overall, our RL model demonstrates superior efficiency by completing liquidation faster, reducing exposure to market volatility and risk, while maintaining comparable performance on the trade execution problem.

3.2. Nonlinear Price Impact Function

In this section, we test our RL model under nonlinear conditions. Traditional partial derivative methods used in the Almgren-Chriss model struggle to handle nonlinearity. Nonlinear price impacts make the objective function non-convex, leading to multiple local minima and complicating the search for a global optimum. In contrast, our RL model learns dynamically without requiring prior knowledge. It offers a flexible alternative capable of handling nonlinear dynamics, adapting to market changes, and optimizing both expected costs and variance effectively.

We adopt most of the parameter settings from Section 3.1, with one key difference: the temporary price impact function. Here, we use the nonlinear temporary price impact function from [8], defined as:

$$h(v_t) = [1 + K_s \operatorname{sgn}(v_t)] \exp\left(K_t \operatorname{sgn}(v_t) |v_t|^{\upsilon}\right), \quad (12)$$

where $K_s = 2 \times 10^{-6}$, $K_t = 0$ and $\upsilon = 1$.

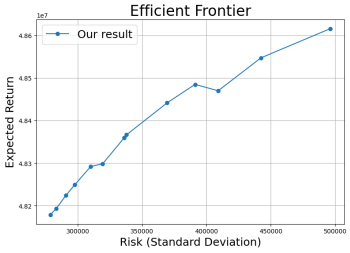


Figure 5: Efficient frontier curves for each λ value.

The efficient frontier plot is shown in Figure 5. As expected, the curve shows that higher returns correspond to higher risks. However, one point deviates slightly, with a lower return but higher risk compared to its left neighbor. Since the evaluation metric is $E(x) + \lambda Var(x)$, the optimal value can arise from different combinations of $E(x)$ and $Var(x)$. The trend of the minimum $E(x) + \lambda Var(x)$ in Figure 6 appears accurate, reinforcing the notion that different combinations of $E(x)$ and $Var(x)$ can yield the same optimal $E(x) + \lambda Var(x)$ value.

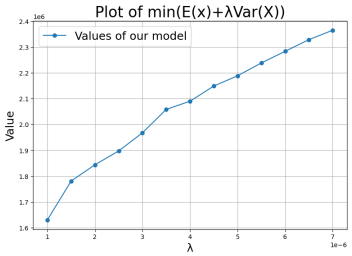


Figure 6: $\min(E(x) + \lambda \cdot \mathrm{Var}(x))$ for each test λ case.

4. Conclusions

In this paper, we propose an enhanced RL framework for optimal trade execution. Our RL model achieves a mean-variance value with less than 3% error across different λ values compared with Almgren-Chriss model, completing trades faster and reducing market exposure. Additionally, it adapts effectively to nonlinear price impacts, where traditional models fail, while maintaining the expected risk-return tradeoff. These results highlight the flexibility, efficiency, and robustness of our RL approach, making it a superior choice for complex trading scenarios and a valuable tool for advancing execution strategies in dynamic market environments.

Acknowledgments

This work is partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC).

References

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

[2] Robert Almgren, Chee Thum, Emmanuel Hauptmann, and Hong Li. Direct estimation of equity market impact. *Risk*, 18(7):58–62, 2005.

[3] Richard Bellman. A Markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[4] Dimitris Bertsimas and Andrew W Lo. Optimal control of execution costs. *Journal of financial markets*, 1(1):1–50, 1998.

[5] Jean-Philippe Bouchaud, Yuval Gefen, Marc Potters, and Matthieu Wyart. Fluctuations and response in financial markets: the subtle nature of random price changes. *Quantitative finance*, 4(2):176, 2003.

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

[7] Lorella Fatone, Francesca Mariani, Maria Cristina Recchioni, Francesco Zirilli, et al. A trading execution model based on mean field games and optimal control. *Applied Mathematics*, 5(19):3091, 2014.

[8] Peter A Forsyth. A Hamilton-Jacobi-Bellman approach to optimal trade execution. *Applied numerical mathematics*, 61(2):241–265, 2011.

[9] Jim Gatheral. No-dynamic-arbitrage and market impact. *Quantitative finance*, 10(7):749–759, 2010.

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

[11] Dieter Hendricks and Diane Wilcox. 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)*, pages 457–464. IEEE, 2014.

[12] Robert W Holthausen, Richard W Leftwich, and David Mayers. Large-block transactions, the speed of response, and temporary and permanent stock-price effects. *Journal of Financial Economics*, 26(1):71–95, 1990.

[13] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[15] Siyu Lin and Peter A Beling. An end-to-end optimal trade execution framework based on proximal policy optimization. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4548–4554, 2021.

[16] Andre F Perold. The implementation shortfall: Paper versus reality. *Journal of Portfolio Management*, 14(3):4, 1988.

[17] Huyên Pham. *Continuous-time stochastic control and optimization with financial applications*,

volume 61. Springer Science & Business Media, 2009.

[18] Hans R Stoll. Inferring the components of the bid-ask spread: Theory and empirical tests. *the Journal of Finance*, 44(1):115–134, 1989.

[19] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

[20] Gerald Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.