Optimal Execution via Reinforcement Learning in Agent Based Simulations

Yadh Hafsi
Edoardo Vittori

International Fintech Research Conference
31st January

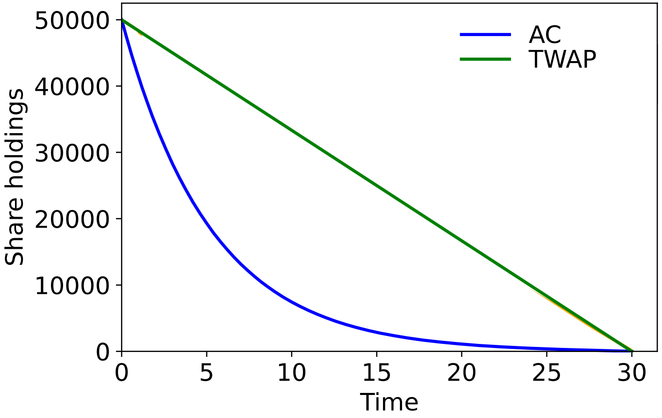
Introduction

Optimal Execution with reinforcement learning is a growing stream of literature

Definition of optimal execution

Given a trade to execute in a specified amount of time, minimize market impact and transaction costs

Execution trajectories



Why reinforcement learning

- Learn a state-based execution policy
- No assumptions on the market simulation

Optimal execution literature

- Karpe, Fang, Ma, Wang. 2020
- Ning, Lin, Jaimungal. 2018
- Hendricks and Wilcox. 2014
- Nevmyvaka, Feng, Kearns. 2006
- Almgren and Chriss., 2001
- Bertsimas and Lo., 1998

Reinforcement Learning for Optimal Execution

Problem definition and MDP description

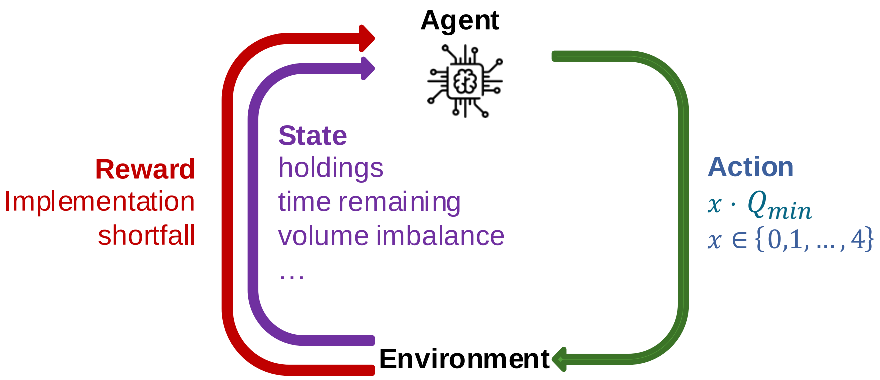
Reinforcement learning basics

- *MDP*: process which describes interaction between agent and environment
- *Objective*: find the policy π which maximizes the discounted sum of the rewards
- $\hat{J}_\pi = \mathbb{E}_\pi[\sum_i \gamma^i r_i]$, with the reward at time *i as* r_i

Optimal Execution MDP

- *State*: holdings, time remaining, current LOB state
- *Action*: market order of 4 different sizes
- *Reward*: $r_t = \underbrace{Q_t^k \times (P_0 - P_t)}_{\text{implementation shortfall}} - \underbrace{\alpha d_t}_{\text{penalty}}$

MDP interaction scheme



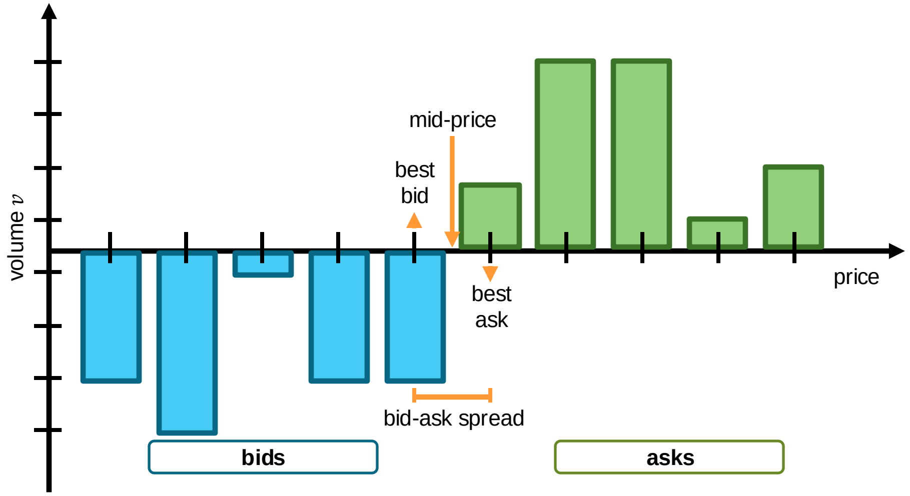
The Limit Order Book

Market and limit orders

Order types

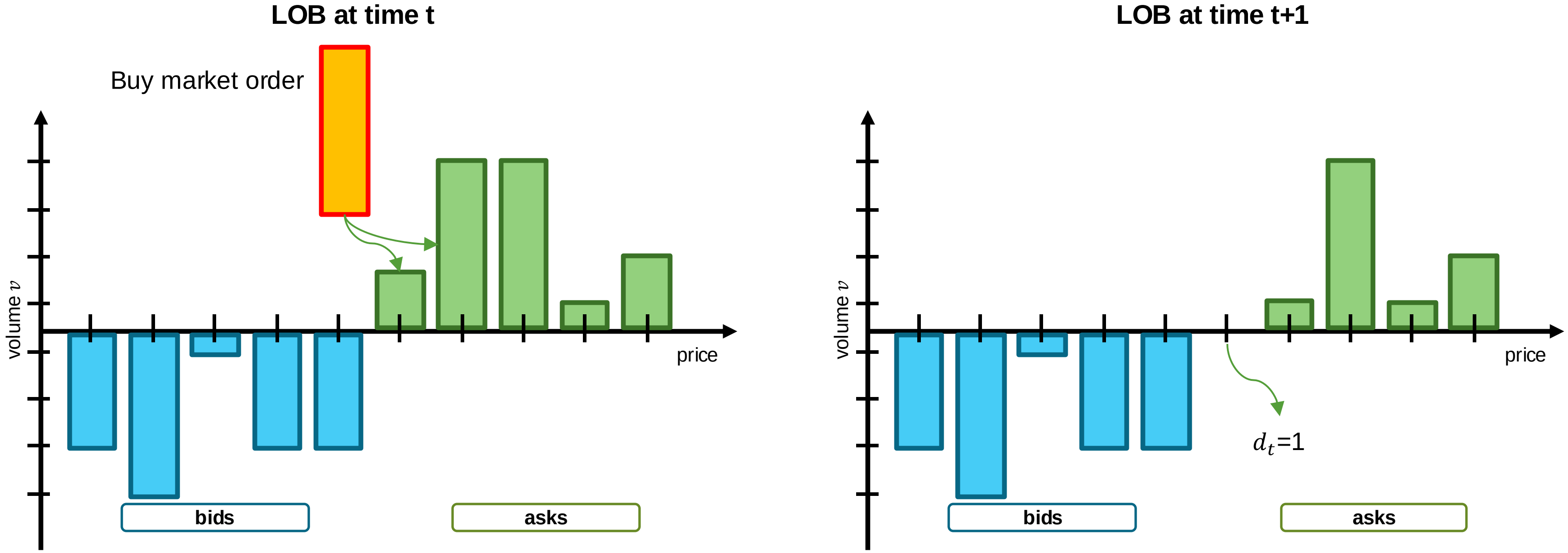
- Market order is an order to execute immediately at the best price possible
- Limit order is an order that specifies both the price and volume of a trade
- A limit order sits in the order book until it is either executed against a matching market order or canceled

Example of Limit Order Book



Trading in the LOB

A large buy order may cause the mid price to move (immediate market impact)



- The execution price of a market order of size V is $\sum_{i=\text{levels}} p_i v_i$ such that $\sum_{i=\text{levels}} v_i = V$
- Reward: $r_t = \underbrace{Q_t^k \times (P_0 - P_t)}_{\text{implementation shortfall}} - \underbrace{\alpha d_t}_{\text{penalty}}$

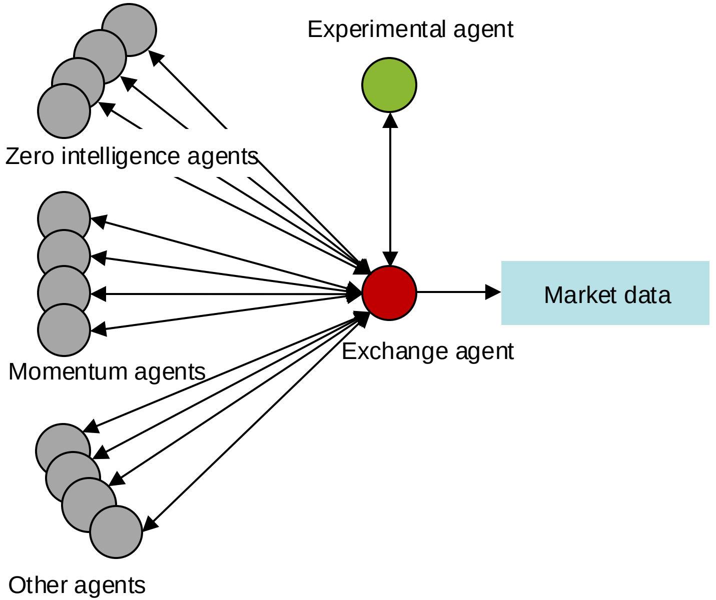
Simulating the LOB

There are three main approaches to simulating LOBs

Models for LOB simulation

- **Stochastic models**: represent the dynamics of the LOB using probabilistic processes
- **Machine Learning models**: learn simulated market behavior directly from historical data
- **Agent based models**: simulate the interactions of autonomous agents, each following a set of rules or strategies

ABIDES



Experimental Setting

Execution setup

- Buy 20k shares
- 30 minutes
- 1-second timesteps
- Actions: do nothing, buy 20, 40, 60, 80

Baseline algorithms

- TWAP: execute 20k/(30*60) at each timestep
- Passive algorithm: 60% do nothing, 40% random action
- Random algorithm: randomly selects an action
- Aggressive algorithm: buy 40 each second

RL Algorithm

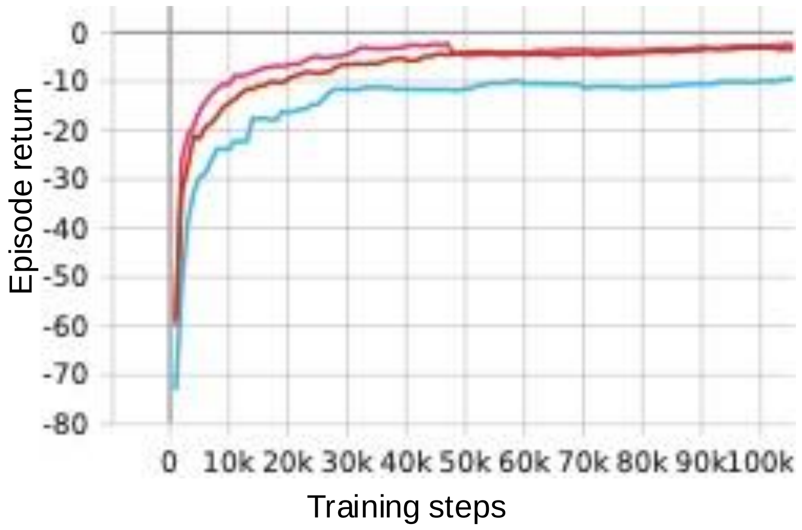
- Q-learning: $Q_t(s,a) = r(s,a) + \gamma \max_{a'} Q_t(s',a')$
- DQN

Initialize replay memory D to capacity N
Initialize action-value function Q and target $\hat{Q}$ with random weights
for episode = 1 to M **do**
    Initialize state s_1
    for t = 1 to T **do**
        With probability ϵ select a random action a_t
        Otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
        Execute action a_t
        Observe reward r_t and next state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in D
        Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D
        Set $y_j = r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \theta)$
        Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$
        Every C steps reset $\hat{Q} = Q$
    end for
end for

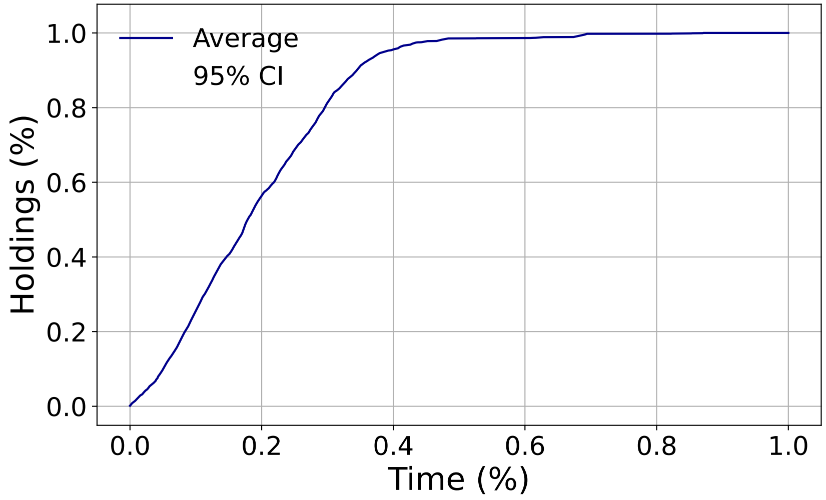
Experimental Results (1/3)

The RL agent executes a significant portion of its holdings rapidly

Learning curves



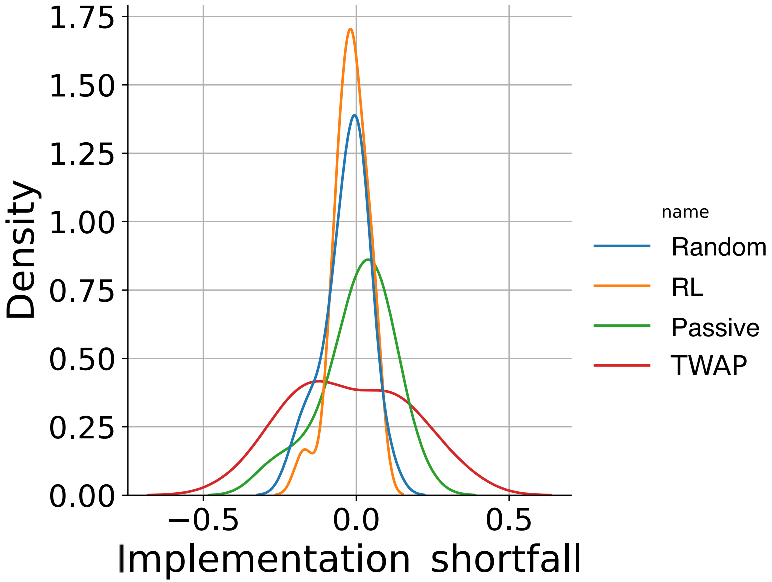
Execution trajectory



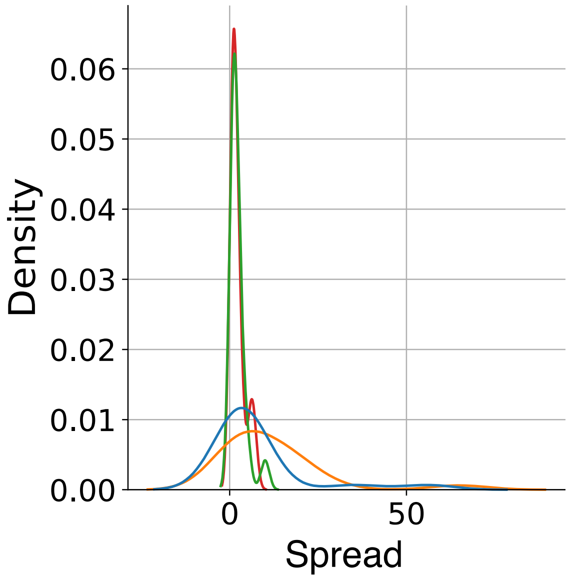
Experimental Results (2/3)

The RL agent trades close to the arrival price but widens the spreads

Distribution of implementation shortfall $Q_t^k \times (P_0 - P_t)$



Distribution of bid-ask spread

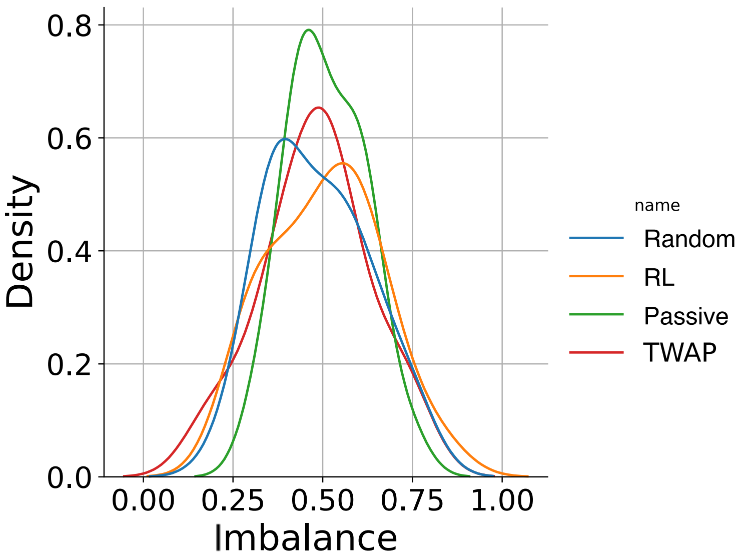


Experimental Results (3/3)

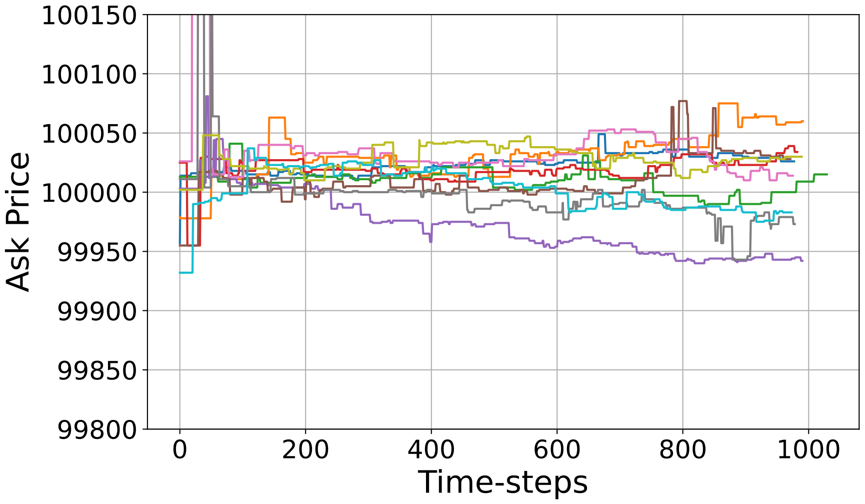
The RL agent manages to minimize market impact

Distribution of ask imbalance, 5 levels

$$v_h^k = \frac{TD_h^k}{TD_{\text{ask}}^k + TD_{\text{bid}}^k} \text{ for } k \in \{1, \ldots, d\},\ h \in \{\text{bid, ask}\}$$



Ask prices during execution



Optimal Execution via Reinforcement Learning in Agent Based Simulations

Q&A

Contacts

Edoardo Vittori
edoardo.vittori@intesasanpaolo.com

Yadh Hafsi
yadh.hafsi@universite-paris-saclay.fr

arXiv link