Reinforcement Learning for Optimized Trade Execution

Yuriy Nevmyvaka, Yi Feng, and Michael Kearns

劉冠銘, 羅名志 / NYCU IMF

Introduction

- the first large-scale empirical application of reinforcement learning to optimized trade execution.
- based on 1.5 years of millisecond time-scale limit order data from NASDAQ (from INET)
- Overall, we can achieve improvement in execution of 50% or more over baseline policy (S&L).
- goal :
  - to SELL V shares of a given stock within a fixed time period H
  - maximizes the revenue received **(use trading cost to measure)**

Introduction

- following slides :
- An RL formulation of Optimized Execution
  - problem identification
  - States, Action, Rewards
  - algorithm and experimental methodology
- Experimental Results
  - part 1 : learning with private variables only
  - part 2 : adding market variables
- Conclusion

An RL formulation of Optimized Execution

problem identification

States, Action, Rewards

algorithm and experimental methodology

Problem

- to sell V shares of a given stock in time horizon H
- baseline： submit and leave (S&L) policies
  - **pick a fixed limit order price p**, and place a sell order for all V shares
  - after H minutes, submit a market order for remaining shares and leave

States

- includes private and market variables
- private variable :
  - elapsed time : t
  - remaining inventory : i
- to discretize these variables, we pick I and T, which are the maximum values of the variables :
  - if V(total shares) = 10000, I = 4, i is represented in rounded units of 2500 shares
  - if H(total time) = 2 min, and T = 4, t is represented in units of 30 min

States

- market variable :
  - Bid-Ask Spread
  - Bid-Ask Volume Misbalance : signed difference between volumes quoted at the bid and at the ask
  - Market Order Cost : how much will it cost to submit a market order for the balance of inventory immediately.
  - Signed Transaction Volume : the signed volume of all trades within last 15 sec
- each state x ∈ X is a vector of attributes includes private / market variable :
  - x = <t, i, o_1, o_2, o_3 …>

Actions

- action a : place a limit order for all of our unexecuted shares at ‘ask-a’
- we are effectively withdrawing any previous outstanding limit order
- $$\begin{cases} a = 0, & comming\ in\ at\ the\ current\ ask \\ a > 0, & crossing\ the\ spread\ to\ the\ buyers \\ a < 0, & place\ our\ order\ in\ the\ sell\ book\ deeper \end{cases}$$
- **L : the number of possible actions**

Rewards

- measure performance :
  - execution prices achieved by a policy relative to the mid-spread price at the start of the episode
  - mid-spread price : $(a_0 + b_0)/2$
  - usually, the exec prices are worse than initial mid-spread price
    - define trading cost of a policy as the underperformance compared to the spread price

Algorithm

- the approximately Markovian nature of trade execution :
  - the optimal action at any given point in time is approximately independent of any previous actions.
- example :
  - t = T, forced to submit a market order for all unexecuted shares
  - we now have all the information we need to determine the optimal action for t = T-1
  - having done that, we move one step t = T-2 and so on until t = 0
  - Having arrived at t = 0, we have a globally optimal policy
- another assumption : our actions do not affect the behavior of other market participants
  - our actions do not affect market state variables

Algorithm

cf.

$$Q^{\pi}(x_n, a_n) = (1 - \alpha_n) Q^{\pi}(x_n, a_n) + \alpha_n \left(r_n + \gamma \max_b Q(x_{n+1}, b) \right)$$

$$Q^{\pi}(x_n, a_n) = r_n + Q^{\pi}(x_{n+1}, \pi(x_{n+1}))$$

- our approach is similar to Q-learning :
  - we try all possible actions and update the expected cost associated with taking each action and following the optimal strategy afterwards
- the update rule is the following :

$$c(x, a) = \frac{n}{n+1} c(x, a) + \frac{1}{n+1} \left[c_{im}(x, a) + \arg\max c(y, p) \right]$$

  - c(x, a) : the cost of taking action a in the state x and in all subsequent states
  - $c_{im}(x, a)$: the immediate cost of taking action a in state x
  - n : the number of times we tried a in x
  - y : new state after taking a
  - p : the action taken in y

Algorithm

Optimal_strategy (V, H, T, I, L)

For t = T to 0

While (not end of data)

Transform (order book) $\rightarrow o_1, o_2, \ldots, o_R$

For i = 0 to I {

For a = 0 to L {

Set x = {t, i, $o_1, o_2, \ldots, o_R$}

Simulate transition x $\rightarrow$ y

Calculate c_{im}(x, a)

Look up argmax c(y, p)

Update c(x, a)

* L is the number of actions in each state

Experimental Methodology

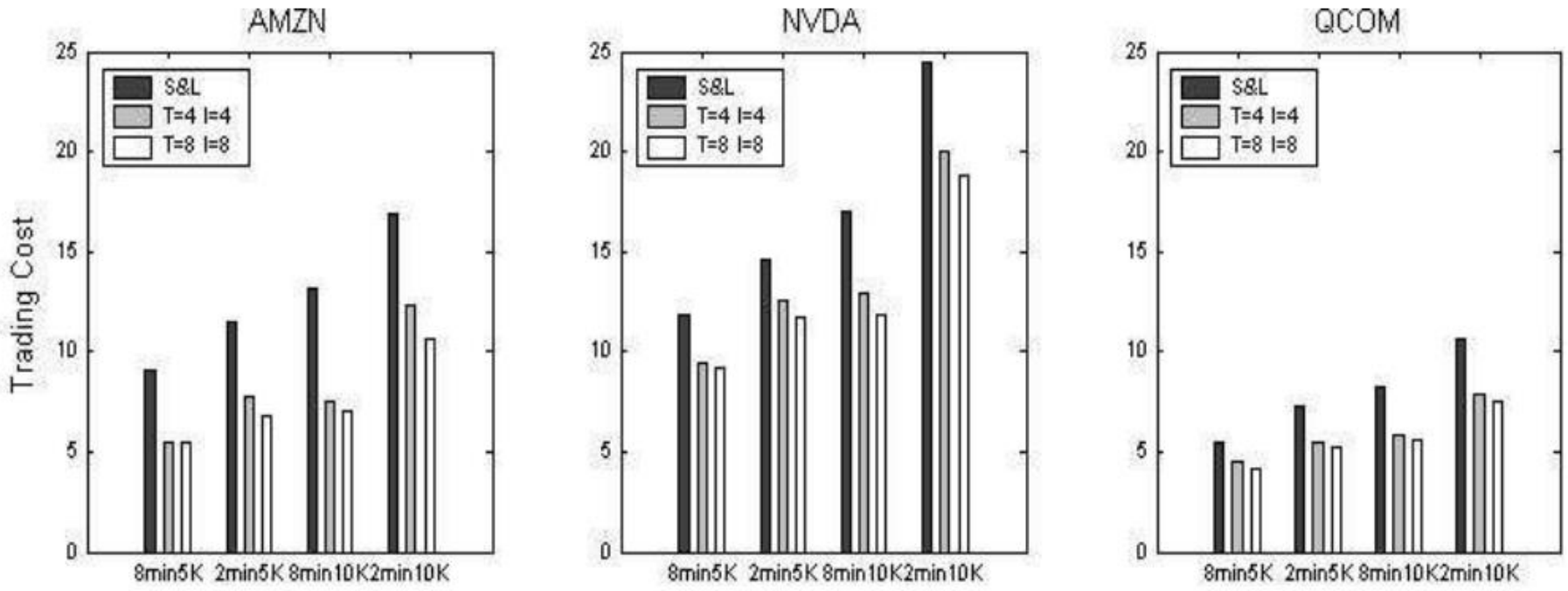
1. investigate 3 stocks : AMZN, NVDA, QCOM; two order sizes : V = 5000 shares / 10,000 shares; two execution horizons : H = 2 min / 8 min
2. choice for the I, T was made, which are the resolutions to I, t
3. market variables were selected
4. apply the algorithm to learn an optimized execution policy over the chosen state space
5. compare on the 6 month test set with several baseline strategies

Experimental Results

part 1 : private var only

part 2 : with market var

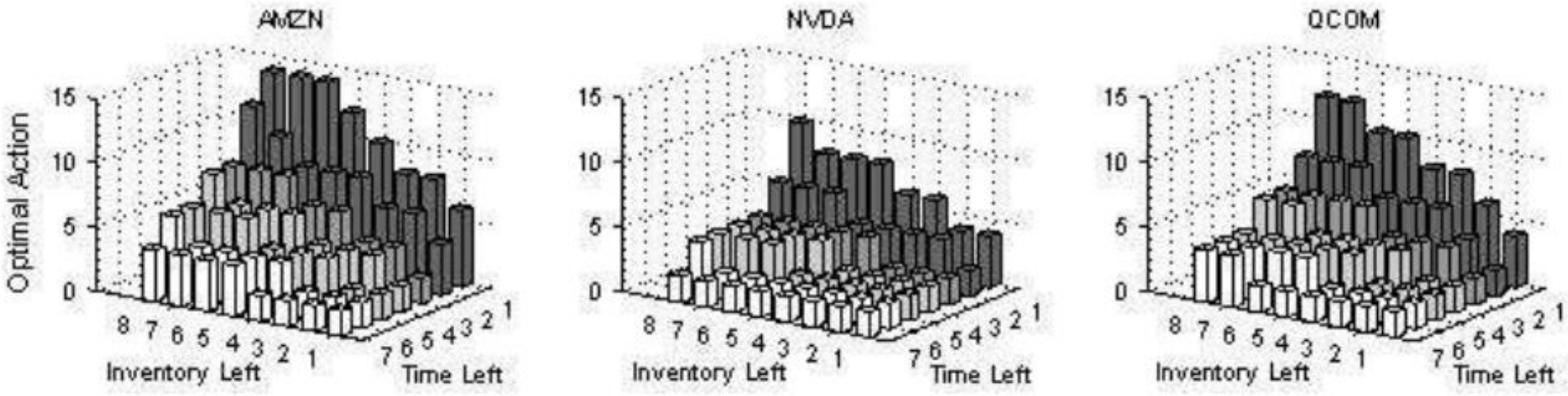
Experimental Results (private var only)



- Relative improvement over S&L ranges from 27.16% to 35.50% depending on our choices of T and I

T : numbers of time interval
I : numbers of inventory interval

Experimental Results (private var only)



- discussion and visualization of time and inventory

*** action : ask-a**

Experimental Results (private var only)

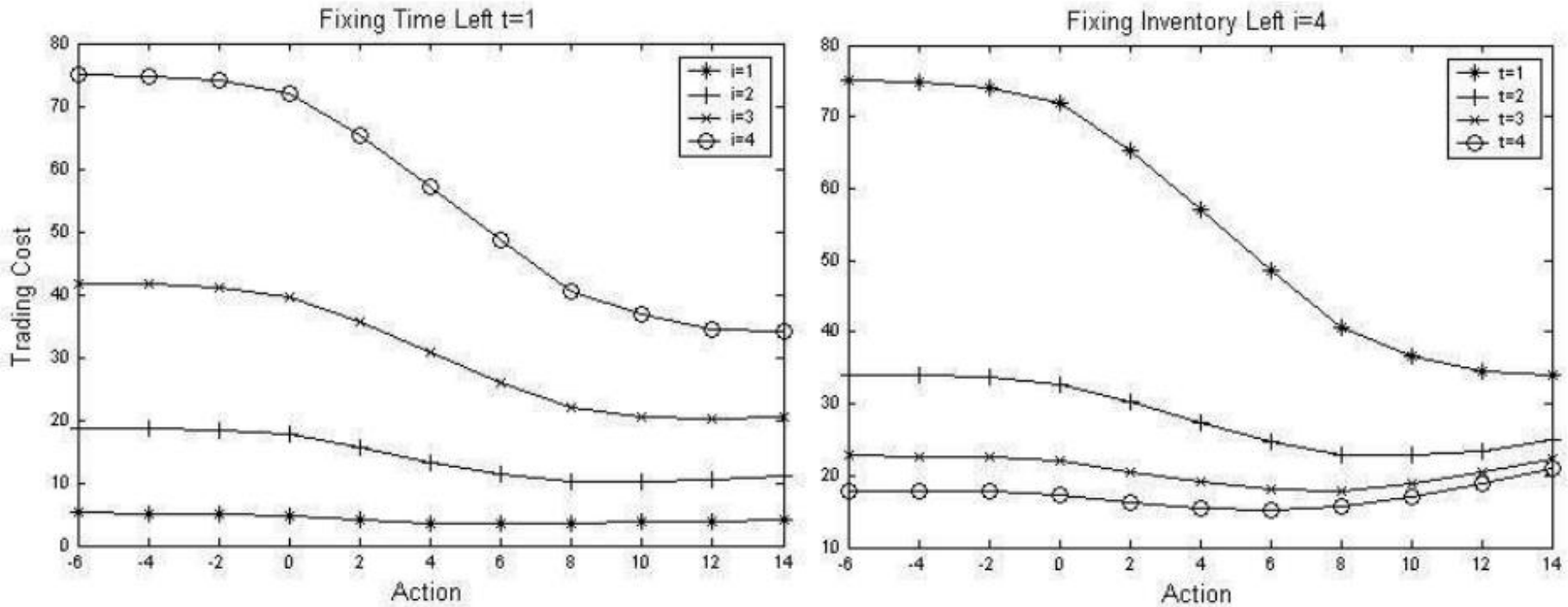


Figure 5. Q-values: curves change with inventory and time (AMZN, T=4, I=4)

Experimental Results (with market var)

- Additional trading cost reduction with introducing market variables
- with different combinations

Bid-Ask Spread	7.97%
Bid-Ask Volume Misbalance	0.13%
Spread + Immediate Cost	8.69%
Immediate Market Order Cost	4.26%
Signed Transaction Volume	2.81%
Spread+ImmCost+Signed Vol	12.85%

Experimental Results (with market var)

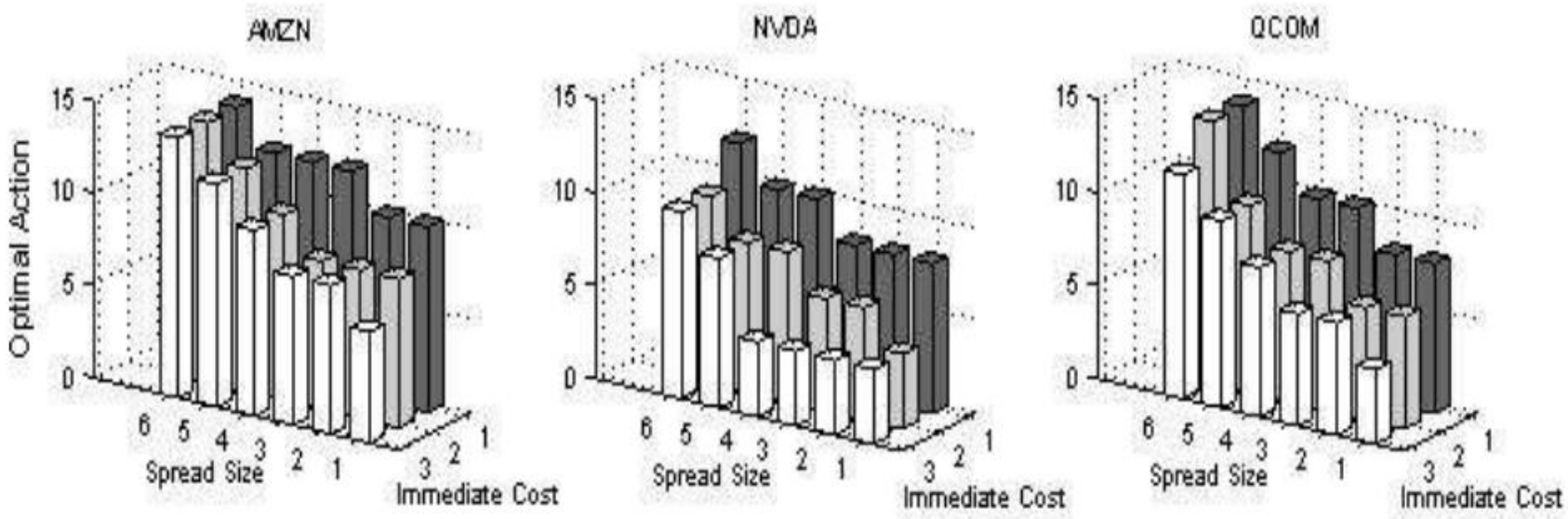


Figure 6. Large spreads and small market order costs induce aggressive actions

*** action : ask-a**

Experimental Results (with market var)

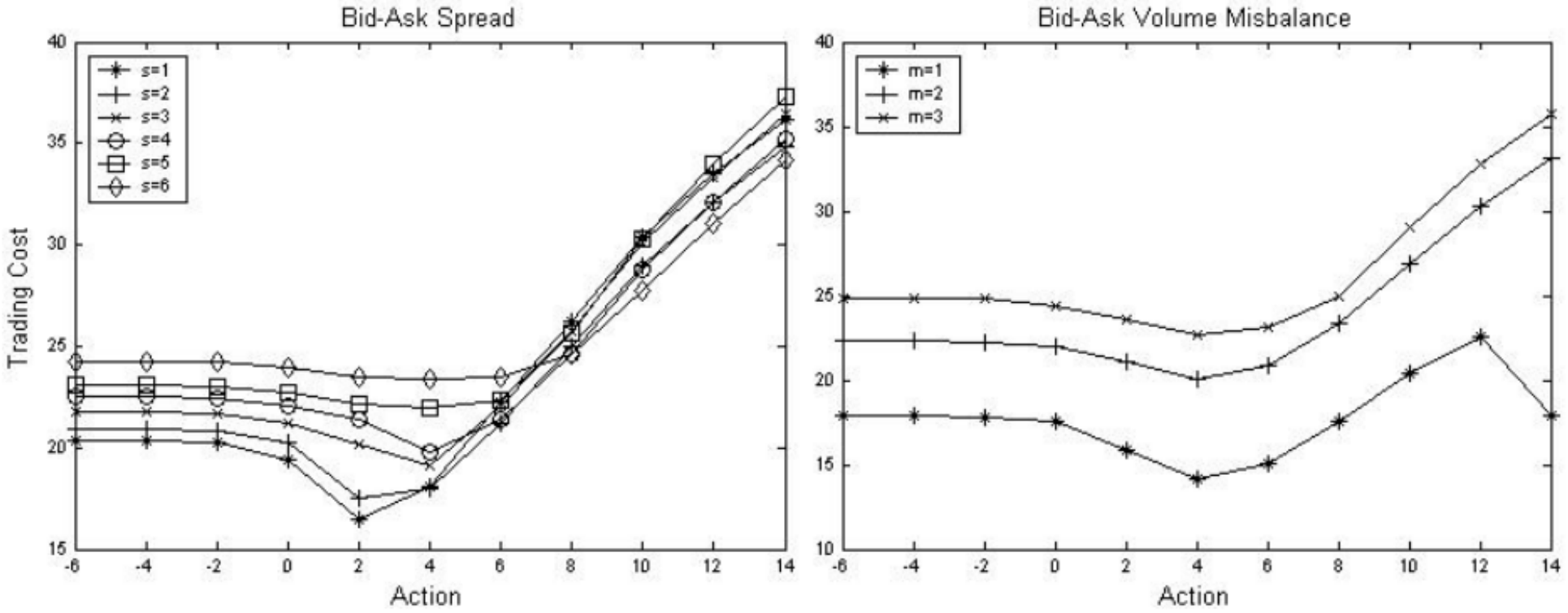


Figure 7. Q-values: cost predictability may not affect the choice of optimal actions

some market variables do not improve the performance of our algorithm

Conclusion

- First large-scale application of RL in optimized trade execution
- Provided improvements of up to 50% or more in comparison to S&L
- Adapter this work to other precisely-defined finance problems

Implementation

- data
- definition of S&L
- action space

Ref.

- https://www.cis.upenn.edu/~mkearns/papers/rlexec.pdf
- https://zhuanlan.zhihu.com/p/68654927?utm_id=0&fbclid=IwAR2AvYLp6k_iiIE5r_FQ6IS_AbBZD3y3ePzkZ4BLdVVySzBjSj-K3xy03Mw
- https://github.com/luliu31415926/Reinforcement-Learning-for-Optimized-trade-execution/tree/master/code
- https://github.com/llSourcell/Q-Learning-for-Trading?fbclid=IwAR2mENVbw8jYygrgX9eLQBMRmGN4berbRK6KkLYFsggNEapEz2rtSlzI8PQ

Ref.

- https://www.cis.upenn.edu/~mkearns/talks/ICML2006.pdf
- https://cs.uwaterloo.ca/~ppoupart/teaching/cs885-spring20/slides/cs885-reinforcement-learning-for-optimized-trade-execution.pdf
- https://www.cupoy.com/qa/club/ai_tw/0000016D6BA22D9700000001637 5706F795F72656C656173654B5741535354434C5542/0000017C0D6897CD 000000296375706F795F72656C656173655155455354
- https://zhuanlan.zhihu.com/p/68654927?utm_id=0&fbclid=IwAR1g3nEvcIIeg58tB8aUUVwQn0ZRLhs1F9J0qYzz87KBBXToZQJ0XkvYKQ0
- https://www.youtube.com/watch?v=rRssY6FrTvU
- https://www.youtube.com/watch?v=D9sU1hLT0QY