Optimal Portfolio Liquidation in Dark Pool

Ranjan Mishra

Optimal Portfolio Liquidation in Dark Pool

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Ranjan Mishra
[Roll No: CS-1519]

under the guidance of

Dr. Diganta Mukherjee
Associate Professor
Sampling and Official Statistics Unit

Indian Statistical Institute
Kolkata-700108, India

July 2017

To everyone in the world

CERTIFICATE

This is to certify that the dissertation entitled **"Optimal Portfolio Liquidation in Dark Pool"** submitted by **Ranjan Mishra** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bona fide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Diganta Mukherjee
Associate Professor,
Sampling and Official Statistics Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would like to show my highest gratitude to my advisor, *Diganta Mukhejee*, Sampling and Official Statistics Unit, Indian Statistical Institute, Kolkata, for his guidance and continuous support and encouragement.

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable suggestions and discussions which added an important dimension to my research work.

Finally, I am very much thankful to my parents and family for their everlasting supports.

Last but not the least, I would like to thank all of my friends for their help and support. I thank all those, whom I have missed out from the above list.

Ranjan Mishra
Indian Statistical Institute
Kolkata - 700108 , India.

Abstract

When a large investor decides to liquidate his portfolio in finite time period, he can put market orders to do so. However large market orders may adversely impact prices. Which in turn may produce lower liquidation return . Dark pools are new kind of market, where not all the information is made public after the trade execution.

We have developed a dynamic strategy to place market orders as well as dark pool orders such that total expected liquidation return is maximized. The problem is formulated as Morkov Decision Process in finite horizon and solved using approximate dynamic programming technique.

.

Keywords: *Portfolio Liquidation, Dark Pool, Dynamic Programming, Markov Decision Process*

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

When a large investor decides to liquidate his portfolio, he can place his whole portfolio as market order. But placing a large order might move the prices adversely. Impact of the order size on the prices has been studied. Several models have been proposed for same, such as [1, 12] . The impact may cause reduced return on portfolio. One simple solution is to put the orders of equal block market orders through time, but that might not produce the best return with the dynamics of the prices through time.

New kind of markets have emerged called as *dark pools* [5, 9] , as contrary to conventional market after the trade execution, dark pools do not make trade details public. One has to submit orders in dark pool to know the depth of the market. Lack of the trade information makes these markets more suitable for large investors since these have less impact on the prices. One does not know the depth of the market. The execution prices in these dark pools are mostly derived from the exchange such as NBBO mid price [5]. However the orders in dark pools are not guaranteed to be executed either whole or in parts. Most of the time it happen that the orders get cancelled. Since the large investor wanted to liquidate his portfolio in limited period of time, it may not be best to put the orders in dark pools only.

We propose the solution to put the orders in both the conventional (market order) and dark pools such that the return of the portfolio liquidation is maximized with the dynamics of the price governed by the size of the orders placed in the conventional market.

We formulate the problem as sequential decision problems, which naturally falls in the realm of Markov decision process, since the liquidation has to be in finite amount of time, this leads to the finite horizon Markov decision process. Optimal solution of a Markov decision problem with small action and state space can be achieved using backward dynamic programming which involves storing the value function of each state in a tabular fashion and then recursing backward in time to solve the Bellman Equation. However when state space is large using table to store values is not computationally feasible. Further to that if the action space is also large it does make the problem more demanding. We use techniques from approximate dynamic programming also called as reinforcement learning in computer science community to solve our problem.

1.2 Our Contribution

Our contributions are summarized as follows.

- As the problem is to make decision through time and based on the dynamics of the market, we have formulated this as Markov Decision Process.
- We provide a solution as non stationary policy based on time and dynamics of the market considering the problem as finite horizon, using actor critic method.
- We show the empirical result that the TD error converges as the number of iteration increases, based on our proposed solution.
- We show the empirical result that policy obtained with our approach produces better return than naive policy of equal block market orders.

1.3 Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2 we briefly discuss about the Markov Decision Process.In chapter 3, we describe the formulation and solution scheme to our problem in detail.In Chapter 4, we provide the analysis of the empirical results obtained based on the proposed scheme. In the concluding Chapter 5, we summarize the work done and future directions related to our work.

Chapter 2

Markov Decision Process

A sequential decision making problem where the state of the system depends on the action taken can be modelled as Markov decision process. Below we discuss the deterministic dynamic optimization problem and then gradually move towards stochastic optimization problem.

2.1 Introduction

Consider a discrete time dynamic system with following dynamics [4],

$$x_{t+1} = f_t(x_t, u_t)$$

where

- x_t is the current state of the system, e.g position and or velocity of particle or price of securities in stock market.
- u_t is the control applied to the system for example force applied to particle or number of stocks applied to buy/sell.

and a reward $r_t(x_t, u_t)$ is generated such as capital gain from selling securities. We take $t \in 0, 1, 2, ..., T$ is set of discrete time steps. Total reward accumulated can be written as

$$\sum_{t=0}^{T-1} r_t(x_t, u_t) + r_T(x_T) \tag{2.1}$$

where x_T is terminal state and $r_T(x_T)$ is terminal reward.

The objective is to maximize the overall reward generated, e.g. maximize the overall capital gain from selling the securities. We can write it as an optimization problem below

$$\max_{u_0, u_1, ... u_{T-1}} \sum_{t=0}^{T-1} r_t(x_t, u_t) + r_T(x_T) \tag{2.2}$$

If we assume a discounting due to time such as in reward generated, we can add a discounting factor γ in the above equation and write it as

$$\max_{u_0,u_1,...u_{T-1}} \sum_{t=0}^{T-1} \gamma^t r_t(x_t, u_t) + \gamma^T r_T(x_T) \tag{2.3}$$

Given the initial state x_0 our goal is to find control sequence $(u_0, u_1, ..., u_{T-1})$. This problem can be solved by enumerating over all actions sequences and looking into exhaustive solution but this is computationally intractable.

2.2 Dynamic Programming

Dynamic programming [4] is technique of breaking the large problem to smaller problems that are easy to solve, popularized by Richard Bellman. Consider the equation (2.2) ,which we can rewrite as

$$\begin{aligned} V_0(x_0) &= \max_{u_0,....,u_{T-1}} \sum_{t=0}^{T-1} \gamma^t r_t(x_t, u_t) + \gamma^T r_T(x_T) \\ &= \max_{u_0} [r_0(x_0, u_0) + \max_{u_1,....,u_T} \sum_{t=1}^{T} \gamma^t r_t(x_t, u_t) + \gamma^T r_T(x_T)] \\ &= \max_{u_0} [r_0(x_0, u_0) + \gamma V_1(x_1)] \end{aligned} \tag{2.4}$$

In general we can write

$$V_t(x_t) = \max_{u_t} [r_t(x_t, u_t) + \gamma V_{t+1}(x_{t+1})] \tag{2.5}$$

This is called Bellman equation [4]. Which we will be able to solve by backward substitution in the following way

- Step 1: Initialize $V_T(x_T) = r_T(x_T)$ for all terminal states x_T set $t = T - 1$
- Step 2: Calculate
$$V_t(x_t) = \max_{u_t} [r_t(x_t, u_t) + \gamma V_{t+1}(x_{t+1})]$$
$\forall x_t \in S$ where $x_{t+1} = f(x_t, u_t)$
- Step 3: if $t \geqslant 0$ return to step 1 else stop

This technique is called backward dynamic programming [4, 16, 8].

2.3 Markov Decision Process

Consider following similar to deterministic discrete time dynamic optimization problem with stochastic component added to it, the dynamic of the system is now governed by equation equation (2.6)

$$x_{t+1} = f_t(x_t, u_t, \epsilon_t) \tag{2.6}$$

where ϵ_t is a random noise.

In this setting we can rewrite the optimal equation as finding optimal policy $\pi = (\pi_0, \pi_1, ...\pi_{t-1}) \in \Pi$ such that

$$V_0(x_0) = \max_{\pi \in \Pi} E_{\epsilon_0, \epsilon_1, ...\epsilon_T}[\sum_{t=0}^{T-1} \gamma^t r_t(x_t, \pi_t(x_t)) + \gamma^T r_T(x_T)] \tag{2.7}$$

Similar to above observation in case of deterministic case, we can establish relation and get a backward substitution method to find optimal sequence of actions

$$\begin{aligned} V_0(x_0) &= \max_{\pi \in \Pi} E_{\epsilon_0, \epsilon_1, ...\epsilon_{T-1}}[\sum_{t=0}^{T-1} \gamma^t r_t(x_t, \pi_t(x_t)) + \gamma^T r_T(x_T)] \\ &= \max_{\pi \in \Pi} \sum_{x_1} P(x_1|x_0, \pi(x_0)[r_0(x_0, u_0) + E_{\epsilon_1, ...\epsilon_{T-1}} \sum_{t=1}^{T} \gamma^t r_t(x_t, u_t) + \gamma^T r_T(x_T)] \\ &= \max_{\pi_0} \sum_{x_1} P(x_1|x_0, \pi(x_0)[r_0(x_0, u_0) + \gamma V_1(x_1)] \end{aligned} \tag{2.8}$$

hence above can be written in general as

$$\begin{aligned} V_t(x_t) &= \max_{\pi_t} \sum_{x_{t+1}} P(x_{t+1}|x_t, \pi_t(x_t))[r(x_t, \pi_t(x_t)) + \gamma V_{t+1}(x_{t+1})] \\ &= \max_{\pi_t}[r(x_t, \pi_t(x_t)) + \gamma \sum_{x_{t+1}} P(x_{t+1}|x_t, \pi_t(x_t)) V_{t+1}(x_{t+1})] \\ &= \max_{\pi_t}[r(x_t, \pi_t(x_t)) + \gamma E V_{t+1}(x_{t+1})] \end{aligned} \tag{2.9}$$

where expectation is based on one step transition i.e. distribution of ϵ_t. We can utilize the backward dynamic programming method to generate the optimal control sequence or optimal policy.

Moreover, Markov decision process can be formally defined [7, 16, 6] as 5-tuple $(S, A, P.(.,.), R(.,.), \gamma)$ where

- S is the set of possible states of system.
- A is set of possible actions that can be taken.
- $P_a(s, s') = Pr(s_{t+1} = s'|s_t = s, a_t = a)$ that is the probability of system being in state s‘ at time $t + 1$ given it was in state s at time t and action taken was a.
- $R(s, a)$ is reward generated when action taken a in state s
- $\gamma \in [0, 1]$ is the discounting factor.

The problem in MDPs is to find a policy that maps state to action i.e. a function $\pi : S \to A$ that decides action $\pi(s)$ given the state is s. It be written as to find a policy $\pi = (\pi_0, \pi_1, ...\pi_T)$ such that accumulated reward is maximized i.e.

$$\max_{\pi \in \Pi} E[\sum_{t=0}^{T} \gamma^t r_t(s_t, \pi_t(s_t))] \tag{2.10}$$

where expectation is taken over $P_\pi(.,.)$ Above discussed problem is finite horizon Markov decision problem, if the time horizon is not limited then the problem is said to be infinite horizon problem, it can be written as

$$\max_{\pi \in \Pi} E[\sum_{t=0}^{\infty} \gamma^t r_t(s_t, \pi_t(s_t))] \tag{2.11}$$

2.3.1 Value Function and Q Function

For some policy π a value function [15, 16, 7] is defined $V : S \to \mathbb{R}$

$$V_t^\pi(x_t) = r_t(x_t, \pi_t(x_t)) + EV_{t+1}^\pi(x_{t+1}|x_t, \pi(x_t)) \tag{2.12}$$

hence our objective is to find a policy π^* such that

$$V_t^{\pi^*}(x_t) \geqslant V_t^\pi(x_t); \forall \pi \in \Pi$$

where Π is set of all feasible policies. Similarly Q function [15, 16, 6] is defined $Q : S \times A \to \mathbb{R}$ due to as value of taking action a and following the optimal policy thereafter

$$Q_t(x_t, a_t) = r(x_t, a_t) + \gamma EV_{t+1}^{\pi^*}(x_{t+1}) \tag{2.13}$$

As it can be seen it does involve expectation calculation which is difficult even when the distribution is known. Using the Q function optimal action by chosen such that Q function is maximized

$$a_t^* = \max_{a_t \in A_t} Q(s_t, a_t); t \in \{0, 1, ..., T-1\}$$

2.3.2 Value Iteration

Consider an infinite horizon control problem. Backward dynamic programming techniques can not be used, as we do not have a horizon and terminal reward function defined. But we can still use the optimality equation in the same way as in backward case, following is the value iteration [16, 15] algorithm for infinite horizon

As can be observed this is uses same optimally equation used in backward dynamic programming, but it starts with an some initial values of all states and then using optimality equation it calculates it approximates the value function. For small state and action space it may produce optimal value too. This is example of forward dynamic programming where we start with initial value function and update the value function based on optimality equation.

Algorithm 1 Value Interation

procedure VALUEITERATION(state s)
Initialization:
    set $V^0(x) = 0; \forall x \in S$.
    set a tolerance parameter $\eta > 0$.
    set $N = 0$.
repeat:
    for $x \in S$ **do**
        $V^{n+1} = max_{a \in A}[r(x,a) + \gamma \sum_{x' \in S} Pr(x'|x,a)V^n(x')$.
    let a^{n+1}*be vector that solves above*
    if $||v^{n+1} - v^n|| < \eta(1-\gamma)/2\gamma$ **then**
        set $a^\eta = a^{n+1}$
        $V^\eta = V^{n+1}$ **return** a^η, V^η
    else
        set $N = N + 1$ **goto** *repeat*

Chapter 3

Problem Formulation and Proposed Solution

3.1 Introduction

An order book is the list of orders (manual or electronic) that a trading venue (in particular stock exchanges) uses to record the interest of buyers and sellers in a particular financial instrument. A matching engine uses the book to determine which orders can be fulfilled i.e. what trades can be made [21]. Assume a large investor who owns significantly large amount of stocks of a particular security, wants to liquidate his portfolio. He wants to sell off his whole portfolio. We consider the perfect and complete market. When he places a market order an entry in market order book is made. This information in order book is public to every investor/broker. This may lead to adverse impact on price. There have already been large number of studies on price impact of market such as [1]. Considering the price impact of market, following can be thought as the market dynamics

$$p_{t+1} = f_t(p_t, x_t, \epsilon_t) \tag{3.1}$$

where p_t is the price security, x_t is the amount of order to sell and ϵ_t is the random noise. More conveniently we can write the impact as

$$\begin{aligned} p_{t+1} &= f_t(p_t, x_t) + g_t(\epsilon_t) \\ p_{t+1} &= p_t^* + g_t(\epsilon_t) \end{aligned} \tag{3.2}$$

where f_t can be regarded as order size impact of trader and g_t as impact due to noise traders, p_t^* is the impact price at which the trade takes place. Considering the equation (3.2) as dynamics of the market, the liquidation in market order can be formulated as an optimization problem below

$$\begin{aligned} &\max_{x_0, x_1, \ldots . x_T} E \sum_{t=0}^{T} p_t^* \dot{x}_t \\ &s.t.\ x_0 + x_1 + x_2 + ... + x_T = X_0 \end{aligned} \tag{3.3}$$

where X_0 is the initial amount available to liquidate. The equation (3.3) is a dynamic optimization problem with price as state variable and amount of stocks to trade as action

variable, but with a constraint, i.e. equation (3.3) is a Markov decision problem with a constraint. We can convert it into an unconstrained optimization problem by introducing new state variables as a two dimensional sets $s_t = \{p_t, X_t\}$ and with upper bound on action x_t as X_t, as described below

$$\max_{x_0, x_1, \ldots . x_T} E \sum_{t=0}^{T} c_t(s_t, x_t) \tag{3.4}$$

with $X_{t+1} = X_t - x_t$ where action space is now bounded as $x_t \in \{0, 1, , , .X_t\}; \forall t \in 0, 1, .., T$.

3.2 Dark Pool

As discussed in previous section the impact of the trade size may adversely affect the price of the security and hence, one has to put the orders in such a way that the impact is minimum. Portfolio liquidating is a process which an investor wants to do in a limited period of time, he may have to compromise with the lower return. Dark pools are exchange markets working privately that allows a trader to put their orders, but not all the information about orders or trdes are made public , hence called *dark pool* [3, 5]. Only way to know the market depth and other information in dark pool is to put orders in dark pool.

Even for an investor putting order in dark pool reveals only small information [3]. There are no publicly available order books as in case of market orders. If an investor let's say a seller puts his order of size z_t at time t, let's say y_t amount is available to buy from buyer side. After trade execution let's say his d_t amount got executed where $d_t \leqslant y_t$, information available to investor is only $min(x_t, y_t)$ as he can only observe d_t. Only information he can learn is the fraction of his order being executed. There have been studies for optimal orders placements in dark pools so that the execution of the trades are maximized such as in [9]. To our best knowledge the study of closest to our work has been considered in [14], where no partial execution in dark pool has been considered, and the problem has been formulated with whole history of orders making impact on prices. Here in this thesis we assume only Markov property that is the price depends only on the previous order size and previous price along with the random component.

As the price in dark pool are mostly derived from the conventional exchange price such as mid price of NBBO [3], it can be said that the order size impact on the price in the dark pool is near to negligible. It seems conclusive that investor should place all his order in dark pool, rather than as market orders. But one thing to note about the dark pool are that there are no guarantee that the orders will be executed in the dark pool. Hence it is suitable to put the orders both in dark pool and conventional exchange market order. In the next section we formulate the problem as Markov decision process and then move on to finding the way to solve the same.

3.3 Problem Formulation

Let us assume

$$\begin{aligned} X_t &= \textit{stocks available at time } t \textit{ to liquidate} \\ x_t &= \textit{amount of order to be placed in primary exchange at time } t \\ y_t &= \textit{number of order to be placed in dark pool at time } t \\ z_t &= \textit{number of stocks executed in dark pool at time } t \\ d_t &= \textit{price of stock in dark pool at time } t \\ p_t &= \textit{price of stock in primary market at time } t \end{aligned} \tag{3.5}$$

Investor by selling his stocks in primary exchange and dark pool gets liquidity as below

$$r(p_t, d_t, x_t, y_t) = p_t^*.x_t + d_t^*.z_t \tag{3.6}$$

considering above as reward function we can formulate it as an optimization problem below

$$\max_{x_t, y_t} E \sum_{t=0}^{T} r(p_t, d_t, x_t, y_t) \tag{3.7}$$

subject to following dynamics of market

$$\begin{aligned} p_{t+1} &= f(p_t, x_t, \epsilon_t) \\ &= h(p_t, x_t) + e(p_t, \epsilon_t) \\ &= p_t^* + e(p_t, \epsilon_t) \\ d_{t+1} &= g(d_t, p_t, y_t, \eta_t) \\ &= o(d_t, p_t, y_t) + m(d_t, \eta_t) \\ &= d_t^* + m(d_t, \eta_t) \end{aligned} \tag{3.8}$$

and constraint

$$\sum_{t=0}^{T} (x_t + z_t) = X_0 \tag{3.9}$$

Where ϵ_t and η_t are random variables. And we can formulate following relationship

$$X_{t+1} = X_t - (x_t + z_t)$$

By similar method as in section 3.1 we can turn this problem into unconstrained problem.

3.4 Backward Dynamic Programming Approach

Let us assume for case of market orders only, as we have discussed we will take state of the system as vector $s_t = \{p_t, X_t\}$ i.e. price and available liquidation amount of stock at time t. Following is the backward dynamic programming algorithm for the same. Assuming the following algorithm, as we can see there are nested for loops and also expectation calculation.

Lets assume that we have some fixed maximum price P and we have discrete probability transition given, and for terminal case

$$\begin{aligned} r_T(s_T) &= p_T^* X_T \\ s_T &= \{p_T, X_T\} \\ a_T &= \{X_T\} \end{aligned} \tag{3.10}$$

Algorithm 2 Backward DP

```
1: procedure BACKDP
2:     Calculate the terminal liquidity and initialize terminal action
3:     V_T(s_T) = r_T(s_T); ∀s_T
4:     A_T(s_T) = X_T; ∀s_T
5:     for t = T − 1 to 0 do
6:         for p_t ∈ {0, 1, ...P} do
7:             for X_t ∈ {0, 1, .., X_0} do
8:                 s_t = (p_t, X_t)
9:                 maximum = 0
10:                action = 0
11:                for a_t ∈ {0, 1, ...X_t} do
12:                    value = r_t(s_t, a_t) + EV_{t+1}(s_{t+1})
13:                    if value > maximum then:
14:                        value = maximum
15:                        action = a_t
16:                    V_t(s_t) = value
17:                    A_t(s_t) = action
```

then based on this we can see that the algorithm is $O(tP^2X_0^2)$ in worst case. Along with the space complexity of $O(tPX_0)$. Consider for an example investor has initially 10000 stocks and maximum price is \$100 and he wanted to liquidate in 10 period of time then we have time complexity of $O(10^{13})$ and space complexity $O(10^7)$ which is not feasible. One way to reduce this is to consider since we are considering a large investor, he will be placing order in blocks and also the possible price transition is limited. Using these let us assume he is placing orders in block size of 100, and only 10 transition is possible for price, hence we have time complexity $O(10^8)$ and space complexity $O(10^5)$, this seems to be feasible. But we can observe from the discussion that as the backward dynamic programming technique is good for only small size problems. This is termed as curse of dimensionality in dynamic programming. This shows that if we consider this for dark pool problem, as the dimnsonality of state and action vector is even more, it will not be feasible to apply this. Along with this we have assumed the prices to be discrete which is not realistic.

In the next section we will look into approximation based dynamic programming algorithms and will apply to the dark pool problem.

3.5 Approximate Dynamic Programming

We have seen that the backward dynamic programming approach is not feasible in case of problems with large state and action space. The obvious way to proceed is to step forward through time. But to step forward in time we need the value function available for the future time, which is not possible. We can consider the approximation of the value function. Let us assume the approximation is given as :

$$\hat{V}_t^n(s_t) = \textit{approximate value function at time } t \textit{ for state } s_t \textit{ at iteration } n$$

Now this value function can be used to update the value function at iteration $n+1$ by the help of information gathered in iteration n. To find the optimal action in iteration n we use the Bellman equation

$$a_t^n = arg\max_{a \in A_t}\{r_t(s_t, a) + E\hat{V}_{t+1}^{n-1}(s_{t+1})\} \tag{3.11}$$

where expectation is on the transition probability of s_t.

As calculation of the expectation can be intractable, we can generate samples and calculate the sample based expectation. Below is the generic forward dynamic forward dynamic programming algorithm due to [15, 4]

Algorithm 3 Forward DP

procedure FORWARDDP
  For all t initialize value function $\hat{V}_t^0(s_t) \forall s \in S, N$
  $n \leftarrow 1$
repeat
  for $t = 0$ *to* $T-1$ **do**
    generate sample Ω_t
    solve :
    $a_t^n = arg\max_{a \in A_t}\{r_t(s_t, a) + \sum_{\omega_t \in \Omega_t} Pr(\omega_t)\hat{V}_{t+1}^{n-1}(f_t(s_t, a, \omega_t))\}$
    choose $\omega_t \in \Omega_t$
    compute $s_{t+1} = f(s_t, a_t, \omega_t)$
    update the approximation $\hat{V}_t^n(s_t)$
    $t = t + 1$
  $n = n + 1$
  if $n < N$ **then**:
    goto repeat
  else:
    stop

There are many methods that can be used to update the estimate of value function. One way is to compute

$$\hat{v}_t^n = \{r_t(s_t, a) + \sum_{\omega_t \in \Omega_t} Pr(\omega_t)\hat{V}_{t+1}^{n-1}(f_t(s_t, a, \omega_t))\} \tag{3.12}$$

which is estimate of the value function of the state s_t, which obviously depends on the our approximation at iteration $n-1$, since the estimate $\hat{v}_t^n$ may have noise, we can smooth the value function based on

$$\hat{V}_t^n(s_t) = (1-\alpha^n)V_t^{n-1}(s_t) + \alpha^n \hat{v}_t^n \tag{3.13}$$

where α is called step size parameter, takes value between 0 and 1. We can see that the above algorithm has assumption that the state space is finite and also the action space is finite. But when we do not have finite space such that the price in the market can not be regarded as discrete then we can not use the above mentioned algorithm directly. But what we can do is use some kind of function approximation such as a linear function with some parameters, or neural network [6, 15, 16]. This helps us to represent the value function with limited number of parameters and based on the state we can evaluate the value function. For example below is a linear function approximation based on state of system

$$\hat{V}_t(s_t) = \theta_1\phi_1(s_t) + \theta_2\phi_2(s_t) + ... + \theta_m\phi_m(s_t) \tag{3.14}$$

where θ_i are parameters and $\phi_i(s_t)$ are features derived from state vector s_t. Now all we need to do is find way to update the parameters $\theta_1, \theta_2, .., \theta_m$, rather than storing the value function for each state we need to store only parameters.

3.5.1 Actor Critic Method

We can group majority of approximate dynamic programming methods also called reinforcement learning in computer science community into actor only, critic only and actor-critic models [13]. In real life cases where action as well as state space is large it is not feasible to store the value function or policies itself for each state and state action pair, hence an approximation function has to be used for value such as described in above section. We describe the critic only and actor only method in following paragraphs and then move to actor critic method.

Critic only methods e.g. Q-learning [23], SARSA [18] use function approximaters for state-action function i.e. Q functions as described in previous chapter. There are no expilicit function used for policy. Deterministic policy $\pi = \{\pi_0, ..., \pi_{T-1}\}$ is calculated from approximated Q function by solving the optimization problem over value function.

$$\pi_t(s_t) = arg\max_{a \in A_t} Q(s_t, a) \tag{3.15}$$

When learning is in online setting there have been examples shown for simple MDP such as [19, 10, 2] that the methods such as Q learning and SARSA do not converge with specific function approximaters. However divergence was further analysed by [20] and shown that the convergence can be assured for linear in parameters if the samples are drawn accordingly.

Actor only methods do not store the value function or parametric value function, however the policy itself is parametric and the optimize the reward based on Bellman equation in the space of parameters of policy function itself. Examples of the actor only methods are SRV algorithm [11], REINFORCE algorithm [22]. The main advantage of actor only method

is their convergence property, convergence are obtained if estimated gradient are unbiased and step size or learning rate satisfy Robbins-Munro condition [17]

$$\alpha_k > 0 \quad \forall k \tag{3.16}$$

and

$$\sum_{k=0}^{\infty} \alpha_k = \infty \qquad \sum_{k=0}^{\infty} \alpha_k^2 < \infty \tag{3.17}$$

Drawback of actor only methods are that they may have large variance in estimated gradient.

The actor critic method combines the benefits of both actor only and critic only method, they are able to produce the continuous actions and hence useful for large action space, and large variance in gradient estimate is reduced by critic presence. Consider the following setting where we are approximating the value function and policies as follows

$$\hat{V}_t(s_t) = \theta_t^T \phi(s_t) \tag{3.18}$$

and

$$\pi_t(s_t) = \omega_t^T \psi(s_t) \tag{3.19}$$

where

$$\phi(s_t) = \{\phi_{1_t}, \phi_{2_t}, ..., \phi_{m_t}\}$$

and

$$\psi(s_t) = \{\psi_{1_t}, \psi_{2_t}, ..., \psi_{m_t}\}$$

are vecor of features based on state s_t also the parameters are considered to be time dependent are

$$\omega_t = \{\omega_{1_t}, \omega_{2_t}, ..., \omega_{m_t}\}$$

and

$$\theta_t = \{\theta_{1_t}, \theta_{2_t}, ..., \theta_{m_t}\}$$

Consider the Bellman equation 2.12

$$V_t^{\pi}(x_t) = r_t(x_t, \pi_t(x_t)) + EV_{t+1}^{\pi}(x_{t+1}|x_t, \pi(x_t))$$

with approximation under a policy π we can rewrite the above equation as

$$\hat{V}_t^{\pi}(s_t) = \{r_t(s_t, \pi_t(s_t)) + E\hat{V}_{t+1}^{\pi}(s_{t+1})\} \tag{3.20}$$

The difference between right hand and left hand side is known as temporal difference(TD) or Bellman error. This error can be used to obtain the gradient following ways

$$\begin{aligned} \delta_t &= \{r_t(s_t, \pi_t(s_t)) + E\hat{V}_{t+1}^{\pi}(s_{t+1})\} - \hat{V}_t^{\pi}(s_t)\} \\ &= \{r_t(s_t, \omega_t^T \psi_t(s_t)) + E\theta_{t+1}^T \phi(s_{t+1}) - \theta_t^T \phi(s_t)\} \end{aligned} \tag{3.21}$$

From the above Bellman error we can easily calculate the gradient of mean square error and update of the parameters of value function can be obtained as follows in case of linear case

$$\theta_t \leftarrow \theta_t + \alpha_\theta \delta \phi_t(s_t) \tag{3.22}$$

similary we can obtain the update for the policy parameters as follows

$$\omega_t \leftarrow \omega_t + \alpha_\omega \delta \psi_t(s_t) \tag{3.23}$$

Eligibility traces are temporary record of occurrence of an even, i.e. system state visited through trajectory which can be defined as

$$e_t(s) = \begin{cases} \lambda e_{t-1}(s) & : if\ s \neq s_t \\ \lambda e_{t-1}(s) + 1 & : if\ s = s_t \end{cases} \tag{3.24}$$

where λ is discounting parameter for history of the state. The above eligibility traces can be generalized for continuous spaces as follows

$$z_t = \begin{cases} 0 & : if\ t = 0 \\ 1 & : if\ t = T - 1 \\ \phi_t(s_t) + \lambda z_{t-1} & : otherwise \end{cases} \tag{3.25}$$

observe that z is now vector of the same dimension as features of value function. Combining all these together we can write the actor-critic algorithm

3.6 Actor Critic method for Dark Pool Problem

We consider the following assumptions

- The impact function of price is known for conventional exchange
- Prices in the dark pool are derived from primary exchange prices
- We consider negligible impact of the trade size in the dark pool.
- We consider that no other fees is charged for order placement in either dark pool of conventional exchange

In addition to above assumptions we normalize the prices to be any real number in $[0, 1]$ also we take stocks available as real number in $[0, 1]$. We trade as fraction of available amount in dark as well as conventional market. Following are the dynamics of the market

- $p_t = f(p_{t-1}, x_{t-1}) + \epsilon_t$ where we have assumed $\epsilon_t \overset{iid}{\sim} \mathcal{N}(\mu = 0, \sigma^2 = 0.001)$
- $d_t^* = p_t + \eta_t$ i.e. no impact of order size in dark pool, however random component $\eta_t \overset{iid}{\sim} \mathcal{N}(\mu = 0, \sigma^2 = 0.001)$ is added
- $y_t \sim Gumbell(\mu = y_{t-1}, \beta = 0.02)$ or $y_t = 0$ with equal probability, where y_t is the fraction of dark pool order executed and initial value y_0 is assumed to be 0
- state of the system is given as $s_t = \{p_t, X_t, y_{t-1}\}$, where p_t is the price in the conventional market, X_t is amount left to liquidate,

Algorithm 4 Actor Critic Algorithm

procedure ACTORCRITIC
initialize
    $\theta_t, \omega_t \forall t \in \{0, 1, ...T-1\}, N$
    $n \leftarrow 1$
repeat
    generate initial state randomly s_0
    initialize eligibility traces $z = 0$
    for $t = 0$ *to* T **do**
        generate sample Ω_t
        calculate TD error : δ_t
        select action based on current parameters : $a_t = \omega_t^T \psi_t(s_t)$
        generate random exploration variable : e_t
        $a_t \leftarrow a_t + e_t$
        if t = T-1 **then**:
            $\delta_t = r_t(s_t, a_t) + \sum_{o_t \in \Omega_t} Pr(o_t) r_T(f_T(s_t, a_t, o_t)) - \theta_t^T \phi_t(s_t)$
            $z = 1$
        else
            $\delta_t = r_t(s_t, a_t) + \sum_{o_t \in \Omega_t} Pr(o_t) \theta_{t+1}^T \phi_{t+1}((f_t(s_t, a_t, o_t))) - \theta_t^T \phi_t(s_t)$
            $z = \lambda z + \phi_t(s_t)$
        $\theta_t \leftarrow \theta_t + \alpha_\theta \delta_t z$
        $\omega_t \leftarrow \omega_t + e_t \alpha_\omega \delta_t \psi_t(s_t)$
        compute $s_{t+1} = f(s_t, a_t, o_t)$
        $t = t + 1$
    $n = n + 1$
    if $n < N$ **then**:
        goto repeat
    else:
        stop

We consider following set of features based on state vector for value function approximation

$$\phi(s_t) = \{X_t p_t, (y_{t-1}) p_t, X_t^2, p_t^2, y_{t-1}\} \tag{3.26}$$

Similarly we consider the feature vectors for dark pool policy and conventional market policy as follows

$$\begin{aligned} \psi_d(s_t) &= \{\sqrt{p}_t X_t, y_{t-1}, X_t\} \\ \psi_m(s_t) &= \{\sqrt{p}_t X_t, p_t, X_t^2\} \end{aligned} \tag{3.27}$$

With above features we take the policy and value function parameters given as below

$$\begin{aligned} \theta_t &= \{\theta_{t_1}, \theta_{t_2}, \theta_{t_3}, \theta_{t_4}, \theta_{t_5}\} \\ \omega_t^d &= \{\omega_{t_1}^d, \omega_{t_2}^d, \omega_{t_3}^d\} \\ \omega_t^m &= \{\omega_{t_1}^m, \omega_{t_2}^m, \omega_{t_3}^m\} \end{aligned} \tag{3.28}$$

the reward function is given as

$$\begin{aligned} r_t(s_t, x_t, z_t) &= p_t^* x_t + d_t^* z_t y_t \\ r_T(s_T) &= p_t^* X_T \end{aligned} \tag{3.29}$$

Policy function are considered as sigmoid function, such that they give the fraction of current available liquidation amount as next order to be placed. Using above dynamics and assumptions we have applied the actor critic learning method described in previous section.

Chapter 4

Simulation and Analysis

In this chapter we analyze results obtained on applying the actor critic method to liquidation problem. First we will discuss the simulation of market price, then we will show empirical result of convergence of temporal difference error. The value function and policy obtained is discussed. Finally, we will show empirically that the policy obtained with our approach is better than naive policy of equal block market orders for each time period.

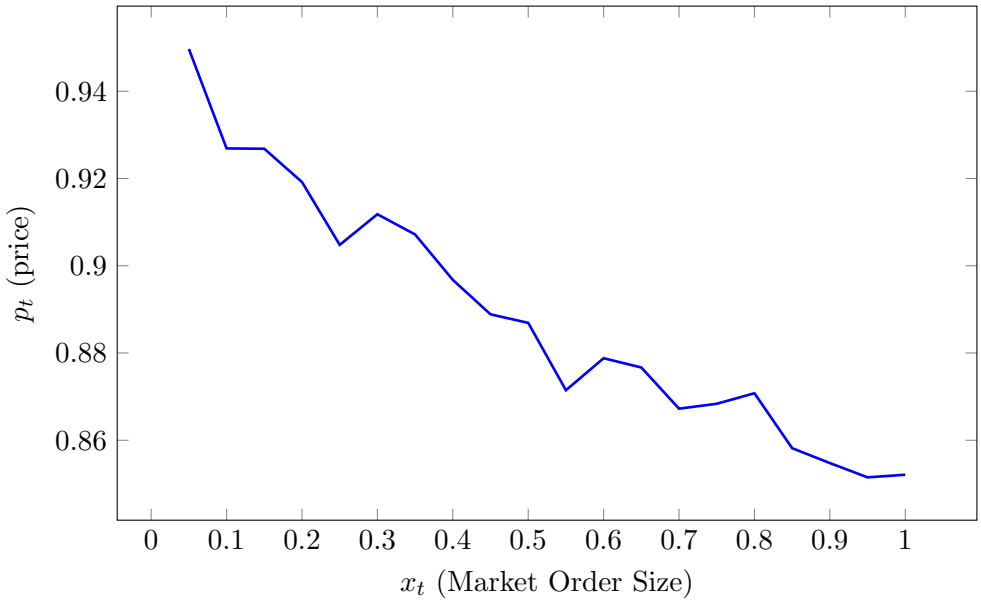

Figure 4.1: Sample Dynamics of Market Price

We have considered finite horizon of 20 time periods. Based on the assumptions and market dynamics discussed in section 3.6, the market price dynamics is shown in figure 4.1 based equation below,

$$p_{t+1} = p_t(1 - k\sqrt{\frac{x_t}{p_t}}) + \epsilon_t \tag{4.1}$$

where k is a constant and $\epsilon_t \overset{iid}{\sim} \mathcal{N}(\mu = 0, \sigma^2 = 0.001)$.

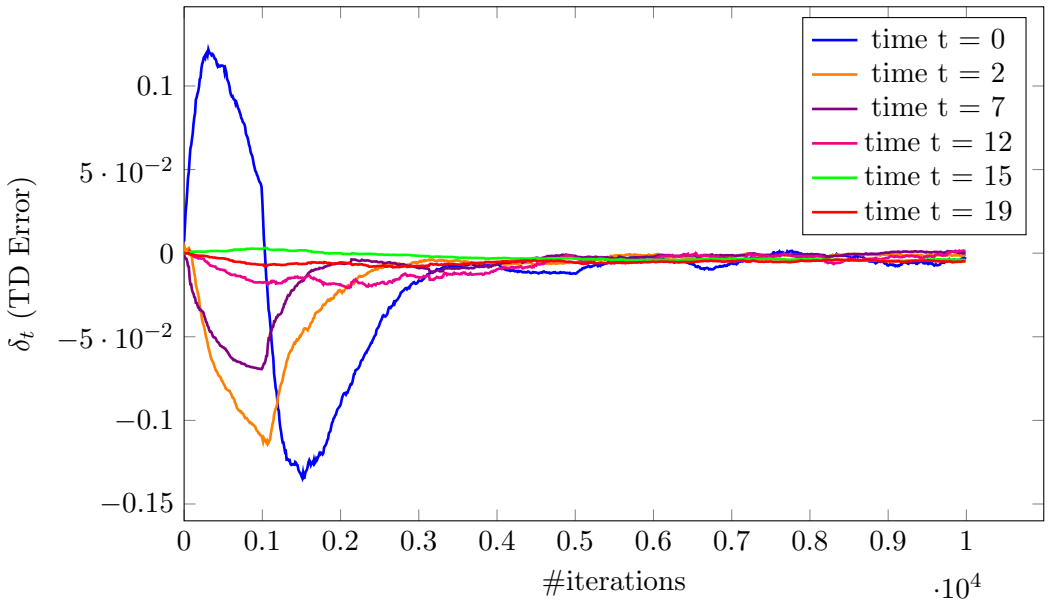

Figure 4.2: Convergence of Temporal Difference Error

We have used on-policy actor critic method, it learns using reward generated from each interaction with the environment. In our case for each period of time it updates the parameters of the policy and value function based on the liquidation amount received. As discussed in section 3.5.1, temporal difference error is used to find the gradient. The convergence of the TD error is desired for learning optimal policy. Figure 4.2, shows the convergence of the temporal difference(δ) error. It can be seen that initially error starts increasing and then as the number of iteration increases TD error starts to converge.

Figure 4.3, shows the value function based on the policy obtained after TD error convergence as discussed above. Samples of value function corresponding to the initial market price have been drawn. Following observation can be drawn from the figure,

- The value function is sum of liquidation returns obtained, it should decrease as time proceeds, we can see that value function is monotonically decreasing.
- For higher initial price of a security, liquidation return should be higher . The graph shows value function has higher values of higher initial price.
- Value function near horizon is close to zero, which indicates as time proceeds lesser stocks are left to liquidate.

Now we discuss the policy obtained for dark pool and market orders. Figure 4.4 shows the policies for dark pool and market order. It can be seen that the size of the market order is most of the time more than the dark pool order, as in dark pool order execution is not guaranteed. We can see that as time proceeds towards horizon, the order size starts increasing. It indicates that policy trades more aggressively as time proceeds, so that whole portfolio is liquidated before the horizon.

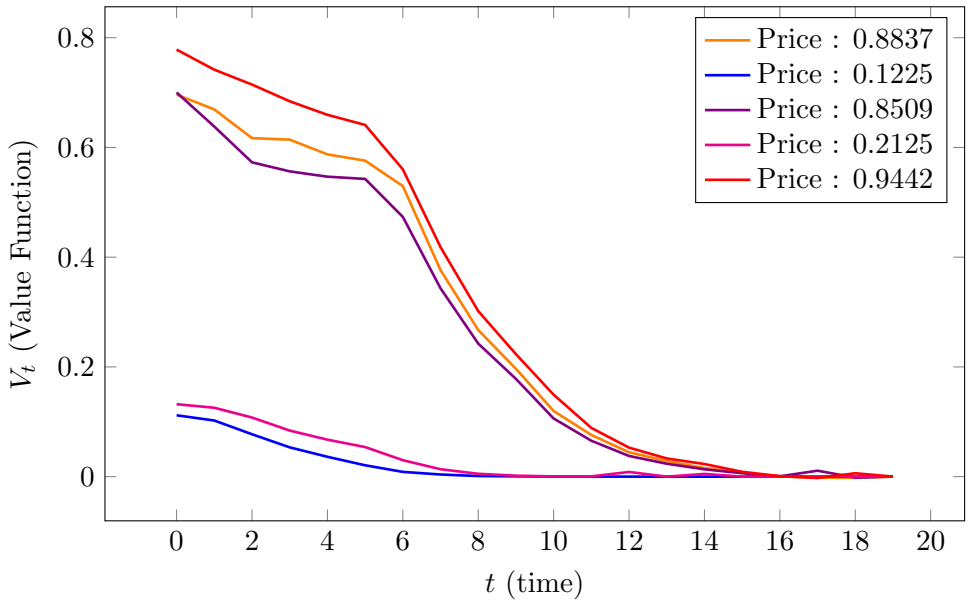

Figure 4.3: Value Function For Different Initial Prices

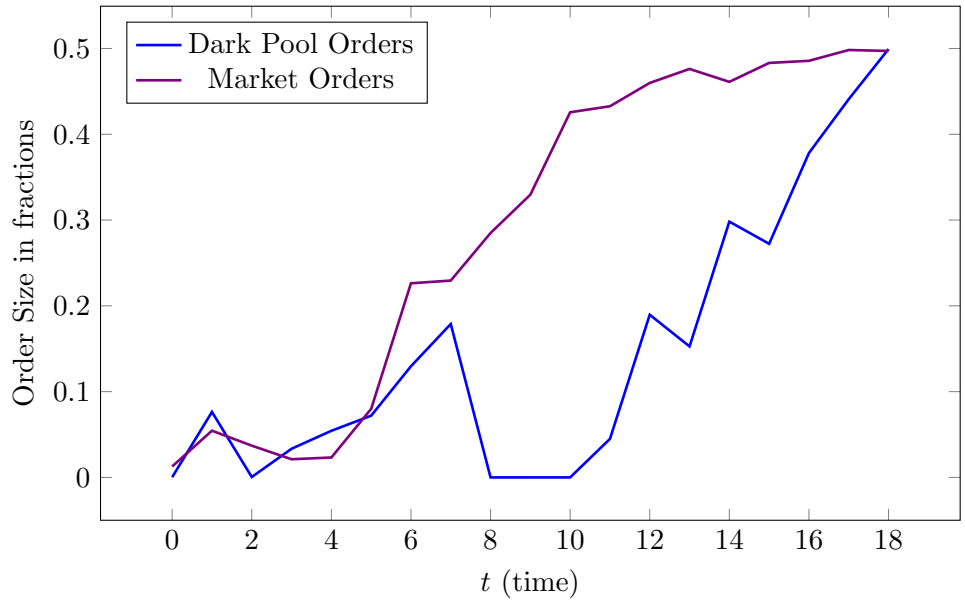

Figure 4.4: Comparison for Market Order and Dark Pool Order

We show that empirically that the liquidation return obtained using actor critic method is better than the return based on equal block market orders. We have generated the samples returns based on the dynamics of the market discussed for equal block size market orders and policy obtained using actor critic method. The difference between liquidation return obtained using both approach is calculated. Null hypothesis is that mean difference between both approach is 0. Results of one sample t-test for different sample sizes, are

Sample Size	Mean Difference	t-statistic	p-value
100	0.0280837810258	0.83388953147427203	0.40535119271146636
1000	0.020419467299	2.0931508570575095	0.036461619930250787
5000	0.0219556654136	4.9122884383925438	9.1447540739443554e-07
10000	0.0219220529495	6.9557773292088916	3.6146034055307383e-12

Table 4.1: Result of t-test

shown in the table 4.1.

For sample size 1000 and above the p value is small (< 0.05), null hypothesis can be rejected. The mean difference calculated shows that the return obtained using actor critic method is better than the naive approach of splitting as equal block market orders. Hence we can say, our approach produces better liquidation returns.

Chapter 5

Future Work and Conclusion

A finite horizon control problem poses additional complexity to the control problem. One has to find a non stationary policy, as the state and action space both changes with the time. Finding stationary policies in case of infinite horizon problem is relatively easy and there have been convergence proof for many methods for stationary policy.

We have applied the finite horizon based actor critic learning method here. We have explicitly put the parameters for each time period to be separate such that we can get the non stationary policies. We have shown convergence of TD error empirically. We have also shown the comparison of splitting equal block market orders and our approach. It was shown empirically that the approach we used produces higher liquidation return.

Because of the non stationarity of the policies in finite horizon problem, non convergence of algorithms have been shown for some problems.Hence a formal proof of convergence is required. As approximation has been used, an error bound on the policies obtained has to be theoretically obtained. These are some future directions on this problem and on finite horizon control problems as whole.

Bibliography

[1] Almgren, R., Chriss, N.: Optimal execution of portfolio transactions. Journal of Risk pp. 5–39 (2001)

[2] Baird, L.: Residual algorithms: Reinforcement learning with function approximation. In: In Proceedings of the Twelfth International Conference on Machine Learning. pp. 30–37. Morgan Kaufmann (1995)

[3] Banks, E.: Introduction to Dark Pools, pp. 3–32. Palgrave Macmillan UK, London (2014), http://dx.doi.org/10.1057/9781137449573_1

[4] Bellman, R.: Dynamic programming. Princeton Univ Pr (1957)

[5] Buti, S., Rindi, B., Werner, I.M.: Diving into dark pools. Working paper series, Ohio State University, Charles A. Dice Center for Research in Financial Economics (2010), http://EconPapers.repec.org/RePEc:ecl:ohidic:2010-10

[6] Dimitri P. Bertsekas, J.N.T.: Neuro-Dynamic Programming. Optimization and neural computation series, Athena Scientific, 1 edn. (1996)

[7] D.P, B.: Dynamic programming and optimal control, vol. 2. Athena Scientific (1995)

[8] (Eds.), D.P.B.: Dynamic Programming and Stochastic Control. Mathematics in Science and Engineering 125, Academic Press (1976)

[9] Ganchev, Kuzman; Nevmyvaka, Y.K.M.V.J.W.: Censored exploration and the dark pool problem. Communications of the ACM 53 (05 2010)

[10] Gordon, G.J.: Stable function approximation in dynamic programming. Tech. rep., Carnegie Mellon University, Pittsburgh, PA, USA (1995)

[11] Gullapalli, V.: A stochastic algorithm for learning real-valued functions via reinforcement. Tech. rep., University of Massachusetts, Amherst, MA, USA (1988)

[12] Keim, D.B., Madhavan, A.: The Upstairs Market for Large-Block Transactions: Analysis and Measurement of Price Effects (Revised: 10-94). Rodney L. White Center for Financial Research Working Papers 21-92, Wharton School Rodney L. White Center for Financial Research (undated), https://ideas.repec.org/p/fth/pennfi/21-92.html

[13] Konda, V.R., Tsitsiklis, J.N.: Convergence rate of linear two-time-scale stochastic approximation. The Annals of Applied Probability 14 (05 2004)

[14] Kratz, P., Schöneborn, T.: Portfolio liquidation in dark pools in continuous time. Mathematical Finance 25(3), 496–544 (2015), http://EconPapers.repec.org/RePEc:bla:mathfi:v:25:y:2015:i:3:p:496-544

[15] Powell, W.B.: Approximate dynamic programming: Solving the curses of dimensionality. Wiley Series in Probability and Statistics, Wiley-Interscience, 1 edn. (2007)

[16] Richard S. Sutton, A.G.B.: Reinforcement learning. Adaptive Computation and Machine Learning, The MIT Press (1998)

[17] Robbins, H., Monro, S.: A stochastic approximation method. The Annals of Mathematical Statistics 22(3), 400–407 (1951), http://www.jstor.org/stable/2236626

[18] Sutton, R.S.: Generalization in reinforcement learning: Successful examples using sparse coarse coding. In: Advances in Neural Information Processing Systems 8. pp. 1038–1044. MIT Press (1996)

[19] Tsitsiklis, J.N., Roy, B.V.: An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control 42(5), 674–690 (May 1997)

[20] Tsitsiklis, J.N., Roy, B.V.: Average cost temporal-difference learning. In: Proceedings of the 36th IEEE Conference on Decision and Control. vol. 1, pp. 498–502 vol.1 (Dec 1997)

[21] Wikipedia: Order book (trading) — wikipedia, the free encyclopedia (2016), https://en.wikipedia.org/w/index.php?title=Order_book_(trading)&oldid=725209547, [Online; accessed 5-July-2017]

[22] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8(3), 229–256 (May 1992), http://dx.doi.org/10.1007/BF00992696

[23] Wyatt, J.: Exploration and Inference in Learning from Reinforcement. Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh (1997)