A THESIS ON ALGORITHMIC TRADING

BY

YINGJIE YU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Industrial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Associate Professor Liming Feng

Abstract

Algorithmic trading is one of the most phenomenal changes in the financial industry in the past decade. While the impacts are significant, the microstructure of algorithmic trading remains unknown.By using Diff-in-Diff analysis, this paper shows that for low price securities, algorithmic trading activities are more active than high price securities. Besides, algorithm trading *per se* may also trigger significant price impact. As a result, algorithmic order execution has to be dynamically adapted to real-time market environments. This makes *dynamic programming* (DP) the most natural approach. This paper builds a optimal order execution model using dynamic programming. It works with the mean-variance utilities of Almgren and Chriss (*J. Risk*, **3**, 2000) to effectively express risk aversion of a typical trader. The new framework is demonstrated through building one particular style called MV-MVP, i.e., the *mean-variance* (MV) objective formulated upon the state variables of *moneyness* and *volume participation* (MVP). The MV-MVP style generalizes the VWAP strategy by facilitating dynamic reactions to moneyness and by embodying the popular street practice of trading aggressively or passively while in the money. Simulated dynamic trading paths illustrates the MV-MVP style oscillates around the VWAP strategy.

To my parents

Acknowledgments

Thanks to my advisers Prof. Jackie Shen and Prof. Liming Feng for their unconditional support in my research.

Table of Contents

List of Tables

List of Figures

Chapter 1

An Introduction to Algorithmic Trading

1.1 Background of Algorithmic Trading

Algorithmic trading is one of the most phenomenal changes in the financial industry in the past decade. As one stream of the technology waves that reshaped the landscape of financial markets, algorithmic trading has two key features: First, fully automated and algorithmic based trading, while humans behind the scene were responsible for designing the algorithms for automation, they themselves were less involved with the direct trading due to the inferior of human reaction speed compare with the computers. Second, the trading speed has come all the way through from second to microsecond and now reaches nanosecond scale. The statistics showed that the computer based trading has taken over more than 75 percent shares in the U.S stock market and even higher percentage in the derivative market(10).

Although many participants are optimistic about the technology breakthrough to provide liquidity for a more efficient market, noticeable social problems emerged alone with the improvement as well. For example, the 2010 Flash Crash, an event that the Dow Johns Industrial Average plunged about 1000 points (around 9%) within 5 minute (21), was the biggest intra-day decline in the history. The plunge caused widespread fears, which led the stock market being frozen temporally. Another example was the sudden crash-down of Knight Capital, one of the largest algorithmic trading firms in the world, which lost 460 millions in 20 minutes due to its mystery algorithmic error in August 6th 2012. Both of the two cases caused huge market volatility and short-term dysfunction. While the causes remained controversial, many people believe the algorithmic trading was the key trigger for the disaster and thought the human rationale would fail to control the risks under the power of automated machines. While the impacts are significant, the phenomenon has few

precedents to follow. In the new land of algorithmic trading, it is crucial to adapt the research to understand what is happening in the dynamically changing market in order to avoid some possible market risks.

1.2 Main Challenges

The challenges confront the researchers and regulators are from three major aspects:
First, the algorithms for algorithmic trading are the private proprietaries belongs to each individual firms, that made it almost impossible for researchers to access, which causes the difficulty to accurately evaluate how various algorithms acts as a whole would lead to abnormal market behaviors.

Second, the technological discrepancy between the industry and academia is getting wider. While the arm race keeps intensive, the academia could hardly catch-up to fully understand and measure the possible effect of all the latest technologies. For example, the arm race has gone beyond the network layer to the physical layer for faster information communication between the data centers in Chicago and New York, some company has started building infrastructures to allow laser for information transformation (18), the technology once US military jet used to communicate with military base.

Third, the data, or the record of trading activities has grown in an exponential rate. For example, the average stock quotes data across the nation is on average Terabyte scale. All those challenges require new framework and much more powerful facilities to calibrate the fundamental changes in the market.

1.3 Main Contribution

The goal of this paper is to better understand the microstructure of algorithmic trading. It is not an easy task to interpret what has been happening in the vigorously changing trading market.

This paper is among the very first to apply cyber-infrastructure and domain knowledge in engineering into finance field. I have discarded old-fashioned research framework due to the

incapability of dealing with massive data. Some of the cutting edge equipment and technologies including supercomputers and parallel computing enable me to investigate among terabytes-scale data, and to run instructions in parallel to discover the ideas that have not been investigated thoroughly before.

I am aiming to answer some of the most concerning questions: what is the cross-sectional variation in algorithmic trading firm's behavior? Does the algorithmic trading bring some impact to the market? What is the optimal strategy for an algorithmic trader after taking into account the price impact?

This paper is organized as follows. Section 2 discusses empirical evidence showing the cross-sectional variation in algorithmic liquidity provision activities. Section 3 builds a dynamic model for optimal algorithmic trading behaviour.

Chapter 2

Empirical Evidence on Algorithmic Trading

2.1 Introduction

> The Sub-Penny Rule (adopted Rule 612 under Regulation NMS) prohibits market participants from displaying, ranking, or accepting quotations in NMS stocks that are priced in an increment of less than $0.01, unless the price of the quotation is less than $1.00. —SEC Reg NMS Subpenny Rule [1]

SEC rule 612 (Subpenny Rule) of regulation NMS restrict the pricing increment to be no smaller than $0.01 if the security is priced equal to or greater than $1.00 per share. For liquid stocks, the uniform minimum pricing increment, or the so-called the tick size is binding. 50% of S&P 500 stocks have bid-ask spread at 1 penny(see Fig. 2.1). In that case, the bid-ask spread is exactly one penny. For liquid stocks, the potential revenue that the market maker can make from this activity is one penny per share. Since the minimum pricing increment is uniform for all stocks in the market, market makers can get more revenue per dollar if they make the market for low-price stocks. For example, holding all else equal, the unit revenue for a market maker to trade on a $5 stock (with relative tick size of 200bps) is twenty times as much as that on a $100 stocks (with relative tick size of 10bps). A low-price stock

1. hinders price competition of market making because of subpenny rule.

2. generates higher rent for market making because of constrained price competition.

Therefore, I find algorithmic market making activities are relatively more active for low-price securities.

[1] Discussion in this chapter is based on (27)

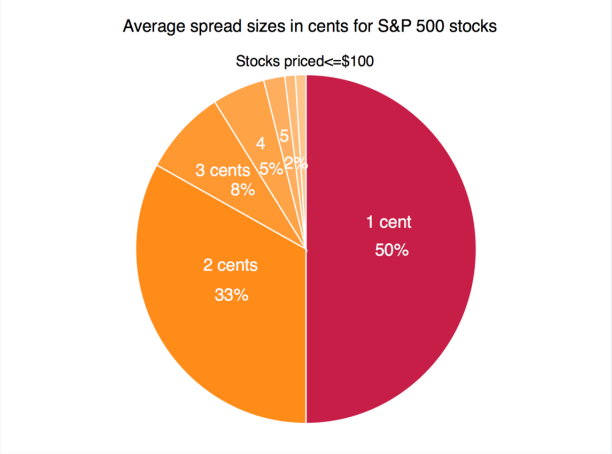

Figure 2.1: Avg. Spread Size in Cents for S&P 500 Stocks

2.2 Testable Hypothesis and Empirical Design

This chapter will test the following two main hypothesis:

I. For low-price securities, the algorithmic trading activities is high.

II. For low-price securities, revenue of algorithmic trading is high.

2.3 Data and institutional details

The empirical section of this paper uses three main datasets: a NASDAQ HFT dataset, the NASDAQ TotalView-ITCH with a nanosecond time stamp, and Bloomberg. CRSP and Compustat are also used to calculate stock characteristics. The sample period is October 2010 unless indicated otherwise.

2.3.1 Sample of stocks and NASDAQ HFT data

The NASDAQ HFT dataset provides information on limit-order book snapshots and trades for 117 stocks selected by Hendershott and Riordan. The original sample includes 40 large stocks from the 1000 largest Russell 3000 stocks, 40 medium stocks ranked from 1001 to 2000, and 40 small stocks ranked from 2001 to 3000. Among these stocks, 60 are listed on the NASDAQ and 60 are listed on the NYSE. Since the sample was selected in early 2010,

three of the 120 stocks were delisted before the sample period (BARE, CHTT and KTII).

The limit-order book data offer one-minute snapshots of the book from 9:30 a.m. to 4:00 p.m. for each sample date and stock. The displayed liquidity for each of these 391 snapshots is classified to two types: liquidity provided by algorithmic traders(algo traders) and liquidity provided by non-algo traders. This paper focuses on the analysis of the displayed quotes from algo traders and non-algo traders at the best bid and ask. The trade file provides information on whether the traders involved in each trade are algo traders or non-algo traders. In particular, trades in the dataset are categorized into four types, using the following abbreviations: HH: algo traders who take liquidity from other algo traders; HN: algo traders who take liquidity from non-algo traders; NH: non-algo traders who take liquidity from algo traders; and NN: non-algo traders who take liquidity from other non-algo traders. Therefore, the trade data allows me to break out trading volume into two types: volume due to the liquidity provision from algo traders and volume due to liquidity provision from non-algo traders.

2.3.2 Sample of stocks and NASDAQ ITCH data

I use a difference-in-differences approach to identify the causal link between price and algorithmic trading activity. The test uses leveraged ETFs that have undergone splits/reverse splits as the treatment group, and uses leveraged ETFs that track the same indexes and do not split/reverse split in my sample period as the control group. Leveraged ETFs are ETFs often appearing in pairs and tracking the same index but in opposite directions. For example, the ETFs SPXL and SPXS both track the S&P 500, but SPXL amplifies S&P 500 returns by 300% while SPXS does so by -300%. These twin leveraged ETFs usually have similar nominal prices when launched for IPO, but the return amplification results in frequent divergence of their nominal prices after issuance. The issuers often use splits/reverse splits to keep their nominal prices aligned with each other.

I search the Bloomberg and ETF Database to collect information on leveraged ETF pairs that track the same index with an identical multiplier, and the data are then merged with

CRSP to identify their splitting/reverse splitting events. To ensure there is enough trading volume in these ETFs, I use leverage ETFs that have at least 100 trades each day. I identify 11 splits and 30 reverse splits from January 2010 through December 2013. Since the NASDAQ HFT dataset does not provide algorithmic trading information for ETFs, I compute algorithmic trading activities based on methodologies introduced by Hasbrouck and Saar (12) using NASDAQ ITCH data, which is a series of messages that describe orders added to, removed from, or executed on the NASDAQ. I also use ITCH data to construct a limit-order book at nanosecond-scale resolution, upon which the calculation of liquidity is based.

2.4 Price and Algorithmic Liquidity Provision

This section first examines the relation between security price and algorithmic liquidity provision by multivariate regression analysis. In addition, it further establishes the causal link between security price and algorithmic liquidity provision (hypothesis I) using difference-in-differences analysis. In order to further test hypothesis I, I need to test the casual effect between price and algorithmic liquidity provision. Section 2.4.2.1 illustrates how the proportion of the liquidity provided by algo traders is measured. Section 2.4.2.2 presents the difference-in-differences test using the splits/reverse splits of leveraged ETFs as exogenous shocks to price to study the impact of price on algorithmic liquidity provision.

2.4.1 Multivariate Regression Analysis

The multivariate regressions in this section confirm that low price results in higher percentage of liquidity provided by algo traders. I control for additional variables to overcome omitted variable bias. The regression specification is

$$AlgoVolume_{i,t} = u_{j,t} + \beta * 1/price_{i,t} + \Gamma * X_{i,t} \tag{2.1}$$

where $AlgoVolume_{i,t}$ is the percentage of volume with algo traders as liquidity providers.

$u_{j,t}$ is the industry−by−time fixed effect. The key variable of interest, $1/price_{i,t}$, is the daily inverse of the stock price. $X_{i,t}$ are the control variables presented in Table 2.1.

Table 2.1 confirms that AlgoVolume is high for low-price securities, consistent with hypothesis I. The table also shows that stocks with larger market cap, more retail trading, and older age have higher percentage of liquidity provided by algo traders. The insignificant coefficients before other variables do not necessarily imply that these variables have no impact on absolute magnitude of the liquidity provision of algo traders. For example, the insignificant coefficient before volatility should be interpreted as that volatility has similar impact on the liquidity provision of algo traders and non-algo traders, and has no statistically significant impact on the ratio of their liquidity provision. The main takeaway from Table 2.1 is that algo traders concentrate their activity on stocks with a low-price securities.

2.4.2 Difference in Difference Analysis

This section examines the causal impact of security price on algorithmic liquidity provision (hypothesis I). Section 2.4.2.1 illustrates how the proportion of the liquidity provided by algo traders is measured. Section 2.4.2.2 presents the difference-in-differences test using the splits/reverse splits of leveraged ETFs as exogenous shocks to price to study the impact of price on algorithmic liquidity provision.

2.4.2.1 Measure of Algo Trader Activity

I use NASDAQ ITCH data to construct a proxy for algorithmic liquidity provision activities. The proxy is called strategic run by Hasbrouck and Saar (12) . A strategic run is a series of submissions, cancellations, and executions that are likely to form an algorithmic strategy. RunsInProcess is the sum of the time length of all strategic runs with 10 or more messages divided by the total trading time of that day (12). Hasbrouck and Saar (12) finds that RunsInProcess has very high correlation with the absolute level of algorithmic market making activities.

Dep. Var	AlgoVolume (%)	
	(1)	(2)
1/price	0.937***	0.956***
	(9.88)	(9.58)
logmcap	0.051***	0.031***
	(33.54)	(9.43)
turnover		0.007***
		(5.44)
volatility		-0.536
		(-1.64)
logbvaverage		0.005***
		(4.01)
idiorisk		-0.547***
		(-3.76)
age		0.003***
		(6.37)
numAnalyst		0.001**
		(2.04)
PIN		-0.407***
		(-6.51)
pastreturn		-0.034
		(-1.09)
R2	0.632	0.664
N	2268	2268
Industry*time FE	Y	Y

Table 2.1: Price and Algorithmic Liquidity Provision

2.4.2.2 Difference-in-Differences Test using Leveraged ETF Splits/Reverse Splits

This section establishes the causal effect between price, and the percentage of algorithmic liquidity provision using a difference-in-differences test. I use ETF split/reverse split as an exogenous shock to algorithmic liquidity provision. Treatment group would be split/reverse split ETFs. Their pairs that track the same index but do not split / reverse split provide an ideal control. Among ETFs, splits/reverse splits are more frequent for leveraged ETFs. The regression specification for the difference-in-differences test is:

$$y_{i,t,j} = u_{i,t} + \gamma_{ij} + \rho * Dtrt_{i,t,j} + \theta * return_{i,t,j} + \epsilon_{i,t,j} \tag{2.2}$$

where $y_{i,t,j}$ is algorithmic liquidity provision activity for ETF j in index i at time t. $u_{i,t}$, the index-by-time fixed effects, controls for time-varying algorithmic liquidity provision activity. γ_{ij} is the ETF fixed effect that absorbs the time-invariant differences between two leveraged ETFs that track the same index. The key variable in this regression is $Dtrt_{i,t,j}$, the treatment dummy, which equals 0 for the control group. For the treatment group, the treatment dummy equals 0 before splits/reverse splits and 1 after splits/reverse splits. Coefficient ρ captures the treatment effect.

	RunInProc(in .1sec)	
	(1)	(2)
	Split	Reverse
Dummytreatment	3.503***	-31.874***
	(3.42)	(-9.77)
return	-3.958	22.435
	(-0.63)	(0.91)
R2	0.978	0.85
N	607	2559
Index*time FE	Y	Y
ETF FE	Y	Y

Table 2.2: Diff-In-Diff Analysis

Table 2.2 shows an increase in algorithmic liquidity provision activities in the treatment group after splits relative to the control group. Since the analysis has controlled for the index-by-time fixed effect, an increase in algorithmic liquidity provision activities cannot be attributed to the change in the underlying fundamental of the leveraged ETF. Specifically, the increased algorithmic liquidity provision activities cannot be ascribed to information events, because the treatment and the control groups have the same underlying information. The ETF fixed effects and the return variable further control other possible differences between ETFs in the same pair. I argue that the increase in algorithmic liquidity provision activities is driven by price. Splits lead to coarser price grids, which can force traders who quoted different prices before splits to quote the same price after splits, thus causing an increase in the length of the queue at the new but more constrained best price. Column 2 reports a reduction in algorithmic liquidity provision activities following the reverse splits.

In summary, I find that splits increase algorithmic trading activity whereas reverse splits reduce algorithmic liquidity provision activities.

2.5 Price and Algorithmic Trading Revenue

In this section I demonstrate that the revenue is also higher for stocks with lower price. In addition, I find that algo traders do not outperform non-algo traders when they trade on the same stock, which is in disagreement with (9). One plausible explanation is that the superior performance of algo traders comes from their overweight in stocks with high revenue securities. I test hypothesis II using the following specification.

$$UnitRevenue_{i,t,n} = \beta_1 * AlgoDummy_{i,t,n} + \beta_2 * 1/price_{i,t} + u_{j,t} + \Gamma * X_{i,t} + \epsilon_{i,t,n} \quad (2.3)$$

1. **Total Revenue.** My revenue measure comes from Brogaard, Hendershott, and Riordan(9) , Menkveld (16) , and Baron, Brogaard, and Kirilenko (5). The algorithmic trading revenue for an individual stock i during for day t is defined as

$$\pi^{Algo,it} = \sum_{1}^{n} -(Algo_n^{it}) + INV\_Algo_N^{it} * P_{close}^{it} \quad (2.4)$$

   The revenue comes from two components. The first term, $\sum_1^n -(Algo_n^{it})$, captures total cash flows throughout the day. I separate day t's transactions into n blocks, within each block I have N transactions. The second term, often referred as positioning revenue, cumulates value changes associated with net position. In the analysis, *INV_Algo* accumulated in day t is cleared at closing midpoint quote P_{close}.

2. **Unit Revenue.** Since I am interested in the market marking revenue per dollar volume, I calculate the unit revenue as:

$$UnitRevenue_{i,t,n} = \pi^{Algo,it}/Dol^{Algo,it} \quad (2.5)$$

   where $Dol^{Algo,it}$ is the algo traders' total dollar volume for stock i on day t. The algo

trading revenue per dollar volume of non-algo traders is calculated analogously.

In specification 2.3, $UnitRevenue_{i,t,n}$ is the unit revenue for each stock i on date t for trader type n. There are two daily observations for each stock: a unit revenue for algo traders and a unit revenue for non-algo traders. These unit variables depends on a number of controls $X_{i,t}$ and two key variables of interest: price and AlgoDummy. $AlgoDummy_{i,t,n}$ equals 1 for the unit revenue of algo traders and 0 otherwise. $1/price_{i,t}$, is the reverse of price of stock i at day t. $u_{j,t}$ is the industry−by−time fixed effect. $X_{i,t}$ are control variables presented in Table 2.3.

Table2.3 shows that the unit revenue is high for low-price securities for all revenue measures. It also shows that algo traders do not have higher performance than non-algo traders do at stock by stock level . One plausible reason would be algo traders' overweight in stocks with higher unit revenue .

Dep. Var	Unit Profit(bsp)			
	(1)	(2)	(3)	(4)
1/price	15.497***	17.513***	18.977***	29.734**
	(5.45)	(3.97)	(3.43)	(2.31)
algo dummy	0.762***	0.421**	0.243	-0.094
	(6.43)	(2.29)	(1.06)	(-0.18)
1/price * algo dummy	7.054**	2.328	-0.261	-16.344
	(2.21)	(0.47)	(-0.04)	(-1.13)
logmcap	-0.160*	-0.129	-0.144	-0.233
	(-1.67)	(-0.87)	(-0.77)	(-0.54)
turnover	-0.085**	0.038	0.062	-0.178
	(-2.37)	(0.68)	(0.89)	(-1.11)
volatility	-19.509**	-91.450***	-105.805***	-179.865***
	(-1.97)	(-5.95)	(-5.48)	(-4.01)
logbvaverage	0.036	0.017	0.082	0.314*
	(0.98)	(0.30)	(1.15)	(1.92)
idiorisk	2.883	11.624*	16.507*	26.079
	(0.65)	(1.70)	(1.92)	(1.31)
age	-0.008	-0.028	-0.019	-0.033
	(-0.61)	(-1.37)	(-0.76)	(-0.57)
numAnalyst	0.040**	0.033	0.033	0.063
	(2.54)	(1.33)	(1.07)	(0.89)
PIN	0.797	2.895	7.518**	9.835
	(0.44)	(1.02)	(2.11)	(1.19)
pastreturn	-1.658*	1.373	4.769**	9.321**
	(-1.66)	(0.89)	(2.45)	(2.06)
R2	0.22	0.214	0.203	0.194
N	4484	4484	4484	4484
Industry*time FE	Y	Y	Y	Y

Table 2.3: Price and Algorithmic Trading Revenue

Chapter 3

Styled Algorithmic Trading under Mean-Variance Framework

3.1 Introduction

For major institutional clients such as mutual funds, retirement funds, or hedge funds, algorithmic trading for optimal execution helps liquidate or acquire big positions with limited impacts on both the markets and the clients. The execution houses (e.g., broker-dealers and agencies) in return generate commission revenues to cover their costs on IT infrastructures (e.g., servers or data warehouses), as well as on model and analytical development. Therefore, execution motivated algorithmic trading has become a crucial financial service for modern electronic markets. [2]

The most striking characteristics of algorithmic trading, as compared with the one-shot trading activities of small individual investors, is its constant flow of trading decisions made throughout the entire execution horizon. The key is to decide how to best "slice" the entire big order into smaller "child" orders to be progressively executed. Ideally such decisions have to resonate with the real-time dynamics of the markets and inventory positions. Dynamic programming (DP) naturally becomes the central approach to optimal algorithmic trading. This could be witnessed from the pioneering works of Bertsimas and Lo (7), and Huberman and Stanzl (13), to more recent ones by Almgren (1), Bouchard, Dang, and Lehalle (8), and Azencott et al. (4). Static algorithms, e.g., those by Almgren and Chriss (2), Huberman and Stanzl (13), Obizhaeva and Wang (17), Avellaneda and Stoikov (3), Kissell and Malamut (15), Gatheral (11), or Shen (24), can provide useful pre-trade estimation for clients. They nevertheless do not respond to fresh market information and thus cannot be designated as actual execution strategies.

[2] The discussion in this chapter is based on (25)

While it provides a natural paradigm, DP does impose a number of challenges on both modeling and computing. For instance, since the *Flash Crash* in the US markets on May 6th, 2011, it has been clear that trading risks can be significantly curbed via setting up even simple constraints like price or volume limits. Thus fundamentally algorithmic trading has to be a *constrained* DP problem, and closed-form solutions generally do not exist. In Section 3.2, I will discuss in more technical details several outstanding challenges in implementing a self-contained DP trading strategy, including, e.g., (a) defining proper DP objectives meaningful to general traders, (b) properly handling key trading constraints, and (c) efficiently dealing with the explosion of dimensions.

There are two main approaches to answering these challenges *in practice*, and both are in the approximation sense. The first is to *periodically* call a *static strategy*, e.g., every 20 minutes or whenever certain events trigger. This is a heuristic way to emulate dynamic trading via static strategies like that of Almgren and Chriss (2). Traditionally many execution houses have adopted this approach as it is much easier to integrate algorithmic trading with efficient static commercial optimizers such as IBM CPLEX or MOSEK. This heuristic approach can take good care of client constraints but generally lacks a coherent theoretical foundation.

The second way turns to the method of *approximate dynamic programming* (ADP). ADP is currently an active and rapidly evolving research area (e.g., (6, 22)). ADP aims at breaking the *curse of dimensionality* via properly designed approximation schemes. The approximations may be made for the objectives, state spaces, control actions, state transitions, the policies, or the Bellman equations. Despite such a vast degree of freedom, there exist no ADP design methods that work universally well for arbitrarily given DP problems. The design of an effective ADP scheme often relies on the particular characteristics of the tasks at hand as well as the insights of users (e.g., (26)).

The current work belongs to ADP in the above general sense. I define a style to be a particular choice of the following two components: (1) state variables, and (2) policies built upon these variables. Section 3.3 shall give a more technical account. Traditionally DP trading can only work with *additive* objectives. A style defined this ways allows a much broader class of trading objectives, including, e.g., the mean-variance utilities of Almgren

and Chriss (2).

The remaining sections demonstrate styled trading via the detailed construction of a particular style using the *mean-variance* objective and the two state variables of *moneyness and volume participation*. For convenience, this particular style is called the MV-MVP strategy. In Section 3.4 I define the two dimensionless state variables - *moneyness* and *forward PoV*, and also derive their state transition equations. In Section 3.5 I introduce the mean-variance objective for the *augmented* implementation shortfall (IS) (19), and define a particular parametric form of trading policies. The popular VWAP strategy can be considered as a special scenario of MV-MVP when all the four parameters are set to zero. Model calibration is discussed in Section 3.6, in which a generic instance of the MV-MVP style is also worked out.

Below are some important settings or assumptions for the current work.

1. The trading horizon $(0, T]$ is assumed to be an intraday time window, e.g., from 11:15 am to 1:30 pm, within the *continuous session*. The starting time is always denoted by $t = 0$. The current work does not cover any auction sessions such as the opens or closes.

2. As in most aforementioned works in the literature, I am primarily concerned with optimal decisions on setting the trade sizes for arrival bins. Let Δt denote an internally chosen child window size, e.g., 5 minutes. The trading horizon $(0, T]$ is then "sliced" into N child bins:

$$(t_0, t_1, \cdots, t_N) = (0, \Delta t, 2\Delta t, \cdots, N\Delta t = T).$$

   At each time t_n, the decision is to be made for the number of shares q_n to trade over the arrival bin $(t_n, t_{n+1}]$.

3. The current work does not address how to actually execute q_n shares within an arrival bin $(t_n, t_{n+1}]$. This is called the process of *Child Order Placement* (COP). A COP strategy often involves the following key components: (a) further slicing each child order into grandchild orders, (b) timing and anti-gaming of placing each grandchild

order, (c) selecting market or limit order types for each grandchild order, and (d) for limit orders determining book sitting levels, waiting times, and cancel-and-replace policies. In particular, one observes that the bin size Δt should be big enough in order to accommodate any COP limit-order strategies. Otherwise, time may be too short for the limit orders to be traded through.

3.2 Challenges in Dynamic Trading

I first discuss some major challenges that a traditional DP strategy has to face, in order to motivate styled DP trading.

$$J_0(q_0, \cdots, q_{N-1}; \mid \boldsymbol{\omega}\boldsymbol{x}_0) = \sum_{n=0}^{N-1} \beta^n g_n(\boldsymbol{x}_n, q_n) + \beta^N g_N(\boldsymbol{x}_N), \tag{3.1}$$

where $\boldsymbol{\omega}$ denotes a market path, and $\beta \geq 0$ a potential discount factor. For a single-factor market, one has $\boldsymbol{\omega} = (\omega_0, \cdots, \omega_{N-1})$ with ω_n denoting the scalar random market movement over $(t_n, t_{n+1}]$. The stagewise cost $g_n(\cdot, \cdot)$ and terminal cost $g_N(\cdot)$ may be stage dependent, as indicated by the subscripts. Let $\mathcal{D}_n(\boldsymbol{x}_n)$ denote the permissible action space for trading bin n, so that one demands $q_n \in \mathcal{D}_n(\boldsymbol{x}_n)$. Dynamical trading is to seek a series of optimal policies

$$\varphi_0^*(\cdot), \varphi_1^*(\cdot), \cdots, \varphi_{N-1}^*(\cdot),$$

such that

$$(\varphi_0^*, \cdots, \varphi_{N-1}^*) = \underset{(\varphi_0, \cdots, \varphi_{N-1})}{\arg\min} \; E_{\boldsymbol{\omega}}[J_0(q_n = \varphi_n(\boldsymbol{x}_n), 0 \leq n < N; \boldsymbol{\omega} \mid \boldsymbol{x}_0)]. \tag{3.2}$$

At the beginning time t_n of $(t_n, t_{n+1}]$, define the *spot* cost

$$J_n(q_n, \cdots, q_{N-1}, \boldsymbol{\omega} \mid \boldsymbol{x}_n) = \sum_{k=n}^{N-1} \beta^{k-n} g_k(\boldsymbol{x}_k, q_k) + \beta^{N-n} g_N(\boldsymbol{x}_N),$$

and the *spot* value function:

$$v_n(\boldsymbol{x}_n) = \min_{(\varphi_n,\cdots,\varphi_{N-1})} E_{\boldsymbol{\omega}}[J_n(q_k = \varphi_k(\boldsymbol{x}_k), n \le k < N; \boldsymbol{\omega} \mid \boldsymbol{x}_n)]. \tag{3.3}$$

Then one has the backward recursive Bellman equation: for $n = N-1, \cdots, 0$,

$$v_n(\boldsymbol{x}_n) = \min_{q_n} \left\{ g_n(\boldsymbol{x}_n, q_n) + \beta E_{\omega_n}[v_{n+1}(F_n(\boldsymbol{x}_n, q_n, \omega_n))] \right\}. \tag{3.4}$$

Here $\boldsymbol{x}_{n+1} = F_n(\boldsymbol{x}_n, q_n, \omega_n)$ is the state transition equation with stage specific market noise ω_n. The main challenges of dynamic algorithmic trading are summarized as follows.

1. **Pros and Cons of Additive Costs.** Additivity has induced the whole Bellman machinery (i.e., Eqn. (3.4)), but also severely restricted the available class of cost functions. It forces one to either work with the risk-neutral setting or adopt additive risk measures such as the *total quadratic variance*. Both do not explain well to traders who typically request the lowest average implementation shortfalls (IS) given a tolerable risk level. For them, the mean-variance objectives of Almgren and Chriss (2) are more pertinent even though they are non-additive.

2. **Completion.** A typical trader prefers an algorithm to complete the entire order within a specified trading horizon, say, from 10:00 am to 11:30 am. Most existing works achieve it by setting q_{N-1} of the last bin $(t_{N-1}, t_N = T]$ to be the entire remaining order (7).

   In reality, this may well deplete the entire market over the last bin if the outstanding order significantly dwarfs the market. One simple mechanism in favor of completion is to enforce a minimum participation rule:

$$q_n = \varphi_n(\boldsymbol{x}_n) \ge \text{minPoV} \cdot V_n \tag{3.5}$$

3. **Explosion of Dimensions.** Except for simple and often constraint-free models, e.g., LQR (6), there are typically no closed forms for either the value functions $v_n(\cdot)$'s or the

optimal policies $\varphi_n(\cdot)$'s. Let $\mathcal{A}_n(\boldsymbol{x}_n) = \mathcal{A}_n \subseteq R^d$ denote the state space for $\boldsymbol{x}_n \in R^d$. Each optimal policy φ_n^* is a mapping:

$$\varphi_n^* : \mathcal{A}_n(\boldsymbol{x}_n) \to \mathcal{D}_n(\boldsymbol{x}_n), \qquad \boldsymbol{x}_n \to q_n^* = \varphi_n^*(\boldsymbol{x}_n).$$

Even in the discretized setting, the search space for any single policy φ_n^* would have size L^M, assuming the state space $\mathcal{A}_n$ is discretized to M states, and the action space $\mathcal{D}_n$ to L actions. The search space is usually humongous.

4. **Constraints Handling.** Besides the completion constraint, typical traders also impose the following two constraints:

   (a) **Monotonicity.** That is, for a buy order, one demands: $q_n \geq 0$ for any n. Similarly, for a sell order, one requires: $q_n \leq 0$ for each n.

   (b) **Volume Participation Limit.** Most traders prefer to cap the maximum rate of PoV, such that:

   $$q_n \leq \text{maxPoV} \cdot V_n, \quad 0 \leq n < N. \tag{3.6}$$

   This is typically used as the ultimate gate keeper for avoiding market crash, and is especially important for illiquid securities.

   Such constraints have not been rigorously enforced in most traditional DP models due to the universal difficulty in accommodating constraints in DP.

3.3 Dynamic Algorithm Trading with Constraints

As the terminology "style" has not yet been commonly circulated in the industry, I first give a more detailed account about styles and style designing.

I define a "style" to be a user's particular choice of the two key factors:

1. State variables $\boldsymbol{x}_n = (x_{n,1}, \cdots, x_{n,d})$ for each bin $(t_n, t_{n+1}]$, based on which trading decisions are solely made.

2. A particular parametric form of trading policies:

$$q_n = \varphi_n(\boldsymbol{x}_n \mid \Theta), \quad n = 1, \cdots, N-1, \tag{3.7}$$

based on the state variables $\boldsymbol{x}_n$, and a finite set of parameters Θ. The set of parameters can be further split to two sets: $\Theta = \Theta_i \cup \Theta_e$. Θ_e denotes the set of trading control parameters exposed *externally* to traders. For example, most clients prefer to cap the volume participation level, e.g. maxPoV = 25%. Θ_i denotes the subset of remaining parameters that add further trading control but are *internally* optimized.

As an example, in the execution industry one often hears the terms "aggressive in the money (ITM)" or "passive in the money". Here,

1. Moneyness must be one of the state variables, which I shall elaborate on later. For the moment, let me denote moneyness by δ_n.

2. Suppose the buying strategy can be expressed by $q_n = \varphi_n(\delta_n, \cdots \mid \Theta)$, and suppose that for a buy order, ITM is defined as $\delta_n < 0$. Then the style of "being aggressive ITM" means that φ_n is monotonically decreasing on δ_n so that one trades aggressively when δ_n becomes more negative.

Since styled trading works with a particular parametric class of policies, it can only be suboptimal compared with the full engine of DP trading. But the gains are also substantial.

1. Significant complexity reduction. Under styled algorithmic trading, one is only required to calibrate a few or several internal parameters Θ_i, via a lower dimensional optimization problem.

2. Flexible handling of constraints. For example, most traders prefer to set the volume limit maxPoV. It suffices to choose a style in the form of: (with $a \wedge b = \min(a, b)$)

$$q_n = \varphi(\boldsymbol{x}_n \mid \Theta) = g_n(\boldsymbol{x}_n \mid \Theta) \wedge (\text{maxPoV} \cdot V_n),$$

where g is a "free" style, and V_n the projected market volume.

To illustrate the main ideas of styled algorithmic trading, for the rest of the paper, I shall focus on designing one specific style called the MV-MVP model. It is built upon the *mean-variance* (MV) objective and the state variables of *moneyness* and *volume participation* (MVP). To some degree, the MV-MVP style extends the popular VWAP algorithm by facilitating extra dynamic response to moneyness.

For simplicity, hereafter I shall use Θ to denote only the *internal* model parameters Θ_i. External parameters Θ_e are trading parameters set by traders and need no computation. Model calibration then means to calibrate Θ. For MV-MVP, Θ only contains four scalar parameters (a, b, c, d).

3.4 States and Dynamics Based on Moneyness and Forward PoV

Let (0, T] denote the trading horizon for buying or selling Q shares in total. Since the current model assumes no biases between buy and sell, I shall fix the running example to be a buy order. Following the Introduction, a DP algorithm decides the number of shares q_n to trade over each arrival bin $(t_n, t_{n+1} = t_n + \Delta t]$, with Δt being 1 minute or 5 minutes, for instance. It then relies upon a sound *Child Order Placement* (COP) strategy for actually executing q_n shares over the bin. A typical COP strategy involves a complex flow of actions on further grandchild order slicing, limiting order placement, and aggressive spread crossing, etc, as addressed in the Introduction.

3.4.1 State Variables: Moneyness and Forward PoV

Let p_n denote the opening price at t_n for each bin $(t_n, t_{n+1}]$. In reality, it could be the mid price of the National Best Bid and Offer (NBBO) at t_n, for example. For $n = 0$, p_0 is the initial arrival price for the entire execution. At each t_n, the state variable "moneyness" is defined by:

$$\delta_n = \frac{p_n - p_0}{p_0} = \frac{p_n}{p_0} - 1.0, \ n = 0, 1, \cdots, N-1. \tag{3.8}$$

Similarly, for analytical or computational purposes, I define the *posterior* moneyness $\bar{\delta}_n$ by:

$$\bar{\delta}_n = \frac{\bar{p}_n - p_0}{p_0} = \frac{\bar{p}_n}{p_0} - 1.0,$$

where $\bar{p}_n$ denotes the average execution price for all q_n shares over $(t_n, t_{n+1}]$:

$$\bar{p}_n = \frac{\sum_{k=1}^{K_n} p_{n,k} \cdot q_{n,k}}{q_n}, \quad \sum_{k=1}^{K_n} q_{n,k} = q_n.$$

Here, $(q_{n,k} \mid 1 \le k \le K_n)$ denotes the individual grandchild orders executed by the COP manager over $(t_n, t_{n+1}]$, and $(p_{n,k} \mid 1 \le k \le K_n)$ their execution prices.

Let Q denote the size of the entire order, and $Q_{exe} = q_0 + q_1 + \cdots + q_{N-1}$ the total executed shares over the designated horizon $(0, T]$. For dynamic trading, generally one could only guarantee that $Q_{exe} \le Q$. The realized *implementation shortfall* (IS) per dollar is defined to be:

$$IS_{exe} = \frac{\sum_{n=0}^{N-1} \bar{p}_n q_n - p_0 \cdot Q_{exe}}{p_0 \cdot Q_{exe}} = \frac{1}{Q_{exe}} \sum_{n=0}^{N-1} \bar{\delta}_n \cdot q_n \tag{3.9}$$

For example, $IS_{exe} = 0.0010$ means that to buy every \$10,000 amount of the security (as valued at $t = t_0 = 0$), one actually ends up paying \$10,010 on average.

By Eqn. (3.9), IS is determined by all the *posterior* moneyness $\bar{\delta}_n$'s which are unavailable at each planning time t_n. They clearly depend on the observables – the state variables of moneyness δ_n's that actually affect trading decisions.

I now introduce the second state variable – the forward PoV ν_n. At the starting time t_n of each arrival bin $(t_n, t_{n+1}]$, I define the residual Q_n via:

$$Q_0 = Q, \quad Q_n = Q_{n-1} - q_{n-1}, \quad 1 \le n \le N, \tag{3.10}$$

where q_{n-1} is the shares acquired over the departing bin $(t_{n-1}, t_n]$. Thus Q_n represents the total shares still needed to be executed at t_n. Let M_n denote the (projected) total *market* volume over the remainder of the execution horizon $(t_n, T]$. In order to minimize complexities, in the current work I assume market volumes obey a deterministic volume "profile". In many execution houses, such profiles are often automatically generated by

overnight or over-weekend processes that utilize the most recent market information. I must point out that for general styled trading, one could also introduce randomness into market volumes and a new state variable associated with it. Such a style can be more realistic but also more complex.

At the starting time t_n of an arrival bin $(t_n, t_{n+1}]$, the forward PoV is defined to be:

$$\nu_n = \frac{Q_n}{M_n}, \quad n = 0, 1, \cdots, N-1. \tag{3.11}$$

Similar to moneyness δ_n defined earlier, the forward PoV ν_n is also *dimensionless*. Moreover,

$$\nu_0 = \frac{Q_0}{M_0} = \frac{\text{Total Shares to Execute over } [0,T]}{\text{Total Market Shares over } [0,T]}$$

is precisely the PoV for a VWAP algorithm.

3.4.2 State Transitions and Permanent Market Impacts

Let $V_n = M_n - M_{n+1}$ denote the market volume over $(t_n, t_{n+1}]$. I define the actual PoV over the bin by:

$$u_n = \frac{q_n}{V_n}, \quad \text{which is dimensionless.} \tag{3.12}$$

For the MV-MVP style being constructed, I define

$$\text{State Variables: } \delta_n,\ \nu_n; \quad \text{and Control Variables: } u_n. \tag{3.13}$$

Notice that all three variables are dimensionless.

Let $\rho_n = \frac{V_n}{M_n}$ denote the volume percentage of a "current" bin $(t_n, t_{n+1}]$ over the remaining horizon $(t_n, T]$. The forward PoV has the transition equation:

$$\nu_{n+1} = \frac{Q_{n+1}}{M_{n+1}} = \frac{Q_n - V_n \cdot u_n}{M_n - V_n} = \frac{Q_n/M_n - V_n/M_n \cdot u_n}{1 - V_n/M_n} = \frac{\nu_n - \rho_n \cdot u_n}{1 - \rho_n}, \tag{3.14}$$

which is linear since ρ_n is known given a volume profile.

For intraday trading, the traditional Brownian motion for dP_t/P_t can be well approximated

by the Brownian motion of dP_t/P_0. Since $d\delta_t = \frac{dP_t}{P_0}$, in the discrete setting one could thus assume that

$$\delta_{n+1} = \delta_n + \sigma_n \sqrt{\Delta t} Z_n + ImpactCost_n(u_n), \tag{3.15}$$

where σ_n is the (annualized) spot volatility for $(t_n, t_{n+1}]$, and Z_n an independently drawn canonical Gaussian. Notice that for intraday trading, I always assume that market buys and sells are in balance so that no directional drift exists. The impact cost of the trading of $q_n = u_n \cdot V_n$ shares is $ImpactCost_n(u_n)$, which affects the execution prices of all trailing trades over $(t_{n+1}, T]$. It is thus called the *permanent impact*. For illustrative purpose, I assume:

$$ImpactCost_n(u_n) = \mu \cdot \sigma_n \sqrt{\Delta t} \cdot u_n^\gamma, \tag{3.16}$$

with some bin independent exponent $\gamma \in (0, 1]$ and coefficient $\mu > 0$. It has been broadly noted (e.g., (15)) that permanent costs are concave functions which requires $\gamma \leq 1$. Since δ_n (moneyness), σ_n (annualized spot volatility), and u_n (realized PoV) are all dimensionless, so must be the permanent impact coefficient μ. It is also possible to incorporate the so-called *transient impact* if necessary, though I will not consider it for the MV-MVP style. Transient impact represents the resilient effects of all the preceding trades $u_{n-1}, u_{n-2}, \cdots$ by time t_n, and is often expressed as a moving average (see e.g., Obizhara-Wang (17) and Shen (24)). To summarize, I have established the following state transition equations: for bin $n = 0, \cdots, N-1$:

$$\nu_{n+1} = \frac{\nu_n - \rho_n \cdot u_n}{1 - \rho_n}, \quad \rho_n = \frac{\nu_n}{M_n} = 1 - \frac{M_{n+1}}{M_n} \tag{3.17}$$

$$\delta_{n+1} = \delta_n + \sigma_n \sqrt{\Delta t} \cdot Z_n + \mu \sigma_n \sqrt{\Delta t} \cdot u_n^\gamma. \tag{3.18}$$

The initial state is:

$$\nu_0 = \frac{Q}{M_0}, \text{ the PoV for VWAP,} \qquad \text{and} \quad \delta_0 = \frac{p_0 - p_0}{p_0} = 0.0. \tag{3.19}$$

3.5 The MV-MVP Style

3.5.1 The Objective, Instantaneous Impact, and Augmented IS Cost

Let Q_N be the residual shares by the end of an execution horizon $t_N = T$. Generally due to the cap on volume participation imposed by traders, Q_N could be non-zero. The number of executed shares is thus

$$Q_{exe} = Q - Q_N = q_0 + q_1 + \cdots + q_{N-1}. \tag{3.20}$$

By the dimensionless Eqn. (3.9), the total dollar amount of the IS becomes

$$IS^{\$}_{exe} = p_0 \cdot Q_{exe} \cdot IS_{exe} = p_0 \cdot \sum_{n=0}^{N-1} \bar{\delta}_n \cdot q_n. \tag{3.21}$$

The *expected* difference between the posterior moneyness $\bar{\delta}_n$ and the moneyness state δ_n at t_n is modeled by the *instantaneous* impact. Due to the short span of Δt, I adopt a linear model (24):

$$\bar{\delta}_n - \delta_n = \eta \cdot \sigma_n \sqrt{\Delta t} \cdot u_n + \eta_{BB} \cdot \sigma_n \sqrt{\Delta t} \cdot Z_n. \tag{3.22}$$

The details are as follows.

1. η represents the instantaneous effect, and is in general inversely proportional to either the liquidity level under macro measures of an security (e.g., average daily volume or spread) or the thickness of the limit order book under market microstructures.

2. The canonical Gaussian Z_n is the same as in Eqn. (3.15), and the η_{BB}-term represents the average random cost of all q_n shares traded over $(t_n, t_{n+1}]$. Thus η_{BB} should be closely correlated to the in-house COP strategies. It can be shown under the theory of Brownian Bridges (14) that if the COP strategies uniformly place individual orders, one has $\eta_{BB} \approx 0.5$.

Due to linearity, η and η_{BB} can be calibrated via linear regression using in-house trading data. To impose the completion condition softly, I define the residual cost:

$$IS^{\$}_{res} = \Gamma \cdot \sigma p_0 \cdot Q_N, \tag{3.23}$$

where σ denotes the annualized daily volatility, and $\Gamma > 0$ is a residual-averse weight to penalize residuals. For instance, one could set $\Gamma = \Gamma_0 \cdot \sqrt{\frac{1}{BDAYS}}$, where $BDAYS$ denotes the number of trading days per year and Γ_0 an order $O(1)$ constant. This way the residual cost $IS^{\$}_{res}$ is roughly proportional to the overnight holding risk before one could trade the rest on the next business day.

Introduce the execution profile $\boldsymbol{q} = (q_0, \cdots, q_{N-1})$. Combining (3.21) and (3.23), I define the total *augmented* IS cost $IS_\Gamma = IS_\Gamma(\boldsymbol{q} \mid Q)$ per dollar to be:

$$IS_\Gamma = \frac{1}{Q}\sum_{n=0}^{N-1} \bar{\delta}_n \cdot q_n + \Gamma \cdot \sigma \frac{Q_N}{Q} = \frac{IS^{\$}_{exe} + IS^{\$}_{res}}{Q \cdot p_0} \tag{3.24}$$

with $Q_N = Q - q_0 - q_1 - \cdots - q_{N-1}$. It also depends on the market path $\boldsymbol{\omega} = (Z_0, \cdots, Z_{N-1})$, as in Eqn. (3.15) and (3.22). Thus one writes $IS_\Gamma = IS_\Gamma(\boldsymbol{q}, \boldsymbol{\omega} \mid Q)$. The objective is to minimize risk adjusted IS in the mean-variance framework of Almgren and Chriss (2, 24):

$$\min_{\boldsymbol{q}} \ \mathbb{E}_{\boldsymbol{\omega}}[IS_\Gamma(\boldsymbol{q}, \boldsymbol{\omega} \mid Q)] + \lambda \cdot \mathrm{VAR}_{\boldsymbol{\omega}}[IS_\Gamma(\boldsymbol{q}, \boldsymbol{\omega} \mid Q)] \tag{3.25}$$

for some given level of risk aversion $\lambda > 0$. As discussed earlier, traditional dynamic trading can hardly handle such non-additive objectives.

3.5.2 Design of The MV-MVP Style

As in Section 3.4, I shall work with the dimensionless PoV variables $u_n = \frac{q_n}{v_n}$. Define the execution PoV profile $\boldsymbol{u} = (u_0, \cdots, u_{N-1})$. Then the objective expression (3.25) can also be written as

$$\min_{\boldsymbol{u}} \mathbb{E}_{\boldsymbol{\omega}}[IS_\Gamma(\boldsymbol{u}, \boldsymbol{\omega} \mid Q)] + \lambda \cdot \mathrm{VAR}[IS_\Gamma(\boldsymbol{u}, \boldsymbol{\omega} \mid Q)] \tag{3.26}$$

The particular model being designed is called the MV-MVP style in order to highlight that (a) it is mean-variance based, and (b) the state variables are given by moneyness and volume participation (via forward PoV). The MV-MVP allows two important features.

1. To set a maximum volume participation limit maxPoV $\in (0, 100\%)$. Most clients prefer a PoV cap like maxPoV $= 10\%, 15\%$, or 25%.

2. To enforce a *minimum* participation rate minPoV. In DP trading, this is to ensure that at least a minimum amount of shares can be executed over a trading horizon, regardless of the actual market conditions.

Consequently I seek a policy in the form of:

$$u_n = \text{minPoV} \vee h_n \wedge \text{maxPoV} = \min(\max(\text{minPoV}, h_n), \text{maxPoV}), \tag{3.27}$$

where h_n is the "free" PoV in the linear form of:

$$h_n = \nu_0 + \alpha_n \cdot \delta_n + \beta_n \cdot (\nu_n - \nu_0), \quad n = 0, 1, \cdots, N-1. \tag{3.28}$$

(Be reminded that the running example is for a buy order with $u_n \geq 0$.) Notice that $\nu_0 = \frac{Q_0}{M_0} = \frac{Q}{M_0}$ is exactly the PoV for the VWAP algorithm. Here the two coefficient series $(\alpha_n)$and (β_n) provide further freedom for style design. To proceed, I always assume this compatibility condition to hold:

$$\text{minPoV} \leq \nu_0 \leq \text{maxPoV}. \tag{3.29}$$

While maxPoV is typically exposed to clients, minPoV could be internally designed.

I make the following comments regarding the roles of (α_n) and (β_n):

1. The free PoV h_n in Eqn. (3.28) is like a *factor model* commonly used in risk analytics (20), where δ_n and ν_n are the two observable factors.

2. The coefficients (α_n) effectively express the popular industrial styles of being "aggressive ITM" or "passive ITM", as discussed in Section 3.3. For the running example of

a buy order, ITM means $\delta_n < 0$. Then a positive α_n would reduce the PoV h_n while being ITM, and thus achieve being "passive ITM". Similarly, a negative α_n amounts to becoming aggressive ITM.

3. When setting $\alpha_n \equiv 0.0$ and $\beta_n \equiv 1.0$, $n = 0, \cdots, N-1$, one has $h_n \equiv \nu_n$. In this scenario one can further show recursively that $u_n \equiv h_n \equiv \nu_n \equiv \nu_0$. That is, the MV-MVP style becomes the canonical VWAP algorithm.

4. Linear policies like Eqn. (3.28) are actual closed-form solutions in the classical constraint-free LQR models (*Linear Quadrature Regulators*), in which linear state transitions are coupled with stagewise quadratic costs (6).

The computational burden for the policies in Eqn. (3.27) and (3.28) under the mean-variance objective (3.25) is still enormous, as one has to optimize the coefficient series in a high-dimensional space:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \cdots; \alpha_{N-1}, \beta_{N-1}) \in R^{2N}.$$

More importantly, Bellman-type backward procedures do not apply as the mean-variance objective is not additive.

As a result, I complete the design of the MV-MVP style by further choosing an explicit parametric form for the decision coefficients: for $n = 0, \cdots, N-1$,

$$\alpha_n = a \cdot \left[1 - \frac{n}{N-1}\right]^{1+b}, \quad \beta_n = 1 + c \cdot \left[1 - \frac{n}{N-1}\right]^{1-d}, \tag{3.30}$$

where one only requires $b > -1.0$ and $d < 1.0$, so that as n approaches the last trading bin $N-1$, one has:

$$\alpha_n \to 0.0, \quad \beta_n \to 1.0. \tag{3.31}$$

When $a = c = 0.0$, one recovers the VWAP algorithm.

In general, the MV-MVP style allows more active response to moneyness in the beginning of trading. Near the end of an execution horizon, however, it starts to prioritize in the completion requirement by converging to the forward PoV, i.e., $h_n \to \nu_n$.

3.6 Calibration of Styled Trading Models

To summarize, the MV-MVP style eventually becomes an optimization problem with only four unknown parameters (a, b, c, d):

$$\min_{(a,b,c,d)} F(a,b,c,d) = \mathbb{E}[IS_\Gamma(\boldsymbol{u}, \boldsymbol{\omega} \mid Q)] + \lambda \cdot \mathrm{VAR}[IS_\Gamma(\boldsymbol{u}, \boldsymbol{\omega} \mid Q)]. \tag{3.32}$$

The policy style is defined via the specific parametric form:

$$\begin{cases} u_n = \mathrm{minPoV} \vee h_n \wedge \mathrm{maxPoV}, \\ h_n = \nu_0 + \alpha_n \cdot \delta_n + \beta_n \cdot (\nu_n - \nu_0), \\ \alpha_n = a(1 - \frac{n}{N-1})^{1+b}, \quad \beta_n = 1 + c(1 - \frac{n}{N-1})^{1-d}. \end{cases} \tag{3.33}$$

Also recall that the state transition equations are:

$$\begin{cases} \delta_{n+1} = \delta_n + \mu \cdot \sigma_n \sqrt{\Delta t} \cdot u_n^\gamma + \sigma_n \sqrt{\Delta t} Z_n, \\ \nu_{n+1} = \frac{\nu_n - \rho_n u_n}{1 - \rho_n}, \quad \rho_n = \frac{\nu_n}{M_n} = 1 - \frac{M_{n+1}}{M_n}. \end{cases} \tag{3.34}$$

The initial state is given by $\delta_0 = 0.0$ and $\nu_0 = \frac{Q}{M_0}$, and the latter is precisely the PoV for the VWAP algorithm.

Finally, as in the preceding section, along any path $\boldsymbol{\omega} = (Z_0, \cdots, Z_{N-1})$,

$$\begin{cases} IS_\Gamma(\boldsymbol{u}, \boldsymbol{\omega} \mid Q) = \frac{1}{Q} \sum_{n=0}^{N-1} \bar{\delta}_n \cdot q_n + \Gamma \cdot \sigma \cdot \frac{Q_N}{Q}, \quad q_n = u_n \cdot V_n, \\ \bar{\delta}_n = \delta_n + \eta \cdot \sigma_n \sqrt{\Delta t} \cdot u_n + \eta_{BB} \cdot \sigma_n \sqrt{\Delta t} \cdot Z_n. \end{cases} \tag{3.35}$$

I must remind readers since the model is buy-sell symmetric , I have been consistently working with a buy order for which $Q > 0, u_n, h_n \geqslant 0$. The setting for a sell order is very similar when working with the absolute or unsigned PoV u_n and absolute order size Q, except for that (i) the plus signs in (3.34) for the permanent impact term and that in (3.35) for the instantaneous impact term should both be flipped, and (ii) $\bar{\delta}_n$ in the first term of IS_Γ in (3.35) should be replaced by $(-\bar{\delta}_n)$. Computationally, one could simply introduce a

new variable for order type:

$$side = +1, \text{ for a buy,} \quad \text{and} -1, \text{ for a sell,}$$

and place it as a multiplier for the aforementioned three terms.

Let $\Theta = (a, b, c, d)$. Then $IS_\Gamma(\boldsymbol{u}, \boldsymbol{\omega} \mid Q)$ can be expressed as $IS_\Gamma(\Theta, \boldsymbol{\omega} \mid Q)$, and the objective (3.32) becomes:

$$\min_{\Theta} \; F(\Theta) = \mathbb{E}[IS_\Gamma(\Theta, \boldsymbol{\omega} \mid Q)] + \lambda \cdot \text{VAR}[IS_\Gamma(\Theta, \boldsymbol{\omega} \mid Q)]. \tag{3.36}$$

Eventually a styled dynamic trading model becomes a parametric model. Due to the nonlinearities associated with the flooring or capping operators, the power exponents in Eqn. (3.33), as well as the nonlinearities in the state equations (3.34), the objective function $F(\Theta) = F(a, b, c, d)$ is generally nonlinear and non-convex. Thus only local optima are sought after via suitably designed iterative schemes. There exists a wealth of literature on optimizing objectives of stochastic nature, and below I briefly describe two major numerical schemes that I have experimented with: *stochastic gradient descent* and *binomial-tree* based gradient descent.

For any given set of styled parameters Θ, the objective involves two ensemble statistics - the mean and the variance. Thus one could employ the method of *stochastic gradient descent* (SGD) by building gradient descending schemes upon Monte-Carlo valuation (23). My extensive experiments have shown that the local optima captured this way are sensitive to initial guesses, and that the convergence usually takes long. Conventional variance reduction techniques such as the antithetic method do *not* work well since the objective is *not* monotonic against random market moves.

Alternatively, the binomial-tree based gradient descent (BT-GD) is more stable and works particularly well for horizons with a moderate number of bins, e.g., 90 minutes in the form of 18 bins of 5-minute child windows. In this implementation, the Gaussian market path in Eqn. (3.35)

$$\boldsymbol{\omega} = (Z_0, \cdots, Z_{N-1})$$

is approximately implemented by the binomial path:

$$\boldsymbol{\omega} = (U_0, \cdots, U_{N-1}), \quad U_k \sim B(\{1, -1\}, 0.5), \quad k = 0, \cdots, N-1,$$

where U_k's are i.i.d.'s of a symmetric Bernoulli type $B(\{1, -1\}, 0.5)$:

$$\text{Prob}(U_k = 1) = \text{Prob}(U_k = -1) = 0.5.$$

All binomial paths are equally likely under this setting, and the objective function for any given parameter set Θ can be evaluated exactly.

At each iteration step with the "current" estimator $\Theta^0 = (\theta_1^0, \theta_2^0, \theta_3^0, \theta_4^0)$ (corresponding to (a, b, c, d)), the gradient is estimated via *synchronized* finite differencing as follows. Define the basis directions:

$$\boldsymbol{e}_1 = (1, 0, 0, 0), \quad \cdots, \quad \boldsymbol{e}_4 = (0, 0, 0, 1).$$

For a set of small parameters $(\varepsilon_1, \cdots, \varepsilon_4)$ fixed throughout the iteration, the gradient components are estimated via the central difference scheme:

$$\frac{\partial F(\Theta^0)}{\partial \theta_i} = \left[F(\Theta^0 + \varepsilon_i \boldsymbol{e}_i) - F(\Theta^0 - \varepsilon_i \boldsymbol{e}_i)\right] / 2\epsilon_i, \qquad i = 1, 2, 3, 4.$$

The evaluation of the eight objective values $F(\Theta^0 \pm \varepsilon_i \boldsymbol{e}_i), i = 1, 2, 3, 4$ is synchronized over all market paths. That is, for each evolving market path $\boldsymbol{\omega}$, the eight IS_Γ's associated with the eight sets of trading parameters are simultaneously computed and recorded. This improves computational efficiency.

As mentioned earlier, the VWAP strategy is a special instance of the MV-MVP style when:

$$a = b = 0 \text{ (i.e., } \alpha_n \equiv 0), \qquad \text{and} \quad c = d = 0 \text{ (i.e., } \beta_n \equiv 1).$$

This provides a natural set of initial guess for any iterative scheme.

Below I demonstrate the behavior of the proposed MV-MVP model via one generic numerical instance. I explain it via a pseudo *object-oriented* fashion, in order to facilitate popular production languages such as C++, Java, or Python.

1. I assume that the market object has the following data fields:

   ```
   market.dayInMinutes=390, market.yearInBusinessDays=252.
   ```

   This is to simulate a *regular* US equities market which trades from 9:30 am to 4:00 pm (except for special half-days), and has about 252 trading days per year.

2. For illustrative purposes, I assume *flat* profiles of intraday volatilities and volumes, and the particular security object contains this information:

   ```
   security.ID="ABC", security.vol=25.0%, security.ADV=4,000,000.
   ```

   Here the *average daily volume* (ADV) of 4 million shares represents roughly the average ADV of S&P 500 stocks, which are large-cap and popularly traded. As common in finance, the volatility is annualized. An annual volatility of 25.0% is roughly the average level of the S&P 500 universe.

3. The trading object trade summarizes both the external trading request from a client and internal trading conventions. For the particular instance, it is assumed that:

   ```
   trade.startIdx=30,  trade.endIdx=120, trade.binWindow=5,
   trade.maxPoV=25%, trade.minPoV=2.5%,
   trade.tradeSize=92300, trade.side="BUY".
   ```

   That is, the trade request is to buy 92300 shares from 10:00am to 11:30am, with PoV capped at 25.0% and floored at 2.5%. The algorithm also internally sets bin size to 5 minutes. The order size corresponds to the 10% PoV for a VWAP strategy over the same horizon.

4. The model object model is assumed to have the following data fields. In reality, these impact model parameters have to be calibrated from the real trading database. In reality, risk aversion λ is internally mapped to the user-friendly external GUI fields like "Urgency Level."

$$\mathsf{model}.\mu = 0.5,\ \mathsf{model}.\eta = 1.0,\ \mathsf{model}.\Gamma = 10.0,\ \mathsf{model}.\lambda = 0.01.$$

I have set $\eta_{BB} = 0.5$ as explained earlier.

Given the above instance of the market, security, trade, and model, my binomial-tree based calibrator converges at the following set of policy parameters:

$$a = 0.23,\ b = -0.48;\ c = 2.24,\ d = 0.48.$$

The resulting policy coefficients of α_n's and β_n's are plotted in Figure 3.1 via the Eqn. (3.30).

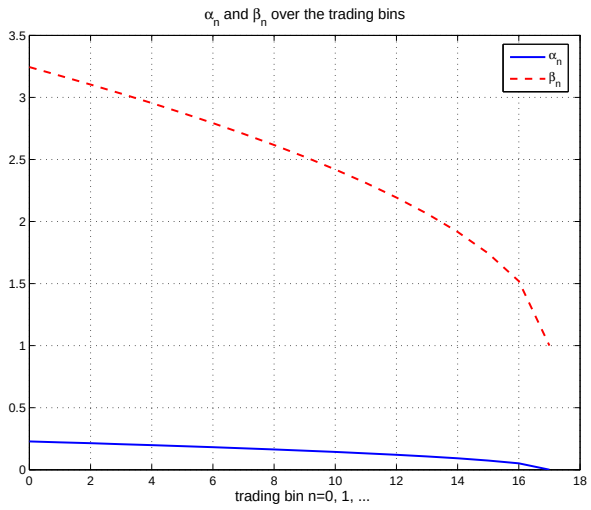

Figure 3.1: α_n and β_n Over The Trading Bin

To better visualize the calibrated MV-MVP style, in Figure 3.2 I have plotted the simulated trading paths given 10 or 100 independently simulated market paths, according to the dynamic policies in Eqn. (3.27) and (3.28). Left: 10 trading paths; Right: 100 trading paths. Both are based on independently simulated market paths. As a reference, pay attention that the original total position has been designed so that the VWAP PoV is 10%. The simulated trading paths oscillate around VWAP for different market scenarios as expected. The MV-MVP incorporates a new degree of dynamic response to the moneyness.

As expected, the MV-MVP style oscillates around the VWAP strategy.

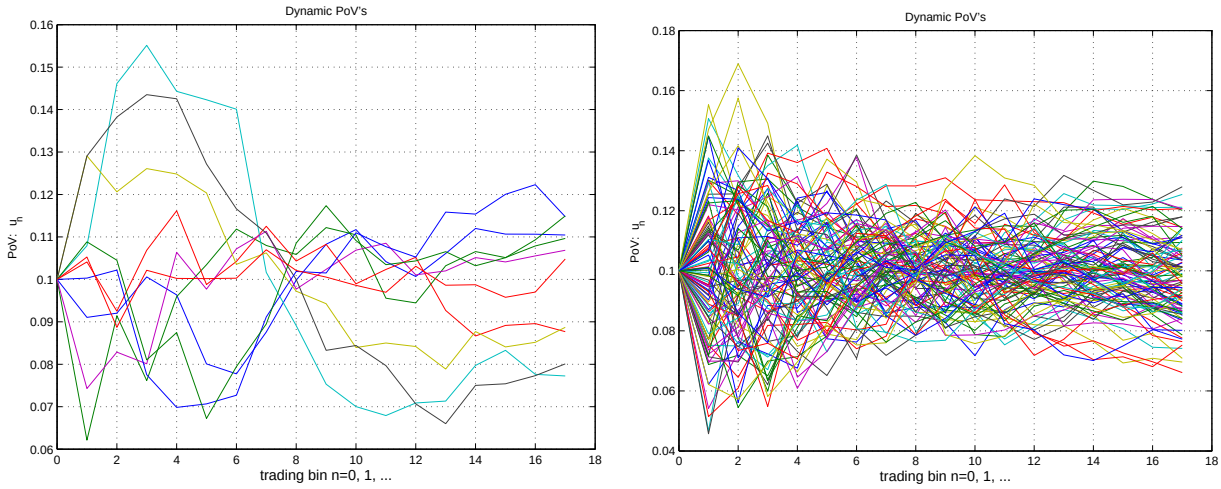

Figure 3.2: Simulated Trading Paths

Appendix A

$$AlgoVolume_{i,t} = \frac{\%\ of\ vol\ with\ AlgoTrader\ as\ market\ maker\ for\ stock\ i\ on\ date\ t}{Total\ trading\ volume\ for\ stock\ i\ on\ date\ t} \tag{A.1}$$

References

[1] R. Almgren. Optimal trading with stochastic liquidity and volatility. *SIAM J. Financial Math.*, 3:163–181, 2012.

[2] R. Almgren and N. Chriss. Optimal execution of portfolio transactions. *J. Risk*, 3:5–39, 2000.

[3] M. Avellaneda and S. Stoikov. High-frequency trading in a limit order book. *Quant. Finance*, 8(3):217–224, 2008.

[4] R. Azencott, A. Beri, Y. Gadhyan, N. Joseph, C.-A. Lehalle, and M. Rowley. Realtime market microstructure analysis: online transaction cost analysis. *ArXiv e-prints, arXiv1302.6363, http://adsabs.harvard.edu/abs/2013arXiv1302.6363A*, 2013.

[5] Matthew Baron, Jonathan Brogaard, and Andrei A Kirilenko. Risk and return in high frequency trading. *Available at SSRN 2433118*, 2014.

[6] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2007.

[7] D. Bertsimas and A. W. Lo. Optimal control of execution costs. *J. Financial Markets*, 1(1):1–50, 1998.

[8] B. Bouchard, N.-M. Dang, and C.-A. Lehalle. Optimal control of trading algorithms: a general impulse control approach. *SIAM J. Finan. Math.*, 2(1):404–438, 2011.

[9] Jonathan Brogaard, Terrence Hendershott, and Ryan Riordan. High-frequency trading and price discovery. *Review of Financial Studies*, 2014.

[10] Justin Dove. *Is High Frequency Trading Causing Higher Volatility?*, 2011. `http://www.investmentu.com/article/detail/23278/high-frequency-trading-causing-volatility#.VT1Z7CFVhHw`.

[11] J. Gatheral. No-dynamic-arbitrage and market impact. *Quant. Fin.*, 10(7):749–759, 2010.

[12] Joel Hasbrouck and Gideon Saar. Low-latency trading. *Journal of Financial Markets*, 16(4):646 – 679, 2013. High-Frequency Trading.

[13] G. Huberman and W. Stanzl. Optimal liquidity trading. *Yale School of Management Working Papers*, YSM 165, 2001.

[14] I. Karatzas and S. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, 1991.

[15] R. Kissell and R. Malamut. Algorithmic decision making framework. *J. Trading*, 1(1):12–21, 2006.

[16] Albert J. Menkveld. High frequency trading and the new market makers. *Journal of Financial Markets*, 16(4):712 – 740, 2013. High-Frequency Trading.

[17] A. Obizhaeva and J. Wang. Optimal trading strategy and supply/demand dynamics. *J. Financial Markets*, 16(1):1–32, 2013.

[18] Scott Patterson. *High-Speed Stock Traders Turn to Laser Beams*, 2014. `http://www.wsj.com/news/articles/SB10001424052702303947904579340711424615716`.

[19] A. F. Perold. The implementation shortfall: Paper versus reality. *J. Portfolio Management*, 14:4–9, 1988.

[20] B. Pfaff. *Financial Risk Modelling and Portfolio Optimization with R.* Wiley, 2013.

[21] Matt Phillips. *Nasdaq: Heres Our Timeline of the Flash Crash*, 2010. `http://blogs.wsj.com/marketbeat/2010/05/11/nasdaq-heres-our-timeline-of-the-flash-crash/`.

[22] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* Wiley, 2011.

[23] A. Shapiro and Y. Wardi. Convergence analysis of gradient descent stochastic algorithms. *J. Optim. Theory Appl.*, 91(2):439–454, 1996.

[24] J. Shen. A pre-trade algorithmic trading model under given volume measures and generic price dynamics. *Applied Math. Res. Express,* Oxford Univ. Press, in press, 2014.

[25] Jackie Shen and Yingjie Yu. Styled algorithmic trading and the mv-mvp style. *Available at SSRN 2507002*, 2014.

[26] Y. Wang, B. O'Donoghue, and S. Boyd. Approximate dynamic programming via iterated Bellman inequalities. *Int. J. Robust Nonlinear Control,* DOI: 10.1002/rnc.3152, 2014.

[27] Chen Yao and Mao Ye. Tick size constraints, market structure, and liquidity. *Unpublished working paper, University of Illinois at Urbana, Champaign, Urbana, IL*, 50, 2014.