Reducing transaction costs with low-latency trading algorithms

Sasha Stoikov*† and Rolf Waeber‡

† Cornell Financial Engineering Manhattan (CFEM), New York, N.Y. 10004 U.S.A.
‡ School of Operations Research and Information Engineering (ORIE),
Cornell University, Ithaca, N.Y. 14853 U.S.A.

September 16, 2015

Abstract

We formulate a trade execution problem at the market microstructure level and solve it using dynamic programming. The objective is to sell a single lot of an asset in a short time horizon T, using the imbalance of the top of book bid and ask sizes as a price predictor. The optimization problem takes into account the latency L of the trading algorithm, which affects the prices at which the asset is traded. The solution divides the state space into a "trade" and a "no-trade" region. We calculate the cost of latency per lot traded and demonstrate that the advantage of observing the limit order book can dissipate quickly as execution latency increases. In the empirical section, we show that our optimal policy significantly outperforms a TWAP algorithm in liquidating on-the-run U.S. treasury bonds, saving on average approximately 1/3 of the spread per share if trades are executed with low latency ($\approx$ 1 millisecond).

Keywords: Optimal asset liquidation, algorithmic trading, transaction costs, market microstructure, high-frequency trading, optimal stopping, trade execution latency, cost of latency, dynamic programming.

1 Introduction

The financial services industry has seen a tremendous push towards faster and faster trading systems and algorithms in the last decade. Fiber optic cables, microwave technologies, collocation services and high performance computers have been adopted by most financial firms that trade in large volumes. This so called "arms race" has been well documented in the popular press, but there is no consensus on whether or not such investments make economic sense.

*Corresponding author. Email: sashastoikov@gmail.com

The optimal execution literature (see, for instance, Bertsimas and Lo, 1998; Almgren and Chriss, 2001; Obizhaeva and Wang, 2005; Alfonsi et al., 2010; Gatheral and Schied, 2012) has generally focused on the optimal splitting problem. Essentially, the concern of these studies is how to divide a large "parent" order into a trade schedule made up of "child" orders so as to minimize transaction costs. Under general conditions, the solution to the order splitting problem is a deterministic function of time; that is, the trading rate does not depend on the immediate market environment Schied et al. (2010). However, the names of popular trading algorithms offered by brokers – "React", "Bolt", "Stealth", "Sniper", "Guerrilla", "Ambush" and "Dagger" to name a few – imply that faster algorithms are better. Our goal in this paper is to evaluate and quantify the extent to which lower latency trading can reduce transaction costs.

Latency occurs for several reasons: first, data (i.e. an order book update) needs to travel from the exchange to the trading server; second, the server needs to process the update and decide whether or not to submit an order; third, if an order is sent, it needs some time to travel back to the exchange. Nowadays, the total latency of the trade execution is very small–the entire process takes place in milliseconds or even microseconds.

Latency in trade execution has only recently received attention in the academic literature. Moallemi and Saglam (2013), for example, estimate the cost of latency by comparing the value of a pegged limit order strategy with and without latency. They express this cost in terms of the volatility and the bid-ask spread of an asset. We will define the value of latency in a similar way, except our trading strategy will involve market orders. Our motivation for using a pure market order strategy is that such a strategy allows for realistic backtesting using Level I quotes data and does not rely on any assumptions of passive executed trading. We assume that the round-trip latency of an algorithm is fixed and given by a known constant L, which we will assume to be in the range 0-5,000 milliseconds. In practice, latency is not constant and may be related to the volatility in the market, see Kirilenko and Lamacie (2015).

Our model determines the optimal time in $[0, T]$ to liquidate *one* single lot, conditional on the state of the limit order book. We will consider T of the order of 1 minute, which has traditionally been the realm of high frequency trading (HFT) firms. In this paper we will quantify to what extent algorithms operating on a longer time scale can also gain from short term price prediction signals. The optimal stopping problem we consider is affected by latency drastically. If a decision is made to sell one share at the bid price at time τ, by the time the order reaches the market, that price may have changed and will be executed at $P^b_{\tau+L}$, the bid price at time $\tau + L$. Our optimization problem with latency is therefore

$$\sup_{t \le \tau \le T-L} \mathbb{E}[P^b_{\tau+L} - P^b_t | \mathcal{F}_t] \tag{1}$$

where $\mathcal{F}_t$ is the information available in the order book at time t. The focus of this paper is to investigate the potential cost saving by using low-latency trading infrastructure, i.e., the change in transaction cost due to latency as L decreases.

The literature on optimal stopping theory is extensive, consider, for example, the

monographs Shiryaev (1978) and Peskir and Shiryaev (2006) as a starting point into this field of research. We assume that the information contained in the limit order book can be reduced to a generic HFT signal I_t which predicts price jumps of the bid price. The only assumptions for our dynamic programming approach is that I_t is a Markov process and that the price dynamics are independent of the actual price level. In this paper we will focus on the top of book imbalance, that is, $I_t = B_t/(A_t + B_t)$ where A_t and B_t are the volumes quoted at the best ask and bid prices, respectively. To solve the above optimal stopping problem, we rely on techniques similar to finding optimal exercise times of American options. We present a general formulation based on a time- and space-discretization and dynamic programming, (see, for example, Chapter 8 in Glasserman (2004)). We then backtest these optimal liquidation strategies on Level I quotes for 5-years U.S. treasury bonds. To the best of our knowledge this is the first paper to backtest optimal liquidation strategies on Level I data.

The structure of the paper is as follows. In Section 2, we motivate and formulate the optimal liquidation problem. In Section 3, we approximate the optimal trade regions by a dynamic programming approach. In Section 4, we apply our analysis to U.S. Treasuries data and find that our algorithm significantly beats a TWAP benchmark policy. This allows us to quantify the cost savings obtained by reducing latency in algorithmic trading. Section 5 concludes and discusses future research directions.

2 The Optimal Stopping Problem

At the millisecond time scale[1], algorithms process order book updates in order to decide when to submit orders. These updates contain market orders, limit orders and changes to existing limit orders. HFT firms observe any update in the limit order book to produce trading signals. In this paper we will focus on one of the most widely documented trading signal in the order book, the Level-I imbalance:

$$I_t = \frac{B_t}{A_t + B_t}$$

where A_t is the volume offered at the best ask price and B_t is the volume available at the best bid price at time t. We assume that $A_t > 0$ and $B_t > 0$ for all $t \geq 0$. This factor has significant predictive power for short term price moves, particularly in liquid assets where the bid ask spread is usually not wider than one tick (see Avellaneda et al. (2011) for an application to US Equity Level-I data). When the imbalance is close to zero, the next price move is more likely to be a down move and when the imbalance is close to one, the next price move is more likely to be an up move. This trading signal is hardly a secret in the 5 year US treasuries market (as well as in many other markets). Most sell trades are submitted to the market when the imbalance is low and most buy trades

[1] As technology evolves, time horizons and latencies have become shorter and shorter. In some markets, the timescale of interest has gone down to microseconds or even nanoseconds. Our data is rounded to the nearest millisecond, but the same analysis and techniques can be easily applied to even more granular time scales.

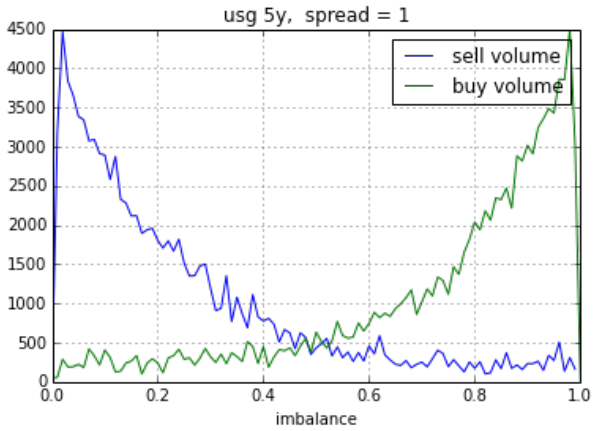

Figure 1: Histograms of imbalances I_t, the millisecond before a (sell or buy) trade arrives at the exchange.

are submitted when the imbalance is high (see figure 1). Note that there is no trading allowed at the mid price in this market. We find that when the imbalance is close to 0.5, there is very little volume traded. We expect this factor to remain an important predictor of price moves, as it represents the instantaneous imbalance of supply and demand in the market microstructure.

From the information contained in the filtration $\mathcal{F}_t$ we will only use the current best bid price P_t^b and the imbalance I_t to solve the optimal stopping problem (1). We assume the following properties for the two-dimensional stochastic process (P_t^b, I_t):

1. (P_t^b, I_t) is a Markov process;
2. Periodicity in the price dynamics, i.e., $(P_{t+\delta}^b - P_t^b, I_t)$ is independent of P_t^b, for all $\delta \geq 0$.

Problem (1) is then equivalent to

$$\sup_{t \leq \tau \leq T-L} \mathbb{E}[P_{\tau+L}^b - P_t^b | I_t]. \tag{2}$$

The payoff function at time $t = T - L$ is

$$g^L(x) = \mathbb{E}[P_T^b - P_{T-L}^b | I_{T-L} = x] = \mathbb{E}[P_L^b - P_0^b | I_0 = x] \tag{3}$$

where we used the Markov property of the imbalance process.

The function $g^L(x)$ captures the expected payoff associated with delaying a trade by L milliseconds, given imbalance x (see Figure 2). Note that for a latency of 1 millisecond, the payoff is practically zero for all imbalances, since the price after one millisecond is not likely to change. As the latency L increases, the payoff function becomes steeper. When the imbalance is relatively low, it is advantageous to sell rather than wait; when the

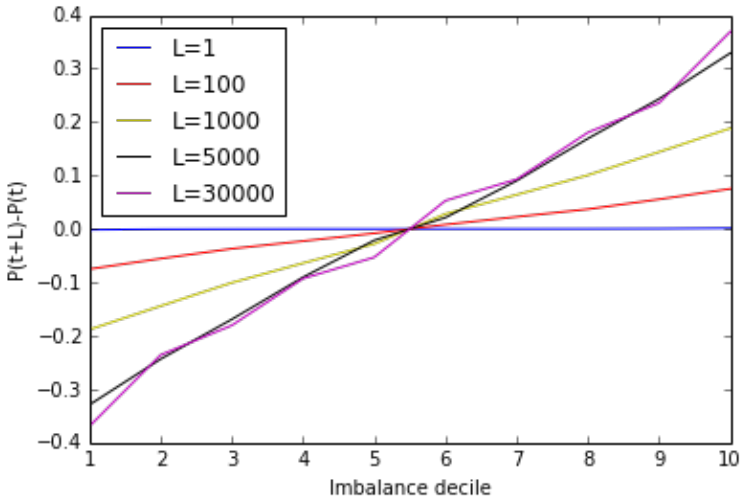

Figure 2: The function $g^L(x)$ expressed in fractions of the bid-ask spread. The latency L is in milliseconds.

imbalance is relatively high, it's worth to wait for better trading opportunities. Beyond a latency of 5 seconds, the payoff $g(x)$ is not noticeably different. At every point in time, we will compare the immediate exercise payoff to the value function

$$v(t,x) = \sup_{t \le \tau \le T-L} \mathbb{E}[P^b_{\tau+L} - P^b_t | I_t = x] \tag{4}$$

conditional on the trading signal I_t, where P^b_t is the arrival price, a standard benchmark in the optimal execution literature. We do not allow for anticipating liquidation rules, hence the liquidation time τ needs to be a stopping time adapted to the filtration generated by I_t. We denote the set of all such stopping times as $\mathcal{T}$ and the supremum is taken over all stopping times $\tau \in \mathcal{T}$.

The theory of optimal stopping for Markov processes (see, for example, Chapter 1.2.2 in Peskir and Shiryaev (2006)) shows that it is optimal to liquidate at a time $t \in [0,T]$ if $V(I_t,t) = g^L(I_t)$ and to wait for a better opportunity if $v(t,I_t) > g^L(I_t)$. This motivates the definition of the "trade" and "no-trade" region

$$D = \left\{(x,t) \in [0,1] \times [0,T] : v(t,x) = g^L(x)\right\},$$
$$C = \left\{(x,t) \in [0,1] \times [0,T] : v(t,x) > g^L(x)\right\}.$$

Define

$$\tau_D = \inf\left\{t \ge 0 : (I_t,t) \in D\right\}$$

as the first entry time into the set D. For the discrete-time and discrete-space case the time τ_D corresponds to the optimal liquidation time. This, however, is not true for the more general continuous-time and continuous-space case which would require additional

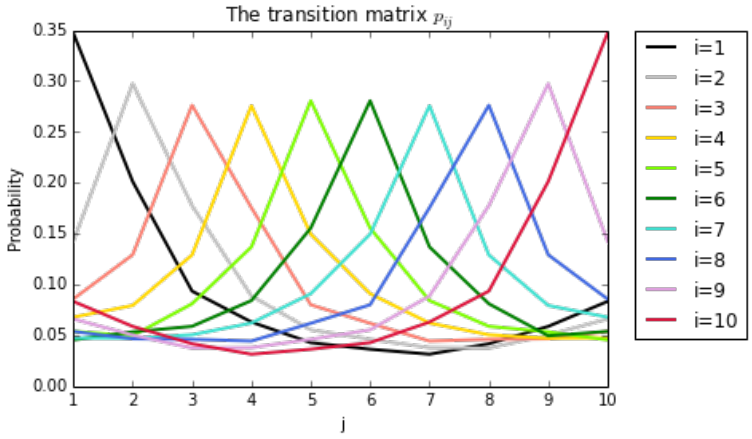

Figure 3: Each graph represents the probability of transitioning from state i to state j in 1 second. Note the peaks represent the probability of remaining in the current state

assumptions on the payoff function g and the process I_t. We will focus on the discrete-time and discrete-space case in this paper, but refer the interested reader to Peskir and Shiryaev (2006) for an in-depth discussion of optimal stopping problems in continuous time and space.

An intuitive consequence of this formulation is the following proposition:

Proposition 2.1 *Fix $t \in [0,T], x \in \mathbb{R}$, then $v^L(t,x)$ is non-increasing in L, for $L \in [0, T-t]$.*

This proposition shows that trading with larger latency can never be better than trading with smaller latency ("low latency trading is better"). The proposition holds trivially, as one can always add "artificial" latency to a trading strategy, and therefore any strategy with longer latency is automatically contained in the set of strategies with shorter latencies.

3 Discrete Model

We solve the stopping problem stated in (2) using a discrete time, discrete space approximation where we assume that the imbalance process I_t takes values in the discrete set $\{1, 2, \ldots, M\}$ for some $M \in \mathbb{N}$ (this can be achieved by computing quantiles of the imbalance I_t). Without loss of generality, we assume that bid prices assume values in $\mathbb{Z}$, which means that our value functions are expressed in fractions of the tick size. Under this assumption prices could be negative, but as we are interested in relative price moves this is not a critical extension of reality where prices cannot be negative.

Therefore we consider a Markov chain process $(P^b(n), I(n))$ with countable state space $\mathbb{Z} \times M$. Due to the periodicity assumption of the price process the transition matrix from one state to the next only depends on the imbalance process $I(n)$.

We solve the discrete-time, discrete-space approximation to the optimal stopping problem

$$V^L(0,i) = \sup_{0\leq\tau\leq T-L} \mathbb{E}[P^b(\tau+L) - P^b(0)|I(0)=i] \tag{5}$$

using the Bellman equation

$$V^L(n,i) = \max\left\{G^L(i), \mathbb{E}[V^L(n+1, I(n+1))|I(n)=i]\right\}. \tag{6}$$

The immediate payoff function $G^L(i)$ can be estimated for various latencies as

$$G^L(i) = \mathbb{E}[P^b_L - P^b_0|I_0 = i]. \tag{7}$$

To calculate the conditional expectation $E[V^L(n+1, I(n+1))|I(n)=i]$ in (6) we use the periodicity assumption of the Markov process $(P^b(n), I(n))$, which implies that on a price jump from the bid price $P^b(n)$ of size d the value function V jumps exactly by d price levels up, resp., down. This is the key observation that allows us to reduce the state space to M states and makes the problem tractable. Let's denote with p^d_{ij} the transition matrix of the imbalance process, conditional on the bid price P^b jumping up d price levels, where $d \in \mathbb{Z}$. Then, based on the periodicity assumption of the Markov process, the conditional expectation can be written as:

$$E[V^L(n+1, I(n+1))|I(n)=i] = \sum_{d\in\mathbb{Z}}\sum_{j=1}^{M} p^d_{ij}(V^L(n+1,j)+d).$$

We use the notation p_{ij} for the transition matrix of the imbalance process $I(n)$, then

$$\begin{aligned} E[V^L(n+1, I(n+1))|I(n)=i] &= \sum_{d\in\mathbb{Z}}\sum_{j=1}^{M} p^d_{ij}V^L(n+1,j) + \sum_{d\in\mathbb{Z}}\sum_{j=1}^{M} dp^d_{ij} \\ &= \sum_{j=1}^{M} p_{ij}V^L(n+1,j) + G^\delta(i) \end{aligned}$$

where

$$G^\delta(i) = E\left[P^b(n+1) - P^b(n)|I(n)=i\right]. \tag{8}$$

Let's further denote with $p_{i\cdot}$ the ith row of the transition matrix P and $V^L(n+1,\cdot)$ the value function at time $n+1$, then the Bellman equation can be rewritten as

$$V^L(n,i) = \max\left\{G^L(i), p_{i\cdot}V^L(n+1,\cdot) + G^\delta(i)\right\}. \tag{9}$$

In order to solve the dynamic program (6) we only need the transition matrix P of the imbalance process $I(n)$ plus an estimate of the payoff function G for the different

time lags δ and L. The time step δ can be chosen to match the latency L in which case the formulation further simplifies to

$$V^L(n,i) = G^L(i) + \left(p_i . V^L(n+1,\cdot)\right)_+ ,$$

where we use the standard notation $(a)_+ = \max(a, 0)$.

Also, the zero latency case can be considered (in which case $G^0(i) \equiv 0$):

$$V^0(n,i) = \left(p_i . V^0(n+1,\cdot) + G^\delta(i)\right)_+ .$$

In the next section we will investigate empirically the shape of different trade and no trade regions and analyze the correlation between the theoretical value function V and the actual cost saved $\hat{V}$ based on backtesting the liquidation strategy on out-of-sample data.

4 Empirical Case Study: U.S. Treasury Data

We now backtest the optimal trade and no-trade regions, using 5-years on-the-run U.S. treasury bonds data. The dataset consists of Level-I data (quotes and trades) recorded from 10:30am to 3pm every day for the first 10 trading days of July 2010 on the eSpeedTM trading platform. The granularity of the data is up to millisecond precision.[2]

For this particular asset, the bid-ask spread is almost always one tick (i.e. 1/128th of a dollar). Hence the dataset is well-suited for our model, which implicitly assumes that the bid-ask spread is never larger than 1 tick.[3] Assets with a one-tick spread are very common in modern electronic markets. It has been shown in Cont et al. (2010) that size imbalances at the best bid and ask prices can lead to short term price prediction, under the assumptions of a simple so-called zero intelligence model. Moreover, Avellaneda et al. (2011) show that imbalance in quote sizes is a strong predictor of short term price moves for equities with one cent bid-ask spreads. We therefore expect our method to easily extend to other assets with a one-tick spread.

In Subsection 4.1, we describe the procedure for computing the value function and trade region. In Subsection 4.2 we present a method for backtesting our market order strategies, using a TWAP algorithm as a benchmark. In Subsection 4.3 we discuss the expected transaction cost savings using an imbalance based algorithm and investigate the sensitivity of these savings to the latency assumption of the algorithm.

[2]The authors are very grateful to Jacob Loveless at Cantor Fitzgerald, L.P. for providing this dataset.

[3]In practice, of course, spreads can be larger than 1 tick size, but often these occurrences last only for a very short time period. In our testing we filter out the time periods when the spread is larger than 1 tick.

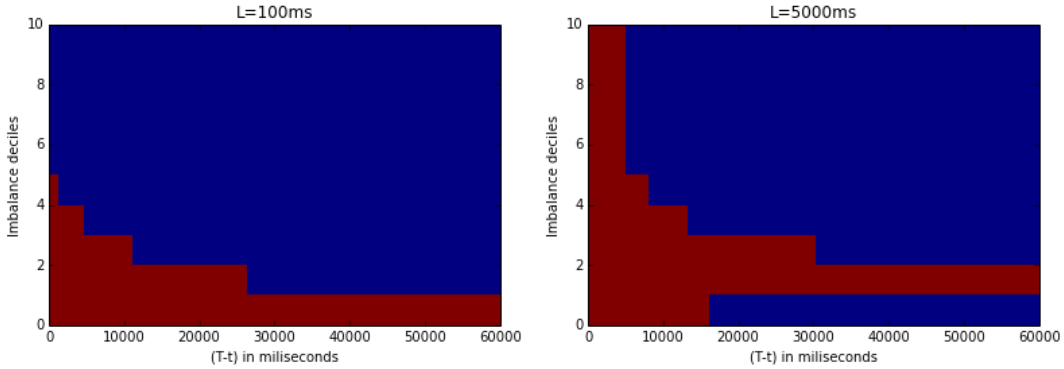

Figure 4: The trade regions D are in red and the no-trade regions C are in blue

4.1 Computing the trade region and value function

In order to backtest the optimal liquidation strategies we explicitly construct the finite state Markov process $I(n)$ and estimate its transition probabilities. Without loss of generality, we assume that there are 10 states ($M = 10$), and compute the 10 deciles of the variable:

$$I_t = B_t/(A_t + B_t),$$

where A_t is the best ask size and B_t is the best bid size. We tested different steps δ and found that $\delta = 100$ milliseconds performs reasonably well. For the remainder of the paper we will fix $\delta = 100$.

The estimation of the value functions $V^L(n, i)$ consist of the following steps:

1. Compute the empirical transition probabilities p_{ij} for $i, j \in [1, 2, 3, \ldots, 10]$. For each time stamp (sampled uniformly), record the imbalance decile now $I(n)$ and δ milliseconds later $I(n + 1)$.

2. Compute the payoff function $G^L(i)$ by averaging $(P^b_{t+L} - P^b_t)/s$ for all occurrences where $I_t = i$ (see equation 7).

3. Compute the payoff function $G^\delta(i)$ by averaging $(P^b(n + 1) - P^b(n))/s$ for all occurrences where $I(n) = i$ (see equation 8), where s is the tick size.

4. Compute the value function $V^L(n, i)$ using Bellman's equation (9).

5. Compute the trade region $D = \{(n, i) : V^L(n, i) = G^L(i)\}$

We avoid biases (for days that have strong directional trends) by estimating the functions G^d, G^L and p_{ij} in a way that keeps them symmetric. More specifically, we consider up and down (sell resp. buy trades) as symmetric events.

4.2 Backtesting a market order strategy

In the previous section we computed the discretized value function V^L and the corresponding trade region D^L. The value function represents the percentage of the bid-ask spread that our algorithm saves over a naive algorithm that submits market orders without looking at the imbalance.

Backtesting the performance of a trade region D on high frequency data poses two major challenges, namely that real orders have *price impact* and *latency*. When we send an order to the market, it impacts the price of the stock permanently, a fact that cannot be captured by historical data. Secondly, there is a latency between the time stamp of the market data and the time at which an algorithm based on this data would actually execute.

We address the *price impact* issue by constraining all algorithms to trade exactly one lot at a fixed frequency, e.g., T equals 1 minute. For a given trading frequency, we will assume that all algorithms have a comparable permanent price impact. The Time Weighted Average Price (TWAP) algorithms, which trades at a deterministic frequency, irrespective of the imbalance, will serve as a natural benchmark . We define TWAP(T) to be the average bid price obtained by the TWAP algorithm with trade frequency T. IMB(T) will refer to the average bid price obtained by the imbalance-based algorithm, which is also constrained to trade once time interval of length T. Each time interval contains a random amount of quotes, and a backtest consists in monitoring the process I_t for t in $[0, T]$ and submitting a simulated market sell order at time τ, when the process I_t is in the trade region D^L .

We address the *latency* issue by introducing a fixed delay of L milliseconds in our backtests. Any market order submitted at time τ will actually be executed at the best available price at time $\tau + L$, which might be different from the best available price at time τ. Hence the realized price of this trade is $P^b_{\tau+L}$. Although this latency has no substantial effect on the benchmark TWAP algorithm, it worsens the performance of the imbalance-based algorithm significantly.

For each day in our sample, the value of an imbalance-based algorithm trading with latency L at a frequency T is estimated as follows:

$$\hat{V}(T, L) = IMB(T, L) - TWAP(T) = \frac{1}{n_T} \sum_{i=1}^{n_T} \frac{P^b(\tau_i + L) - P^b(T_i)}{s}, \tag{10}$$

where the day is split into n_T intervals of length $T = T_{i+1} - T_i$, τ_i is the first time the imbalance enters the trade region D, $T_i \leq \tau_i \leq T_{i+1} - L$. In the next section we will compare $\hat{V}$ versus V on an out-of-sample data set.

4.3 The impact of latency on the transaction cost

In order to validate our model, we compare the value functions computed in Section 4.1 to the actual realized savings as backtested, using Equation (10). For the kth day in our

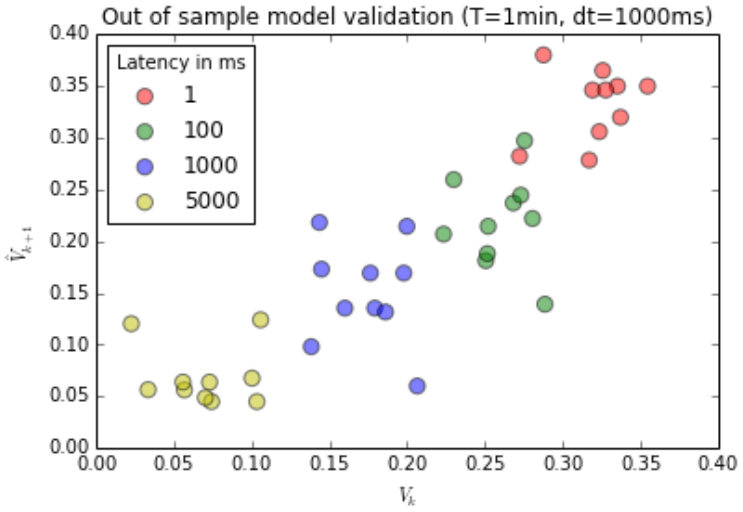

Figure 5: Comparing V to the backtest $\hat{V}$ out of sample

dataset, we compute the value function

$$V_k(T, L) = \frac{1}{M} \sum_{i=1}^{M} V^L(0, i; T)$$

at a given latency L and horizon T, and the trade region D_k. We then use the trade regions D_k to backtest the empirical performances $\hat{V}_{k+1}$. We compare V_k and $\hat{V}_{k+1}$ for various latencies and various days in Figure 3.

The decision to invest in low-latency technology is a challenging one. The costs of trading fast are relatively transparent as most exchanges will provide collocation services for a fee. However, the benefits of trading fast are challenging to estimate, as they may depend on the type of proprietary signals an algorithm is using. Our model focusses on one particular factor, the imbalance, which provides an opportunity to delay trades and save money relative to algorithms that trade deterministically in time. The amount that an imbalance-based algorithm saves depends on the trading frequency T and the latency L. Table 1 provides an overview how the latency L and trading frequency T affect the cost savings from a low-latency trading algorithm for our dataset. The cost savings are increasing in T, which makes intuitive sense as an algorithm that trades less frequently has more time to delay a trade and wait for a better trading opportunity. The cost savings decrease with increasing latency L, as it is less likely to take advantage of a short lived trading signals when the time for the order to reach the exchange increases (also see Proposition 2.1). Note that the values are expressed in fractions of the tick size. For example, an algorithm that trades once a minute ($T = 1\text{min}$), with a latency of one second ($L = 1000\text{ms}$), may be expected to save $0.17 * 270 * \$78 = \$3,580.20$ per day, since there are 270 minutes in a trading day and the notional value of the tick size is \$78 (as is the case for 5-years US treasury bonds). Notice that reducing the latency

T	$L = 1$ms	$L = 100$ms	$L = 1000$ms	$L = 5000$ms
10s	0.20	0.17	0.11	0.05
1min	0.32	0.26	0.17	0.07
5min	0.33	0.35	0.18	0.07

Table 1: The performance of the optimal imbalance-based algorithm over TWAP for various values of T and L. The values are expressed as percentages of the minimum bid ask spread.

to one millisecond can be expected to save $0.32 * 270 * \$78 = \$6,739.20$ per day.

The imbalance-based execution algorithm saves on average about 1/3 of the spread as compared to a naive TWAP strategy if the trading latency is low (around 1ms) and the trading frequency is not too high (1 trade per minute). A trading firm that trades at this kind of frequency may find that the savings in execution costs may be worth the cost of the technology or collocation fees.

5 Conclusions and Future Research

We study an asset liquidation problem over a very short time frame which uses the dynamics of the limit order book to identify good liquidation times. The problem is similar to optimally exercising an American option, in that the state space is divided into a trade and no-trade region, separated by a free boundary. We describe a dynamic programming method for approximating the optimal trade region. The trade and no-trade regions allow us to directly backtest our liquidation strategy on empirical data and we demonstrate that one can save a significant fraction of the bid-ask spread compared to a naive TWAP algorithm.

There are a few directions that we would like to explore in our future research:

- Extending to various classes of stochastic processes that could lead to efficient methods of computing the value function and the free boundary. Obtaining closed form solutions for processes that are easy to calibrate to data could be a promising avenue to explore;

- Including the possibility of trading with limit orders, this will definitely improve the performance of the algorithm, however, efficient backtesting remains challenging;

- Testing this type of liquidation approach on other assets. It would be particular interesting to extend similar trading algorithms to assets where the usual bid-ask spread is greater than one tick, in which case not only the timing of the market and limit order, but also the chosen price level will be of importance;

- Extending the analysis to trading signals beyond the imbalance process, something that's done by many algorithmic trading firms, but the available literature is very scarce;

- Adjusting the liquidation policy to risk-aversion. This would allow us to embed a problem like ours within a longer term execution algorithm such as the Almgren and Chriss (2001) model.

References

A. Alfonsi, A. Fruth and A. Schied (2010): Optimal execution strategies in limit order books with general shape functions. *Quantitative Finance* 10:143–157.

R. Almgren and N. Chriss (2001): Optimal execution of portfolio transactions. *Journal of Risk* 3:5–40.

M. Avellaneda, J. Reed and S. Stoikov (2011): Forecasting Prices from Level-I Quotes in the Presence of Hidden Liquidity. *Algorithmic Finance* 1.

D. Bertsimas and A. Lo (1998): Optimal control of execution costs. *Journal of Financial Markets* 1:1–50.

R. Cont, S. Stoikov and R. Talreja (2010): A stochastic model for order book dynamics. *Operations Research* 58:549–563.

J. Gatheral and A. Schied (2012): Dynamical models of market impact and algorithms for order execution. *Available at SSRN: http://ssrn.com/abstract=2034178* .

P. Glasserman (2004): *Monte Carlo Methods in Financial Engineering*, volume 53. Springer.

A. Kirilenko and G. Lamacie (2015): Latency and asset prices. *Available at SSRN: http://ssrn.com/abstract=2546567* .

C. C. Moallemi and M. Saglam (2013): The cost of latency in high frequency trading. *Available at SSRN: http://ssrn.com/abstract=1571935* .

A. Obizhaeva and J. Wang (2005): Optimal trading strategy and supply/demand dynamics. Technical report, Techreport MIT.

G. Peskir and A. N. Shiryaev (2006): *Optimal Stopping and Free-Boundary Problems*, volume 10. Birkhauser.

A. Schied, T. Schöneborn and M. Tehranchi (2010): Optimal basket liquidation for CARA investors is deterministic. *Applied Mathematical Finance* 17:471–489.

A. N. Shiryaev (1978): *Optimal Stopping Rules*. Springer.