Modelling Execution Strategies for Algorithmic Trading via MDP's

October 22, 2010

Blaise Standish

z3223539

Supervisor: Dr Gareth Peters

Supervisor: Dr David Oliver

School of Mathematics

The University of New South Wales

Submitted in partial fulfillment of the requirement of

the degree of Masters in Financial Mathematics

Abstract

Optimal execution concerns the minimal cost realization of a trading strategy. Within this thesis the execution domain is reviewed and several seminal execution models presented, before considering the problem as a Markov decision process. To fully define the execution domain the concept of a trading strategy is defined and a number of popular pricing models outlined, with particular emphasis given to market impact - the component of the pricing model related to the execution problem. Implementation shortfall, one of the key measurement metrics, is detailed leading to the definition of a commonly utilized optimality criteria. The existing analytical and dynamic programming solutions presented initiated significant interest in this research area which has since been significantly extended. The novelty introduced involves developing a Markov model, in which the models have been constructed to explore the execution problem from a different perspective. The Markov decision process model drives the problem from signals generated through price movement changes, depending on the inventory position. The partially observed (not observed) Markov decision process model sets the problem up for scenario analysis, that is the study of the cause-effect relationship for a trading action. The hidden (not hidden) Markov model formulation uses particle filtering to identify belief states in a noisy environment. Each of the models is novel and exposes many interesting areas for further and future research

Contents

Nomenclature

$1_{\{.\}}$	Indicator variable
α	Q-Learning rate.
β^p	Order of power law function for permanent impact.
β^t	Order of power law function for temporary impact.
γ	Q-Learning discount term.
ι	Recovery time of Limit Order Book.
κ	Inverse urgency, represents half life or the desired time scale for liquidation
λ	Trader utility.
$\mathbb{P}$	Standard probability measure on S.
μ	Drift, represents actions of informed traders on stock price.
ω	Temporary impact parameter.
π	Policy, for action selection.
$\Psi(.)$	Permanent market impact function.
$\psi(.)$	Temporary market impact function.
ρ	Memory parameter for information term.
σ	Represents volatility. The actions of the uninformed traders.
τ	Time duration of individual time interval from k to k+1.
Θ	Resilience parameter of Limit Order Book.
θ	Impact parameter.
$\tilde{S}$	Weighted average price received per asset, on individual execution.
ς	Effect of Information parameter.
*	Indicates optimal solution.
a_k	Action selection at time index k.

b_k	Belief state vector at time index k.
$c(.)$	Cost function.
$g(S)$	Volume of asset available at price level S in Limit Order Book.
I_k	Market information term at time index k. Autoregressive lag 1.
L, l_k	Leaves, residual trade volume left for execution, at start then at each time index.
M	Mid or centerprice of asset.
m	Spread, difference between S^+ and S^-.
N	Number of intervals that T is divided into.
n_k	Trade package, an individual trade list entry executed at time point k.
p_k	Total value of trade execution at time index k.
$r(.)$	Reward function.
S	Asset price.
S^+	Best ask price of asset.
S^-	Best bid price of asset.
T	Time limit for execution of trade strategy.
u	Input vector, used in dynamic programming formulation.
$V(.)$	Value function, summarizes reward or cost.
v_k	Trade rate, the velocity of trading at time index k.
W	A standard Brownian motion.
x	State vector.
y_k	Observation vector at time index k.
k	Time point index.

1 Introduction

Optimal execution characterizes the minimal cost realization of an investment decision in a financial market. An investment decision concerns the change in portfolio position of an investor through the acquisition or liquidation of listed assets. This process would appear on first impression to be a simple task of buying or selling an asset at a given price. However for an investor wishing to buy (sell) a given volume at a particular price there must exist a counterparty (or number of counterparties) that wishes to sell (buy) the same volume. In a listed financial market the buyers and sellers post orders for the assets that they wish to acquire or liquidate, quoting prices and corresponding order quantities, the order prices and volumes continually change like the eb and flow at the point of change of the tide as investors jokey for position. The availability of order volume is referred to as market liquidity. For our investor wishing to buy (sell) at a particular price there must be enough counter liquidity at the specified price. If there is insufficient counter liquidity then the investor must choose to either transact at a worse price where there is sufficient counter liquidity or over a period of time in the hope that market dynamics attract more counter liquidity at an acceptable price. On completion of the given transaction the performance of the trade execution may then be measured and benchmarked. Measurement can be a simple process of taking the initial price at the start of the transaction, referred to as the arrival price, multiplied by the total volume that was exchanged, and comparing this against the weighted average price received multiplied by the volume exchanged over the duration of the transaction. The difference between the theoretical bench mark price and the actual price received is the Implementation Shortfall [31].

For financial institutions the importance of understanding the optimal execution problem and the mechanisms that can be utilized to minimize cost cannot be emphasized enough. This is especially relevant in today's market in which there is a proliferation of algorithmic trading systems, that is software applications that have trading decision logic encoded, which are increasingly being used in the execution of investment decisions. The area of algorithmic trading itself is very broad, it encompasses the benchmark execution algorithms often referred to as structural algorithms. These execution algorithms have been developed by sell-side brokers concerned with minimizing cost for buy-side clients. It also encompasses the more sophisticated situational algorithms that have been developed for proprietary trading desks to take advantage of arbitrage opportunities. It is the structural execution algorithms that we will be concerned with within this thesis. In the United States at the start of 2010, 65%-75% of message volume resulted from algorithmic trading, in the United Kingdom over 40% of trading activity resulted from algorithmic trading and in Australia 10% was attributed to algorithmic trading activity [10]. Going forward it is believed that the competition in the structural execution algorithm space will only increase, requiring a greater understanding of the optimal execution problem and an understanding of the options available to minimize execution cost for a competitor.

This thesis is concerned with the optimal execution problem, the motivation is the provision of execution strategies that minimize the associated execution cost, as measured by the slippage away from a theoretical bench mark price, that is the deviation from the benchmark price actuated by the transaction. The approach taken is to consider optimal execution strategies in the context of Markov models. Markov models are stochastic models with particular conditional independence properties. The Markov property defines a model in which the dependence structure is on the present, not the past. Markov Models encompass Markov Chains, Markov Decision Processes (MDP)'s, Hidden Markov Models (HMM)'s and Partially Observable Markov Decision Processes (POMDP)'s. This study of execution strategies is relatively new. Since the initial paper by Perold [31], defining execution cost, and the initial seminal paper by Bertismas and Lo[14], utilizing aspects of dynamic programming, it has predominately been through the works of Almgren [3][4][9][5][8][7][6][2] that the optimal execution problem has been explored. Although there has been a more recent proliferation in the number of papers published in this area to the extent of the authors knowledge this is the first time that the area has been studied in the context of Markov Decision Processes, Partially Observed Markov Decision Processes and Hidden Markov Models.

Minimization of execution cost through optimal execution provides an advantage to brokers concerned with agency trading, allowing them to quote lower fees for agency trades to prospective clients, and for principal trading institutions allowing them to maximize their return on investment through simple reduction in cost. The minimization of the execution cost associated with a change in position can provide a unique competitive advantage for a financial institution. Optimal execution strategies are a jealously guarded area of intellectual property covertly developed and religiously protected by financial institutions.

This thesis is divided into several sections. Section 2 reviews the execution domain, section 3 reviews several existing approaches to the optimal execution problem followed by section 4 which covers Markov models and their applications to trade execution, and section 5 provides the conclusion and a discussion of future work.

Section 2 covers the Execution Domain. It sets the scene by covering the background to the optimal execution problem and defining the concept of trading strategy mathematically. A trading strategy is the realization of an investment decision through the execution of buy/sell orders over a period of time. The placement of orders over time defines a trading trajectory that projects a move from one capital position to another. This provides the basis for our optimization, as the ultimate objective is to find the trading trajectory with a cost/reward that is minimal/maximal according to some optimization criteria.

Pricing models used by the various authors are presented, with particular emphasis on market impact, a component of the pricing model that is specific to trade execution. Market impact is defined to be the effect on the price of an asset resulting from the execution of a trade on the market. To fully understand market impact, we briefly look at the microstructure of the limit order book and the effects that a trade has on price momentum. Predominately the pricing models are Bachelier type algebraic random walks, as considered by Bertismas and Lo [14], all of the Almgren models without drift [3][4][9][5][8][7][2] and with drift [6], and Obizhaeva and Wang [30]. Some of these models univariate and some multivariate. Some are discrete [14][4], and some continuous [30][2]. With the trading strategy and pricing law of motion defined we are in a position to consider optimality, the utility criteria that is used to find an optimal policy in the MDP setting. The key optimality criteria is based around minimization of expected cost, or mean-variance cost. The cost term in optimal execution modelling is referred to as implementation shortfall.

Section 3 reviews some existing approaches to the optimal execution problem. It builds directly on the aspect of the execution domain introduced in Section 2. Although there are numerous papers published in the area we focus on the seminal works of Bertismas and Lo 1998 [14], R. Almgren and N. Chriss [4], R. Almgren and J. Lorenz [7], and R. Almgren [2]. This section starts by introducing dynamic programming [12] drawing heavily from the works of [19] introducing the Bellman equation, for the discrete formulations, and the Hamilton-Jacobi-Bellman (HJB) equation for the continuous case. Two dynamic programming solutions are provided, and two analytical solutions presented, the solutions are presented chronologically as each solutions draws from and builds on the earlier solution. The first dynamic programming solution is a minimization of expected cost, for a statically defined trading strategies [14]. The second solution is analytic, it addresses mean-variance optimization for static trading strategies [4], this is followed by an analytical solution for the dynamic trading optimization case [7]. The final solution outlined is a dynamic programming solution using the HJB equation for a multi-dimensional problem [2], which is simplified by using a single variation parameter to the single dimensional case. These solutions are provided to allow insight into the optimal execution problem to be gleaned.

Section 4 on Markov Models follows Littman's categorization [27]. Splitting Markov Models into categories pertaining to controllability, and observability. Controllability refers to the ability of an external actor to influence state transition. Observability, to whether an actor is able to see the model state space. The categories are divided such that Markov Chains are defined as non-controllable, non-observable Markov Models, MDP's as controllable, observable. POMDP as controllable, non-observable, and HMM's as non-controllable, non-observable. We provide background to each model, introducing Q-learning, an area of reinforcement learning that allows a system to optimize itself based on reward and punishment signals, in which we implement trading models for each of the Markov Model categories. We provided sample trading solutions for the MDP, POMDP, and HMM formulations, the models are built using a Geometric Random Walk as the pricing model. This approach is used as we model over a large number of steps and wish to avoid a negative stock price. Arithmetic Random Walk pricing models are used in the existing Section 3 formulations due to their analytic tractability. With our numerical approach the more practical Geometric Random Walk pricing models are used instead.

2 Execution Domain

This chapter introduces the various facets of the optimal execution problem; the trading strategy, the pricing model and the conditions for defining optimality. A trading strategy is the definition of a trade schedule, the decomposition of a large order into smaller packages which are submitted to the market over a period of time. The definition of an efficient trading strategy is important as a cost will be incurred during the execution of the trade package list which is dependent on the distribution of the order volume across the trade list. That is the volume may be weighted to trade more heavily at the start of the transaction, to reduce risk, weighted evenly across the transaction, or weighted to trade more heavily towards the end of the transaction to possibly maximize return. Alternatively the rate of trading may be varied to suit changing market conditions, distributing the volume of trades over the execution period dynamically, and adapting the trading rate online over the execution period. The pricing model defines the price of an asset across the transaction duration. It is central to the understanding of cost, as an asset price may not only vary as a result of random market conditions and serial information, but as a result of the trading strategy as assets are acquired or liquidated in accordance with market supply/demand dynamics. The pricing models used include a market impact term, which accounts for the effect of the executed volume on the asset price. Market impact is key to the optimal execution problem. This chapters sections introduce the pricing term, market impact, and summarizes some existing formulations used in the literature which shall be built upon within later chapters. It is also instructive to understand how various authors perceive what is significant to the execution problem. Implementation Shortfall [31] is then introduced, it is one of the defining elements of optimal execution. Historically it has been used to conceptualize the problem. Implementation shortfall leads us directly into the definition of optimal execution. The optimality condition is an important concept as it directly relates to the value functions used in later chapters to solve the execution problem in an MDP setting.

2.1 Trade Strategy

A trading strategy is the realization of an investment decision through the execution of buy/sell orders over a period of time. The placement of orders over time defines a trading trajectory that projects a move from one capital position to another. The ultimate goal for optimal execution is to produce a trading strategy that is optimal according to a specified criteria. In the literature the common optimality criteria is either to minimize the expected cost of execution, or minimize the mean-variance of the cost, that is the expected cost of execution, given a traders risk profile manifesting itself as a cost variance sensitivity.

Numerically the definition of a trading strategy is the execution of L units of listed asset over a time period $[0, T]$, by definition both L and T are fixed. Fixing a trading strategy to terminate at T however may in itself be sub-optimal, since if there is sufficient liquidity it may be optimal to execute immediately, or over a shortened time period in accordance with a mean variance minimization, hence the trade residual can become zero prior to T. In the discrete setting T is divided into N intervals indexed by k each with duration $\tau = T/N$ such that the trading trajectory is given by

$$\left\{ n_k : k \in [0, N], n_{\{0<k\leq N\}} = 0, \sum_{k=0}^{N} n_k = L \right\}$$

where the trade list is $\{n_k : n_k = l_{k-1} - l_k\}$, l_k is the residual volume or 'leaves quantity' at time index k, L is the total volume to be executed at $k = 0$, we also explicitly note that $n_{\{0<k\leq N\}} = 0$ as the trade package size must be equal to the leaves quantity at time $k = 0$ but may then be 0 at any time point thereafter [30]. Although the discretized model infers that a trade list package be executed at a given time index, in actual fact it may simply be executed over the duration τ. This is the practical reality, the trading strategy logic is decoupled from the market sitting above an order randomization and optimization layer, necessary to camouflage order placement to protect a trading strategy from gamers; opportunistic traders that would front run large execution strategies and surf a wave of risk free profit given the opportunity. Optimizers in turn change the placement of the orders to better fit the market, optimizing the order submission queues into the market by splicing orders into the limit order book, taking into account market microstructure dynamics, e.g. pegging to best. The continuous setting can be used to provide a simpler analytical environment in which to solve a given problem - this is then translated into a discrete trading model when implemented.

An additional caveat is that buy strategies are normally defined to only place bids, and sell strategies to place asks, although this simplifies the problem mathematically, in the investment banking world if a customer wants a trading strategy that buys/sells or buys/sells/buys, then they can have one. Trading strategies can additionally be defined as static or dynamic. Static strategies define the trading list up front, the trading list is fixed for the duration of the order placement $[0, T]$. Dynamic strategies take into account information up to and including the time point that the size of a trade list member indexed at the given time point is calculated [4].

2.2 Pricing Model

There are numerous pricing models and laws of motion that govern asset price movement, which attempt to capture the dynamics of asset prices as trades are executed against their listed volumes. These models provide the basis for the optimization's that are presented in later sections. Predominately the models are Bachelier type arithmetic random walks as considered by Bertismas and Lo [14], all of the Almgren models without drift [3][4][9][5][8][7][2] and with drift [6], and Obizhaevva and Wang[30]. The basic form of the arithmetic random walk pricing model during execution is

$$S_{k+1} = S_k + \sigma W_k + [\mu\tau] + \mathit{impact} \tag{1}$$

where S_k is the initial period stock price, S_{k+1} the stock price at interval end, W_k is a one dimensional standard Brownian motion under $\mathbb{P}$ a standard probability measure on S, μ the drift and τ the interval duration. The drift term is enclosed in square brackets as it is an optional feature used in some formulations but not in others. *Impact* is the effect on the price as a result of executing against the market. Geometric Brownian motions are considered for such models but are less popular due to their tractability with the optimizations considered. The benefits of using a geometric Brownian motion over an arithmetic random walk are that the geometric Brownian motion remains positive, whereas the arithmetic random walk could produce a negative stock price which is not possible in the real world. However arithmetic Brownian motion models are often chosen due to the short term duration of the problem, the benefits of tractability outweigh the negative aspects of the model.

In general the pricing model, Equation (1), is built up from several components the $k = 0$ asset price S_0, a noise term σW_t that is representative of the trading activities of uninformed traders, a drift term μt that represents the informed traders, and a price impact component (discussed in section 2.2.1).

The activity of the uninformed traders may be implied from the volatility of the asset price over a relatively short period. The trading activity of the informed traders, results from the investment decisions that are made prior and during the trading day, which are executed throughout the day. The aggregated weight of the activities of the informed traders, trading activity causes drift in the asset price [6]. In addition serial correlation of market information, asset price and the correlation effect of pairs in the multivariate case may also be influential factors in the pricing model, depending on the problem being analyzed. The above mentioned Bachelier type model covers the evolution of the stock price S_0 with regards to a Brownian motion σW_k with drift $\mu\tau$. The component that has not yet been addressed is market impact, which is deserving of it's own section as it is a feature of the pricing model that is central to trade execution.

2.2.1 Market Impact

Modern financial markets have evolved from the open outcry systems of yesteryear to the low latency exchanges of today. Orders that are posted to the market may be meant for immediate execution (Immediate or Cancel, Fill or Kill) or for execution at a given limit price. Limit orders, as they are known, are maintained within a Limit Order Book (LOB) controlled by the exchange. The LOB has two sides, a buy side and a sell side. The buy (sell) side contains the orders posted by clients that wish to buy (sell) at a specified limit price. LOB is composed of all limit orders posted to the market that have not been matched. A match occurs when the buy side and sell side order prices overlap resulting in a trade, on execution the executed order volume is effectively lifted out of the LOB.

Market impact is the instantaneous and temporal perturbation of the asset price caused by a trade execution. It represents a key component in the modelling of the actual price received for an asset on a trade, which we shall denote $\tilde{S}$ the weighted average price on the trade. On arrival orders are queued per symbol (asset), commonly prioritized by price and by time of arrival, in buy side and sell side lists, or legs. The length of the buy/sell side legs provide the depth of the order book. To fully understand price impact we must look at the structure, the composition of price level order volume in the LOB, elasticity of the LOB, or the tendency of the LOB to maintain its previous composition, in which it is the limited elasticity of supply and demand which ultimately govern the price of assets [30]. A commonly held view is that asset prices change mechanistically as a result of imbalances between the buy and sell side [1]. The number of limit orders on the bid and ask side of the LOB at any given price S is $q_{S^+}(S)$ and $q_{S^-}(S)$ respectively, S^+ is the best ask price and S^- the best bid price, $S^+ > S^-$. During normal trading $q_{S^+}(S) \geq 0$ when $S \geq S^+$, $q_{S^+}(S) = 0$ otherwise, and $q_{S^-}(S) \geq 0$ when $S \leq S^-$, $q_{S^-}(S) = 0$ otherwise. The mid/centerprice is defined as $M = (S^+ + S^-)/2$ and the spread $m = S^+ - S^-$ [30]. Given a reference point at one side of the spread the near touch price is the best price at the top of the order book at the point of reference, the far touch price is the best price on the opposing side of the spread. The order volume over a price interval $S + dS$ is defined by Obizhaeva. A, and Wang. J, to have a simple block shape [30] but may take a more complex form. An order posted to the LOB will sweep through the price levels on the opposing side, starting at the far touch price until its limit price is met or until its volume has been filled. If sufficient liquidity is available, the resulting trade execution value is given by the following equation

$$\int_{S_k^+}^{S_{k+1}^+} q(S)dS = p_k \tag{2}$$

where k is the discrete time index as previously defined on T, p_k is the execution value of the trade.

The resilience of the LOB is the speed of recovery, the time that it takes to be rebuilt to its original shape or structure. The price momentum is governed by the elasticity of the LOB, the price momentum is the pressure on the centerprice of the LOB. The centerprice is also subject to volatility σ and drift μ. Considering the centerprice as opposed to the last traded price smooths the saw tooth path of buy/sell trades that cross the respective spreads to match on the opposing side through a trading day. It also allows us to focus on the price without necessarily considering the variance in the width of the spread. Within this framework for a buy trade the actual price received is

$$\tilde{S}_{k+1}^+ = M_k + m/2 + p_k/(2q). \tag{3}$$

The dynamics are such that the ask price moves by

$$S_{k+1}^+ = M_k + m/2 + p_k/q \tag{4}$$

and a trade will shift the centerprice linearly with respect to the size of the trade, $M_{k+1} = M_k + \theta p_k$, where θp_k gives the permanent impact of the trade with $q \leq \theta \leq 1/q$. The convergence of the centerprice back to its M_k value is given by subtracting the post trade ask price $S^+_{k+1} = S^+_k + p_k/q$, from the steady state price $S^+_\infty = S^+_k + \theta p_k$, which yields $S^+_{k+1} - S^+_\infty = p_k(1/q - \theta)$. Now assuming that the order book converges exponentially in time to its original structure, $q_k(S) = q1_{\{S \geq S^+_k\}}$where $\tilde{S}^+_{k+1} = S^+_k + p_k\Theta e^{-\iota k}$. Thus a trade execution of price p_t, has an impact function of $\Psi(p_t)$ and recovery function of $\varrho(p_t)$

$$\Psi(p_t) = \theta \sum_{k=0}^{n(t)} p_{t_k} \tag{5}$$

$$\varrho(p_t) = \sum_{k=0}^{n(t)} p_{t_k} \Theta e^{-\iota(t-t_k)} \tag{6}$$

the recovery time is given by ι which is a measure of the resilience of the LOB [30]. Studies by Alfonso, Fruth and Schied in their paper 'Optimal Execution strategies in limit order books with general shape functions' extend this framework using more complex LOB density functions to allow nonlinear impact functions producing a LOB with a greater rage of equilibrium dynamics [1].

Throughout the literature the dynamics of market impact are often simplified to a permanent and temporary component. Permanent impact is the impact that persists for the duration of the trade strategy effecting all future prices. Temporary impact is an impact that has dissipated by the time of the next trade, effectively decaying to 0. The combination of impact functions and the complexity of the impact functions are usually selected to produced pricing models that are tractable and descriptive for a given methodology used in producing an optimal execution strategy.

Bertsimas and Lo use a single linear permanent impact function of form $\theta(L - l_k)$, where θ is the permanent impact parameter, and $(L - l_k)$ the volume of the trade executed up to a point k [14]. They later extend this model into the multivariate setting, in which impact costs of assets executed within a single portfolio may impact each other, due to correlation effects of price movement between the assets [13]. Almgren [4] uses linear permanent and temporary impact functions before moving to nonlinear permanent and temporary impact functions. As the linear functions were not considered realistic in a real world sense, in addition a further temporary impact function was also introduced to represent the uncertainty of trade execution, with the effect of amplifying variance [9]. The later work by Almgren for convenience drop the permanent impact function altogether [7].

The structure of the Almgren nonlinear model treating market impact as a power law function, which encompasses the linear case, yields permanent and temporary impact functions of form

$$\Psi(n_k/\tau) = \theta v^{\beta^p} \tag{7}$$

$$\psi(n_k/\tau) = \omega v^{\beta^t} \tag{8}$$

where $v = n_k/\tau$ can be interpreted as the trading velocity. In this formulation θ is the permanent impact parameter as previously defined andω the temporary impact parameter. The order of the power law functions are given by β^p for permanent impact and β^t for temporary impact. In the paper 'Direct Estimation of Equity Market Impact', (2005), Almgren et al, fit the nonlinear model to a real world equity data set provided by Citigroup US. Calibration of the model is subject to the data set being cleansed, so as to only encompass substantial proprietary trades across liquid asset to measure cost of trading. The aim was to provide a trading model that would be used directly as the basis for a trade scheduling algorithm, within the Citigroup Best Execution Consulting Services (BECS) software, to allow costs to be estimated and controlled.

It is interesting to note that the Obizhaeva and Wang, impact functions Equation (5), align with the nonlinear temporary impact functions used by Almgren, Equation (8), in the limiting case when the time between trade packet submission is of sufficient duration that the asset price is able to return to its resting equilibrium [30].

2.2.2 Model Formulation

Within this section there is a brief review of some of the pricing models most commonly used for the optimization of execution. Ultimately the pricing models are the sums of assumptions and trade offs used to provide a tractable model that is mostly representative of the real world asset price trajectory across a trading day. Once selected the chosen pricing model is used to formulate an optimal trading strategy with exogenous interaction with the asset. Depending on the practitioner the pricing models tend to be selected for tractability for a given mathematical methodology used to formulate an optimal trading strategy. It should always be remembered that the goal of any pricing model is to explain phenomena present in real markets, however, given that there may be a great deal of variability between assets or even within the same asset over time [14] there is plenty of worth in these models from a theoretical sense as they allow the optimal execution trading problem to be set, analyzed, simulated and fitted to real market data. The models are shown in Euler discretized form, and contrast there salient features. The Euler discretization admits a closed form expression for the transition density of the price dynamic in the Markov models, higher order discretization schemes such as Milstein could be considered with additional complexity for the MDP problem.

Model 1: Bertismas and Lo 1. The first model presented by Bertismas and Lo is a simple arithmetic random work with a single permanent impact term

$$S_{k+1} = S_k + \theta n_{k+1} + \sigma W_{k+1} \tag{9}$$

[14] where S_k is the asset price, θ the market impact parameter with $\theta > 0$, n_{k+1} the order size at time index $k+1$, σ the volatility and W_{k+1} a standard one dimensional Brownian motion. This can be considered one of the most simplistic models for the execution problem. It serves as a good starting point from which naive trading strategy can be obtained [14].

Model 2: Bertismas and Lo 2. The second model presented extends the previous model with the addition of an information component, which is auto regressive with lag 1

$$\begin{aligned} S_{k+1} &= S_k + \theta n_{k+1} + \varsigma I_{k+1} + \sigma W_{k+1} && (10) \\ I_{k+1} &= \rho I_k + \sigma' W'_{k+1} && (11) \end{aligned}$$

[14], where I_k is the auto regressive information term, ς the effect of the term on S, ρ the memory of the process, σ' the volatility of the information term with W'_{k+1} a standard one dimensional Brownian motion. Execution optimization tends to be robust in the variability exhibited by the information term used to represent market information. Almgren notes that the information term as being largely insignificant in optimal execution models due to the minimal impact [4]. However, it does introduce a latent process, a key component of a

Hidden Markov Model.

Model 3: Bertismas and Lo 3. Bertismas and Lo also introduce a Linear Percentage Temporary (LPT) model, which they considered more representative of the actual market, for the evolution of an asset price as it is constructed from a geometric Brownian motion. The impact and information components from previous models are maintained

$$\begin{cases} S_{k+1} = & S_k exp(\sigma W_{k+1})(1 + \theta n_{k+1} + \varsigma I_{k+1}) \\ I_{k+1} = & \rho I_k + \sigma' W'_{k+1} \end{cases} \tag{12}$$

[14].

Model 4: Almgren 1. This model contains both permanent and temporary impact components, with impact terms that are functions of trade velocity

$$\begin{cases} S_{k+1} = & S_k + \sigma W_{k+1} + \tau \Psi(n_{k+1}/\tau) \\ \tilde{S}_{k+1} = & S_k + \psi(n_{k+1}/\tau) \end{cases} \tag{13}$$

[4], n_{k+1}/τ represents trade velocity as defined in section 2.2.1 for the market impact Equations (7) and (8). Where $\Psi(n_{k+1}/\tau)$ represents the permanent impact and $\psi(n_{k+1}/\tau)$ the temporary impact.

Model 5: Almgren 2. The next model excludes the effect of permanent impact and includes a drift term. The drift is included to simulate the effects of other informed traders

$$\begin{cases} S_{k+1} = & S_k + \tau\mu + \sigma W_{k+1} \\ \tilde{S}_{k+1} = & S_k + \psi(n_{k+1}/\tau) \end{cases} \tag{14}$$

[6].

The model simply reflect an asset price with a arithmetic random walk, drift and a temporary impact term. For presenting the existing formulations a mix of these models are used.

2.3 Implementation Shortfall

The implementation of trading decisions may incur an implementation cost or shortfall composed of both execution costs and opportunity costs [31]. Execution costs are related to the actual transaction and can have both direct and indirect components. The direct component is composed of the commissions and fees levied on a transaction. The indirect costs are incurred from the inability of a market to be able to fully absorb an execution at a given price level [8]. Opportunity costs are related to not transacting or queuing transactions for execution over a given duration. Capital employed for one purpose may not be employed for another purpose.

In general implementation shortfall is die polar, with execution cost at one pole and opportunity cost at the other. A decrease in one cost is usually at the expense of an increase in the other. Implementation shortfall is the measure of the cost of transacting, it is in essence the cost of the real world realization of an investment decision [31]. Within this paper we address only the minimization of the indirect component of the execution cost, as the direct component is easily determined, and we treat the opportunity cost as a subjective component of the investment decision, hence all references to implementation shortfall or execution cost refer only to the indirect cost of transacting.

With the above definition in mind it is useful to consider the implementation shortfall as being the difference between a theoretical or hypothetical benchmark best price and the actual price of execution. The theoretical price of transacting a given volume L of a market listed asset instantaneously with best price S_0 is LS_0, at the start $k = 0$ of the execution. Referring back to the proceeding sections 2.1 and 2.2.1, the actual price of execution may differ from the theoretical price as the market may not be able to absorb a transaction instantaneously at the best price. Therefore the transaction may need to be broken up into packages n_k executed over a time duration T indexed by k yielding a volume weighted price for each package transaction of $\tilde{S}_k$, the trade value or price paid for a given trade is thus $p_k = n_k\tilde{S}_k$. The actual price received is the sum of the price of each package executed at each time index $\sum n_k\tilde{S}_k$. The difference between these two prices provides us with the total cost of trading [4].

$$C = |LS - \sum n_k\tilde{S}_k|. \tag{15}$$

Within the continuous time setting the actual price received over the transaction traded at an execution rate of $v(t) = -dl/dt$ is given by $\int v(t)\tilde{S}(t)dt$. The rate of execution is negative as our starting volume at $l_0 = L$ and our ending volume at $k = T$ is $l_T = 0$. This gives us the total cost of trading [7].

$$C = |LS - \int_0^T v(t)\tilde{S}(t)dt|. \tag{16}$$

The implementation shortfall is mathematically defined as the cost of deviation from the initial price actuated by a trade execution, it is the measure that we use in our quest for best execution, the optimal trading strategy employed to minimize execution cost [14]. To fully understand how to minimize implementation shortfall we must introspect the components that define cost, we must define the price model $\tilde{S}$ the trading strategy that defines the rate of execution $v(t)$ and the effects of the duration of the transaction T.

Discrete and continuous representations are given. In discrete formulations change is modelled using recurrence relations. In continuous formulations change is modeled using calculus, a function is modelled as an infinitesimal momentary change at a point in time. Continuous mathematics is considered to be more theoretically rich [16] and hence is often used in preference to discrete mathematics due to its tractability, continuous formulations may be mapped to discrete formulations and vice versa. When a problem is simulated or otherwise realized in a computational context the discrete formulation must be used, hence a problem is often solved using continuous tools then mapped to a discrete representation for simulation.

2.4 Optimality Criteria

An optimal trading strategy is the trading list for a buy (sell) strategy that has the minimum associated mean cost, or minimum mean-variance cost, depending on whether trader utility λ is considered as a factor, it may be obtained by solving the following minimizations (maximizations). For buy strategies we will only need to minimize the total actual price paid on the transaction. For sell strategies we maximize the actual price received see Section 2.3 for definition of actual price. The following equations are for a buy strategy, they give the minimum mean, mean-variance cost for the transaction

$$min_{n_k \in [l_k, 0]} \mathbb{E}_{\mathbb{P}(S_k)} \left[\sum_{k=1}^{N} \tilde{S}_k n_k \right] = \sum_{k=1}^{N} \int_{-\infty}^{\infty} \phi(S_k) P(S_k | S_{k-1}) dS_k \tag{17}$$

$$min_{n_k \in [l_k,0]} \mathbb{E}_{\mathbb{P}(S_k)} \left[\sum_{k=1}^{N} \tilde{S}_k n_k \right] + \lambda Var_{\mathbb{P}(S_k)} \left[\sum_{k=1}^{N} \tilde{S}_k n_k \right]. \tag{18}$$

Solutions have been achieved for various pricing models with differing impact functions dependent on the various author preferences for mean cost using stochastic dynamic programming in the form of the Bellman equation [13][30], for mean/variance cost directly [4],[9] Hamilton-Jacobi-Bellman stochastic dynamic programming [2], and using Bayesian methods [6]. We will aim to provide additional solutions using Markov Models, the topic of this thesis.

2.5 Summary

Within this section we have covered the basics of the trading environment, the trading problem is outlined introducing the concept of trading scheduling, a central them that sits at the core of all trading solutions. The most popular pricing models are reviewed, with particular emphasis being given to market impact a new term peculiar to the execution problem. Then cost and optimality are considered, these are the driving forces which are used to determine an optimal solution. We now move on to looking at the most significant existing solutions to the optimal execution problem.

3 Existing Approaches

Within this chapter we review existing solutions to the optimal execution problem. This allows us to better understand the problem by exploring it from a number of different perspectives. The solutions outlined follow analytical and dynamic programming approaches. In the analytical case solutions are provided for static trading strategies [4], and for dynamic trading strategies [7]. In the dynamic programming case both discrete [14] and continuous [2] solutions are provided. It is worth first considering the topic of dynamic programming in greater depth as it serves to provide some continuity into the Markov formulations presented in the following chapter. The solution outlines are then provided indexed historically, as it is interesting to see the evolution of the exploration of the problem.

3.1 Dynamic Programming

Dynamic programming is a methodology for solving complex problems, that is central to many of the existing formulations. The technique is based on the principle of optimality, which breaks a problem up into a number of sub-problems and then recursively solves them. This is achieved by backward substitution of the end time subproblem into the previous time subproblem and then continuing this process recursively until an optimal solution for the entire problem has been constructed. The principle allows a problem to be broken up and solved as a set of smaller less complex problems. The area was first introduced by Richard Bellman in the 1950's [12].

Many of the existing optimal execution formulations are treated as optimal control problems, reliant on state space techniques for their solutions. State space approaches can be either time-varying or time-invariant, over a non-infinite time interval or over an infinite non terminating duration [19]. In the non-infinite time horizon case the state space approach may be thought of as a system with an input vector and an output vector(u, y), where $u \in U \subset \mathbb{R}^n$and $y \in Y \subset \mathbb{R}^n$over $T \subset \mathbb{R}^n$. The state of the system describes the present and future input output pairs based on past behavior, realized through a state vector x defined on the subset $X \subset \mathbb{R}^n$, to which the trajectory is confined. The state vector x is composed of the minimum set of state variables required to describe the system. The time varying state space equations can thus be defined as

$$\begin{cases} \dot{x} & = A(t)x + B(t)u \\ y & = D(t)x \end{cases}. \tag{19}$$

The matrices A,B and D are square matrices. The state space equations describe the relationship between the input and output vector. For the initial state problem there exists a unique solution in the form of the integral equation

$$x(t) = \phi(t, t_0)x(t_0) + \int_{t_0}^{t} \phi(t, \tau)B(\tau)u(\tau)d\tau \tag{20}$$

where $\phi(s, t)$ is the state vector transition property. The state space equations can be seen as a control observation feedback process, in which the control vector $u = u(t)$, influences the state function $x = x(t)$ over its trajectory. The trajectory of $x(t)$ is defined by the above integral equation. The state space problem is defined as being controllable if it can be returned to the origin in a finite amount of time by the control function after starting from an initial state vector position x_0 at an initial time 0. That is the given integral equation has a solution u in the class of admissible control functions. The transfer function is $h(t)$, where

$$y(t) = \int_{t_0}^{t} D(t)\phi(t, s)B(s)u(s)ds = \int_{t_0}^{t} h(t, s)u(s)ds. \tag{21}$$

To define optimal control we must first introduce a scalar valued function $C(x, u, t)$ defined on $X \times U \times \mathbb{R}^n$, such that the cost function is defined as

$$V(u) = \int_{t_0}^{t_1} C(x, u, t)dt. \tag{22}$$

This becomes a Lagrange problem if x depends on u. Furthermore, the terminal time is limited to a non-infinite duration e.g. is considered over a subset of $\mathbb{R}^n$ for the methodologies considered. The optimal control problem is then reduced to finding the optimal control function $u^\star$ and the optimal trajectory $x^\star$, the pairs $(u^\star, x^\star)$ define the optimal control and trajectory pair over T. This leads us to the Bellman equation and the value function defined by

$$V(\tau, x, u) = min[\int_{t_0}^{t_1} C(x, u, t)dt : u \in U(\tau, y)] \tag{23}$$

where $V(\tau, x, u)$ is the optimal value obtainable by minimizing the function subject to constraints. This may be expanded, through the use of the optimality principle, to establish the method of dynamic programming. The optimality principle over the optimal control trajectory pair requires for any t and τ, where $t_0 \leq t < \tau < T$, that

$$V(t, T, x(t), u(t)) = min_{u \in U(t, x^{\star}(t))}[\int_t^{\tau} C(x, u, s)ds + V(\tau, T, x(\tau), u(\tau))]. \quad (24)$$

The discrete from of Equation (24), is used by Bertismas and Lo [14] to minimize expectation of cost. Considering Equation (24), taking $\tau = t + \epsilon$, where ϵ is a small time increment, and expanding $V(\tau, x(\tau), u(\tau))$ using a Taylor expansion, allows us to formulate the Hamilton-Jacobi-Bellman (HJB) equation over $t_0 \leq t \leq T$, used in Section 3.2.4.

$$\begin{cases} 0 &= \frac{dV}{dt}(t, x^{\star}) + [\frac{dV}{dx}(t, x^{\star})]f(x^{\star}, u^{\star}, t) + C(x^{\star}, u^{\star}, t) \\ 0 &= V(T, x^{\star}(T)) \end{cases} \quad (25)$$

$$V(t, T, x(t), u(t)) = min_{u \in U(t, x^{\star}(t))}[C(x^{\star}, u, t) + [\frac{dV}{dx}(t, x^{\star})]f(x^{\star}, u, t)]. \quad (26)$$

Removing the $T \subset \mathbb{R}^n$limitation facilitates the infinite time horizon approaches considered at a future point.

3.2 Solution Outlines

3.2.1 Bertismas and Lo - Bellman Equation.

The initial Bertismas and Lo 1998 model [14] assumes the liquidation of a single block of trades L over a period T of equally spaced increments with a given price impact function. The investors objective is to minimize cost by finding the sequence of trades that minimizes the following cost equation as given in Section 2.4

$$V_k(\tilde{S}_k, l_k) = min_{n_k \in [l_k 0]} \mathbb{E}_{\mathbb{P}(S_k)} \left[\sum_{k=1}^{N} \tilde{S}_k n_k \right] \quad (27)$$

with V_k used to represent the value function or utility, subject to the constraints $\sum_{k=1}^{N} n_k = L$. This can be posed as a dynamic programming problem within the Bellman formulation [12]. Using model 1 from Section

2.2.2 [14]

$$S_{k+1} = S_k + \theta n_{k+1} + \sigma W_{k+1} \tag{28}$$

where S_k is the asset price, θ the market impact parameter with $\theta \geq 0$, n_k the order size at time index k, σ the volatility and W_k a standard one dimensional Brownian motion. Noting that l_k is used to represent the number of shares that remain to be purchased e.g. $l_k = l_{k-1} - n_{k-1}$, with the boundary conditions $l_1 = L$, and $l_{N+1} = 0$. The discrete-time Bellman formulation yields

$$\begin{cases} V_k(\tilde{S}_{k-1}, l_k) & = min_{n_k \in [l_k 0]} \mathbb{E}_{\mathbb{P}(S_k)} \left[\tilde{S}_k n_k + V_{k+1}(\tilde{S}_k, l_{k+1}) \right] \\ V_{N+1}(\tilde{S}_k, l_{k+1}) & = 0 \end{cases} . \tag{29}$$

Solving recursively at $k = N$, and noting that $l_{N+1} = 0$,

$$V_N(\tilde{S}_{N-1}, l_N) = min_{n_k \in [l_k 0]} \mathbb{E}_{\mathbb{P}(S_N)} \left[\tilde{S}_N l_N \right] = (\tilde{S}_{N-1} + \theta l_N) l_N \tag{30}$$

$$n_N^* = l_N \tag{31}$$

e.g. execute all remaining volume. Continuing the recursion approach at $k = N - 1$

$$\begin{aligned} V_{N-1}(\tilde{S}_{N-2}, l_{N-1}) &= min_{n_{k-1} \in [l_{k-1} 0]} \mathbb{E}_{\mathbb{P}(S_{N-1})} [\tilde{S}_{N-1} n_{N-1} + V_N(\tilde{S}_{N-1}, l_N)] \\ &= \begin{matrix} min_{n_{k-1} \in [l_{k-1} 0]} \mathbb{E}_{\mathbb{P}(S_{N-1})} [(\tilde{S}_{N-2} + \theta n_{N-1} + \epsilon_{N-1}) n_{N-1} \\ + V_N(\tilde{S}_{N-2} + \theta n_{N-1} + \epsilon_{N-1}, l_{N-1} - n_{N-1})] \end{matrix} \\ &= min_{n_{k-1} \in [l_{k-1} 0]} [\tilde{S}_{N-2} l_{N-1} + \theta l_{N-1}^2 - \theta l_{N-1} n_{N-1} + \theta n_{N-1}^2] \end{aligned}$$

$$V_{N-1}(\tilde{S}_{N-2}, l_{N-1}) = l_{N-1}(\tilde{S}_{N-1} + \frac{3}{4} \theta l_{N-1}) \tag{32}$$

$$n_{N-1}^* = \frac{l_{N-1}}{2} \tag{33}$$

where n_{N-1}^* is found by direct convex optimization, as the function $f = \tilde{S}_{N-2} l_{N-1} + \theta l_{N-1}^2 - \theta l_{N-1} n_{N-1} + \theta n_{N-1}^2$ is quadratic on n_{N-1}, taking the derivative of the function with respect to n_{N-1} and setting it to 0 yields n_{N-1}^*. Substituting n_{N-1}^* back into the function yields V_{N-1}. This recursion may be continued, leading to the result

$$n^*_{N-k} = \frac{l_{N-k}}{(k+1)} \tag{34}$$

$$V_{N-k}(S_{N-k-1}, l_{N-k}) = l_{N-k}(S_{N-i-1} + \frac{k+2}{2(k+1)}\theta l_{N-k}). \tag{35}$$

Finally at $k = 1$,where $l_1 = L$

$$V_1(S_0, l_1) = E_1\left[\sum_{k=1}^{N} \tilde{S}_k n^*_k\right] = S_0 L + \frac{\theta L^2}{2}(1 + \frac{1}{N}). \tag{36}$$

$$n^*_1 = \frac{L}{N} \tag{37}$$

Hence, by back substitution, the optimal trading strategy is

$$n^*_1 = n^*_2 = ... = n^*_N = \frac{L}{N} \tag{38}$$

This result follows from the fact that the market impact is permanent, V only depends on trade size in any given interval, it is independent of the other intervals. The execution cost is reduced by increasing the number of intervals, however the execution cost can never reach 0. The execution cost minimization it achieved across all intervals by minimizing the convex value function, e.g. this minimum is achieved where the marginal costs are equal [14]. This approach is termed the 'naive' strategy. It is not suitable for real world application, however it is useful to benchmark against. We have presented this strategy as it clearly shows the application of the Bellman equation to find an optimal solution, and yields the naive strategy which we bench mark against using our own MDP strategies in a later chapter.

3.2.2 Almgren and Chriss - Analytic.

The paper 'Optimal Execution of Portfolio Transaction's by Almgren and Chriss [4], expands on earlier works introducing trader utility. Trader utility provides a selection criteria between the trading extremes of trading everything up front with high cost but no uncertainty, or over an extended period with minimum cost but extreme levels of uncertainty. It is the selection point or an efficient frontier which traces the trading strategies with minimum expected cost for a given level of uncertainty. The optimality criteria used within the formulation as described in Section 2.4 is

$$V_k(\tilde{S}_k, n_k) = argmin_{n_k \in [l_K, 0]} \mathbb{E}_{\mathbb{P}(S_k)} \left[\sum_{k=1}^{N} \tilde{S}_k n_k \right] + \lambda Var_{\mathbb{P}(S_k)} \left[\sum_{k=1}^{N} \tilde{S}_k n_k \right]. \quad (39)$$

The solution is a static trading strategy optimization, as defined in Section 2.1. The model is extended to piece-wise static strategies when unanticipated market events are occur. Recalculating the trade schedule when market conditions change provides a rudimentary dynamic trading strategy. Using Section 2.2.2, model 4 [4]

$$\begin{cases} S_{k+1} = & S_k + \sigma W_{k+1} + \tau \Psi(n_{k+1}/\tau) \\ \tilde{S}_{k+1} = & S_k + \psi(n_{k+1}/\tau) \end{cases} \quad (40)$$

where S_{k+1} is the asset price, n_{k+1} the order size at time index $k+1$, σ the volatility, W_{k+1} a standard one dimensional Brownian Motion, τ the time interval between decision points, permanent impact represented by $\Psi(n_{k+1}/\tau)$, and temporary impact by $\psi(n_{k+1}/\tau)$, as previously specified in Section 2.2.2. The implementation shortfall [31] or the total cost of trading is given by

$$LS_0 - \sum n_k \tilde{S}_k \quad (41)$$

where by substitution of Equation (40) into Equation (41) gives

$$\sum n_k \tilde{S}_k = LS_0 + \sum_{K=1}^{N} (\sigma W_k + \tau \Psi(n_k/\tau)) l_k + \sum_{k=1}^{N} n_k \psi(n_k/\tau) \quad (42)$$

which yields the expected cost of trading

$$\mathbb{E}_{\mathbb{P}(S)}(C) = -\sum_{k=1}^{N} \tau l_k \Psi(n_k/\tau) - \sum_{k=1}^{N} n_k \psi(n_k/\tau) \quad (43)$$

and the variance

$$Var_{\mathbb{P}(S)}(C) = \sigma^2 \sum_{k=1}^{N} \tau l_k^2. \quad (44)$$

Almgren and Chriss explore the idea of efficient frontier the trading strategy in which the expected cost has no lower variance and vice versa e.g. $min_r(\mathbb{E}_{\mathbb{P}(S)}(C) + \lambda Var_{\mathbb{P}(S)}(C))$ where λ is the Lagrange multiplier, in this trading context a measure of risk aversion. This then yields when solving for the minimum impact,

$$V_1(S_0, l_1) = \mathbb{E}_{\mathbb{P}(S)}\left[\sum_{k=1}^{N} \tilde{S}_k n_k^*\right] = S_0 L + \frac{\theta L^2}{2}(1 + \frac{1}{N}) + L\psi(L/N) \tag{45}$$

aligning with the solution provided by the Bertismas and Lo simple formulation, but with the addition of a temporary impact term where $\lambda = 0$. The trade schedule is $n_k = L/N$. This is one extreme of the formulation. The second extreme is for the minimum variance which is not handled in the Bertismas and Lo formulation as they do not include utility in their model. The efficient frontiers are then the trajectories within these bounds traced by the minimization of the mean variance with respect to trader utility.

3.2.3 Almgren and Lorenz - Analytic

The later works of Almgren in collaboration with Lorenz address dynamic execution algorithms, answering the question; should a trading strategy be static or dynamic? For example dynamic strategies were seen to adjust the trading strategy, spending trading gains to reduce risk as information becomes available. The paper 'Adaptive Arrival Price', [7] takes price information into account to adapt the trading rate, introducing a negative correlation between past trading gains and future market impact costs. This type of formulation provides an 'aggressive in the money strategy' (AIM) a strategy that would take while the goings good as opposed to a 'passive in the money strategy' (PIM), a strategy that would wait for further gains [24]. Using Section 2.2.2 model 5 [6] without the drift term

$$\begin{cases} S_{k+1} = & S_k + \sigma W_{k+1} \\ \tilde{S}_{k+1} = & S_k + \psi(n_{k+1}/\tau) \end{cases} \tag{46}$$

The implementation shortfall is analyzed in the continuous setting for simplicity of analysis. The implementation shortfall is given by

$$C = \int_0^T \tilde{S}(t)v(t)dt - LS_0 = \sigma \int_0^T l(t)dW(t) + \omega \int_0^T v(t)^2 dt \tag{47}$$

where $v(t)$ is the trading rate $v(t) = n_k/\tau$. The expected cost and variance in cost are given by

$$\mathbb{E}_{P(S)}(C) = \omega \int_{0}^{T} v(t)^2 dt \tag{48}$$

$$Var_{\mathbb{P}(S)}(C) = \sigma^2 \int_{0}^{T} l(t)^2 dt. \tag{49}$$

This has solution $h(t, T, \kappa) = sinh(\kappa(T - t))/sinh(\kappa T)$, where $\kappa = \sqrt{\lambda\sigma^2/\omega}$ termed the urgency parameter is related to the desired time of liquidation. The static trajectory of which reaches $l = 0$ at $T = 0$. The dynamic component is driven by the urgency parameter, which is recalculated at each decision point, that can be simply the half way point $T/2$ or after new market information becomes available. If the urgency parameter κ does not change then the static strategy and dynamic strategy are equivalent this can be seen by observing the solution for the static case and for two, three and by induction an infinite number of product term solutions are equivalent

$$h(t, T, \kappa) = h(s, T, \kappa)h(t - s, T - s, \kappa) = ... = h(i, T, \kappa)h(t - i, T - i, \kappa) \tag{50}$$

where $0 \leq i \leq s \leq t \leq T$.

3.2.4 Almgren - Hamilton-Jacobi-Bellman

'Optimal Trading in a Dynamic Market', [2], by Almgren builds on the earlier paper 'Adaptive Arrival Price' [7], by considering the mean variance criteria, which contain a square of an expectation requiring the use of the HJB partial differential equations to provide a viable dynamic solution.

This paper considers dynamic execution under what Almgren terms 'co-ordinated variation', where liquidity and volatility vary together $\sigma(t)^2\omega(t) = constant$. Almgren argues that coordinated variation is highly relevant in the adaptive execution of smaller cap assets. This is because the trading profile of small cap assets throughout given contains periods of minimal liquidity and periods of high volatility, times when trading is expensive and times when trading is cheap. Coordinated variation is introduced to provide an alternative to solving the HJB equation over a multidimensional space, allowing the problem to be reduced to a single dimension. However it is instructive from a dynamic programming perspective to see the HJB solution. Also it should be noted that the parameters a, b are redefined locally for this model and do not follow the notation definition. The trading cost is given by

$$C = \int_0^T l(t)dW(t) + \sigma^2\omega\int_0^T v(t)^2 dt \tag{51}$$

where Almgren and Chris make an assumption that the amount traded is small enough such that price changes due to market impact are small compared to price volatility, termed the 'small impact' approximation. This allows the expectation of both terms in Equation (51) to be taken, which leads to

$$c(t, l, \omega, \sigma) = min_{v(s), t \leq s \leq T} E\int_t^T (\lambda\sigma(s)^2 l(s)^2 + \omega(s)v(s)^2)ds \tag{52}$$

in which $c(.)$ is a nondimensionalized cost (specific to this formulation) and treating the coefficients as random with

$$\begin{cases} \xi(t) & = log\frac{\omega(t)}{\hat{\omega}} \\ d\xi & = a_\xi dt + b_\xi dW_L \end{cases} \tag{53}$$

and

$$\begin{cases} \zeta(t) & = log\frac{\sigma(t)}{\hat{\sigma}} \\ d\zeta & = a_\zeta dt + b_\zeta dW_V \end{cases} \tag{54}$$

such that $\xi(t)$ and $\zeta(t)$ are nondimensional values that fluctuate around zero, and $a_\xi, b_\xi, a_\zeta, b_\zeta$ are coefficients with values that depend on $\xi(t)$ and $\zeta(t)$. W_L, W_V are correlated Brownian motions independent of W where $E(dW_L dW_V) = \rho dt$. On substitution into the dynamic programming model yields

$$c(t, l, \xi, \zeta) = min_v[\lambda\sigma^2 l^2 dt + \omega v^2 dt + Ec(t + dt, x + dx, \xi + d\xi, \zeta + d\zeta)] \tag{55}$$

where the subscripts on c represent partial derivatives. In the HJB formulation this becomes

$$0 = c_t + \lambda\sigma^2 l^2 + min_v[\omega v^2 - vc_x] + a_\xi c_\xi + a_\zeta c_\zeta + \frac{1}{2}b_\xi^2 c_{\xi\xi} + \rho b_\xi b_\zeta c_{\xi\zeta} + \frac{1}{2}b_\zeta^2 c_{\zeta\zeta} \tag{56}$$

which has a minimum at $v = c_x/2\omega$. This model can be transformed to an Ornstein-Uhlenbeck mean reverting process with relaxation times.

3.3 Summary

Within this chapter we have provided a brief background to dynamic programming and introduced the seminal solutions to the optimal execution problem. The general theme is to provide a simplified model of the problem, focusing on the specific aspects of the system that warrant further exploration and then to find an analytic or dynamic programming solution. The problem must often be set up in such a way that an analytical solution can be produced. These solution techniques provide insights into the optimal execution problem, exposing various aspects of each of the models and solution techniques, such as boundary conditions and possible simplifications.

Obizhaeva and Wang 2006 see optimal strategy as consisting of an initial discrete trade, followed by a sequence of continuous trades, this form of impulse strategy is focused on driving the resting price of the LOB away from its point of equilibrium and then absorbing the new liquidity attracted to the market the strategy is concluded by a final terminating discrete trade when further effects on the LOB are no longer of concern[30].

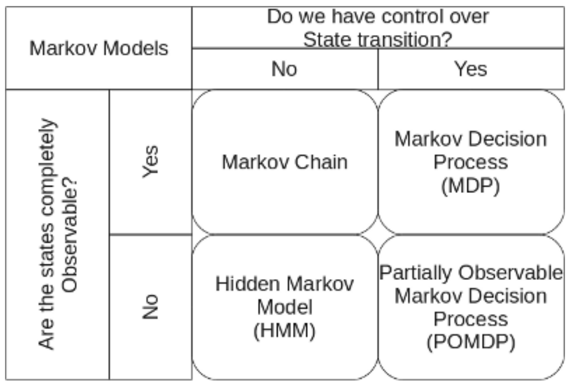
One common trait that can be seen through this section is that the model must be manipulated for an analytical solution. What about more complex models for which an analytical solution may not exist? Is there a more general framework in which the optimal execution framework may be considered? This question leads us directly into the next chapter on Markov models.

4 Markov Models

Markov models, are a class of models satisfying the Markov property, which states simply that the transition from one state to another is independent of past behavior, that is given the past and present state, future states only depend on the present state

$$p(x_{k+1}|x_k) = p(x_{k+1}|x_{1:k}). \tag{57}$$

Markov models themselves can further be categorized as Markov Chains, Markov Decision Processes (MDP)'s, Hidden Markov Models (HMM)'s, or Partially Observable Markov Decision Processes (POMDP)'s. Within this chapter we will aim to provide optimal execution formulations using MDP, POMDP, and HMM formulations.



Markov Models		Do we have control over State transition?	
		No	Yes
Are the states completely Observable?	Yes	Markov Chain	Markov Decision Process (MDP)
	No	Hidden Markov Model (HMM)	Partially Observable Markov Decision Process (POMDP)

Figure 1: Markov Models: Michael Littman's explanatory grid[27]

Figure 1, depicts the Markov model categorization that we will use to section this chapter. The dimensions are based on control over the state transition, and observability of the state space. Controllability refers to the input vector u in the case of Section 3.1, Equation (19). For a controllable system u is treated as an action formally defined within the Section 4.1, which allows an external agent to effect change on the system. Observability concerns whether the current state is observed or treated as latent and unobserved.

We are primarily concerned with discrete, finite Markovian formulations, that is the problems in which the state space and action sets are finite, and the decision points k are equidistant [25]. Planning problems that utilize Markov models, attempt to provide an optimal policy, an optimal configuration for the model to follow. Each of the models that we are concerned with are formalized in the following sections of this chapter, starting with MDP's, then moving onto POMDP's. Q-learning is then introduced as the execution models that are developed are modeled in a reinforcement leaning context. HMM models follow with a brief discussion of the more general Bayesian methods, before we develop the last model that utilizes particle filtering, estimate position within a noisy trading environment.

4.1 Markov Decision Process (MDP)

Markov Decision Processes (MDP)'s are discrete time stochastic processes that extend Markov Chains, with the addition of actions and rewards that govern the evolution of the process in accordance with the Markov property. In this thesis we consider discrete state space Markov chains, which can be discrete or continuous time. MDP's may be of finite or infinite time horizon. We will primarily be concerned with finite time horizon and discrete state models as we have fixed end times for our trade executions. MDP's in which the state is observable are often referred to as completely observable Markov decision processes COMDP's to distinguish them from POMDP's.

An MDP model can be thought of as a system with an input and an output, that allows it to interact synchronously with the world. The input parameter is a completely observable and certain present state, the output an action, that both gathers information and affects the world [22]. The key components of a Markov decision model, are an observable state space $\{X_k\}_{k\geq 0} \equiv X$, action set $\{A_k\}_{k\geq 0} \equiv A$ and equidistant decision time points $k \in \mathbb{N}^+$. The MDP model contains a reward earned for a transition from one state, given an action $r(x_k, a_k) : x_k \in X, a_k \in A$ e.g. $r(x_k, a_k) : X \times A \to R$. Where the probability of a transition from state x_k to state x_{k+1} is $p(x_{k+1}|x_x, a_k) : x_{k+1} \in X, x_x \in X, a_k \in A$ e.g. $p(x_{k+1}|x_k, a_k) : X \times A \times X \to T$ where $T(.) \in X$ and T is the state transition function and $p(x_{k+1}|x_k, a_k) \geq 0$ and $\sum_{x_{k+1}\in X} p(x_{k+1}|x_k, a_k) = 1$ as discussed in [25]. The objective is to determine an action selection policy at each decision point that optimizes the system performance. This model may be represented as the tuple $< X, A, T, R >$.

The optimality criteria are constructed from the reward that is achieved for the next N steps (in the finite horizon case), with the objective of maximizing the reward R or minimizing the cost C. Cost may be mapped to reward by the following simple equality $R = -C$, allowing the reward maximization notation to be used on cost if desired. The optimality criteria within the optimal execution framework is normally constructed by minimizing expected cost but in the literature for the MDP formulation maximization of expected reward is normally considered

$$max_N \mathbb{E}\left[\sum_{k=0}^{N} r_k\right]. \tag{58}$$

Note, the model would contain a reward discount factor for an infinite horizon problem, to keep the sum finite. Rewards received prior to the current state are sunk, that is they are not considered and have no bearing on the optimization. A policy $\{\Pi_k\}_{k\geq 0} \equiv \Pi$ defines the actions that may be selected given a current state $X \rightarrow A$. This effectively defines the behavior of the system. An optimum policy defines the behavior of the system that will maximize the reward [22]. A policy may be stationary or non-stationary, in a stationary policy the state action selection is unchanged across time, for a non-stationary policy the state action mappings change at each decision point indexed by k. The evaluation of a policy, and selection of an optimal policy can thus be ascertained inductively using the following value function $V_{\pi,k}(.)$ iteration, for a policy at decision point k

$$V_{\pi,k}(x_k) = r(x_k, \pi_k(x_k)) + \gamma \sum_{x_{k+1}\in X} p(x_{k+1}|x_k, \pi_k(x_k))V_{\pi,k+1}(x_{k+1}) \tag{59}$$

the reward received for a given policy at $k = 0$ is $V_{\pi,0}(x_0)$. With the discount factor given by γ. The value on the last step N is simply $V_{\pi,N}(x_N) = r(x_N, \pi_N(x_N))$. The model evaluates the expected reward for a policy across all possible transitions at each time point, accumulating the discounted reward over time. In [22] the optimal policy is defined as the policy that maximizes reward $V^*_{k+1}(.)$, the star notation idicates an optimal solution, it is given by

$$\pi^*_k(x_k) = argmax_{a_k}\left[r(x_k, a_k) + \gamma \sum_{x_{k+1}\in X} p(x_{k+1}|x_k, a_k)V^*_{k+1}(x_{k+1})\right]. \tag{60}$$

The value function of the optimal policy corresponds to the Bellman equation for discounted MDP's [33], as discussed in section 3.1, it is the discrete form representation of Equation (24) with reward instead of cost, at time index k, using policy π for state x_k.

The solution to these recursions can not in general be found in closed form, it must therefore be approximated numerically. There are now numerous algorithms to compute an optimal policy [23][22][25]. Linear programming, policy iteration, and value iteration are the standard methods. We have already looked at linear programming in the existing methods chapter of this thesis in the Section 3.2.1 Bertismas and Lo - Bellman Equation. In policy iteration a sequence of improving policies is determined, until the optimal policy is achieved [25]. Value iteration is one of the earliest and serves as a basis for a number of other optimal policy search algorithms. The Value iteration algorithm, starts at decision point $k = 0$ and finds the maximum reward for a state action pair, recursively from the maximum of the value function $V_{\pi,k}(x)$, by looking ahead and rolling the future value back to the present. The algorithm terminates when the difference between two successive value functions is less than a given constant value ϵ known as the Bellman error, $V_{\pi,k}(x) - V_{\pi,k+1}(x) < \epsilon$.

4.1.1 Simple Value iteration MDP trading strategy

To illustrate this concept we will consider a simple trading strategy for the sale of 2 assets over a time period of 2 intervals, that is $N = 2$, $k \in \{0, 1, 2\}$, formulated under the simple pricing model: Bertismas and Lo 1 [14] with permanent impact. We set the initial price, S_0 , price to 10 dollars and the impact parameter $\theta = 1$

$$S_{k+1} = S_k + \theta n_{k+1} + \sigma W_{k+1}.$$

The result of any impact is to depress the price. There are 3 states, state 1 which contains 2 assets, state 2 which contains 1 asset and state 3 which contains 0 assets. The following restrictions to behavior are considered as illustrations of more complex real world scenarios. They involve action 1 to sell 1 asset and action 2 to sell 2 assets. A sale must occur at a decision point if possible and only assets that are held can be sold and no assets can be bought. The MDP formulation consists of state transition matrices for each action which in the Cartesian space are represented by $X \times X \times A$, e.g. one matrix for each action representing the transition probability, the system can transition to a state with the same number or less assets. The state transition matrix is stationary for each action, across time

$$\begin{array}{c c} a1 & \begin{array}{ccc} x1 & x2 & x3 \end{array} \\ \begin{array}{c} x1 \\ x2 \\ x3 \end{array} & \begin{pmatrix} P & 1-P & 0 \\ 0 & P & 1-P \\ 0 & P & 1-P \end{pmatrix} \end{array} \qquad \begin{array}{c c} a2 & \begin{array}{ccc} x1 & x2 & x3 \end{array} \\ \begin{array}{c} x1 \\ x2 \\ x3 \end{array} & \begin{pmatrix} P & 0 & 1-P \\ 0 & P & 1-P \\ 0 & P & 1-P \end{pmatrix} \end{array}.$$

To simplify for illustrative purposes we set $P = 0$. The reward matrices for each state action in the Cartesian space of the form $X \times A$. The corresponding reward matrices defined by these action state transition matrices are non-stationary, over time as the price of the asset is affected by the stock purchase, the reward function is

$$r(x_k, a_k) = \mathbb{E}_{\mathbb{P}(S)}\left[S_{k+1} n_k\right] = \int S_{k+1} n_k p(S_{k+1}|S_k) dS_{k+1} \tag{61}$$

with impossible state/action rewards being set to -99. As in this case we are maximizing reward. The reward matrices for each time index considered are then given by

$$k=1 \quad \begin{array}{c c} & \begin{array}{cc} a1 & a2 \end{array} \\ \begin{array}{c} x1 \\ x2 \\ x3 \end{array} & \left(\begin{array}{cc} 9 & 16 \\ 9 & 0 \\ 0 & 0 \end{array}\right) \end{array} \quad k=2 \quad \begin{array}{c c} & \begin{array}{cc} a1 & a2 \end{array} \\ \begin{array}{c} x1 \\ x2 \\ x3 \end{array} & \left(\begin{array}{cc} 9 & 16 \\ 8 & 0 \\ 0 & 0 \end{array}\right) \end{array} .$$

Selection of the optimal policy can be achieved for the purposes of this simple example through the use of the value iteration algorithm which uses the equation $V_{\pi,k}(x_k)$, in which the discount term is set to 1. The optimal policy is then the sequence of actions that produces the minimum implementation shortfall, e.g. in this case the maximum value for the sale. This can be calculated by hand as it is such a simple example or put into a numerical package the resultant policy is $\pi^* = a1_1, a1_2$, that is the optimal policy is action 1 selected at $k = 1$, and action 2 selected at $k = 2$. with a maximum expected market value of 17 dollars, and Implementation Shortfall of 3 dollars. This can be explicitly shown by making explicit the value iterations under each action and showing the max achieved wealth does correspond to $\pi^* = a1_1, a1_2$ as follows

$$\begin{array}{lll} V_{a1,k=1}(x1) = 9 + 1 * V(x2) & V_{a1,k=2}(x2) = 8 & Total \pi^*_{a1,a1} = 17 \\ & V_{a2,k=2}(x2) = 0 & Total \pi_{a1,a2} = 9 \\ V_{a2,k=1}(x1) = 16 + 1 * V(x3) & V_{a1,k=2}(x3) = 0 & Total \pi_{a2,a1} = 16 \\ & V_{a2,k=2}(x3) = 0 & Total \pi_{a2,a2} = 16. \end{array}$$

The value in this case is identical to the market value, if a more complex transition matrix is used or subjective rewards, that is rewards used to coerce the system to follow a given path are considered, then the value does not necessarily match the wealth realized. But the value tree branch that achieves the highest reward will still provide the optimal policy.

4.2 Partially Observable MDP (POMDP)

A partially observable Markov decision process is similar in nature to completely observable Markov decision processes except that the current state is treated as latent and unobserved, as such it must be estimated for use in an MDP formulation. An observation process is introduced with a distribution conditioned on the current state or past states and the action or action sequence taken. The model extends MDP's by adding the observation probability $p(y_k|x_k, a_k)$: $y_k \in Y, x_k \in X$, $a_k \in A$. Where the observable stochastic process is $\{Y_k\}_{k\geq 0} \equiv Y$, and the latent state space stochastic process $\{X_k\}_{k\geq 0} \equiv X$ is partially observed, all other components remain the same. Thus a POMDP model can be specified by the tuple $< X, A, T, R, \Omega, Y >$ in which A, T, R maintain their original definitions as given in Section 4.1. The main objective remains to ascertain the optimal policy, given an optimality criteria [22].

The input to the optimization procedure are the observations, the output of the procedure the actions. The system maintains an internal belief state that summarizes the past updating its internal belief state online using a state estimator as in [29]. A state estimator updates the belief state based on the previous belief state, the observation, and the action taken. It should also be noted that the belief state process is a Markov process. The policy that governs which action is taken within the POMDP setting depends on the belief state, not the known state as in COMDP's. The belief state provides the POMDP with the ability to act under uncertainty, and provides a sufficient statistic or summary for the state of the system [33]. Given that $p(b_k|x_k)$ is the belief state density, that provides the belief of being in a given state $x_k \in X$, where $0 \leq p(b_k|x_k) \leq 1$ and $\sum_{x\in X} p(b_k|x_k) = 1$. The belief state can be the entire distribution, a summary of the distribution through a sufficient statistic or an empirical estimate with dimension reduction of the information contained in the filter distribution of the latent process conditioned on the action and observations. Within the following formulation b_k summarizes $y_{1:k}, a_{1:k}$ providing a summary of the conditional filtering distribution given in Equation (62). The new belief state b_{k+1} given x_{k+1} can be calculated from the previous belief state $b_k|x_k$, given an action a, and an observation y from

$$\begin{aligned} b_{k+1} &= p(x_{k+1}|y_{k+1}, a_{k+1}, y_{k:1}, a_{k:1}) = p(x_{k+1}|y_{k+1}, a_{k+1}, b_k) \\ &= \frac{p(y_{k+1}|x_{k+1}, a_{k+1}, b_k)p(x_{k+1}|a_{k+1}, b_k)p(a_{k+1}, b_k)}{p(y_{k+1}|a_{k+1}, b_k)p(a_{k+1}, b_k)} \\ &= \frac{p(y_{k+1}|x_{k+1}, a_{k+1}) \sum_{x_{k+1} \in X} p(x_{k+1}|a_{k+1}, x_k, b_k)p(x_k|a_{k+1}, b_k)}{p(y_{k+1}|a_{k+1}, b_k)} \end{aligned}$$

$$b_{k+1} = \frac{p(y_{k+1}|x_{k+1}, a_{k+1}) \sum_{x_{k+1} \in X} p(x_{k+1}|a_{k+1}, x_k)b_k}{p(y_{k+1}|a_{k+1}, b_k)}. \tag{62}$$

An optimal policy may then be identified by mapping the belief state to an action using

$$p(b_{k+1}|a_{k+1}, b_k) = \sum_{y_k} p(b_{k+1}|a_{k+1}, b_k, y_k)p(y_k|a_{k+1}, b_k) \tag{63}$$

the value function is then

$$V_p(b_k) = \sum_{x_k \in X} b_k V_p(x_k) \tag{64}$$

[22][29]. With the POMDP model defined, it is worth considering the options for calculating the value function. The options are to compute the exact value function given an exact belief state, to calculate an approximate value function given an exact belief state, to calculate an exact value function given an approximate belief state or to compute an approximate value function for an approximate belief state [29].

To compute the exact value function given an exact belief is computationally expensive for a large state space i.e. it involves an exhaustive search over all action state sequences. To approximate the value function given an approximate belief state can be achieved by using a belief state grid approach [15]. Calculating an exact value function given an approximate belief state can be achieved with particle filters, this is an approach used for one of our MDP execution model formulations outlined in Section 4.4.2. Approximating the value function based on an approximate belief state can also be achieved by using particle filtering or Sequential Monte Carlo for the belief state, combined with nearest neighbor functional approximators for representing the value function [36]. Q-Learning, an approach that is used for representing the value function representation in the MDP's and POMDP's is briefly covered now.

4.3 Q-Learning

Q learning is an area of reinforcement learning, that allows a system to optimize itself based on reward and punishment signals. The system learns and adapts online through trial and error. On each interaction of the system with the outside world the system receives an input, an observation of the current state, which allows the system to pick an action which changes the state of the system. The result is a reward that is communicated to the system that is used to optimize the future response, the goal is to find a policy that maximizes some measure of reinforcement [23].

Model free methods of reinforcement learning can have, but do not require the well defined structure of COMDP models The state may be fully observable for COMDP models or partially observable when applied to POMDP models, that is models where the state is not directly observable. Q learning [37] is an approach that can be used for COMDP's or POMDP's. Considering the non deterministic case of Q learning, $Q^*(x_k, a_k)$ is the expected discounted reinforcement of taking action a_k in belief state x_k

$$Q^*(x_k, a_k) = r(x_k, a_k) + \gamma \sum_{x \in X} p(x_{k+1}|x_k, a_k) max_{a_{k+1}} Q^*(x_{k+1}, a_{k+1})$$

where $V^*(x_k) = max_{a_k} Q^*(x_k, a_k)$, hence $\pi^*(x_k) = argmax_a Q^*(x_k, a_k)$ is the optimal policy. The action selection rule is then the maximum Q value obtainable from the current state. The update is

$$Q(x_k, a_k) = Q(x_k, a_k) + \alpha(r_k(x_k, a_k) + \gamma max_{a_{k+1}} Q(x_{k+1}, a_{k+1}) - Q(x_k, a_k))$$

where α is the learning rate, and γ the discount factor [23]. The learning rate can affect the ability of the solution to converge locally to an optimum or globally for a given computational budget. The learning rate can also be set via Robbins Monro type stochastic approximation algorithms, see [35]. Q learning is exploration insensitive hence the Q values will converge to the optimal values no matter how the problem is explored, as long as the state action pairs are connected often enough, that is a state is visited and an action taken from the state enough times for the model to form a representative value of Q. There are however computational issues with large state/action spaces, in which the algorithm may converge slowly, this is where approximation can improve results.

There are numerous models that have been documented throughout the literature used to illustrate the above outlined Q-learning and Markov modelling techniques in a physical setting, that have inspired the MDP execution models that are outlined in this text. It is instructive to understand the dynamics of these models to gain a better understanding of the new execution models presented. These models include 'The puck on a hill' model described in Moore.A.W, and Atkeson.C.G, (1995) in their machine learning journal article 'The Parti-game Algorithm for Variable Resolution Reinforcement Learning in Multidimensional State-space' [28]. This model described as the mountain car problem by Rasmussen.C.E, and Kuss.M. in their paper 'Gaussian Processes in Reinforcement Learning' [34], describes an object/vehicle stuck in a valley. The objective is to park on a spot near the top of the valley, but the vehicle does not have enough power to simply drive to the spot, it must gain momentum in one direction then using the force of gravity combined with its own momentum, oscillate backwards and forwards until enough momentum has been achieved to reach the parking spot. In addition, another restriction is that the vehicle must not overshoot the parking spot.

Another model that is a favorite within the literature is the heaven hell problem used by [36]. Within this problem an agent is placed in one corner of a box, of the three other corners one is heaven (the goal), the other is hell, and the remaining corner contains a priest. The shortest path is from the agents corner to heaven, but the agent does not know which corner is heaven and which is hell, hence the safe course of action is to ask the priest first. The safest route to heaven is to go to the corner with the priest to ask the priest which corner heaven is, then onwards to heaven.

These scenarios and associated models are interesting from the perspective of understanding how to set up new models for the execution context, one can glean insight into the components of the models that drive the learning process. The set up of the model is of course the selection of the state space equations that define the model. Noting that a fully observed state space presents an MDP model, a partially observed state space yields a POMDP model. The state space model is discrete when represented as an MDP otherwise it is a Markov state space model.

For simulation the realizations of the model must map the state space to a discretized form. This is achieved in the discrete setting by taking the discrete or continuous state of the model and projecting it onto a state space grid, using for example the minimum Euclidean distance between the true state and the state grid points to allocate to the grid points. The state space grid provides the index into the Q table, used to provide the state to action mapping. Alternatively a cell splitting approach can be utilized, using a nearest neighbor methodology [11] to represent the Q table as used by Thrun.S. in the paper "Monte Carlo POMDP's" [36] This model is described in more detail within a later section of this chapter. The nearest neighbor approach works by finding a K nearest neighbor corresponding to a 'belief' state, and linearly averaging the values. If a belief state is not found that is close enough, a new belief state is added to the Q table [36]. Thrun.S. uses an approach that utilizes Gaussian kernels, and Monte Carlo sampling to approximate the Kullback-Leibler (KL) divergence between the states and the Q table belief states. This means that the Q table state space is continuous, and can grow.

The reward or cost function is effectively used to drive the learning process, it defines the greedy policy for the action that corresponds to a state within the Q table. For the finite horizon problem no discounting is necessary. For the infinite horizon problem then discounting is necessary along with a cutoff for the iteration which is usually achieved with some form of Bellman error. That is the iteration continues until there is no further benefit. In many problems outlined in the literature such as the puck on a hill problem, the objective is a shortest path to the goal. The reward is thus an accumulative penalty for each step. With a large penalty for not achieving the end result, e.g. overshooting the parking spot in the puck on a hill problem, or going to hell in the heaven hell problem.

4.3.1 Q-Learning MDP Execution Strategy

Drawing from the previous models that we have so far explored, we will outline a Q-learning MDP for a sell strategy, under the pricing model: Bertismas and Lo 3 but without the market information term. The model will be set up as a finite time horizon problem, with a fixed number of steps or decision points, with a caveat that the trade execution must occur in its entirety by the last time step. The initial model will be configured with an observable state space so that it remains an MDP problem. The reward will be based on the maximum attainable wealth for the strategy over the period of execution. In this model the optimality criteria are expected mean cost based, it does not consider mean variance minimization, as discussed in Section 2.4 of this document.

The set up of the model is as follows, the state vector $\boldsymbol{x}_k$ is composed of the leaves quantity l_k, and asset price S_k, $\boldsymbol{x}_k = \{l_k, S_k\}$. The action set is composed of actions that can increase the velocity by 1, do nothing or decrease the velocity by 1, $A = \{-1, 0, 1\}$ hence $v_{k+1} = v_k + a_k$ the state space is thus defined as

$$\begin{cases} l_{k+1} &= l_k + v_{k+1} \\ S_{k+1} &= S_k exp(\sigma W_k)(1 + \theta v_k) \end{cases} \tag{65}$$

with initial conditions $S_0 = S$ and $l_0 = L$, and transition probability $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, a_k)$.

One of the key criteria for having a functioning model is to ascertain the correct state space representation. From a practical perspective, for Q learning to occur successfully the state space must be well traveled, so that extensive state space exploration occurs and an action selection policy can be defined. The state space provided in the previous models are primarily set up for making decisions at a physical position decision point. We have previously defined the execution strategy position as the leaves quantity, that is for a sell strategy the number of assets remaining that must be sold. The trading rate is the number of assets that will be sold in a trading interval. The realization of this model is adapted for this simulation to a reduced discretized state space $\boldsymbol{x}_k \in \{\hat{l}_k, \hat{S}_k\}$, composed of a leaves ratio $\hat{l}_k$ and a asset price movement indicator $\hat{S}_k$. The leaves ratio $\hat{l}_k$ is used to indicate the stage in the execution strategy life cycle, with 1 representing initial period, and 0 the end period that is $\hat{l}_k \in \{1, 0.75, 0.5, 0.25, 0\}$. The asset price movement indicator $\hat{S}_k$ represents the directional movement of the asset price in relation to the initial price S_0 and the last price S_k manifesting itself as up/up state 4, up/down state 3, down/up state 2, or down/down state 1, that is $\hat{S}_k \in \{4, 3, 2, 1\}$. The Q table representation has an index of length 20 slots, each index entry is a tuple containing the Q value associated with each action. The greedy policy action selection process is driven by this association

As with the previous puck on a hill problem, or the heaven hell problem the end goal is to achieve an end position. In the context of the sell execution strategy this is equivalent to achieving a zero leaves position on conclusion of the execution strategy. However the objective is not necessarily the shortest path to the end position. The minimum cost strategy is the strategy with the minimum Implementation Shortfall or in the context of a sell strategy this can also be seen as the strategy that achieves the maximum wealth on conclusion. Many of the analytical models that we have outlined impose a sell only restriction for sell strategies or a buy only restriction for buy strategies. In the analytic dynamic trading strategy setting active in the money (AIM) strategies or passive in the money (PIM) strategies simply accelerate or decelerate the trading rate but without changing sign that is buy strategies only buy, sell strategies only sell. Q-learning approaches are driven by the cost/reward function, hence if actions are made available to the system then they will be explored, if the action provides a benefit, then the action will have an increased likelihood of being selected by the greedy action selection policy. The greedy policy will select the action attributed with the highest Q function value, for a given state space grid point. As previously mentioned for a sell strategy although the cost may be represented by the Implementation Shortfall, the reward is the wealth achieved for the sale.

$$r(x_k, a_k) = S_k v_k \tag{66}$$

where v_k represents the number of assets sold in the interval, equivalent in this context to n_k. To confine the system to a given interval with a specified termination time a large penalty must be administered if the leaves quantity is not 0 at the end time. This is a hard constraint, an approach of using a power law increase in penalty size depending on leaves at the end time could be used instead, in either case this is particularly important to constrain the system to a desired area of operation. The choice of reward/cost function exposes a number of interesting points, that is the system may learn to step out side of the conceived bounds of the system to achieve a better reward. An example of this in the context of this trading model is the system learning to manipulate the impact term, increasing it to the point that the stock price turns negative to influence the reward in a way that is positive to the system. A real world negative stock price is not possible as stock is a limited liability asset hence $S = (S, 0)^+$. Another point is if the model is set up to represent one type of understanding, then if the polarity of the problem reverses and the model structure changes, the reward function will shift sign causing the reward function to become a cost function. This would mean that the system would start learning suboptimal actions. Consider futures commodity trading the model may shift from contango in which the distant delivery price of a futures contract is greater than the current spot price, to backwardization, the opposite of contango. In this case the reward function effectively switches sign, hence the model would need to be able to handle this change. For our model in a very basic setting the system learns to speculate to buy low and sell high, whilst still having time to execute the full position by close of business, at the end time fixed for the given execution.

The numerical example provided (Figure: 2) shows that the model although simple, demonstrates potential. After an initial training period the Q-learning table has crystallized with an optimal configuration for a single stock trajectory, on which online reinforcement learning has been conducted. The system has learned to buy assets when the cost is low, acquiring more stock before selling when the asset price is increased, whilst balancing this trade off with the transaction costs associated with the sale. Although this model is not practical as it is too simple to be applied across a range of asset trajectories it does show that with a good understanding of the problem, a discretized state model representing the problem can be a valuable tool.

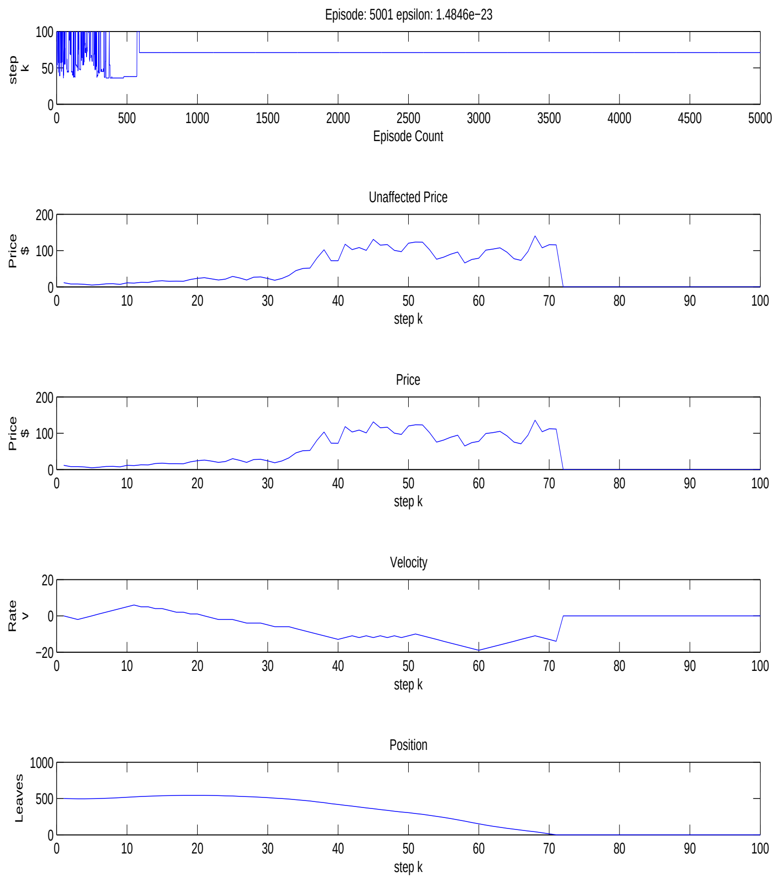


Figure 2: Q-learning MDP Execution Strategy

Referring to the plots in Figure 2. The initial plot, shows the steps that are taken to solve the problem for each episode, an episode is a full execution of the strategy across all time steps from start to end. For this and the following simulations this is fixed at 100 decision points e.g. $k \in [1...100]$. The second plot, shows the unaffected price, with a volatility of $\sigma = 0.2$ in this example, this is simply the stock price with no impact from trading. The third plot, shows the price graph with impact applied, the impact $\theta = 0.001$ so as not to overwhelm the price, setting θ to be too big forces the stock price to 0 on relatively small trade package executions. The fourth plot, shows the trading velocity starting from 0 this initially increases positively indicating that the system is buying assets, this then becomes negative indicating the sale of assets. The final plot, shows the position of the model, that is the initial asset holdings, as they initially increase then decrease to 0.

One of the main shortcomings of this model is that there is no representation in the state space model of the position of the system with regards to the end time. The model simply adapts itself to a sell bias, as it knows that it must conclude execution by a given time hence, more often than not the strategy concludes prematurely, and when bench-marked against the naive strategy under performs as the market impact term imposes too greater cost.

4.3.2 Q-Learning POMDP Execution Strategy

Drawing from the previous MDP Execution strategy, and reintroducing the market information term to the model: Bertismas and Lo 3, as the unobserved component of the model, where the information series is precalculated. In addition we use the step index k as the position indicator in the state space, this approach is similar to the clock concept used to reduce an POMDP to an MDP with the belief distribution being implicit within the model [26]. This provides a scenario analysis model, which allows the modelling of cause and effect type relationships of trading actions on the market. The velocity, price and the step are the observable components of the model. The position is unobserved, and the information term hidden (we cover latent processes in more detail in the following Hidden Markov Model section of this document). The system must adapt to the new information term. Within this framework the model state vector is composed of the time index, and asset price S_k, $\boldsymbol{x}_t = \{k, S_k\}$ is set up such that $v_{k+1} = v_k + a_t$ and $l_{k+1} = l_k + v_{k+1}$the state space is defined as

$$\begin{cases} I_{k+1} &= \rho I_k + \sigma W_k^{'} \\ S_{k+1} &= S_k exp(\sigma W_k)(1 + \theta v_k + \gamma I_{k+1}) \end{cases} \tag{67}$$

where W and W' are standard Brownian motions. The distributions that describe the model are for the unobservable state transition $p(I_{k+1}|I_k)$ defined by

$$\begin{cases} I_0 \sim N(0, \zeta) & , x = 0 \\ I_{k+1} \sim N(f(I_k), \zeta) & , x > 0 \end{cases} \tag{68}$$

the observation likelihood, as we can observe the stock price, is $p(S_{k+1}|S_k, I_{k+1,}a_k)$ defined by

$$\begin{cases} S_0 = S(.) & , k = 0 \\ S_{k+1}|S_k, I_{k+1}, a_k \sim N(h(S_k, I_{k+1}, a), \sigma) & , k > 0 \end{cases} \tag{69}$$

where $h(.)$ is a nonlinear function. the belief state follows from the continuous time version of Equation (62)

$$b_{k+1} \propto p(S_{k+1}|I_{k+1}, a_k) \int p(I_{k+1}|a_k, I_k) b_k dI_k. \tag{70}$$

The previous MDP execution strategy was run over a single price series, effectively training it for a specific market behavior. Running the system over a random set of price series would effectively cause the system to converge on the naive Bertismas and Lo strategy of executing equal sized packets at each time point. In fact this is a good test of the model to check that its behavior during initial setup is correct, and to calibrate initial parameters. It allows the correct costs to be identified, as having a extremely large overrun cost effectively constrains the system to the initial decision points that is for the initial values of k, the system avoids configurations that would approach $k \rightarrow N$. To allow the system to explore the complete space successfully, a lesser penalty must be levied, but this does mean that in some cases the system may not execute an order in it's entirety. The sell strategy, may not only choose to buy as part of it's sell strategy, but may additionally complete with a non-zero residual quantity.

As the model is now based around a hidden information series the model has been run over randomly generated price series (Figure: 3), with one fixed information series. The first 2000 runs are not used when calculating performance statistics to allow the reinforcement learning process to work and the system to train. The wealth for both Bertismas and Lo naive strategy, and the POMDP trading strategy are then averaged, to give the final result over a final 1000 runs. This graph is produced from the last run, run number 3000.

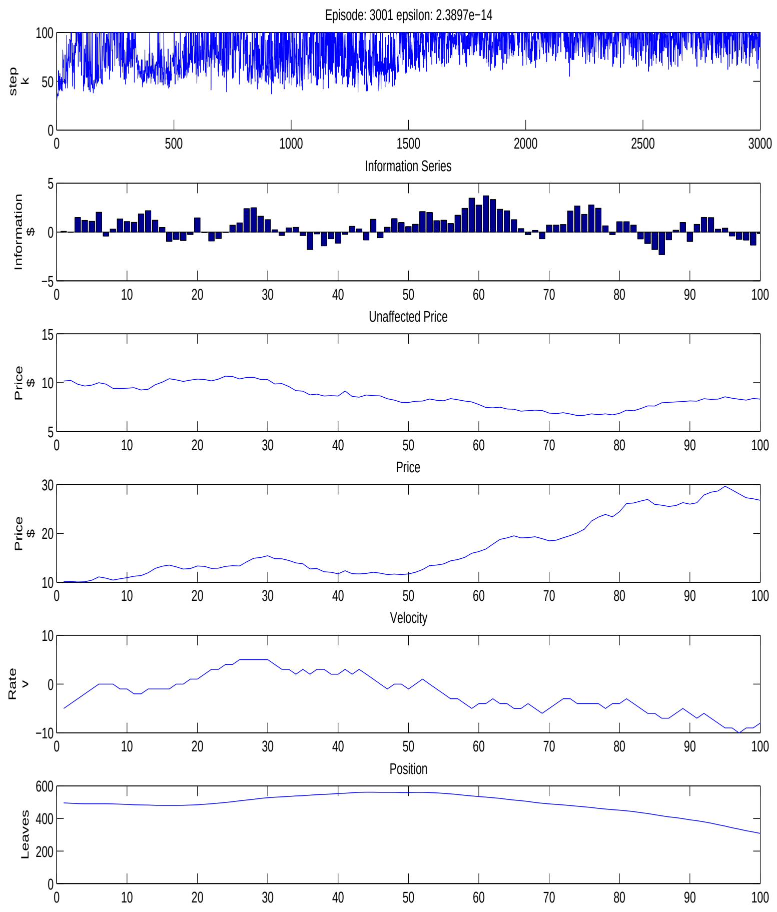


Figure 3: Speculating end run, POMDP

the initial Episode plot shows the exploration of the state space, initially the space is explored quite extensively but this narrows as the system continues to step through trading runs across each generated price series. This unaffected price only has a small random component $\sigma = 0.02$. The information price series are auto-correlated with lag one, with $\rho = 0.9$, and $\gamma = 0.02$. This means that the information series has a significant effect on the unaffected price, this can be seen in the price plot. The price plot is the actual price experienced by the system, including the market impact resulting from trading activity. The impact term $\theta = 0.0001$, has enough of an effect to penalize large trading packets but does not drive the price to 0, for realistic trading. With this setup it can be seen for this given run the information series is weighted towards the end, the resultant price increases from step 60. The velocity plot shows that the system starts off selling then starts to buy whilst the price of the asset is low, the system then starts to sell as the price rises, however for this particular run the trading position is not closed out by the end of the trading run. For the last 1000 episodes, or runs for this configuration the POMDP trading model on average outperforms the Bertismas and Lo naive trading strategy. The expected resultant wealth for one run of the naive strategy is 10294.15 dollars with a standard deviation of 1481.85 dollars the expected resultant wealth for the POMDP trading strategy is 12628.99 dollars with a standard deviation of 3547.88 dollars.

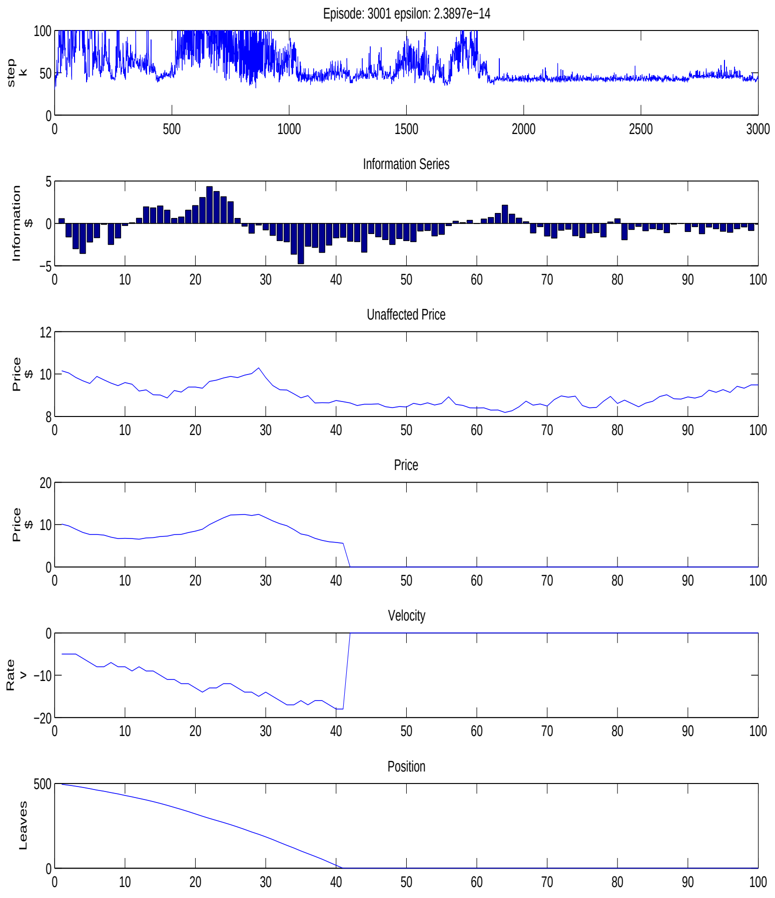


Figure 4: Start run, POMDP

Considering now an information series that is more positive at the start of the run (Figure: 4), it can see from the episode plot that the POMDP trading strategy gravitates to completing trading around step 50. As can be seen from the episode once the system has crystallized some understanding of the information series there are no overruns. The system completes all trading activity for each run by the half way point of the trading period. The expected wealth calculated over the last 1000 runs of the trading strategy for the randomly generated price series for the naive strategy is 3064.38 dollars with a standard deviation of 330.34 dollars, for the POMDP trading strategy it is 4419.12 dollars with a standard deviation of 411.62 dollars.

This model exposes a number of interesting features of Q-learning, in the finite horizon case. Firstly that the end reward/cost can serve to severely limit the exploration of the state space forcing a sub-optimal solution if the cost of over running is too high. This can also be seen as a risk aversion parameter, as defined previously by Almgren [4]. The risk aversion parameter previously defined on the variance component for the mean-variance optimization, effectively skews the trade package size loading to the start of the trading period. An additional parameter that has the same effect is γ the discount factor with in the Q-learning model. If this discount factor $\gamma < 1$, then the current rewards will have greater effect than future rewards, front loading the trading behavior, this behavior is similar in effect to a Implementation Shortfall strategy. If this discount factor $\gamma > 1$ then future rewards are credited with a greater effect than the current reward effectively end loading the strategy, which has the effect of turning the strategy into a Market on Close strategy, with an effect of targeting an end of period trading asset price. Within Q learning γ is primarily viewed as a discount factor where $0 \leq \gamma \leq 1$, allowing γ to range larger than 1 effectively credits future rewards.

4.4 Hidden Markov Model (HMM)

The term Hidden Markov Model (HMM) stems from using hidden states with a prior dependence structure to provide a statistical model. The concept of hidden state differentiates this representation from the state space representation given in Section 3.1 which was observable. The hidden state space can be finite e.g. Markov chain, or a continuous called simply a continuous state space model, we are primarily concerned with the discrete setting. Cappe et al describe a HMM as a Markov chain observed in noise [17]. In the discrete setting a HMM is a bi-variate discrete time stochastic process $\{X_k, Y_k\}_{k\geq 0}$ where k is an integer index, $\{X_k\}_{k\geq 0} \equiv X$ is a hidden state space stochastic process often called the latent process, and $\{Y_k\}_{k\geq 0} \equiv Y$ an observable stochastic process that is coupled to $\{X_k\}_{k\geq 0}$, that is $Y_k$is a sequence of independent random variables when conditioned on X_k. In the discrete setting $x_k \in X_k$ is a state vector and

$y_k \in Y_k$, is an observation vector

$$x_{k+1} = f(x_k, w_k) \quad (71)$$

$$y_k = g(x_k, w'_k) \quad (72)$$

where w'_k and w_k are white noise random sequences, and f and g are measurable functions. Equation (71) is the latent process or state equation. Equation (72) is the observation equation [17]. The equations characterize the state transition probability $p(x_{k+1}|x_k)$, and the additional measurement noise probability $p(y_k|x_k)$ [18]. We have already briefly touched on Hidden Markov Models with the information component of Section 2.2.2 model 3 [14] that was used in the POMDP trading model see Section 4.3.2.

HMM formulations are used in smoothing, filtering and projection applications, we will borrow from this to enrich our previous model using particle filters, and particle projections to ascertain a statistical estimate of the state in a noisy environment. But before we do this, we will briefly introduce Bayesian methods, and how they fit within the Markov setting. While noting that we have already solved some Bayesian recursions for the belief process in the POMDP models, in the discrete then continuous context. HMM characteristics differ from there POMDP counterparts through the controllability of the operations that change the state space. POMDP's control this process, HMM are passive, simply subject to the outcome of a change.

4.4.1 Bayesian Methods in Markov Models

Bayesian methods provide a rigorous statistical framework for dynamic state estimation [21], To achieve this they utilize prior knowledge, that is prior distributions on the unknown quantities and likelihood functions that map the latent process to the observations. Inference is then possible from the posterior distribution constructed from Bayes' theorem [20]. In essence Bayes law is the fundamental probability law of logical inference, that often results in an optimal solution.

For recursive Bayesian estimation, two assumptions are made. Firstly that the state transitions are Markov that is $p(x_{k+1}|x_k) = p(x_{k+1}|x_{1:k})$, and secondly that the observations are independent. The general framework requires that a prior distribution density $p(x_0)$, the marginal distribution or likelihood density $p(y_k|x_k)$, and a conditional transition distribution density $p(x_{k+1}|x_k)$ are known at least up to proportionality and can be evaluated point-wise and sampled from. The noise terms are white noise. The estimates are obtained recursively, the posterior distribution is then calculated using the Bayes update rule

$$\begin{aligned}
p(x_{k+1}|y_{1:k+1}) &= p(x_{k+1}|y_{k+1}, y_{1:k}) \\
&= \frac{p(x_{k+1}, y_{k+1}, y_{1:k})}{p(y_{k+1}, y_{1:k})} \\
&= \frac{p(y_{k+1}, y_{k:1}|x_{k+1})p(x_{k+1})}{p(y_{k+1}, y_{1:k})} \\
&= \frac{p(y_{k+1}|y_{1:k}, x_{k+1})p(y_{1:k}|x_{k+1})p(x_{k+1})}{p(y_{k+1}|y_{1:k})p(y_{1:k})} \\
&= \frac{p(y_{k+1}|y_{1:k}, x_{k+1})p(x_{k+1}|y_{1:k})p(y_{1:k})p(x_{k+1})}{p(y_{k+1}|y_{1:k})p(y_{1:k})p(x_{k+1})}
\end{aligned}$$

$$p(x_{k+1}|y_{k+1}, y_{1:k}) = \frac{p(y_{k+1}|x_{k+1})p(x_{k+1}|y_{1:k})}{p(y_{k+1}|y_{1:k})}. \tag{73}$$

The prior knowledge of the model is given by using the Chapman-Kolmogorov equation

$$p(x_{k+1}|y_{1:k}) = \int p(x_{k+1}|x_k)p(x_k|y_{1:k})dx_k \tag{74}$$

where the denominator for the Bayes recursion is called the the evidence is

$$p(y_{k+1}|y_{1:k}) = \int p(y_{k+1}|x_{k+1})p(x_{k+1}|y_{1:k})dx_{k+1} \tag{75}$$

and the likelihood of the $k+1$ observation of y is $p(y_{k+1}|x_{k+1})$. These equations map directly to the previously derived POMDP belief state estimation formulations, in a Markov context. Analytical solutions or numerical approximations to these equations are the basis for Bayesian recursive inference [18]. State filtering is the process of estimating sequentially the state of a system from a sequence of noisy measurements, with the aim of providing an optimal latent process trajectory with regards to a cost function [32].

The optimality criteria selected for a problem define the best solution. In Bayesian filtering the optimality condition may be, Minimum mean-squared error (MMSE), Maximum a posterior (MAP), Maximum Likelihood (ML), Minimax, Minimum conditional inaccuracy, Minimum conditional KL divergence, or Minimum free energy to name but a few [18]. The MMSE estimate may be computed from the following formula

$$\hat{x}_{k|k}^{MMSE} = \mathbb{E}\{x_k|y_{1:k}\} = \int x_k p(x_k|y_{1:k})dx_{1:k}.$$

At the heart of Bayesian inference lie three problems [18], normalization, marginalization and expectation. Normalization used to calculate the posterior is given by

$$p(x_{1:k}|y_{1:k}) = \frac{p(y_{1:k}|x_{1:k})p(x_{1:k})}{\int_X p(y_{1:k}|x_{1:k})p(x_{1:k})dx_k} \tag{76}$$

marginalization using

$$p(x_{1:k}|y_{1:k}) = \int_Z p(x_{1:k}, z_{1:k}|y_{1:k})dz_k \tag{77}$$

and expectation from

$$\mathbb{E}_{p(x_{1:k}|y_{1:k})}[f(x_{1:k})] = \int_X f(x_{1:k})p(x_{1:k}|y_{1:k})dx_k. \tag{78}$$

The posterior density is not easily handled as it does not admit a sufficient statistic with finite and constant dimension except in the case of a limited set of dynamic stochastic systems. Various approximation methods exist including analytical methods, numerical methods, multiple mode, and sampling approaches. Sampling methods include the unscented Kalman filter, and particle filtering methods. State filtering is a powerful approach ideal for estimating in real time, high frequency non-stationary environments [32]. The model that is described in Section 4.4.2, has non-linear elements, numerical approximation is used to obtain the latent state.

A normal HMM is defined as having a Gaussian conditional distribution of $p(x_{1:k}|y_{1:k})$. If it is possible to model the data with a linear Gaussian HMM, then an exact analytical solution exists that allows the recursive calculation of the posterior distribution [20]. This is know as the Kalman recursion, it lies at the heart of the Kalman filter. For nonlinear or non-Gaussian data there is no analytic solution to allow the recursive calculation of the posterior distribution, therefore a numerical method must be used. It is also worth noting that in general for nonlinear, non-Gaussian and non-stationary problems there is no exact solution to the recursion in Equation (73), hence the numerical solution is approximate, a locally optimal solution may result from the approximation [18].

Approximation methods include deterministic and random grid based, or Monte Carlo sampling approaches which are often used for analytically intractable problems. They can be used to calculate the probability distribution function if it is not possible to sample directly from the required posterior distribution. However these methods are often computationally expensive.

Sequential Monte Carlo (SMC), an alternative monte carlo simulation method takes into account the salient statistical properties of the problem, it provides an alterative approach that can be used to more efficiently recursively calculate the posterior distribution [20][21][17]. In SMC methods an empirical estimate of a distribution, called the proposal distribution, is obtained which is used to estimate an integrand, this estimate is then adjusted according to the closeness of fit to the target distribution this is termed Importance Sampling. For recursive estimation the importance weights must be recalculated to stop the model degenerating [20]. To summarize this approach if we wish to compute the integral of some function $I_k(\phi(x_{1:k}))$, the estimate can be obtained for a set of independently sampled variables $X^i_{1:k} \sim p(x_{1:k}|y_{1:k})$ for the distribution p by the following equation

$$I_k(\phi(x_{1:k})) = \int \phi(x_{1:k})p(x_{1:k}|y_{1:k})dx_{1:k} \approx \sum_{i=1}^{N} W^i_k \phi(x^i_{1:k}) \tag{79}$$

where approximation given is estimated from the empirical distribution

$$\hat{p}_k(x_{1:k}) = \sum_{i=1}^{N} W^i_k \delta_{X^i_{1:k}}(x_{1:k}), \tag{80}$$

where the normalized weight function is

$$W^i_k = \frac{w_k(X^i_{1:k})}{\sum_{j=1}^{N} w_k(X^j_{1:k})} \tag{81}$$

and the importance weight is given by

$$w_k(x_{1:k}) = \frac{p(x_{1:k}|y_{1:k})}{d(x_{1:k})} \tag{82}$$

where d is the importance distribution used to sample I. It can be seen that the approximation for $I_k(\phi_k)$ satisfies the Central Limit Theorem with asymptotic variance

$$\frac{1}{N}\int \frac{p^2(x_{1:k}|y_{1:k})}{d(x_{1:k})}(\phi(x_{1:k}) - I_k(\phi_k(x_{1:k})))^2 dx_{1:k}. \tag{83}$$

The advantages of using an SMC approach are that the rate of convergence is independent of the dimension of the integrand, which contrasts with numerical integration techniques which diverges as the dimension increases. In general as the sample size is large then from the law of large numbers the empirical estimate of an integral almost surely converges on the analytical estimate. The SMC approach is used in the following section 4.4.2.

4.4.2 Q-Learning POMDP Execution Strategy with particle projection

An approach for representing the belief states of POMDP's in a continuous real-valued state spaces taken by Sebastian Thrun, in the paper Monte Carlo POMDP's [36] is an adaptive approach which draws belief states from weighted samples drawn from the belief distribution. These samples are then represented with corresponding rewards in a Q setting using KL divergence to locate a nearest neighbor or as the decision impetus for the creation of a new state mapping. Adaptive approaches for the definition of the state space allow the system to add state space grid points to the state space representation. The cell splitting approach allows the Q table to develop in the areas that are important to the system to allow the system to focus learning in the areas that are most significant by reducing the distance required to trigger the creation of a new state [28].

For this particular trading formulation we will continue to use our fixed state space definition, but use Thun's particle projection and particle filter algorithms for the non-linear, Gaussian model. The MMSE estimate of the belief and reward set is returned from the particle projection algorithm, as we are using the original euclidean distance for calculating the mapping to the fixed predefined discrete state space. We will continue to use the pricing Section 2.2.2 model 3 [14] to define the price dynamics, again removing the information term from the model. With this formulation we will enrich the trading velocity with a Gaussian indecision component introducing to model the possibility of not being filled over a given interval. The model is set up such that v_k is drawn from a set of possible velocities. The model is defined as follows:

$$\begin{cases} l_{k+1} & = l_k + v_{k+1}(1 - \eta_k W_k^{'}) \\ S_{k+1} & = S_k exp(\sigma W_k)(1 + \theta(l_{k+1} - l_k)) \end{cases} \tag{84}$$

the distributions that describe the model are for the unobservable state transition $p(I_{k+1}|I_k)$ defined by

$$\begin{cases} l_0 \sim L & , x = 0 \\ I_{k+1} \sim N(f(l_k), \eta) & , x > 0 \end{cases} \tag{85}$$

the observation likelihood as we can observe the stock price $p(S_{k+1}|S_k, l_{k+1}, a_k)$ is defined by

$$\begin{cases} S_0 = S(.) & , k = 0 \\ S_{k+1}|S_k, l_{k+1}, a_k \sim N(h(S_k, l_{k+1}, a), \sigma) & , k > 0 \end{cases} \tag{86}$$

the belief state is again the continuous time version of Equation (62)

$$b_{k+1} \propto p(S_{k+1}|l_{k+1}, a_{k+1}) \int p(l_{k+1}|a_{k+1}, l_k) b_k dl_k. \tag{87}$$

Within this framework sampling is from the the belief state, an importance weight is then assigned based on the observation. The particle weights are then normalized. The particle and corresponding normalized weight are then entered into a particle summary set. The expectation is then calculated which provides the resultant expected new belief state calculated from the particle filter. The particle filter function is defined by the following algorithm from [36]:

Algorithm 1 particle_filter

```
Algorithm particle_filter($b_k$,$a_k$,$y_{k+1}$)
$b_{k+1} = \emptyset$
do N times
  draw random state $x_k$from $b_k$
  sample $x_{k+1}$according to $p(x_{k+1}|a_k, x_k)$
  set importance factor $p(x_{k+1}) = p(y_{k+1}|x_{k+1})$
  add $< x_{k+1}, p(x_{k+1}) >$to $b_{k+1}$
normalize all $p(x_{k+1}) \in b_{k+1}$so that $\sum p(x_{k+1}) = 1$
return $b_{k+1}$
```

Particle projection then allows a set of belief state to reward functions to be calculated which may then be used to define the Q learning process. However we take the expectation, and update our fixed Q learning table value function with this summarized MMSE estimate. The particle projection wraps the particle filter, sampling an observation [36]

Algorithm 2 particle_projection

```
Algorithm particle_projection($b_k$, $a_k$)
$\Theta_k = \emptyset$
do N Times
  draw random state $x_k$from $b_k$
  sample a next state $x_{k+1}$according to $p(x_{k+1}|a_k, x_k)$
  sample observation $y_{k+1}$according to $p(y_{k+1}|x_{k+1})$
  compute $b_{k+1}$ =particle_filter($b_k$,$a_k$,$y_{k+1}$)
  add $< b_{k+1}, r(y_{k+1}, a_k) >$ to $\Theta_k$
return $\Theta_k$
```

The model used within this configuration is a little less stable than the previous models as the state space vector models the position exactly. The outcomes of this is that a change in an action for an upstream position results in the downstream position actions being offset. Changes to the action mappings that occur early on have repercussions that ripple through the remaining action choices. The model itself is not optimal. The ability of the particle filter to operate within these noisy conditions however, allows us to investigate some of the noise tolerance properties of the particle filter.

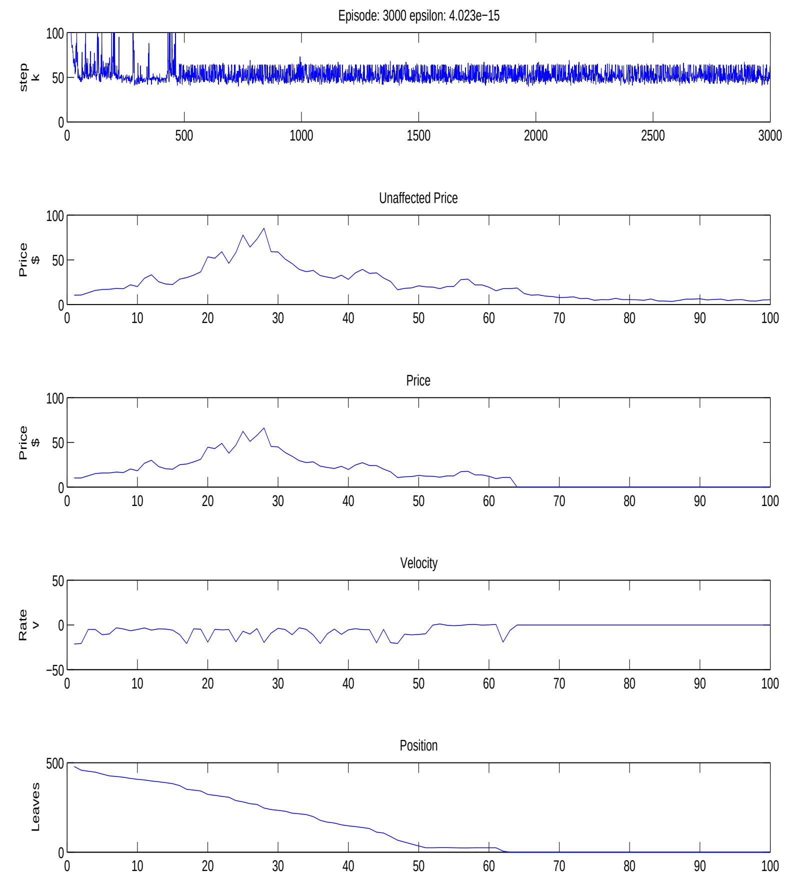


Figure 5: Particle Project POMDP

In running the model (figure 5), over a generated price series, the resultant model wealth of one run of the naive strategy calculated over the last 1000 runs for this price series was 10413.88 dollars with a standard deviation of 329.59, the wealth for the particle projection POMDP was 13884.13 dollars with a standard deviation of 347.50 dollars. The position series is linear, it follows a naive strategy trajectory, this model almost always ends prematurely, hence provides better performance than the Bertismas and Lo naive strategy when the price series has a greater price at the initial time period. The model although suboptimal, does demonstrate the possibilities of using particle filtering within the Q-Learning context. Possible extensions would derive online parameter estimation algorithms based on the gradient of the filter distribution within the particle filter for volatility modelling, allowing the calibration of the Q to be optimized for the problem in hand.

4.5 Summary

This chapter has introduced at a high level the main characteristics of each of the Markov model types as categorized by Littman [27]. A trading strategy has then been provided for each of the outlined MDP categories to investigate the modelling framework and analyze the optimal execution problem from a novel perspective. It can be seen that utilizing a numerical method for solving the optimal execution problem allows significant flexibility in the representation of the problem. Furthermore the Q-Learning approach is extremely powerful allowing the system to calibrate itself. However to obtain a practical solution, the correct state space representation, market signals, and value function must be studied further.

5 Conclusion

In conclusion the goal of exploring the problem of optimal execution for algorithmic trading in the context of MDP's has been achieved. Like the ancient art of alchemy, the execution problem has been decomposed into it's base elements, of trading strategy, price model, market impact, and optimality criteria. Each of these constituent components has been further explored, looking first at the mechanisms that govern their behavior and then distilling these mechanisms to the most important underlying factors and finally representing them as sets of equations used to model the problem throughout this work. The motivation stated for attacking this problem of minimizing cost, has introduced us to the concept of Implementation Shortfall, a topic of research in itself. Implementation Shortfall in the context of this thesis has been further reduced to the indirect component of execution cost, leading us to the optimality criteria of expected cost minimization and mean-variance cost minimization.

It has been these optimality criteria that have driven our solutions, predominately through the Bellman value function, that has provided a thread of continuity through the linear programming solution of Bertismas and Lo, through the HJB formulation of Almgren to the MDP value function presented in Chapter 4. The initial solutions given were presented to allow greater insight into the problem domain to be ascertained, this objective was achieved. These naive solution has in addition provided a point of reference to contrast the Markov execution strategies against.

Re-framing the problem in the Markov setting, using MDP's and Q-Learning for the optimal execution problem to the authors knowledge is novel and unique. To this end all aspects of the formulation, and representation have been a substantial learning experience. The problem has been broken down and the boundary conditions used to constrain the original solution to sell (buy) only or complete execution in full discarded. If an adequate state space representation, market signals and reward feed back mechanisms are provided, the system itself using reinforcement learning can discover the optimal trading trajectory, calibrating itself for the task at hand.

The MDP models provided have shown that it is possible to not only provide structural algorithms using this approach but that these models are more than capable of being used in the sophisticated situational algorithm space. However, the environment that each of the models has been run in has been tailored to allow the ultimate success of the model outlined in this thesis. Actual implementation would require further modelling and research.

Going forward, as a research area this has great potential. Identifying the correct model state space, market signals and reward criteria, using Gaussian nearest neighbor state space representation techniques and parameter inference within underlying particle filters someone somewhere will make a lot of money.

References

[1] A. Alfonsi, A. Fruth, and A. Schied. Optimal execution strategies in limit order books with general shape functions. *Quantitative Finance*, 10(2):143–157, 2010.

[2] R. Almgren. Optimal trading in a dynamic market. *Preprint*, 2009.

[3] R. Almgren and N. Chriss. Value under liquidation. *Risk*, 12(12):61–63, 1999.

[4] R. Almgren and N. Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–40, 2001.

[5] R. Almgren and N. Chriss. Competitive Bids for Principal Program Trades. 2003.

[6] R. Almgren and J. Lorenz. Bayesian adaptive trading with a daily cycle. *The Journal of Trading*, 1(4):38–46, 2006.

[7] R. Almgren and J. Lorenz. Adaptive arrival price. *Trading*, 2007(1):59–66, 2007.

[8] R. Almgren, C. Thum, E. Hauptmann, and H. Li. Direct estimation of equity market impact. *Risk*, 18:57–62, 2005.

[9] R.F. Almgren. Optimal execution with nonlinear impact functions and trading-enhanced risk. *Applied Mathematical Finance*, 10(1):1–18, 2003.

[10] ASX. Algorithmic trading and market access arrangements. *ASX Review*, 2010.

[11] C.G. Atkeson, A.W. Moore, and S. Schaal. Locally weighted learning. *Artificial intelligence review*, 11(1):11–73, 1997.

[12] Bellman. On the theory of dynamic programming. 1952.

[13] D. Bertimas, A.W. Lo, and P. Hummel. Optimal control of execution costs for portfolios. *Computing in Science & Engineering*, 1(6):40–53, 2002.

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

[15] R.I. Brafman. A heuristic variable grid solution method for POMDPs. pages 727–733, 1997.

[16] D. Brigo and F. Mercurio. Discrete time vs continuous time stock-price dynamics and implications for option pricing.

[17] O. Cappé, E. Moulines, and T. Rydén. *Inference in hidden Markov models*. Springer Verlag, 2005.

[18] Z. Chen. Bayesian filtering: From Kalman filters to particle filters, and beyond. *adaptive Syst. Lab., McMaster Univ., Hamilton, ON, Canada.[Online]. Available: http://soma. crl. mcmaster. ca/ zhechen/download/ieee bayesian. ps.*

[19] CK Chui and G. Chen. *Linear systems and optimal control*. Springer-Verlag New York, Inc. New York, NY, USA, 1989.

[20] A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer Verlag, 2001.

[21] NJ Gordon, DJ Salmond, and AFM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. 140(2):107–113, 2002.

[22] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[23] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Arxiv preprint cs/9605103*, 1996.

[24] R. Kissell and R. Malamut. Algorithmic decision-making framework. *The Journal of Trading*, 1(1):12–21, 2006.

[25] K. Lodewijk. Markov decision processes. *Leiden University, The Netherlands.*

[26] Martin.J.A.H. Mountain car. *http://www.dia.fi.upm.es/ jamartin.*

[27] A. Michael Littman, M. Cassandra. Michael littman's explanatory grid. *http://www.cassandra.org/pomdp/pomdp-faq.shtml.*

[28] A.W. Moore and C.G. Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21(3):199–233, 1995.

[29] K.P. Murphy. A survey of POMDP solution techniques. *environment*, 2:X3, 2000.

[30] A. Obizhaeva and J. Wang. Optimal trading strategy and supply/demand dynamics. *NBER Working Paper*, 2005.

[31] A.F. Perold. The implementation shortfall: Paper versus reality. *Streetwise: the best of the Journal of portfolio management*, page 106, 1998.

[32] Peters. G Piggott. M. Filtering for linear/nonlinear state space models. 2010.

[33] Fabiani.P. Rachelson.E, Garcia.F. Extending the bellman equation for mdp's to continuous actions and continuous time in the discounted case. *Symposium on AI and Math*, Tenth Intl, 2008.

[34] C.E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems*, 16:751–759, 2004.

[35] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[36] S. Thrun. Monte carlo pomdps. *Advances in neural information processing systems*, 12:1064–1070, 2000.

[37] C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.