IGIDR Proceedings/Project Reports Series PP-062-04

Best Execution of a Trade: An Adaptive Approximate Dynamic Programming Approach using Actor-Critic

Paris Pennesi and Gangadhar Darbha

Quantitative Approaches to Public Policy –
Conference in Honour of Professor T. Krishna Kumar

Held in conjunction with the
Fourth Annual International Conference on Public Policy and Management
Indian Institute of Management Bangalore (IIMB)

9-12 August 2009

School of Business and Management
Queen Mary, University of London
London, United Kingdom

Indira Gandhi Institute of
Development Research
Mumbai, India

Centre for Public Policy
Indian Institute of Management
Bangalore, India

http://www.igidr.ac.in/pdf/publication/PP-062-04.pdf

BEST EXECUTION OF A TRADE: AN ADAPTIVE APPROXIMATE DYNAMIC PROGRAMMING APPROACH USING ACTOR-CRITIC

PARIS PENNESI AND GANGADHAR DARBHA

Abstract. This paper proposes a trading strategy aiming to minimise the execution cost of a large trade in a double-auction, or order driven, market. The problem faced by the trader is to complete the transaction in a given time, while minimising the impact of the trade on the market price. To reach such objective a large trade is to be split into a sequence of several smaller trades to reduce the potential market impact and exploit the dynamic of the market liquidity.

The trading problem informally described above involves a sequence of decisions, i.e. when and how much to trade, which are interdependent because each decision changes, due to market impact, the underlying market microstructure, i.e. the order book, upon which the subsequent decisions have to be taken. Due to this dynamic dependence the decision process can be modelled as a dynamic programming problem.

1. Introduction

The actual execution of a given trading strategy may affect the final results of the trade to a great extent. The difference in performance between a "paper" investment, or trading idea, and its actual execution has been called by Pérold in [12] "the implementation shortfall". Such shortfall can be thought as an investment cost and it can be decomposed into four main components, namely, cost due to manager's delays, explicit costs, implicit cost and opportunity costs. The cost due to manager's delay is simply the volume of the order multiplied by the change in the mid-offer price between the time when the portfolio manager decides on the transaction and the time when she passes the order to a trader. Explicit costs are known. They are the sum of actual brokerage commissions, fees, etc. Implicit costs are calculated as the total value paid/received for that portion of the order filled during the trading period minus the total value that would have been paid/received if all those trades had been executed at the mid-offer price when the order was placed. Opportunity cost is calculated as the total value of that portion of the order that remains unfilled at the end of the trading period based on the closing mid-offer price minus its value based on the mid-offer price when the order was placed.

The main focus of this note is on how to minimise the implicit costs component of the "implementation shortfall". This is both because a trader's perspective is taken and because explicit costs are known and easier to dealt with than, the "a priori" unknown, implicit ones. Opportunity cost are partially addressed while discussing about the risk of the trade and the volatility on the execution performance, but it

Date: February 25, 2009.

is assumed that the trade has to happen in a given time horizon and that when to place the trade is not a decision variable. Hence, the objective of the note is to suggest an optimal strategy to execute a given order, leaving aside its underlying motive. This is called the optimal execution problem.

In order to solve the optimal execution problem, it is useful to explore more in details the nature of implicit costs. It becomes evident that such costs are not only unknown but they are also dynamic depending both on current market conditions, i.e. current market liquidity, and on how the order is placed throughout the trading horizon. This latter characteristic of the implicit costs make them endogenous and dependent on the full path, or profile, of the trading strategy implemented. As noted by Obizhaeva and Wang in [9] the dynamics of the supply/demand, rather than its static properties, is of critical importance to the optimal trading strategy of a given order. Thus, it is important to study how market participants trade to understand how liquidity is provided and consumed, and how it affects the behaviour of security prices.

The optimal execution problem emerges as suitable to be cast into a dynamic programming framework where a key role is played by the liquidity dynamic model. Previous research along this route has, for simplicity, analysed optimal trading strategies in the context of a order driven market where the instantaneous liquidity is captured by a limit order book. One of the main contribution to the study of the optimal execution problem is due to Bertsimas and Lo ([5]). The authors make a first attempt to model the optimal execution problem as a dynamic programming problem, and provide an analytical solution to it, in the case of simple models for the liquidity dynamics. They consider an investor who wishes to buy a large number $\bar{S}$ of shares, of a given security, over a fixed time interval $[0, T]$. They divide T into N intervals of length $\tau = T/N$ and consider the discrete times $t_k = k\tau$, for $k = 1, \dots, N$ as the only instant where is possible to observe the market and trade. The state variable at time t_k is composed of the last observed price of the security, p_{k-1}, and the number of shares remaining to be purchased, w_k. s_k is the control variable at time t_k and denotes the order to send to the market ($s_k > 0$ means we are buying shares while $s_k < 0$ means we are selling them). The investor's objective is to minimize the following expected cost of buying $\bar{S}$ shares over a the fixed time interval:

$$\min_{\{s_k\}} E\left[\sum_{k=1}^{T} p_k s_k\right] \tag{1.1}$$

subject to

$$\sum_{k=1}^{T} s_k = \bar{S}. \tag{1.2}$$

The law of motion for the state variable w_k is given by:

$$w_k = w_{k-1} - s_{k-1},\ s_1 = \bar{S},\ w_{T+1} = 0. \tag{1.3}$$

To complete the statement of the problem, the "law of motion" for p_k needs to be specified. This includes two distinct components: the dynamics of p_k in the absence of our trade (the trades of others may be causing prices to fluctuate), and the impact that our trade of s_k shares has on the execution price p_k. The authors suppose that the former component is given by an arithmetic random walk, and

the latter component is simply a linear function of trade size so that a purchase of s_k shares may be executed at the prevailing price p_{k-1} plus an impact premium of θs_k, $\theta > 0$. Then the law of motion for p_k may be expressed as:

$$p_k = p_{k-1} + \theta s_k + \epsilon_k,\ \theta > 0,\ E[\epsilon_k | s_k, p_{k-1}] = 0 \tag{1.4}$$

where ϵ_k is assumed to be a zero-mean independently and identically distributed (IID) random shock, i.e., white noise.

Given the previous setting the authors showed that the optimal trading strategy is to split equally the order during the trading horizon, this "naivë" strategy is still optimal if the market impact of the trade is convex and there is no temporal correlation in the movement of the underlying liquidity. In the same paper, the authors show that if the law of motion presents temporal and if we are "privileged" information, i.e. we are able to predict price movement, splitting the order in equal chunks is not optimal and it is possible to derive an optimal strategy depending on our "privileged" information. More specifically, if the price impact is assumed to be temporary and represented by Δ_k:

$$\Delta_k = (\theta s_k + \gamma x_k)\tilde{P}_k \tag{1.5}$$

$$x_k = \rho x_{k-1} + \eta_k \tag{1.6}$$

where x_k is a proxy for market conditions and represents our "privileged" information and η_k is white noise with mean 0 and variance σ_η^2. ρ represents the correlation in the information while the θ and γ parameters measure the sensitivity of price impact to trade size and market conditions. In this model, also known as linear price impact (LPT) law of motion, the dynamics of no-impact price $\tilde{p}_k$ in the absence of our trade is modelled by a geometric Brownian motion:

$$\tilde{p}_k = \tilde{p}_{k-1} exp(z_k) \tag{1.7}$$

where z_k is an IID normal random variable with mean μ_z and variance σ_z^2. The closed form solution, derived solving the relative Bellman equation, is:

$$s^*_{T-k} = \delta_k^w w_{T-k} + \delta_k^x x_{N-k} + \delta_k^1. \tag{1.8}$$

where δ_k^w, δ_k^x and δ_k^1 are coefficients depending on θ, γ, ρ, μ_z and σ_z^2.

The obvious drawbacks of the trading strategy suggested by Bertsimas and Lo ([5]) is that it relies heavily on perfect knowledge of the dynamics of the liquidity dynamic. Moreover, the dynamic used is restricted to a very simple one and it has not been empirically verified, in fact it is not even theoretically consistent with the efficient market hypothesis. A step forward a more realistic scenario and to a more profitable trading strategy has been done by Almgren ([1]), Almgren and Chriss ([2]) and more recently by Obizhaeva and Wang ([9]). Both papers introduce a more realistic market impact model and [9], in particular, introduces a framework to model liquidity dynamics which is both consistent with the market efficient hypothesis and empirically plausible. The main qualitative insight found by Obizhaeva and Wang ([9]) is that the optimal trading strategy should be $\bigcup$ shaped, starting and finishing with two relatively large trades and with small and equal trades in-between. This can potentially give a justification to the observed $\bigcup$ shaped volume profile observed in many markets and trade paths followed by institutional investors.

The main limitation of the previous works is the fact that the solution of the optimal execution problem relies on a specific model for the liquidity dynamic, or

market impact. This is a major limitation for two reasons. First, the correctness and the accuracy of the model is crucial to the performance of the trading strategy. Second, since, as noted before, the liquidity dynamic is an endogenous process created by the trading activity itself, hence, it is bound to change over time in response to the relative behaviour of the market players, defeating any effort to estimate it using historical data. What is really needed is an adaptive trading strategy, being capable at the same time of reacting to the current mood of the market and respond with an effective trading decision. It is possible to think of a meta-strategy which encompasses more simple trading strategies as particular cases switching among them as needed in response to market signals. A first step in this direction is described in [3] and [4] where the authors show that substantial improvement against basic, or static, strategies can be obtained using rules to switch between trading profile in response to certain market events or change in price of the asset being traded. Although, such results are interesting and represent a step forward in the search for a more profitable adaptive trading strategy, they are based mainly on heuristic which can be hard to justify and leave room for improvement.

The algorithms presented in this paper mixes rigorous results from approximate dynamic programming (ADP) ([7, 10, 11]) and heuristics to extend the work done in the field of optimal execution problem. In particular, no model for liquidity dynamic, or market impact, is assumed but this is continuously estimated while trading, using tick data and market microstructure. This additional feature of the algorithm is critical to adapt to a variety of market environments, and to dynamically react not only to variations on the state variable, i.e. the order book, but also to structural changes of the market conditions. Effectively, the proposed algorithm acts, at the same time, as a real-time estimator of the market impact function and as an optimiser given that market impact function. This give a natural interpretation to the name of the algorithm, actor-critic, where the critic part estimates the future reaction of the market to a given trade and the actor part estimates the best action to take.

The paper is organised as follows. In Section 2 the dynamic programming problem is formally stated and the use of approximate dynamic programming methods are motivated, alongside with the description of the actor-critic algorithm and its extension to include limit orders as part of the trading strategy. In Section 3 the algorithm is applied to orders in the bond futures market, using tick data level information which represent an advance of the work present in literature. In this section the algorithm is backtested against historical data and the result obtained from a forward testing against live market data is presented as well. The section concludes with a sensitivity analysis of the algorithm's performance against different trade parameters as trade size, trade frequency and trade horizon. In Section 4 a distributed implementation of the algorithm is presented to show how it is possible to scale-up the algorithm to be use by multiple and concurrent trading desk without an explosion on resource requirements, while cross-exploiting information between different traders and desk to get a more accurate picture of the market conditions. Finally, in Section 5 an agenda is set against future research challenge which takes into account the effect of game theoretic aspect of the trading problem and the opportunity to include prediction features to the algorithm by the use of econometric models.

Notational Conventions: Throughout the paper all vectors are assumed to be column vectors. We use lower case boldface letters to denote vectors and for economy of space we write $\mathbf{x} = (x_1, \ldots, x_R)$ for the column vector $\mathbf{x}$. $\mathbf{x}'$ denotes the transpose of $\mathbf{x}$, $||\mathbf{x}||$ the Euclidean norm of $\mathbf{x}$, $\mathbf{0}$ the vector of all zeros, $\mathbf{e}$ the vector of all ones, and $\mathbf{e}_i$ the ith unit vector. We use upper case boldface letters to denote matrices and write $\mathbf{I}$ for the identity matrix and $\mathbf{0}$ for the matrix of all zeros. The notation $\boldsymbol{A} > \boldsymbol{B}$ represents an element-wise comparison.

2. Methodology

In this paper the optimal execution problem is approached using Actor-Critic algorithms which belong to Approximate Dynamic Programming methods (ADP). ADP methods aim to solve Dynamic Programming Problem without incurring in the curse of dimensionality problem which affects exact algorithms. The main characteristic of ADP method is to find an exact solution to the Dynamic Programming Problem constrained to a certain set of feasible solutions. Usually, the subset of feasible solutions is described by a parametrised family of functions, which reduce the Dynamic Programming Problem from an infinite dimensional problem (in the case of continuous state space) to a finite dimensional problem whose dimension is determined by the number of parameters describing the chosen set of feasible functions.

In particular, Actor-Critic algorithms are characterised by two different kind of approximations. On one side, the control policy, i.e. the solution to the Dynamic Programming Problem, is approximated with a parametrised family of functions, on the other side also the Q-function, associated with the original Dynamic Programming Problem, is approximated within a, potentially, different set of parametric functions. The Actor-Critic algorithm then proceed iteratively simultaneously updating the two set of parameters while new observations regarding the pair state-control and the relative cost are gathered.

The advantage of the Actor-Critic algorithm is that it does not require to specify the differential equation governing the state dynamic, because all the information needed to build the solution to the Dynamic Programming Problem are extracted from the on-line observations. This feature of the algorithm is a key capability to address the issue with the current solution of the optimal execution problem, since it guarantees adaptability to changing market conditions, i.e. changing on liquidity dynamic, and robustness to "a priori" assumptions regarding the underlying behaviour of prices and liquidity. The price to pay for that is the restriction to a predetermined set of feasible solutions, which should be dictated by expert knowledge and heuristic reasoning, and the convergence time lag to get the approximated solutions. Regarding the last question about the rate of convergence it is possible to say that the problem can be partially overcome by use of simulation against historical data, i.e. backtesting, to have a "warm" initialisation for the two set of parameters but it will rely ultimately on the convergence properties of the algorithm itself. In the following it will be shown that there are theoretical guarantees that convergence of the parameters is achieved under mild conditions on the quality of the data gathered and that the parameters converge to the approximated optimal solution.

To formally introduce Actor-Critic algorithm and its main convergence results it is necessary to introduce it in the framework of Markov Decision Processes.

Consider a Markov decision process with finite state space $\mathcal{X}$, and finite action space $\mathcal{U}$. Let $c : \mathcal{X} \times \mathcal{U} \longmapsto \mathbb{R}$ be a given reward function. Let $\{\mu_{\boldsymbol{\theta}}, \boldsymbol{\theta} \in \mathbb{R}^n\}$ be a set of *randomized stationary policies (RSPs)*, parametrised in $\boldsymbol{\theta}$. In particular, $\mu_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x})$ denotes the probability of taking the action $\mathbf{u}$ given the state $\mathbf{x}$, under the *RSP* θ. The sequence of states $\{\mathbf{x}_k\}$ and the sequence of state-control pairs $\{(\mathbf{x}_k, \mathbf{u}_k)\}$ of the Markov decision process, generated by an *RSP* θ, form Markov chains for every $\boldsymbol{\theta}$.

We are interesting in finding the parameter vector $\boldsymbol{\theta}$ that maximizes the average reward function $\overline{\alpha} : \mathbb{R}^n \longmapsto \mathbb{R}$, given by:

$$\overline{\alpha}(\boldsymbol{\theta}) = \sum_{\mathbf{x}\in\mathcal{X}, \mathbf{u}\in\mathcal{U}} c(\mathbf{x}, \mathbf{u})\eta_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}). \tag{2.1}$$

For each $\boldsymbol{\theta} \in \mathbb{R}^n$ let us now define a differential reward function $V_{\boldsymbol{\theta}} : \mathcal{X} \longmapsto \mathbb{R}$, as solution of the following Poisson equation:

$$\overline{\alpha}(\boldsymbol{\theta}) + V_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{\mathbf{u}\in\mathcal{U}} \mu_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x}) \left[c(\mathbf{x}, \mathbf{u}) + \sum_{\mathbf{y}\in\mathcal{X}} p(\mathbf{y}|\mathbf{x}, \mathbf{u}) V_{\boldsymbol{\theta}}(\mathbf{y}) \right], \tag{2.2}$$

where $p(\mathbf{y}|\mathbf{x}, \mathbf{u})$ is the probability that the next state is $\mathbf{y}$ given that the current state is $\mathbf{x}$ and action $\mathbf{u}$ is taken. We also need to define the Q-value function $Q_{\boldsymbol{\theta}} : \mathcal{X} \times \mathcal{U} \longmapsto \mathbb{R}$ by:

$$Q_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) = c(\mathbf{x}, \mathbf{u}) - \overline{\alpha}(\boldsymbol{\theta}) + \sum_{\mathbf{y}\in\mathcal{X}} p(\mathbf{y}|\mathbf{x}, \mathbf{u}) V_{\boldsymbol{\theta}}(\mathbf{y}). \tag{2.3}$$

The following useful result is from [8].

Theorem 1 (Average Reward Gradient)**.** *We have*

$$\nabla\overline{\alpha}(\boldsymbol{\theta}) = \sum_{\mathbf{x}\in\mathcal{X}, \mathbf{u}\in\mathcal{U}} \eta_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) Q_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}), \tag{2.4}$$

where

$$\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) = \nabla_\theta \ln \mu_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x}). \tag{2.5}$$

We write $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) = (\psi_{\theta,1}(\mathbf{x}, \mathbf{u}), \ldots, \psi_{\theta,1}(\mathbf{x}, \mathbf{u}))$.

The actor-critic algorithm ([7]) works with a parametrization of the Q-function in terms of a vector $\mathbf{r} = (r_1, \ldots, r_m) \in \mathbf{R}^m$:

$$Q^{\mathbf{r}}_\theta(\mathbf{x}, \mathbf{u}) = \sum_{l=1}^{m} r_m \phi_{\theta,l}(\mathbf{x}, \mathbf{u}).$$

For reasons explained in [7], a typical choice for the features $\phi_{\theta,l}(\mathbf{x}, \mathbf{u})$ is to set $m = n + 1$, $\phi_{\theta,l}(\mathbf{x}, \mathbf{u}) = \psi_{\theta,l}(\mathbf{x}, \mathbf{u})$ for $l = 1, \ldots, n$, and fix $\phi_{\theta,n+1}(\mathbf{x}, \mathbf{u})$ to the constant function that is everywhere equal to one except at $\mathbf{x} = \mathbf{0}$ where $\phi_{\theta,n+1}(\mathbf{0}, \mathbf{u}) = 0$ for all $\mathbf{u}$. The algorithm interchanges estimation of the parameter $\mathbf{r}$ based on observations from a sample path of the Markov process (critic) with gradient-based updates on θ using an estimate of $\nabla\overline{\alpha}(\boldsymbol{\theta})$ from the same sample path (actor).

In the following two versions of the actual updating iteration for the critic parameters are introduced. The critic iteration is based on a temporal difference algorithm and we differentiate between the $TD(1)$ and the $TD(\lambda)$ version. Let $\alpha_k \in \mathbf{R}$ the average reward estimate at time k, $\mathbf{r}_k \in \mathbf{b}R^m$ the estimate of parameter vector $\mathbf{r}$ at time k, and $\mathbf{Z}_k \in \mathbf{b}R^m$ the estimate of Sutton's eligibility trace at time k. Let

also $\boldsymbol{\theta}_k \in \mathbb{R}^n$ be the actor parameter vector at time k. The updates take place at states-action pairs visited by a single sample path (potentially generated by a simulation) of the Markov process. Let $(\mathbf{X}_k, \mathbf{U}_k)$ be the state-action pair sampled at time k. The critic update equations are as follows:

$$\alpha_{k+1} = \alpha_k + \gamma_k(c(\mathbf{X}_{k+1}, \mathbf{U}_{k+1}) - \alpha_k), \tag{2.6}$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \gamma_k d_k \mathbf{Z}_k, \tag{2.7}$$

$$d_k = c(\mathbf{X}_k, \mathbf{U}_k) - \alpha_k + \mathbf{r}_k^{'} \boldsymbol{\phi}_{\boldsymbol{\theta}_k}(\mathbf{X}_{k+1}, \mathbf{U}_{k+1}) - \mathbf{r}_k^{'} \boldsymbol{\phi}_{\boldsymbol{\theta}_k}(\mathbf{X}_k, \mathbf{U}_k), \tag{2.8}$$

where in the $TD(1)$ case

$$\mathbf{Z}_{k+1} = \begin{cases} \mathbf{Z}_k + \boldsymbol{\phi}_{\boldsymbol{\theta}_k}(\mathbf{X}_{k+1}, \mathbf{U}_{k+1}), & \text{if } \mathbf{X}_{k+1} \neq \mathbf{x}^* \\ \boldsymbol{\phi}_{\boldsymbol{\theta}_k}(\mathbf{X}_{k+1}, \mathbf{U}_{k+1}) & \text{otherwise,} \end{cases} \tag{2.9}$$

and in the $TD(\lambda)$ case, for $0 < \lambda < 1$,

$$\mathbf{Z}_{k+1} = \lambda \mathbf{Z}_k + \boldsymbol{\phi}_{\boldsymbol{\theta}_k}(\mathbf{X}_{k+1}, \mathbf{U}_{k+1}). \tag{2.10}$$

In the above γ_k is a positive step-size parameter, and $\mathbf{x}^*$ is a special state that the Markov process visits infinitely often.

The actor update iteration is as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \beta_k \Gamma(\mathbf{r}_k) \mathbf{r}_k^{'} \boldsymbol{\phi}_{\boldsymbol{\theta}_k}(\mathbf{X}_{k+1}, \mathbf{U}_{k+1}) \boldsymbol{\psi}_{\boldsymbol{\theta}_k}(\mathbf{X}_{k+1}, \mathbf{U}_{k+1}), \tag{2.11}$$

where $\Gamma(\cdot)$ is a scalar that controls the step-size β_k on the basis of the value of $\mathbf{r}_k$.

Given the actor-critic algorithm described in this section it is possible to state the following theorem proved in [7] and in a more general version allowing a distributed implementation in [11]. (We point to these two references for detailed descriptions of the mild technical assumptions needed for the theorem to hold.)

Theorem 2 (Distributed Actor-Critic)**.** *The sequence $\{\boldsymbol{\theta}_k\}$ generated according to the actor-critic algorithm satisfies:*

(a) *in the $TD(1)$ case*

$$\liminf_{k \to \infty} ||\nabla \overline{\alpha}(\boldsymbol{\theta}_k)|| = 0, \quad w.p.1;$$

b) *in the $TD(\lambda)$ case, for each $\epsilon > 0$, there exists λ such that*

$$\liminf_{k \to \infty} ||\nabla \overline{\alpha}(\boldsymbol{\theta}_k)|| < \epsilon, \quad w.p.1.$$

So far, the actor-critic algorithm and main convergence results have been described, what is left is to identify a suitable parametrised family of control policies to use as $\mu_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x})$ in the optimal execution problem. This step of the methodology is unavoidably heuristic and guided by expert knowledge of the problem and intuition, here it is were science becomes an art to cope with multifaced and complex real-world issues. Additionally, we need to formally define the optimal execution problem as a Markov Decision Process, e.g a state space, a control space and a reward function are needed.

The state space for the optimal execution problem comprises all the relevant information needed to take a trading decision, which in the context of a order driven market are represented by the limit order book (the effect of macroeconomic news and announcement is discarded in our approach) and the number of shares left to buy. Formally, we represent with x_k a measure of liquidity extracted from the limit order book, with p_k the price of the asset and with w_k the number of

remaining shares to buy (or to sell) at time k. More in detail, $\mathbf{x}_k$ is calculated as the difference between the sum of the sizes on the bid side minus the sum of the sizes on the ask side of the limit order book, dived by the sum of all the positions in the limit order book. The control space is at first limited to only market order, which are bounded from below by 0 (it is not allowed to sell during a buy order and not to buy during a sell order) and from above by the remaining number of shares to buy. We represent the number of shares to buy at time k as u_k. The reward associated with the pair state-order is represented by the average price obtained in the trade and our objective, in the case of a sell order, is to maximise the some of the reward obtained while filling the full order (or to minimise the some of the costs in the case of a buy order). We are interested in the solution of a particular case of the Markov Decision Process previously described, where we are fixing the time horizon over which we want to maximise (minimise) our reward. This is called an "episodic" Markov Decision Process and the length of the episode is fixed to T steps. Moreover, we restrict our control policies to be deterministic rather than to be random.

The parametric family of control policies with which we are constrained the Dynamic Programming problem are composed by three factors. The first factor, modelled by the following function,

$$F_1(p_k, x_k) = w_k e^{\frac{p_k^2 \ln(\frac{1}{T-k})}{p_0^{2e^{2f_1(\theta_k^1)k}}}} \tag{2.12}$$

captures and tries to exploit the opportunity carried on by the current price of the asset. The main characteristic of this factor are that it descents with price (if buying), it buys the remaining shares divided by the time left to the end of the episode T if the price is growing as "expected". Our expectation on the price trend are linked to the parameter θ_k^1 which is the first component of the actor parameter vector, at time k. In 2.12, $f_1(\cdot)$ is a continuous function which is bounding the actor parameter value and p_0 is the asset price at the beginning of the episode.

The second factor, modelled by the following function,

$$F_2(p_k, x_k) = K_1 - \frac{K_2}{1 + 2e^{-f_2(\theta_k^2)x_k}} \tag{2.13}$$

takes into account the effect of unbalance in the limit order book liquidity and it is equal to 1 if ask and bid side are in equilibrium while it amplify or dumps 2.12 if one side of the book is prevailing. In 2.13 K_1, K_2 and $f_2(\cdot)$ provide bound to the strength of the corrections, reflecting our beliefs on the importance of the liquidity unbalance and on the second component of the actor parameter vector θ_k^2.

The third factor, modelled by the following function,

$$F_3(p_k, x_k = e^{-2\frac{k - f_1(\theta k^1)}{f_3(\theta_k^3)}} \tag{2.14}$$

takes in the account the traders preferences about buying (selling) more at the beginning rather than at the end of the episode and it is a measure of risk aversion. In 2.14 f_3 provides bound for the third component of the actor parameter θ_k^3 which can be though of as shaping the "aggressiveness" of the trading strategy.

As result, at any point in time k the number of shares to buy (or sell) as market order is given by

$$u_k(p_k, x_k) = F_1(p_k, x_k)F_2(p_k, x_k)F_3(p_k, x_k) \tag{2.15}$$

To allow additional flexibility in the trading strategy the actor-critic algorithm has been extend with a simple first passage time (FPT) model, which is giving the capability to fire not only market orders but limit orders as well. The introduction of limit orders capabilities has never been done or analysed in previous work in literature. Hence, even in this simple form it represents a strong advancement on the practice of automated optimal execution.

The idea is as follow. The movement of the market price of the asset are modelled as a geometric Brownian Process with constant drift and volatility. Then, the drift and volatility parameters of the model are continuously updated using tick data observations from the market. At the beginning of the episode the model for the market price movement is used to estimate the probability that at the next step the mid-point price will drop (or will rise for a sell) below the current best ask. Proportionally to such probability, let's name it p_{drop} the remaining number of shares to buy (or to sell), w_k are split and a limit order for $p_{drop}w_k$ number of shares is issued with limit price equal to the current best ask while $w_k^{'} = (1 - p_{drop})w_k$ number of shares are executed using the actor-critic algorithm. At the next step the any number of shares of the limit order that have not been executed are added to the remaining number of shares to buy for the total order and the process is repeated. Note that to avoid a large, and risky, market order at the very end of the episode the "splitting" process is terminated a certain number of steps before the end of the episode.

3. Application

The algorithm proposed in Section 2 has been tested with orders for the bond futures market, both against historical data and against live market data using a pre-production implementation of the algorithm. Figure 1, 2, 3, 4, 5 and 6 show the forward testing results.

Complete the session with figures from backtesting, tables with a summary of the results and an investigation and discussion of the sensitivity to different trading parameters, i.e. trading horizon, trade size (and potential underlying dynamic of the asset's price).

4. Distributed Implementation

The algorithm described in Section 2 it is much more computationally efficient than a traditional dynamic programming algorithm but can still suffer of scalability problems when used by a large number of traders and across several trading desks. The scalability issue can be approached leveraging from some recent results that suggest an efficient way of distributing the implementation of control policies, sharing information across a variable communication networks ([6, 11]).

In the context of our problem it would be useful for the traders to share information about the market conditions they observe while trading. This is exactly what the distributed implementation does while keeping the computational requirements low. Every time a trader invoke the best execution algorithm a trading strategy is created with a set of initial parameters. Such initial parameters get updated while the trade progress as a response of the changing market conditions. Ideally, what is needed is a mechanism to propagate such changes in the algorithm's parameters across traders running the algorithm at the same time and across different orders placed at different times.

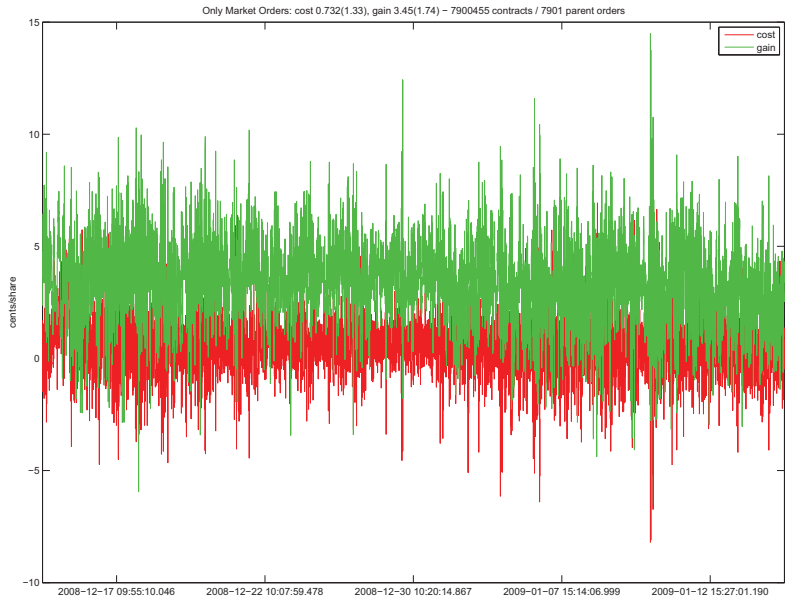

FIGURE 1. Evolution of cost against arrival price (red) and gain against the execution of the full order at once (green). Using only market orders.

In this paper the following mechanism is proposed to achieve such objective. For each traded instrument there exists a set of default parameters called the *Master Context*. Then every time a trader starts the execution of the Actor-Critic Algorithm for a new order a set of parameters specific for that order is created, *The Context*. The initialisation of *The Context* for a particular order, let's name it $O*$, is different whether other strategies are running or not at the moment of the creation of $O*$. If there are no other strategies running, *The Context* for $O*$ is initialised as a copy of the current *Master Context*, if there are other strategies running, *The Context* for $O*$ is initialised as a copy of the closest order to $O*$ running at the moment. (It is assumed that a distance between orders is defined).

To correctly update the Actor-Critic parameters, every time the Actor-Critic Algorithm is called to decide about the trade of a particular order $O*$, *The Context* of $O*$ and all the other contexts existing at the moment are sent as input for the Actor-Critic Algorithm. If the number of strategies running at the same time is too large, and passing all the contexts every time becomes too expensive, it is possible to select only the closest orders to $O*$, i.e. all the orders within a certain threshold distance or only the closest M orders, for passing their contexts to the Actor-Critic Algorithm at the moment it is executed for $O*$.

At the end of the trading period associated with an order $O*$, if there are not other orders running at the moment, *The Context* associated with $O*$ becomes the new *Master Context* otherwise *The Context* associated with $O*$ is not used any more.

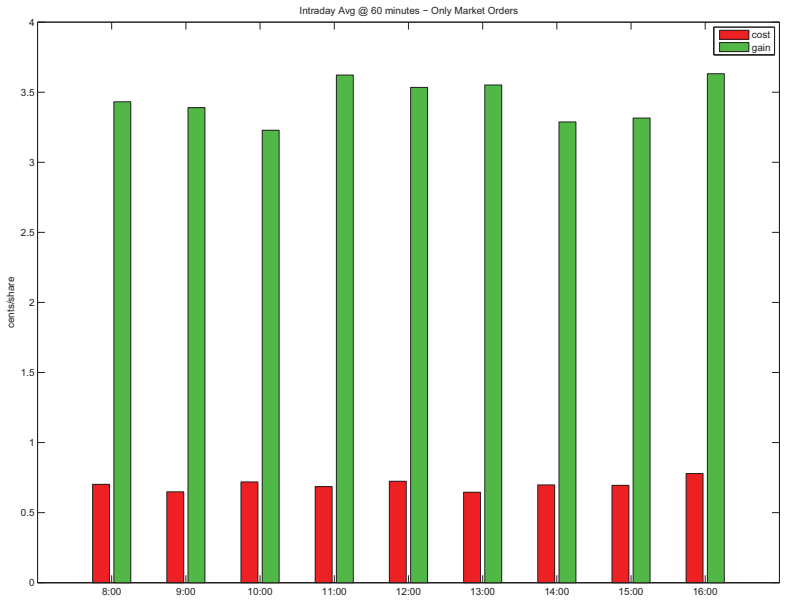

FIGURE 2. Intraday evolution of cost against arrival price (red) and gain against the execution of the full order at once (green). Using only market orders.

5. Future Research

Future research should set the optimal execution problem in the context of a game theoretic framework to take into account the effect of other player of the market who are potentially observing our trades and play strategically to take advantage of our trading patterns. Additionally, the need to incorporate econometric model to be used as predictors in the Actor-Critic algorithm should be fully investigated.

6. Conclusions

Stress the contribution of the paper, the advance with respect the current literature and the numerical results observed in both back and forward testing.

References

[1] Robert Almgren. Optimal execution with nonlinear impact functions and trading-enhanced risk. *Applied Mathematical Finance*, 10(1):1–18, 2003.
[2] Robert Almgren and Neil Chriss. Value under liquidation. *Risk*, 12(12):61–63, 1999.
[3] Robert Almgren and Julian Lorenz. Bayesian adaptive trading with a daily cycle. *Journal of Trading*, 1(14):38–46, 2006.
[4] Robert Almgren and Julian Lorenz. Adaptive arrival price. In Brian R. Bruce, editor, *Algorithmic Trading III: Precision, Control, Execution*, pages 59–66. Institutional Investor Journals, 2007.

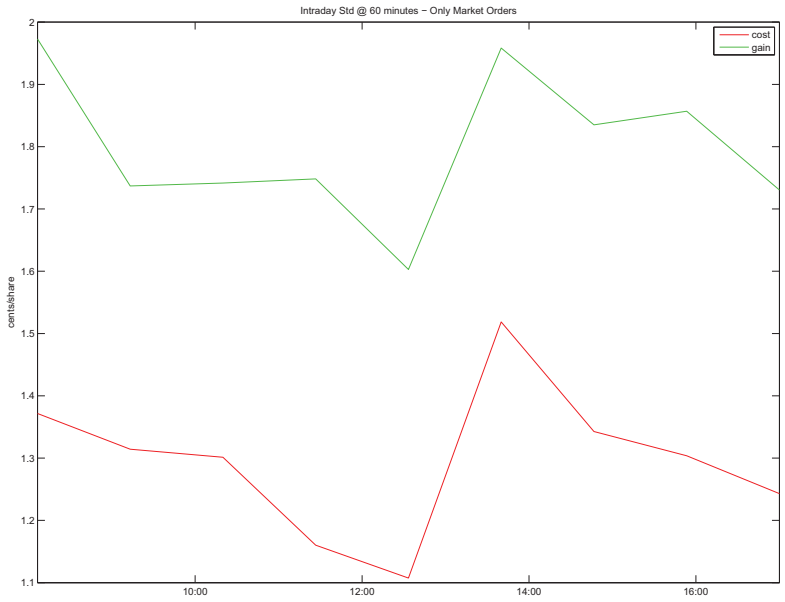

FIGURE 3. Intraday evolution of risk with respect to arrival price (red) and to the execution of the full order at once (green). Using only market orders.

[5] Dimitris Bertsimas and Andrew W. Lo. Optimal control of execution costs. *Journal of Financial Markets*, 1(1):1–50, 1998.
[6] Giuseppe Conte and Paris Pennesi. The rendezvous problem with discontinuous control policies. Accepted for publication on IEEE Transactions on Automatic Control, 2009.
[7] Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.
[8] Peter Marbach and John N. Tsitsiklis. Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, 2001.
[9] Anna Obizhaeva and Jiang Wang. Optimal trading strategy and supply/demand dynamics. NBER working paper, No. 11444, 2006.
[10] Paris Pennesi and Ioannis Paschalidis. Solving sensor network coverage problems by distributed asynchronous actor-critic methods. In *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, Louisiana, December 2007.
[11] Paris Pennesi and Ioannis Paschalidis. A distributed actor-critic algorithm and applications to mobile sensor network coordination problems. Accepted for publication on IEEE Transactions on Automatic Control, 2009.
[12] André Pérold. The implementation shortfall: paper versus reality. *Journal of Portfolio Management*, 14(2):4–9, 1988.

(P. Pennesi) RBS GLOBAL BANKING & MARKETS,
135 BISHOPSGATE, LONDON, EC2M 3UR, UK
E-mail address, P. Pennesi: `paris.pennesi@rbs.com`

(G. Darbha) RBS GLOBAL BANKING & MARKETS,
135 BISHOPSGATE, LONDON, EC2M 3UR, UK
E-mail address, G. Darbha: `gangadhar.darbha@rbs.com`

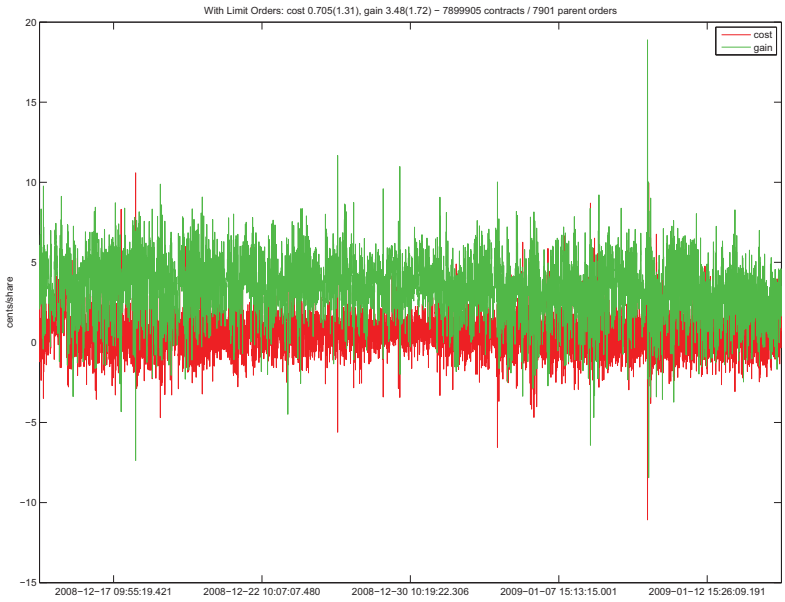

FIGURE 4. Evolution of cost against arrival price (red) and gain against the execution of the full order at once (green). Using both market and limit orders.

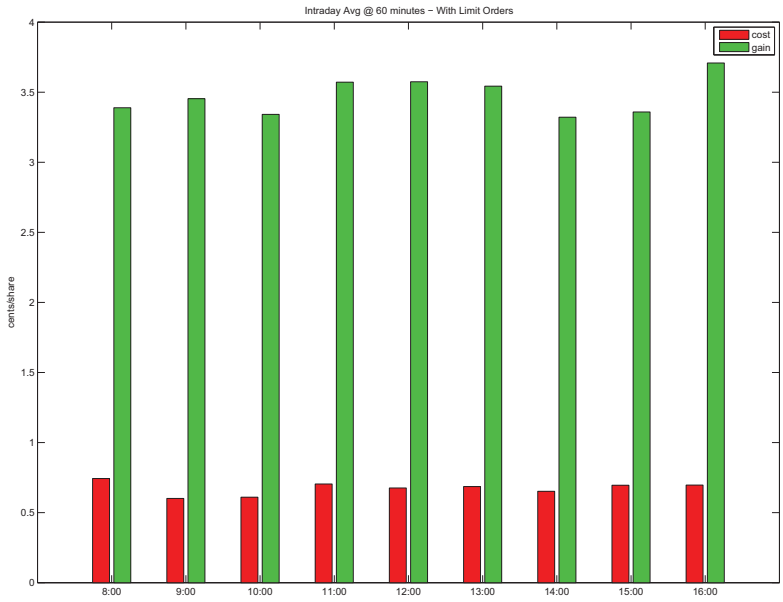

FIGURE 5. Intraday evolution of cost against arrival price (red) and gain against the execution of the full order at once (green). Using both market and limit orders.

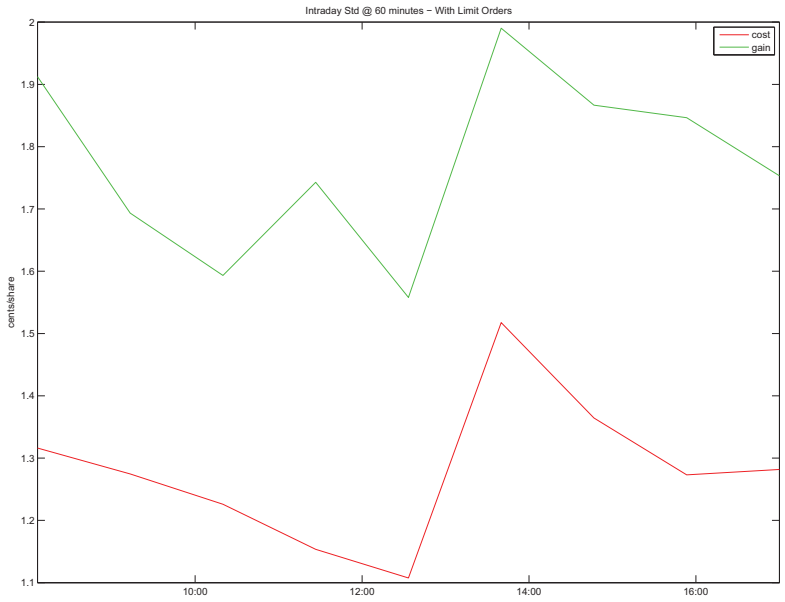

FIGURE 6. Intraday evolution of risk with respect to arrival price (red) and to the execution of the full order at once (green). Using both market and limit orders.