Final Project - Limit Order Routing in Fragmented Markets

Jean-Francis Carignan (20002083)

IFT6521 - May 2020

1 Problem description

In this report, the problem of limit order routing in fragmented markets is considered. In a first time, the problem is described. Let's start by an important definition:

Limit Order: A limit order is an order to buy (sell) a specific quantity of an asset at a maximum (minimum) price. A transaction will only take place if a counter-party is willing to transact at that same price, in which case the order is filled. A limit order may be partially filled and have remaining quantity. It can also stay unfilled for an indefinite amount of time and be cancelled.

The electronic place where this matching between buyers and sellers takes place is called a limit order book. A detailed description of the matching mechanism is provided by Bouchaud et al. [2018].

Nowadays, many financial markets are fragmented. This means that the same financial asset can be traded in different limit order books and under different conditions. Some popular examples of fragmented markets are U.S. equity markets or foreign exchange (FOREX) markets. Exchanges (limit order books) are the constituents of a fragmented market. They are owned by private entities with their own interests. Exchange owners aim to maximize the share of total trading happening on their exchange, because they make a profit every time a buyer and a seller agree to transact on it. Owners usually pay traders who us limit orders a rebate to attract more of these orders to their exchange. That means, for example, that every one dollar stock X sold with a limit order on exchange Y will return to its owner an 1.02$ payoff if the rebate paid is 2%. Paying investors for using limit orders has its advantage; it makes an exchange more liquid. Liquidity is a sought after characteristic for more impatient traders, who prefer using different types of orders than limit. Exchange owners can charge these traders a fee for enjoying liquidity. The difference between the fee charged to impatient traders and the rebate rewarding limit orders is their profit margin.

Usually, limit orders compete on more liquid exchanges where rebates are high. This translates in longer delays before a seller counter-party is found. On the contrary, exchanges where rebates are lower have less competing limit orders, thus execution is faster in general. Colliard and Foucault [2012] and Malinova and Park [2015] find empirical evidence of this phenomenon. If patient investors are in fact not so patient and a bit concerned about lengthy execution, they might seek a compromise between a faster execution and a higher rebate.

The goal of this project is to find a solution for routing limit orders between different exchanges based on an investor's preference for fast execution and high rebate.

2 Modelling

The order routing problem is modelled as an infinite horizon stochastic shortest path problem, where a trader must decide at each step (1 second interval) how many shares to buy on different exchanges using only limit orders. The trader will eventually have bought all the desired shares and reach a terminal state, thus no discounting factor of the future costs is used. Because price moves are less relevant at shorter time scales, suppose that the stock price is the same on all exchanges and is constant. This price is assumed to be 1$ for simplicity.

2.1 States

The trader initially wants to buy x_0 shares. At step k, the trader has a remaining number of shares to buy (inventory) $x_k > 0$. If $x_T = 0$ for some T, then the trader has bought all desired shares and T is a termination time.

2.2 Controls

Controls consist of how many shares the trader chooses to buy on each exchange by routing separate limit orders. In this report, only two exchanges are considered:

1. The number of shares routed to the first exchange at step k is $u_k^{(1)}$. This exchange offers a $r_1 = 0.1\%$ rebate for limit orders. The number of shares effectively bought when a limit order is posted there for 1 second follows a Poisson distribution with mean $\lambda_1 = 2$.

2. The number of shares routed to the second exchange at step k is $u_k^{(2)}$. This exchange offers a $r_2 = 0.2\%$ rebate for limit orders. The number of shares effectively bought when a limit order is posted there for 1 second follows a Poisson distribution with mean $\lambda_2 = 1$.

At each step, all limit orders with unfilled quantity are cancelled and the trader can decide how to complete the buying with new orders at next step. At time k, $u_k^{(1)} \in U_k(x_k) = \{0, ..., x_k\}$ and $u_k^{(2)} = x_k - u_k^{(1)}$. Also, $u_k = [u_k^{(1)}, u_k^{(2)}]$ is the control vector.

2.3 Costs

Perturbations are the number of shares that are executed (terms "filled" or "matched" are also used) in each of the two exchanges. These are the number of shares that can, but may not, be effectively bought by the trader during a one second interval for each exchange. They are denoted $w_k^{(1)}$ and $w_k^{(2)}$ respectively at step k. $w_k = [w_k^{(1)}, w_k^{(2)}]$ is the perturbation vector. To simplify notation, let $y_k^{(i)} = max(u_k^{(i)} - w_k^{(i)}, 0)$ be the quantity remaining after a limit order with initial quantity $u_k^{(i)}$ is posted for 1 second on exchange i.

Moreover, let $y_k = [y_k^{(1)}, y_k^{(2)}]$ and let the rebate vector be $r = [r_1, r_2]$. Then, the cost of taking control u_k in state x_k with perturbations w_k is

$$g_k(x_k, u_k, w_k) = g_k(x_k, y_k) = (x_k - y_k) \cdot (1 - r)^T + \underbrace{\beta \times (y_k^{(1)} + y_k^{(2)})}_{\text{Risk aversion}}$$

This cost represents the cash paid to obtain $min(u_k^{(1)}, w_k^{(1)}) + min(u_k^{(2)}, w_k^{(2)})$ shares plus a risk aversion penalty that quantifies the impatience of the trader. Higher levels of β will incite the trader to execute his/her limit orders faster to minimize the risk aversion penalty associated with unfilled quantity $y_k^{(1)} + y_k^{(2)}$.

2.4 Transition function

The transition function associated with control u_k taken in state x_k given perturbations w_k is

$$x_{k+1} = f_k(x_k, u_k, w_k) = f_k(x_k, y_k) = y_k^{(1)} + y_k^{(2)} = max(u_k^{(1)} - w_k^{(1)}, 0) + max(u_k^{(2)} - w_k^{(2)}, 0)$$

2.5 Recurrence equations

Th recurrence equation associated with the problem will be used by the Value Iteration algorithm. Subscript k is now used to denote the k^{th} iteration of this algorithm. All functions and variables are as previously defined except without subscript k to avoid confusion. The equation is

$$J_{k+1}(x) = min_{u \in U(x)} \mathrm{E}_{W|x,u} \Big[g(x, u, W) + J_k(f(x, u, W)) \Big] = min_{u \in U(x)} \mathrm{E}_{Y|x,u} \Big[g(x, Y) + J_k(f(x, Y)) \Big]$$

where W is the perturbation random vector. The expected value on the right-hand side is computed using the distribution of random vector $Y|u$. $Y|u$ represents the unfilled quantities on each exchange after one second, conditional on the routing of initial quantities u. The conditional distribution $Y^{(i)}|u^{(i)}$, i.e. the quantity remaining for exchange i, is

$$P[Y^{(i)} = y^{(i)}|u^{(i)}] = \begin{cases} 0 & y^{(i)} > u^{(i)} \\ \frac{\lambda_i^{u^{(i)} - y^{(i)}} e^{-\lambda_i}}{(u^{(i)} - y^{(i)})!} & u^{(i)} \geq y^{(i)} > 0 \\ 1 - \sum_{z=0}^{u^{(i)}-1} \frac{\lambda_i^z e^{-\lambda_i}}{z!} & y^{(i)} = 0 \end{cases}$$

This distribution considers three cases. The first one happens when the unfilled quantity in one exchange is higher than the initial limit order quantity and is thus impossible. The second one happens when there is

an unfilled quantity on the exchange after one second and this probability depends on the Poisson distribution of order matching for exchange i. The last one is when a limit order sent to exchange i is completely filled during a one second interval. This happens when the number of matched shares turn out to be higher than the routed quantity $u^{(i)}$ for this exchange. Since Poisson distributions for both exchanges are independent, the recurrence equation for each state x becomes

$$J_{k+1}(x) = min_{u \in U(x)} \Big[\sum_{y^{(1)}} \sum_{y^{(2)}} P[Y^{(1)} = y^{(1)} | u^{(1)}] \times P[Y^{(2)} = y^{(2)} | u^{(2)}] \times [g(x, [y^{(1)}, y^{(2)}]) + J_k(f(x, y^{(1)} + y^{(2)}))] \Big]$$

Where $J_k(0) = 0$ for the absorbing state $x = 0$ at every iteration k.

3 Algorithms

The algorithms used to solve the limit order routing problem are presented in this section. First, the Value Iteration is used to find an exact solution. However, as the number of shares to buy grows, the number of computations necessary for the exact solution explodes. Hence, the Fitted Q-Iteration algorithm with a second order polynomial regression is used to find a good approximate solution.

3.1 Value Iteration

The Value Iteration algorithm for the problem considered is described with pseudo-code 1 and adapted from Bertsekas [2019]. Given a certain tolerance level ϵ, the algorithm loops for each state and over each action to find the action with the lowest expected future cost. This lowest cost J_{min} is then saved for the considered state. For one iteration of the algorithm, vector J is obtained and $J[j]$ represents the expected future costs of being in state j at this iteration. The algorithm stops when the largest update for every elements of J between two iterations is smaller than ϵ. Value Iteration eventually finds an optimal policy given that a sufficient number of iterations is performed, thus ϵ is initially set to a very small value. Finally, the optimal controls $\mu(x)$ for all x are found using the output J of the algorithm and $\mu(x) = argmax_u \, \mathrm{E}_Y[g(x, Y) + J(f(x, Y))]$.

3.2 Fitted Q-iteration

The Fitted Q-iteration algorithm is described by pseudo-code 2. The algorithm was first introduced in Ernst et al. [2003] and later applied in Ernst et al. [2005]. The version of the algorithm in this report uses a polynomial regression to approximate the Q-factors of the problem and this choice of supervised learning algorithm is motivated in the results section. The algorithm maintains two fitted regressors that are iteratively improved. The first one, Q_{last}, is used to compute Q-factor targets and the second one, Q, is fitted on a simulated set of transitions to predict each Q-factor target. After the second regressor Q is fitted, we assign $Q_{last} \leftarrow Q$ an repeat the process. The simulated set consists of n simulated transitions which are reused at each iteration. The transitions are generated by picking a random action in a random state and observing the resulting cost and next state according the the model specified in previous section. An approximate solution for the Fitted Q-iteration algorithm is obtained when the maximum absolute difference between the prediction of the two polynomial regressions for each pair (x, u) is smaller than a tolerance ϵ. The algorithm finally returns Q and an estimator for $\mu(x)$ can be obtained by taking $\mu(x) = argmax_u Q(x, u)$.

Algorithm 1 Value Iteration

J = zeros vector of size x_0
J_{last} = infinity vector of size x_0
ϵ = set to a small value
while max(abs(J_{last}-J)) > ϵ **do**
  for j=1 to len(J) **do**
    for u=0 to j **do**
      $expVal_u = 0$
      J_{min} = inf
      x1 = u
      x2 = j - u
      for y1=0 to x1 **do**
        # Event that quantities y1 and y2 are unfilled on both exchanges
        for y2=0 to x2 **do**
          prob = {(probability of y1) times (probability of y2)}
          g = {Cost associated with y1 and y2}
          y = y1 + y2
          $expVal_u$ += prob × (g + J_{last}[y])
        end for
      end for
      if $expVal_u < J_{min}$ **then**
        $J_{min} = expVal_u$
      end if
    end for
    J[j] = J_{min}
  end for
  $J_{last} = J$
end while
return J

Algorithm 2 Fitted Q-iteration

Q = polynomial regression algorithm with inputs (x, u)
Initialize Q's parameters to zeros and let $Q(x, u)$ be the prediction of Q for inputs (x, u).
Q_{last} = polynomial regression algorithm with inputs (x, u)
Initialize Q_{last}'s parameters to large values.
$simuls$ = simulate n transitions starting from random state x, performing random action u, obtaining costs g and ending in next state x_{next}. Obtain n tuples (x, u, g, x_{next}).
ϵ = set to a small value
while $max_{x,u}$(abs($Q_{last}(x, u)$-$Q(x, u)$)) > ϵ **do**
  targets = [g + $max_{u_{next}}$ Q_{last}(x_{next},u_{next}) for each (x, u, g, x_{next}) in $simuls$]
  Fit Q on inputs [(x, u) for x, u in $simuls$] and labels $targets$
  $Q_{last} = Q$
end while
return Q

4 Results and analysis

Both algorithms are used to find Q-factors and policies for small, medium and large instances of the problem. An initial number of shares to buy x_0 is set to 5, 10 and 20 respectively for these instances and it is the only parameter that changes. The trader has a fixed risk aversion (impatience) coefficient $\beta = .0005$. The same convergence tolerance threshold $\epsilon = 10^{-10}$ is used for the Value Iteration and the Fitted Q-iteration algorithms. Moreover, the number of simulated transitions used for Fitted Q-iteration is $n = 5000$ for all instances.

To compare both methods, curves of the Q-factors at each inventory level are plotted in figures 1 (small instance), 3 (medium instance) and 5 (large instance). It can be seen that the Q-factors are slightly overestimated by the Fitted Q-iteration algorithm when compared to the exact method due to the approximation using simulated transitions. However, it can also be seen that the shape of the Q-factors using polynomial regression approximates the true shape found with Value Iteration. This results in the the minimum cost control for the approximate method being close or equal to the exact method in most states. Thus, second order polynomial regression seems like a good supervised learning algorithm to approximate the Q-factors for this problem.

A policy comparison for both methods is presented in figures 2, 4 and 6 for the small, medium and large instances respectively. We can observe that the number of shares routed to each exchange is similar for both methods and is almost equally split between the two exchanges when the inventory is in the medium range. When the inventory is high, more shares are routed to the first exchange. An intuitive explanation is that the trader prefers to buy shares faster when he/she has a lot of inventory to buy. To avoid penalty on the unfilled quantity, the trader prefers to route more shares to the first exchange, where rebate is lower but execution is faster. As the trader completes his/her buying, he/she prefers to route more shares to exchange 2 where the rebate is higher and the execution speed is slower. An explanation is that the risk of non-execution is smaller for lower order quantities and thus the trader has a lower chance of getting a penalty on unfilled quantity.

The Fitted Q-iteration algorithm has great computational efficiency when compared to the exact Value Iteration because one does not have to compute the expected value of future costs for each control-state pair. This expected value is extrapolated for most pairs using the polynomial regression's parameters and simulated transitions instead. Although the Fitted Q-iteration method does find good approximate solutions, it makes some mistakes in different regions of the state space. The method tends to underestimate the expected future costs of routing shares to exchange 2 for higher inventory levels and tends to overestimate these costs for lower inventory levels. Thus, using this approximate method, the trader will route less shares than needed to the first exchange when a lot remain to buy and route too many shares to exchange 1 when not many remain to buy (see the larger instance policies of figure 6 for a clear example).

4.1 Small instance

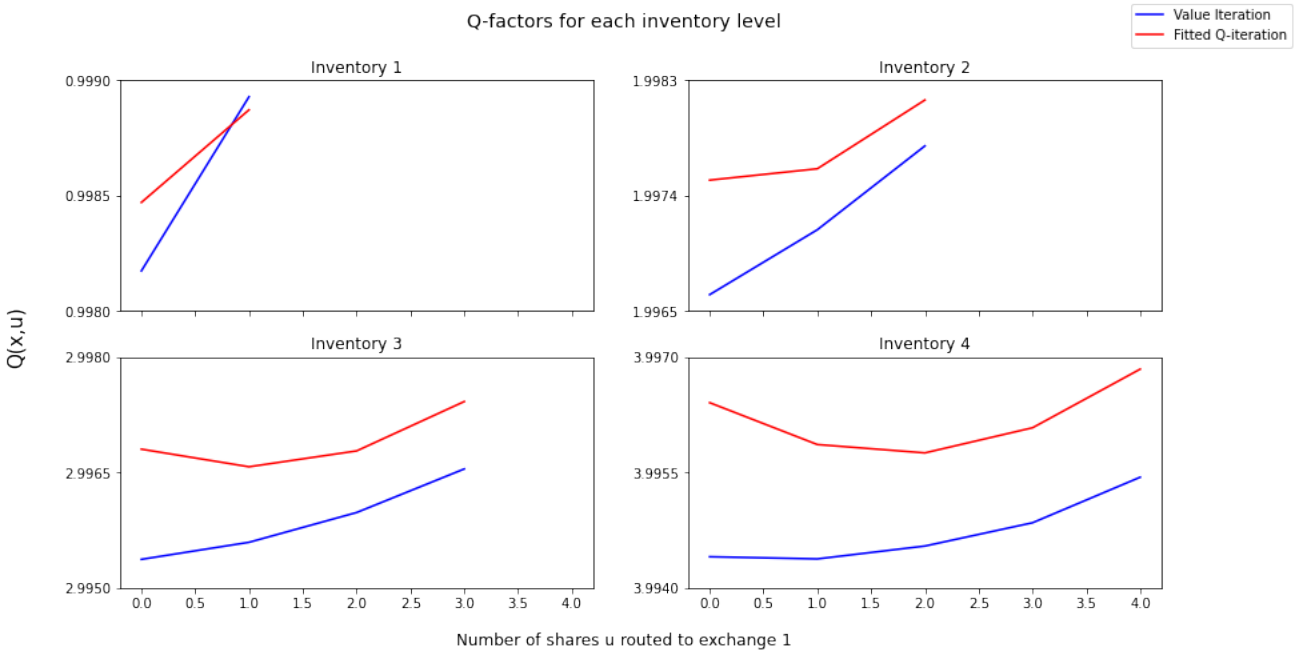

Figure 1: Q-factors found for the exact Value Iteration and the Fitted Q-iteration algorithm for the small instance.

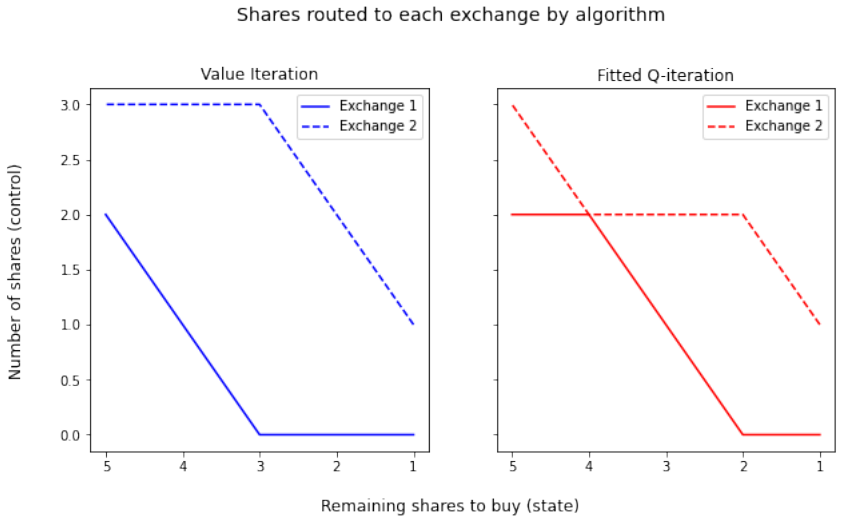

Figure 2: Policy comparison for the small instance. Shares routed to each exchange given number of shares left to buy for Value Iteration and Fitted Q-iteration algorithms.

4.2 Medium instance

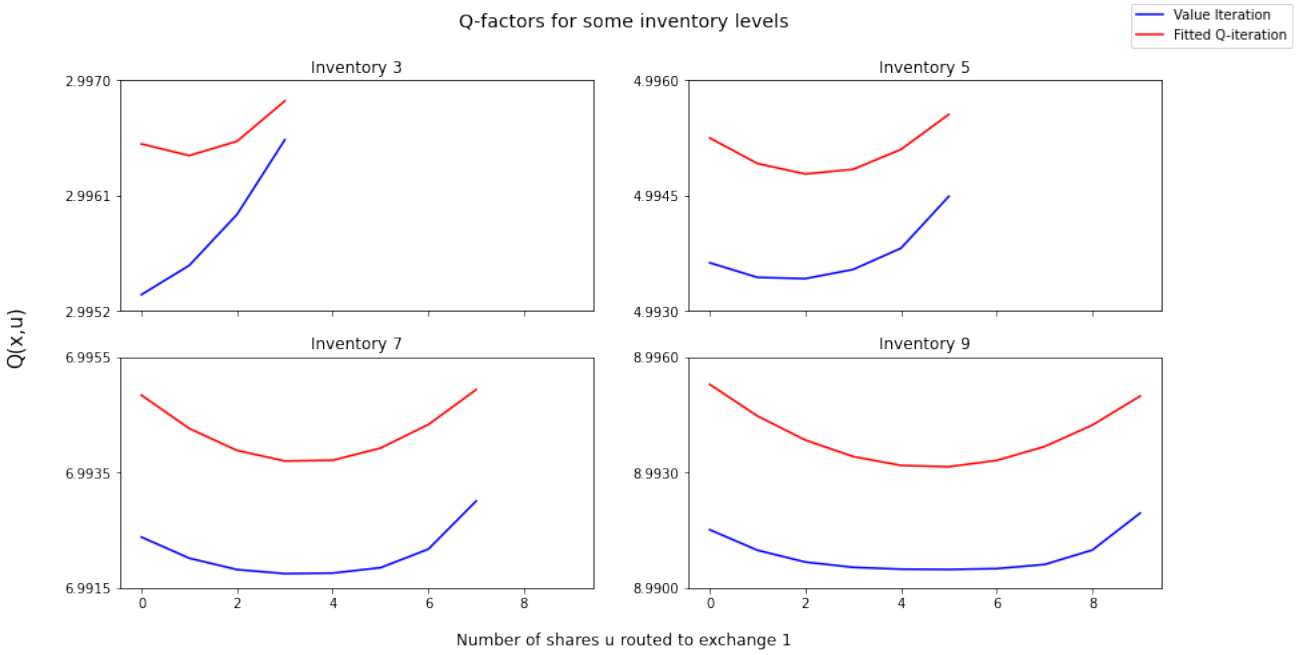

Figure 3: Q-factors found for the exact Value Iteration and the Fitted Q-iteration algorithm for the medium instance.

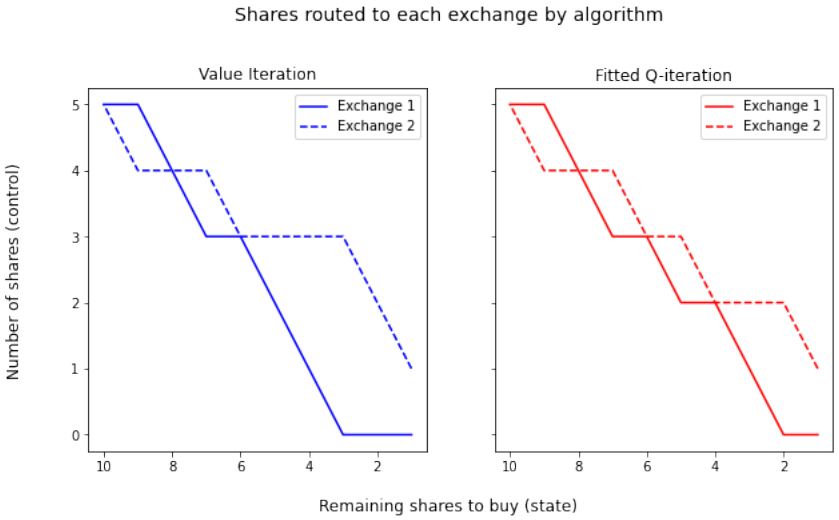

Figure 4: Policy comparison for the medium instance. Shares routed to each exchange given number of shares left to buy for Value Iteration and Fitted Q-iteration algorithms.

4.3 Large instance

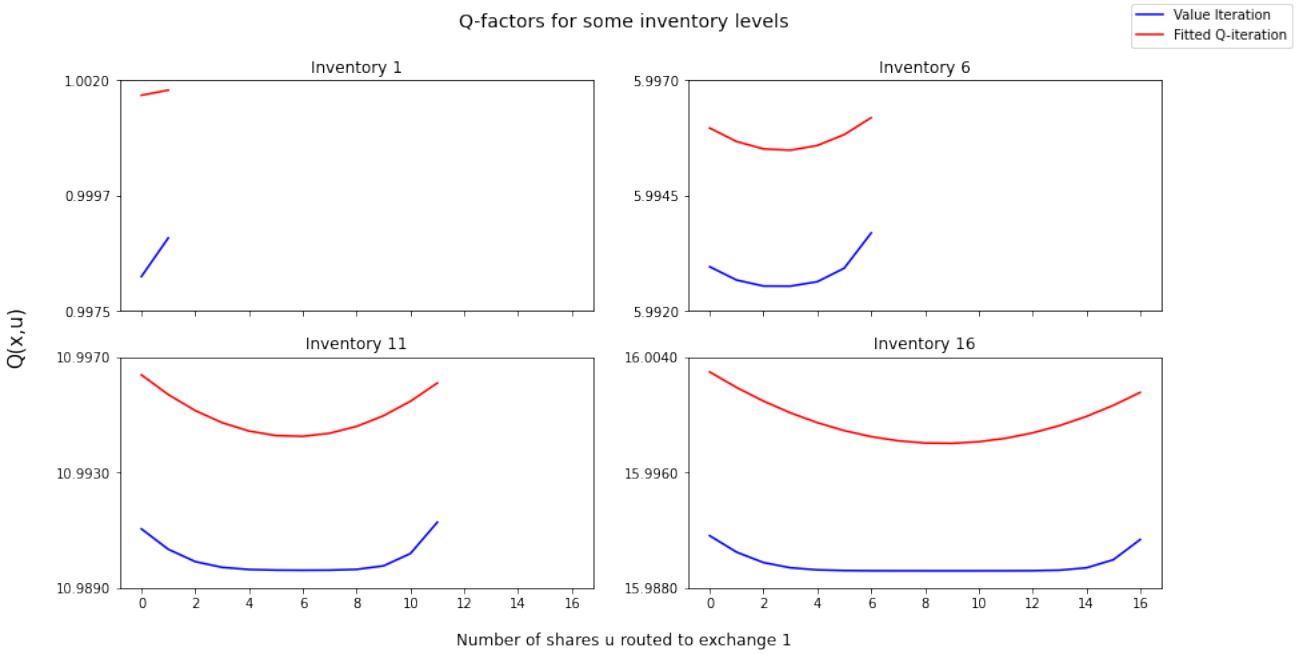

Figure 5: Q-factors found for the exact Value Iteration and the Fitted Q-iteration algorithm for the large instance.

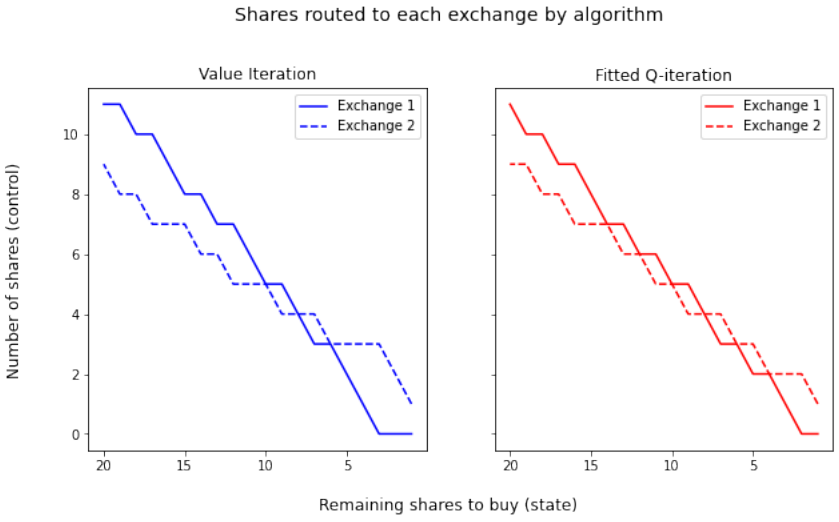

Figure 6: Policy comparison for the large instance. Shares routed to each exchange given number of shares left to buy for Value Iteration and Fitted Q-iteration algorithms.

5 Conclusion

In this report, a model for solving the limit order routing problem in fragmented markets has been proposed. An important assumption made is that the number of shares filled when routing a limit order to an exchange depends on the arrivals of counter-party orders, which are in turn Poisson distributed with mean λ during a one second interval. With this assumption, a distribution for the number of shares filled for a limit order routed to any specific exchange has been derived. The problem has then been formulated as a stochastic shortest path problem with infinite horizon. The setting for which a trader, subject to risk aversion (impatience) initially wants to buy x_0 shares and has access to two different exchanges with different rebates and order filling processes has been considered. An exact solution has been found with the Value Iteration algorithm and an approximate method using Fitted Q-iteration has also been proposed. The approximate method finds solutions close to the optimal ones on all instances considered while being computationally faster for larger state spaces. It is faster because it relies on simulating transitions in the state-control space and estimating the Q-factors using a polynomial regression. The polynomial regression has shown to approximate well the shape of the exact Q-factors. The main characteristic of the solutions found is that the trader should choose exchanges where the rebate is lower but the execution is more likely when he/she has more shares to buy. As the trader's limit orders get filled, he/she can send a bigger proportion of shares to higher rebates and slower execution exchanges.

6 Instructions

To run the code, it is necessary to have Python and its package manager PyPI installed. The code is inside the Jupyter Notebook submitted with the rest of the project. To open it, first launch a terminal or command window and then follow these steps:

- enter: pip3 (or pip) install jupyter numpy matplotlib scipy scikit-learn
- navigate to the folder containing the .ipynb file with the code
- enter: jupyter notebook
- a new tab will open in your browser and you can open the code file from there
- To run the complete code click on the double arrow button with description: "Restart the kernel, then re-run the whole notebook".

The code is split in 4 sections following this order:

1. Experiment parameters
2. Value Iteration algorithm
3. Fitted Q-iteration algorithm
4. Plotting

References

Dimitri P Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

Jean-Philippe Bouchaud, Julius Bonart, Jonathan Donier, and Martin Gould. *Trades, quotes and prices: financial markets under the microscope*. Cambridge University Press, 2018.

Jean-Edouard Colliard and Thierry Foucault. Trading fees and efficiency in limit order markets. *The Review of Financial Studies*, 25(11):3389–3421, 2012.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Iteratively extending time horizon reinforcement learning. In *European Conference on Machine Learning*, pages 96–107. Springer, 2003.

Damien Ernst, Mevludin Glavic, Pierre Geurts, and Louis Wehenkel. Approximate value iteration in the reinforcement learning context. application to electrical power system control. *International Journal of Emerging Electric Power Systems*, 3(1), 2005.

Katya Malinova and Andreas Park. Subsidizing liquidity: The impact of make/take fees on market quality. *The Journal of Finance*, 70(2):509–536, 2015.

Self-assessment

1. Problem description: A+. It is original and supported by references.
2. Modelling: A. The notation is clearly explained an divided in subsections for each component.
3. Algorithms: A+. Both a pseudo-code and a text description are provided for the two algorithms used.
4. Results and analysis: A. Results are presented in clear plots and every axis is labelled. Experiment parameters for each instance are specified. A detailed description and analysis of the main findings for all instances is also given and an intuitive explanation of the results is provided.
5. Conclusion: A+. All the main steps of the report are summarized and the most important ideas and findings are reminded.
6. Other parts: A+. None is missing and they are all complete.
7. Code: A+. The code is clearly organized and the logic is explained with comments. It follows the structure of the report with various markdowns throughout the Jupyter Notebook to indicate what each piece does.
8. Poster: A+. The poster is aesthetically nice and contains all relevant parts of the report in an organized way. The most interesting results, i.e. for the large instance, are presented.