Optimal order execution. S&P's test problem

Boris Garbuzov

PhD Statistics, University of Toronto, boris.garbuzov@gmail.com
30 April 2023,
Toronto, Canada

1. Problem setting

1.1 S&P's problem formulation

A portfolio manager has several bond positions that are potentially illiquid. They are concerned that buying or selling large quantities of these positions may move the market price in an adverse way. They want a software solution that can help them make decisions on when and how many positions to buy or sell.

If you are assigned to design a methodology to help this portfolio manager, how would you approach it?

Clearly state:

- What assumptions are you making and why?
- What data would you need (you can assume such data is accessible) and why?
- What modeling techniques would you use and why?
- What are some cases where your methodology may fail to give reasonable results and why?

1.2 Author's problem characterization

It is often necessary to change the position in an illiquid portfolio. For example, such a portfolio may have over-the-counter assets (not necessarily bonds) or any assets of large volume. Then we face many institutional and technical issues to minimize the cost. For example, we may have an agency problem. Our trusted broker or an investment bank may forerun our order, or just leak the information, so the rumor of a specific large transaction may spread, making other market participants forerun us. Here, we assume perfect agency, and conentrate only on modeling the market impact. This problem is still very complicated for modeling, as different assets may interact. So most of the basic models consider a single asset.

2. Survey of existing solutions to related problems

2.1 Optimal control approach

This approach assumes minimization of some objective functional, defined on a family of functions. The optimal function is called a minimizer. In our case, the minimizer is a function

$q(t)$ of purchased quantities, defined on a discrete or continuous time set. The objective funtional is the total cost of purchasing/selling the assets. It is based on variational calculus theory [Wik23a].

The deterministic optimal control theory is generalized to stochastic control. In this case, the objective funtional can be random. The motivation for this upgrade is due to the consideration of random factors of price change. Particularly, article [qua] is based on this approach.

2.2 Reinforcement learning approach

In the absence of data with explicit xy-pairs, supervised learning becomes impossible, and the reinforcement learning approach could be used, which allows the generation of the necessary dataset on the fly. First of all, this will require creating an environment the way to maximize the reward function. For Python implementations, the Gym library can be used. It provides the base class that needs to be inherited. In that class, the step and reward methods need to be implemented. A prediction model, like neural a network, should predict the actions, supported by the environment.

Our case needs to compute the optimal schedule of purchases/sales. Such conditions as time limit should be specified in the environment. The step function returns the reward. It should discourage purchases for high prices. In other words, the reward should be negatively related to the purchasing price. The better we specify and formalize this problem, the faster will be its training.

The paper [Wen18] provides an example of a reinforcement learning application to the optimal control problem. The paper [Kin19] provides the details on how to create the environment.

2.3 Other approaches

This seminal paper [RN01] proposed a new method known as Almgren-Criss algorithm and based on financial considerations. It gave rise to many software implementations. Time limit does not allow the author to work on its details.

The VWAP (Volume-Weighted Average Price) trading strategy, described in [Wik22], aims to execute trades at a price that is close to the VWAP of the security being traded. The VWAP is calculated by dividing the total value of all trades by the total volume of shares traded during a specified time period. The algorithm divides the order into smaller portions and executes them over time to minimize market impact and slippage.

2.4 Software implementations of related methods

Descriptions of many Python's packages claim to assist in the related problems. The pyfolio package [Gitd] claims to solve the adjacent problem of portfolio optimization and performance evaluation. However, the explicit application of this package to portfolio optimization was not found.

The PyAlgoTrade library, originated in [Com], and accessible in [Gita], is dedicated for developing trading algorithms in Python. It contains implementations of various algorithms for optimal order execution, including VWAP and TWAP.

The Quantopian [Gitc] is a platform for developing and testing trading algorithms in Python. It provides libraries for implementing various optimal order execution algorithms such as VWAP and TWAP. The QuantConnect [Gitb] is a cloud platform dedicated to the same problems.

The paper [Lam20] claims to provide the code example for optimal order execution for stocks and crypto-currency. The author says, the data lies in his S3 bucket, but he does not expose that data. The code also has some problems with dependencies. The code lacks a clear description. Particularly, the point of entry is not specified. The data is expected to be in an unusual format - Apache parquet.

3. Author's two simple models and solutions from basic principles

3.1 Over-simplified mathematical formulation and closed-form solution

Suppose, we stand at an initial time point $t = 0$. Denote T our time horizon, the deadline by which we need to execute the order. The order consists of purchasing the quantity Q of a specific single asset, which can be either positive or negative.

Denote $p_0(t)$, $t \in [0, T]$, the price of our target asset without our actions. The actual price that incorporates our activities is $p(t)$. Their difference is our market impact function $I(t) = p(t) - p_0(t)$. Our activities lie in the purchased quantity function $q(t)$, which is negative when we sell the asset and positive when we buy it.

We need to execute the order, having a minimal total cost C. Then the formalization is

$$\sum_{t=0}^{T} q(t)p(t) \to \min,$$

subject to

$$\sum_{t=0}^{T} q(t) = Q.$$

If we further assume $p_0(t) = p_0$ for all t, the objective simplifies to

$$\sum_{t=0}^{T} q(t)(p_0 + I(t)) = p_0 Q + \sum_{t=0}^{T} q(t)I(t) \to \min.$$

If we specify the impact to linearly depend on the instant purchase quantity

$$I(t) = \alpha q(t),$$

where $\alpha > 0$, the problem reduces to

$$\alpha \sum_{t=0}^{T} q^2(t) \to \min,$$

subject to

$$\sum_{t=0}^{T} q(t) = Q.$$

This problem has an obvious solution

$$q^*(t) \equiv \frac{Q}{T+1}, \quad 0 \leqslant t \leqslant T.$$

for all t. The picture below illustrates it for $T = 1$.

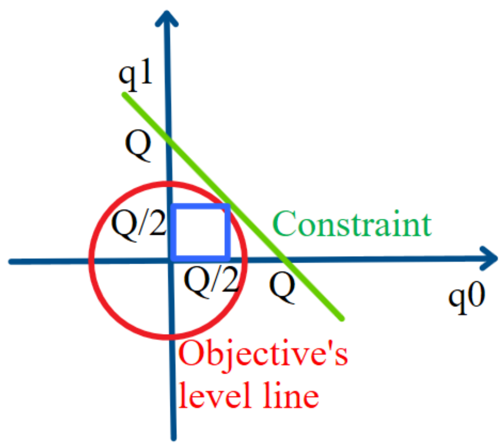

Figure 1: Solution for bivariate optimization (T=1).

At the end of this solution, let us answer the following S&P questions.

- What assumptions are you making and why?
  - A single asset in the portfolio.
  - Market impact linearly depends on the instant quantity traded.
  - An infinite supply of the asset in question for both - long and short operations.
  - No transaction cost.
  - The quantity purchased is the only factor influencing the market impact. In the absence of trade, the price is constant.
- What data would you need (you can assume such data is accessible) and why?
  - Any real cases will show this model's inefficiency. It is just an illustration.
- What modeling techniques would you use and why?
  - A linear model is used for market impact dependence on quantity traded. It was convenient to provide the first rough solution.
- What are some cases where your methodology may fail to give reasonable results and why?
  - A linear model is used for market impact dependence on quantity traded. It was convenient to provide the first rough solution.

3.2 Slightly more complicated model. A solution involving data

Now we assume that

$$I(t) = \alpha_0 q(t) + \alpha_1 q(t-1),$$

where $\alpha > 0$, the problem reduces to

$$\alpha \sum_{t=0}^{T} q(t)(p_0 + \alpha_0 q(t) + \alpha_1 q(t-1)) \to \min,$$

subject to

$$\sum_{t=0}^{T} q(t) = Q.$$

This reduces to

$$p_0 Q + \alpha_0 \sum_{t=0}^{T} q^2(t) + \alpha_1 \sum_{t=1}^{T} q(t)q(t-1) \to \min.$$

It can be solved using [Wik23b] and [Wik23c]. Indeed, the Lagrange function has the following form:

$$L = \alpha_0 \sum_{t=0}^{T} q^2(t) + \alpha_1 \sum_{t=1}^{T} q(t)q(t-1) + \lambda \left(\sum_{t=0}^{T} q(t) - Q \right).$$

Setting its derivatives with respect to all $q(t)$ and λ equal to zero, we get

$$\begin{cases} 2\alpha_0 q(0) + \alpha_1 q(1) = -\lambda, \\ \alpha_1 q(0) + 2\alpha_0 q(1) + \alpha_1 q(2) = -\lambda, \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \alpha_1 q(T-2) + 2\alpha_0 q(T-1) + \alpha_1 q(T) = -\lambda, \\ \alpha_1 q(T-1) + 2\alpha_0 q(T) = -\lambda, \\ q(0) + q(1) + \ldots + q(T) = Q. \end{cases}$$

The first T equations form a linear system with the tridiagonal matrix

$$\begin{pmatrix} 2\alpha_0 & \alpha_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ \alpha_1 & 2\alpha_0 & \alpha_1 & 0 & \dots & 0 & 0 & 0 \\ 0 & \alpha_1 & 2\alpha_0 & \alpha_1 & \dots & 0 & 0 & 0 \\ 0 & 0 & \alpha_1 & 2\alpha_0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 2\alpha_0 & \alpha_1 & 0 \\ 0 & 0 & 0 & 0 & \dots & \alpha_1 & 2\alpha_0 & \alpha_1 \\ 0 & 0 & 0 & 0 & \dots & 0 & \alpha_1 & 2\alpha_0 \end{pmatrix}$$

and the right-hand side $-\lambda(1, 1, \ldots, 1)$. Therefore, this system can be solved (in terms of λ) using the Thomas algorithm. Substituting the obtained values into the equation

$$q(0) + q(1) + \ldots + q(T) = Q,$$

we get an equation with respect to λ. Finding λ, we can then find the optimal $q(0), q(1), \ldots, q(T)$. The allocated time period does not allow us to fully solve it analytically. So we try to solve it by random trial and error.

For start, we can find the coefficients α_0, α_1 by regression of p onto q. The available pq-pairs can be taken from Yahoo finance. For example, if our asset is Apple stock, we can take it from [Fin]. Figure 2 shows both time series, and figure 3 visualizes their relations in scatter plots. As a result of regression

$$p(t) = \alpha_0 q(t) + \alpha_1 q(t-1) + \varepsilon(t),$$

we obtain the estimates $\hat{\alpha_0} = 4.153 \cdot 10^{-8}$ and $\hat{\alpha_1} = 4.779 \cdot 10^{-8}$.

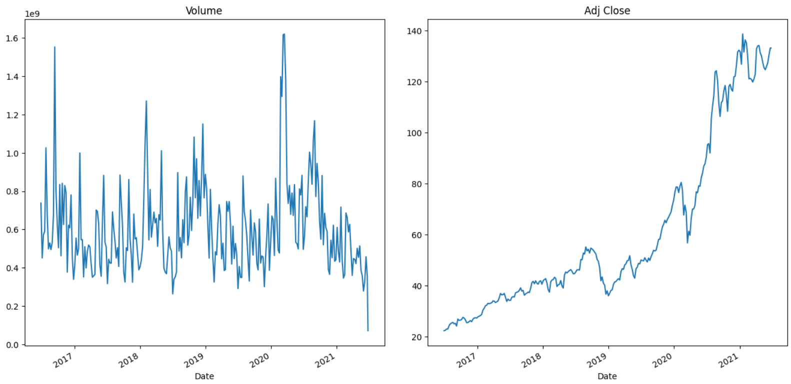

Figure 2: Plot of Apple stock time series. Left - traded volume. Right - price

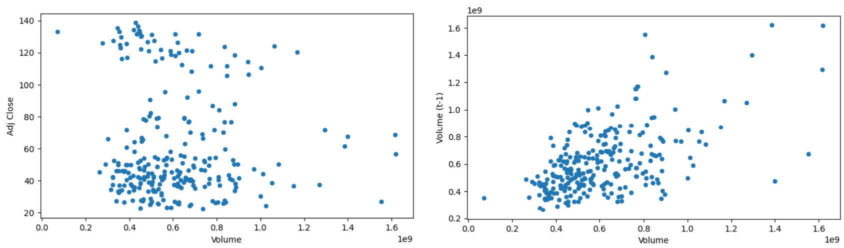

Figure 3: Scatter plots for Apple stock traded volume (q) and price (p). Left - q(t) vs p(t). Right - q(t) vs q(t-1).

To find the optimal solution, we apply the Monte Carlo approach. More specifically, we randomly initialize the variables $q(0), q(1), \ldots, q(T)$ as having the standard normal distribution, normalize them so that they sum to Q, and compute the objective functional; we repeat this procedure $N = 1000$ times. The histogram of the resulting values of the objective functional is plotted below. Since there are many large values that appear very rarely, we also show a histogram with the 100 largest values dropped.

As an illustration, we show the optimal values of q for $T = 3$. First, let us observe the total cost distribution. The left histogram at 4 shows that the largest quantities concentrate around zero, but at the same time, have long right tail. Truncating the lafgest 100 observations, gives the right histogram at 4. The situation does not significantly change for larger values of T.

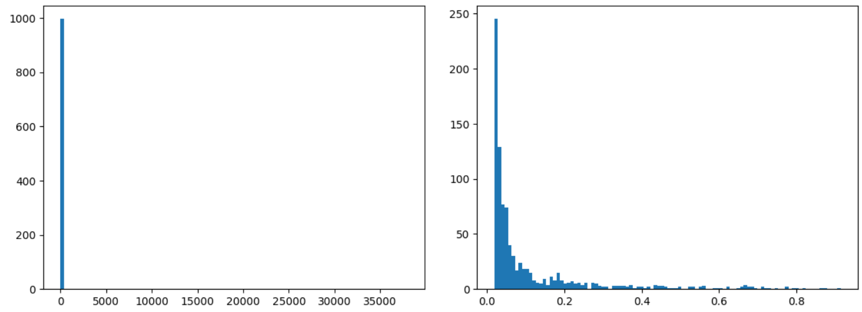

Figure 4: Histograms of solution results. On the left - full histogram; on the right - right-truncated histogram.

Finally, 5 shows the optimal schedule for $T = 3$. The quantities are first large, then decrease and in the end, they rise again. The repetition of this random procedure with different seeds, gives quite a similar picture. So this pattern is persistent for $T = 3$. One of such solutions is [365.01, 119.34 , 198.48, 317.16]. For higher time horizons our ad-hoc method gives alternating patterns. This probably means that we need to take away some degrees of freedom of the solution. For example, we can tie them into a family of fixed degree polynomials.

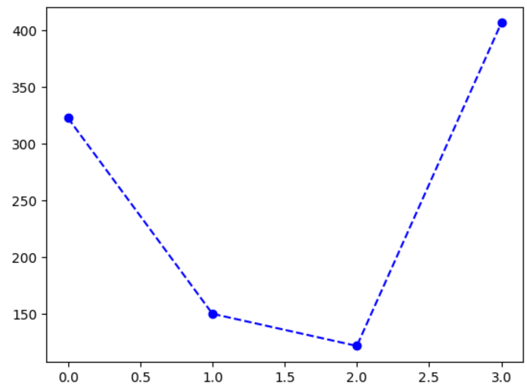

Figure 5: Optimal trade schedule for $T = 3$.

Again, at the end of this solution, let us answer the following S&P questions.

- What assumptions are you making and why?
  - A single asset in the portfolio.
  - Market impact linearly depends on the instant quantity traded and on the previous quantity traded.
  - An infinite supply of the asset in question for both - long and short operations.
  - No transaction cost.

  - The quantity purchased is the only factor influencing the market impact. In the absence of trade, the price is constant.

- What data would you need (you can assume such data is accessible) and why?

  - This publically available data [Fin] is sufficient for the simple model we chose in this subsection.

- What modeling techniques would you use and why?

  - A linear model is used for market impact dependence on quantity traded. It was convenient to provide the first rough solution.

- What are some cases where your methodology may fail to give reasonable results and why?

  - A linear model is used for market impact dependence on quantity traded. It was convenient to provide the first rough solution.

The code behind this solution is posted at public repository [Bor] along with data and this report.

References

[Bor] Boris Garbuzov. Optimal order execution. solution to s&p's test problem. https://github.com/borisgarbuzov/optimal_order_execution. [Online; accessed 30-April-2023].

[Com] Gian Luca Comandini. Pyalgotrade: event-driven algorithmic trading library with focus on backtesting and support for live trading.

[Fin] Yahoo Finance. Apple inc. (aapl;). https://finance.yahoo.com/quote/AAPL%3B. [Online; accessed 30-April-2023].

[Gita] GitHub contributors. Python algorithmic trading library. https://gbeced.github.io/pyalgotrade/. [Online; accessed 30-April-2023].

[Gitb] GitHub contributors. Quantconnect/lean lean algorithmic trading engine by quantconnect. https://github.com/QuantConnect/Lean. [Online; accessed 30-April-2023].

[Gitc] GitHub contributors. Quantopian builds software tools and libraries for quantitative finance. https://github.com/quantopian. [Online; accessed 30-April-2023].

[Gitd] GitHub contributors. quantopian/pyfolio: Portfolio and risk analytics in python. https://github.com/quantopian/pyfolio. [Online; accessed 30-April-2023].

[Kin19] Adam King. Creating a custom openai gym environment for stock trading. https://towardsdatascience.com/creating-a-custom-openai-gym-environment-for-stock-trading-be532be3910e, Oct 2019. [Accessed 28-Apr-2023].

[Lam20] Max Lamberti. Optimal order execution. https://github.com/maxlamberti/optimal-order-execution, 2020. [Accessed 28-Apr-2023].

[qua] High Frequency Trading III: Optimal Execution | QuantStart — quantstart.com. https://www.quantstart.com/articles/high-frequency-trading-iii-optimal-execution/. [Accessed 28-Apr-2023].

[RN01] R.Almgren and N.Chriss. Optimal execution of portfolio transactions. *J. Risk*, page 5–39, 2001.

[Wen18] Lilian Weng. Implementing deep reinforcement learning models with tensorflow + openai gym. https://lilianweng.github.io/posts/2018-05-05-drl-implementation/, May 2018. [Accessed 28-Apr-2023].

[Wik22] Wikipedia contributors. Volume-weighted average price — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Volume-weighted_average_price&oldid=1111172588, 2022. [Online; accessed 30-April-2023].

[Wik23a] Wikipedia contributors. Calculus of variations — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Calculus_of_variations&oldid=1135796495, 2023. [Online; accessed 30-April-2023].

[Wik23b] Wikipedia contributors. Lagrange multiplier — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Lagrange_multiplier&oldid=1150353682, 2023. [Online; accessed 28-April-2023].

[Wik23c] Wikipedia contributors. Tridiagonal matrix algorithm — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Tridiagonal_matrix_algorithm&oldid=1140941727, 2023. [Online; accessed 28-April-2023].