U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

MSc
2.º CICLO
FCUP
2014

U.PORTO

Design and implementation of an algorithm trading system for the Sifox application

Mário André Ferreira Pinto

FC

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Design and implementation of an algorithmic trading system for the Sifox application

Mário André Ferreira Pinto

Dissertação de Mestrado apresentada à
Faculdade de Ciências da Universidade do Porto em
Ciência de Computadores

2014

MSc

Design and implementation of an algorithmic trading system for the Sifox application

Mário André Ferreira Pinto
Mestrado Integrado de Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2014

Orientador
Daniel Lima Eusébio, BSc., Finantech

Coorientador
Inês de Castro Dutra, Prof. Doutor, FCUP

Acknowledgments

I would like tho thank my supervisor Daniel Eusébio and his team at Finantech. The knowledge and experience they shared will accompany me through my professional life, and was an important part of my entry on the financial world. I also would like to thank the support of my supervisor at the Department of Computer Science, Prof. Inês Dutra, for the reassurance, the kind words and the helpful advice in developing this document.

I am grateful to BPI Investment Banking Manager Pedro Costa for his availability and time on providing a insight on the most common trading algorithms and trader priorities.

A special thanks to my parents for all the support and patience. Thank you for providing all the means that allowed me to follow my own path.

I would like to thank my friends from the Faculty of Science of Oporto University, for the important support provided. The gaming nights, the jokes and laughs, helped me to keep going forward. Among them, I would like to emphasize my thanks to Mafalda, Daniel, Cristiano, Ramalho, Pedro, Luís, Carlos and Miguel.

Finally, a special thanks to my lunch buddies, José, Nuno and Carla. Their companionship and advice was very important.

Abstract

The financial markets thrive on information, as its existence makes them increasingly more efficient. As new information technology was developed, the exchanges quickly adopted them, eager to take advantage of the opportunities provided. The new technology allowed for faster communication, greater accessibility to the markets and the new infrastructure helped decreasing the cost of doing business. Today, a trader with a large number of accounts to manage, has his time stretched thin, as the information available for analysis is abundant, if not overwhelming. The new properties of the electronic exchange, mainly on the order book, allows the use of algorithms that can be useful for automating some trading processes. This automation can improve the trader time management and efficiency.

In this thesis we provide algorithmic functionality to the Sifox trading application. We analyzed the requirements and issues of three market-impact algorithms (TWAP, VWAP and POV) and designed a prototype which we integrated in the Sifox application. During the development, for the VWAP algorithm, we tested several models for volume prediction, obtaining best results for a exponentially weighted mean average of eleven days for the Google stock. We further develop our prototype including these findings.

Resumo

Os mercados financeiros prosperam com a existência de informação, ficando mais eficientes. À medida que novas tecnologias de informação foram desenvolvidas, a Bolsa apressou-se a adoptá-las, tendo em conta as novas oportunidades fornecidas. Esta nova tecnologia permitiu uma maior velocidade de comunicação, facilidade de acesso aos mercados e, com a nova infraestrutura, ajudou a reduzir o custo de fazer negócio. Hoje em dia, um *trader*, com um grande número de clientes, tem dificuldade em gerir o tempo disponível, pois a informação necessária para análise é abundante ou até mesmo avassaladora. As novas propriedades do mercado digital, nomeadamente as relativas ao livro de ordens, permitem o uso de algoritmos que são uteis para automatizar certas tarefas no processo de negociação. Esta automatização pode melhorar a capacidade de gestão e a eficiência de um *trader*.

Nesta tese fornecemos à plataforma de negociação Sifox, a capacidade de utilizar algoritmos. Analisámos os requisitos e as dificuldades de três algoritmos (TWAP, VWAP e POV), cuja função é a redução do impacto no mercado , e projectámos um protótipo que integrámos na aplicação Sifox. Durante o desenvolvimento, testámos para o algoritmo VWAP vários modelos de previsão de volume. Os melhores resultados obtidos foram com uma média exponencial pesada de onze dias para as acções da Google. Desenvolvemos o prototipo acrescentando estes resultados.exacarbate

Contents

Abstract ii

Resumo iii

List of Tables vi

List of Figures viii

1 Introduction 1

2 Investment Banking 4

2.1 Equities 4

2.2 Financial markets 5

2.3 Limit Order Book and Order Types 7

3 Algorithmic Trading 10

3.1 Definition 10

3.2 Architecture 10

3.3 Trading Algorithms 12

3.4 TWAP 14

3.5 VWAP 15

3.6 Participation Rate 17

3.7 Order placement strategy and Execution Probability . . . 19
3.8 Evaluation . . . 23

4 Prototype Development **25**
4.1 Architecture Overview . . . 26
4.2 Algorithms . . . 28
4.2.1 TWAP . . . 29
4.2.2 Participation Rate . . . 31
4.2.3 VWAP . . . 32
4.3 Volume Profile Model Construction . . . 32
4.3.1 Data-set . . . 33
4.3.2 Tick Size . . . 34
4.3.3 Evaluation Method . . . 35
4.3.4 Models . . . 35
4.3.5 Results . . . 38
4.3.6 Integration . . . 38

5 Conclusion **42**
5.1 Critical Discussion . . . 42
5.2 Future Work . . . 43

Bibliography **46**

A Appendix **47**
A.1 Acronyms . . . 47
A.2 Alpha calculation for EWMA . . . 48
A.3 Priority Queue Implementation . . . 48
A.4 Predicting the future using Time Series analysis . . . 50

List of Tables

4.1 Sample of volume profile construction from the data obtained via Google API. a) Data as received by the google API. b) Volume agregattion by Date, each tick has a size of 15 minutes, except the first and last ones. First and last tick correspond to close and open bid. c) Conversion to volume percentage. 34

4.2 Chi-Squared evaluation results applied to the three mean methods for volume profile calculation. The best result is presented in bold and bellongs to EWMA with 11 days. 38

A.1 Alpha value calculation. Results of the alpha value calculation process applied to 22 and 11 trading day data. The best values are shown in boldl. a) alpha range between 0 and 1. b) Calculations for $\alpha \pm \Delta$ with alpha being the best obtained value and $\Delta = 0.1$. c) Calculations for $\alpha \pm \Delta$ with $\Delta = 0.01$. 49

List of Figures

1.1 Photography of the New York Stock Exchange in 1959 (left) and 2005 (right). . . . 2

2.1 Structure of the financial markets as defined by [4]. The three main markets are the Money Market (short-term money loans) the Foreign Exchange Market (trading of foreign coin) and the Capital Market (long-term and mid-term trade of financial instruments). . . . 6

3.1 Algorithmic trading system components and their relations. The algorithms described here belong to the Trade Execution component. . . . 11

4.1 Algorithm Module placement alternatives on the Finantech solution. The dotted blue line represents the path taken when the module is in the server side. The green full line represents the module placement on the Client side directly on the SifoxDeal application. . . . 26

4.2 UML diagram of the Algorithm Module implementation in the SifoxDeal application, focusing on the AlgorithmManager and Algorithm classes. . . . 27

4.3 Screenshot displaying the Algorithm Log system. . . . 29

4.4 The GUI of the TWAP algorithm for order placement. . . . 29

4.5 The GUI of the POV algorithm for order placement. . . . 31

4.6 The GUI of the VWAP algorithm for order placement. . . . 32

4.7 Intraday percentage of volume variation and the associated cumulative error from five consecutive days (19 to 25 February, GOOG stock). a) Intraday percentage of volume variation between five consecutive days in different time scales. The first and last tick are respectively the open and close. b) Cumulative error per tick size. The first and last tick are respectively the open and close. The error is calculated as the absolute difference between the points of all five consecutive days. 36

4.8 Description of the volume profile implementation. 1 - At the end of the trading period, intraday data is compiled and retrieved; 2 - The information is fed into the model and the volume profile is created and stored in the database; 3 - When an order is set for execution, the profile is retrieved; 4- Calculations are made to fit the profile in the current order time-span; 39

4.9 Entity-Relationship model for the volume profile database and related components. 40

Chapter 1

Introduction

Men screaming and waving papers in their hands, shouting and receiving orders is the semi-romanticized view of the financial markets for a great number of people. Today, with the constant advances of technology, that image is getting further away from the truth. Although some of those men still persist, much of the trading has moved to the machine's domain. Now in a exchange pit, the number of computers exceed those of humans (Fig. 1.1), and investing in stock is as easy as pointing and clicking with a mouse . But it isn't only on the trading floor that the change had an impact, the market itself has changed. Markets thrive on information. There is a natural need to move goods to profitable markets, but the knowledge on which of those markets might be can lag behind, sending the product to a less profitable one[1]. It is stated by Stigler [28] that the price dispersion between markets is a measure of ignorance, taking this into consideration, it's no wonder that the development of information related technology has contributed for an increase of efficiency and productivity of the financial markets. As a past example of this relationship we have the introduction of the telegraph and the construction of the trans-Atlantic cable, which increased the speed and flow of information, having a huge effect on market price dispersion, narrowing it, making it more efficient [27]. These effects can only be exacerbated with the development of new technologies, driving the markets to a greater state of efficiency. A trader, that has been dependent on his world understanding and market insight to act and react to the exchange moves and whims, sees his efficiency threatened as the data availability and speed of information increases. Now, information that must be analyzed for a successful run at the market is abundant, even overwhelming, as news and market data grow bigger and become more readily available. Furthermore the development of communication technology and electronic access to the markets (DMA) allow for a growing number of investors to participate. These developments can reduce the efficiency of a trader in comparison to one that uses automated or

[1] As stated in the Merchant's Magazine, page 18 [1], "The products of the world, which, if left free to move naturally would be attracted to the most profitable markets, are frequently impelled towards unprofitable ones because the knowledge of which are the best ones becomes known in most cases only after the product has been shipped to an unprofitable one.".

Figure 1.1: Photography of the New York Stock Exchange in 1959 (left) and 2005 (right).

semi-automated methods. The ability to create automation in some stages of the trading process greatly improves the reaction time of a trader, by processing and incorporating the information as it arrives and managing a growing number of accounts. This can increase the trader efficiency and profit margin. In the new electronic market, algorithmic trading is an essential tool for the trader and is widely used. In 2014 first quarter, algorithmic trading was responsible for 17 to 25% of the total market volume per month [25].

Finantech is a software development company producing solutions for the Portuguese financial markets [8]. Its products aim to fulfill the needs of financial institutions in all their market operations. Their main solution, the Sifox platform offers an integrated and wide array of services, ranging from the front and back-office to compliance and risk management services, facilitating the communication, management and execution of market operations. As at the time of this writing, the algorithm component they offer is restricted only to the interface, being the execution outsourced to online brokers. The lack of that particular service has an impact on the company. The use of third party services entails the loss of some control regarding the development process, as any alteration to their interface must be followed and implemented as quickly as possible. But more important than that loss of control is the issue of marketability. One thing is to market the application as having support to algorithmic trading via a third party interface (which is important), other is declaring the product implements algorithmic trading, bringing all the client needs to one place, the Sifox application. It's of strategic importance to Finantech the addition of algorithmic trading to its product portfolio. This addition will add value to its solution and help to further compete on the market.

Our objective in this internship consists in obtaining understanding of how the market operates and to develop a functioning prototype of an algorithmic trading platform into the Sifox solution, taking into account its already created infrastructure. We study the basics of financial market and,

through the analysis of our client needs and most wanted features we implement the most commonly used trading algorithms, allowing some space for customization.

In order to work with a system closely related to the financial market it is very important to understand its inner workings and vocabulary. This knowledge is not only essential for making informed decisions about the algorithms, but also to ease the communication of information in the company environment. In the second chapter we will briefly explain the inner workings of the financial market, getting into detail on information regarding order placement and order book management. In the proceeding chapter we will define algorithmic trading, describing its general architecture, the algorithm theories and evaluation methods. In section 4 we describe the technologies used to implement and integrate the algorithms in the system as well as details regarding the implementation of each different algorithm. We finally present some results for benchmark use for the presented algorithms, concluding with a critical consideration about this report and a description of future work.

Chapter 2

Investment Banking

2.1 Equities

A company that finds itself in need for capital, for whatever reason, has at its service a number of ways to fulfill that need. The money can be raised by going into debt, borrowing it from an entity (usually a bank) or by issuing bonds (debt issued by the company). The contraction of debt is not without cost. In debt financing with the money borrowed comes associated an interest rate that must be paid, raising the total amount of capital due in the short term and conditioning the refinancing of the institution. An alternative to entering in debt is to get funded by selling a part of the fixed assets of the company by issuing stocks, this is referred to as equity financing[3]. An equity (or more commonly known as stock or share) represents ownership and, the number of stocks one has, represents the percentage owned of the enterprise. The bigger the number of socks, more investment and consequently, more stake on the company. The equity term not only comprises the common stock, it can be also used to refer to the possibility of future acquisition of stock, like, for example, the case of options, warrants and stock convertible bonds.

A stockholder that invests his capital in the fixed assets by buying shares is given a limited partnership on the company. The limited participation is important, because it entails limited liability. In a limited liability partnership the shareholder is only responsible for the amount of money he invested. If there is the need for a liquidation because, for example a bankruptcy, the investor does not have to turn in personal property in order to cover the debts. The only capital lost is the investment money, that can be fully or partially regained if, after all the creditors have been paid, some assets remain. To own part of a company also grants some rights. Although the stockholder doesn't have direct control over the company's policies or strategies he can exert control via his voting right at the general meetings. This influence is limited to the election of board members and is restricted by the number of stock possessed. A minority shareholder has voting rights, but his participation is very

small compared to a major one. A shareholder is considered major if he owns more than 50% of the stocks, has more than one representative in the board of directors, or is he himself the director of the company. Other important consequence of owning part of a company through investment are the returns obtained. An investor will usually invest in a company he believes will bring him benefits, by generating future profits. There would be no point in investment and stocks if there were no possible returns on the investment made. A company is expected to provide returns for the investments made and the most direct representation of those returns is the dividends. The dividends are part of the profit attributed to the shareholders in proportion to the capital invested in the institution and its regulation is subject of company policy. A company is not obliged to pay dividends for various reasons, either because they need money for liquidity issues or growth strategies, nevertheless some kind of compensation is expected to be delivered to the investors, as well as an increase in dividends enough to cover the inflation over time. Because of these requirements, the dividends can also be seen as the cost of getting funded in a public offering. Comparatively with debt financing the potential cost in the long-term is greater, because of the increase in dividends amount over time, but this is easily offset by the various advantages provided, like the permanence of the capital, the low-risk of not being exposed to refinancing and the flexibility of deciding when and how to pay dividends. Companies may opt to give compensations to their shareholders in the form of their stock, by influencing its behavior and price. This can be an alternative to distribute dividends. The correct application of the profit money outside of dividends and management decisions can affect stock price and its market behavior, granting shareholders the option of obtaining returns on their investment by selling their shares to others, in the secondary market.

2.2 Financial markets

The financial market is the place where the trading of securities and commodities occur. It comprises a set of different markets, (as described in figure 2.1) each specialized in some instrument and with different defined rules. Generally, when we refer to the exchange or market, we are talking about the section of the capital market that handles equities, where the trading of stocks occur. This market is additionally divided in two sections: (1) the primary and, (2) secondary markets [4] . The primary market is where the issuing company goes to sell itself and get the investment cash, the investors trade money for shares and a promise of future returns[1]. After a stock is issued in the primary market is usually also available in the secondary market, the place where we will operate. In the secondary market shareholders trade with others having no interaction with the issuing company. This market is of extreme importance for the issuing company as the shares represented here reflect the health of the enterprise and its desirability to investors. A liquid secondary market with good conditions will allow investors to cash in their investment instantly and hopefully obtain returns, this adds to

[1] if it's the first initial equity issuing the process is called an IPO, that is, and initial public offering.

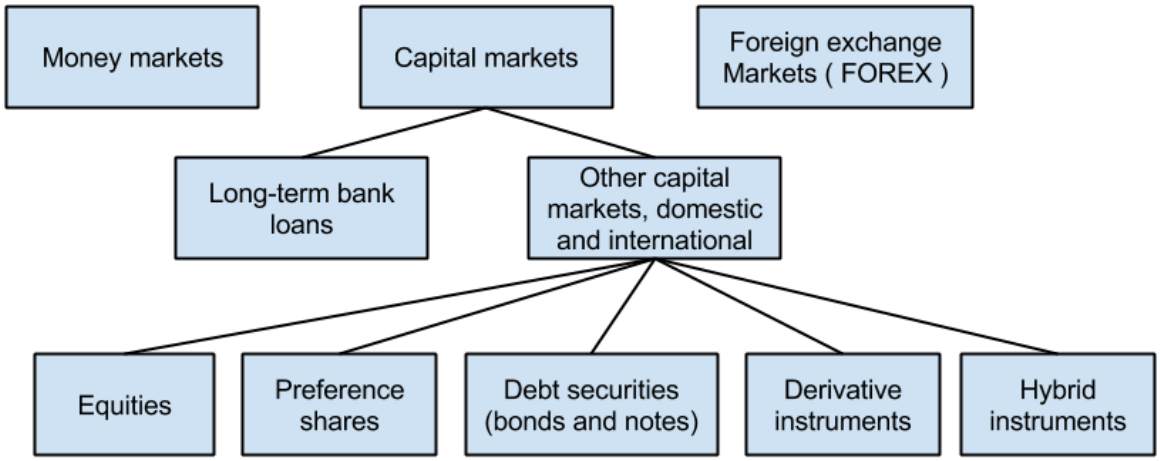


Figure 2.1: Structure of the financial markets as defined by [4]. The three main markets are the Money Market (short-term money loans) the Foreign Exchange Market (trading of foreign coin) and the Capital Market (long-term and mid-term trade of financial instruments).

the value of the investment, as the possibility to obtain great returns with ease will attract more partners. When we imagine the stock market, we are essentially envisioning the secondary market. These markets, present all around the world, are operated, managed and owned by private financial entities whose technology and services provide access and interaction with the exchanges. The most important financial services providers are the ICE group[2], owner of the NYSE EURONEXT[3], and the NASDAQ OMX group[4]. These companies manage the most important markets and provide access to intermediaries whose objective is to facilitate the meeting of buyers and sellers, the brokers.

The advancement of technology has changed the way trades take place in the market and how information is obtained. New technologies were quickly adopted and currently the most important exchanges operate almost entirely electronically, allowing us to access the market and place orders via online platforms[6]. The trades used to be processed by personnel on the trading floor, using the public outcry method, have moved to data centers, and most of the trades are now electronically matched[5]. It is important to understand the trading process and how the orders are matched in order to have insight on which strategies will work and their consequences.

[2]https://www.theice.com
[3]https://nyse.nyx.com
[4]http://www.nasdaqomx.com/
[5]NYSE EURONEXT offers a Hybrid system for equities allowing public outcry, while NASDAQ is electronic only.

2.3 Limit Order Book and Order Types

Usually, in secondary markets, trades are based on an auctioning system and at its heart there is a limit order book, used to keep record of unfulfilled trades [12]. When the orders first arrive in the exchange, they are processed in a first-arrived-first-served (FIFO) fashion. Trades whose price immediately matches one existent on the book are processed, and the transaction information is disseminated through the exchange, while those with no matching price stay on the book. The unexecuted orders that constitute the limit order book will be either buy orders or sell orders. A buy order, or commonly referred to as a bid, consist of a price and a quantity one is willing to pay for shares of a certain company. A sell request, or ask order, is similar to a buy order, but represents the desire to offload stock. These orders exist in the book ordered by a specific set of rules, that will dictate the priority of transaction for these orders. At the moment of the arrival, the orders will be sorted by arrival price (price priority). On the top of the book there will be the orders with the best price. The best price is relative to the order type (buy or sell) and means the most competitive. A most competitive bid (buy request) is one with the highest price, a best ask is one with the lowest sell price. After price priority they will be ordered by arrival time. Having not been matched the two different request orders are separated by a gap, the difference between the best Ask and Bid, called the spread. In order for a transaction to happen, the spread must be beaten (the gap crossed) by the side who wants to make the trade. The orders contained in the book are not static and can be subject to constant and quick change. The owner of a order is able to cancel or edit it, but doing so will incur on priority penalty, as in practice the change is nothing more as the canceling and submitting of a new order. Although market orders are essentially restricted to those two sides (ask and bid) , the market execution conditions and validity parameters can be tweaked granting them different functionalities. The use of different order types depends heavily on the broker being used and their support on the exchange[6]. In order to have a good execution and achieve the desired result, it is essential for anyone that operates on the markets to have knowledge of the methods that will allow them to execute the chosen strategy.

The execution conditions available on most markets are the following:

- Limit Orders - An order with an imposed price limit. The order will stay visible on the limit order book until it is executed at the price limit or better. This type of order guarantees the price but does not guarantee execution, because the market can move away from the targeted price and never return.
- Market Orders - An order without price limit targeted at the best Bid/Ask price of the moment. This type of order suffers from the possibility of slippage, that is, the price targeted might not be the price sold. If our order is slower than another, we will possibly miss the Bid/Ask price

[6] A difference can be seen between orders supported by the same group (ICE) in different equities markets, like Euronext [5] and NYSE [18].

and get the order matched on the next available price. This slip of the target price can make one incur in great losses. As such, a Market Order does not guarantee the price and in high liquidity markets always guarantees fulfillment.

- Stop Orders - A Buy/Sell Market Order, triggered by a security reaching a certain price. When the selected trading equity reaches a prearranged price hit, a market order is sent. Usually this type of order is used by traders as a best practice, to stop losses by selling when a price drops too low[7]. In the first phase execution is not guaranteed, but when price triggers, it behaves like a Market Order with all its properties.

There is also the possibility for the creation of execution conditions that are variations of the above, or even the same, but because of their objective, they can get different denominations (like the Stop-Loss Orders). One that is quite used is the Limited Market Order (plus or minus one) this order is nothing more than a Limit Order with the price set to the current best bid/ask, having the possibility of being one tick above or bellow.

Other property of the order relative to the market is the validity. As the name suggests, validity is the span of time the order is considered as valid and is available for execution. The order validity which the Sifox application supports are the following[8]:

- Day Validity (DAY) - Order remains in the book until the end of the trading day.
- Fill and Kill / Immediate Or Cancel validity (FAK / IOC) - An order is sent, gets fulfilled as much as it can, then it is canceled.
- Fill or Kill (FOK) - The order is sent to the market to be fulfilled, if it cannot be fulfilled in its entirety, the order is canceled.
- Good Till Canceled (GTC) - Order stays in the book until fulfillment or a cancel order is issued (in EURONEXT up to a period of one year is supported).

The validity is combined with the order type, a side (buy/sell), a price (or no price, in case of a market order), a quantity and is sent to a specific market for matching and execution.

The execution and validity properties and how the book is organized, influence our order fulfillment and its execution costs, it will decide if our order will be executed, walk the book or simply stay visible

[7] It is interesting to notice that while using a stop order to prevent losses (stop-loss) is considered to be a best trading practice (simply by obeying the rule of trading with limits), it can be responsible for major capital loss. A sudden fast market variation or a minor crash, can activate the price limit, and because the orders are placed autocratically and of the Market Orders type, the potential for slippage will be huge. Furthermore the Stop-Loss can exacerbate the impact of those fall even further, by causing a chain reaction and lower the price even more[24].

[8] Some of these validity types are defined by EURONEXT at *https://europeanequities.nyx.com/pt-pt/trading/order-validity* using other denominations, for consistency with the system we will be using the Sifox Deal application nomenclature but will indicate the Euronext name.

to all. The limit order book and its functioning is of pivotal importance to several strategies, its inner working is incorporated in most of the order placement strategies.

Chapter 3

Algorithmic Trading

3.1 Definition

The term is difficult to define, due to its confusion or interchangeable use with several other related terms. Generally, and the definition we adopt, algorithmic trading is the use of a computer program to create automation in one, or several, steps of the trading process. The definition given is very broad and usually the algorithms can be sub-categorized, depending on the technology used, objective or where in the traders pipeline the automation occurs. On the general definition of algorithmic trading we include the following sub-categories:

- Systematic Trading (ST) - Systems that use trading strategies based on a set of pre-defined rules received as Human input.
- High-frequency Trading (HTF) - The term is usually used in systems characterized with fast execution speed (in the order of milliseconds) and holding stock for a short time-span. Usually an HFT algorithm proceeds without the intervention of a Human.

It is important to note that the use of the Direct Market Access (DMA) or other online platforms, supplied by a lot of brokers, is not considered Algorithmic Trading. DMA is only a platform for electronic trading, allowing a more direct access to the markets, every decision and trade operation still has to be made by the trader.

3.2 Architecture

An algorithmic trading system can have several well defined components, each in charge of a specific aspect of the trading process [22]. The components are the data, pre-trade analysis, trading signal

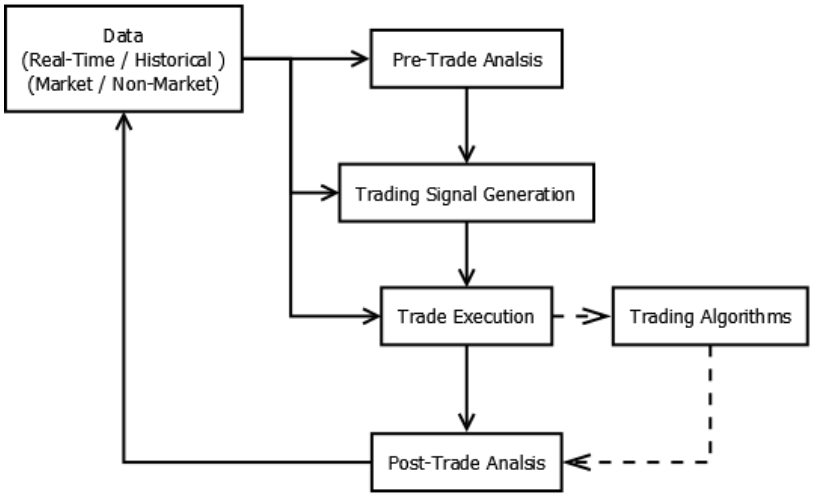


Figure 3.1: Algorithmic trading system components and their relations. The algorithms described here belong to the Trade Execution component.

generation, trade execution and post-trade analysis (Figure 3.1).

The Data component interacts with all the other components and is the basis of the system. Its function is to gather and clean information retrieved from market or non-market sources. This is an important component, not only because it is used by every section of the system, but also because of the relationship between the performance of the system and the quality of the data. Due to the importance and dependency of the system on this component, all data obtained must be cleaned in order to treat wrong, incomplete and meaningless data that, if left untouched, can create noise capable of impairing the strategy used. In our algorithms the most common data source will be the market data, current and historical. However, there are systems that can use other non-related market data in order to produce market evaluations, like news, tweets and data obtained by mining social media sites.[1]

In pre-trade analysis, financial and derivative data (like news, or social media) are used in order to predict trends. These predictions will inform us what type of portfolio to build in order to trade and on which side of the trade we should be (buy or sell). Some of the techniques used to get a market prediction include fundamental analysis, technical analysis and quantitative analysis. Our system does not use pre-trade analysis due to its function being mainly execution support. The decision of what stock to trade and which algorithm to use comes from the trader and has no input from the system we are building.

[1]Usually the type of data obtained via social media can be used as a trend indicator [23]. The use of this information raises growing concerns, as hacked accounts can disseminate false news with serious market repercutions. The blame usualy falls on the HFT that with its full automation and speed, lacking human constant supervision, judges bad data and acts accordingly [16].

The trading signal generation component will place a value and a quantity on our chosen portfolio. This component usually uses information obtained in the pre-trade and apply other technical analysis techniques using relevant market data in order to obtain the expected goal, be it maximizing profits, minimizing trade costs or even developing an entry and exit strategy, as well as risk management. This component can be easily confused with the pre-trade analysis and the trade execution. Its main difference from the pre-trade analysis is that the decisions made can evolve to possible trades, it gives more concrete information that can be part of a strategy, and not just a recommendation on the selection of the portfolio. It diverges from trade execution by not actually executing or giving execution details for the trade, like a schedule.

The Trade execution layer will make decisions regarding the actual placement of orders in the market, that is, its execution strategy. Having received a quantity and a time frame, the component must read market conditions and create orders that fulfill a stated objective. Even if that requires dividing the orders, will never go against the previous component decisions. Trading venues, schedules, order types and order quantities are several options this component must decide.

Finally, in post-trade analysis, new information, obtained from the algorithm interaction with the market, is incorporated as new data in the application. This feedback will allow for strategies corrections and application control.

Giving the proposed architecture, and per our definition of a trading algorithm, the system must have at least two components, being the Data one of them. The algorithms we implement belong to the trade execution component, their main objective is to guarantee an efficient placement of the order. Our Data component is obtained through the Sifox application and data retrieved from specialized finance sites, like Google and Yahoo finance, in order to obtain historical values. We do not offer decision support tools, all the other components are the responsibility of the trader. It is he/she that will decide on what stock to trade, the venue, the signal, the quantity and the possible price limits to be used in the trading day.

3.3 Trading Algorithms

Currently all the major companies in investment banking offer algorithmic services packages to their clients. These packages usually consist of trade execution algorithms, that aim to facilitate trading and minimize trading costs for the client. The selection of the correct algorithm to fit a certain objective and market conditions is of extreme importance for a trader. The algorithms can be categorized according to their function[13] in the following manner:

- Impact-Driven - Minimize the impact of the trade on the market. Orders with big quantities can have a negative impact on the market, as its required quantity can be larger than the

number of orders offered at a certain price level, causing the order to "walk on book" and be fulfilled at increasingly worst market prices. In order to obtain a better trade for big blocks, this must be taken into account, and dispersion of quantity along the trading day should occur.

- Cost-Driven - The objective of cost driven algorithms is to reduce the general cost of trading. They must take into consideration all factors that can influence the cost. Similar to the impact-driven algorithms, they must take into consideration market impact, but not to the point of avoiding it in its totality. If an impact to the market must be made to achieve a cost effective trade, they will proceed to act on it.
- Opportunistic - These algorithms want to obtain liquidity and trade in the best possible way, finding opportunities and taking them, disregarding market impact and trading cost to achieve their goal.

The characterization of the algorithms by their functions seems closely related to the mechanics they use to achieve their objectives. The following categories, proposed by Yang et al. [30], further divide the algorithms, helping to the understanding of their purpose based on their internal mechanics:

- Schedule-Driven - These algorithms follow a well known schedule, having no or very little variation. The use of a schedule allows for a predictable division of order quantity to be fulfilled in small intervals (from one hour to 5 minutes). In general, thanks to their schedule, these algorithms are the most passive and therefore are used to reduce market impact of large orders. Although they are regarded as passive algorithms, the sub-orders themselves can vary in aggressiveness. Depending on the trader objective and the risk one is willing to take, the actual placement of the orders can be different. For example, using market orders for aggressive strategies and limit orders when willing to take the risk. This mechanic is one that relates to the objective of impact-driven algorithms.
- Evaluative - A middle ground between schedule-driven and opportunistic. The Evaluative algorithms combine the two approaches. They try to obtain the best possible trade while reducing as much as possible the impact they have on the market. They can use historical data and quantitative analysis to figure out a schedule and incorporate real-time data in order to obtain the best cost. Balancing these two factors they try to achieve the best cost, the same objective as a cost-driven algorithm.
- Opportunistic - Dynamic aggressive algorithms that constantly react to market change trying to obtain the upper hand. These are usually the liquidity seeking algorithms, they take advantage of the market evolutions at real time. It has the same name as the previous categorization.

In a meeting we attended with Pedro Costa, BPI Investment Banking Manager, it came to our understanding that only a small set of algorithms is procured by their clients. The most commonly

used algorithms are the impact-driven ones. We believe that this preference happens not only because they are specifically requested by clients who do not have sufficient knowledge to use more case specific algorithms, but also because they allow a trader to better manage orders with big quantities. This allows the traders to focus on orders that they deem more important, automating those less critical that otherwise would require constant monitoring. The most common used algorithms, and the one we will be focusing are:

- Participation Rate / POV (Participation Over Volume)
- TWAP (Time-Weighted Average Price)
- VWAP (Volume-Weighted Average Price)

In order to obtain information of how the algorithms perform, and what parameters they take, we read several documents of algorithm implementation of the competition, available online. This information is, as expected, very superficial. The publications must disclose the strategy for the interest of the clients but at the same time, their inner workings have to be kept secret because of the competition. Still, these documents provided much needed insight in the principle of the algorithms, pointing out features that they should have and serving as a guideline to the trader concerns and what he/she was expecting.

We will now describe the algorithms one by one. Unless otherwise noted, all examples will be made as if we were selling securities. Usually buying and selling are symmetric when taking into account the algorithm strategy, they are not symmetric however in terms of order placement. In order placement the conditions of the market vary according to the side (buy or sell) we are on, so predictions and analysis believing the sides possess symmetry, can be wrong.

3.4 TWAP

TWAP is an impact-driven algorithm, whose purpose is to trade a block of stock over a period of time, causing the minimal amount of impact on the market [13]. To achieve this, the algorithm simply fractions the quantity of stock available in equal parts over a period of time, hoping to achieve a value as close as possible to the twap. The twap, as calculated using the formula 3.1, is the benchmark used to evaluate the quality of the algorithm execution. The objective is to have a twap value of our executed trades close to the market value.

$$TWAP = \frac{\sum Price}{Number\,of\,Trades} \tag{3.1}$$

The simple nature of being a schedule based algorithm brings problems that can degrade its

execution, depending of market conditions. It can suffer from serious signaling risk, by giving information of its operation. If we place orders of the same quantity at the same time interval we can fall prey of competitors that might be looking for patterns indicating the usage of this type of algorithms. This is a dangerous behavior, as an observer with that information can cause an unfavorable market movement, given that he now has information regarding our intentions and that gives him the upper hand in negotiating. Because of this risk there exists the need to mask our orders by not sending them at regular time intervals and with the same quantity. These alterations must not change in any way the algorithm behavior or disrupt its purpose. The actual order placement is also an important factor that must be taken into consideration. Just sending the order at market or limited price might not achieve the best price or targeted execution, by selecting a favorable order execution strategy we can improve the results and overall algorithm performance.

The schedule is generated by dispersing a given quantity per time, that does not however guarantee us the total execution of the order. The guarantee of execution is dependent on the strategy used. There is also the possibility of using order execution parameters that will interfere with the certainty of execution. If the parent order has a limiting price, the algorithm cannot override it, causing the non fulfillment of orders if the market trends away from us. We could use some kind of catch up logic, where we would accumulate the unexecuted volume and dispatch it at our earliest convenience in the next scheduled time. Using these kinds of strategies would go against the main theory behind the algorithm, as market impact would rise with increased quantity to be traded, causing a poor performance.

The parameters the algorithm needs are the quantity to trade and the price limit. The time-span of execution and other implementation parameters are optional.

3.5 VWAP

The VWAP algorithm, like TWAP, is also an impact-driven algorithm, thus making use of a schedule. Its use is to automate the trading of blocks equities in a passive manner, in order to have the minimum possible impact on the market [13]. In order to achieve the result, it will slice the order in quantities that are proportionally similar to the volume being made by the market. At the time-span defined by the algorithm, if the current stock volume being traded correspond to a percentage of the final total market volume, we want to mimic that percentage taking into account our own quantity. This algorithm strategy, as indicated by its name, is based on the vwap benchmark. This benchmark is widely used due to the fact of being considered unbiased in relation to other indicators that employ non-representative points (for example the twap). The VWAP achieves this by taking into consideration the weight a transaction has. A trade with a higher volume has more impact on the market volume, and therefore contributed more. The use of transaction weigh mitigates the effect

that some prices, like closing time prices, have on the evaluation of the true trade value [2]. The formula is given by summing the multiplication of the traded shares volume (V) at their respective price (P), over a particular time (j) and then divide them by the total number of traded shares ($\sum_j V_j$) to get the weighted mean (formula 3.2).

$$VWAP = \frac{\sum_j (V_j \times P_j)}{\sum_j V_j} \tag{3.2}$$

Any interval of time can be taken into account, but usually it is calculated from start to finish of the trading day. If we consider the VWAP the closest value to the fair price of a share, we can define the market impact cost of a transaction by comparing it to the trade we made. A trader top priority, while using this strategy, is to minimize the market impact so he will be considered successful if the trades he participates on that day give a market impact of exactly zero.[2]

$$Market\,Impact\,Cost = Volume\,Weighted\,Average\,Price - Buy\,Price \tag{3.3}$$

$$Market\,Impact\,Cost = -Volume\,Weighted\,Average\,Price + Sell\,Price \tag{3.4}$$

Looking at the formula previously described (3.3 and 3.4) in order to achieve the desired cost of zero market impact, in the best case, our prices must be equal to the VWAP or at least as close as possible. To get the closest possible value we would need to participate in each available trade exactly in proportion of our order relative to the final market value and at the same price value. This is not possible, because we do not know what is our order size relative to the final volume but, more importantly, we cannot know when will trades happen and their characteristics. Instead of this approach, it is shown by Kissel et al. [14], that the best strategy is, instead of considering all trades, to divide the trade period in slices and use the volume and average execution price of that period (formula 3.5).

$$VWAP_j = \sum V_j \bar{P}_j \tag{3.5}$$

Assuming we can obtain the same average price of each slice, to approximate the value is to find the slices that minimize the difference between the VWAP benchmark and our average execution price. The solution is to participate in proportion with the orders in each slice. In order to do that we must find beforehand which percentage of the final volume each slice will have so we can divide the

[2] If a trader is truly concerned about market impact he will want the trades to be inline with the vwap benchmark. In practice hovever that might not be true, as a positive market impact means also that the trades where better than average.

order quantity and proceed with the schedule.

As expected, the final volume is only known at the end of the trading period, which is a problem. By the time we know for certain the value of the final traded volume, all trades in which we wanted to participate will be over and we have missed our opportunity. We can try to predict the final volume of that day, but by doing so, we would incur on errors. The prediction of just one value, that is quite volatile among trading days, would not give us any reliable information on how the trade volume progresses along the day. Instead we will believe that the volume follows a pattern in relation to its final volume and its variation is of no consequence. For example, it is known that the volume of certain equities is not random but follows a pattern achieving peaks at the open and close time. There is also the possibility that, in some markets, the volume at the time of the opening hours of other exchanges can induce on volume variation. By observing this effects using historical intraday data we can create a volume profile of the stock which the algorithm will use to create the schedule.

3.6 Participation Rate

Also known as Participation Over Volume (POV), the participation algorithm aims to fulfill a percentage of the market traded volume [13]. POV is an impact-driven algorithm, its purpose is to reduce market impact by trading inline with the volume. The algorithm monitors the quantity of all the trades executed, since the order submission start date, and calculates how many stock it has to trade in order to keep close to a chosen target percentage of that volume. This may categorize the algorithm as an evaluative one, taking into consideration that it reacts to the market, but it is really schedule-driven, the schedule is dynamic and only known at the time a trade is executed but is predictable. In its simpler form and assuming that the market is liquid enough to support it, this algorithm guarantees complete execution. In practice though that will not be the case, thanks to the restrictions we must enforce to ensure a safe execution.

In order to participate a volume according to the target value, we cannot just place an order containing the percentage needed for volume completion. If we reacted to a trade that just occurred with an order quantity of exactly the needed to obtain the participation target number we would fail, because in the next instant the total volume has changed by an amount equal to our order placement. To avoid that situation, we need to offset our quantity placed according to the impact we will have on the market. In case we fail to do so, the algorithm will be always a step behind, and a cycle can form, never achieving stabilization. We must participate in the desired quantity as defined by formula 3.6.

$$Quantity = \frac{Volume * Participation\,Target}{1 - Participation\,Target} \tag{3.6}$$

As with TWAP the POV algorithm is subjected to signaling risk, having the risk a particularly

stronger impact [17] causing the algorithm to be easily manipulated. Every time a trade occurs if the algorithm rushes to place an order a pattern can arise. A competitor could easily use this information, and our performance could drop. Other issue that could hinder our performance is the competition between algorithms of the same type. If we had two or more of the same algorithms running, they would compete with eachother, tying to fill their order volume as fast as possible. In this case once one completes a trade, the other has to issue more orders, because the market volume has changed. If the order book has enough orders and the order prices are not limited, the performance of both algorithms would decrease quickly. To prevent these actions we must regulate the order placement, trading more periodically with the volume by adding a deviation to the target percentage. By allowing the algorithm to incur in taking the percentage with error, we can either get ahead or behind the target value, and plan the placement of our order. This would also allow an increase in options for our algorithm by creating a parameter regarding the aggressiveness of execution, that would define how closely we would track the target values. Having this margin for execution we could even propose the placement of orders outside of the schedule in order to do opportunistic trades. Of course if the market is liquid enough this behavior might not be necessary, we will not have problems in signaling a Human competitor if a huge amount of different trades are happening in the market, but still, we can fall prey to proprietary algorithms.

Apart from measures against the signaling risk we will also need safeguards to guarantee the performance does not degrade to an unsuportable state, incurring in possible big losses. Volume spikes can degrade our execution and should be taken into account. A large spike will degrade the prices on the order book by taking away liquidity and may force the drop of orders on the book to an unfavorable state. Tracking the volume closely would exacerbate the problem, because after the spike we would go to the market and force execution, getting the worst deal. This can be solved by not tracking closely the value, ignoring the volume spike or by placing a limit price on our orders. With this measure we would gain better execution but at the expense of order completion and depending on the trader objective, this could be desirable or not.

The price limit is a parameter of the algorithm and can be used to counter volume spikes, by limiting our entrance in bad trades, but can also create those spikes and force a possible non-completion of the algorithm. If we place a price limit and the security market value drops under it, the algorithm cannot place orders that go against that value. If we continue monitoring the volume while the price is against the limit, after the recovery the algorithm will find itself with a lot of ground to cover and will be overly aggressive in order to maintain the target value. This spike can be prevented if the trades made bellow the limit value are ignored, not being counted for the final total volume. An alternative is to use future prediction and start compensating pre trade, fulfilling more quantity than that needed to achieve the target percentage, when we know the price will go under our limit value, or even use a smoother catch up logic. The catch up logic has the same issues the TWAP had regarding market impact, but we can support it if market conditions regarding liquidity allows for an

increase of small volume traded without severe market impact.

3.7 Order placement strategy and Execution Probability

An aspect that must be taken into account by all algorithms is the placement strategy. The schedule will dictate how the algorithm should divide its orders to achieve its objective, the placement strategy is responsible for the actual placement of the order on the market. It assigns a price and decides which order type it should have. This strategy is of paramount importance to our module because it will have direct influence on how well the algorithm performs on the short term, aiming to minimize the transaction cost. The cost we are willing to pay, the transaction cost, is directly related to our ability to execute our order in the required time-frame. When we want a guaranteed transaction, we are sure that will be completed in an arbitrary time-frame, we have to pay an increasing price. By putting our order with a more desirable price we will incur in a loss, but because of the properties of the limit order book, our order will be closer to the top of the queue (if it is a sell order, lowering the price will make it more desirable for the buy). If the market trends in the opposite direction we will have to change our order so it can follow it. By doing so we are trading the execution for profit, as our order will have to be cancelled and submitted at a new worse price. By going long (holding our stock and placing it at a more favorable price) we are trading guarantee of execution for price. This is exacerbated if the time we have to dispatch the order is limited and small. If we have some information that will benefit us, or a tight schedule to follow, being able to comply with the time constraint is important, we will not only want to guarantee execution but also do it in the allotted time period. This reduces our option for waiting for a better situation, meaning we will have to manage the execution risk that can impact the execution cost. A more desirable order for the market, one that has the most competitive price, will have priority on the book and be more quickly executed. In an unpredictable market, changing the position on the order book might not be enough, the price can trend in an opposite direction, leaving us behind. We may add to the cost in order to guarantee immediate execution. The extra cost we must pay is the spread. By paying to go over the spread we will directly consume an order at the current worst possible price, because we had to actively press forward and make the transaction. We can pay this cost by submitting a market order. This cost however can be bigger if we allow for slippage to occur. If our order arrives later than other concurrent orders, we might fail the price we thought it would be achieved at the moment of submission, and instead get a worse (or better) one. In short, its a dilemma, a trader must juggle the execution timing and price, paying more for fulfillment or waiting and possibly achieve a better price.

The algorithms will have to manage these options at the order placement moment and they will base their decision depending on the objective (guaranteed execution or best price) defined by the parameters submitted by the user. Please note that order placement strategy is not to be mistaken for the algorithm strategy. The strategy of the algorithms answers the questions before the

order placement "How much quantity should my orders have?" and "What time should I place it?". Questions such as "How much should the order be priced, in order to guarantee the best relation between price and execution time?" is part of the order placement strategy. The placement strategy can be based on the notion of probability of execution.

The Execution Probability indicates the likelihood of an order placed at time t with price p be executed in the $t + 1$ instant. The analysis of the trader objective is an important consideration to have, as it influences the strategy to be used. Harris [11] defines three stylized trading problems, focusing on small volumes (due to market impact problems), that represent the diferent trader objectives. The small volume restriction is on importance to us, as the algorithm scheduling handles that particular problem by negating the quantity impact on the market. We can consider that order submissions do not affect the market. The three trading problems are the following:

- Liquidity Traders (LT) - These traders need to fulfill an order before a deadline is reached. They will try to obtain the best possible price but their priority is guarantee the order is executed. Usually they start working the market in a passive fashion, but as the deadline approaches they take a more aggressive stance.
- Informed Trader (IT) - An Informed Trader has private information about price changes and believe that that information will soon be available to everyone. They have time constraints, as they need to react quickly, in order to profit from their information. Unlike the Liquidity Traders they are not obliged to trade before the deadline and will only trade if it is profitable.
- Value-Motivated Traders (VT) - A Value-Motivated Trader is always receiving private information of security values. Usually they are continually estimating the value of the securities they follow, and need immediate execution if they believe an abrupt price change will happen, otherwise they will place limit orders and try profiting from pricing errors. Unlike Informed Traders, they have continuous price information and are not constrained by a deadline.

The order placement strategy that will be used on the algorithms have all the qualities to be related to the first type, it is a problem regarding the Liquidity Traders. After having a schedule, our primary objective is to dispatch the orders with the best available price in the most swift manner, until the given due date, as a liquidity trader would require. Furthermore, the algorithm is somewhat obliged to trade, preventing it to belong to the other types, due to the promise of execution that it offers to the trader. If a trade is not made, the quantity of unfulfilled orders can accumulate, damaging the defined schedule and the intended objective of reduced market impact.

In the literature there are several strategies that use some specific market indicators (such as volume, book imbalance, book depth) in order to get insight on order placement. Yingsaeree [31] further divides these strategies in static or dynamic types. The Dynamic type possesses the ability of monitoring the placed orders and changes them as it pleases (taking into account the properties

of the limit order book). The Static types only allow one shot orders, having no ability to change them. Simply because our order book can use dynamic order strategies, this does not mean that the static types should be neglected as information contained in them can be useful, furthermore, the implementation of dynamic orders may prove a challenge taking into account the infrastructure of the SifoxDeal application and the available data. The small time-span for order execution, obtained from the schedule of the algorithms may also cause the unavailability of dynamic order. Dynamically altering orders requires time, for the editing of orders, communication and analysis of factors, time that in a tight schedule will not be available. Also, once the order changes, it goes back to the bottom of the queue for the same price range further limiting the execution time. As such the static option can be the only option for some order placement.

Handa and Schwartz [10] investigate the use of market and limit orders as a strategy. They conclude that using limit orders is detrimental if the market only reacts to information and there is no market imbalance. An uninformed trader will loose by placing limit orders (when competing with informed traders). However, if there is enough market orders causing market imbalance, moving the price to reach the limit order value, there will be a gain. In case of order non-execution the following two options are given: not execute or execute at the end of trading period at closing price. The second option will result in an additional cost, the cost of exchanging a limit order into a market order. They conclude that patient traders, those that are willing to wait are the ones that will benefit the most from the placement of limit orders and that there is not a big turnover difference between using market or limit orders. We can use this information by only placing limit orders when the market has enough market orders, reflected on volatility and number of trades, to catch our limit orders in our allotted time interval. It is also of our interest as we predicted, to not execute the orders at the end of the closing period to force their execution, for the non-execution cost produces worst results than the limit orders. This however indicates that simply using a market order could be enough. From our observations this would not be the case or sufficient. For example in the VWAP algorithm we must guarantee the mean price, or better, in order for the execution to be more successful. By using more information we might be able to hit a sweet spot with the limit order, achieving the best cost/execution tradeoff.

Nevnvaka et al. [19] combine market orders with limit orders to obtain a better execution, somewhat contradicting Handa and Schwartz [10] findings. They propose using a risk-return curve based on historical data capable of finding the efficient pricing frontier. The return graph will indicate how far to place an order in order to achieve the least execution price. The risk profile (being the risk of non-execution, mid-spread volatility and volume volatility combined by defined it as the standard deviation of returns) , indicates the risk of non-execution, the more we hide on the book the higher the risk. Combining these two profiles we get a curve that relates risk and the return. With it we can choose a strategy by picking price improvement or risk. In this case the risk (non-execution) is a more important factor due to the type of problem we are trying to solve (Liquidity Trader). Some

considerations need to be taken into account regarding some variables in this process, that can be controlled in order to achieve better performance of this model. It is indicated that special care must be given to Order Size, Time of the Day, Time Window and Market Conditions. Big order sizes force the orders to be more aggressive in order to complete. This will not be an issue with the algorithms since the schedule they do is exactly for the restriction of big orders. The time window available for order completion changes considerably this approach [19]. A small time window means less time for the market to hit our limit order, while a large one gives more time for that to happen. Taking into account market properties and with a small time window we might be more successful if the order is placed directly on the market. Time of day can influence the order speed due to the variations of volume and should be considered if the trading period is long. The trading volume, and volatility are the most important factors of the market conditions, and our model data should be constructed with these values in mind. Concluding the article the authors point out that this approach can be a generalization and the optimal strategy can change over time, where we must reevaluate the pricing strategy. It is proposed the creation of a function that using the described elements will produce our limit distance. We can use the ideas on this article to create a risk-return and use this to express the probability of execution, attributing an option to our placement strategy that will enable the trader to change the risk parameters for the placement strategy.

There is the idea that if the book permits order adjustment, its use will make for better results. By altering a limit order we can place it in a more favorable position, hence beating the price of a static limit order. Furthermore, in the current electronic market, there is the potential of more information being available that can allow for a better execution [20]. Before the only available information was the best bid / ask, and the price of the executed trades, now with eletronic trading, we have access to more details of the book, capable of seeing the outstanding limit orders and their parameters, in any depth. The incorporation of these new variables allied with dynamic order adjustment can improve performance on the algorithms. Wang and Zang [29] propose a dynamic focus strategy (DF), adjusting the volume of market orders by monitoring one of two parameters, the inventory unexecuted volume or the order book imbalance. For quantitative analysis they propose a formula based on a sigmoid function that reacts to those parameters taking into account the remaining time. They claim that the method can also be applied to limit orders in order to achieve better performance. Taking into account that order book imbalance offers very little performance improvement [20] and is seen as a future trading cost predictor, the inventory DF strategy is best suited for the algorithms described.

In our work we incorporated some of these ideas when developing the order placement strategy that will be common to all algorithms. In our opinion, the order placement is the most relevant part in the trading system, as this is the one that will define the losses and profits, therefore it is the part that should receive more attention and be allocated more resourses. If we manage to find a method that reduces transaction cost, independent of the algorithm used, the overall performance can greatly increase and we can also provide more options for a trader. To demonstrate further the importance of

the order execution we reefer to the documentation provided by companies that provide algorithmic solutions. In their documentation, the description of the algorithm is guarded. The trader knows what type of algorithm is (the scheduling principle is usually common knowledge) and to what extent the option they input alter the execution, but the actual order strategy is never shown or even hinted. Usually their order placement strategy is marketed as a new and better special system and is followed by graphical plots of execution performance as proof of their worth. This is the core and one of the best kept secrets of the trade execution section of algorithmic trading (Fig. 3.1).

3.8 Evaluation

It is not in the scope of this report to compare the algorithms among themselves. It is not useful to know if one algorithm provides better results than other, for some market conditions, but we do require to know , among the variations of our implementation, which one obtains better performance. This introduces the problem of defining what is the best algorithm and how do we proceed to compare their variations. Usually we evaluate an algorithm by determining the amount of resources, in terms of time and space, required for its execution. It this case however we are less interested on performing complexity analysis and more focused on measuring the quality of the solution. It is easy to make the mistake of thinking that different algorithms try to achieve different objectives, that their schedules are created in a specific form and that they try to solve slightly different kinds of problems. A trader that uses the POV algorithm might be looking to end the day with a percentage of the volume, but the theory behind the algorithm operation is not concerned to obtaining that percentage as its ultimate goal. The algorithm uses the percentage as a means to an end, because it theorizes that by following the schedule provided it will achieve better results on its trades, by reducing market impact. In reality the algorithms share the same ultimate price driven objective: profit [26]. They use different tactics specific to different market situations to try to consistently achieve good returns while minimizing the risk. In this case the profits obtained are the result of two components of the algorithm, the schedule calculation and the order placement strategy.

In trading, each day is different from the previous one and the decisions are made as the orders arrive on the market. This makes testing the algorithms a dificult problem, as they can be seen as belonging to the online algorithms class. By definition an online algorithm is an algorithm that receives a sequence data as discrete pieces and has no prior knowledge of the contents of the next piece. Decisions must be made as new information is received and this constitutes a problem if an optimization is to be made, due to the lack of details about the future. These types of algorithms contrast with the offline algorithm variety, which have access to the whole information in one go, allowing them to make better decisions. In algorithmic trading, although some information regarding future patterns can be obtained, the data received in order to make decisions is provenient from the market intraday movements, which are known to be unpredictable to a certain degree. One of the

ways to analyze online algorithms is by means of competitive analysis. By comparing the algorithm to an optimal offline counterpart (the same algorithm with access to the full data and knowledge of it) we can obtain a measure of the quality between the optimal, offline solution, and the online solution. Since we use a quantitative metric based on the returns obtained, we do not need to do a comparison with the optimal solution, to determine which variation performs best. However, doing a comparison against an optimal solution can give us a basis point to compare how well the algorithms perform in general.

Ideally, we would test our algorithms in the real live market, but that is risky and unwise. The advantage of doing such a test would be the unpredictability and our impact on the market would be felt and considered for evaluation, the disadvantage would be the possible loss of capital. So in order to test the algorithms we must use historical data, which will be suboptimal as the market impact that we will induce will not be accountable.

Chapter 4

Prototype Development

In this section we describe the development and implementation of the prototype for the algorithm trading, integrated with the Sifox Deal application. Taking into account the infrastructure available and the request for an immediate functional prototype in order to showcase the solution to the company clients, priority was given to produce visible results instead of the deployment of the solution in its most logical and ultimately final place. This decision is reflected in the direct integration of the prototype with the Sifox Deal solution. The service that should be server based was developed on the client side having therefore access to the required infrastructure needed for a quick development and test of the application. In the future this component is to be migrated into a server, leaving in the terminals application only the means to interact with it (Figure 4.1).

The implementation of the prototype on the terminal side entails some problems that must be taken in consideration and reflects the future need to migrate the application. The machine where the application is running is not controlled by us. Not having that control entails not knowing what type of machine is running our code or what programs exist there. This can potentially cause a loss of performance, which will impact time sensitive operations, like following a schedule, or quick responses to market events. The complete delegation of the entire module execution to the client can create a risk of the schedule non-completion, either because the system has been turned off or simply crashed. The application only executes when the system is up and running. Regarding system crashes, this approach could potentially improve the system overall stability. By increasing the number of breaking points and moving the issue from the server to the clients, in case of failure the only affected would be the terminals with problems, and not all the available clients. While this could be true, we believe that it would not affect significantly the application, due to the way the other system components are organized in the Sifox application suit. Furthermore, focusing on only one point makes the company take special attention to it, supplying more resources to its maintenance and development, therefore minimizing the risk of a failure. One other issue that arises with this configuration is speed - not really a factor on the types of algorithms presented if not exceedingly slow - and the maintenance of

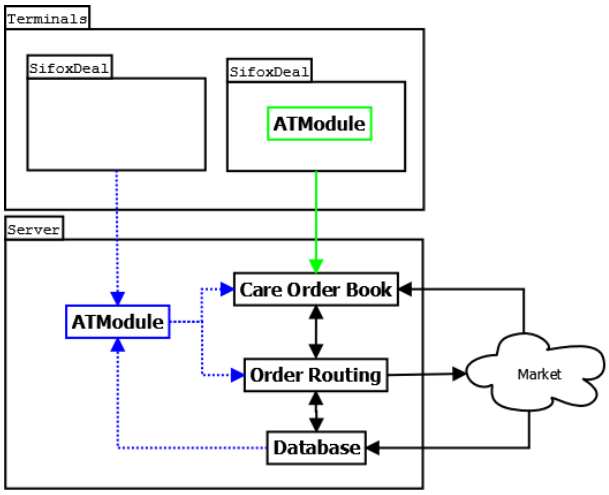


Figure 4.1: Algorithm Module placement alternatives on the Finantech solution. The dotted blue line represents the path taken when the module is in the server side. The green full line represents the module placement on the Client side directly on the SifoxDeal application.

repeated information across multiple clients. All these problems surpass the downsides of having a more centered service, therefore in the future we recommend the port of the application. Other than some small details on the architecture and technology features, that we mention along this report, no particular step was taken in order to prepare for the realization of the port or architecture change.

The SifoxDeal application was developed with the .NET framework[1] (version 4.0) using the native WPF (Windows Presentation Foundation) for its GUI display and interaction. For the IDE we used the Visual Studio 2010 with the support of TFS (Team Foundation Server) for versioning control, creating a branch of the current SifoxDeal application.

4.1 Architecture Overview

The SifoxDeal application follows a modified model-view-controler software pattern. Using its architecture as a reference we created a package to house our implementation in the required sections, the main module sections and the GUI, as described in figure 4.2.

We use the Singleton pattern design to create the following main components with the described functionality:

- AlgorithmManager - Class responsible for the management of the algorithm Instances. It

[1] http://www.microsoft.com/net

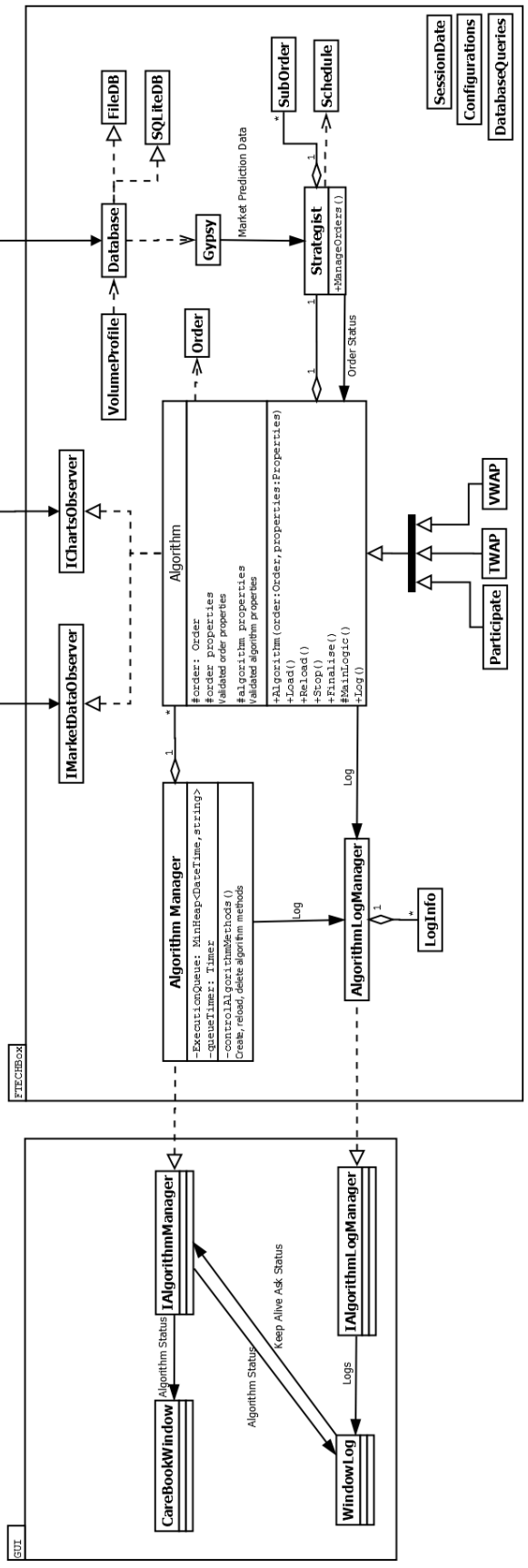


Figure 4.2: UML diagram of the Algorithm Module implementation in the SifoxDeal application, focusing on the AlgorithmManager and Algorithm classes.

provides the point of interaction with the algorithms and manages their schedules (start and stop times) and state.

- AlgorithmLogManager - Manages the Log system of the application.
- Gypsy[2] - This component is responsible for the prediction models and provides information required to the algorithms. It can also provide required information for the client.

Due to the way SifoxDeal was designed, these components have well defined interfaces for communicating with the rest of the application components.

The submission of orders was integrated in the application by substituting the care orders menu. A care order is simply an order to be managed by a trader and can interchangeably be traded by a normal order, when submitting to the algorithm box. We use the ID from the care order to identify the algorithm, piggybacking on the text field of the care to pass the information to our algorithm. Understandably we were unable to obtain access to the main database responsible to store the information of the care order. The API connecting the application to the database was not ready to receive requests to store and retrieve algorithm related information, so we used the text field.

In order to monitor the progress of execution of the orders, a log system was created and added to the Sifox application (Figure 4.3). This Logger not only receives the messages from the algorithm manager regarding the algorithms, but also monitors them, checking if they are running, pooling the algorithm every 20 seconds for its state.

4.2 Algorithms

We will now describe the details of the implementation of each particular algorithm in the described system. Each of them has the same core capabilities with the interaction with the system, implemented by their parent Algorithm class.

The parent abstract class Algorithm has all the components necessary for their interaction with the other system components and creates the template for their functioning. It manages the reception of market data, handles the interaction with the AlgorithmManager, creates the log events and sanitizes the care order properties, converting them to the required type while verifying them (for example converting the order quantity from string to int). The abstract methods that it implements work as a template for the classes that inherit it. These methods are concerned with the algorithm execution mechanic (load , reload, finish, stop and execute) , particular parameters validation and its logic (buy or sell).

[2]The Gypsy is given because of the ability of predicting the fate seen in the romanticized gypsy of novels and films.

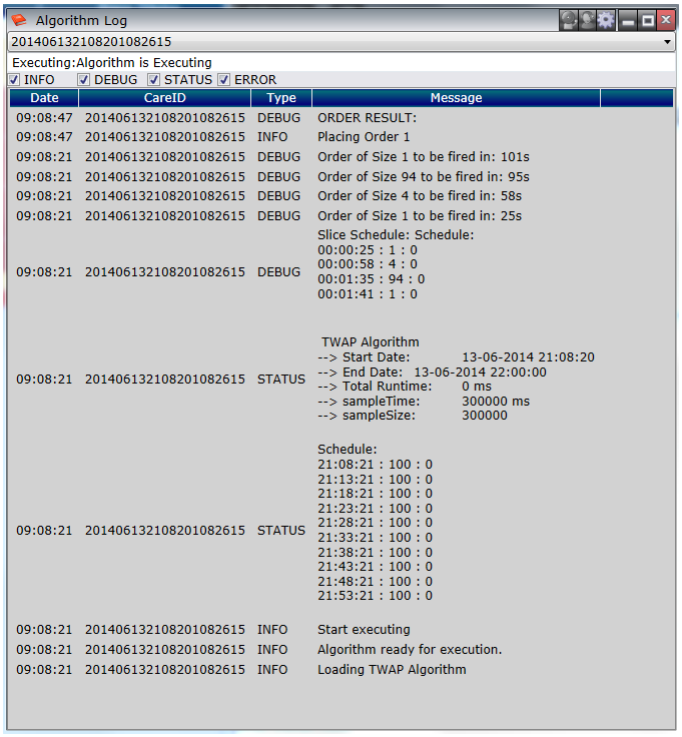


Figure 4.3: Screenshot displaying the Algorithm Log system.

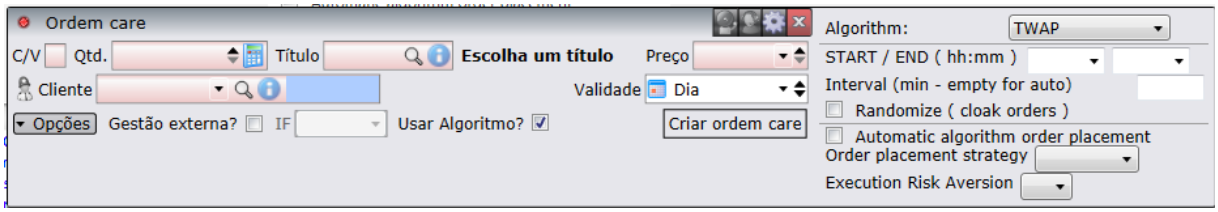


Figure 4.4: The GUI of the TWAP algorithm for order placement.

4.2.1 TWAP

As seen in the UML, (Figure 4.2) the TWAP implements the abstract class Algorithms, overloading all the setup and execution methods. Its unique characteristics are the schedule creation and the randomization process. As with the other algorithms we added the options to the care order window (Figure 4.4).

The schedule creation on the implementation of TWAP is quite straightforward, given a start and end time we divide the quantity using a minimum size interval. We have to make modifications for cases where the quantity of stock is small enough as to not cover all the intervals. It is not recommended to use the TWAP for small quantities of stock, but it was asked from us to not restrict the available quantity. Even though nobody would use TWAP to trade a volume of 100 of a 10 cent stock, some might use it if the stock value is of 500 euros, as thought the volume is small the high

price of the stock increases its impact. In order to accommodate small quantities we automatically stretched the interval window. On the limit, with a quantity of one, there would only exist one interval with the size of the total time-span. We also offer the option for the user to define size of interval, this is to allow the client to place orders using the time-span he/she wants. We propose this use only in the testing phase, as we believe that if the client wants to put a quantity in a chosen interval he/she can simply use the systematic trading methods, selecting the periods and the time, creating his own schedule.

In order to avoid the signaling risk referred in section 3.4, we provide an option for randomizing orders created by the schedule. As with some other options we provide, the use of randomization on the algorithm is optional. It came to our attention that some traders would have a personal idea of how the algorithm should work. Some would like to indicate the period of time the order should be placed and see it happen exactly by the clock, for example every half an hour, even though the behavior would most likely cause a degradation of performance of the algorithm (due to signaling risk, as described in the respective section). This resulted on a dilemma, we can restrict the algorithm operations in order to preserve performance or, we can give a wide array of options to the client. On one hand, restricting options to preserve performance will reduce the possibility of failure caused by the client, saving us the hassle of having to deal with those cases when things go wrong. On the other hand, the lack of control may not be attractive to the client and, even if some options are not commonly used, their existence may be an asset to comfort for the prospective trader. Being this a prototype work we chose to allow the client to do as he wishes and specify the randomization as an optional parameter. This allows us to test the algorithm on both hypotheses of execution, and see which one performs better.

Given a quantity and a time interval, the TWAP algorithm creates a schedule for the time-span. Between each order on that schedule, the randomization will occur. We take the time-span between each order submission and divide it in smaller time intervals (slices). Each slice is created taking into account some parameters (minimum and maximum interval length) in order to guarantee that the total number of stock sent is maintained. Using a defined minimum and maximum slice limit size, we first divide the time-span by the maximum slice size, obtaining the maximum limit number of slices. With this we guarantee that when randomizing the slices sizes the values do not get outside the time-span limit. We then generate the random slices with a min to max range in an equal number to the limit number of slices obtained before. Next, we fill the slots with a number of stock ranging from a minimum chosen value to a maximum never going above the quantity limit, redistributing the remaining in order to force that all the attributed stock is in the slices schedule. The restriction on the number of stock must be better controlled depending on how the payment is made relative to market operations. Usually the market collects as a per-volume disregarding the number of orders. In case of collection per-order we have to be more careful in dividing the orders to obtain the minimum price possible. After the placement of the quantity in each slot, it is likely there will exist a certain

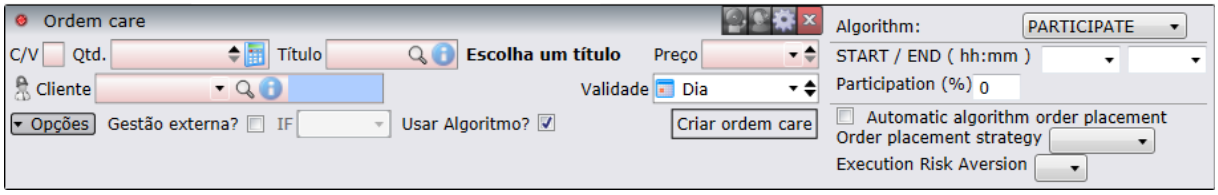


Figure 4.5: The GUI of the POV algorithm for order placement.

number of empty slices at the end of the time-span due to the fact of the existence of a limit on the stock quantity and range imposed on the stock per slot. To look random we need to spread those values over the total time interval. We proceed to iterate over the list of time and quantity associated and swap its members in a random fashion, each element has an uniform probability distribution to be in any place of the schedule. We now have randomized the quantity and time, creating a sub-sample schedule that when executed will not be noticed in the market.

The orders provided from the schedule are kept in a queue and afterward sent to the algorithm strategist for market order placement, according to the selected order placement strategy. We could not find any relevant information regarding the execution parameters, mainly the sub sample minimum/maximum size and time limits. We offer a fixed value for these parameters, but if available we try to obtain the maximum/minimum value for the order size by looking at the mean the market is making, arguing that an order that is in line with the market will attract less attention. Because we do not have access to the detailed market information needed to perform this evaluation, we opted for a default number of one. The time limits should reflect the market liquidity, and vary according to it.

4.2.2 Participation Rate

We implemented the POV algorithm taking into consideration the properties described in section 3.6 and added the functionality to the care order submission form (Figure 4.5). Considering the signaling risk, we opt for introducing the option of delaying execution by indicating the deviation to the tracking percentage value. For each order placed by our algorithm we calculate a random value between 0 and the tracking number. Then, for each new order, we check if the volume of the trades get us behind the target percentage plus tracking deviation, if it does, we place a new order to follow it and recalculate the deviation between the given values. This option is not mandatory. In case of non-selection of a tracking deviation value, POV will be aggressive and place an order whenever a trade occurs (getting us behind our target value). We can also limit the order by its volume. If the required volume to be placed is bellow a selected value, the order will not be placed and will be stored for later. This is particularly useful if the market transaction costs are pay per order and not per-volume. Usually it is per-volume, either way, this option serves as an alternative to the tracking error percentage.

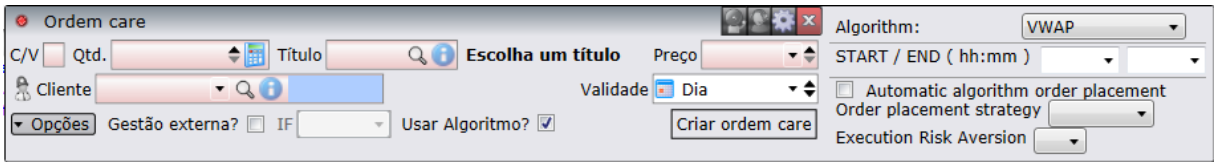


Figure 4.6: The GUI of the VWAP algorithm for order placement.

The Market information is received by this class, via the Sifox existing interface IMarketDataObserver (Fig. 4.2), and it is used to keep track of the executed trades volume. Due to data feed limitations we must keep track of this ourselves, opposite to receiving that information directly from the feed, in order to filter the desired trades. This control allows us to monitor the prices of the executed trades and only add the volume of those whose price conforms with our order characteristics. This way we opt for the option of ignoring the volumes that fall below our execution price, eliminating the need for the algorithm to worry about the trades lost, that could possibly come up again as a volume spike. Although we also talked about detecting spikes in prices due to normal market movements, we do not achieve that, transferring the responsibility to the existence of a limit on the care order price, inserted by the trader.

4.2.3 VWAP

The VWAP algorithm as the previous ones, implements the abstract class Algorithm class and was added to the care order submission form (Figure 4.6). The VWAP requires the construction of a volume profile to function (discussed in section 3.5).The profile is loaded and the schedule is created at the moment of the algorithm execution. As previous algorithms the scheduled next order waits execution and when time arrives, the strategist executes it taking into account the order placement strategy selected.

The Volume Profile is the cornerstone of this algorithm, the schedule generation is dependent on its existence. Section 4.3 provides the details of the construction of the profile, how it is implemented and its use.

4.3 Volume Profile Model Construction

In order for the VWAP to create a schedule it must be supplied a volume profile (as described in 3.5), constructed by using historical data. There are several known facts about volume profiles that reflect their stability [14]:

- Usage of more than 30 days of data offers no statistical improvement to the profile shape. The

more liquid or stable a stock is the less data-points we will need. This is not applicable to illiquid stock.

- An old data-set is not detrimental to the estimate of the profile curve, given that the trade structure is stable and the traders operate the same way now as they did in the data-set time period.

This provided stability indicates some limits that the data used to calculate the profile will have: up to a point having the best data won't work to improve the accuracy of the profile, and the performance will depend on the model used. The ability of producing quality profiles is dependent on the stock used. A stable stock, or one which has gradual trends, will obtain better results on the predicted volume [13]. This means that for some equities the algorithms that use volume prediction will not be the best. There is also the issue of seasonal one day trends, caused by events like Christmas and earnings reports. These anomalous days may cause a spike variation on the volume that may be difficult to handle by some volume profile models.

4.3.1 Data-set

The historical data used comprises the intraday trades for the Google stock (GOOG) on the NASDAQ exchange from 19 February to 7 March of 2014 and 20 March to 15 April of 2014 on a total of 32 trading days. The files where obtained using the now deprecated Google finance API[3]. They are CSV files with a custom header and data consisting of the intraday trades accumulated in the minute interval having each the respective values for the open, close, high, low and volume. We verified that the data-set needed to be cleaned, mainly because of missing intraday ticks. Analyzing the data, we found that the number of consecutive jumps were no bigger than two (2% of all the gaps were of size two), of all days 22 had gaps (69% of the total days) averaging 6 gaps per file in a total number of 138 gaps in a total of 12480 ticks (1% of the total ticks) . This missing data can have two provenances, simply in that minute there where no trades made (improbable since the stock is known to be liquid) or there was an error registering the trades. Either way, the missing points were absorbed into the time-frame chosen periods, not having a big impact on the overall data, as the tick volume can be negligible for a considerable large trading period. A script developed in Python[4] is used to clean and transform the data, exporting it as a CSV file. The data obtained is organized as a file per day and each file has listed the time period, aggregated from the intra-day, with the corresponding total volume (Table 4.1).

In order to test the described methods without the overhead of having to implementing them on the system, we used the R [5] software with the default pre-installed packages. By using R script

[3]Deprecated in June 3 2011, but not shutdown[7].
[4]https://www.python.org/
[5]http://www.r-project.org/

Date	Close	High	Low	Open	Volume
a1392820200	1204.11	1205.6	1204.11	1205.3	37744
1	1206.135	1206.57	1203.15	1204.715	24794
2	1205.16	1205.39	1204.241	1204.28	9565
3	204.67	1206.42	1204.44	1205.18	10328
...					
387	1204.4	1204.86	1204.35	1204.86	3524
388	1204.89	1204.99	1204.01	1204.39	5114
389	1204.832	1205.15	1204.76	1204.96	6256
390	1204.11	1205.08	1203.23	1204.85	138213

(a)

Tick (15 min)	Volume
0	37744
1	96032
2	71801
3	53115
...	
24	84763
25	114032
26	103992
27	141545

(b)

Tick (15 min)	Volume (%)
0	0.0256
1	0.0653
2	0.0488
3	0.0361
...	
24	0.0576
25	0.0775
26	0.0707
27	0.0963

(c)

Table 4.1: Sample of volume profile construction from the data obtained via Google API. a) Data as received by the google API. b) Volume agregattion by Date, each tick has a size of 15 minutes, except the first and last ones. First and last tick correspond to close and open bid. c) Conversion to volume percentage.

capabilities we transform processed data and produce the results which we analyze.

4.3.2 Tick Size

We must now take into consideration the minimum time slice allowed for the volume profile, the length of time we will base our prediction. If we could predict the volume within a one minute tick, our results would be great, not only we would approximate the volume on a smaller scale and more closely to what is happening but also, we can offer a time scale to the minute allowing an order placement to be done in any time without loss of precision. This, however, is extremely difficult and in practice will perform terrible comparatively with the increase of the tick size. With a sample taken of five consecutive days, we took their volume profile and calculated the cumulative error per tick size (Figure 4.7) and, as expected, the more we increase the tick size the lower the error. At the limit, a tick size of the whole day, will correspond with an error of 0. This is in line with the intuition that

predicting a volume over a longer period of time is better than doing it in every minute. A good error reduction can be achieved at intervals of 15 minutes, while maintaining flexibility for the choice of time periods.

4.3.3 Evaluation Method

As described in section 3.5 the VWAP performance is dependent on how the profile tracks the current day volume, assuming the average price is achieved. Having this information in mind, the best volume profile will be the one whose deviation from the current day volume is as close as possible to zero. This difference can be measured by calculating the absolute deviation between zero and the difference of the profile with the current evaluated day trade as an absolute value, described by the following formula:

$$D_i = (\mid (p_i - t_i) \mid) \tag{4.1}$$

where i is the period considered. Alternatively we can check how similar is the testing profile graph to the volume profile using a chi-squared test (Pearson's chi-squared test formula 4.2) to evaluate the goodness of fit between them. Being the observed values the test profile and the expected the pre-calculated volume profile we can apply the following formula:

$$X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{4.2}$$

This value will reflect the absolute deviation, the lower the result of X^2 the closer is our model created volume profile to the current volume.

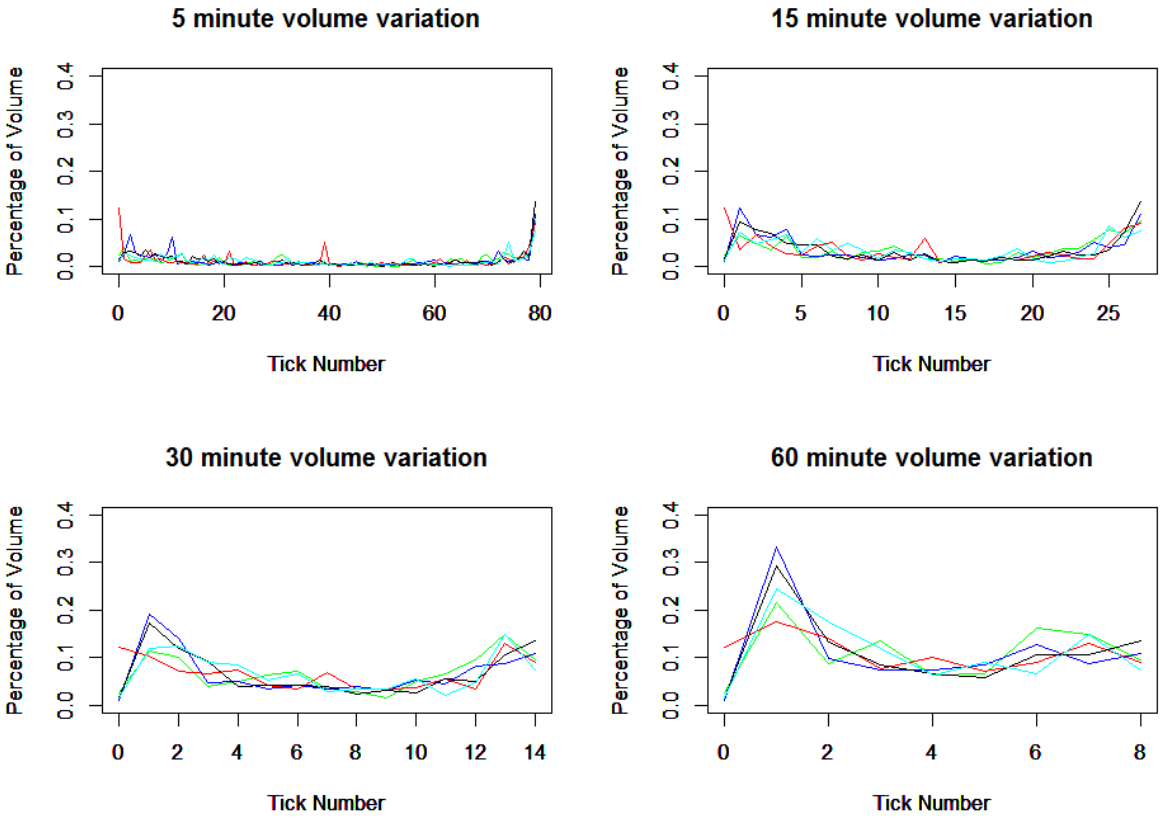
4.3.4 Models

Having the historical data and a method of evaluation we just need a model that will allow us to create the profile. Usually the calculation of the profile is made by applying an average over historical data [13, 14]. We apply the following models to the data-set:

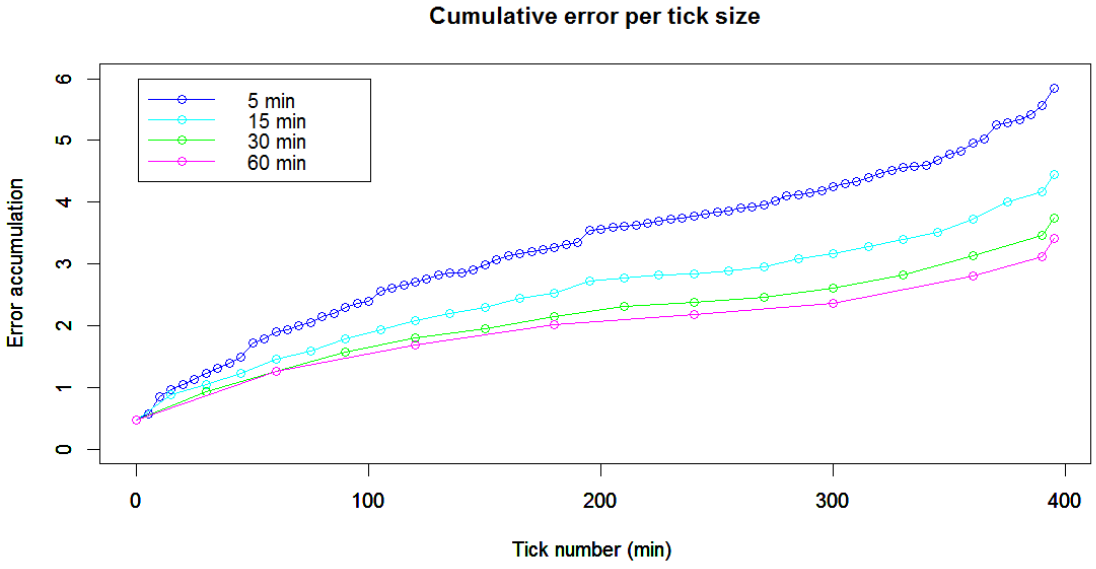
- Simple Average (SA) - By using an average along the data-set period for the trading days n, we calculate for each trading day i and each period j of that trading day:

$$u_j = \frac{1}{n} \sum_{i=1}^{n} \frac{v_{ij}}{V_i} \tag{4.3}$$

  where V_i is the total volume of that day. The result will be a vector u_j containing the percentage



(a)



(b)

Figure 4.7: Intraday percentage of volume variation and the associated cumulative error from five consecutive days (19 to 25 February, GOOG stock). a) Intraday percentage of volume variation between five consecutive days in different time scales. The first and last tick are respectively the open and close. b) Cumulative error per tick size. The first and last tick are respectively the open and close. The error is calculated as the absolute difference between the points of all five consecutive days.

values for each interval of the chosen length. The data used will be set in a fixed time frame. Overtime, because of seasonal trends and variations, the performance of this profile will degrade forcing the value to be recalculated in order to catch up to the recent trends. In terms of storage space this means we will have to maintain the historical values of stock for at least the time-frame used on calculating the SA and the profile itself.

- Simple Moving Average (SMA) - With a moving average we will do a rolling average as the time-frame moves. Using a selected time period (n), as time passes, we will add the new day, dropping the oldest day. This will allow us to follow the trend of the data variation.

$$SMA_{n+1} = SMA_n - \frac{v_0}{n} + \frac{v_{n+1}}{n} \tag{4.4}$$

  In order to calculate the SMA we must have in storage all the days the profile contains, in order to allow swapping it out for the new values.

- Exponential Weight Moving Average (EWMA) - EWMA applies a weight to the moving average by stating that the closer the historical day is to the present day the more importance (weight) it will have when calculating the volume profile. This is achieved by using formula 4.5. Selecting the value of the α parameter will control the importance of the more recent values on the calculation. An α with value of one, will place extreme importance on the present values, while a value of zero will only use the past measures for the calculation. This allows us to follow a trend on the volume profile changes over time, placing importance in past or present values as we see fit.

$$\overline{x}_k = \alpha x_k + (1 - \alpha)\overline{x}_{k-1} \tag{4.5}$$

  The alpha value can be correlated with the stock variation and its choice will depend on the equity used. For a stable liquid stock a high value of alpha should be chosen, because the trend will move slowly in each day. For an equity with quick variations and high volatility that creates subsequent trends, a lower value should be used. The value we use for alpha is 0.165 (the result was obtained as described in annex A.2). One of the other advantages of EWMA comes from the way of how it is calculated. There is no need to store all historical data to update the value (as opposed to the other two methods), we only need the previous calculated profile and the most recent previous day.

In order to test these methods, we divide our data-set in two parts, data that we use to construct the model and data for testing. In case of SMA and EWMA the test data will also be contained in the model, because of methods specifications. Using a time-span of one month (averaging 22 trading days), half a month (11 trading days) and five days we check which of the methods produce the best evaluation metric.

Model	Chi-Squared (X^2)	Standard Deviation (σ)	Variance(σ^2)
SA5	0.2539663	0.1794603	0.03220601
SA11	0.2509637	0.2084890	0.04346765
SA22	0.2565831	0.1850758	0.03425305
SMA5	0.2308193	0.1696247	0.02877254
SMA11	0.2257696	0.2028504	0.04114827
SMA22	0.2254268	0.1893183	0.03584141
EWMA5	0.2235382	0.1673634	0.02801052
EWMA11	**0.2131483**	0.1794608	0.03220616
EWMA22	0.2149000	0.1767366	0.03123581

Table 4.2: Chi-Squared evaluation results applied to the three mean methods for volume profile calculation. The best result is presented in bold and bellongs to EWMA with 11 days.

4.3.5 Results

The results obtained (table 4.2) show that, for the GOOG stock, the best model used is the EWMA. As expected, the SA method has performance degradation over time and even taking a month of data the results are not better than five days. Recalculating the SA in order to improve performance will transform it in a SMA with a time lag. The two best results are close to each other, but although the EWMA 22 has a worse result than its 11 days counterpart, its standard deviation is lower, being possible to produce better results on the long run (as it happens also with the SMA). Because of the constraints we have on space and data availability, we implemented the eleven day option of the EWMA algorithm that gave us the best results. This will allow us to store less data and calculate the needed value along the day, while incurring in only a small loss of accuracy.

4.3.6 Integration

The calculation of the volume profile is a key component of the system for the correct functioning of the VWAP algorithm and possesses some properties that will dictate how we implement it. Taking into account the model chosen in the previous subsection, the EWMA 11 , we will have to apply constraints that will guide some of the development decisions. After creation, the model must be updated every day using the previous day information. We could use a lazy-evaluation logic and start building the profile when it is needed, assuming we have the data to do so. This approach would be detrimental to our application performance, mostly regarding the quality of service offered. Although the calculations are not computationally expensive to run on the client side, to do so would introduce a delay, causing problems in case of a tight schedule time for execution (in terms of seconds). A more important impact would be felt in the QoS, as the previously calculation of the profile can add useful information to the client at the moment of execution. For example, allowing the system to issue a warning in case of failure or insufficient data, not having the user to discover the error at the moment

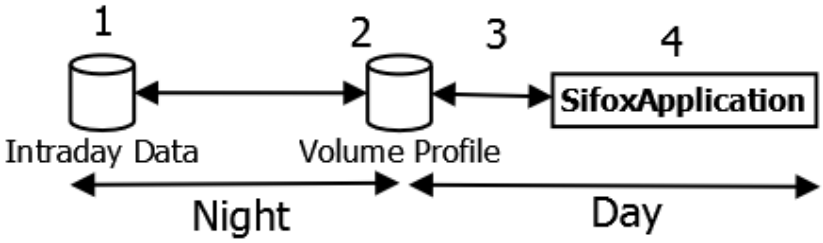


Figure 4.8: Description of the volume profile implementation. 1 - At the end of the trading period, intraday data is compiled and retrieved; 2 - The information is fed into the model and the volume profile is created and stored in the database; 3 - When an order is set for execution, the profile is retrieved; 4- Calculations are made to fit the profile in the current order time-span;

when he is counting for the feature to work, lot leaving time to plan accordingly. It is also possible for some stocks to not be used everyday with algorithms that require volume profile calculation. The next day, after a gap of this type, the update would fail, because previous updates were not made and we will have to store more past information to handle this issue, invalidating the advantages of using the EWMA 11 model.

Considering the previous constraints and the server architecture described in subsection 4.1, we implemented the solution that follows the path described in Figure 4.8. The model used (EWMA 11) consists of several simple calculations that are possible to execute by using of a relational database script[6] as such we do not see a problem of allowing it to do so. Usually the working hours of the investment banking are limited for a set period of time. In our case, the venues of investment chosen are the European, North-African and American thus for a period of time the database will have no workload, as all venues will be closed. Instead of adding more complexity to the algorithm engine we opt for the separation of the volume profile data creation from the rest of the application logic. This way we do not need to add new machine infrastructure to execute the maintenance, we use the built-in capabilities of the database to create the volume profile to be used for the next day, while maintaining relevant information, like the validity of the profile. Although we specify in the schemes and architecture a database detached from the rest of the system, this might not be ideal. We maintained the two separate because of our inability to use and access the proprietary SQL database present in the Sifox Solution. In the future these two parts can be combined and resources conserved. If proved necessary though, this service can be detached and maintained as a middle-tier cache service between the data tier and the logic tier.

Due to the fact that the period of execution is of 15 minutes, we restrict the time available for the schedule in fixed periods of the same size. This control is advantageous since it allows us not to worry if the time periods of the volume profile fit the selected algorithm execution time-span, saving us the need to add complexity by performing more operations on the data. Furthermore these

[6]Assuming the script the database uses possesses similar capabilities to the mysql aggregator functions (SUM) and arithmetic operations.

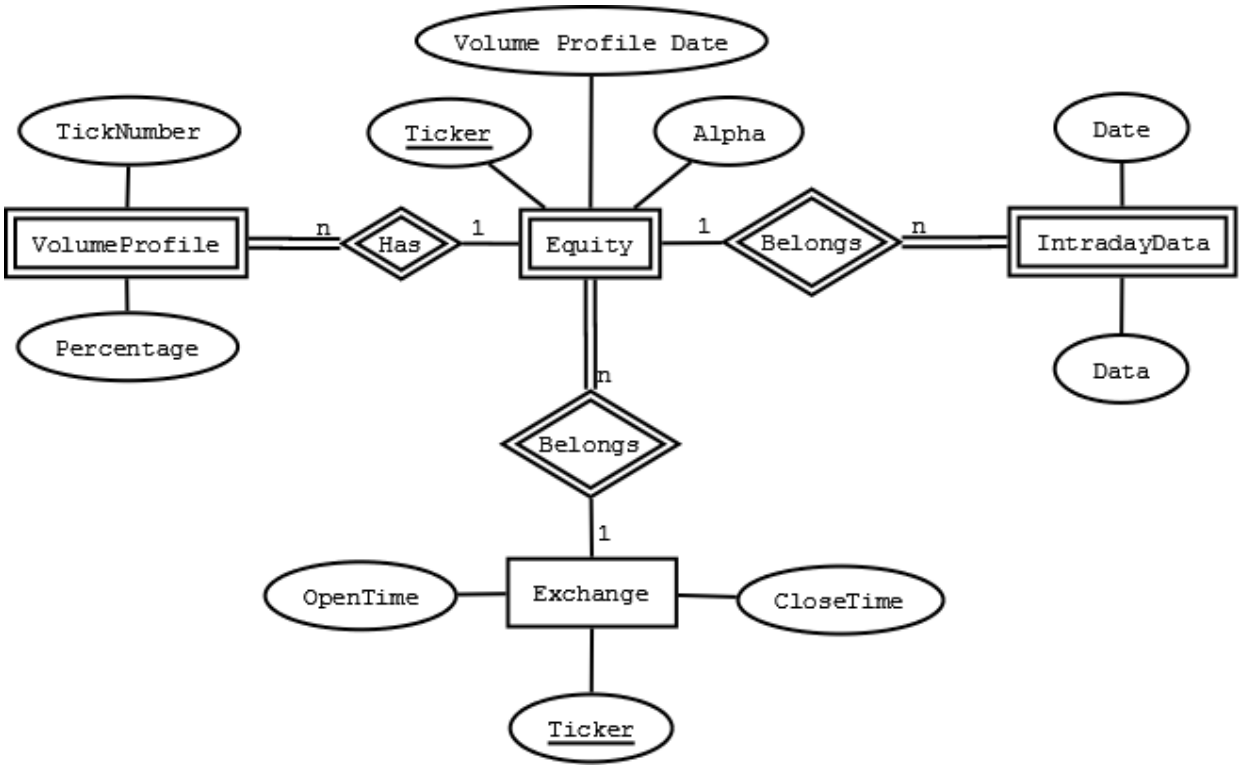


Figure 4.9: Entity-Relationship model for the volume profile database and related components.

operations would have to be executed at run time, adding to the problems described above. We also stop orders from entering the intervals at the middle. Making the periods static, we can simply postpone orders to the next available interval. But that would induce the trader in possible error, because we would not know how exactly his orders were being dispached. We find that by forcing the indication of the available time, we provide more information and do not trick the trader.

For the prototype we used a SQLite database[7] to achieve the described functionality. Using it allow us to focus on the prototype creation instead of worrying excessively with infrastructure. SQLite has small code footprint, is self-contained and requires zero-configuration, while maintaining all the most important features we need from a relational database. This properties make it an ideal candidate to ship with the client of the application (as our prototype development structure suggests).

We create an entity-relationship model based on our problem specifications (Fig. 4.9) and apply it to the database. Because some equities exist in multiple exchanges with the same tickers they are dependent on the Exchange. The OpenTime and CloseTime are used to determine the time of equity trading according to the exchange, however the equities are further subdivided in groups that can trade in a special schedule inside or outside that of the Exchange. We do not take into consideration those cases, but for further development they must be taken into consideration.

[7] http://www.sqlite.org/

We had difficulty in obtaining intraday data with minute resolution (1 to 15 minutes) and had no access to the Sifox SQL database so, for testing purposes, we included the intraday data and filled it using the the Google API data (as described in section 4.3.1). We then write queries that can process the data and create the volume profile and code them in our Sifox Application. These queries can also be run outside the application for creating and maintaining the volume profile. Finally our application just needs to call for the profile using an internal API to access to the the database[8]. We first check if the data field on the equity is in order, the time must be from the previous day and its presence (not null) indicates the presence of the profile on the database. Not finding it, it throws an error, stopping the algorithm from executing. The application receives the complete volume profile and adapts it to fit the required time-span chosen at the algorithm creation.

There is also the possibility of adding a cache system to the profile. Because of the tick size the time for order submission is limited to sections of time, making the choice of schedule limited in size. We can reduce the burden of calculating the schedule on the database by using a cache for those values, decreasing our execution time. In our case we decided not to deploy it, because we believe that the performance gain from its use on this case would be minimal and the extra infrastructure would prove unnecessary. This would not be the case if we had the calculation on the server side, in that case the saving could be greater, provided that there were enough people using the system and placing a number of order constantly. In that case, extra infrastructure should be created as a cache-only memory.

[8] In previous versions this was done using the file system for storing the information as a file.

Chapter 5

Conclusion

We studied the financial market in order to obtain a better understanding of the problems assossiated with algorithmic trading. We then describe the general architecture of trading algorithms and detail the most commonly used, impact-driven algorithms. We study some possible model for order placement execution, and explain a method for evaluating the performance of the algorithms. Finally we implement the algorithms and integrate them on the Sifox application, developing and testing a model for the VWAP, volume profile model construction.

5.1 Critical Discussion

We focused on the creation of the prototype and its implementation directly on the SifoxDeal application. We did not account for the state the application was at. The lack of documentation regarding the SifoxDeal application became a time hindrance, and conforming to the current implementation proved to be a greater challenge. We discovered also that the heart of the system was not on the schedule generation as we were first led to believe, but on the order placement strategy, though this became apparent latter on. Regarding the execution strategy we lacked the data necessary to apply the models and test them. The SifoxDeal testing environment, if receiving market data, would provide for a quick simulator, capable of testing the strategies in real time.

The data we have for test, analysis and model construction proved to be insufficient. The company had no access to any reliable data feed *in situ*. The data feed available to the application is provided by the client that buys the solution and unlucky for us, the company satellite feed was disrupted mid January due to a storm, preventing us the use to it. A closer look to the topics discussed in this internship report should be studied in more detail and tested with quality data.

5.2 Future Work

We started this internship with the objective of creating a prototype system for automatic trade execution and, after we started investigating the problem, we discovered the many aspects that can influence the performance of the algorithm and its execution. Performance, measured by the returns, is the most important objective of these algorithms and they come from the correct use of strategies at the correct time. In our opinion the order placement strategy is the core of a system of this type. Having a strong, reliable and consistent strategy system would allow an overall improvement on all algorithms, substantially raising the profile of the Sifox Solution to the market. More resources should be spent at developing the order placement strategy and maintain a continuous investment and monitoring, due not only to equities dynamic changes (more generally market conditions changes) but also to the application of new rules by the exchanges. With the use of real time and historical quality market data, the use of dynamic models capable of harnessing the extra information would be enough to provide that increment and the option of developing models to function on higher steps of the architecture chain (section 3.2).

We learned that traders (and clients) like to have absolute control over the strategy they employ, efforts should be made to create a system that would allow for systematic trading. By empowering the trader with the means to create their own market strategy, while attributing to the Sifox algorithm component the proper order execution, great value could be added to the solution. This only increases our awareness to the importance of providing a robust order execution strategy. There is also the need to provide a report on the algorithm functioning. All the trades must be registered and information on why the decision to place an certain order were made. This is of extreme importance because of auditing, to justify the algorithm behavior to the client (as so he does not think it was a software fault, but a sequence of decisions made on the available facts), and to debug and develop better and more precise algorithms.

Finally the application should be ported to the server infrastructure in order to get better support and robustness. This will allow it to directly tap to all the resources available, improving performance, and truly achieve a complete product.

Bibliography

[1] The Atlantic Telegraph. *Merchants' Magazine & Commercial Review*, pages 1–19, 1865.

[2] Stephen A Berkowitz, Dennis E Logue, and Eugene A Noser. The Total Cost of Transactions on the NYSE. *The Journal of Finance*, XLIII(1), 1988.

[3] Brian Coyle. *Debt & Equity Markets: Equity Finance*. Financial World Publishing, 2002.

[4] Brian Coyle. *Debt & Equity Markets: Overview of the Markets*. Financial World Publishing, 2002.

[5] NYSE EURONEXT. EURONEXT Order Types. https://www.euronext.com/trading/order-types. Accessed on March 2014.

[6] NYSE EURONEXT. NYSE EURONEXT - Technology Timeline. http://www.nyse.com/about/history/timeline_technology.html. Accessed on March 2014.

[7] Adam Feldman. Spring cleaning for some of our APIs, 2011. http://googlecode.blogspot.pt/2011/05/spring-cleaning-for-some-of-our-apis.html. Accessed on April 2014.

[8] Finantech. Finantech Webpage, 2014. http://www.finantech.pt/. Accessed on March 2014.

[9] Sami Franssila, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, third edition, November 2009.

[10] Puneet Handa and Robert Schwartz. Limit Order Trading. *Journal of Finance*, 51(5):1835–1861, 1996.

[11] Lawrence Harris. Optimal Dynamic Order Submission Strategies In Some Stylized Trading Problems. *Financial markets, institutions & instruments*, 7, 1998.

[12] Joel Hasbrouck. Empirical Market Microstructure: The Institutions, Economics, and Econometrics of Securities Trading. page Chapter 2. Oxford University Press, 2006.

[13] Barry Johnson. *Algorithmic Trading & DMA: An introduction to direct access trading strategies*. 4Myeloma Press, London, first edition, 2010.

[14] Robert Kissel and Morton Glantz. *Optimal Trading Strategies: Quantitative Approaches for Managing Market Impact and Trading Risk*. Amacom, 2003.

[15] Andrew W Lo and A Craig Mackinlay. *A Non-Random Walk Down Wall Street*. Princeton Universty Press, 1999.

[16] Christopher Matthews. How Does One Fake Tweet Cause a Stock Market Crash?, 2013. http://business.time.com/2013/04/24/how-does-one-fake-tweet-cause-a-stock-market-crash/ . Accessed on June 2014.

[17] Tom Middleton. Understanding how algorithms work. Where does time slicing and smart order routing end and randomising your orders through complex algorithms begin? *Algorithmic trading, a buy-side handbook.*, pages 21–27, 2005.

[18] NASDAQ. NASDAQ REFERENCE GUIDE - Order Types and Routing Strategies. 2014.

[19] Y. Nevmyvaka, M. Kearns, a. Papandreou, and K. Sycara. Electronic Trading in Order-Driven Markets: Efficient Execution. *Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, pages 190–197.

[20] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 673–680, 2006.

[21] NIST/SEMANTECH. *e-Handbook of Statistical Methods*. 2014. http://www.itl.nist.gov/div898/handbook/. Accessed on March 2014.

[22] Giuseppe Nuti, Mahnoosh Mirghaemi, and Philip Treleaven. Algorithmic Trading. *IEEE COMPUTER*, 44(11):61–69, 2011.

[23] Brian O'Connell. Can Tweets And Facebook Posts Predict Stock Behavior?, 2014. http://www.investopedia.com/articles/markets/031814/can-tweets-and-facebook-posts-predict-stock-behavior-and-rt-if-you-think-so.asp. Accessed on June 2014.

[24] Carol L. Osler. Stop-loss orders and price cascades in currency markets. *Journal of International Money and Finance*, 24(2):219–241, March 2005.

[25] Sneha Padiyath. High-tech algo trades pick up, 2014. http://www.business-standard.com/article/markets/high-tech-algo-trades-pick-up-114061001132_1.html. Accessed on June 2014.

[26] Robert Pardo. *The Evaluation and and Optimization of Trading Strategies*. WILEY, second edition, 2008.

[27] William L. Silber and Kenneth D. Garbade. Technology, communication and the performance of financial markets: 1840-1975. *The Journal of Finance*, XXXIII(3):819–833, 1978.

[28] George J. Stigler. The Economics of Information. *The journal of political economy*, LXIX(3):213 – 225, 1961.

[29] Jiaqi Wang and Chengqi Zhang. Dynamic Focus Strategies for Electronic Trade Execution in Limit Order Markets. 2006.

[30] Jian Yang and Brett Jiu. Algorithm Selection: A Quantitative Approach. *Institutional Investor Journals*, 1:26–34, 2006.

[31] Chaiyakorn Yingsaeree. *Algorithmic Trading : Model of Execution Probability and Order Placement Strategy*. PhD thesis, University College of London, 2012.

Appendix A

Appendix

A.1 Acronyms

API Application Programing Interface

CSV Comma Separated Value

EWMA Exponentially Weighed Moving Average

FAK Fill And Kill

FOK Fill Or Kill

GTC Good Til Canceled

ICE Intercontinental Exchange

IDE Integrated Development Environment

IOC Immediate Or Cancel

ICE Intercontinental Exchange

IPO Initial Public Offering

MSE Mean Squared Error

NASDAQ OMX National Association of Securities Dealer Automated Quotation Optionsmäklarna Exchange

NYSE New York Stock Exchange

POV Percentage Over Volume

QoS Quality of Service

SA Simple Average

SMA Simple Moving Average

SSE Sum of the Square Error

TWAP Time Weighed Average Price

VWAP Volume Weighed Average Price

WPF Windows Presentation Foundation

A.2 Alpha calculation for EWMA

As described in section 4.3, the value of alpha on the EWMA formula is related to the characteristics of the equity used. In order to proceed with our implementation of the VWAP algorithm we didn't focus on discovering which parameters influence this value but instead used a trial-and-error approach. Using the data-set for the model calculation as our sample (in order of not to data-fit to the rest of our sample) we try to find an alpha value that will have the lowest MSE (mean squared error)[21]. In order to achieve this purpose we will progressively test the values of alpha, beginning in the range of 1 to 0 with 0.1 increments, and select one with the lowest SSE (sum of the square error), then we will test for $\alpha \pm \Delta$ increasing the resolution to the required decimal places[1]. MSE is the sum of the square error (SSE) mean, we calculate this value from our data-set (D) by iteratively calculate the EWMA of the previous values and compare them with the current .

$$SSE = \sum_{i=2}^{D} (y_i - ewma(y_{i-1}, \alpha))^2 \tag{A.1}$$

$$MSE = \frac{SSE}{\#D} \tag{A.2}$$

We apply the method to the time-frame comprising 22 and 11 days to calculate the value of alpha to up 3 decimal places (table A.1). We obtain for 22 days an alpha value of 0.148 and for 11 days 0.183 we opted to use the mean of this two values 0.165, as both days will be used to calculate the volume profile and a close enough value of alpha capable of fitting both of them is essential. The GOOG stock is know to have a stable traded volume, so a lower value of alpha is consistent with the description.

A.3 Priority Queue Implementation

In order to add time constraints to the algorithms we implement a priority queue. A priority queue [9] will maintain a set of elements ordered by a key, in this particular problem the key is a date time object. We will associate to the key the corresponding algorithm instance. When getting the start time, we start the algorithm execution, the equivalent is made for the stop queue. We use a priority queue because of the need to alter and add values with a priority fashion. We will be using a minimum binary heap. A binary heap is a data structure that can be represented as a tree that satisfies the following properties:

- The tree is completely filled in all levels except the last, which can be partially filled.

[1] Alternatively we if the calculation is not computationally expensive, search all the range of values.

Alpha	MSE (22 days)	MSE (11 days)
0	0.0005323462	0.0006208016
0.1	0.0003987647	0.0005249954
0.2	**0.0003968432**	**0.0005118719**
0.3	0.0004144375	0.0005243555
0.4	0.0004378793	0.0005462036
0.5	0.0004648764	0.0005729843
0.6	0.0004957227	0.0006041932
0.7	0.0005316607	0.0006408296
0.8	0.0005744062	0.0006844927
0.9	0.0006260465	0.0007370641
1	0.0006891653	0.0008007120

(a)

Alpha	MSE (22 days)	MSE (11 days)
0.10	0.0003987647	0.0005249954
0.11	0.0003964730	0.0005214924
0.12	0.0003948892	0.0005186354
0.13	0.0003938878	0.0005163514
0.14	0.0003933691	0.0005145762
0.15	**0.0003932533**	0.0005132533
0.16	0.0003934760	0.0005123328
0.17	0.0003939852	0.0005117709
0.18	0.0003947383	**0.0005115289**
0.19	0.0003957006	0.0005115726
0.20	0.0003968432	0.0005118719

Alpha	MSE (22 days)	MSE (11 days)
0.20	0.0003968432	0.0005118719
0.21	0.0003981424	0.0005124002
0.22	0.0003995781	0.0005131341
0.23	0.0004011338	0.0005140528
0.24	0.0004027954	0.0005151379
0.25	0.0004045511	0.0005163732
0.26	0.0004063908	0.0005177444
0.27	0.0004083061	0.0005192387
0.28	0.0004102895	0.0005208451
0.29	0.0004123351	0.0005225536
0.30	0.0004144375	0.0005243555

(b)

Alpha	MSE (22 days)	Alpha	MSE (22 days)
0.140	0.0003933691	0.150	0.0003932533
0.141	0.0003933406	0.151	0.0003932613
0.142	0.0003933160	0.152	0.0003932726
0.143	0.0003932953	0.153	0.0003932871
0.144	0.0003932783	0.154	0.0003933049
0.145	0.0003932652	0.155	0.0003933258
0.146	0.0003932557	0.156	0.0003933499
0.147	0.0003932498	0.157	0.0003933769
0.148	**0.0003932475**	0.158	0.0003934070
0.149	0.0003932487	0.159	0.0003934401
0.150	0.0003932533	0.160	0.0003934760

Alpha	MSE (11 days)	Alpha	MSE (11 days)
0.170	0.0005117709	0.180	0.0005115289
0.171	0.0005117329	0.181	0.0005115209
0.172	0.0005116981	0.182	0.0005115158
0.173	0.0005116663	0.183	**0.0005115134**
0.174	0.0005116377	0.184	0.0005115138
0.175	0.0005116121	0.185	0.0005115170
0.176	0.0005115896	0.186	0.0005115228
0.177	0.0005115700	0.187	0.0005115313
0.178	0.0005115534	0.188	0.0005115425
0.179	0.0005115397	0.189	0.0005115562
0.180	0.0005115289	0.190	0.0005115726

(c)

Table A.1: Alpha value calculation. Results of the alpha value calculation process applied to 22 and 11 trading day data. The best values are shown in boldl. a) alpha range between 0 and 1. b) Calculations for $\alpha \pm \Delta$ with alpha being the best obtained value and $\Delta = 0.1$. c) Calculations for $\alpha \pm \Delta$ with $\Delta = 0.01$.

- The nodes must follow a heap property. In case of a minimum heap a node is parent to another, that implies that the fist one is ordered in respect to the other, being the first key lower or equal then the child. The up-most root is the global minimum.

A.4 Predicting the future using Time Series analysis

In order to apply our order placement strategy we need to have information of how the market will behave in the next instant. The more accurate our prediction is, better will be the performance of our model, provided that the model assumptions are correct. At the extreme, when given a market state, if we know exactly the next state we can act in order to maximize our price, that is, minimize our trading cost, and we would be millionaires for certain. Sadly, in our case, we are not able to forecast the market reaction with an accuracy of 100%, but a small glimpse is better than going in completely blind. For analysis methods to work and be useful we first must ask if it is possible to predict the market future.

Two hypothesis oppose the notion that the market follow predictable patterns, the Random Walk Hypothesis and Efficient-Information Market. The former states that the market prices change in a random walk pattern and therefore cannot be predicted, this notion is consistent with the latter. The latter states that the market is efficient and thus price changes reflects all the available information. If there are no inefficiencies there is no way to beat the market therefore making the use of our predictions void, the information we learned in the prediction is already reflected on the price and so it is of no use to us. The Random Walk Hypothesis has been refuted, there is the general consensus that the market has some patterns and follows trends, and its prediction is possible to some extent[15]. The Efficient-Information Market hypothesis on the other hand has seamed difficult to prove, or disprove and some argue that the market can never be efficient, for example, traders that act on noise information and others that need liquidity will cause inefficiencies in the market. Although we ignore the effect of the efficient market in our prediction perspective it is interesting to notice how it has been influenced by the evolving information technology. In fact, a study from [15] observed that recent market data was closer to the Random Walk hypothesis than the older. In our opinion, this is the effect of the information technology in the market. As it previously had happened with the trans-Atlantic cable the new information technology increases the market efficiency, the disparity of prices is reduced the cost of doing businessmen goes down but still, in the present time, there are inefficiencies that still prevail.

In order to do the forecast we will apply time series analysis. A time series is an ordered sequence of values sampled at spaced time intervals. By using analysis tools on this time ordered data, a model can be created with the ability of predicting trends.