Algorithmic Trading Strategies

Project Report - Big Data

Team members:

Jelena Antić, Paulina Grnarova, Filip Hrisafov,

Vidor Kanalas, Miloš Stojanović, Alexios Voulimeneas

TA: Aleksandar Vitorović

Professor: Christoph Koch

Lausanne, 20.05.2014

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Table of Contents

Abstract

In the last few years, electronic limit order books, which collect incoming limit orders and automatically match market orders against the best available limit order have been introduced by almost all major stock exchanges. The introduction of limit order books has significantly changed trading strategies as the speed of trading has increased dramatically and traders have the choice between different order types, which automatically imposes the question which of them should be used and under which conditions. This represents a large amount of electronic financial data that can be stored and processed in order to exploit underlying patterns. Financial institutions are using this data to create advantage for them on the market. One of the applications is the creation of automated trading strategies that use these patterns to trade with competitive edge, which is the objective of our project.

The data we are using is Order Book Data from Microsoft for 42 consecutive trading days. We are using Genetic Algorithm as machine learning technique in order to train our algorithm to be able to detect certain patterns that are commonly used as algorithmic strategies. These patterns are based on technical stock analysis, and can be fully explained by the market psychology. For our purposes, we have split the data in 2 sets, training set (30 trading days) and evaluation set (12 days). By doing so, we were able to find certain parameters that characterize those patterns, and later evaluate them in the simulation of real time trading. For each of the implemented strategies, Double Bottom, Double Top, Rectangle and Head and Shoulders, we obtained profit in the evaluation phase.

1. Introduction

1.1. Motivation

The financial market, and the way financial assets are traded have been revolutionized. The major technological changes include investors that are using computers in order to automate their trading process, and reorganization of the markets by becoming virtually electronic limit order books. Algorithmic trading is thus encouraged by the speed and the quality of the access to such markets. These algorithmic trading strategies are computer algorithms that are able to automatically make trading decisions, submit orders, and manage the orders after submission.[1] Naturally, it is an advantage to have an algorithm that is both fast and able to make the correct decisions. Making the correct decisions depends heavily on the way the algorithm is able to predict future price. Hence, our goal is to explore past order book data, in order to find frequent patterns with which we can train our algorithm, so that it makes correct decisions in order to gain profit. We then want to test it on real time simulation to see how correct these decisions actually were.

[1] Hendershott, T. "Algorithmic Trading and Information - Faculty & Research." 2011. <http://faculty.haas.berkeley.edu/hender/ATInformation.pdf>

1.2. Data

An order book is the list of orders that a stock exchange uses to record the interests of buyers and sellers. An engine is used to determine which orders can be fulfilled, i.e. which trades can be executed.[2]

For our project, we are using Tick Data derived from Order Book Data. Due to the high price needed to obtain Order Book Data, we were limited in terms of data that we can use. We managed to find datasets containing Order Book Data from Microsoft for 42 consecutive trading days, which we obtained from the website www.tradingphysics.com. The order book represents the data from the NASDAQ stock exchange. The format of the data that we have retrieved is described below:

Column	Description
Timestamp	Milliseconds after midnight
Ticker	Stock identifier
Order	Unique order ID
T	Message type. Allowed values: ● "B" - Add buy order ● "S" - Add sell order ● "E" - Execute outstanding order in part ● "C" - Cancel outstanding order in part ● "F" - Execute outstanding order in full ● "D" - Delete outstanding order in full ● "X" - Bulk volume for the cross event ● "T" - Execute non-displayed order
Shares	The quantity for the number of shares for the order for all

[2] Order Book Definition | Investopedia. Retrieved May 15, 2014, from http://www.investopedia.com/terms/o/order-book.asp.

	message types, except 0 for the message types “F” and “D”
Price	The order price, available for the message types “B”, “S”, “X”, “T”, 0 for cancellation and execution. The last 4 digits are decimal digits. The number needs to be divided with 10000 to convert into the currency value
MPID	The ID for the market participant issuing the order

Table 1. Format of the order book data

1.2.1. Preprocessing of data

In order to use the data, we needed to preprocess it and transfer it to tick data. Tick data means executed transactions. We processed the downloaded data and created new data in the following format:

Column	Description
Tick ID	Unique Tick ID for the file
Timestamp	Milliseconds after midnight
T	Message type (not used)
Shares	Number of traded shares
Price	The order price, the last 4 digits are decimal digits. The number needs to be divided with 10000 to convert into the currency value

Table 2. Format of the tick data

1.3. Organisation of report (structure)

The report is organized as follows. In Section II we discuss different trading strategies that we implemented, while in Section III we present the genetic algorithm and an overview of its structure. Section IV first provides a description of the experimentation environment and then we present and discuss the results obtained. We finally conclude and discuss our plans for future work in Sections V and VI.

2. Algorithmic Strategies

Algorithmic trading is defined as the computerized execution of financial instruments, rather than having human interaction. This includes trading of stocks, bonds, currencies etc. As this type of trading comes with a lot of benefits, such as efficiency, anonymity, and reduced trading cost and risks, it has become a great area of research and investment for Wall Street.[3]

We implemented four trading strategies, Double Bottom, Double Top, Head and Shoulders, and Rectangle, which are among the most commonly used strategies. They are focused on finding a time series pattern and then acting upon it. All of these patterns are used in technical stock analysis. In order for these patterns to be successful, it is important to act only after they have developed, as the prediction of the upcoming trend is based on the formation of the complete pattern. The type of trading for each of these patterns can be distinguished by the risk and possible gain. For this project, we decided to use less aggressive trading, in order to minimize the risk, hence when describing the patterns below, we will focus on reacting upon them, according to such type of trading. All the figures for the implemented strategies given below, were created using our collected data and our own algorithm.

2.1. Double Bottom

Double bottom is a reversal pattern, which means it is used to catch trend reversal.[4] This pattern describes the drop of the stock, a rebound (rise) and then another drop and then another rebound, hence it looks like the letter “W”.[5] It is formed after an extended downtrend. This looks obvious in the hindsight, but in real life it is really difficult to detect while trading. Acting upon it means that you should buy shortly after the second bottom is found, as when this happens it is expected that the price will rise.

However, it is also possible for the stock price to drop as well. In order not to lose too much if the price continues to drop, we need to set a low price to protect how much we lose. We do something similar in order to know where to sell. This is done by setting the high price with summing up the current buying price with a percentage of the average difference between the middle top and the two bottoms. The percentage can be chosen depending of the risk of the trading that we want to practice. For example, if we want to earn more money with higher risk, we will setup the percentage to be high. This is how one iteration of this strategy looks like. The process starts from the beginning after we sell the shares.

[3] Kissell, Robert L. "Algorithmic trading strategies." (2006).
[4] Philip Thygesen. "Double Bottom - Stock Market Strategy." 2013. 15 May. 2014 <http://www.stock-market-strategy.com/double-bottom/>
[5] "Double Bottom Definition | Investopedia." 2002. 15 May. 2014 <http://www.investopedia.com/terms/d/doublebottom.asp>

Figure 1. Double Bottom pattern

To detect this pattern (strategy) for our Genetic Algorithm we have a chromosome that is consisted of 5 parameters (genes). The first parameter is the one that represents the difference from the first bottom to the top, the second one represents the difference from the top to the second bottom. The third parameter is the percentage for calculating the higher price, and the fourth is the percentage for calculating the lower price. The last, fifth parameter represents the established trend strength - before the pattern is detected, downtrend has to be established and its strength has to correspond to the value of this gene.

Figure 2. Double Bottom chromosome

2.2. Double Top

This pattern is the opposite of the previously described Double Bottom. It consists of a rise of the stock, a drop and then another rise. Therefore, it looks like the letter “M”. As the

previous pattern, it is really obvious by looking backwards, but it is difficult to detect while trading. Acting upon this strategy means that you should sell shortly after the second top is found. When this happens it is expected that the price will drop. After we sell, we set target price for buying the next shares so we can start the strategy from the beginning.[6]

Figure 3. Double Top pattern

To detect this pattern (strategy) for our Genetic Algorithm we have a chromosome that consists of 5 parameters. The first parameter is the one that represents the difference from the first top to the bottom, whereas the second one represents the difference from the bottom to the second top. The third parameter is the percentage for calculating the higher price for buying, and the fourth is the percentage for calculating the lower price for buying. The last, fifth parameter represents the established trend strength - before the pattern is detected, uptrend has to be established and its strength has to correspond to the value of this gene.

Figure 4. Double Top chromosome

[6] "Double Top Definition | Investopedia." 2002. 15 May. 2014 <http://www.investopedia.com/terms/d/doubletop.asp>

2.3. Head and Shoulders

Head and Shoulders is a reversal chart pattern that is formed after an extended price move to the upside.[7] The name of this pattern comes from its graphical representation. It looks like a head with shoulders on the left and right. It depicts a rise in the price of the share, followed by a drop, then a next rise which is higher than the previous, then followed by a second drop, followed by a third rise, reaching a top at the same level as the first rise. The pattern is finished with a last drop reaching the price of the previous two drops. Once this is found it is expected for the price to start decreasing, therefore we need to sell our stocks at this point and then buy new shares (the buy price is defined when we sell).[8] Once we buy shares, we start looking for this pattern from the beginning.

Figure 5. Head and Shoulders pattern

To detect this pattern (strategy) for our genetic algorithm we have 6 parameters. The first parameter depicts the minimum distance from the bottoms to the shoulder, the second one the minimum distance between the shoulders and the head, the third parameter depicts the maximum distance between the two bottoms. The fourth and the fifth are the parameters depicting where we need to buy once we have sold our shares, the last parameter is the one expressing the strength of the trend (upward). This parameter is used in order to decide whether we can even start looking for a pattern.

[7] Philip Thygesen. "Head and Shoulders - Stock Market Strategy." 2013. 15 May. 2014 <http://www.stock-market-strategy.com/head-and-shoulder/>

[8] "Head And Shoulders Pattern Definition | Investopedia." 15 May. 2014 <http://www.investopedia.com/terms/h/head-shoulders.asp>

Min Distance Bottom Shoulder	Min Distance Shoulder Head	Max Distance Two Bottoms	Protect Buy Gain	Protect Buy Loss	Trend Strength

Figure 6. Head and Shoulders chromosome

2.4. Rectangle

The Rectangle is a continuation pattern that forms as a trading range during a pause in the trend. The pattern is easily identifiable by two comparable highs and two comparable lows. The highs and the lows can be connected to form two parallel lines that make up the top and bottom of a rectangle.[9] Once this pattern is formed we act upon it with buying shares, since we expect to have a rise in the price. Similar as in the Double Bottom we set stops for where to sell the shares. Once we sell the shares we have, we start looking for the pattern from the beginning.

Figure 7. Rectangle pattern

To detect this pattern (strategy) for our genetic algorithm we have 5 parameters. The first parameter represents the difference between equal levels - highs and lows, while the second one represents the difference between same levels - between two highs (lows). The

[9] "Rectangle Definition | Investopedia." 2006. 15 May. 2014 <http://www.investopedia.com/terms/r/rectangle.asp>

third parameter is the percentage for calculating the higher price, and the fourth is the percentage for calculating the lower price. The last, fifth parameter represents the established trend strength - before the pattern is detected, uptrend has to be established and its strength has to correspond to the value of this gene.

Figure 8. Rectangle chromosome

3. Genetic Algorithm

Genetic algorithms (GAs) are adaptive methods which can be used for solving optimisation and search problems. As their name suggests, genetic algorithms are based on the genetic processes of real biological organisms. As in nature, populations evolve according to the principles of natural selection, also known as “survival of the fittest”. By this, genetic algorithms, if being suitably encoded, are able to “evolve” solutions to real world problems.[10]

Genetic algorithms work with a population of “individuals”, represented by chromosomes, which are a possible solution to the given problem. In the process of “evolving”, each such chromosome is assigned a fitness score, that represents how good the chromosome is as a solution to the problem. The individuals with higher fitness are then more likely to be selected for reproducing, which is represented by the crossover phase, after which the obtained offsprings, i.e. their chromosomes can be mutated in the mutation phase, as it is common in nature. Since the evolving of populations in genetic algorithms, works as natural selection, the final solution should be a chromosome with high fitness score, meaning a suitable solution to the problem. This is based on the fact that chromosomes with higher fitness value are more likely to be chosen for reproduction, whereas the other ones simply die out.

For the purposes of our project, genetic algorithm is used in order to find such fittest individual (chromosome) that can be further used in real-time to identify correct patterns so that we could act upon them. What we are interested in is finding parameters for each of the aforementioned strategies, which are tuned in a way that they are not too large, such that it is realistic to detect patterns with those parameters. Hence, it is very important to create a suitable fitness function to evaluate how good a chromosome is. Additionally, it is also important to set up the constants that are part of the genetic algorithm (Table 3).

Constant	Description
NUM_OF_GENERATIONS	Number of generations of the individuals (chromosomes)

[10] Busetti, F. "Genetic algorithms overview - GEOCITIES.ws." 2010. <http://www.geocities.ws/francorbusetti/gaweb.pdf>

	which will go through selection, crossover and mutation phase after which the solution (best chromosome of the last generation) is obtained
NUM_OF_CHROMOSOMES	Number of chromosomes in the population
SELECTIVITY	The percentage of chromosomes that survive the selection phase
NUM_ELITE	Number of chromosomes with highest fitness value that certainly survive the selection phase
CROSSOVER_PROBABILITY	The percentage of chromosomes in the population that are chosen for breeding
MUTATION_PROBABILITY	The percentage of genes in all chromosomes of the population that are mutated

Table 3. Constants of the Genetic Algorithm

The first population is generated randomly. After the execution of all phases for the number of generations we set, the fittest chromosome is obtained. The process is exactly the same for each strategy. The output and the flow of the genetic algorithm are shown on Figure 9.

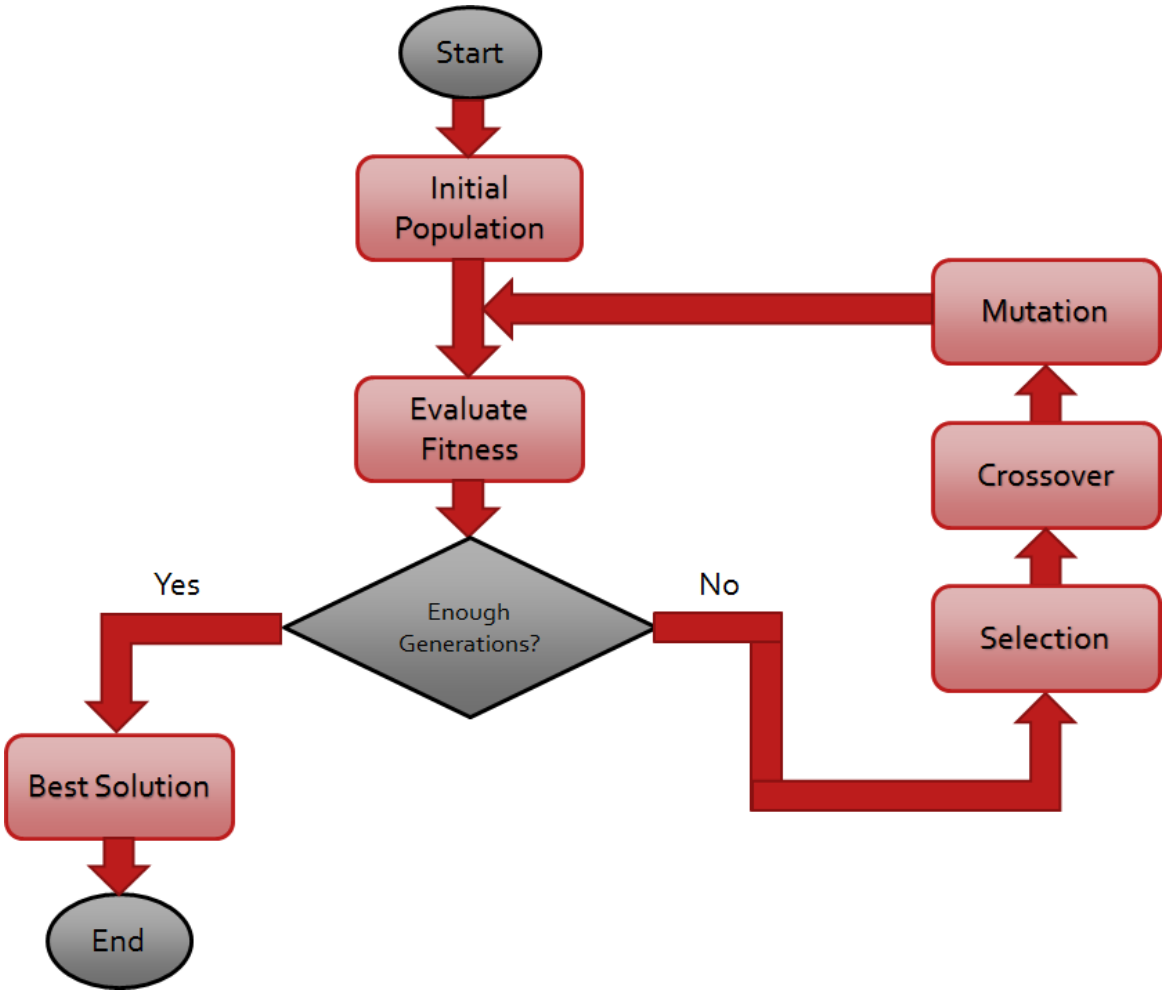

Figure 9. Genetic Algorithm process

3.1. Why choose GA for this project?

There are a lot of machine learning techniques which can be used for algorithmic trading strategies. Among those, we chose Genetic Algorithm for several reasons. Besides it being very successful in identifying graphical patterns, there are additional advantages such as the following:

- It is very useful for complex, and not well defined problems
- It easily solves problems with multiple solutions
- Any bad solutions do not affect the end solution in a negative way, as they are simply being discarded
- It is simple to understand and implement
- It works by its own internal rules, hence does not need to know any rules of the problem
- Data from recent history have more impact on the final result without any additional cost.

However, it also has certain drawbacks, such as that there is no guarantee that it will find the global optimal solution, and has longer running time.[11] As the advantages for using genetic algorithms outweigh the shortcomings, it was a natural choice for our project.

3.2. Selection

Selection is the stage of genetic algorithm in which individuals (chromosomes) are chosen from a population for later breeding (recombination or crossover).[12] The number of the individuals that survive this phase is determined by the previously mentioned constant SELECTIVITY. *Elitism* or *elitist selection* is frequently used in genetic algorithms and it means that certain number of the best individuals in a generation is retained unchanged in the next generation.[12] We included the elitism in our version of GA and it works in combination with one of the two selection methods we implemented *Roulette Wheel Selection* and *Rank Selection*.

3.2.1. Roulette Wheel Selection

Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a roulette wheel where all chromosomes in the population are placed, every has its place big accordingly to its fitness function, like on the following picture.[13]

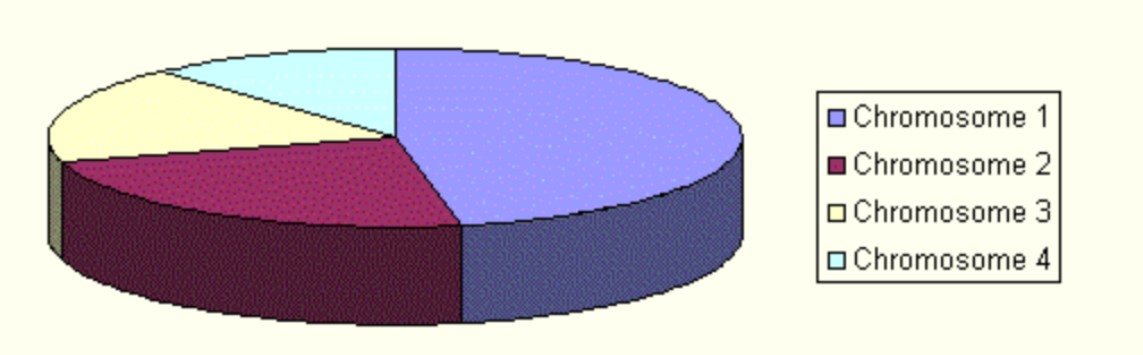

Figure 10. Roulette wheel selection[13]

Then a marble is thrown there and the chromosome is selected. This step is repeated until enough chromosomes are selected (SELECTIVITY * population_size). Chromosomes with bigger fitness will be selected more times.[13]

3.2.2. Rank Selection

[11] Albarran, DG. "Combined Pattern Recognition and Genetic Algorithms for ..." 2013. <https://fenix.tecnico.ulisboa.pt/downloadFile/2589873946364/Relatorio%20da%20Dissertacao50975.pdf>

[12] Selection (Genetic Algorithm)

[12]<http://en.wikipedia.org/wiki/Selection_%28genetic_algorithm%29>

[13] <http://www.obitko.com/tutorials/genetic-algorithms/selection.php>

The only difference from the previously described method is in assigning the parts of the wheel to chromosomes. The previous selection will have problems when the fitnesses differ very much. For example, if the best chromosome fitness is 90% of all the roulette wheel then the other chromosomes will have very few chances to be selected. This is why Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2, etc. and the best will have fitness N - number of chromosomes in the population. This way, all chromosomes have a chance to be selected. However, this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones.[13]

3.3. Crossover

Crossover is the stage in genetic algorithm used to vary the chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. Crossover is a process of taking two chromosomes as parents and producing a new chromosome (offspring) from them.[14] There are different crossover methods and we implemented *Single Point Crossover*, *Two Point Crossover* and *Uniform Crossover*.

3.3.1. Single Point Crossover

In this type of crossover, one crossover point is selected. Till this point the genes are copied from the first parent, then the rest of the genes are copied from the second parent (Figure 11).[15]

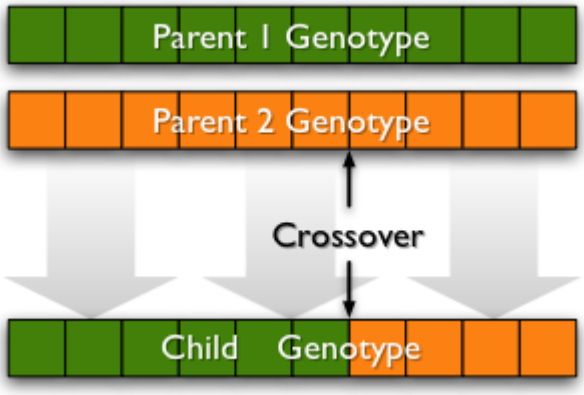

Figure 11. Single point crossover[16]

3.3.2. Two Point Crossover

[14] <http://en.wikipedia.org/wiki/Crossover_%28genetic_algorithm%29>
[15] <http://www.obitko.com/tutorials/genetic-algorithms/crossover-mutation.php>
[16] http://web.arch.usyd.edu.au/~rob/applets/house/House2.html

In *Two Point Crossover*, two crossover points are selected. The genes are copied from the first parent until the first point, then the genes between the two points are copied from the second parent and finally the genes after the second crossover point are again copied from the first parent (Figure 12).[15]

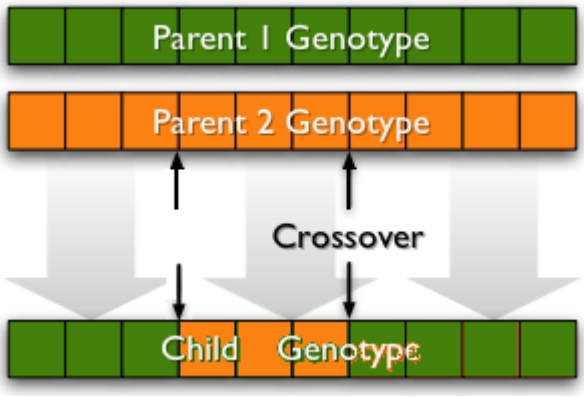

Figure 12. Two point crossover

3.3.3. Uniform Crossover

In *Uniform Crossover*, genes are randomly copied from the first or from the second parent (Figure 13).[15]

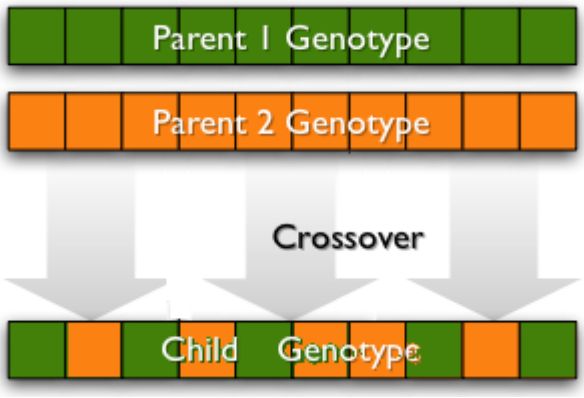

Figure 13. Uniform crossover

3.4. Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to better solution by using mutation. Mutation occurs during evolution according to previously mentioned

MUTATION_PROBABILITY. This probability should be set low (≤0.05). If it is set too high, the search will turn into a primitive random search.[17] We have implemented *Uniform Mutation* in our GA.

3.4.1. Uniform Mutation

This operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.[17]

3.5. Fitness function

A fitness function must be devised for each problem to be solved. Given a particular chromosome, the fitness function returns a single numerical "fitness" which is supposed to be proportional to the "utility" or "ability" of the individual which that chromosome represents. For many problems, particularly function optimisation, the fitness function should simply measure the value of the function.[18]

In our case, fitness function is implemented for each strategy (Double Bottom, Double Top, Head and Shoulders, Rectangle) and returns the amount of money that chromosome will make on a given set of transactions.

3.6. Tuning parameters and genes

In order to get the most out of our Genetic Algorithm, we were trying out

- different methods for selection, crossover
- different values for the GA constants (Table 3)
- binary representation of the genes - in this case, mutation and crossover are not on “gene-level”, but on “bit-level”.

[17] <http://en.wikipedia.org/wiki/Mutation_%28genetic_algorithm%29>
[18] Franco Busetti “Genetic algorithms overview”
[18]<http://www.geocities.ws/francorbusetti/gaweb.pdf>

4. Implementation (Application Flow)

4.1. Organisation of implementation

The organization of our implementation is split in three parts. The first part is the Genetic Algorithm itself. We have already explained how it works in the previous section. The second part is the training phase over the data that we have chosen for training, and the last part is the evaluation phase over the data that we have chosen for evaluation (different than the data for training). In order to test certain strategy, one needs to extend the Fitness Function explained earlier and implement the calculate fitness method in his/her own way. Before starting with the training and evaluation, the parameters for the genetic algorithm have to be defined. After the GA parameters are defined by the user, the data has to be split in training and evaluation data. When everything is setup, one can run the algorithm which will find the parameters for the strategy that is being trained.

Figure 14. Application flow

4.2. Training

The training part of the algorithm is executed in such way that each generation is trained over a set of days (window for generation training). In our case, the window contains 5 consecutive days. This window is moved after certain number of generations have been trained over it, for us this number is 4. More precisely, after every 4th generation we move the window for training. The total number of days for training for our project is 30 days. In the setup of the training part, we also define the starting amount of money for the strategy, for us this is 3000 dollars. After the training is finished we output the best chromosome and start the evaluation phase with the parameters derived from the training. We can see on figures 15, 16 and 17 how one chromosome is evaluated. In each strategy we have a pattern detection phase, which is different for each strategy. After the pattern is detected, we have a phase in which we act (buy => sell or sell => buy).

4.3. Real time simulation - evaluation

The evaluation part of the algorithm is the part where we test the parameters derived from the training phase. The evaluation part can also be a real time simulation. This is possible because of the way the data is received - it doesn't matter if the data is read from a file or read from somewhere else. While we receive data we try to detect the desired pattern and act upon it. One evaluation for the Double Bottom strategy is depicted on the pictures below. When we start the evaluation (trading) we check if we are in an open position (this means that we have bought shares and now we need to sell them). If we are in an open position we check if we can sell or not, otherwise we enter the pattern detection phase where we look for the pattern (in this case for Double Bottom).

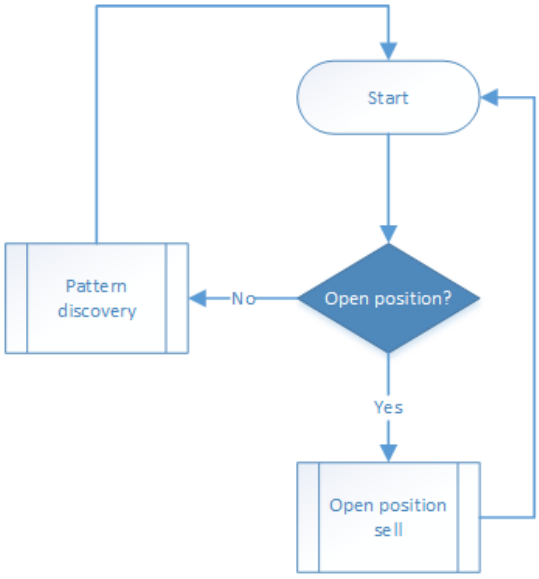

Figure 15. Main for trading

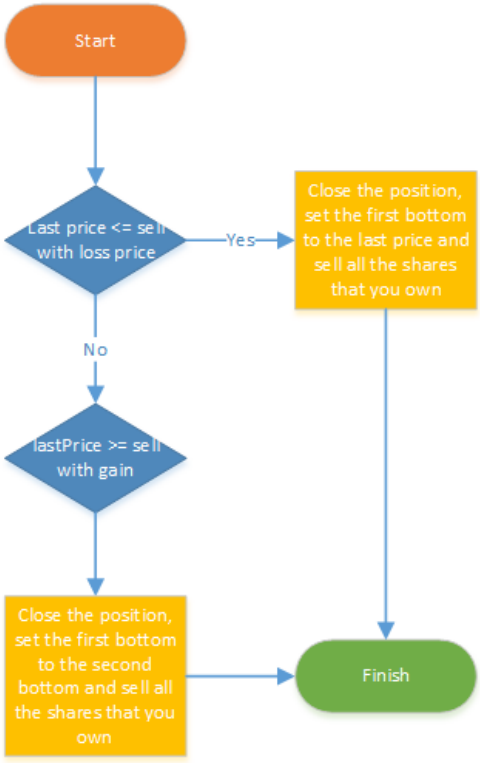

Figure 16. Flow for open position

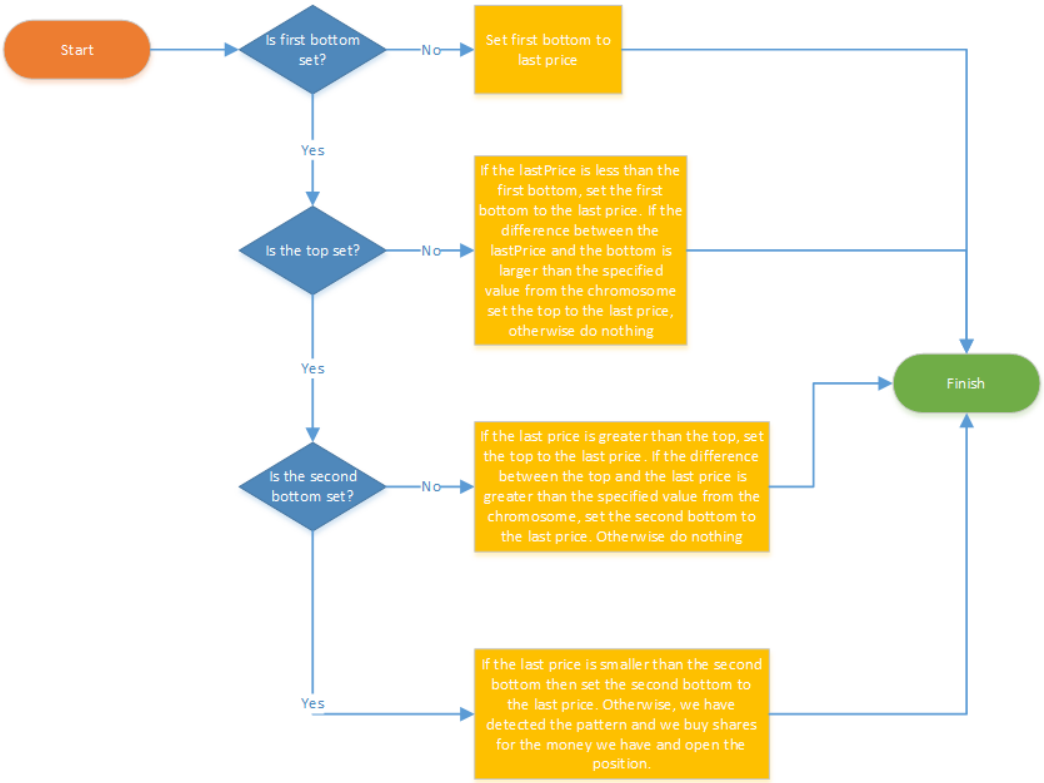

Figure 17. Flow of the pattern detection

4.4. Discussion of results

In the evaluation of our algorithm for the strategies that we have implemented we observed the following results. We have retrained and evaluated each strategy 10 times. Every training had the following parameters:

- Starting number of chromosomes was 100
- Training was performed on 30 consecutive days
- 100 generations of chromosomes were created
- Each generation was evaluated over a window of 4 consecutive days
- After every 4th generation the window was moved for one day
- Starting amount of money was 3000 dollars

When each training phase finishes, we start the evaluation of the best chromosome over the rest of the data (12 consecutive days). In average for every strategy we received the following results (The average is calculated over 10 runs):

- Double Bottom - amount 3125, number of transactions 139.8
- Double Top - amount 3378, number of transactions 109.8
- Rectangle - amount 3078, number of transactions 69.2
- Head and Shoulders - amount 3240, number of transactions 6
- Random - amount 17, number of transactions 318.3

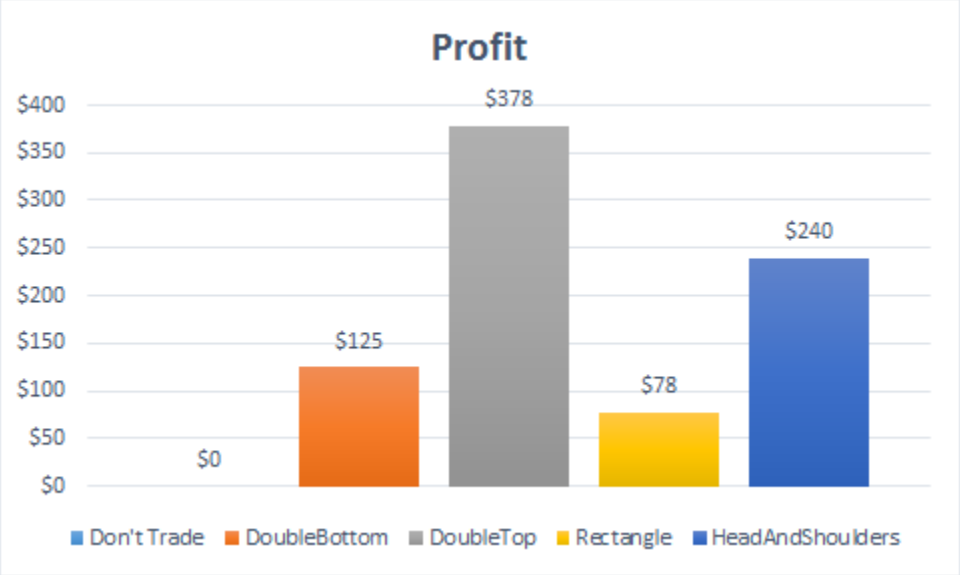

Figure 18. Average profit after trading for 12 days

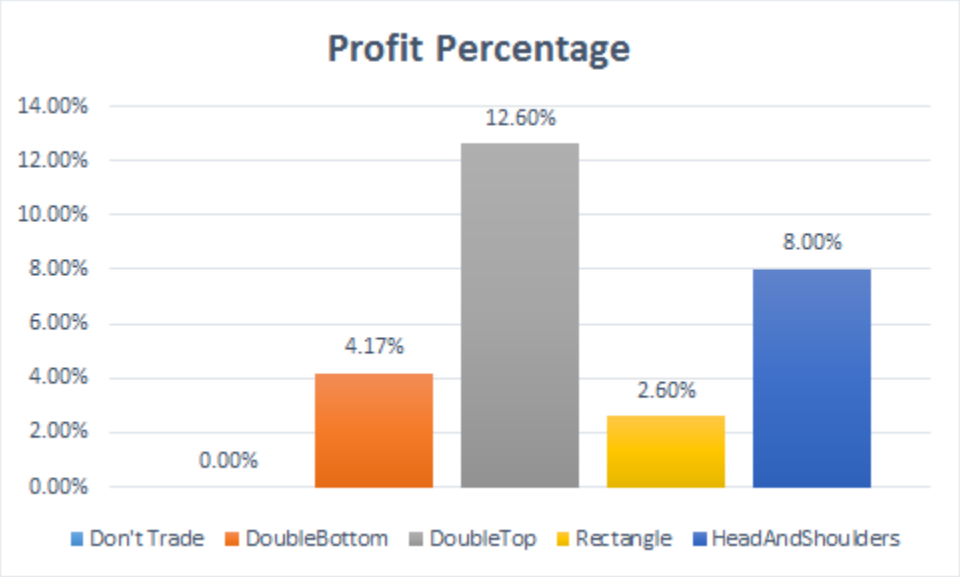

Figure 19. Average profit percentage after trading for 12 days

4.4.1. Random

We created additional algorithm in order to test the performance of the previously implemented trading strategies against it. Our idea for this algorithm was not to implement any particular strategy, but to make decisions for buying and selling based on some probability. The probability we used was tuned such that it makes as many transactions, as the average number of the transactions made by the other implemented strategies. The reason for such algorithm was based on the interest to test whether we gain profit with the other strategies due to the patterns they implement, or if it is also possible to obtain profit on average, by random decisions.

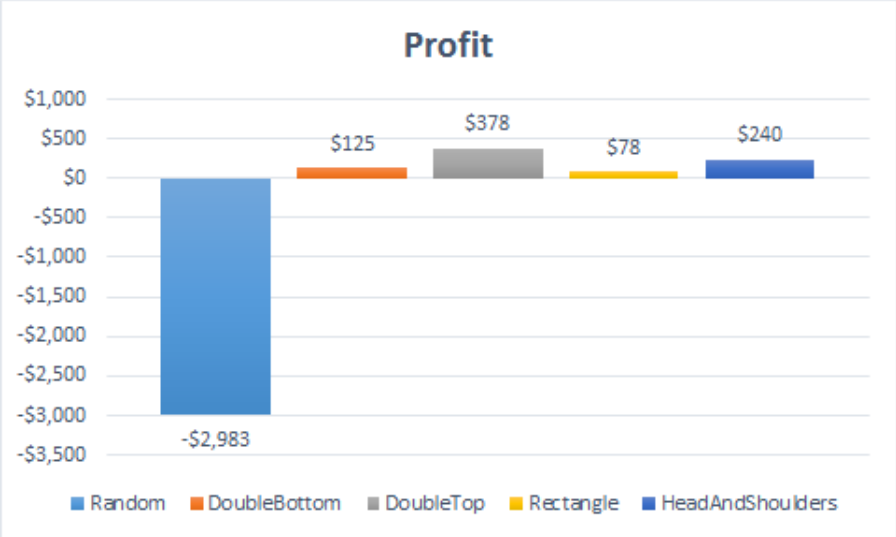

Figure 20. Average profit after trading for 12 days

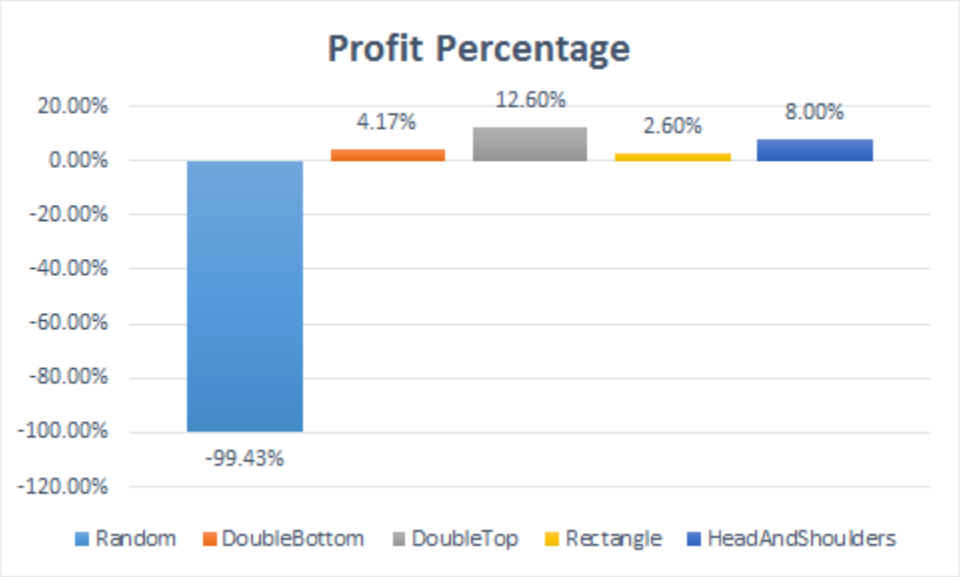

Figure 21. Average profit percentage after trading for 12 days

5. Conclusion

As the area of algorithmic trading is gaining more and more attention, order book data which is the list of orders that a stock exchange uses to record the interests of buyers and sellers, is a great source of data that can be exploit. For this project, we divided our dataset (order book data for Microsoft for 42 consecutive days) into two sets, one for training and one for evaluation, respectively. For the training part, we used genetic algorithms in order to obtain the fittest chromosomes with parameters defining the patterns for each of the implemented trading strategies, Double Bottom, Double Top, Rectangle and Head and Shoulders.

In the evaluation phase, we used each of those chromosomes in a real time simulation in order to evaluate how well they perform on unseen data. The results show that using each of those strategies, we obtained profit, with the highest profit of ~13% of the given starting amount. Furthermore, the results were compared with the results of a random algorithm that buys and sells with a certain probability, such that it makes transactions that are the average number of transactions made by the other implemented algorithmic strategies. The comparison shows that the random algorithm loses on average, and it performs much worse than the other strategies.

It follows that the implementation of these four strategies, and their combination with genetic algorithms, is a promising starting point for investigating algorithmic trading strategies for real life implementation. Although our implementation is not perfectly realistic, as it makes some simplified assumptions, it is able to show that Double Bottom, Double Top, Rectangle and Head and Shoulders perform well in predicting the future trend. Furthermore, genetic algorithms turned out to be a good choice for this project.

6. Future Work

The genetic algorithm we are using may be further improved. For example, we can do so by smartly choosing the initial search area, like in a quantum-inspired evolutionary algorithm. Furthermore, we can try out new genes for our strategies (for example by tracking volatility), which may lead to additional improvement.

Also, some of our assumptions (the reaction time after a transaction, the commision fee, etc.) should be tested, and improved so it is closer to reality. Another thing which should be taken into account is the influence of our transactions on the market. Finally, we can work on finding a way to combine the strategies we implemented into a new complex strategy.