Self Adaptive Voting System for Stock Market Investment Strategy based on Evolutionary Computing

Gonçalo Capela

Abstract— Financial market movements are highly influenced by complex factors that make it difficult to profit from these movements. Investors often use strategies that help them determine when to buy or sell stocks. Technical rules have been widely used in financial markets for more than a century as analytical tools to assess the safety of a given investment. This work describes an application, based on an evolutionary computation technique, in particular, Genetic Algorithms, and aims to generate entry and exit points of investments using technical analysis indicators (EMA, RSI, MACD, among others). In order to validate the proposed solution, an exhaustive evaluation is defined, comparing the strategy developed with other investment methodologies, such as Buy & Hold. The different time horizons applied in the evaluation allows testing the solution under different market conditions, including the most recent financial crash. The results are promising, as the current solution can overcome the homologous strategies during the crash. The most extensive case study resulted in a return on investment 55 times higher than the Buy & Hold strategy.


I. INTRODUCTION

INVESTING in this market is one of the significant investment options for investors to make a high profit. However, the high return rates are associated with high-risk investments. It is very common for stock prices to fluctuate, causing doubts about the investment options, making it difficult for investors to decide on when to buy or sell particular stocks to maximize returns. Investors often use strategies that help them determine when to buy or sell stocks. Many strategies can be found in the literature, and many more strategies are developed, tailored so that they can be used to assess markets.

Optimizing the investments in the stock market is of extreme importance for investors since they manage to increase the return on investment (ROI). Increasing profits will most often also increase the risk of the investment. To counter this risk is possible to develop some rules that support the investor in the decisions to be made. However, these rules are not always easy to compose, much less to optimize. This process of creating strategies can be simplified by using fewer data and fewer indicators, but in doing so may result in losing some of the information. To make this process friendlier to the investor is possible to automate the creation and optimization of the rules, giving the user buy or sell signals.

This work is part of the development of an application for the support of the investor in decision-making on the stock exchange. The module to be developed is one capable of generating and optimizing technical rules and generating buy or sell signals considering the historical data from a specified stock. These buy and sell signals will be generated from several technical indicators/rules using a voting system (rewarding indicators with good performance). To do this it is essential to identify which indicators are consistently resulting in good return rates, so it is possible to reward those.

The presented paper provides a detailed discussion on a new approach for intelligent investment strategies. The paper is structured as follows: In section 2 it is given a brief overview of different methodologies which can be used to address the investment strategies problematic. Section 3 illustrates the system architecture. Section 4 proposes the validation procedure used to evaluate the developed strategy. Section 5 summarizes the provided document and supplies the particular conclusion.

II. RELATED WORK

In recent years there have been numerous attempts to improve the profit of financial trading. In these attempts, all kind of machine learning tools has been used such as Neural Networks, Genetic Algorithms, Support Vector Machines, and others.

The solution to this problem can be divided into different parts of data processing so that the solution will incorporate different submodules, each with different functionalities. The first phase, data capture will be responsible for acquiring the data and compute some indicators from them if necessary. Next will be the generation of buy or sell signals with Evolutionary algorithm. Finally, a module to evaluate and simulate the resulting strategy (also to be able to implement a reward-based system).

A. General Architecture

The architecture illustrated in Figure 1 is the base model for most of the applications explored in this section. Starting with data intake and processing, followed by the optimization.
process alongside the simulator and ending with outputting the results.

For instance, Gorgulho, Neves, and Horta (Gorgulho, Neves, and Horta 2011) proposed a traditional layer architecture composed of three distinct layers to solve the portfolio management and technical analysis optimization. These three layers consist of Presentation Layer, Business Logic Layer, and Data layer, and each layer is composed of several modules.

With a more detailed analysis, several works (Gorgulho, Neves, and Horta 2011; Pinto, Neves, and Horta 2015; Silva, Neves, and Horta 2015) propose very similar architectures with minor differences. All these works try to optimize in some way the strategies based on technical indicator rules taking as inputs the user settings and the historical prices of the stock to be optimized. The optimization process of these works is done based on technical rules, and as expected all of them have technical rules modules, also an investment simulator as a way to find the performance of each strategy.

### B. Data Processing

Whether optimizing the parameters of the technical indicators or optimizing strategies based on technical indicators a common module is the admission of data and their respective processing, such as the computation of the technical indicators and their rules.

The work (Gorgulho, Neves, and Horta 2011) proposes a solution for the data intake process, and it describes an architecture where the data layer is divided into two modules, the financial data processing module and the technical rules module. The financial data processing module e is accountable for processing all the financial data downloading a complete history of all the available data on distinct markets. All the financial data relative to the former index was downloaded through the Yahoo Finance Database. In the technical rules module, several technical indicators are computed In order to be used in the optimizations process.

### C. Optimization / Machine Learning

Several works have been done using Neural Networks to improve return on the financial market. For example (Boonpeng and Jeatrakul 2016) compared several multi-binary classification techniques using neural networks. They used One-Against-One (OAO) and One-Against-All (OAA) techniques and compared them with the traditional neural network. Results show that OAA-NN outperforms OAO-NN and the multi-class classification using a single NN.

Qinghua Huang et al. (Huang et al. 2015) propose a solution to mine trading rules from trading data. The authors used a biclustering algorithm and the k nearest neighbor to mine patterns as trading rules and classified as three trading rules (buy, sell and no-action). It was implemented on four historical datasets, and then the results were compared with the conventional buy-and-hold strategy. The experimental results demonstrate that the proposed trading system outperforms the buy-and-hold strategy.

In the study of S. Boonpeng and P. Jeatrakul (Boonpeng and Jeatrakul 2016), several classifiers were created, and the classification is made by combining the results generated by the classifiers. Bootstrap aggregating is one of the ensemble learning methods, which creates several classifiers using bootstrapping samples, and makes classification by majority vote. To generate the rules a genetic network is used. After this, all rules are stored, and a voting method is used to generate the sell or buy signal. The weights of the weighted voting system are the output of a Multi-layer perceptron. After the multi-layer generates the vote weight of each rule the buy or sell rule is easily calculated.

Iskrich and Grigoriev (Iskrich and Grigoriev 2017) suggested the use of an evolutionary algorithm to generate and select the most suitable trading rules for interday trading. For this, it was used binary decision trees whose leaf nodes contain decisions (buy or sell). To determine the best tree a genetic algorithm is used. The advantages of this approach are the simplicity of the generated rules, and the extensibility since the technical indicators are relative (from VERY LOW to VERY HIGH).

An intelligent hybrid system is proposed by Kim et al. (Kim et al. 2017) for discovering technical trading rules using a genetic algorithm. The first phase of analysis consists of generating a decision table. The second stage is the rule discovery mechanism that consists of two steps. First, extract rules using rough set analysis, and then applying GA to discovering optimal decision rules. These two steps are repeated until the stopping condition of the GA is met. The third stage is the trading signal generation from the generated rules. For this study, a six-month training period and a set of 50 decision rules provided the highest annualized return rate compared to other experimental combinations.

As described before A. Gorgulho et al. (Gorgulho, Neves, and Horta 2011) uses GA to optimize technical analysis rules and optimizing portfolio composition. J.M. Pinto et al. (Pinto, Neves, and Horta 2015) uses a Multi-Objective Evolutionary System to predict the future tendency of assets price and optimize a set of Trading or Investment Strategies.

Table 1 summarizes some of the most relevant existing solutions to approach the investment strategy optimization, specified according to several parameters.

### III. PROPOSED ARCHITECTURE

The purpose of this chapter is to describe the solution developed to generate buy or sell signals. It starts by giving an overview of the general architecture developed, and the various modules that make up the solution are also described below.

#### A. Overall Architecture

The developed application tries to find the best investment strategy for a specific stock (entry and exit points). In an initial phase, the data enters the application, and technical indicators are computed from data provided. Then there is a training phase, in which the optimal voting powers of each indicator are discovered. These voting powers are then inserted into a voting system, and a weighted average is calculated to find out entry or exit points. In this last phase, two different strategies are built, with different levels of risk and profit.
This way different types of investors can choose which strategy suits their taste.

The following diagrams propose the architecture of a possible system which tries to handle the optimization issue.

As we can see in the Figure 2, the system is constituted by different modules, with different functionalities and specializations. Each of the modules is responsible for a different phase throughout the process. It is also possible to identify four different modules whose responsibilities are explained below.

Technical Rules Module: Starts by reading the data from a file, it computes all the technical indicators and transform each indicator into entry and exit signals applying the corresponding technical rules.

Optimization Module: This is the heart of the system. It is responsible for finding the best set of weights/voting power for each indicator. It operates with a genetic algorithm optimizing the return on investment and the associated risk.

Voting-Based Decision Module: This module is responsible for creating the strategy to be tested according to the voting power of each technical indicator used.

Investment Simulator Module: This is the system that is responsible for simulating every strategy. It simulates the strategy given by the optimization module, computing the return on investment and the risk involved in each strategy.

B. Data Flow

In order to start the application, the user must define the inputs required to execute the optimization algorithm, the initial budget, the training, and execution window, the period, and others.

Afterward, the system computes the technical indicators, chosen by the user, and with these values, it computes buy and sell signal from each indicator according to the rules.

After this process, the GA starts its execution by defining several random individuals, which correspond to different voting powers for each technical rule.
In order to evaluate each individual, a Strategy Simulation module is needed. In this module, each individual is transformed in a strategy, and it is computed the Return on investment and Maximum Drawdown of each strategy/individual.

When the GA converges in a final solution, the system executes the investment simulation again, but to the current date period.

The diagram in Figure 3 illustrates the different stages of the optimization process. It is worth stressing that the simulator and the GA process run alongside because is the simulator that computes the fitness functions to evaluate each strategy. After the GA process is finished, the best strategy must be simulated again (but this time not in the training set) to yield the final results.

C. Technical Rules Module

This module is responsible for processing all the financial data. After the stock data is processed from the input files, the technical indicators must be computed and transformed in buy and sell signals for each indicator according to each rule.

1) Input Files

The first step of the process is to read and store the information about the stock prices. The input files must obey some rules, must be a “.csv” file, and with the following configuration:

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Adj. Close</th>
<th>Volume</th>
</tr>
</thead>
</table>

where:

- **Date**: the date of each instance, in the format “yyyy-mm-dd”
- **Open**: the opening price of a specific interval (day, month, year…)
- **High**: the highest price of a specific interval
- **Low**: the lowest price of a specific interval
- **Close**: the closing price of a specific date.
- **Adj. Close**: the adjusted closing price
- **Volume**: the number of shares traded in a security during a specific interval.

2) Implementation and Functionality

To use a technical indicator is necessary to define some parameters, such as the calculation period. For example, a simple indicator such as the moving average, which calculates the average of the last x days for each day. This indicator depends on the number of days x.

By setting rules for each indicator is possible to generate buy and sell signals. Analyzing which indicators generated the best buy and sell signals (with a higher return and lower risk) in the past is possible to attribute in the present and future more importance to the signals generated by indicators with a better score.

For each technical indicator calculated for each period (day, week, or month) in the data set, a score was assigned. Three distinct scores were used:

- **Buy Score**: Assigns 0 points, indicates a buy signal,
- **Sell Score**: Assigns -1.0 points, indicates a sell/short signal,
- **Out Score**: Assigns 0 points, indicates an out signal,
- **Strong Buy**: Assigns 1.0 points, indicates a buy signal,
- **Strong Sell**: Assigns -1.0 points, indicates a sell signal.

The voting system of this system needs several technical indicators to function. In this way, several different types of technical indicators were implemented, among them: Exponential Moving Average (EMA), the Simple Moving Average (SMA), the Relative Strength Index (RSI), the Moving Average Convergence Divergence (MACD).

D. Optimization Module

This module is the core of the application. It is this module that is responsible for finding the best set of weights and thresholds possible. In this chapter, the various steps of the multi-objective genetic algorithm are described.

As explained before, the goal of the optimization is to find several optimized solutions regarding the Return on Investment and the Maximum Drawdown. To do this, this module performs a multi-objective optimization. The algorithm chosen to perform this task was the NSGAII. The set of several optimized solutions is called the Pareto frontier. From this Pareto frontier two solutions are chosen, one with high risk and high profit, and other with low risk and low profit.

1) Chromosome Representation

The design of the chromosome and its parameters is by necessity specific to the problem to be solved. In the solution presented in this work, the chromosomes are represented in binary as strings of 0s and 1s. The number of bits for each parameter is defined by the user. The search space depends on each parameter. Here the Chromosome represents different weights (importance) given to each technical rule. In addition to the weights described, several thresholds are also encoded in the genome. These thresholds transform the numerical result of the weighted average into buy or sell signals. These signals are then needed to take long or short positions and even to know how much to invest at a certain point in time. Next, the structure of the genome used in this application is presented and described.

<table>
<thead>
<tr>
<th>1st Rule</th>
<th>…</th>
<th>Last Rule</th>
<th>Strong Buy Threshold</th>
<th>Buy Threshold</th>
<th>Sell Threshold</th>
<th>Strong Sell Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0;1]</td>
<td>[0;1]</td>
<td>[0;1]</td>
<td>[-1;1]</td>
<td>[-1;1]</td>
<td>[-1;1]</td>
<td>[-1;1]</td>
</tr>
</tbody>
</table>

In the representation above the search space for each parameter is presented as an interval, but, the representation is not continuous, so the values, that each parameter can take, depends on the number of bits used for each parameter.

2) Initialization

Unlike other steps, the initial population is generated only at the beginning of the algorithm, not repeating itself over the next generations. This population is generated randomly, allowing the full range of possible solutions. If there are repeated individuals, that is, if an individual already exists in the population, it will be ignored, and another will be created.
3) Selection
To select the individuals for the crossover and mutation operators a binary tournament selection procedure is used. First, the procedure selects two solutions of the population and then selects the better according to non-dominated sorting technique and crowding distance value. If both solutions are selected from the same front (same rank), the solution with the higher crowding distance is selected. This is called a Tournament Selection Operator.

4) Mutation
This process involves a probability, which defines the probability of each bit being changed. The method of implementing the mutation operator involves generating a random variable for each bit in a sequence (Flip – Bit). This random variable tells whether a particular bit will be modified or not.

5) Crossover
For this application, the crossover operator chosen was the uniform crossover. This method constructs a new chromosome (child) based on two chromosomes (parents). For each of the child bits, a random variable is generated that defines whether to inherit the bit from one of the parents or the other (Uniform crossover).

E. Voting-Based Decision Module
Using several technical indicators to build a strategy seems to be ideal because the technical indicators complement each other. Sometimes during certain periods, a technical indicator produces wrong signals and therefore should be penalized in the following uses, and on the other hand, indicators with good performance should be rewarded, and therefore their signals should be important in decision making.

This module is responsible for implementing the voting system for the indicators. It runs alongside the optimization module because this it handles a process needed to compute the fitness of each individual of the population. From the genome resulting from the optimization process, this module computes the trading rules for the next period. The way of calculating the buy/sell signals considers the weights and thresholds defined by the genome.

1) Implementation and Functionality
For every single period (day, week, month) a buy/sell must be computed. To do this, the indicators buy/sell signal (-1, 0 or 1) is multiplied by the corresponding weight/voting power, summed all together and then normalized. This calculation can be defined as:

\[
\text{normalized weighted average} = \frac{\sum_{i=1}^{N} w_i \times s_i}{\sum_{i=1}^{N} w_i}
\]  

(1)

where:

- \(w_i\) is the weight / voting power of the indicator \(i\);
- \(s_i\) is the signal for the indicator \(i\) (-1, 0 or 1);

\(N\) is the number of indicators.

Formula 1 always yields a resulting signal which is a number between -1 and 1. After this is computed, the resulting numerical signal must be converted into buy/sell signals. There are five different signals: Strong Sell, Sell, Out, Buy, Strong Buy. As explained before, the genome also includes four thresholds, and are these values that defined and transform the numerical signal in the five different signals.

Figure 4 – Thresholds

Figure 4 illustrate the process to finding the resulting strategy (Buy / Sell Signals) from the genome and the buy / sell signals from each indicator.

This module receives the buy/sell signals from each indicator (of the Application window) and the genome (optimized with data from the training window) and finds the resulting strategy for the application window. With the resulting strategy, it also computes the Return on Investment and Maximum Drawdown.

The first step is to apply the normalized weighted average formula on each day, according to the signals (from each indicator) for that same day, and the corresponding weight. This results in a normalized weighted average for each day. Then these values are compared with the thresholds and transformed into buy/sell signals.

The implementation of this module and the Investment Simulator Module was done in such way that all entries (buys) were done at the beginning of the period (Open price), and all exits (sell/shorts) were computed at the end of the period (Close price).

Let’s take the resulting strategy (buy/sell signals) of the Figure 5. The first signal is a “1” corresponding to a Strong Buy, this means that all available budget is invested in a long position on the beginning of that same period (Open price). The second signal is also a “1”, so no changes, the position is still long. On the third day we get a signal of “0.5” this means that only 50% of the available budget must be invested. Taking into account that until now all the budget is invested, to get to the 50% investment, half of the investment must be sold (this operation is computed with the closing price). On the fourth and fifth days a strong sell signal is present, this means that the simulator must sell the remaining 50% investment (long), and must take now a short position in the amount of all the budget available. In the last day, only half of the budget must be in a short position, so the simulator must close half of the investment made in the previous short operation, remaining 50% of the available budget in a short position.


**F. Investment Simulator Module**

In order to be able to evaluate each strategy associated with each individual, an investment simulator is necessary. The simulator applies the signals of buy and sell resulting from the voting system. Simulating the strategies also computes the return on investment and the Maximum Drawdown associated with the simulated strategy.

Several settings must be defined in the simulator depending on the preference of the investor.

**Budget:** The initial capital available to invest.

**Short Selling:** This parameter defines if the simulator can take short positions.

**Transaction Costs:** Used for the consideration of transaction costs. This parameter is used to include the commission costs involved on buying or selling shares. This number defines the percentage to be charged.

**Training window size:** Size of the period to find the optimum set of weights of the indicator rules.

**Application window size:** Size of the window in which apply the result of the previous optimization.

In Figure 6, in the first iteration, the two months immediately before January first are used to find the best set of weights. Then those same weights are applied to the first month (First application window).

After all the sliding windows are shifted the second iteration may start, finding the new best weights to the new sliding training window, and then applied again to the new application window.

**IV. Validation**

In this chapter, we describe the evaluation method to validate the developed system. To validate the application it was necessary to use a test strategy called Backtesting (Ni and Zhang 2005). This process consists of testing a specific strategy in an earlier period to determine its effectiveness. For example, suppose we want to evaluate our strategy against the year's performance. Instead of waiting a full year to do so, we can extract the data from the past and then evaluate the procedure in the periods considered. Applying the strategy developed to the previous data can be substantially beneficial in order to
detect strategy failures and improve their potential.

A. Classification Parameters

Besides the two presented measures used to evaluate the performance of the generated strategies, is possible to define a list of relevant parameters which can be used to classify the designed strategy:

- **Number of positions**: The number of positions taken in the investment within a specific period.
- **Percentage of profitable positions**: From the total number of positions made in a specific period, it will be essential to observe how many of them are profitable.
- **Percentage of non-profitable positions**: As to measure the number of winning positions, it is also desirable to determine the non-profitable ones.
- **Greatest profit**: It will be interesting to observe what was the most significant profit obtained within all the investment period.
- **Greatest loss**: As the highest profit, it is also essential to determine the most significant loss.
- **Average Profit**: Indicates the average profit considering all positions taken within a specific period.

B. Case studies

1) **Case study I - Dow Jones Industrial Average ETF (DIA) 2003/2009**

The presented case study exhibits the results obtained when evaluating the implemented strategies during the years of 2003 to 2009. For the designed case study, the following configuration was applied:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock</td>
<td>DIA</td>
</tr>
<tr>
<td>Period</td>
<td>02/01/2003 – 16/06/2009</td>
</tr>
<tr>
<td>Budget</td>
<td>100000 USD</td>
</tr>
<tr>
<td>Short Selling</td>
<td>TRUE</td>
</tr>
<tr>
<td>Commission Costs</td>
<td>0.1%</td>
</tr>
<tr>
<td>Number of executions</td>
<td>100</td>
</tr>
</tbody>
</table>

In respect to the evolutionary strategy, the following parameter specification was employed:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>50</td>
</tr>
<tr>
<td>Mutation Rate</td>
<td>10%</td>
</tr>
<tr>
<td>Crossover Rate</td>
<td>90%</td>
</tr>
<tr>
<td>Nº Generations</td>
<td>100</td>
</tr>
<tr>
<td>Sliding Window</td>
<td>125 days / 125 days</td>
</tr>
</tbody>
</table>

To highlight the superiority of the evolutionary strategy, on the end of the testing period, the histogram in Figure 8 demonstrates the ROI obtained for the different 100 executions experimented per each investment strategy.

The following table shows the performance of each strategy according to the parameters described in the first section of this chapter. 100 different executions were experimented to evaluate each methodology thoroughly. The results for those strategies correspond to the confidence interval achieved when using a confidence degree of 95%.
In the big crash start, B&H strategy, but in the last situation first five years a sideways market is dominant, in this situation, the voting system cannot get better results than the B&H strategy, but in the last three years of the simulation, when the big crash start, the voting system get much better ROI than the B&H.

In Figure 11 the evolution of the different strategies corresponding to each indicator used is shown. It is possible to note that these strategies almost follow the buy & hold.

In Figure 12 is possible to see the different stages of the sliding window, and on the table V the chromosomes for all periods is shown.

### Table V: Genome for Each Period

<table>
<thead>
<tr>
<th>H</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
<th>I6</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>0.67</td>
<td>0.33</td>
<td>1.00</td>
<td>0.67</td>
<td>0.33</td>
<td>-0.33</td>
<td>-0.33</td>
<td>0.33</td>
</tr>
<tr>
<td>2</td>
<td>0.33</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>-1.00</td>
<td>-1.00</td>
<td>-0.33</td>
</tr>
<tr>
<td>3</td>
<td>0.67</td>
<td>1.00</td>
<td>0.33</td>
<td>0.33</td>
<td>0.67</td>
<td>1.00</td>
<td>-1.00</td>
<td>-1.00</td>
<td>-0.33</td>
</tr>
<tr>
<td>4</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.67</td>
<td>-0.33</td>
<td>-0.33</td>
<td>0.33</td>
</tr>
<tr>
<td>5</td>
<td>0.00</td>
<td>0.67</td>
<td>1.00</td>
<td>0.67</td>
<td>0.33</td>
<td>0.67</td>
<td>-0.33</td>
<td>0.33</td>
<td>1.00</td>
</tr>
<tr>
<td>6</td>
<td>1.00</td>
<td>0.00</td>
<td>0.33</td>
<td>0.00</td>
<td>0.67</td>
<td>0.33</td>
<td>-1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>7</td>
<td>0.33</td>
<td>0.33</td>
<td>0.33</td>
<td>0.67</td>
<td>0.00</td>
<td>0.00</td>
<td>-1.00</td>
<td>0.33</td>
<td>0.33</td>
</tr>
<tr>
<td>8</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.67</td>
<td>0.67</td>
<td>0.00</td>
<td>-1.00</td>
<td>-0.33</td>
<td>0.33</td>
</tr>
<tr>
<td>9</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.33</td>
<td>-1.00</td>
<td>-0.33</td>
<td>0.33</td>
</tr>
<tr>
<td>10</td>
<td>0.33</td>
<td>1.00</td>
<td>0.67</td>
<td>0.33</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.33</td>
<td>-0.33</td>
<td>0.33</td>
</tr>
<tr>
<td>11</td>
<td>0.00</td>
<td>1.00</td>
<td>0.33</td>
<td>0.33</td>
<td>0.00</td>
<td>0.00</td>
<td>-1.00</td>
<td>0.33</td>
<td>1.00</td>
</tr>
<tr>
<td>12</td>
<td>0.67</td>
<td>0.67</td>
<td>1.00</td>
<td>1.00</td>
<td>0.67</td>
<td>0.33</td>
<td>-0.33</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>13</td>
<td>0.00</td>
<td>0.67</td>
<td>0.67</td>
<td>0.00</td>
<td>0.00</td>
<td>0.33</td>
<td>-0.33</td>
<td>0.33</td>
<td>0.33</td>
</tr>
</tbody>
</table>

![Figure 10 - Yearly ROI for the voting system and Buy & Hold](image)

In Figure 10, it is possible to understand where the voting system is robust and can beat the Buy & Hold strategy. In the first five years a sideways market is dominant, and in this situation, the voting system cannot get better results than the B&H strategy, but in the last three years of the simulation, when the big crash start, the voting system get much better ROI than the B&H.
In the table V:

1 – is MA20 voting power
2 – is MA40 voting power
3 – is MACD25,12 voting power
4 – is MACD50,20 voting power
5 – is RSI14 voting power
6 – is RSI28 voting power
T1 – is the Full Short Threshold
T2 – is the Half Short Threshold
T3 – is the Half Buy Threshold
T4 – is the Full Buy Threshold

Looking at Table V, the genome 12 is a solution that only allows for short positions (Half and Full). This is because T2 is 1.0 (Maximum value), knowing that the weighted average can only take values between -1 and 1. Is worth to note that the optimization for this individual was done in the 11th period, where is presented a situation that also takes the advantage from only is allowed short positions.

For the first part of the simulation where short positions are allowed, it is possible to see that in the beginning when it is presented a sideways market, the optimization algorithm does not perform better than the buy & hold strategy. However, when the market crashes the evolutionary strategy is capable of maintaining a reasonable profit without collapsing as the competitor strategies; being less risky and volatile, and subsequently obtaining a much higher ROI on the end of the testing period.

Although the evolutionary strategy tries to perform intelligent investment decisions, it has much more transaction costs when compared with the B&H approach, decreasing its profitability when the market is rising. In contrast, when the crash occurs, the intelligent investment decisions are more notorious which allow the strategy to pick the ideal positions while maintaining its profit.

For the second part of the experiment, where short selling is not allowed (Figure 14) it is evident that the ROI is not as high as in the first experiment, this is also obvious because we can see that the advantage of the evolutionary algorithm comes when the market crashes. In this simulation it is possible to see that simply instead of taking a short position, it just gets out of the market, avoiding the significant losses that the Buy & Hold cannot avoid.

2) Case study II - Bausch Health Companies Inc. (BHC) 2013/2015 (Valeant)

The presented case study exhibits the results obtained when evaluating the implemented strategies during the years of 2013 to 2015. For the designed case study, the following configuration was applied:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock</td>
<td>BHC</td>
</tr>
<tr>
<td>Period</td>
<td>02/01/2013 – 31/12/2015</td>
</tr>
<tr>
<td>Budget</td>
<td>100000 USD</td>
</tr>
<tr>
<td>Short Selling</td>
<td>TRUE</td>
</tr>
<tr>
<td>Commission Costs</td>
<td>0.1%</td>
</tr>
<tr>
<td>Number of executions</td>
<td>100</td>
</tr>
</tbody>
</table>

In respect to the evolutionary strategy, the parameters are the same as the previous study case.

Figure 13 - ROI Evolution for Case Study II

Figure 13 exhibits the results obtained for the considered strategies within the years of 2013 to 2015.

Each curve represents the return on investment achieved by the respective investment methodology.

Table VII shows the performance of each strategy according to the parameters described in the first section of this chapter. 100 different executions were experimented to evaluate each methodology thoroughly. The results for those strategies correspond to the confidence interval achieved when using a confidence degree of 95%.

<table>
<thead>
<tr>
<th>GA (HIGH RISK)</th>
<th>GA (LOW RISK)</th>
<th>BUY &amp; HOLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROI (%)</td>
<td>[236%, 262%]</td>
<td>[83%, 97%]</td>
</tr>
<tr>
<td>MAX DRAWDOWN</td>
<td>[-41%, -40%]</td>
<td>[-39%, -37%]</td>
</tr>
<tr>
<td>N' POSITIONS</td>
<td>[36, 38]</td>
<td>[79, 82]</td>
</tr>
<tr>
<td>PROFIT POS</td>
<td>[45%, 47%]</td>
<td>[40%, 41%]</td>
</tr>
<tr>
<td>NON-PROFIT POS</td>
<td>[51%, 53%]</td>
<td>[58%, 59%]</td>
</tr>
<tr>
<td>AVG PROFIT/POS</td>
<td>[4.4%, 4.8%]</td>
<td>[0.9%, 1.1%]</td>
</tr>
<tr>
<td>MAX PROFIT</td>
<td>[64%, 67%]</td>
<td>[38%, 40%]</td>
</tr>
<tr>
<td>MIN PROFIT</td>
<td>[-21%, -19%]</td>
<td>[-14%, -13%]</td>
</tr>
</tbody>
</table>
Here in this experiment the evolutionary strategy is put against a profitable buy & hold strategy. Although there is a significant drop near the middle of the simulation, the buy & hold strategy ends the investment with a 66% ROI. The B&H strategy cannot profit while the market is on a downtrend, this is where the evolutionary strategy gains against the B&H.

For the first part of the simulation where short positions are allowed, it is possible to see that in the beginning when it is present a bullish market, the optimization algorithm does not perform better than the buy & hold strategy. However, when the downtrend begins the evolutionary strategy is capable of maintaining a reasonable profit without collapsing as the competitor strategies; being less risky and volatile, and consequently obtaining a much higher ROI on the end of the testing period.

Although the evolutionary strategy tries to perform intelligent investment decisions, it has much more transaction costs when compared with the B&H approach, decreasing its profitability when the market is rising. In contrast, when the downtrend begins occurs, the intelligent investment decisions are more notorious which allow the strategy to pick the ideal positions while maintaining its profit.

V. Conclusion

The work presented proposes a potential strategy optimization system that combines several indicators of technical analysis with evolutionary computation algorithms to choose the best entry and exit points for various actions in the market. To evaluate the developed system and its strategies, it was compared with the market itself and several other investment methods. The results are promising, and much more can be done to improve them. The algorithm can be easily extended and parameterized. This chapter proposes several improvements to refine the current system.

The validation demonstrated in the previous chapter shows that the combination of EC and technical analysis indicators has potential. In this work, several investment methodologies were described, as well as different computational techniques to address the presented problem. While there is a prospect of using this application to manage an investment, human capabilities cannot be replaced entirely. So this application should be used while keeping a close eye on the financial market news to understand if there is any problem that the system is not aware of.

REFERENCES


