FEDERAL UNIVERSITY OF MINAS GERAIS
Institute of Exact Sciences
Graduate Program in Computer Science

Felipe Vital Cacique

PATTERN SEARCHER FOR DECISION MAKING OF TRADING AGENTS USING GENETIC ALGORITHM

Belo Horizonte
2020

Felipe Vital Cacique

PATTERN SEARCHER FOR DECISION MAKING OF TRADING AGENTS USING GENETIC ALGORITHM

Final version

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Adriano C®sar Machado Pereira

Belo Horizonte
2020

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

Cacique, Felipe Vital.

C119p Pattern searcher for decision making of trading agents using genetic algorithm / Felipe Vital Cacique. — Belo Horizonte, 2020.
xviii, 62 f. il.; 29 cm.

Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação
Orientador: Adriano César Machado Pereira.

1. Computação – Teses. 2. Algoritmos genéticos 3 Agentes de negociação. 4. Mercad. I. Orientador. II.Título.

CDU 519.6*82 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Pattern Searcher for Decision Making of Trading Agents using Genetic Algorithm

FELIPE VITAL CACIQUE

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. ADRIANO CÉSAR MACHADO PEREIRA - Orientador
Departamento de Ciência da Computação - UFMG

PROF. CRISTIANO ARBEX VALLE
Departamento de Ciência da Computação - UFMG

PROFA. GISELE LOBO PAPPA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 21 de Fevereiro de 2020.

Abstract

In the last few years, there was a growth regarding the use of computational methods in the field of finance, especially to negotiations in the stock market. Investors have been using computational tools to automate investment strategies with the goal of maximizing profits and reducing risks. In this work, we aim to bring new ideas and approaches to the development of automated trading robots based on historical data of financial series. Our model, named Pattern Searcher, was inspired in machine learning methods and evolutionary optimization. Given a trading agent with its predefined parameters, the method uses the power of Genetic Algorithm (GA) to search, within a set of financial indicators, for the region that provides a higher positive return. This implementation exhibited desirable properties compared to some Machine Learning methods, such as the simplification of the system flow and the generation of rules that humans can clearly understand. Besides, we have generated strategy portfolios, composed by the strategies derived from the Pattern Searcher method, that were also optimized via GA. The system was able to generate very profitable trading agents and portfolios on the Brazilian stock market, surpassing important benchmarks.

Keywords: Genetic Algorithm, trading agent, stock market, portfolio, finance, strategies.

Resumo

Nos últimos anos, houve um crescimento no uso de métodos computacionais na área financeira, principalmente nas negociações no mercado financeiro. Os investidores vêm usando ferramentas computacionais para automatizar estratégias de investimento com o objetivo de maximizar lucros e reduzir riscos. Neste trabalho, nosso objetivo é trazer novas ideias e abordagens para o desenvolvimento de robôs de negociação automatizados com base em dados históricos de séries financeiras. Nosso modelo, chamado Pattern Searcher, foi inspirado em métodos de aprendizado de máquina e otimização evolutiva. Dado um agente de negociação com seus parâmetros pre-definidos, o método utiliza o poder do Algoritmo Genético (GA) para pesquisar, dentro de um conjunto de indicadores financeiros, a região que fornece um retorno positivo mais alto. Essa implementação exibiu propriedades desejáveis em comparação com alguns métodos de Aprendizado de Máquina, como a simplificação do fluxo do sistema e a geração de regras que os humanos podem entender mais claramente. Além disso, foram gerados portfólios de estratégias, compostos pelas estratégias derivadas do método Pattern Searcher, que também foram otimizados via GA. O sistema conseguiu gerar agentes e portfólios muito lucrativos no mercado brasileiro, superando importantes benchmarks.

Palavras-chave: Algoritmo Genético, agente de negociação, mercado financeiro, portfólio, finanças, estratégias.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The stock market can be a very profitable place for those who have good trading strategies. Based on rules, people can decide when is the right time to buy, sell or hold a stock, find out the involved risks, the financial income, and the amount of necessary investment. If one finds out a pattern that leads him to buy a stock when the price is low and sell it when the price is high, for example, he will receive profitable returns. A good strategy may depend on exhaustive tests and validation on historical data, in order to know how the proposed strategy would have worked in the past, to estimate how it will work in the future, and to best adjusts its parameters.

It is common for investors to make trading decisions based on technical indicators, which is an analysis focused on the pattern of price movements ([17]). In a simplified manner, the buy and sell signal can be represented by Boolean expressions using a combination of those indicators. For example, in a typical moving average crossover strategy ([18]), a buy order is sent when the price crosses above the moving average in x periods and there is an increase of the traded volume, that may confirm an uptrend movement. Similarly, when the price cross below the moving average, a sell order is sent.

We can improve and create new strategies by combining different financial indicators and adjusting their parameters on historical price data. Usually, this is a time expensive process due to the huge number of indicators and parameters that can be used, even for computers. Thus, looking for computational optimization techniques, such as Genetic Algorithm, is a must.

Genetic Algorithm (GA) is an optimization technique that can be used for a variety of problems, such as designing aircraft, evolving electronic circuits, finding hardware bugs and optimizing asset portfolios. GA is an analogy to Darwin‘s theory of evolution of species and the genetic field, in order to develop an algorithm that searches for solutions in a variety of computational problems ([37]). The idea is to evolve a population of possible solutions in

the searching space of the problem. At each generation, the strongest individuals within the population are more likely to survive and combine their genetic material to generate better individuals.

In this context, the goal of this project is to develop an algorithm that searches for profitable patterns for decision making of automatized trading agents (also called trading robots), using Genetic Algorithm. In other words, creating a generator of trading strategies. The main idea is that the GA will evolve trading rules to create more profitable strategies, which tells the right time to buy or sell a stock. In addition, the GA will also be applied to generate and optimize portfolios using the found strategies, aiming to potentially improve the financial returns. The implementation and optimization were done in the trading software Metatrader 5 (MT5), using its MQL5 programming language and its strategy tester. This choice was done intending for a simple and quick transition between simulated and live trade.

1.1 Motivation

In the last few years, there was a growth regarding the use of computational methods in the field of finance, especially to Stock Market negotiations. Investors have been using computational tools to automate investment strategies with the goal of maximizing profits and reducing risks.

Several tools use Machine Learning (ML) methods, such as Neural Networks, Support Vector Machine (SVM), and Decision Tree. They use a huge amount of data as input for training the model and generate signals such as buy, sell, hold, up and downtrend, and so on. The goal is to obtain models with generalization capacity that are capable of performing very well in new unseen data. These approaches have led to profitable models, however, they may be susceptible to overfitting. Also, they are "black boxes", in the sense that the investor does not know what is inside the model, how and why it works.

In this context, we came up with the idea of using Genetic Algorithm for rule discovery in algorithmic trading. Studies applying GA for this field have been increasing in the last few years, but few papers have been published so far [16]. Based on Darwin's theory of evolution of species, the GA can evolve profitable strategy rules, which are used for decision support ([37]).

According to [12], rules found by GA exhibit several desirable properties compared to machine learning methods. For example, the rules generated are not "black boxes", and humans can clearly understand. In addition, the simplicity of the method may reduce the chance of overfitting.

The Machine Learning system flow is usually divided into 2 major steps: training the model in terms of accuracy and using the ML model to create trading strategies. The inconvenience of this double step is that the ML model accuracy, for example, does not necessarily result in financial return. It may be necessary to repeat all the process over and over. On the other hand, in the model proposed in this work, the trading strategy and training are directly intertwined, happening simultaneously. The model training directly optimizes the financial return, which is what we are most interested in. That makes the process flow simpler and easier to work with.

The motivations behind the use of Metatrader 5 are due to its enormous library containing many financial indicators, the MT5 built-in optimization tools and the possibility of using the programmed agents to perform in live trade (many Brazilian brokers support and distribute the MT5). Besides, the MQL5 language is similar to the very popular C++ and it is very robust. as it has been using and tested by a large community for many years, especially in real account.

1.2 Objectives

The **general objective** of its work is to develop automated trading agents and strategy portfolios using Genetic Algorithm.

The **specific objectives** of this work are:

1. To improve the knowledge of algorithmic trading and computational optimization techniques;

2. To learn how to develop automated trade systems;

3. To create a method of hopefully finding profitable patterns for decision making of trading agents;

4. To design a methodology for developing strategy portfolios;

5. To develop a method for estimating the number of contracts in leveraged trade systems;

6. To apply the system on the Brazilian stock market and compare its performance with important benchmarks;

7. To produce a document useful for research in the algorithmic trading field.

1.3 Contributions

With this work, we expect to bring new ideas and approaches to the development of automated trading strategies based on historical data of financial series. Specifically, the contributions are:

A configurable robot: This work created a generic robot on Metatrader 5 with easily configurable parameters, which can be used by an investor without the need of knowing any programming language.

Model for generating trading agents: Elaborate a model to automatically search for profitable trading patterns.

Methodology for estimating a safe leverage ratio: Propose a methodology for estimating the amount of leverage that safely maximizes the profits.

1.4 Organization

This dissertation is divided into 6 chapters. In Chapter 2 is given a theoretical foundation, with relevant topics including stock market, Genetic Algorithm, and portfolio management. In Chapter 3 is given a literature review with themes related to Machine Learning techniques widely applied to trading systems and Genetic Algorithms to generate trading rules. Chapter 4 consists of a description of the methodology used to achieve the proposed goal, including the Pattern Searcher modeling, the controlled leverage methodology, and the generation of strategy portfolios. In Chapter 5 is presented the experimental validation of the trading agents and the agent portfolios trained via Genetic Algorithm. Finally, Chapter 6 concludes the work, summarizing the most relevant results and discussing future works.

Chapter 2

Theoretical Foundation

This work uses many terms and concepts related to the financial market, computational optimization, automated agents and portfolio management. Thus, we attempted to describe its most relevant characteristics. In Section 2.1 we show basic concepts about the stock market and automated robots. In Section 2.2 we formalize a definition of an agent. In Section 2.3 we explain how Genetic Algorithm works. In Section 2.4 we discuss about portfolio management. Finally, in Section 2.5 we show details about the platform Metatrader 5, used to implement our trading agents.

2.1 Stock Market

According to [3], the Stock Market is the place where are negotiated corporation shares, commodities, etc. The stock market serves 2 main functions: for a company, it provides access to capital, usually in the form of cash. If a company needs to finance a project, for example, it can sell its shares in the stock market and raise some capital instead of borrowing money from a bank. From the perspective of who acquires the share, the market provides a way to participate in a company's growth and quickly convert shares into cash. A significant part of the investors aims to maximize their profits based on the idea of buying a stock at a low price and sell it in another moment for a higher price, getting a positive financial return. However, a big challenge is to know when is the best time to buy or sell an asset, as the asset price may depend on many factors, such as politics, economy, news, and speculation, making very hard the price prediction [35].

In Brazil, all negotiations of shares and contracts are managed by the *Brasil, Bolsa, Balcão* (B3). The activities include the creation and management of trading systems, compensation, liquidation, and register for the most important classes of assets, from stocks to currencies, income rate and commodities. In 2019, B3 had over 300 registered companies.

In the stock market earlier days, the stocks were traded person to person on the floor of the exchange. Nowadays, most trades are done electronically ([3]). It makes possible for a huge volume of trades be done in a very short time interval.

Automated Robots

In this electronic environment, automated trading systems are dominating the market. Automated trading systems, also referred to as investing robots and algorithmic trading, allow traders for programming specific rules and trading strategies that are automatically executed via computer ([14]). Approximately 75% of shares traded on U.S. stock exchanges come from investing robots ([22]).

The advances of the automated trading system are: Minimizing emotional trading, allowing for backtesting, providing discipline, enabling multiple accounts, allowing diversification of trade in multiple strategies at one time. However, these advances come with some disadvantages: mechanical failures can happen, requires the monitoring of functionality, and can perform poorly possibly due to some overfitting ([14]).

Technical Analysis and Indicators

According to [4], "Technical indicators are heuristic or mathematical calculations based on the price, volume, or open interest of a security or contract used by traders who follow technical analysis". The indicators are used to predict future price movements and to give entry and exit points for trades. Besides, they are often used for the decision making of automated systems or used as input in Machine Learning models. There are two basic types of technical indicators: *overlays* and *oscillators*. The *overlays* use the same scale as prices and are plotted over the top of the prices on a price chart, while the *oscillators* oscillate between a local minimum and maximum and are plotted above or below a price chart ([4]). Some of the most common indicators are:

RSI: Relative Strength Index (RSI) is a momentum indicator that measures if the price is oversold or overbought.

MACD: Moving Average Convergence Divergence (MACD) measures the relationship between two moving averages of an asset

MA: Moving Average (MA) is a trend-following indicator that smooth out the price by filtering out the noise.

ADX: Trend Strength Indicator (ADX) determines when the price has a strong trend.

MFI: Money Flow Index (MFI) is an oscillator that combines price and volume for identifying if the price is overbought or oversold.

STO: Stochastic Oscillator (STO) is a momentum indicator that measures if the price is oversold or overbought.

Pivot Points: Lines that show important price levels calculated by using the high, low and close prices of the previous day.

Bollinger Bands: Lines plotted two standard deviations (positively and negatively) away from a simple moving average (SMA) of the price.

The chart on Figure 2.1 shows some of those indicators, including the RSI, the MACD, and the moving averages.

Figure 2.1: Stock graph with different technical indicators. There is an RSI with a period of 14 on the top, moving averages with periods of 50 and 200 on the middle, and a MACD with parameters 12, 26 and 9 on the bottom of the graph. Source [4]

Leverage

Leverage is when an investor uses borrowed money in an attempt to increase the ratio of return on an investment, which increases the potential return ([15]). For example, a restaurant owner can borrow money to open new restaurants around the town, improving its returns. In the context of the individual investor in the stock market, brokers generally allow some leverage. A leverage ratio of 1:10, for example, means that investors can negotiate 10 times more cash than what they have in the broker's account. This can potentially increase the returns, but also the losses. There are 2 noticeable Future assets in the Brazilian market

which are given the possibility of using very high leverage: the BMF Mini Ibovespa Futures (WIN) and the Mini Dollar Futures (WDO). In the Futures market, the buyer commits to buy the contract asset at the due date by a pre-defined price. The seller commits to sell and give the asset to the buyer.

2.2 Trading Agent

An agent can be considered as an entity that perceives the environment through its sensors and takes actions through its actuators (see Figure 2.2).

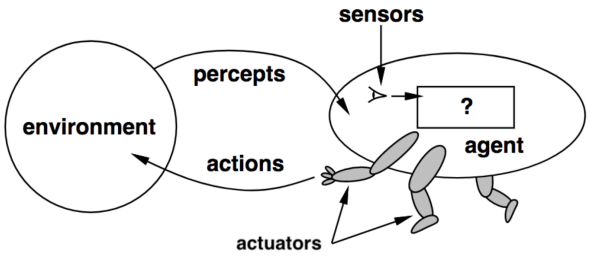


Figure 2.2: An agent interacting with the environment through its sensors and actuators.

Some examples of agents are:

Humans: they perceive the environment through vision, hearing, tact and actuate trough hands, legs, etc.

Robots: they perceive the environment through a camera, laser and actuate through motors.

Software: they perceive the environment through keyboard, mouse, internet and actuate trough the screen, internet, etc.

Overall, agents are designed to be rational. For each sequence of perceptions, rational agents choose the action in order to maximize its performance, considering its current perception about the environment and some previous knowledge ([38]).

Autonomous trading agents, also called trading robots, “are computer programs that bid in electronic markets without direct human intervention” ([29]). Their environment is the stock market, including a diversity of assets, and variables such as prices and volumes. They perceive this information through the internet, by requiring the data from a broker. Then, the agent process that information, with some previous knowledge and intelligence coded in, and make intelligent decisions, regarding when to buy and sell an asset, at which price, calculate the stop loss and take profit, etc. Finally, it takes actions through trading functions, that send buy and sell orders directly to the broker via internet. In this project, the rationality of the agents was achieved though training a decision-making system that searches for profitable patterns.

As with many computer-based activities, trading agents can provide many advances over humans. According to [29] they can monitor many markets and assets simultaneously, they can process large amounts of data, make complex numerical calculations in real-time, and take actions very quickly. However, they have some disadvantages such as learning from experience and taking complex judgments such as humans do. These are typically the subjects of the trading agents research ([29]).

2.3 Genetic Algorithm

Genetic Algorithm (GA) is a searching-heuristic biologically inspired by Charles Darwin's theory of natural selection. According to [26]:

> "The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance of surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found".

In Genetic Algorithm, this idea is applied for searching solutions for a problem. Each individual is a solution to the problem, and the bests are selected. Figure 2.3 illustrates the GA process.

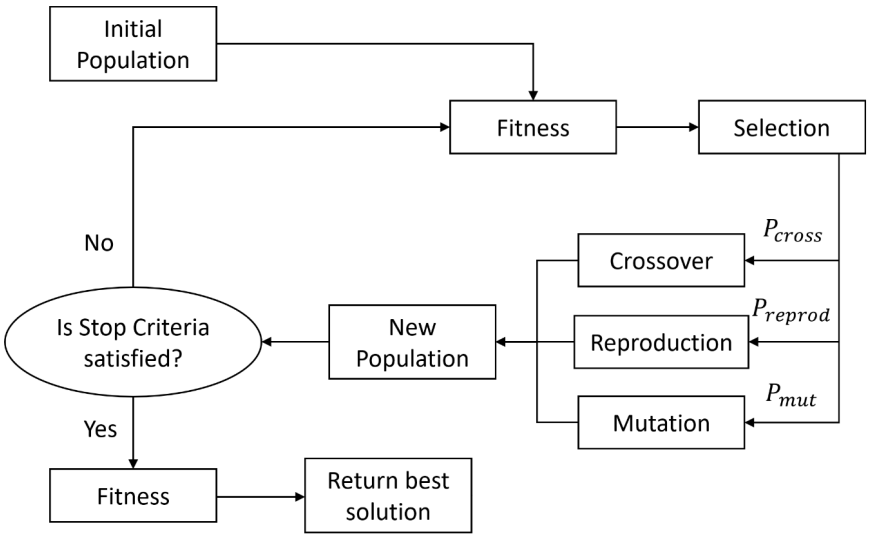


Figure 2.3: GA diagram. Modified figure from [33].

The system can be divided into Initial *Population, Fitness Function, Selection, Genetic Operators* and *Termination*. Their details are described in the following subsections. We have used [38] and [26] as reference.

Initial Population

The first algorithm step is to create the initial population with a set of random individuals. Each individual is a solution to the problem that is desirable to solve. Individuals are composed of genes, which are a set problem's parameters (see Figure 2.4).

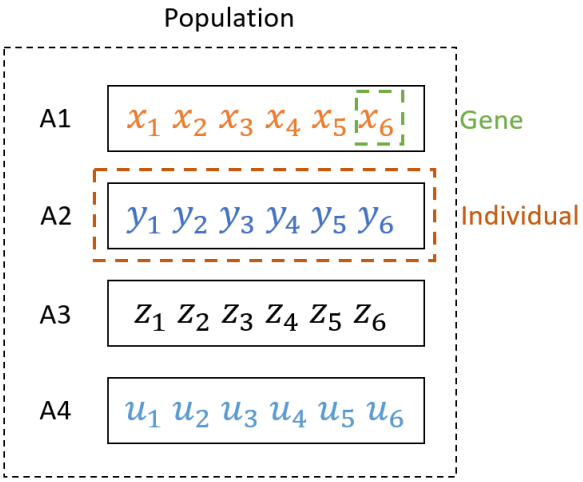


Figure 2.4: Population, individuals and genes.

Fitness Function

The fitness function evaluates the quality of the solution represented by an individual. It gives a score that determines how good the individual is compared to other individuals, being very important during their selection.

Selection

The selection is responsible for selecting the fittest individuals, through a selection algorithm, and letting them pass their genes to the next generation ([26]). A pair of individuals are selected based on their fitness for crossover and one individual is taken for mutation. Individuals with great fitness have higher chances to be selected for reproduction.

Roulette Wheel Selection is a very popular selection algorithm. The likelihood of selecting an individual is proportional to the fitness. The probability p_i of the ith individual, with a fitness of f_i, be selected from a population of size N can be calculated by the formula:

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \tag{2.1}$$

There are other selection methods, such as Rank, Tournament and Elitism selection.

Genetic Operators

The genetic operators are operations applied to the selected individuals in order to create a new and "better" generation of individuals. There are 3 very common genetic operators: crossover, mutation, and reproduction.

In the crossover, a pair of "parents" is selected. Their chromosomes are picked randomly and exchange between them, generating 2 children with genes from both parents. Figure 2.5 shows the uniform crossover, where each gene is exchanged with a probability p_c.

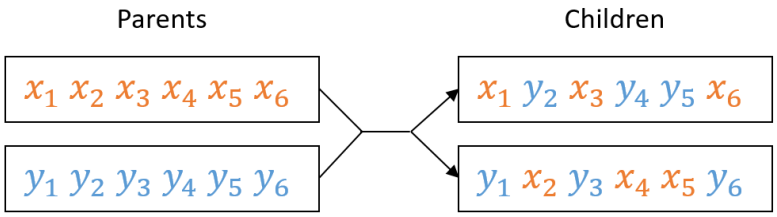


Figure 2.5: Uniform Crossover. Parents randomly exchange their genes generating children.

Yet, Figure 2.6 shows the single-point crossover, where a crossover point is chosen randomly and all the genes from the right or left are exchanged.

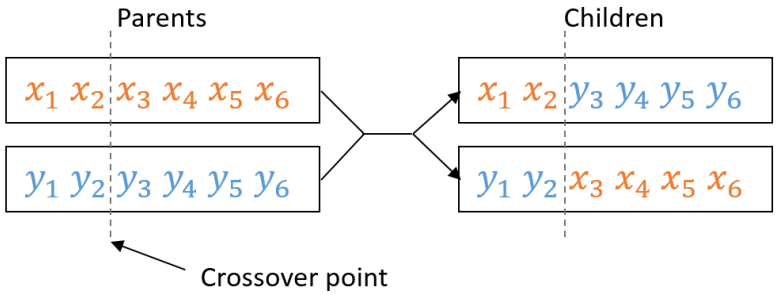


Figure 2.6: Single-point Crossover. A crossover point is chosen randomly and all the genes from the right or left are exchanged.

After the crossover is done, a mutation can be performed. The mutation randomly changes some genes for random ones, with low probability. It is a very important genetic operator that maintains the diversity within the population and prevent premature convergence ([26]).

Reproduction is a genetic operator in which the best individual goes to the next generation without having any modification of its genes. It guarantees that the best individual of the next generation will be equal or greater than the previous generation's best individual.

Termination

The optimization terminates when a stop criterion is satisfied when the number of generations reaches the maximum number of generations, or when the population has converged. In the last case, it means that the subsequent generations are not significantly different from the previous one. When the GA terminates, it provides the best solution found for the problem.

Example

In order to facilitate the understanding of GA, we will give a simple example. Suppose we have the problem: given 6 variables, from 0 to 9, find the combination which maximizes the sum of those variables. In this case, our fitness function is equal to the sum of those variables. Figure 2.7 shows the optimization process.

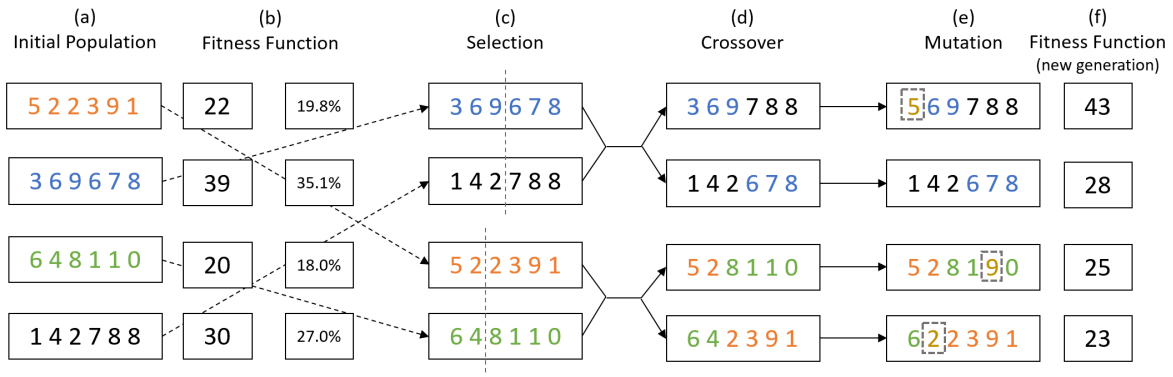


Figure 2.7: Demonstration of the GA process applied to a simple example: maximization of the sum of 6 variables.

Firstly, an initial population is generated with random values for each variable (a). The fitness function was defined to be equal to the sum of the variables, once the greater the sum, the better is the individual (b). Using the Roulette Russel selection, we calculate the probability of each individual being selected, and then we select pairs of individuals accordingly to a probability (c). They perform crossover by mixing their genes (d), and mutation by having some of their genes replaced by some random number (e). After all of this process, we have a new generation (f). Notice that the fitness of the new generation best individual (43) is greater than the fitness of the initial population best individual (39). At

each new generation, individuals tend to have higher fitness than the previous generation. This process is repeated several times until the stop criterion is satisfied.

2.4 Portfolio Management

In finance, a portfolio is a group of financial assets that can include stocks, bonds, commodities, currencies and cash equivalents ([5]). Investors construct an investment portfolio according to tho their objectives and the amount of risk they are willing to take.

Figure 2.8 shows an investor portfolio where 12.57% of the funds are allocated in the CDI, while 80.43% are allocated in stocks. There is also a different allocation of funds in each particular stock.

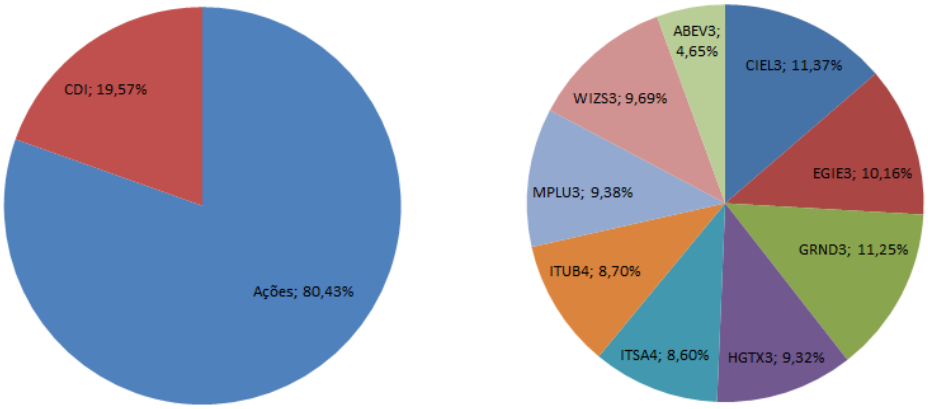


Figure 2.8: Porftolio.

Portfolio Theory was proposed by Harry Markowitz in 1952, which lead him to award the Nobel Prize in Economics in 1990. According to [25] Portfolio theory consists of "exploring the optimal allocation of wealth among different assets in an investment portfolio, based on the twin objectives of maximizing return while minimizing risk".

Returns reflect the efficiency of an investment, the expectation that the investment will increase over time. The risk is concerned with the uncertainty about future market behavior, since for many assets, such as stocks, forward contracts, and options, their future values cannot be predicted with certainty. The concept of risk can be interpreted in many ways. A very common way to mathematically measure the risk is the variance or the standard deviation of the returns. The drawdown (larger consecutive sequence of loss) can be also a measure of risk.

The construction of portfolios consists in allocating of resources among the assets that compose the portfolio. Proportion for each asset in the portfolio can be represented by a weight w.

According to [28], stocks face both systematic risk and unsystematic risk. The systematic risk is a risk that affects the whole market, due to factors such as interest rate changes, inflation, recessions, and wars. The unsystematic risk is a risk that can affect specifically a single stock or industry, due to factors such as management changes or poor sales.

Although the diversification of securities and assets can not prevent systematic risk, it has the benefit of eliminating or at least decreasing the unsystematic risk.

2.5 Metatrader 5

According to [9], Metatrader 5, also known as MT5, is an electronic trading platform developed by MetaQuotes Software, released in 2010 (see Figure 2.9). It is a multi-asset platform that allows trading forex, stocks, and futures. It offers tools for price analysis and the use of algorithmic trading applications.

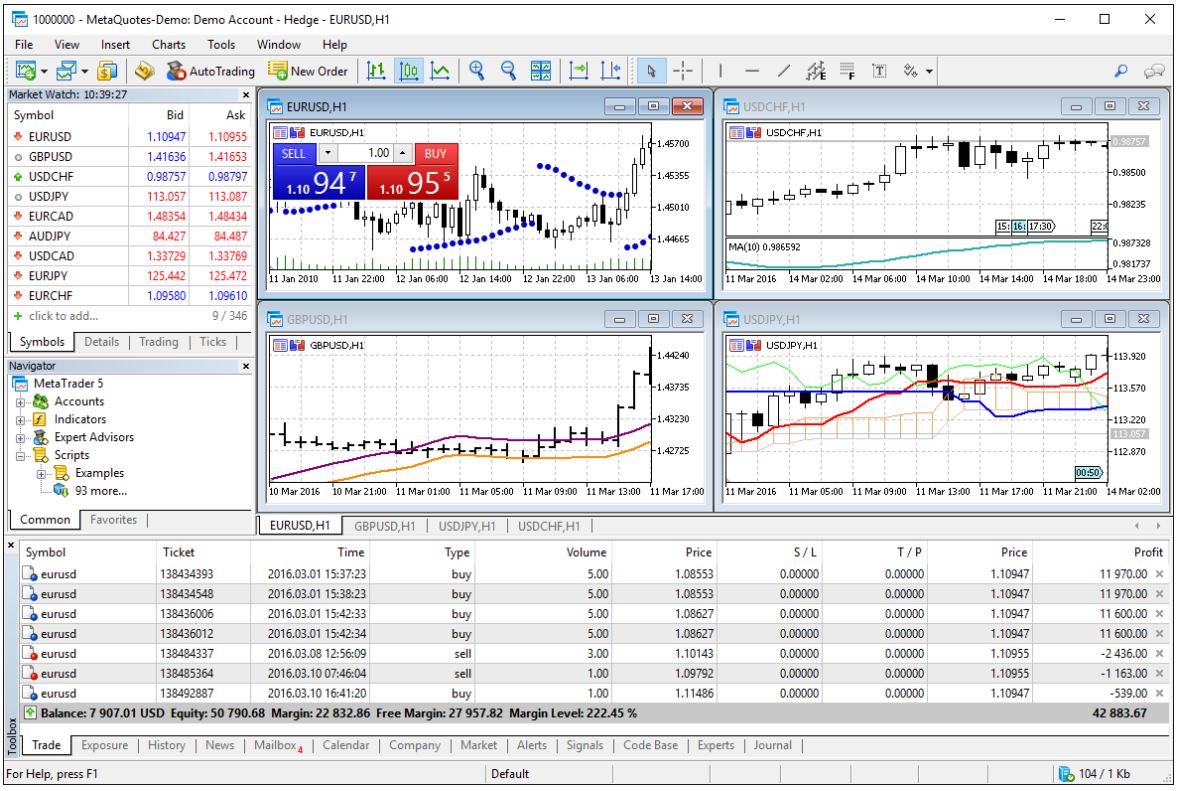


Figure 2.9: Metatrader 5 software. Source [9].

Algorithmic trading is a feature that implies automated trading using trading robots, also called as Expert Advisors (EA) by the MT5 community. The robots can analyze quotes and perform trading operations following a coded algorithm and trading rules, with-

out the participation of the trader. The EAs are written in a programming language called MetaQuotes Language 5 ([9]).

According to [7], MetaQuotes Language 5 (MQL5) is a high-level language designed for developing technical indicators, trading robots, and utility applications, which automate financial trading. The language syntax is object-oriented similar to C++. The MQL5 language provides specialized trading functions and predefined event handlers to help programmers develop Expert Advisors. Besides, MQL5 allows developing custom technical indicators, libraries, and scripts.

Strategy Tester

The built-in MetaTrader 5 Strategy Tester facilitates the testing of automated robots on historical market data. This tool allows for both testing the efficiency of an Expert Advisor and detecting the best input parameters before running the EA on a real account ([9]).

During the test on the Strategy Tester, the Expert Advisor goes through a historical quotes of currencies, stocks or other assets, and performs virtual transactions according to the EA algorithm. This procedure allows an evaluation of the robot performance in the past. Besides, it takes only a few minutes in the Tester rather than days, weeks or months needed to test an EA in the real market.

The Strategy tester also provides other interesting features: *Graphical display of test results, Visual testing, Optimization* and *Graphical display of optimization results*. The *graphical display of test results* shows the EA results including profit/loss percentage ratio, number of profitable/loss-making deals, risk factor, expected payoff. The *visual testing* allows monitoring the trading robot's operation in real-time on historical data. The *optimization* allows finding the best input parameters for a specific trading robot. The *graphical display* provides powerful 2D and 3D tools for visual analysis of the optimization results.

Genetic Optimization

The Strategy Tester has an optimization module, which allows finding the best input parameters for a specific trading robot (see Figure 2.10). With optimization, the parameters are modified to achieve maximum profitability and minimum risk.

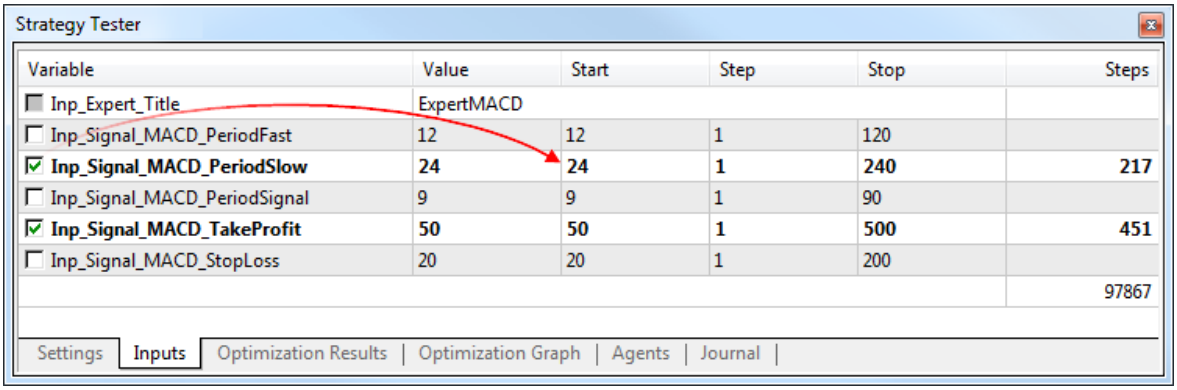


Figure 2.10: MT5 optimization panel. Source [10].

During the optimization process, one trading robot is tested multiple times with different sets of parameters. After the optimization, results that can be compared in order to select the parameters that provide the best robot performance.

The optimizer gives the possibility of using Genetic Algorithm, once there can be a huge number of combinations of input parameters in the optimization. This feature selects only those parameters that best meet the optimization criteria set. At each phase, the "optimal" combinations are crossed until it achieves the best possible result. The genetic algorithms help to considerably reduce the number of combinations and the total optimization time ([10]).

[8] describes in a general form how their GA implementation works:

1. From the total number of all possible combinations of chosen parameters, two populations (sets) are selected by a random sample;

2. Both sets are tested and the one with the best results (according to the optimization criterion) remains;

3. The set members are randomly crossed with one another, undergoing random mutations and inversions of parameters;

4. The descendants are sorted out by the best results, and crossing repeats;

5. Sorting and crossing operations are repeated as long as there is an improvement of results (the best result among descendants is better than the best one among the parents). If the optimization criterion values are not improved during several crossings (generations), the optimization process is completed.

Some of the GA parameters are:

1. **Population size**: it is calculated based on the number of possible combinations of optimization parameters, and may range from 64 to 256 individuals;
2. **Number of generations**: it may range from 15 to 31. It is defined by the presence of improvements in the optimization. If there are 6 generations without any improvement of the best individual, optimization is stopped.

The following optimization criteria (fitness function) can be chosen to evaluate the individuals:

1. **Balance max** — balance;
2. **Balance + max Profit Factor** — a product of balance and profit factor;
3. **Balance + max Expected Payoff** — a product of balance and the expected payoff;
4. **Balance + min Drawdown** — a balance value and the drawdown level are taken into account: Balance/Equity drawdown;
5. **Balance + max Recovery Factor** — a product of the balance and the recovery factor;
6. **Balance + max Sharpe Ratio** — a product of balance and Sharpe ratio;
7. **Custom max** — allows using any custom value for the Expert Advisors optimization.

Chapter 3

Related Works

The majority of computational tools used by investors to automate investment strategies and create stock trading systems uses Machine Learning methods, especially Long Short Term Memory (LSTM). LSTM is a Recurrent Neural Network that has memory mechanisms that are well-suited for classifying and making predictions based on time series data [32].

[13] combined stock prices with stock news data from Google Trends using the LSTM network for decision support. The use of LTSM improved the predictability from 51.9% to 58.2%, compared to the simple model of Neural Network MLP (Multi-Layer Perceptron). Similarly to this work, [24] used the LSTM to predict the volatility of the Shanghai Shenzhen CSI300 Index, achieving an accuracy of 78% in the predictions.

The LSTM neural network has shown to be very appropriate for temporal series like financial data. However, as market behavior changes over time, this model needs to be periodically retrained. This creates a necessity for models that are able to dynamically adapt to new market conditions. An alternative is the use of Reinforcement Learning.

Reinforcement Learning (RL) is a learning system where an agent perceives the environment and takes actions in order to maximize a notion of cumulative reward [40]. In the context of automated trading systems, [30] used the Q-Learning and proposed algorithm, named as Recurrent Reinforcement Learning (RRL), to trade a portfolio. They used the Sharpe index as the reward function, which measures a relation between the portfolio profitability and the investment risk. Both algorithms were profitable and surpassed the baseline Buy-and-Hold.

Still, in the context of Machine Learning, [34] compared the accuracy of 4 forecast methods: ANN, Random Forest, Naive Bayes, and SVM, to predict the next day up and down trends of a stock and index price. It used as input 10 technical indicators with a window of 10 days, calculated on daily close prices. In the first approach, they have directly used the indicators, while in the second, they represented those indicators by trend information. The

second approach gave the best results, obtaining an accuracy of 90%. [39]'s work aimed a similar goal of predicting the next day's trend of the Brazilian Index, with the difference of using a Bayesian Network. The hypothesis is that every closing market influences the next one around the world. Thus, they grouped a few important indicators by their continents and created a Bayesian Network with 24h and 48h windows size. Using the Bayesian theorem, they calculated the probability of price movement is up or down, by each combination of those indicators, achieving an accuracy of 71%.

[27] proposed a stock market day-trading system that uses the outputs of an ANN to guide the user into buying and selling stocks. Unlike a couple of works that try to predict trends, the ANN is used to forecast the lower and higher stock prices of the current trading day, tested with BM&FBovespa, Vale and Petrobras (Brazilian stocks). The model received as input the high and low prices of the past 5 days and the open price of the current day. They used those predictions in a trading strategy, where they buy when the price touches the lower predicted price, and sell when it touches the higher predicted price. This approach gave great results, doubling the invested deposit in a period of 150 days, using the annualized return metric. From this work, we may conclude that what people are looking to predict with their models and how they use them, in terms of strategy, makes a big difference in terms of profitability of the model being used in real trading.

In literature, there is a parallel approach to Machine Learning that uses Evolutionary Algorithms (EAs), such as Genetic Algorithm (GA), for creating trading decision support systems. Evolutionary Algorithms has been used for feature selection (select the best inputs for a Machine Learning model), rule parameter (find the best parameters for a specific rule or system), rule combination (combine the best rules) and rule construction (evolve more complex rules from simple ones).

[23] compared two Genetic Systems for creating trading strategies. In the first, GA was used to train the weights of a Neural Network, which received technical indicators of NASDAQ stocks as inputs. The second system used Genetic Programming (GP) to derive trading strategies in a tree representation. Both systems returned, as output, trading signals such as buy, sell and hold. They found that the Genetic Programming method generated better trading strategies compared to the neurogenetic prototype based on genetic optimization of neural network weights, and found that both were more profitable than the Buy-and-Hold strategy, which is usually used for comparison in many works.

In [12]'s work, GA was used for rule construction in intraday trading, and its performance was compared to a Reinforcement Learning system, a Markov Decision Process-based system and a heuristic. The rules found by GA exhibited several desirable properties. Unlike many Machine Learning implementations, the rules are not "black boxes" and can be understood by humans. The GA outperformed the other methods in new unseen data,

although none of the methods produced significant profits at realistic transaction costs.

[36] has developed a decision making and trading algorithm named Goldminer. It used the power of Genetic Programming, combined with indicators of financial technical analysis to identify the time of purchasing and selling a stock. The individual was composed of the logical operators AND, OR, XOR and the indicators of technical analysis, while the fitness was calculated simply by counting the profitability (all profits subtracted from all losses). The inclusion of the two windows verification to validate the models was a key distinguishing point for reaching the target time and profit in more than 90% of the runs tested on the Brazilian stock market BM&FBOVESPA.

[19] did similar work. They have implemented an Evolutionary Algorithm for generating trading rules for intraday trading, but with different individual representation. The individual was a binary decision tree, directed acyclic, whose leaf nodes contained buy and sell decisions. As the fitness function, they used the total stock return. When the strategies generated by the algorithm were tested in the forward test, in the major cases, the profitability degraded drastically. However, the strategies remained efficient, and in all cases, it exceeded the rate of risk-free return, and the maximum drawdown values (worse consecutive sequence of losses) stayed within 7%.

There is a financial forecasting tool based on Evolutionary Algorithms designed to help investors to create trading rules in a form of Genetic Decision Trees that can be readable, called EDDIE (Evolutionary Dynamic Data Investment Evaluator) [41]. According to [20], its most recent version (EDDIE 8) is not constrained in using pre-defined indicators, but it automatically chooses the optimal ones. The tests were performed in an artificial data set, which the authors knew patters existed. Eddie 8 was able to find those patterns, but it seems that it was having difficulties with searching effectively in the space state. In an attempt to improve EDDIE 8, [21] used hyper-heuristics, under a GP framework, to do a more efficient search in the space of financial indicators.

According to [16], studies applying GA for rule discovery in algorithmic trading have been increasing in the last few years, but few papers have been published so far. There is still a lack of comparison between Evolutionary Algorithm implementations and other techniques. It indicates that there is a need for more study on this topic and there is still plenty to explore. In their paper, they suggested some future research directions, including:

1. Predicting future market trends: predict future market trends and search for more predictable and influential classification standards can bring much help (e.g., strategies can be selected accordingly to the prediction);

2. Considering liquidity and transaction costs in more precise and positive ways, and combining portfolio selection techniques and period for training and learning;

3. Optimization of training and learning periods: “The relationship of stock classification, periods of training and testing, and profitability may be an interesting topic” [16];

4. Combining portfolio selection techniques: researchers need to consider the correlation of stocks in different environments, relationships between trading rules (such as priority and sequence), transaction cost, and selection of fitness function.

In the field of finance, backtest overfitting is one of the most important open problems ([11]). Backtest overfitting is when the developed strategy performs well in the backtest because it has “memorized” random patterns of the data set, rather than learning important features about it. As those patterns are unlikely to happen in the future, the strategy fails in real account trading. Several steps can help to reduce the overfitting presence, including developing models for entire asset classes, rather than for specific securities ([11]), and reducing the model complexity ([1]).

In our proposed work, we aimed to address some of the suggestions of [16] and also to develop a system with reduced complexity, attempting to avoid overfitting.

Chapter 4

Methodology

In this work, we have developed a model that searches for profitable buy and sell patterns and generated portfolio of strategies. In order to achieve those goals, the methodology of this work was divided in 4 sections: *Configurable Agent*, *Pattern Searcher*, *Controlled Leverage*, and *Strategy Portfolio*. Figure 4.1 shows the work's diagram with these sections.

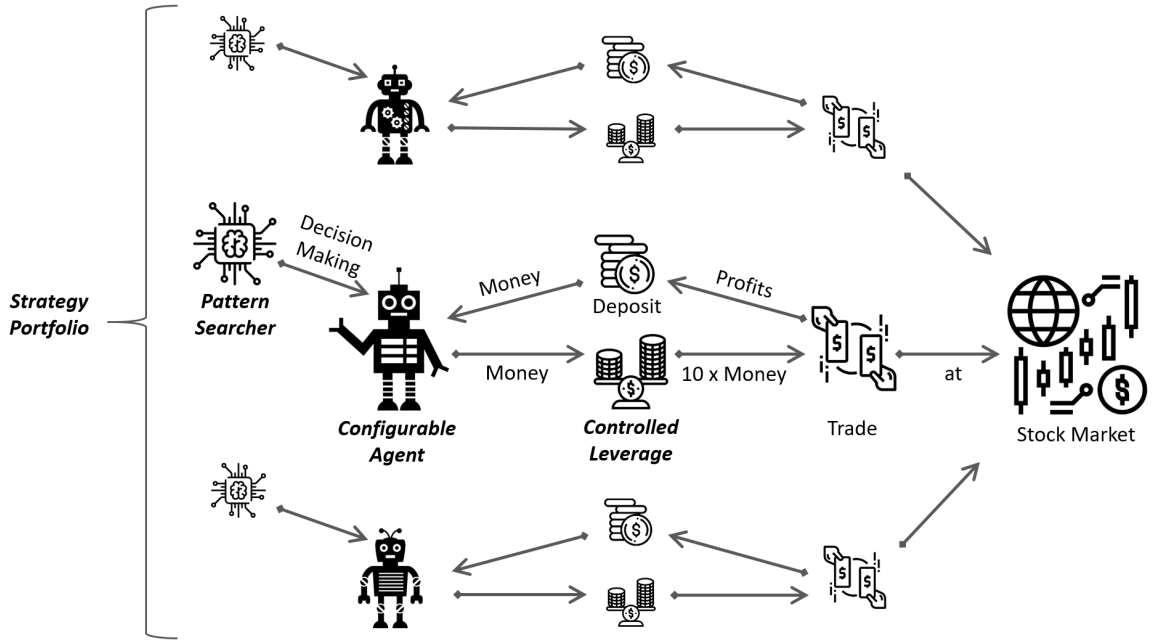


Figure 4.1: Work's diagram. We create a portfolio with agents, where each of them takes different decisions based on their Pattern Searcher model. The agents use controlled leverage to multiply their money, which is used for trading assets in the stock market and hopefully generate profits.

In Section *Configurable Agent* we explain details of the agents' implementation on MT5, such as its configurable parameters, risk management, and pseudo-code. In Section

Pattern Searcher, we show the model of the pattern searcher system and its training procedure. In Section *Controlled Leverage* we propose a methodology to determine the amount of leverage that will be used and the calculation of the monthly return. In the Section *Strategy Portfolio* we generate portfolios with the agents found by the pattern searcher method.

In summary, the methodology sections approach the topics: developing a configurable robot on Metatrader 5 using the MQL5 programming language; modeling a pattern searcher system using Genetic Algorithm to give trade signals of buy and sell; implementing a risk management system; creating several agents with different hopefully profitable entry rules using the pattern searcher model; generating strategy portfolios using the strategies found by the pattern searcher method; optimizing the portfolio's strategies allocation using GA.

4.1 Configurable Agent

Aiming for a simple and quick transition between simulated and live trade, we have used the software Metatrader 5 (MT5) and its programming language called MQL5 to program the trading agents of this work. The trading robots are also called Expert Advisors (EA) in the MT5 community. We have written a more generic code that contains a customizable diversity of parameters that dictate how the EA will work. By changing its parameters in an intuitive MT5 painel, we can make the EA trade a variety of strategies, such as trend followers and reversion based ones.

Agent parameters

The parameters were organized in 6 groups: *General Parameters*, *Position Management*, *Order Management*, *Risk Management*, *Entry Conditions* and *Pattern Searcher*. The robot allows both creating an entry condition manually, by selecting and combining indicators, and creating an entry rule automatically by using the pattern searcher module.

In the group of *General Parameters*, there are the basic parameters necessary for any EA to work. The *Position Management* group is responsible for dealing with positions, such as the maximum number of positions allowed and the time for closing a position. The *Order Management* group deals with pending orders, such as the kind of order (limit or stop) and expiration order time. In the *Risk Management*, we can control the risks by setting parameters such as take profit and stop loss. In *Entry Conditions* we can manually create a rule using financial indicators to determine when the agent will buy or sell. Finally, in *Pattern Searcher* there are the variables of the Pattern Searcher model, where we can select which patterns to be used, its radius and center. A summary with the most important variables used in this work is shown in Table 4.1.

Table 4.1: Most relevant agent's variables.

General Parameters	
MagicNumber	Every EA must have a different magic number. It allows the EA to track the trades that it opened.
Symbol	Symbol of the asset which the agent will trade.
Timeframe	Timeframe of the candles which the agent will trade.
Lot	Volume size of the positions that will be opened by the agent.

Position Management	
IsDaytrade	If it is daytrade, the EA will exit at the end of the day.
ClosePosTime	Time which all positions will be closed.
ClosePosAfterMinutes	Maximum time (in minutes) a position can stay opened. Passed n minutes, the position will be closed.
AllowTwoPositions	Specify if the EA can have BUY and SELL positions simultaneously, or just one position.
MaxPositions	Maximum number of positions that can be opened by this agent.

Risk Management	
StopType	Specify the stop type: (1) Proportional to the price (2) Proportional to the ATR of n periods.
TakeProfit	Take Profit size in percentage %.
StopLoss	Stop Loss size in percentage %.

Order Management	
OrderType	Order Type: (1) Stop (2) Limit.
OrderExpirationTime	Time the order will expire.
DeleteOrderAfterCandle	Keep the placed order until the expire time orremove it after one candle has passed.
WherePlaceOrder	Where the orders will be placed: (1) Next to the candle close price (2) At the max/min of the last n candles.
MaxMinNCandles	Number of candles used for calculating the max and min.
OrderDeviation	An small increment, in ticks, added to the order price.
ChangeStopByLimit	Change Stop orders by Limit Orders (e.g.; a buy stop order placed above the current price becomes a sell limit order).

Pattern Searcher	
UsePattern1	Use pattern 1?
Pattern1_Type	Pattern 1 type: (1) BUY (2) SELL.
Pattern1_Indicator1	Select the indicator/rule which will be used in the pattern 1.
Pattern1_Indicator1_Period	Specify the period/parameters of the chosen indicator in the pattern 1.
Pattern1_Indicator1_Center	Center of the primitive of the indicator 1 of the Pattern 1. It will be found by using GA.
Pattern1_Indicator1_Radius	Radius of the primitive of the indicator 1 of the Pattern 1. It will be found by using GA.
Pattern1_Indicator2	Select the indicator/rule which will be used in the pattern 2.
Pattern1_Indicator2_Period	Specify the period/parameters of the chosen indicator in the pattern 2.
Pattern1_Indicator2_Center	Center of the primitive of the indicator 2 of the Pattern 1. It will be found by using GA.
Pattern1_Indicator2_Radius	Radius of the primitive of the indicator 2 of the Pattern 1. It will be found by using GA.
...	
UsePattern2	Use pattern 2?
Pattern2_Indicator1	Select the indicator/rule which will be used in the pattern 1.
...	

In this project, we will not discuss the creation of manually entry conditions, since it is not in the scope.

Pseudo-code

The Listing 4.1 shows a pseudo-code of the implemented algorithm. It was implemented in MQL5 using the standard library and its trade classes. A complete tutorial of how to create an EA using the standard library can be found in [31].

Listing 4.1: EA pseudo-code.

```
Set_Parameters(); // set the parameters

OnInit()  // this function is called just once
{
    CheckMargin(); // check if the EA has enough margin to trade
    Create_Trade_Class(); // create the trading class, responsable
       for sending orders
    Create_Indicators(); // create the indicators
}

OnTick()  // this function is called at every price tick
{
    Calculate_Indicators(); // Update the indicators values.
       Includes MA, RSI, MACD and Pivots
    Close_Positions_End_Day(); // close position at the end of the
       day and lock the EA
    Get_Price(); // get close price, ask and bid
    Calculate_TakeProfit(); // calculate the take profit based on
       the price
    Calculate_StopLoss(); // calculate the stop loss based on the
       price
    Calculate_OrderPrice(); // calculate the price where the order
       will be placed
    if (CheckClosePos()==true) Close_Position(); // close a position
        if a close condition is satisfied
    if (CheckBuy()==true) Place_Buy_Order(); // if the buy condition
        is true, place a buy order
    if (CheckSell()==true) Place_Sell_Order(); // if the sell
       condition is true, place a sell order
}
```

4.2 Pattern Searcher

This section describes the pattern searcher modeling, the training process of the model using GA and the normalization of the indicators that were used as model inputs.

System modeling

In this work, we propose a model for decision making of trading agents, which we named Pattern Searcher. This model, inspired by Machine Learning methods and Evolutionary Algorithm, automatically searches for buy and sell patterns using as input a set of indicators. In order to present details about the modeling, we have divided it into: *Idea*, *Hypothesis*, *Problem*, *Solution*, and *How to use the system*:

Idea: Given a trading agent with its pre-defined proprieties (e.g., stop loss, take profit, stop trail, close position time, indicator parameters) and a set of financial indicators (e.g., MACD, RSI, MA, PIVOT), the method will search, within this set of indicators, for the region that provides the higher positive return.

Hypothesis: There is a region within the multidimensional space of indicators that gives highly positive returns for a given operation (e.g., buy or sell), which can be covered by a geometric primitive (e.g., ellipses, boxes). Figure 4.2 illustrates a profitable region, composed by 2 indicators, being covered by a box.

Problem: Find the properties of the geometric primitive that cover the region (e.g., center and radius for ellipses, or center and edge for boxes).

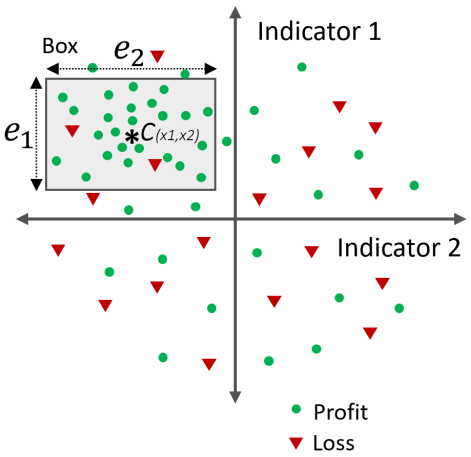


Figure 4.2: The dots and triangles represent, respectively, positive and negative returns from trades on historical market data, performed by an agent randomly opening positions. For a given set of indicators, it might end up generating a concentration of positive returns, which is a region of high expected return. If this region exists, with the Pattern Searcher method we could cluster it using geometric figures, which in this case is a box of center c and edge e.

Solution: Use of Genetic Algorithm to find the edges *e*s and the centers *c*s of each indicator, in case of a box, in order to maximize the returns for a given historical dataset.

How to use the system for decision making: The agent will only open a position if the current indicator signals are inside the geometric primitive.

In the situation where the geometric primitive is a box, we can express the rules derived from the Pattern Searcher in the form of Boolean expressions. This is an example of how the rule would looks like: **if** $MACD(x_1,x_2,x_3) > y_1$ **and** $RSI(x_4) < y_2$ **then** BUY. This Boolean expression means that it opens a buy position if *MACD* signal is above y_1 and RSI signal is bellow y_2. Notice that, beside the specification of the indicators, this expression have two kinds of parameters: the indicator‘s parameters (x_1,x_2,x_3,x_4) and the signal thresholds (y_1,y_2). Our model focuses primarily on finding the signal threshold *y*s, while fixating the values of the indicator‘s parameters *x*s. Even though the indicator‘s parameters could be optimized simultaneously with the signal threshold, it would increase the complexity of the model and also the chance of overfitting ([1]), since too many parameters would be optimized at once. Thus, keeping our model as simple as possible may help to prevent overfitting.

We have defined the domain of the center and the edge of the Pattern Searcher model as $-1 \leq c \leq 1$ and $0 \leq e \leq 2.4$, respectively. All the indicators were normalized within the interval from -1 to 1, and by this, the geometric figure can cover the whole space, in case of $c = 0$ and $e = 2.4$, for example. This allows our model to not only optimize the parameters but also to do *feature selection*. It means that poor features added to the model can be automatically removed during training if the figure covers completely the feature dimension.

Geometric primitive

The regions with highly expected returns within the multidimensional space of indicators may have different shapes, so different geometric figures could be used. In this work, we have implemented ellipses and boxes:

Ellipse: It can be represented by a center *c* and a radius *r* for each indicator: $Ellipse = (c_1,r_1,c_2,r_2 \ldots c_n,r_n)$. The region inside an ellipse is given by Equation (4.1):

$$\frac{(c_1-r_1)^2}{r_1^2} + \frac{(c_2-r_2)^2}{r_2^2} + \ldots + \frac{(c_n-r_n)^2}{r_n^2} \leq 1 \tag{4.1}$$

Box: It can be represented by a center *c* and an edge *e* for each indicator: $Box = (c_1,e_1,c_2,e_2 \ldots c_n,e_n)$. The region inside a box and the trade rule can be given by a Boolean expression:

$$\begin{aligned} \textbf{If} \quad & (c_1 - e_1/2 < indicator_1 < c_1 + e_1/2) \quad \textbf{and} \\ & (c_2 - e_2/2 < indicator_2 < c_2 + e_2/2) \quad \textbf{and} \end{aligned}$$

...

$(c_n - e_n/2 < indicator_n < c_n + e_n/2)$ **then**

Open Position($TYPE$)

In our experiments, we have used boxes only. The reason for this is that the possibility of representing the trade rules by Boolean expressions makes easier their interpretation.

Indicators and normalization

The dimensions of the multi-dimensional space can be represented by technical and customized indicators. We have attempted to chose and create indicators that capture different information on the price movement.

Each indicator can produce very different outputs signals. The *RSI* for example, gives a value in a scale form from 0 and 100, while the *MACD* returns a value within $\mathbb{R}$. On the other hand, the moving averages (*MA*s) signal is not a value, but a curve superimposed on price. Thus it is essential doing a normalization of the signals of the indicators. We have normalized all of them within the interval from -1 to 1.

For some indicators such as *RSI*, normalization is straightforward, by just applying a "rule of three". For others, like *MA*s, we have slightly modified them without losing the original information that the indicator was designed for. The technical and the customized indicators used in this work, with details about its normalization, were:

1. **RSI**: Relative Strength Index (RSI) is a momentum indicator that measures if the price is oversold or overbought. The signal varies from 0 to 100. For normalization, we have applied a "rule of three".

$$signal = \frac{RSI}{50} - 1 \tag{4.2}$$

2. **Volume**: It is the number of contracts traded in an asset during a given period of time. For normalization we have applied the formula:

$$signal = \frac{Volume - MA(Volume)}{25000} \tag{4.3}$$

This formula does not use the absolute value of volume but the variation of volume instead, showing if the volume is increasing or decreasing.

3. **MACD**: Moving Average Convergence Divergence (MACD) measures the relationship between two moving averages of an asset. We have applied the normalization:

$$signal = \frac{MACD \times 150}{price} \tag{4.4}$$

We have normalized the MACD histogram by dividing it by the price.

4. **MA**: Moving Average (MA) is a trend-following indicator that smooth out the price by filtering out the noise. For normalization we have applied the formula:

$$signal = \frac{price - MA \times 100}{price} \tag{4.5}$$

A difference between the price and the moving average gives the idea if the price is moving up or bellow the MA, and the distance between them. It was normalized by the price.

5. **Time**: Day time in minutes. The normalization was done using a "rule of three".

$$signal = \frac{2 * ((Time.hour \times 100 + Time.minute) - 900)}{900} - 1 \tag{4.6}$$

6. **ADX**: Trend Strength Indicator (ADX) determines when the price is trending strongly.

$$signal = \frac{DI_+ - DI_-}{|DI_+ - DI_-|} \times \frac{ADX}{60} \tag{4.7}$$

If the $DI_+ > DI_-$, the signal is positive, otherwise it is negative. It is useful to indicate the direction of the trend.

7. **MFI**: Money Flow Index (MFI) is an oscillator that uses price and volume for identifying if the price is overbought or oversold. The normalization was done using a "rule of three".

$$signal = \frac{MFI}{50} - 1 \tag{4.8}$$

8. **STO**: Stochastic Oscillator (STO) is a momentum indicator that measures if the price is oversold or overbought. The signal varies from 0 to 100. The normalization was done using a "rule of three".

$$signal = \frac{STO}{50} - 1 \tag{4.9}$$

9. **Pivot points**: Lines that show important price levels calculated by using the high, low and close prices of the previous day.

$$signalR1 = \left(\frac{PivotR1 - PivotS1}{price - PivotS1} - 0.5 \right) * 1.5 \tag{4.10}$$

$$signalR2 = \left(\frac{PivotR2 - PivotS2}{price - PivotS2} - 0.5 \right) * 1.5 \tag{4.11}$$

$$signalR3 = \left(\frac{PivotR3 - PivotS3}{price - PivotS3} - 0.5 \right) * 1.5 \tag{4.12}$$

It indicates how much the price has gone up or down, using the pivot lines as reference.

10. **BBS**: Bollinger Bands Squeeze (BBS) uses the indicators Bollinger Bands and the Average True Range (ATR) to know if an asset price has high volatility or if it is congested, that is walking sideways in a given period of time. It is calculated by:

$$signal = \left(\frac{HighBollingerBands - LowBollingerBands}{2 \times ATR} - 1 \right) \times 0.5 \tag{4.13}$$

11. **MA9**: It signalizes the direction of a hull moving average.

$$signal = \frac{\Delta MA \times 450}{price} \tag{4.14}$$

Model optimization using Genetic Algorithm

We have stated the problem of finding the parameters of the geometric primitive, which are the boxes centers and edges for each indicator. To accomplish that, we have used Genetic Algorithm, as it has shown to be very efficient in optimizing parameters in a variety of problems ([37]). In the GA perspective, the problem resumes in "finding the optimal parameters in order to maximize the profits". The process was divided into: *Setting the agent parameters*, and *starting the optimization*. Fig. 4.3 shows these steps and their details.

Before the optimization, we have *set the agent parameters*. Firstly we have configured the agent strategy variables regarding risk management, position management and order management, which includes variables such as take profit, stop loss and volume of contracts. The values of those parameters were chosen accordingly to some prior experience with trading strategies design and the desirable outcome. About the later, we desire to capture large movements of the market, since it would improve the returns per entry while decreasing the brokerage costs. Because of this, we have set large take profits and stop losses, around 1% of the asset price. Next, we have configured the *pattern searcher parameters*. Here, we have chosen the indicators to be used, the trade type (buy or sell) for each pattern and the

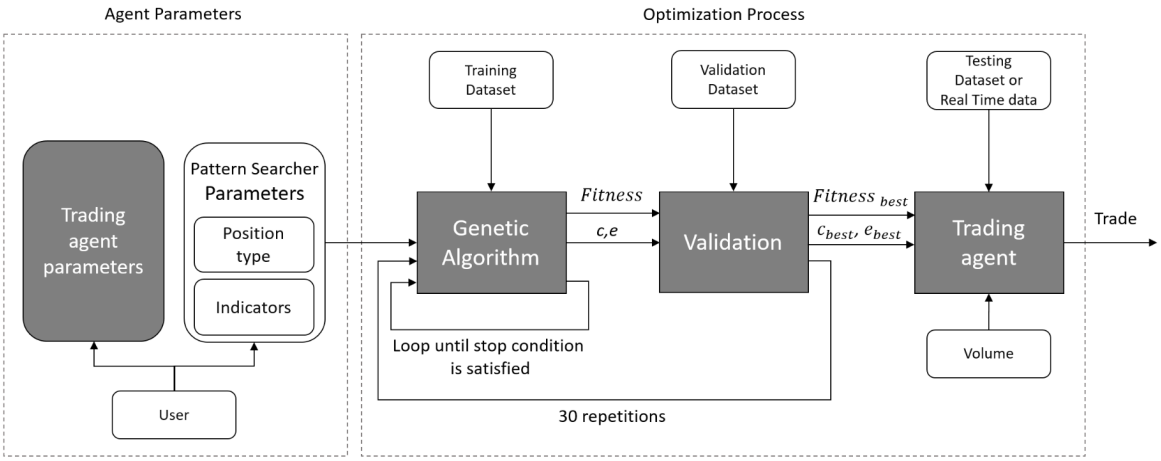


Figure 4.3: Pattern Searcher optimization diagram. After all the parameters are configured, the agent's Pattern Searcher is trained on a historical dataset using GA. The best individual is taken for validation and testing on new unseen data.

number of patterns the agent will trade (up to 4 patterns). Each pattern was trained separately, attempting to reduce the number of the GA input variables, speeding up the training. Figure 4.4 shows the MT5 panel with the Pattern Searcher parameters.

Variable	Value	Start	Step	Stop
********** Pattern Detection **********				
Use Patterns	true	false		true
Use Pattern 1	true	false		true
Pattern 1 type	[1] SELL	[0] BUY		[1] SELL
--- Pattern 1 - Input x1	[9] p_Stoch	[0] p_Vol		[8] p_Candle
--- Pattern 1 - Input x2	[10] p_Pivot	[0] p_Vol		[8] p_Candle
--- Pattern 1 - Input x3	[5] p_ADX	[0] p_Vol		[8] p_Candle
--- Pattern 1 - Input x4	[2] p_RSI	[0] p_Vol		[8] p_Candle
--- Pattern 1 - Input x5	[13] p_Bands	[0] p_Vol		[8] p_Candle
--- Pattern 1 - Input x6	[11] p_None	[0] p_Vol		[10] p_Pivot
--- Pattern 1 - Input x7	[11] p_None	[0] p_Vol		[10] p_Pivot
--- Pattern 1 - Input x8	[11] p_None	[0] p_Vol		[10] p_Pivot
------ Pattern 1 - Input x1 - Val	0.90383	**-1.2**	**0.00001**	**1.2**
------ Pattern 1 - Input x2 - Val	0.83641	**-1.2**	**0.00001**	**1.2**
------ Pattern 1 - Input x3 - Val	0.28007	**-1.2**	**0.00001**	**1.2**
------ Pattern 1 - Input x4 - Val	-0.31331	**-1.2**	**0.00001**	**1.2**
------ Pattern 1 - Input x5 - Val	0	**-1.2**	**0.00001**	**1.2**
------ Pattern 1 - Input x6 - Val	0	-1.2	0.00001	1.2
------ Pattern 1 - Input x7 - Val	0	-1.2	0.00001	1.2
------ Pattern 1 - Input x8 - Val	0	-1.2	0.00001	1.2
**********				
------ Pattern 1 - Input y1 - Val	0.68156	**0**	**0.00001**	**1.2**
------ Pattern 1 - Input y2 - Val	0.50494	**0**	**0.00001**	**1.2**
------ Pattern 1 - Input y3 - Val	0.02015	**0**	**0.00001**	**1.2**
------ Pattern 1 - Input y4 - Val	0.83025	**0**	**0.00001**	**1.2**
------ Pattern 1 - Input y5 - Val	0	**0**	**0.00001**	**1.2**
------ Pattern 1 - Input y6 - Val	0	0	0.00001	1.2
------ Pattern 1 - Input y7 - Val	0	0	0.00001	1.2
------ Pattern 1 - Input y8 - Val	0	0	0.00001	1.2

Figure 4.4: Pattern Searcher panel on MT5. The inputs *x*s and *y*s in this panel correspond to the centers and edges, respectively. They were selected for optimization, where its value can assume a number from *Start* to *Stop*, with step of *Step* (see the 3 columns on the right).

After all the parameters were chosen, we finally started the *optimization process*. It was done using the Metatrader 5 Genetic Optimizer ([10]). Since it is a built-in MT5 tool, we could not have access to the crossover and mutation types and rates. The optimization consisted in *training*, *validating*, and *testing*.

In the *training*, the Genetic Algorithm evolved near to 40 *generations* of individuals until the stop criteria were satisfied. The *individuals'* chromosome were represented by the centers (*c*s) and edges (*e*s) of each indicator that compounded an agent pattern. The *c*s' domain varied from -1 to 1, and the *e*s' domain from 0 to 2.4. The *initial population* was generated with a range from 64 to 256 individuals ([Corp]), each of them representing a solution. All individuals were evaluated through a trading simulation onto the stock historical data and received a value of fitness that express the quality of the strategy. Our *Fitness function* was a combination between total profit and drawdown, given by: $Fitness = TotalProfit \times Drawdown$. It measures a relationship between profitability and risk, leading the GA not only to evolve individuals that have great profits but also individuals with low drawdown.

In the *validation*, we have checked the agent's performance on unseen data, quantified by the same fitness function used for training. It is a very important step for removing agents that have probably suffered from overfitting. Agents that passed the validation have higher chances to be more robust and perform well in real trading. Since GA is a stochastic process, we have repeated the whole training and validation processes for 30 times, and the best individual from validation was taken for *testing*.

In the *testing*, we have evaluated the performance of the agents in a new unseen data, which reflects what we would expect in a real account trading.

Other implementation details

There are some other relevant considerations that worth to be discussed, which are: the number of simultaneously positions allowed per agent, the inverse condition being satisfied, and the simplicity of the system flow.

The agents are only allowed to have one opened position at each time. This results in uncorrelated patterns within an agent. The reason for this is because correlated pattern signals would probably overlap and their trade signal would be ignored. Consequently, there would not be any improvement in the agents' performance. The only way to improve the total profit while not violating the restriction of contracts per agent would be finding patterns in which signals do not overlap. It would significantly improve the agent's efficiency.

In addition, the agents were programmed to close an open position if has passed 2 hours after the position was opened, or if the inverse condition was satisfied. About the later,

if an agent has a buy position, and one of its patterns sends a signal to sell, for example, then the agent closes the buy position and opens a sell position.

Indeed, the implementation exhibited a desirable property compared to some Machine Learning methods, which is the simplification of the system flow. The model and the strategy are interconnected, in a way that the training directly optimizes the agent's performance in terms of profitability.

4.3 Controlled Leverage

Leverage is when an investor uses borrowed money in an attempt to increase the ratio of return of an investment ([15]). A leverage ratio of 1:10, for example, means that we can negotiate 10 times more cash of what we have in the broker's account. This can potentially increase the returns, but also the losses.

One can ask: "How do we calculate the initial investment needed to trade with leverage, in order to have great returns while minimizing the risks?" To answer this question, we should not think about maximizing the returns but think about controlling the risks. What we want is a trade-off between returns and risk. Aiming to calculate the initial deposit, we have proposed a method that we will refer to as "Controlled Leverage". We have not found much information about this in literature, but we believe that many professional traders use a similar approach.

Controlled Leverage calculation

The drawdown (*DD*), which is the worse consecutive loss, needs total attention because it is capable of breaking an account balance if there is not enough margin to trade in that period (see Figure 4.5). Brokers may ask little margin per contract of Future assets, for example, which allows people to start trading with a very small amount of money. However, to reduce the risks and "survive" from the *DD*, we should invest more money per contract than the margin asked by the broker.

In this work we estimated the amount of initial investment ($BalanceNeeded$) based on the *DD* of the backtest ($DrawnDown$), defined by (4.15):

$$BalanceNeeded = DrownDown * RiskCoef \tag{4.15}$$

Where the $RiskCoef$ is a risk coefficient. If the $RiskCoef$ is set as 5, it means that we will have 5 times more money in the account than the *DD* from backtesting. The higher the

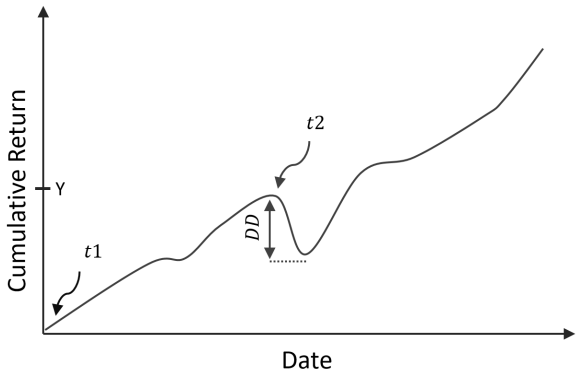


Figure 4.5: Typical cumulative return growth.

value we chose, the less will be the risk of breaking an account. We could also rewrite this formula by (4.16):

$$BalanceNeeded = DrawnDown/TradePct \tag{4.16}$$

Where $TradePct$ represents the percentage of the initial deposit that we are effectively trading. Lower $TradePct$ results in higher returns but also higher risks.

Monthly Return

Once the initial deposit was estimated and we have the monthly profit ($MonthlyProfit$), we can calculate the monthly return by (4.17):

$$MonthlyReturn = MonthlyProfit/BalanceNeeded \tag{4.17}$$

Notice that the return is inversely proportional to the DD. Lower DD reduces the $BalanceNeeded$, and consequently increases the Monthly Return. Aiming to decrease the DD, we have generated strategy portfolios, which are described in Section 4.4.

4.4 Strategy Portfolio

According to the Controlled Leverage approach, the return and the amount needed for trading is inversely proportional to the DD. Minimizing the DD can potentially increase the profitability. It can be done by combining single strategies in a "strategy portfolio", in the sense that, instead of negotiating all contracts in one single strategy, we would distribute those contracts among a set of strategies. Depending on how their drawdown curves are, the combined DD can be smaller than investing all contracts in one individual strategy (Figure 4.6). If one strategy is having a sequence of losses, another might be having a sequence of

gains to compensate. A portfolio can also decreases the variance of the returns, which is desirable.

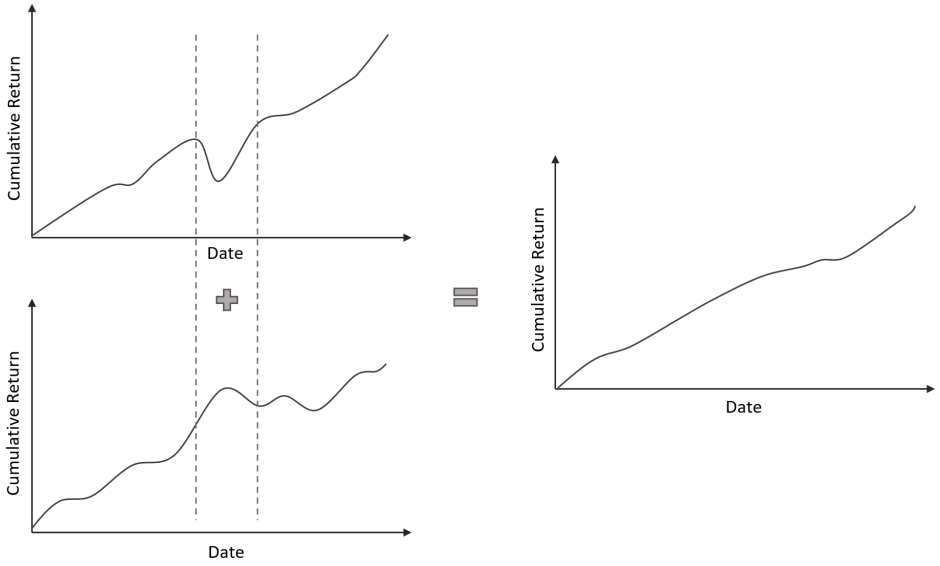


Figure 4.6: Typical cumulative return of a strategy portfolio.

The weight w represents the percentage volume that each agent within a portfolio will trade. The weights were optimized through Genetic Algorithm, using the Excel's Solver tool. Figure 4.7 synthesize the portfolio's training elements.

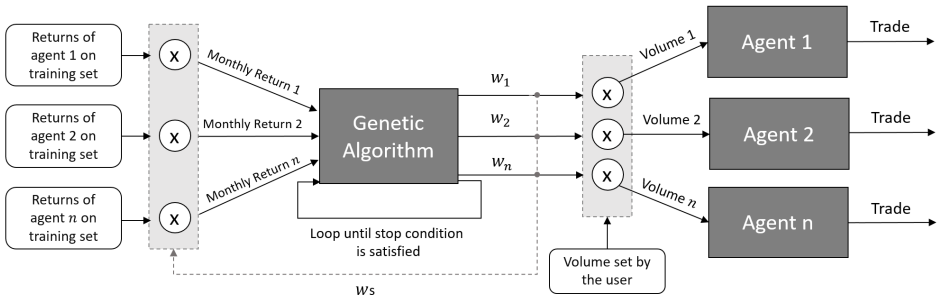


Figure 4.7: Portfolio optimization diagram. The GA individuals are represented by weights. At each generation, the weights multiply the agents' historical financial returns, to generate portfolios with w's distribution. We calculate the monthly returns of those portfolios and use them as the GA fitness function.

The *individuals* were represented by an array with the portfolio weights (w's). w's could assume R values from 0 to 1. The *initial population* was generated with 100 individuals, that were decoded in solutions for the problem. It was done by, firstly, normalizing the weights with (4.18):

$$w_i = \frac{w'_i}{\sum_{i=1}^{n} w'_i} * 100\% \tag{4.18}$$

Where the w_i value is the percentage of contract volume for each agent, and $\sum_{i=1}^{n} w_i = 100\%$. Then, the w_i was multiplied to the return distribution of the agent i, resulting in portfolios with w weighted distribution. Each portfolio was evaluated by a *fitness function*, which in this case is the monthly return. We have considered it an interesting fitness measure because its calculation incorporates both the drawdown and the total profit. Consequently, the GA would look for a trade-off between risk and return. In addition to the usual GA setup, we have incorporated a restriction that limits the standard deviation of the weights. This would lead to more balanced weights, without too large or too little w values (that would resulting in removing strategies from the portfolio and increasing the chance of overfitting). The *crossover* and *mutation* rates were 92% and 8%, respectively. The algorithm have ran for a total of 30 *generations*, and the best individual was taken for validation and testing.

We have not explored the GA parameters and its effects on training quality, since it was not in the scope of this project.

Chapter 5

Experiments: Results and analysis

In this chapter, we present the experimental validation of the trading agents and the agent portfolios trained via Genetic Algorithm. We have organized the results and analysis in 4 parts. In the first part, we explain *general considerations*, including the platform and data used to train the agents, the trading costs and the Brazilian benchmarks. In the second part, we describe the *portfolios' generation*, giving details about the generation of 3 different portfolios. In the third part, we show *the agent patterns* obtained through GA. In the fourth part of the analysis, we discuss the *agents' and portfolios' performance* without and with leverage, in terms of profitability of the testing set. We show the cumulative profit graphs and compare the results among the agents, portfolios, and benchmarks.

5.1 General considerations

In this section we have discussed about general considerations regarding to the experiments, including the platform and dataset used, the initial deposit and trade margins we have used, the trading costs, the Brazilian benchmarks, and the normalization of contracts.

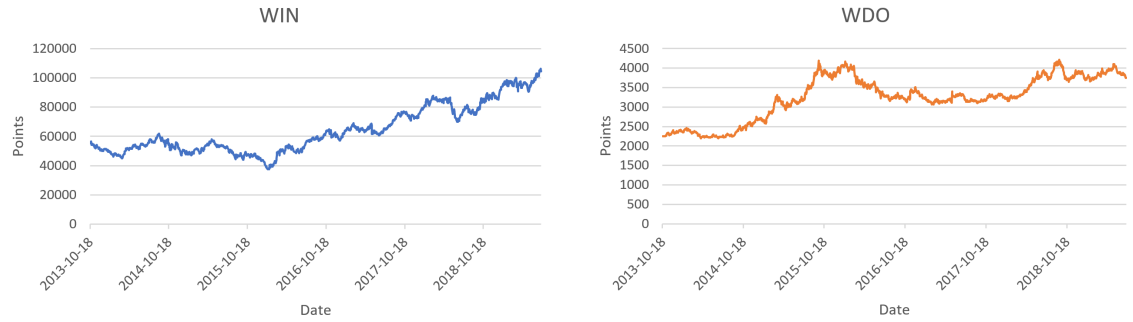
Platform

Most of the implementations and tests were done using the trading platform Metatrader 5 and its programming language called MQL5. They allow the development of trading robots, performing backtests and trading with a real account.

Dataset

The dataset where we have run the model, consisted of the Brazilian BM&F Mini Ibovespa Futures (WIN) and Mini Dollar Futures (WDO). Each WIN and WDO point corresponds to

R$ 0.20 and R$ 10.00, respectively. They were chosen because of their high liquidity, return potential, low brokerage fees and high leverage allowed by the broker. The data contained candlesticks of different timeframes with volume, open, high, low and close prices. Figure 5.1 shows the historical prices of those datasets.



(a) WIN **(b)** WDO

Figure 5.1: Historical quotation of the Brazilian BM&F Mini Ibovespa Futures and Mini Dollar Futures from 2013 to 2019.

The data period chosen for the experiments went from 2013-10-18 to 2019-07-12. The data was split as following:

1. 2013-10-18 to 2017-01-01: Data set for training;
2. 2017-01-01 to 2018-01-01: Data set for validation;
3. 2018-01-01 to 2019-07-12: Data set for testing.

Those same periods were used both for training the Pattern Searcher model and training the portfolio weights.

Initial deposit and trade margin

For our analysis, we have considered an initial deposit of R$100,000.00 and $TradePct = 10\%$, which means that we are using R$10,000 for trade.

Trade costs

We have considered all the trading costs, totaling R$0.36 per contract (see Table 5.1). We have applied R$ 0.48 in the experiments.

Table 5.1: Costs per contract.

Cost	Value per contract
Operational Fee	R$ 0.11
Register Fee	R$ 0.17
BM&F Fee	R$ 0.08
Total	R$ 0.36

Brazilian Benchmarks

There are 2 important Brazilian benchmarks: SELIC (rate of risk-free return) and IBOV (floating income rate).

SELIC: is the Brazilian interest rate. It had an average annualized rate of 6.46% during the period between 2017-12-07 and 2019-07-31 ([2]).

IBOV: is the Brazilian stock market index. It had a growth of 48% from 2018-01-01 to 2019-07-12, that is approximately 31% per year, and had a maximum cumulative fall, or "drawdown", of 23%.

Contract's normalization

Our agents' stop-loss has not a fixed size, instead, it is proportional to the current price (e.g., $stopLoss = 1\% \times price$). If we have used a fixed stop-loss, the higher the price, the greater would be the price variation (we have assumed that the asset volatility is proportional to the current price) and the stop-loss would be more likely to be reached. Based on our assumption, making the stop-loss being proportional to the price seems to work better.

The proportional stop-loss, however, comes with the disadvantage of not having a fixed financial loss in terms of cash, as it depends on the price. Let's suppose that we have a stop loss of 1% over the prices R$50,000.00, R$100,000.00 and R$200,000.00. Their financial stop-losses would be R$500.00, R$1000.00 and R$2000.00, respectively. The agent would be losing more by each stop for higher prices, and the drawdown would be probably higher. And in this case, we could not estimate the future drawdown based on the past drawdown.

In order to normalize the contracts, we have fixated the financial loss by firstly choosing a price $p1$ that we will trade 1 contract, and make the volume of contracts being inversely proportional to the stop-loss size. This would be the same as making the volume of contracts being inversely proportional to the price. See equation 5.2.

$$volume\_of\_contracts = \frac{p1}{price} \tag{5.1}$$

We believe that by this approach, we would have stop-loss size more appropriated to the current volatility at that moment, while having a fixed financial stop-loss.

In the experiments, we have used $p1_{WIN} = 100,000.00$ points or $R\$20,000.00$, and $p1_{WDO} = 3500.00$ points or $R\$35,000.00$.

The number of contracts of Table 5.16 from Section 5.4 was calculated by:

$$contracts = \frac{InitialDeposit}{p1} \tag{5.2}$$

That is the number of contracts each agent can trade considering the initial deposit and the asset price for when the contract is 1 ($p1$).

Pattern Searcher training

We have repeated the training process 30 times for each agent. Figure 5.2 shows the training curve of Agent 1 considering the 30 training repetitions. We can see that the fitness increases after each generation, indicating that the genetic algorithm is successfully evolving the individuals.

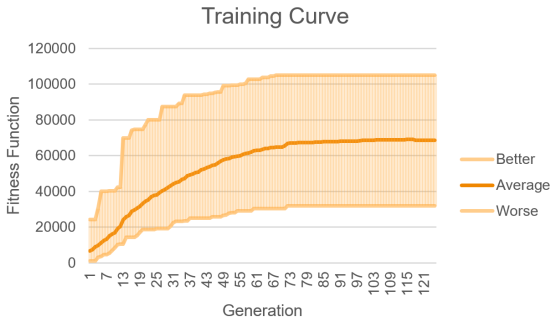


Figure 5.2: Training curve of Agent 1 considering the 30 training repetitions.

Figure 5.3: Box-plot showing the total net profit and the drawdown of the best individual from each of the 30 training repetitions, performing in the validation set.

Figure 5.3 is a box-plot showing the total net profit and the drawdown of the best individual from each of the 30 training repetitions, performing in the validation set. This is interesting for evaluating the robustness of the model. As we can see, the individuals' total net profit was around 15% on average, however, the standard deviation was high.

5.2 Portfolios' generation

We have used the agents generated by the Pattern Searcher model to create 3 strategy portfolios: Portfolio A, Portfolio B, and Portfolio C. In each of them, the weights were determined differently.

In the Portfolio A, the weights are equal, with each agent trading the same volume of contracts. In the Portfolio B, we have used the Genetic Algorithm without restricting the weight's standard deviation. In the Portfolio C we have optimized the weights such as the Portfolio B optimization, however with the restriction that limits the standard deviation of the weights. The Table 5.2 shows the *w*s' distribution.

Table 5.2: Portfolio's weights (w_s).

Weight	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	Total	Std	Description
Strategy Portfolio A	12.50%	12.50%	12.50%	12.50%	12.50%	12.50%	12.50%	12.50%	100.00%	0.00%	Equal w_s
Strategy Portfolio B	29.87%	12.81%	4.78%	18.82%	18.98%	3.33%	1.95%	9.46%	100.00%	9.62%	w_s trained via GA
Strategy Portfolio C	13.41%	9.56%	12.57%	12.71%	18.08%	10.28%	10.17%	13.22%	100.00%	2.71%	w_s trained via GA with std restriction

5.3 Analysis of the agent patterns

We have trained 8 agents, 5 on WIN and 3 on WDO, using the Pattern Searcher model described in Section 4.2. Each of them received different parameters, indicators, timeframes, and symbols. It yields to improve the diversity of the found patterns, leading to agents that capture the most different behaviors of the market. Also, it would further improve the performance of the portfolios by having more uncorrelated strategies. After training, the agents were selected based on their performance in both training and validation, and then applied in the testing set.

The experiments were focused on day trading. In this case, positions are opened and closed on the same day (one can never sleep with a stock bought or sold). If there is an open position, the program exits right before the market closes at the end of the day.

In Tables 5.3 to 5.7 we show the agent parameters that were used for training.

Table 5.3: Agent parameters A*.

A*:	TP/SL type:	percentual of the entry price
	TakeProfit:	0.08%
	StopLoss:	0.008%
	Order placement:	placed 3 ticks after the candle close price
	Close position:	at the end of the day
		120 minutes after the position was opened
		if inverse condition is satisfied

Table 5.4: Agent parameters B*.

B*:	TP/SL type:	percentual of the entry price
	TakeProfit:	0.08%
	StopLoss:	0.008%
	Order placement:	placed 3 ticks after the candle close price
	Close position:	at the end of the day
		if inverse condition is satisfied

Table 5.5: Agent parameters C*.

C*:	TP/SL type:	percentual of the entry price
	TakeProfit:	0.006%
	StopLoss:	0.005%
	Order placement:	placed in the max/min of the last 3 candles (buy order in the max)
	Close position:	at the end of the day

Table 5.6: Agent parameters D*.

D*:	TP/SL type:	percentual of the entry price
	TakeProfit:	0.006%
	StopLoss:	0.005%
	Order placement:	placed in the max/min of the last 5 candles (buy order in the max)
	Close position:	at the end of the day

Table 5.7: Agent parameters E*.

E*:	TP/SL type:	percentual of the atr(14) indicator
	TakeProfit:	1.3%
	StopLoss:	2.6%
	Order placement:	placed in the max/min of the last 25 candles (buy order in the max)
	Close position:	at the end of the day

In Tables 5.8 to 5.15 we show the entry patterns expressed in a form of Boolean expressions. Then, we analyzed how those patterns work.

Agent 1 - Pattern 1: It is a reversion setup. It sells near the superior Pivot line R3 when stochastic is overbought and ADX has a slight uptrend. Possibly, it is shortening at the end of an uptrend, and the beginning of a downtrend.

Table 5.8: Agent 1's patterns.

Agent 1					
Symbol:	WIN				
Timeframe:	m5				
Parameters:	A*				
Pattern 1:					
if	-1.00<RSI(6)<0.52	**and**	0.26<ADX(14)<0.30	**and**	
	0.22<STO(14)<1.00	**and**	0.33<PvtR3()<1.00	**then**	SELL
Pattern 2:					
if	-1.00<RSI(6)<-0.57	**and**	-1.00<MA(1,28)<-0.80	**then**	BUY

Agent 1 - Pattern 2: It is a reversion setup. It buys when RSI is oversold and the fast MA is far below the slow MA. In other words, it buys after the price has done a large down movement. That might indicate a higher chance that the price is at the end of a downtrend and the beginning of an uptrend.

Table 5.9: Agent 2's patterns.

Agent 2					
Symbol:	WIN				
Timeframe:	m15				
Description:	A*				
Pattern 1:					
if	0.10<MACD(20,65,9)<0.34	**and**	-0.89<STO(14)<-0.23	**and**	
	-0.59<PvtR3()<1.00	**then**	BUY		
Pattern 2:					
if	0.15<RSI(6)<1.00	**and**	-1.00<Candle()<-0.30	**and**	
	0.25<STO(14)<1.00	**and**	-0.49<PvtR3()<1.00	**then**	SELL
Pattern 3:					
if	-0.11<MACD(20,65,9)<1.00	**and**	-0.80<RSI(6)<1.00	**and**	
	-1.00<PvtR3()<0.96	**then**	BUY		

Agent 2 - Pattern 1: It is a reversion setup. It buys when MACD is positive, Stoch is oversold and the price is above the Pivot line S3. The positive MACD signal may indicate that the price is getting and up acceleration after a down movement (Stoch oversold). Strong downtrends that last the whole day might cross down the S3 line. As it only buys above the S3, it may escape from strong downtrends.

Agent 2 - Pattern 2: It is a reversion and impulsive setup. It sells when the Stoch and RSI are overbought and there is a formation of a long downside candle, which may indicate the entry of volatility in the market, by news for example. In addition, it only sells if the price is above the pivot R3, which means that, from a macro perspective, the price still has a lot of room for a downtrend.

Agent 2 - Pattern 3: It buys when MACD is high, RSI is not strongly oversold and the price is below the pivot line R3.

Table 5.10: Agent 3's patterns.

Agent 3					
Symbol:	WIN				
Timeframe:	m10				
Parameters:	A*				
Pattern 1:					
if	0.21<RSI(6)<1.00	**and**	0.38<ADX(14)<1.00	**and**	
	-1.00<PvtR1()<0.72	**and**	-0.61<PvtR2()<1.00	**then**	BUY
Pattern 2:					
if	-0.54<RSI(6)<1.00	**and**	0.13<Candle()<0.94	**and**	
	-1.00<STO(14)<-0.19	**and**	0.03<PvtR3()<1.00	**and**	
	-0.48<PvtR1()<1.00	**then**	BUY		
Pattern 3:					
if	-1.00<MACD(16,58,9)<0.96	**and**	-0.79<RSI(6)<1.00	**and**	
	0.15<MA(1,28)<1.00	**and**	-0.64<MFI(5)<-0.05	**and**	
	-1.00<Candle()<0.03	**and**	-0.86<PvtR3()<1.00	**and**	
	-1.00<PvtR1()<0.03	**then**	SELL		

Agent 3 - Pattern 1: It is a trend-following setup. It buys when ADX signalizes an uptrend, and the RSI is a little oversold, which means that the price is doing an up movement. In addition, the price must be between the Pivot lines S2 and R1.

Agent 3 - Pattern 2: It is a reversion setup. It buys when the Stoch is oversold and there is a formation of a long upside candle, which may indicate the entry of volatility in the market, by news for example. The RSI should not be oversold, and as it has a smaller window size compared to the Stoch, the interpretation here is that, although the price is oversold (by the Stoch indicator), it already started going up from micro a perspective (by the RSI not being oversold).

Agent 3 - Pattern 3: It sells when the price is going up (MA is higher than zero), MFI is smaller than zero, and there is a formation of a downside candle between the Pivot lines S3 and P. It seems to be a weak setup, with very few entries.

Table 5.11: Agent 4's patterns.

Agent 4:					
Symbol:	WIN				
Timeframe:	m10				
Parameters:	A*				
Pattern 1:					
if	-0.71<RSI(6)<1.00	**and**	0.37<ADX(14)<0.78	**and**	
	-0.23<MA9(15)<1.00	**and**	-1.00<PvtR1()<0.68	**then**	BUY
Pattern 2:					
if	-1.00<RSI(6)<0.97	**and**	-0.29<MACD(16,58,9)<0.89	**and**	
	-0.67<ADX(14)<1.00	**and**	-0.31<STO(14)<1.00	**and**	
	-1.00<PvtR3()<-0.06	**and**	-0.19<BBS(12,1.5,8)<1.00	**then**	SELL

Agent 4 - Pattern 1: It is a trend-following setup. It buys when the price is below

the Pivot line R1, and has an uptrend, signalized by the high value of ADX and the positive HullMa value. Looking at the macro perspective, the price being below R1 might mean that it has gone up little, so there is still plenty of room for a strong uptrend. In this situation, it would open the position at a low price point, having a great statistical advantage.

Agent 4 - Pattern 2: It sells when the price is below the pivot line R3, the Stoch is not oversold, and the MACD does not indicate a strong down acceleration. Here it bets that the price might go down if there is not a strong signal showing an uptrend.

Table 5.12: Agent 5's patterns.

Agent 5					
Symbol:	WIN				
Timeframe:	h1				
Parameters:	B*				
Pattern 1:					
if	0.00<MA9(9)<1.00	**and**	-1.00<BBS(17,1.5,8)<0.77	**and**	
	-0.50<STO(14)<0.60	**then**	SELL		
Pattern 2:					
if	-1.00<MA9(9)<0.00	**and**	-1.00<Candle()<0.04	**and**	
	-1.00<STO(14)<-0.67	**and**	-0.77<BBS(17,1.5,8)<1.00	**then**	BUY

Agent 5 - Pattern 1: It is a reversion setup: It sells when there is an uptrend (MA9 greater than zero), the Stochastic is not oversold and overbought neither (maybe losing the uptrend strength), and the price decreases its volatility, starting to walk by side (BBSqueeze not high). In other words, this pattern opens a sell position when an uptrend loses strength.

Agent 5 - Pattern 2: It is a reversion setup. It buys when there is a downtrend (MA9 below zero), the Stoch is oversold and there is a formation of a downside candle. This setup is trying to enter at the bottom of a possibly new uptrend.

Table 5.13: Agent 6's patterns.

Agent 6					
Symbol:	WDO				
Timeframe:	m5				
Parameters:	E*				
Pattern 1:					
if	0.55<RSI(7)<1.00	**and**	-0.89<Time()<1.00	**and**	
	-0.69<STO(14)<1.00	**and**	-1.00<PvtR3()<-0.17	**and**	
	-0.75<BBS(14,2.0,4)<0.41	**then**	BUY		
Pattern 2:					
if	-0.50<RSI(7)<1.00	**and**	-0.68<Time()<1.00	**and**	
	0.48<BBS(14,2.0,4)<1.00	**and**	-0.38<STO(14)<0.80	**then**	SELL

Agent 6 - Pattern 1: Trend following setup: Buys when the price decreases its volatility, starts walking by side (small BBS), it is below the pivot line R3 and a few minutes after

the market has opened. The RSI must be to be overbought, what is a kind of unusual for reversion patterns, however, in this case, the RSI being overbought is used just to signalize that the price, in a short period of time, is going up. In summary, this pattern waits for the price getting congested, and when it moves up with relativity strength, it buys.

Agent 6 - Pattern 2: Sells when the market has high volatility (BBS high), the Stoch and RSI are not oversold, and it has passed several minutes after the market was opened.

Table 5.14: Agent 7's patterns.

Agent 7					
Symbol:	WDO				
Timeframe:	m30				
Parameters:	C*				
Pattern 1:					
if	-0.97<Time()<-0.05	**and**	0.44<MA9(14)<0.82	**and**	
	-0.54<Candle()<0.84	**and**	-1.00<BBS(14,1.0,4)<0.41	**then**	SELL

Agent 7 - Pattern 1: Sells in the morning only, after the market has decreased its volatility, walks by side (small BBS), and started moving up with some strength (high MA9). By analyzing the WIN setups, we would expect this to be a BUY signal because it seems to start a strong uptrend. However, in the WDO the inverse seems to work well for the 30 minutes timeframe.

Table 5.15: Agent 8's patterns.

Agent 8					
Symbol:	WDO				
Timeframe:	h1				
Parameters:	D*				
Pattern 1:					
if	-0.78<Time()<0.06	**and**	0.20<MA9(14)<0.84	**and**	
	-1.00<STO(21)<0.80	**and**	-1.00<BBS(14,1.0,4)<0.24	**and**	
	-1.00<PvtR3()<0.17	**then**	BUY		
Pattern 2:					
if	-0.76<Time()<1.00	**and**	-0.90<MA9(14)<1.00	**and**	
	0.33<STO(21)<1.00	**and**	-1.00<PvtR3()<0.58	**then**	SELL

Agent 8 - Pattern 1: Buys only in the morning and below the pivot line R3, after the market has decreased its volatility, started walking by side (small BBS), and is moving up with some strength (high MA9). Surprisingly, it is the same as the setup Agent 7 - Pattern 1, however that one is a SELL pattern applied to the 30 minutes timeframe, and this one is a BUY pattern applied to the 1 hour timeframe.

Agent 8 - Pattern 2: Sells below the Pivot line R3 when the Stoch is overbought.

We could effectively find strategies with different timeframes and indicators. It yielded to improve the diversity and the trading strategies, as they tends to be less correlated. This is

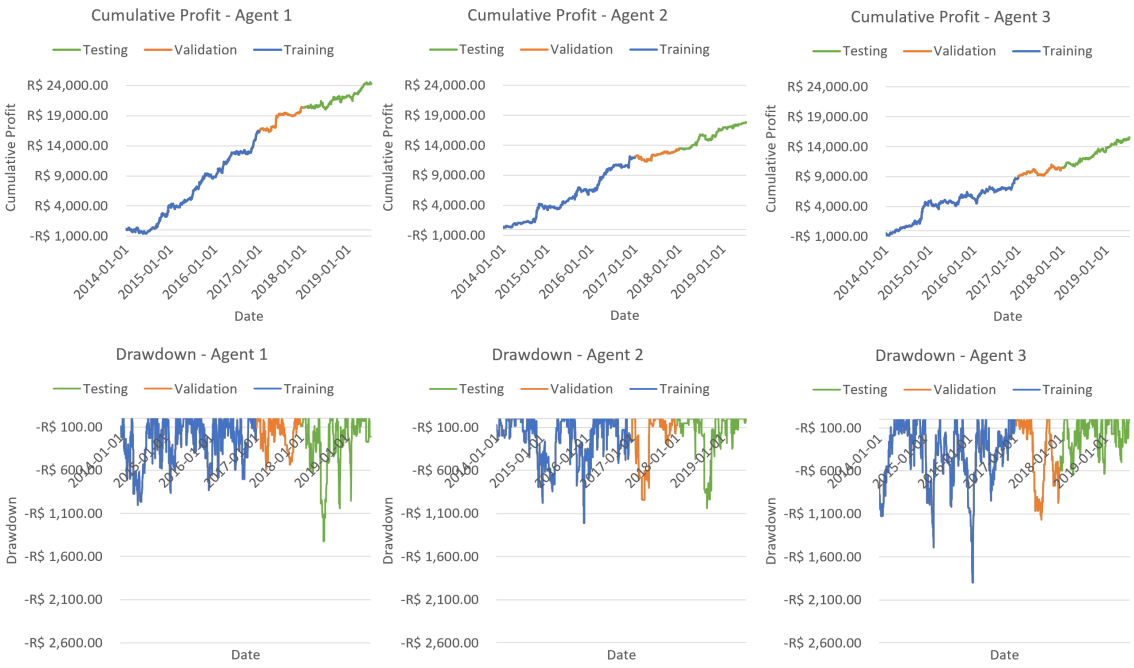
very desirable when composing portfolios, because uncorrelated or inverse correlated strategies may have a greater impact on decreasing the volatility and the drawdown.

5.4 Analysis of the agents' and portfolios' performance

In this section, we show the agents' and portfolios' performance graphs, the results without leverage and the results with leverage in terms of financial metrics. We have compared the agents' performance with and without leverage, the agents' performance with the portfolios' performance, the performance among the portfolios, and compared the agents and portfolios with Brazilian benchmarks: SELIC and IBOV. The entry costs were already applied.

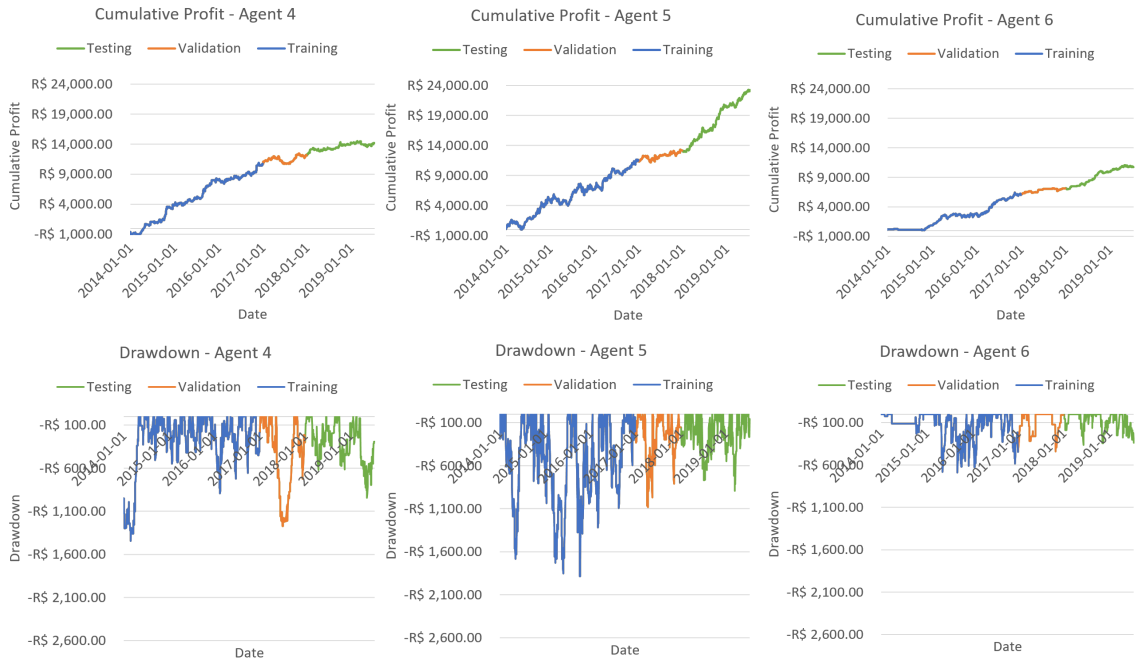
Agents' and portfolios' graphs

Figures 5.4, 5.5, 5.6, and 5.7 show the agents' and portfolios' cumulative profit and drawdown graphs obtained from training, validation, and testing.



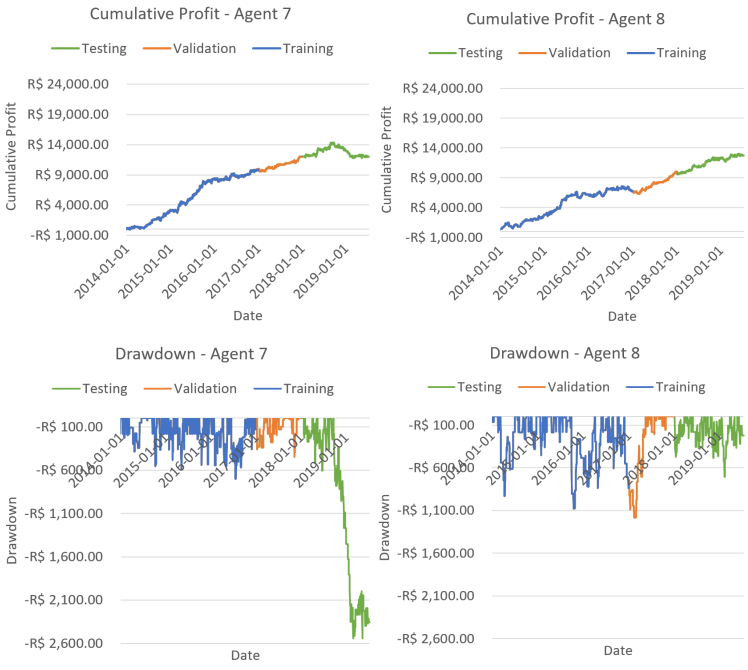
(a) Agent 1 **(b)** Agent 2 **(c)** Agent 3

Figure 5.4: Cumulative profit and drawdown graphs of the agents 1, 2 and 3.



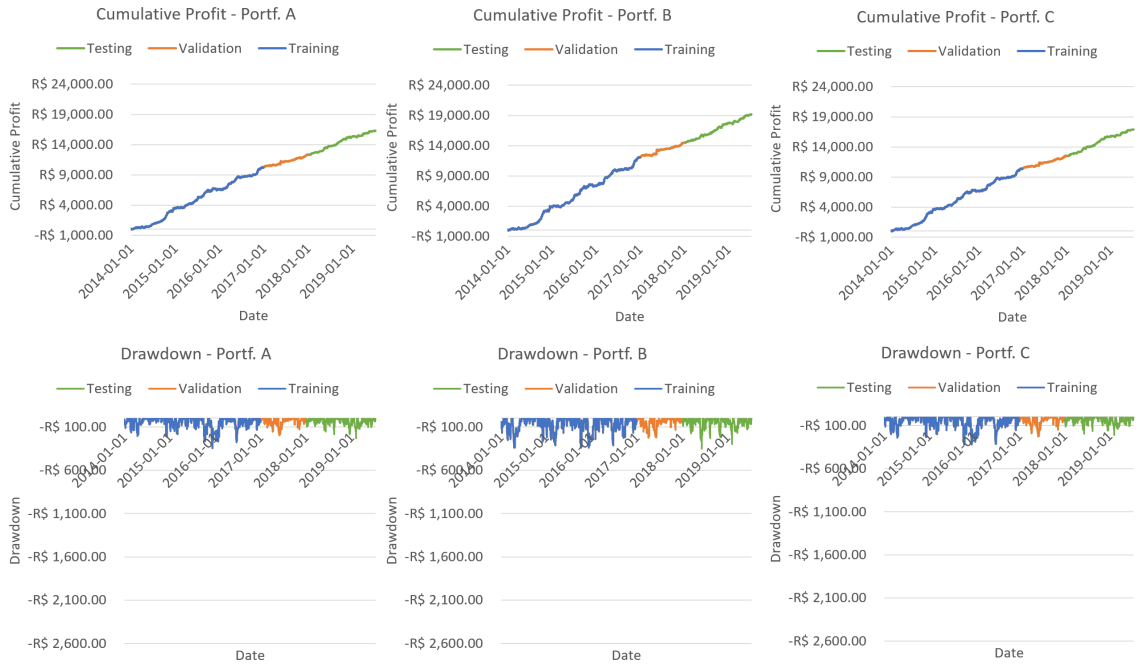
(a) Agent 4 **(b)** Agent 5 **(c)** Agent 6

Figure 5.5: Cumulative profit and drawdown graphs of the agents 4, 5 and 6.



(a) Agent 7 **(b)** Agent 8

Figure 5.6: Cumulative profit and drawdown graphs of the agents 7 and 8.



(a) Portfolio A **(b)** Portfolio B **(c)** Portfolio C

Figure 5.7: Cumulative profit and drawdown graphs of the portfolios A, B and C

We have maintained the same scale among all graphs, to facilitate their visual comparison. In these graphs, the profit was normalized by 1 contract, which means that each agent is trading 1 contract only. In the case of the portfolios, trading 1 contracts mean that it is split according to the weights' distribution (see Table 5.2) among all agents within the portfolio.

By the graphs, we can see that the Pattern Searcher model proposed in this work trained with GA could effectively find patterns that performed well in the given datasets. Except Agent 7, all other agents had a positive slope in the testing set. Also, their testing drawdown looks consistent with the training and validation drawdowns.

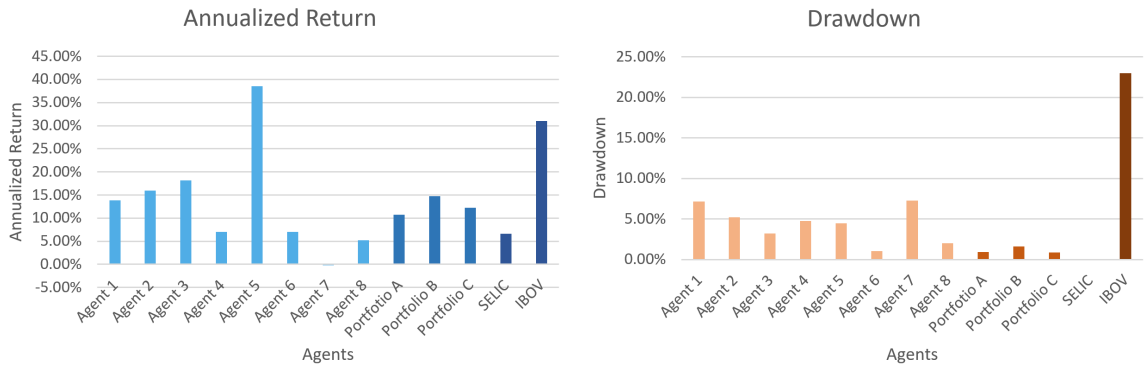
Agents' and portfolios' performance without leverage

The Table 5.16 shows the agents' and portfolios' performance without considering any leverage.

Table 5.16: Agent's and portfolio's results without leverage.

Agent	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Portfolio A	Portfolio B	Portfolio C
Initial Deposit	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00
Contracts	5.00	5.00	5.00	5.00	5.00	2.86	2.86	2.86	3.90	4.50	3.99
Gross Profit	R$ 70,910.01	R$ 49,750.35	R$ 63,374.32	R$ 47,442.65	R$ 111,861.63	R$ 21,730.72	R$ 28,774.65	R$ 27,776.29	R$ 29,998.35	R$ 42,565.98	R$ 32,869.77
Gross Loss	-R$ 50,992.32	-R$ 26,997.03	-R$ 37,683.19	-R$ 37,104.41	-R$ 61,417.03	-R$ 11,386.55	-R$ 28,806.10	-R$ 19,940.64	-R$ 14,303.44	-R$ 21,396.58	-R$ 15,165.00
Total Net Profit	R$ 19,917.68	R$ 22,753.33	R$ 25,691.13	R$ 10,338.24	R$ 50,444.59	R$ 10,344.17	-R$ 31.46	R$ 7,835.66	R$ 15,694.92	R$ 21,169.40	R$ 17,704.77
Total Return	19.91%	22.75%	25.69%	10.34%	50,44%	10.34%	0.00%	7.84%	15.69%	21.16%	17.70%
Drawdown	R$ -7,131.67	R$ -5,184.22	R$ -3,216.35	R$ -4,728.66	R$ -4,451.29	R$ -1,037.95	R$ -7,283.18	R$ -2,011.67	R$ -895.19	R$ -1,583.43	R$ -848.93
Drawdown (%)	-7.13%	-5,18%	-3,21%	-4,72%	-4,45%	-1,03%	-7,28%	-2,01%	-0,90%	-1,58%	-0,85%
Profit per Month	R$ 1,072.77	R$ 1,225.49	R$ 1,383.72	R$ 556.82	R$ 2,716.94	R$ 557.14	-R$ 1.69	R$ 422.03	R$ 845.33	R$ 1,140.18	R$ 953.58
Monthly Return	1.07%	1.23%	1.38%	0.56%	2.72%	0.56%	0.00%	0.42%	0.85%	1.14%	0.95%
Annualized Return	13.86%	15.97%	18.20%	6.99%	38.56%	6.99%	-0.02%	5.26%	10.78%	14.79%	12.24%

The most important metrics from the table are summarized in Figure 5.8.



(a) Annualized return without leverage. **(b)** Drawdown without leverage.

Figure 5.8: Annualized return and drawdown without leverage.

We can see that all agents, except Agent 7, were profitable during the testing and had an average monthly income of around 1%. The best agent was Agent 5, with a monthly income of 2.72% and an annualized return of 38.56%. In this situation where we did not consider any leverage, the portfolios did not give any improvement in terms of monthly return, however, their drawdown was considerably smaller compared to the single agent's, providing lower risks.

6 out of 8 agents, and the 3 portfolios, surpassed the SELIC in terms of annualized return. However, only the Agent 5 overcame the IBOV, with 38.56% against 31% of annualized return. In terms of drawdown, the agent's and portfolio's drawdowns were smaller.

Agents' and portfolios' performance with leverage

When dealing with portfolios, there are usually a trade off between risk and return. Usually, more returns leads to more risks, and vice versa. In this scenario, we can not affirm that one agent is better that another just based on the returns. We would also need to look at the risks. To facilitate the comparison among the agent results, we attempted to normalize the results by equaling the risks of the agents, in a way that all agents have the same drawdown of the training set. We have used the $TradePct = 10\%$, and used the "controlled leverage" approach to calculate the number of contracts. Table 5.17 shows the number of contracts of each agent and each portfolio, and Table 5.18 shows the test results using the distribution of contracts calculated previously.

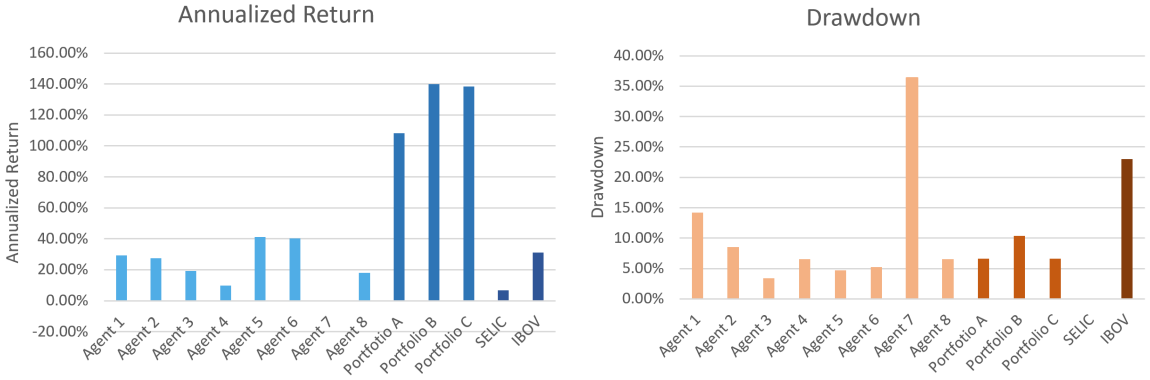
Table 5.17: Estimation of the number of contracts per agent.

Agent	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Portfolio A	Portfolio B	Portfolio C
Balance Drawdown (from training set)	R$ 1,004.32	R$ 1,213.39	R$ 1,901.23	R$ 1,443.08	R$ 1,887.21	R$ 690.52	R$ 700.88	R$ 1,079.81	R$ 348.92	R$ 339.51	R$ 322.62
Balance Needed (from training set)	R$ 10,043.18	R$ 12,133.86	R$ 19,012.30	R$ 14,430.81	R$ 18,872.14	R$ 6,905.23	R$ 7,008.81	R$ 10,798.10	R$ 3,489.24	R$ 3,395.10	R$ 3,226.20
Contracts to make DD = 10%*Deposit	9.96	8.24	5.26	6.93	5.30	14.48	14.27	9.26	28.66	29.45	31.00

Table 5.18: Agents' and portfolios' results with leverage.

Trading Agents	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Portfolio A	Portfolio B	Portfolio C
Initial Deposit	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00	R$ 100,000.00
Contracts per Entry	9.96	8.24	5.26	6.93	5.30	14.48	14.27	9.26	28.66	29.45	31.00
Gross Profit	R$ 141,210.23	R$ 82,002.53	R$ 66,666.65	R$ 65,751.88	R$ 118,546.83	R$ 110,034.78	R$ 143,548.75	R$ 89,941.64	R$ 220,308.34	R$ 278,480.57	R$ 255,216.85
Gross Loss	-R$ 101,546.14	-R$ 44,498.67	-R$ 39,640.85	-R$ 51,423.87	-R$ 65,087.51	-R$ 57,656.46	-R$ 143,705.68	-R$ 64,569.22	-R$ 105,044.65	-R$ 139,983.45	-R$ 117,748.44
Total Net Profit	R$ 39,664.08	R$ 37,503.86	R$ 27,025.80	R$ 14,328.01	R$ 53,459.32	R$ 52,378.32	-R$ 156.93	R$ 25,372.42	R$ 115,263.70	R$ 138,497.12	R$ 137,468.41
Total Return	39.66%	37.50%	27.03%	14.33%	53.46%	52.38%	- 0.16%	25.37%	115.26%	138.50%	137.47%
Profit Factor	1.39	1.84	1.68	1.28	1.82	1.91	1.00	1.39	2.10	1.99	2.17
Drawdown	-R$ 14,206.30	-R$ 8,543.60	-R$ 3,383.60	-R$ 6,553.92	-R$ 4,718.37	-R$ 5,255.08	-R$ 36,339.52	-R$ 6,513.31	-R$ 6,578.52	-R$ 10,362.66	-R$ 6,595.70
Drawdown (%)	- 14,21%	- 8,54%	- 3,38%	- 6,55%	- 4,72%	- 5,26%	- 36,34%	- 6,51%	- 6,58%	- 10,36%	- 6,60%
Entries	314	163	272	252	531	170	144	130	1976	1976	1976
Entry Costs	R$ 1,500.72	R$ 644.81	R$ 686.71	R$ 838.21	R$ 1,350.56	R$ 1,181.71	R$ 986.19	R$ 577.88	R$ 3,397.88	R$ 4,194.16	R$ 4,005.61
Leverage	1.99	1.65	1.05	1.39	1.06	5.06	4.99	3.24	7.34	6.54	7.76
Days with Entry	215	120	188	176	282	149	139	124	368	368	368
Profit per Month	R$ 2,136.31	R$ 2,019.96	R$ 1,455.61	R$ 771.71	R$ 2,879.32	R$ 2,821.09	-R$ 8.45	R$ 1,366.56	R$ 6,208.10	R$ 7,459.45	R$ 7,404.04
Recovery Time (days)	199.50	126.89	69.74	254.78	49.16	55.88	-128982.98	142.99	31.79	41.68	26.72
Largest Profit Day	R$ 4,619.59	R$ 4,044.43	R$ 3,972.80	R$ 2,881.82	R$ 3,465.22	R$ 3,790.56	R$ 5,961.68	R$ 3,348.19	R$ 6,515.48	R$ 5,277.23	R$ 7,061.76
Largest Loss Day	-R$ 7,037.89	-R$ 5,331.70	-R$ 1,845.37	-R$ 2,942.32	-R$ 2,115.25	-R$ 2,626.30	-R$ 4,124.68	-R$ 3,335.11	-R$ 3,563.10	-R$ 6,471.72	-R$ 3,676.04
Success Rate	54.88%	60.00%	57.45%	54.55%	53.90%	69.13%	51.80%	53.23%	57.61%	56.79%	58.97%
Std. Dev. of the returns	1100.14	759.29	522.44	566.01	708.49	755.42	1374.65	805.30	1098.57	1462.34	1256.46
VaR(95%)	-R$ 1,648.65	-R$ 798.30	-R$ 877.96	-R$ 1,094.27	-R$ 837.59	-R$ 1,366.14	-R$ 2,574.00	-R$ 1,648.83	-R$ 1,288.71	-R$ 1,702.48	-R$ 1,470.83
Monthly Return (%)	2.14%	2.02%	1.46%	0.77%	2.88%	2.82%	-0.01%	1.37%	6.21%	7.46%	7.40%
Annualized Return	29.33%	27.55%	19.22%	9.80%	41.25%	40.28%	-0.10%	17.96%	108.09%	139.96%	138.46%
Discounting Income Tax of 20%											
Total Net Profit (-IT)	R$ 31,731.27	R$ 30,003.09	R$ 21,620.64	R$ 11,462.41	R$ 42,767.46	R$ 41,902.66	-R$ 125.54	R$ 20,297.94	R$ 92,210.96	R$ 110,797.69	R$ 109,974.73
Profit per Month (-IT)	R$ 1,709.04	R$ 1,615.97	R$ 1,164.49	R$ 617.37	R$ 2,303.45	R$ 2,256.88	-R$ 6.76	R$ 1,093.25	R$ 4,966.48	R$ 5,967.56	R$ 5,923.23
Monthly Return (-IT)	1.71%	1.62%	1.16%	0.62%	2.30%	2.26%	-0.01%	1.09%	4.97%	5.97%	5.92%
Annualized Return (-IT)	22.90%	21.54%	15.13%	7.78%	31.93%	31.20%	-0.08%	14.14%	80.35%	102.43%	101.40%

The most important metrics from Table 5.18 are summarized in Figure 5.9.



(a) Annualized return with leverage.

(b) Drawdown with leverage.

Figure 5.9: Annualized return and drawdown with leverage.

Agents' performance

A total of 7 out of 8 agents had positive total net profit during the test period. The total net profit varied from R$ 14,328.01 to R$ 53,459.32, corresponding to a growth of 14.33% to 53.46%. The drawdown of the test stayed between 3,38% and 14.21%, deviating up to 40% from the training drawdown.

The Agent 7 was the only one that presented poor results. It's possible to see strategies that work well for a long time and suddenly stop to work. We could address the reason for it to the change of market behavior over time, possibly for external factors, such as economy, interest rate, and inflation. All the other agents are likely to have loss periods like Agent 7 at some moment. This reinforces the benefits of using a strategy portfolio.

Portfolios' performance

The portfolios A, B, and C presented a total net profit of R$ 115,044.55, R$ 138,983.47 and R$ 137,468.44, and an annualized return of 108.09% 139.96% and 138.46%, respectively. All portfolios managed to more than double the initial deposit in a period of 1 year. The drawdown went from 6.58% to 10.36%, which is coherent with the 10% drawdown of the training set.

The way the portfolio's weights were determined significantly implicated in the results. The Portfolio A had great profitably even with no training. The training using GA improved further the returns of the Portfolios B and C. The Portfolio B has a slightly higher monthly income than the Portfolio C, 7.46% against 7.40%, however, the Portfolio C is better since its drawdown was only 6.60%, 36% smaller than the Portfolio B drawdown. The reason for that is the possibility of having occurred some overfitting of the agent B. Its weights assumed an unbalanced distribution, with a very high volume of contracts in the Agent 1 and only a very few in the Agents 6 and 7 (see Table 5.2). The effect of this is similar to removing agents from the portfolio and relying only on a few.

On the other hand, restricting the standard deviation of the weights of the Portfolio C resulted in a soft optimization with weights ending up having close values from each other. This soft optimization was enough to increase the performance of the portfolio while maintaining its robustness on unseen data.

Agents' performance vs Portfolios' performance

The portfolios had a considerably higher performance compared to the single agents' performance. The portfolio's highest annualized return was 3 times greater than the highest single agent's annualized return, 139.96% against 42.25%. This wide difference is mostly due to the drastic risk decrease that a portfolio can yield. The decrease of the drawdown opens room for increasing the leverage, and consequently, achieving higher profitability for the same amount of risk.

Comparison with Benchmarks

6 out of 8 agents, and the 3 portfolios, overcame the SELIC rate in terms of annualized return. Only 2 agents (Agent 5 and 6) were able to beat the IBOV, with 41.25% and 40.28%, respectively, against 31% of annualized return. On the other hand, all of the 3 portfolios surpassed the IBOV with considerable advantage. The Portfolio C, which we have considered as being the best among all portfolios, had an annualized return of 138.46%, which is over

4 times greater than the IBOV's. Also, the Portfolio C's drawdown was 6.60%, which is around 3.5 times smaller than the IBOV's cumulative fall of 23%.

Further details of Portfolio C

The Portfolio C presented the best results among all agents, portfolios and benchmarks. Thus we will present further details regarding to its cumulative profit (Figure 5.10), return distribution (Figure 5.11) and cumulative return by month, weekday (Figure 5.12) and day (Figure 5.13).

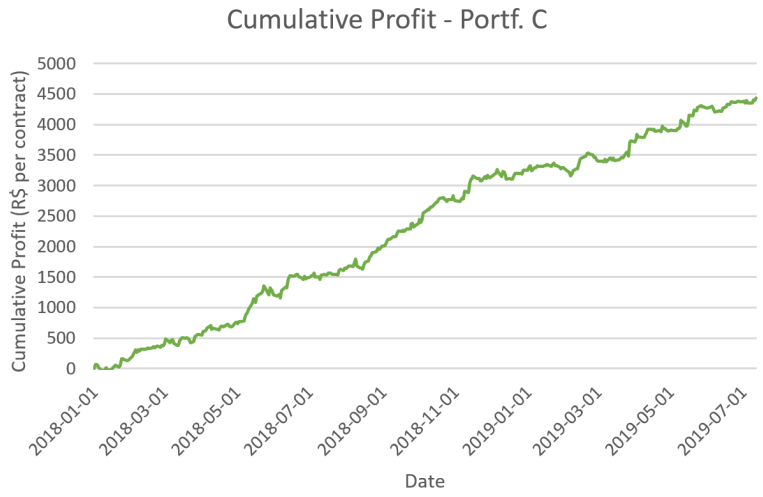


Figure 5.10: Cumulative profit of Portfolio C in the testing set.

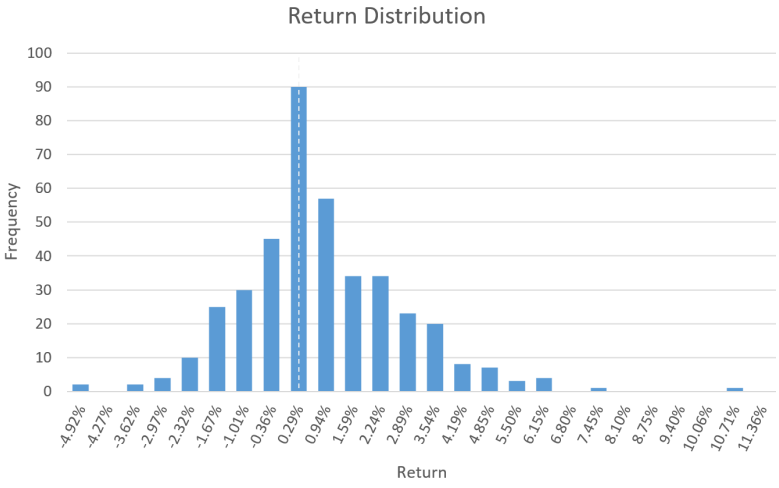
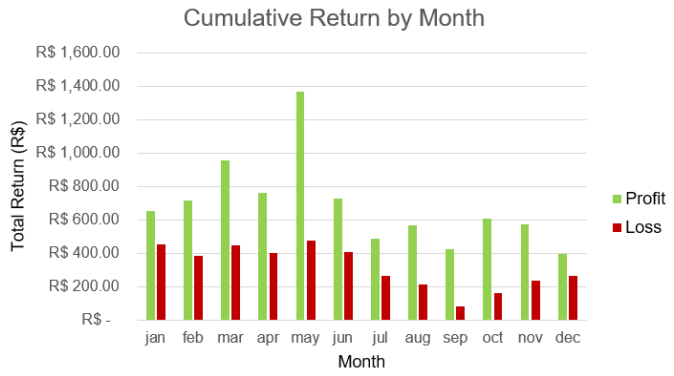
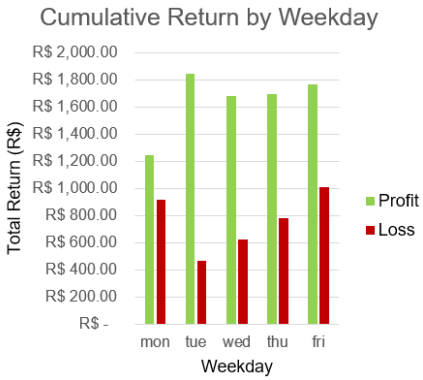


Figure 5.11: Return distribution of Portfolio C in the testing set.



(a) Cumulative return of Portfolio C filtered by month.



(b) Cumulative return of Portfolio C filtered by weekday.

Figure 5.12: Cumulative return of Portfolio C in the testing set filtered by month and weekday.

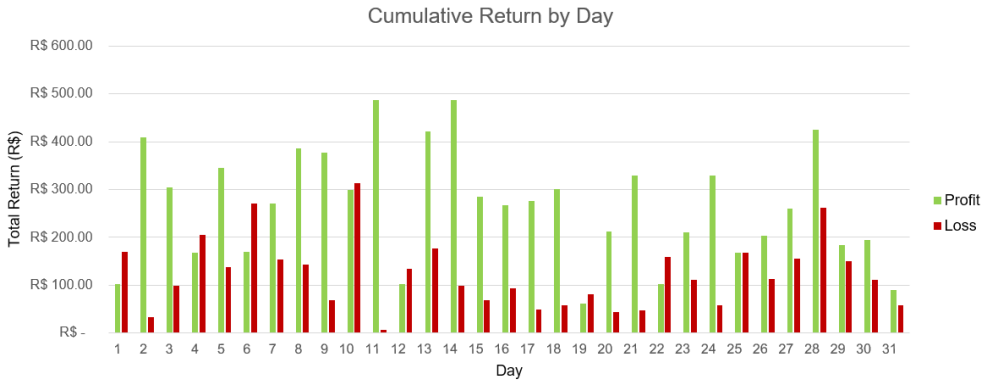


Figure 5.13: Cumulative return of Portfolio C in the testing set filtered by day.

From the figures, we can see that the cumulative profits are higher than the losses for all months, weekdays, and 23 out of 31 days of the month. Also, the return distribution is asymmetric, with more positive than negative returns.

Chapter 6

Conclusion

This work proposed a model that automatically searches for profitable trading patterns, named Pattern Searcher, and generated strategy portfolios, both trained via GA. As a result, 7 out of 8 agents generated through the Pattern Searcher model were very profitable.

The creation of portfolios considerably improved the profitability compared to single agents', outperforming the Brazilian benchmarks SELIC and IBOV. The best portfolio achieved an annualized return of 138.46% and a drawdown of 6,56%, With this return, it would more than double the initial deposit in one year.

The system showed some desirable properties. One of them is the generation of rules than can be easily interpreted by humans, allowing the investor to accept or reject the strategy according to his knowledge. Another desirable characteristic is the simplification of the system flow compared to some Machine Learning methods. The model and the strategy are interconnected, in a way that the training directly optimizes the agent performance in terms of profitability.

Although the results seem promising, it is important to have in mind that past results are not a guarantee of future results. The stock market may change its behavior over time, and the agent's strategy might lose effectiveness.

Future work would consist of evaluating the robustness and consistency of the GA agents, and applying the method in other datasets, including international financial markets (e.g., NYSE, NASDAQ, Tokyo), cryptocurrencies and forex. Besides, we would extend the approach to agents that trade multiple assets simultaneously, rather than specific assets. According to [11], this can help to prevent overfitting, resulting in more robust systems. In addition, Machine Learning techniques can be incorporated to generate new indicators, based on the book orders, for example, that may improve the statistical advantage of the agent's trade signals.

Bibliography

[1] Bailey, D., Borwein, J. J., Lopez de Prado, M., and Zhu, Q. (2014). Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance. *Notices of the American Mathematical Society*, 61:458.

[2] BancoCentral (2020). Taxas de juros básicas. Available at `https://www.metatrader5.com/en/automated-trading/strategy-tester`. [Online; accessed 13-January-2020].

[3] Chen, J. (2019). Stock market. Available at `https://www.investopedia.com/terms/s/stockmarket.asp`. [Online; accessed 07-January-2020].

[4] CHEN, J. (2019a). Technical indicator. Available at `https://www.investopedia.com/terms/t/technicalindicator.asp`. [Online; accessed 21-January-2020].

[5] CHEN, J. (2019b). What is a portfolio? Available at `https://www.investopedia.com/terms/p/portfolio.asp`. [Online; accessed 06-January-2020].

[Corp] Corp, M. S. Optimization types. `https://www.metatrader5.com/en/terminal/help/algotrading/optimization_types`.

[7] Corp, M. S. (2020a). Documentation. Available at `https://www.mql5.com/en/docs`. [Online; accessed 05-January-2020].

[8] Corp, M. S. (2020b). Optimization types. Available at `https://www.metatrader5.com/en/terminal/help/algotrading/optimization_types`. [Online; accessed 05-January-2020].

[9] Corp, M. S. (2020c). A powerful platform for forex and exchange markets. Available at `https://www.metatrader5.com/en/`. [Online; accessed 05-January-2020].

[10] Corp, M. S. (2020d). Trading strategy tester: Test and optimize your trading robot before you use it for real trading. Available at https://www.metatrader5.com/en/automated-trading/strategy-tester. [Online; accessed 05-January-2020].

[11] de Prado, M. L. (2018). *Advances in Financial Machine Learning*. Wiley Publishing, 1st edition. ISBN 1119482089.

[12] Dempster, M. A., Payne, T. W., Romahi, Y., and Thompson, G. W. (2001). Computational learning techniques for intraday fx trading using popular technical indicators. *Trans. Neur. Netw.*, 12(4):744--754. ISSN 1045-9227.

[13] Faustryjak, D., Jackowska-Strumillo, L., and Majchrowicz, M. (2018). Forward forecast of stock prices using lstm neural networks with statistical analysis of published messages. pages 288--292.

[14] Folger, J. (2019). Automated trading systems: The pros and cons. Available at https://www.investopedia.com/articles/trading/11/automated-trading-systems.asp. [Online; accessed 07-January-2020].

[15] Hayes, A. (2019). Leverage. Available at https://www.investopedia.com/terms/l/leverage.asp. [Online; accessed 07-January-2020].

[16] Hu, Y., Liu, K., Zhang, X., Su, L., Ngai, E., and Liu, M. (2015). Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing*, 36.

[17] Investopedia (2018a). Technical analysis. Available at https://www.investopedia.com/terms/t/technicalanalysis.asp. [Online; accessed 25-November-2018].

[18] Investopedia, C. M. (2018b). Moving averages: Strategies. Available at https://www.investopedia.com/university/movingaverage/movingaverages4.asp. [Online; accessed 25-November-2018].

[19] Iskrich, D. and Grigoriev, D. (2017). Generating long-term trading system rules using a genetic algorithm based on analyzing historical data. In *2017 20th Conference of Open Innovations Association (FRUCT)*, pages 91–97. ISSN 2305-7254.

[20] Kampouridis, M. and Tsang, E. (2010). Eddie for investment opportunities forecasting: Extending the search space of the gp. In *IEEE Congress on Evolutionary Computation*, pages 1–8. ISSN 1089-778X.

[21] Kampouridis, M. and Tsang, E. (2011). Using hyperheuristics under a gp framework for financial forecasting. In Coello, C. A. C., editor, *Learning and Intelligent Optimization*, pages 16--30, Berlin, Heidelberg. Springer Berlin Heidelberg.

[22] Kolakowski, M. (2019). How robots rule the stock market (spx, djia). Available at `https://www.investopedia.com/news/how-robots-rule-stock-market-spx-djia/`. [Online; accessed 07-January-2020].

[23] Kroha, P. and Friedrich, M. (2014). Comparison of genetic algorithms for trading strategies.

[24] Liu, S., Liao, G., and Ding, Y. (2018). Stock transaction prediction modeling and analysis based on lstm. pages 2787--2790. ISSN 2158-2297.

[25] Maciej J. Capiński, E. K. (2014). *Portfolio Theory and Risk Management*. Mastering Mathematical Finance. Cambridge University Press, 1 edition. ISBN 1107003679,9781107003675.

[26] MallawaarachchI, V. (2017). Introduction to genetic algorithms — including example code. Available at `https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3`. [Online; accessed 03-January-2020].

[27] Martinez, L., Hora, D., Palotti, J., Meira Jr, W., and Pappa, G. (2009). From an artificial neural network to a stock market day-trading system: A case study on the bmf bovespa. pages 2006–2013.

[28] MCCLURE, B. (2019). Modern portfolio theory: Why it's still hip. Available at `https://www.investopedia.com/managing-wealth/modern-portfolio-theory-why-its-still-hip/`. [Online; accessed 06-January-2020].

[29] Michael P. Wellman, Amy Greenwald, P. S. (2007). *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition*. Intelligent Robotics and Autonomous Agents series. The MIT Press. ISBN 026223260X,9780262232609.

[30] Moody, J. and Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 12(4):875--889.

[31] Olowoyo, S. (2010). The use of the mql5 standard trade class libraries in writing an expert advisor. Available at `https://www.mql5.com/en/articles/138`. [Online; accessed 22-January-2020].

[32] Padua Braga, A. (2007). *Redes neurais artificiais: teoria e aplicaes*. LTC Editora. ISBN 9788521615644.

[33] Pappa, G. L. (2018). Algoritmos genéticos. Unpublished Presentation.

[34] Patel, J., Shah, S., Thakkar, P., and Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259 – 268. ISSN 0957-4174.

[35] Perfeito, P. M. (2011). Análise do comportamento do mercado de acções utilizando técnicas de data mining. *Mestrado em sistemas integrados de apoio à decisão*.

[36] Pimenta, A., Guimarães, F. G., Carrano, E. G., Nametala, C. A. L., and Takahashi, R. H. C. (2014). Goldminer: A genetic programming based algorithm applied to brazilian stock market. In *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 397–402. ISSN .

[37] R. Koza, J. (1990). Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems.

[38] Russell, S. J. and Norvig, P. (2009). *Artificial Intelligence: a modern approach*. Pearson, 3 edition.

[39] S. Malagrino, L., Roman, N., and Monteiro, A. (2018). Forecasting stock market index daily direction: a bayesian network approach. *Expert Systems with Applications*, 105.

[40] Sutton, R. S. and Barto, A. G. (2018). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 2 nd edition. ISBN 0262193981.

[41] Tsang, E., Yung, P., and Li, J. (2004). Eddie-automation, a decision support tool for financial forecasting. *Decis. Support Syst.*, 37(4):559--565. ISSN 0167-9236.