TÉCNICO LISBOA

Analysis of the viability of a Dynamic GA coupled with Fuzzy Membership functions in the Forex Market

Paulo Jorge Laranjeira Bernardo

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. Rui Fuentecilla Maia Ferreira Neves

Examination Committee

Chairperson: Prof.Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. Rui Fuentecilla Maia Ferreira Neves
Member of the Committee: Prof. Alexandre Paulo Lourenço Francisco

November 2021

DECLARATION

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Abstract

Trading services have never been in higher demand than in the recent years, with many investors turning their eyes to the Foreign Exchange Market due to its high volume and low volatility. This thesis explores the background of this market, as well as the background of some machine learning techniques applied to it throughout the last years. This work also proposes and presents an implementation of a Dynamic GA based in the technical analysis of the FOREX market. An hybrid approach using this Dynamic architecture and a Fuzzy Logic component is also proposed, with the objective of analysing its viability to forecast a financial market such as the FOREX. By testing both frameworks in three different testing periods of six months each, the Dynamic implementation presents steady and profitable results in all three periods. On the other hand, the hybrid approach presented itself as a non viable option in its current version to real time trading due to its unstable results in the the same testing periods. However these results, present some promising metric values that point to the possibility of creating a profitable and steady implementation.

Keywords: FOREX, Fuzzy Logic, Genetic Algorithms, Technical Indicators

Resumo

A procura de serviços de *trading* nunca foi tão elevada como nos últimos anos, tendo sido notado um aumento de investidores no mercado FOREX devido ao elevado volume e à sua reduzida volatilidade. Esta tese procura explorar o funcionamento deste mercado e de técnicas de *machine learning* aplicadas ao mesmo durante os últimos anos. Este trabalho também propõe a implementação de um Algoritmo Genético Dinâmico, que se baseia na análise técnica do mercado FOREX. Também é proposta uma abordagem híbrida entre esta implementação e uma arquitectura *fuzzy*, com o objectivo de analisar a sua viabilidade na previsão de um mercado financeiro como este. Depois de testar a implementação Dinâmica em três períodos diferentes de seis meses cada, apresentaram-se resultados estáveis e lucrativos em todos os três períodos. Por outro lado a abordagem híbrida provou ser pouco viável na sua versão corrente para *trading* em tempo real. Apesar de tudo, esta versão apresenta resultados promissores que fazem com que exista a possibilidade de ser criada uma arquitectura otimizada que obtenha resultados lucrativos e estáveis.

Palavras-Chave: FOREX, Lógica *Fuzzy*, Algoritmos Genéticos, Indicatores Técnicos

Contents

List of Tables VIII

List of Figures XI

Acronyms XII

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Objective . . . 2
- 1.3 Main Contribuitions . . . 3
- 1.4 Document's Organization . . . 3

2 Background 4
- 2.1 Forex Market Concepts . . . 4
  - 2.1.1 Basics of Forex . . . 4
  - 2.1.2 Leverage . . . 4
- 2.2 Forex market analysis . . . 5
  - 2.2.1 Fundamental Analysis . . . 5
  - 2.2.2 Technical Analysis . . . 5
- 2.3 Fuzzy Sets . . . 10
- 2.4 Genetic Algorithm . . . 11
  - 2.4.1 Genetic Algorithm: Structure . . . 11
  - 2.4.2 Genetic Algorithm: Diversity . . . 13
  - 2.4.3 Genetic Algorithm: Adaptive Approach . . . 13
- 2.5 State of the art . . . 14
  - 2.5.1 Use of Technical Indicators . . . 14
  - 2.5.2 Fuzzy Approaches . . . 14
  - 2.5.3 Genetic Algorithm Approaches . . . 15
  - 2.5.4 Chapter Conclusions . . . 17

3 Model Architecture 19
- 3.1 Overview . . . 19
- 3.2 Inputs . . . 20
  - 3.2.1 Historical Data Import . . . 20
- 3.3 Genetic Algorithm . . . 21
  - 3.3.1 Population Creation . . . 21
  - 3.3.2 Population Training (Wide Search) . . . 31
  - 3.3.3 Population Testing . . . 35

3.3.4 Overall Algorithm Parameters . . . 37
3.4 Fuzzy Implementation . . . 37
3.4.1 Membership Functions . . . 38
3.5 Chapter Conclusions . . . 41

4 System Validation **42**
4.1 Evaluation Metrics . . . 42
4.1.1 Return on Investment . . . 42
4.1.2 Accuracy . . . 43
4.1.3 Drawdown . . . 43
4.2 Case Studies . . . 43
4.2.1 Methodologies . . . 43
4.2.2 Case Study A . . . 45
4.2.3 Case Study B . . . 50
4.2.4 Case Study C . . . 53
4.2.5 Case Study D . . . 56
4.2.6 Dynamic vs Static . . . 59
4.3 Chapter Conclusions . . . 64

5 Conclusions **65**
5.1 Conclusions . . . 65
5.2 Future Work . . . 66

Bibliography

A Case Study A: Periodic Results

B Case Study B: Periodic Results

C Case Study C: Periodic Results

D Case Study D: Periodic Results

List of Tables

1 Most common leverage ratios and margin requirements 5
2 Review of the state of the art. 18

3 Review of all the implemented trading rules. 25
4 Ranges of the variables of each trading rule. 27
5 Summary of the algorithm parameters. 37
6 Behaviour of each trading rule to the membership function 2. 39

7 Two examples of the Return on Investment. 42
8 Example of the metric position accuracy. 43
9 Training and testing data time periods. 44
10 Configuration of the parameters of the Wide Search (GA training). 45
11 Default simulation parameters. 45
12 Parameters of the baseline configuration. 46
13 Variations of the baseline configuration tested in the Narrow Search method. 46
14 Trading rules and ranges used in case study A. 46
15 Overall results of all configurations tested in Case Study A. 49
16 Parameters of the baseline configuration. 50
17 Sets of trading rules used during Case Study B. 50
18 Overall results of all the sets tested in Case Study B. 53
19 Configuration 1 for the simulation parameters. 54
20 Configurations 2 and 3 for the simulation parameters. 54
21 Overall results of all the sets tested in Case Study C. 55
22 Sets of trading rules used during Case Study D. 56
23 Overall results of all the sets tested in Case Study D. 58
24 Results of the baseline approaches compared with the benchmark strategies 63

List of Figures

1 Comparison between fuzzy and crisp approaches. 11
2 Example of a binary chromosome. 12
3 Structure of a genetic algorithm. 12
4 Two examples of crossover methods. 13

5 Simplified overview of the implemented model. 19
6 Overall architecture. 20
7 Slice of the data used during the project. 21
8 Process of creating the population. 22
9 Representation of the general structure of this architecture's chromosome. 26
10 Example of a chromosome in a model with RSI, EMA, Bollinger and Stochastic trading rules. 27
11 Representation of the Population Training. 32
12 Representation of the selection process. 33
13 Representation of the one-cut crossover, in the context of the implementation. 33
14 Representation of the two-cut crossover, in the context of the implementation. 34
15 Simple representation between Simple Test (right) and Narrow Search (left). 36
16 Representation of the Narrow Search method. 36
17 Fuzzy implementation of the process of creating a population. 38
18 Example of a membership function of type 1. 39
19 Example of a membership function of type 2. 40
20 Example of a membership function of type 3. 40

21 Example of the drawdown of standard ROI evolution graph. 44
22 Histogram of the configurations in which the number of samples is altered. 47
23 Histogram of the configurations in which the diversity parameters are altered. 48
24 Histogram of the ROIs obtained by each trading rule set, using the Dynamic implementation of the GA. 52
25 Histogram of the ROIs obtained by each trading rule set, using the Dynamic implementation of the GA. 55
26 Histogram of the ROIs obtained by each trading rule set, using the Dynamic implementation of the GA. 57
27 Evolution of ROI obtained by the crisp and fuzzy approaches throughout the totality of the testing period. 59
28 Evolution of ROI obtained by the relevant configurations of case study A throughout the totality of the testing period. 61
29 Evolution of ROI obtained by the relevant rule sets of case study B throughout the totality of the testing period. 62

30 Evolution of ROI obtained by baseline implementation, the static implementation and the benchmarks throughout the totality of the testing period. 63

Acronyms

ADX Average Directional Movement Index.

AUD Australian Dollar.

B&H Buy and Hold.

CAD Canadian Dollar.

CHF Swiss Franc.

EMA Exponentially Moving Average.

EUR Euro (European Monetary Unit).

EWMA Exponentially Weighted Moving Average.

FOREX Foreign Exchange.

GA Genetic Algorithm.

GBP British Pound.

IIR Infinite-Impulse Response.

JPY Japanese Yen.

MA Moving Average.

MACD Moving Average Convergence/Divergence.

MFI Money Flow Index.

NZD New Zeland Dollar.

ROC Rate of Change.

ROI Return on Investment.

RSI Relative Strength Index.

S&H Sell and Hold.

SMA Simple Moving Average.

TI Technical Indicators.

USD United States Dollar.

WMA Weighted Moving Average.

Chapter 1

Introduction

In the last two decades, it has been noted an increasing trend of implementing machine learning models in order to take advantage of the existing financial markets. Several approaches throughout the years presented profitable and accurate results, making it feasible to implement an architecture that invests automatically in real time. Among the machine learning approaches that have been applied successfully to these types of markets one can account for the Genetic Algorithms, Neural Network implementations, among others. This thesis proposes itself to implement a Dynamic GA and test the viability of an approach that couples this GA with a fuzzy approach, in a stable financial setting.

The concept of Genetic Algorithms is fairly old, with the first mentions of an algorithm based on the Darwin's evolutionary principles being made by Alan Turing in 1948 [1]. The Evolutionary Computation concepts, which are the basis for Evolutionary algorithms such as the GA, are based in the natural selection and evolutionary processes present in Darwin's "On the Origin of Species", which are translated to the GA's mechanisms such as fitness based selection, mutation, and even crossover breeding. Genetic Algorithm have been employed in the last three decades a little bit throughout all science fields, namely in IIR adaptive filtering, time delay estimation, active noise control, and speech processing ([2]), and in image enhancement and segmentation [3]. In recent years, several relevant implementations of these principles have been applied to the financial markets with positive results, in works by Gorgulho et al. [4], by Hirabayashi et al. [5] and by Almeida et al.[6].

The idea of Fuzzy Logic is more recent, having first been mentioned in a proposal by the Azerbaijani scientist Lotfi Zadeh [7] in 1965. The general concept associated with fuzzy logic is to discard binary/ternary signals and instead associate values that represent partial truth, as opposed to a Boolean truth. The application of this concept to the analysis of financial markets is rather new, with few literature in this field. Some relevant works that employ the concept of fuzzy logic to financial markets are the works published by Juszczuk et al. [8] and Naranjo et al. [9].

The environment selected to test the performance of the implemented architecture was the Foreign Exchange Market (FOREX). This is the most traded financial market in the world [10], even larger than the stock market, with a daily volume of around $6.6 trillion, according to the 2019 Triennial Central Bank Survey [11]. In this market the financial instruments sold and bought are the countries' currencies. Several currencies compose the Foreign Exchange Market, but 95% of all daily transactions involve only eight currencies, the U.S Dollar (USD), the Euro (EUR), the British Pound (GBP), the Japanese Yen (JPY), the Swiss Franc (CHF), the New Zealand Dollar (NZD), the Australian Dollar (AUD) and the Canadian Dollar (CAD) [12].

This market's variety and high volume, as well as the fact that transactions are available 24h a day, every day from Monday to Friday, make the FOREX one of the most appealing markets for traders and researchers alike, making it a promising environment to apply machine learning methods and algorithms

with the objective of forecasting its behaviour.

To forecast this financial market, while using a machine learning architecture, two types of market analysis were considered. The Fundamental analysis and the Technical Analysis. In the context of this market, the first relates to the macro economic factors of the currency market, such as geopolitical factors as well as inflation rates. The second analysis of the bases itself solely on the values of the market to generate technical indicators that forecast the market's behaviour, making it perfect type of analysis to use in a machine learning framework such as the one that this work proposes.

To this effect several methods have already been proved to achieve profitable results while forecasting the FOREX market with the use of technical analysis of the market allied to Neural Networks, or coupled with an Evolutionary Algorithm. Within the same scope of these works, Fuzzy logic approaches have also proven to present more consistent results than some benchmarks such as the Buy & Hold (B&H) strategy.

1.1 Motivation

The review of the state of the art done by Pradeepkumar et al., "Soft Computing Hybrids for FOREX Rate Prediction: A Comprehensive Review" [13], highlights several of the hybrids that still need further research with one of them being hybrids involving Fuzzy Logic and Evolutionary Computation [13].

With that in mind, the motif of the work at hand is to study the possible application of an Evolutionary Algorithms coupled with a Fuzzy Logic approach to the FOREX market, with the objective of covering this specific gap in the current state of the art. An approach that delivers profitable results could also increase the interest of applying a Fuzzy Logic-Genetic Algorithm based approach to other financial markets or even to other applicable research fields.

The possibility of applying the approach to real data of such a stable market as the FOREX is also a very enticing motive for the work at hand, with several recent trading services implementing features that recommend currencies and stocks to invest.

1.2 Objective

With the main goal of implementing a model that results of coupling a Fuzzy logic approach with a Dynamic Genetic Algorithm and testing its performance in a market such as the FOREX, several objectives are expected to be accomplished during this work, namely:

- Using Technical Indicators (data for the model) to study future prices of currency pairs of the Forex market;
- Produce a Dynamic Genetic Algorithm that introduces an adaptable approach;
- Study the performance of the Dynamic Genetic Algorithm using dynamic criteria;
- Study and find applicable membership functions to apply a Fuzzy Logic;
- Validate the viability of a Fuzzy approach to the detriment of a Crisp approach.
- Implement a simulator that computes the profit using the real world constraints.

1.3 Main Contribuitions

The main contributions of this dissertation are:

- Produce a Dynamic Genetic Algorithm based in the Technical analysis of the FOREX market, that adapts itself to this market's behaviour, through the use of a Wide Search and consequently a Narrow Search.
- Create an implementation that allows the study of the viability of employing fuzzy logic in the context of financial markets.

1.4 Document's Organization

The document will start with the Background chapter 2, in which the background of the financial markets, and the machine learning tools that were utilized during the implementation, is explored. An overview of the state of the art and analysis of the related works are also provided in this chapter.

In chapter 3 - Proposed Architecture, a detailed description of the architecture of the implementation is elaborated and explored. The Validation chapter (chapter 4) analyses the performance of the the proposed architecture through a series of Case Studies. Finally, a Conclusions chapter (5) concludes this thesis, presenting a summary of the work done, a series of conclusions about it and propositions of improvements to be done in future works.

Chapter 2

Background

2.1 Forex Market Concepts

This section has the purpose of clarifying how the FOREX Market works and how to interact with it, since its operation is outside of the scope of the Master Degree in Electrical and Computer Engineering. In the current section several mechanisms, that will be taken into account during the implementation of the architecture, will be explored.

2.1.1 Basics of Forex

As previously stated, the various currencies traded in FOREX are exchanged in pairs (e.g., EUR/USD, USD/JPY). In a giver pair, the first currency is the base currency, which shows how much the base currency is worth, when measured against the second currency (quote currency) [14]. For example, if the USD/CHF rate equals 1.6215, then one USD is worth CHF 1.6215. It is also noteworthy that the smallest unit of price in any currency pair is called a pip (e.g., in EUR/USD, the value of 1 pip is $0.0001).

Let us now suppose that a trader buys currency at a certain price, with the objective of selling it at a later date for a higher price. This is called a long position. When the trader decides to sell the currency previously bought, this position is closed.

This trader can, however, predict that same currency is going to drop in value face to another one. The correct position to make is to short this currency, or in other words, to sell the currency, wait for it to lower in price and to buy it back, closing the short position.

2.1.2 Leverage

A concept widely used in financial markets, such as Forex, is leverage, which is the ratio between the amount of currency used in a transaction to the required security deposit (required margin) [14]. This mechanism is used generally to increase the profit made from fluctuations in exchange rates [6]. Examples of common leverage ratios and corresponding required margins [15] are displayed in Table 1.

If the leverage applied to an investment of 1000 euros (required margin) is, for example, 100:1, then the total investment value becomes 100 000 euros (loan provided by broker/investor). Suppose now that an investor goes long 100 000 in EUR/USD with a buying price of 1.2853 and closes his position with a selling price of 1.2873, with a difference of 20 pips. The profit can be computed as: 100 000 € × 0.0020 = $ 200 [16] [10].

Table 1: Most common leverage ratios and margin requirements

Margin Requirements	Leverage Ratio
2%	50:1
1%	100:1
0.5%	200:1

2.2 Forex market analysis

Now that the intricate mechanisms of the FOREX market are exposed, the necessity of analysing its state at a given time arises. To this effect, two schools of thought emerge, the Fundamental Analysis and the Technical Analysis.

2.2.1 Fundamental Analysis

Fundamental analysis studies the relationship between the evolution of exchange rates and economic indicators [14], of the currencies' country. This analysis focus heavily in the assessment of the economic, social and political forces surrounding the intervening economies.

Most fundamental studies rely heavily on macro-economic indicators, such as economic growth rates, interest rates, inflation, and unemployment. Other criteria taken into account, while forecasting the FOREX market, are the geopolitical factors, such as, monetary and economic policy changes, and changes in governments or international relationships.

The importance of the aforementioned indicators is evidenced by the fluctuations that are felt in the market during important economic meetings. One example of such fluctuations are the ones that exist in the evolution of the EUR/USD, during the meetings of the European Central Bank or the Federal Open Market Committee.

2.2.2 Technical Analysis

While fundamental analysis is used to forecast the long term evolution of the currency rates, based on the macro-economical indicators, technical analysis focus solely on the past events of the currencies' price rates, to forecast the future rate evolution and capitalize from it. Due to its focus on the past rates, the technical analysis has become the primary tool to successfully analyze and trade shorter-term price movements [14].

This type of study is based on three premises, enunciated by Ozturk et al. [16]:

- Market action discounts everything: any factor that can affect the prices is already reflected in the price.
- Prices move in trends: the purpose of the technical analysis is to detect a price trend in the early phases of development.
- History repeats itself: technical analysis uses patterns which have shown success in the past and assumes they will work in the future.

This type of analysis consists primarily on the study of technical indicators, some of which can be interpreted to predict market direction or to generate buy and sell signals, as well as to set profit targets and stop-loss safeguards, due to its ability to generate price-specific information and forecasts [14].

As Almeida et al. [6] states, exist two main types of technical indicators:

- Trend following - Used to understand trends, that is, to identify if a trend has begun or ended. These indicators are used usually to identify entry and exit points.
- Momentum oscillators - Predict sudden changes on the asset's behaviour, such as, the speed of the price movement variation, which is of major importance when considering the amount of leverage to be used.

2.2.2.1 Forex - Relevant Indicators

Most Forex prediction techniques that are based in indicators, rely heavily in Technical Indicators (TI), some of which will be the focus of the current section.

a) Relative Strength Index

One of the most typical indicators is the Relative Strength Index (RSI). As enunciated in [5], this indicator tries to capitalize from the market, aiming to buy, when the currency is sold too much (the price is low, normally below 30 %) and to sell when it is bought too much (the price is high, normally above 70 %). Default values for N are between 9 and 14.

$$RSI_N(\%) = \frac{|U|}{|U| + |D|} \times 100 \tag{1}$$

$|U|$ - Sum of the absolute value of rising width in the past N periods
$|D|$ - Sum of the absolute value of falling width in the past N periods

b) Rate of Change

The Rate of Change (ROC) is the ratio between the current price of a currency and the price of the same currency N periods in the past, as represented in the following expression:

$$ROC_N(\%) = \frac{A_t - A_{t-N}}{A_{t-N}} \times 100. \tag{2}$$

$|A_t|$ - Currency price at period t.

This ratio measures how rapidly the price of a given asset is changing [4]. If the price is rising or falling too quickly it will probably indicate, respectively, overbought or oversold conditions.

c) Moving Averages

Another widely used indicator is the Moving Average (MA), which is a technique for smoothing the short-term variation of price [5]. The simplest MA is called the Simple Moving Average (SMA), which can be obtained by calculating the mean value of the past N periods' prices, as expressed in the following equation:

$$SMA_N(t) = \frac{\sum_{n=0}^{N} A_{t-n}}{N}. \tag{3}$$

A more complex MA is the Weighted Moving Average (WMA). This indicator gives weights to each of the prices, with the objective of making the recent measures more preponderant, as opposite to SMA in which the measures for each period have the same weight and importance. This indicator can be obtained as follow:

$$WMA_N(t) = \frac{\sum_{n=0}^{N} A_{t-n} \cdot (N-n)}{\frac{N \cdot (N+1)}{2}}. \tag{4}$$

Exponentially Weighted Moving Average (EWMA) is also an indicator that values more the recent periods' prices, as opposite of every measure having the same importance. Contrary to the linear weight putting in WMA, EWMA assigns weights exponentially [5] through a constant (α), as present in the following equation:

$$EWMA_N(t) = \frac{\sum_{n=0}^{N} A_{t-n} \cdot \alpha^n}{\sum_{n=1}^{N} \alpha^n}. \tag{5}$$

$|\alpha|$ - Arbitrary coefficient comprised between $]0, 1[$.

The last MA that should be studied is the Exponential Moving Average (EMA), which instead of using N previous prices, uses just the last measure and the last value obtained for the EMA, as demonstrated in the next expression:

$$EMA_N(t) = A_t \cdot \frac{S}{1+N} + \left(1 - \frac{S}{1+N}\right) \cdot EMA(t-1). \tag{6}$$

$|S|$ - Smoothing factor, with the most common value being 2.
$|EMA(t-1)|$ - EMA computed in the the previous period.

Lastly, it should be noted that, since MAs are used to understand the present trend, these can be considered "trend following" indicators. Other particularity of these indicators is that its computation needs N previous entries, which means these indicators take no value in the first N measures of the data.

d) Moving Average Convergence/Divergence

Moving Average Convergence/Divergence (MACD) is an indicator that demonstrates the difference between two EMA [6], one short-term (usually 12 periods) and one long-term (26 periods), which can be computed by:

$$MACD_{line}(t) = EMA_{short}(t) - EMA_{long}(t). \tag{7}$$

$|EMA_{short}(t)|$ - Current value of short term EMA (S periods).
$|EMA_{long}|$ - Current value of long term EMA (L periods).

The result of this calculation results in the "MACD line". A n-period EMA of the MACD can be then computed, which is called the "signal line" (8). Traders use the crossing between these lines as buying rules, for example, when MACD crosses above signal line a buy signal is generated [17].

$$MACD_{signal}(t) = EMA_N(MACD_{line}). \tag{8}$$

e) Bollinger Bands

In the 1980's John Bollinger created a new type of adaptative bands, the Bollinger bands. These bands are a technical tool created with the objective of projecting trading rules that rely on measures of the central tendency and volatility to compose trading rules [18].

The components of this tool are three bands, a middle band (central tendency), computed using a SMA (, usually of 20 periods), and two trading bands placed above and below this moving average. Both of the trading bands' calculation (,10 and 11,) are based on the standard deviation of the samples of the moving average, which functions as a measure of volatility [16].

$$MB = SMA_N(Close) \tag{9}$$

$$UB = SMA_N(Close) + 2 \times \sigma(N) \tag{10}$$

$$LB = SMA_N(Close) - 2 \times \sigma(N) \tag{11}$$

$|MB|$ - Mibble Bollinger Band.
$|UB|$ - Upper Bollinger Band.
$|LB|$ - Lower Bollinger Band.
$|SMA_N(Close)|$ - Closing price's SMA of the of the last N periods.
$|\sigma(N)|$ - Standard Deviation over last N periods of SMA.

In his book, Bollinger also formulates two indicators, the Bandwith and the *%b*, which are derived from the adaptative bands. This thesis will only explore the *%b* indicator (12), which quantifies the price values relative to the Upper and Lower Bollinger Bands [19].

$$\%b = \frac{Close - LB}{UB - LB} \tag{12}$$

Several trading rules can be implemented using the adaptative bands and the indicators derived from them, one of which is to assume an overbought market when a commodity price continually touches the upper band, and oversold when it continually touches the lower band.

f) Money Flow Index

The Money Flow Index is a volume indicator created by Quong and Soudack [16]. To compute this index the first step is to obtain values for the money flow (14), which can be computed using the typical price (13) and volume of the market. The money flow values are then divided into positive and negative money flows. A final ratio between the positive and negative money flows is used to reach the Money Flow Index values.

$$TP = \frac{High + Low + Close}{3} \tag{13}$$

$$MF = TP + Volume \tag{14}$$

$$MFI = \frac{\sum PMF(N)}{\sum NMF(N)} \tag{15}$$

$|High|$ - Highest price value in the period.
$|Low|$ - Lowest price value in the period.
$|Close|$ - Price value in the close of the period.
$|TP|$ - Typical price of a given period.
$|MF|$ - Money flow of a given period.
$|PMF(N)|$ - Positive money flow in N period.
$|NMF(N)|$ - Negative money flow in N periods.

MFI values range from 0 to 100, which researches use to evaluate the overbought/oversold levels of the market. Typically values close to 100 implies that the market is overbought, and values close to 0 imply that the market is oversold.

g) Average Directional Movement Index

The Average Directional Movement Index is a technical indicator, created by J. Welles Wilder, used to verify the existence of market movement and to assess the strength of a trend [9] [20]. The first step is to evaluate the trend strength in each period using the Plus Directional Movement (18) and the Minus Directional Movement (19), which determines the strength of the bullish movement and the bearish movement, respectively.

$$High_{dif} = High(t) - High(t-1) \tag{16}$$

$$Low_{dif} = Low(t-1) - Low(t) \tag{17}$$

$$DM^{+} = \begin{cases} High_{dif} & , \quad if \quad High_{dif} > 0 \quad \wedge \quad High_{dif} > Low_{dif} \\ 0 & , \quad otherwise \end{cases} \tag{18}$$

$$DM^{-} = \begin{cases} Low_{dif} & , \quad if \quad Low_{dif} > 0 \quad \wedge \quad Low_{dif} > High_{dif} \\ 0 & , \quad otherwise \end{cases} \tag{19}$$

$|DM^{+}|$ - Plus Directional Movement
$|DM^{-}|$ - Minus Directional Movement
$|High(t)|$ - Highest price in period t
$|Low(t)|$ - Lowest price in period t.

Through the previous indicators the ADX values can be computed using the expressions:

$$DI_N^{+}(t) = \frac{\sum_{i=0}^{N-1} DM^{+}(t-i)}{TR} = \sum_{i=0}^{N-1} DM^{+}(t-i) \tag{20}$$

$$DI_N^{-}(t) = \frac{\sum_{i=0}^{N-1} DM^{-}(t-i)}{TR} = \sum_{i=0}^{N-1} DM^{-}(t-i) \tag{21}$$

$$DX(t) = \frac{|DI_N^{+}(t) - DI_N^{-}(t)|}{DI_N^{+}(t) + DI_N^{-}(t)} \times 100 \tag{22}$$

$$ADX_N(t) = \frac{\sum_{i=0}^{N-1} DX(t-i)}{N}. \tag{23}$$

$|DI_{+}|$ - Positive Directional Indicator
$|DI_{-}|$ - Negative Directional Indicator
$|DX|$ - Directional Movement Index.

Since the Directional Movement Index ranges from 0 to 100, the ADX values are also going to range

from 0 to 100, since the ADX is the average of the former. The theory states that 20/25 are a good values to distinguish a market with an ongoing trend (ADX$>$20/25), from a market without one (ADX$<$20/25) [20] [21]. However, fearing that the trend might not be strong enough to open a position, traders and researchers alike, use higher values ranging from 30-45 as indicator that a trend is strong enough to trade. It also should be noted that the bibliography suggests the use of 14 periods in the computation of the above indices.

One final remark regarding the previous computations is that, although both the Positive Directional Indicator 20 and the Negative Directional Indicator 21 are dependent on the True Range of the current period, this value can be discarded since both the numerator and the denominator, of the ADX formula, are divided by it.

h) Stochastic Oscillator

Developed by George Lane in the 1950's, the Stochastic Oscillator is a momentum indicator that aims to detect and evaluate the speed and momentum changes of a commodity's price. As explained by Lane in an interview, the momentum changes usually precedes changes in the price [22], which means that a correct evaluation of this indicator can a helpful trading tool.

The first step in the computation of the Stochastic Oscillator is to calculate the %K indicator 24, in which the highest and lowest prices of the last N (typically 14) periods are needed, as well as the current period's closing time. The second step is compute a N-period simple moving average of the %K indicator 25.

$$\%K(t) = \frac{Close(t) - Low_N}{High_N - Low_N} \times 100 \tag{24}$$

$$\%D(t) = SMA_N(\%K) \tag{25}$$

$|Close(t)|$ - Price at close of the current period.
$|Low_N|$ - Minimum value between the Low prices of the last N periods.
$|High_N|$ - Maximum value between the High prices of the last N periods.

The previous computations implies that the values of the Stochastic Oscillator (%D) are always comprised between 0 and 100, since both the numerator and the denominator are always positive. The interpretation of the aforementioned indicator made by Juszczuk et al. [8] is to assume the existence of an uptrend when an increase of the indicator values exists, and a downtrend when a drop in the values is observed. Other interpretation that could be made is to assume an overbought market when the value %D is higher than 80 and an oversold market when the value is lower than 20 [22].

2.3 Fuzzy Sets

When considering technical analysis based forecast of a financial market, the traditional approach to take would be a crisp one. In this approach a BUY/SELL signal is generated when conditions are satisfied, for example a BUY signal can be generated as the following:

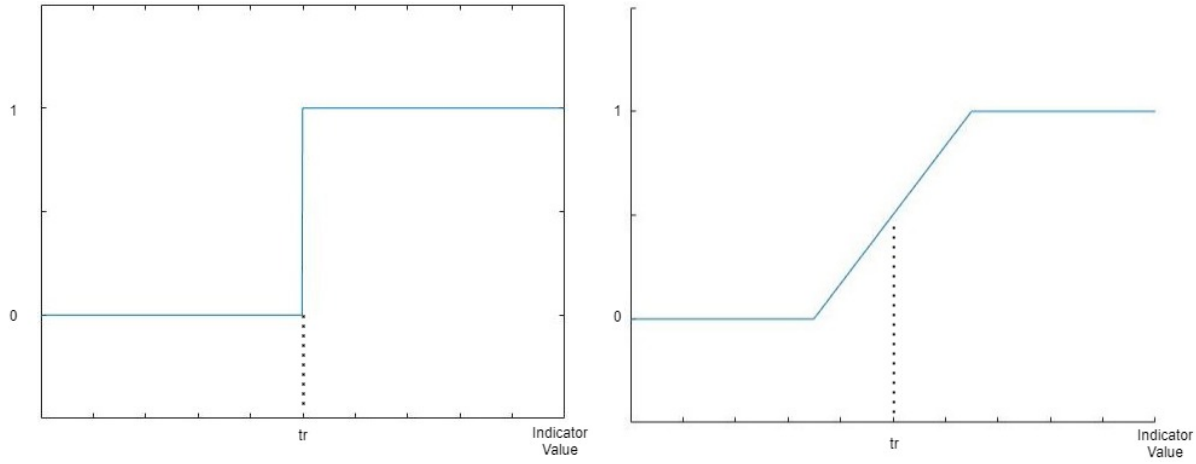
$$f_{BUY} = true \quad , if(cond_{1_{BUY}} = true \vee cond_{2_{BUY}} = true \vee \quad ... \quad \vee cond_{N_{BUY}} = true). \tag{26}$$

These conditions are binary, taking only values of *True* or *False* (1 or 0), and are derived from the indicators' value. One such condition can be exemplified by the logic present in 27, that corresponds to

a condition based in an indicator's value, in which, if this indicator exceeds tr, the condition becomes true. This condition's behaviour is graphically depicted in Figure 1a.

$$cond_1 = true \quad , if(ind_{1_{value}} > tr). \tag{27}$$

On the other hand, in a multi-criteria fuzzy approach, such as the ones proposed by Naranjo et al. [9] and by Juszczuk et al. [8], the signals BUY/SELL are generated by analyzing a vector of the type $y_c = [y_1, y_2, .., y_n]$, where n is the number of indicators, and $y_1, y_2, ..., y_n$ are the "fuzzied" values of each indicator, which are computed by inserting each of the indicator's value in its corresponding membership function. These membership functions can be constructed in a way that the value which the indicator takes is non-binary. An example of a membership function, that corresponds to the fuzzy approach explained previously can be observed in Figure 1b.



(a) Crisp approach Boolean truth.

(b) Fuzzy approach membership function.

Figure 1: Comparison between fuzzy and crisp approaches.

This type of approach, while making the computation more difficult, tolerates uncertainty in decision making [8], and can theoretically be used to make a more flexible approach.

2.4 Genetic Algorithm

The following section will serve as an introduction to genetic algorithms, which are used as search and optimization techniques [23]. These were based on the survival of the fittest principle, as enunciated by Charles Darwin in his book "On the Origin of Species".

2.4.1 Genetic Algorithm: Structure

A traditional GA, as the one described by Professor Luis Martí [23], would have a structure as described by the flowchart present in Figure 3. The first step, as with several algorithms, is the algorithm's initialization process, in which an initial population is generated. This population is comprised of individuals that can be described by a data structure, called chromosome. These data structures are normally composed of binary, real or integer arrays. An example of a simple binary chromosome is present in Figure 2.

Defining and designing the chromosome is a task that depends heavily on the problem at hand, since the solution of the algorithm will be the chromosome of the fittest individual when the termination criteria

Figure 2: Example of a binary chromosome.

is met. After defining the chromosome structure to be used, the population is randomly initialized.

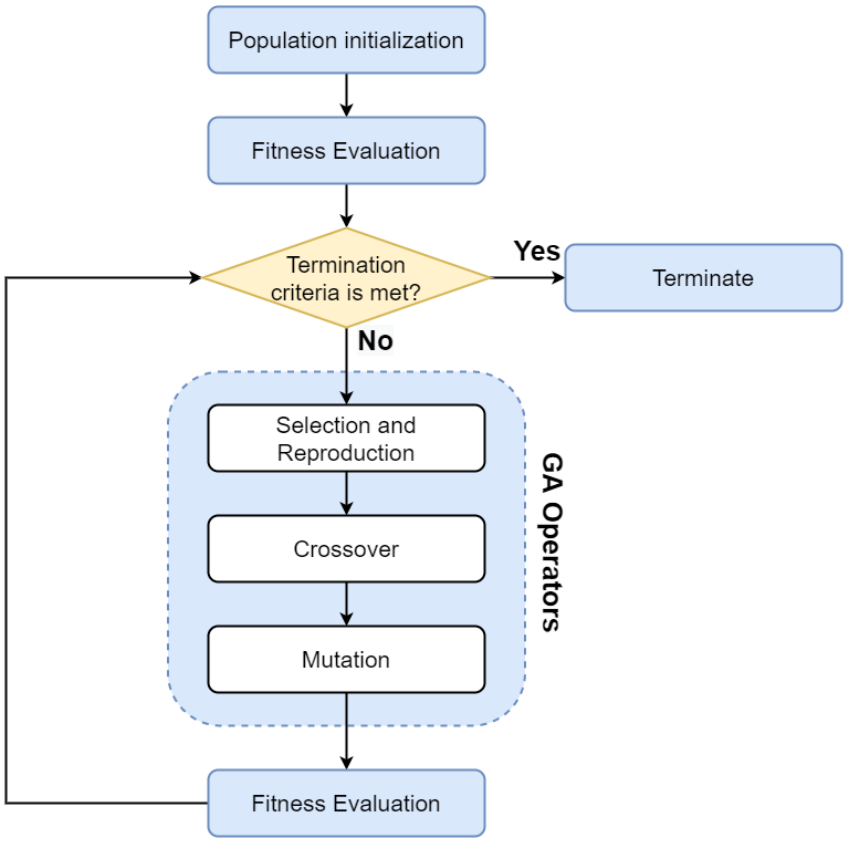


Figure 3: Structure of a genetic algorithm.

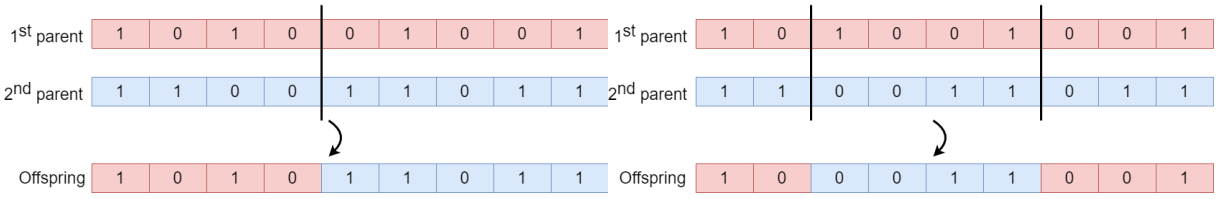
The next step is an initial fitness evaluation, which can be optional, with many researchers disregarding this step. A fitness evaluation should be made whenever changes to the population are made, with the objective of testing if an individual meets the termination criteria. In GA models tasked with forecasting a financial market, a very standard fitness evaluation is the Return on investment (ROI) measured for each individual in a simulated time series using training data, which can be formulated as in equation 28.

$$ROI(X) = \frac{Returns(X) - Investment(X)}{Investment(X)} \tag{28}$$

The individuals' fitness evaluation is also of great importance to the next step, selection. The objective of this step is to pick a set of individuals that will transit to the next "generation". There are several methods for selecting individuals, such as Truncation Selection [4] and Tournament Selection [5].

Afterwards, a new generation of individuals is generated, which is comprised of the set of selected individuals and a set of new individuals created through crossover. The process of crossover selects two individuals ("parents") from the selected pool, and through crossover techniques, generates new chromosomes ("offspring"). Most common techniques, such as Two-Cut-Point Crossover [5] or One-Cut Point Crossover [4], involve "cutting" the chromosome in one or more points and generating a different chromosome using parts of the "parents" chromosomes. The final GA operator procedure, before eval-

uating the population's fitness, is mutation. A mutation event is a low-probability occurrence, changing one or several genes (variable) inside a given chromosome.



(a) One-Cut-Point Crossover (b) Two-Cut-Point Crossover

Figure 4: Two examples of crossover methods.

This algorithm ends up converging to an optimal population, after several generations have passed. The best individuals from this population are then chosen to test their performance, which in this case, is to forecast the forex market.

2.4.2 Genetic Algorithm: Diversity

The main problem of trying to obtain an optimal solution using a genetic algorithm is the population stagnation around local maxima. This problem is explored by MIT's Professor Patrick H. Winston in his lectures [24], in which several populations are initialized and updated during several generations with the objective of obtaining an optimal solution, ending up to stagnate around a local maximum without reaching this solution. The origin of the stagnation is the loss of diversity that affects the population after converging to one of these local maxima, severely reducing the search space. Such event happens due to the fact that the best individuals of the current generation, which are the ones present in this local maximum, will be the progenitors of the next.

A basic approach to help maintaining a population diverse is to increase the probability of mutation [23], which is the only operator inside a standard GA that introduces diversity to a pool of genes. Other solution for this inherent problem is the one explained by Professor Winston [24], in which the evaluation of each individual is based on its fitness value coupled with how different it is from other individuals already selected, that is, the individuals that are most likely to be selected for the next generation are the ones that possess high fitness values and have less similarities between themselves. Other concept that may help improve diversity is "immigrants". This concept consists in the insertion of randomly created individuals in each generation [25].

The use of the methods stated above during several generations will keep the population diverse during the course of the algorithm, which in turn increases the solution search space, such that every possible solution (local maxima and, hopefully, the global maximum) may be found. Such effect is desired when searching for possible solutions, however, when the possible solutions are already covered by the population, some fine-tuning is necessary to single out the optimal solution to the problem, which can be achieved by drastically reducing the diversity of the population [24].

2.4.3 Genetic Algorithm: Adaptive Approach

Traditionally, GAs aim to solve static problems, whose solutions are precise and quick to obtain, on the other hand, when facing a real world problem, challenges arise. These challenges may be dealt with, using different and more effective approaches.

One such problem that arises from the work at hand is the fact that, although financial markets are proven to be cyclic in the long term, in the short term they are subject to continuous changes. These

constant changes mean that an instance of a technical indicator that works well on a particular trend may fail when the fitness landscape changes [25]. To deal with this challenge the genetic algorithm must be able to find the optimal solution for each instance without having to restart the whole algorithm, giving the model an adaptive behaviour.

2.5 State of the art

During this section, several pieces of investigation that are believed to be significant to the work at hand will be explored and discussed. Through the different approaches carried out to fulfill the objective, that is to forecast the forex market, will be evidenced and their results analyzed.

2.5.1 Use of Technical Indicators

The use of technical indicators to forecast financial markets is an approach widely carried out by several academics, while forecasting financial markets.

An example of the use of TI is the approach carried out by Gorgulho et al. [4], that has the objective of trying to forecast the stock market. In this work a GA model generates BUY/SELL signals based on technical rules which are designed through the technical analysis of 7 indicators (EMA, Hull Moving Average, ROC, RSI, MACD, True Strength Index and On Balance Volume) and through the Double Crossover method. The genetic algorithm proposed by Silva et al. [26], also leans in the stock market analysis, although it being more based on the fundamental indicators. Both of these works also explore a way to maximize profit using a portfolio composition module, which is achieved by choosing the best stocks in the market to invest in. Fernandez et al. [25] also proposes an evolutionary algorithm that uses data from technical indicators (namely RSI, EMA, MACD and WMA).

Literature more focused in the financial market at hand (FOREX) includes the approach authored by Almeida et al.[6], which uses 5 different technical indicators (RSI, ROC, MACD, MA and EMA). In their SVM-Genetic Algorithm hybrid approach, in which both the technical indicators and a sequence of prices are used to train both the SVM and Genetic Algorithm models and to generate BUY/SELL signals. Other works in which technical indicators are used in conjunction with a genetic algorithm, are the one published by Hirabayashi et al. [5], in which 4 indicators are used (RSI, Percent Difference from Moving Average, Rising (falling) Rate and the RSI of a EWMA), and the study made by Ozturk et al. [16]. The latter work uses a total of 24 indicators(parameterized using a GA), to capture the underlying ”rules” and testing them with several selection modules, namely a genetic algorithm and a greedy search heuristic.

A different use of technical indicators applied to the forecast of Forex, is present in the works of Naranjo et al. [9] (uses RSI, Average Directional Movement Index and a custom-made MACD) and Juszczuk et al. [8] (uses RSI, Commodity Channel Index, Stochastic Oscillator, DeMarker indicator, Bulls indicator and Moving Average of Oscillator). In both works, membership functions are created for each indicator ”fuzzing” the indicators’ values. This fuzzy information is used later to generate BUY and SELL signals through custom-made algorithms developed independently in each work.

2.5.2 Fuzzy Approaches

Several authors have explored different ways to approach the application of fuzzy logic on forecasting financial markets, these are the works that will be explored next. One such work is the study applied to the stock market done by Naranjo et al. [9] which, as previously mentioned, uses fuzzy information, gotten from technical indicators, together with a custom-made algorithm. This approach defines three labels for each of the 3 indicator, which generates $3^3 = 27$ combinations, and for each combination a

rule was made (buy, sell, etc). Ultimately, this approach uses these trading rules jointly with a risk control and money management modules to implement a trading system. The results presented demonstrate less profit than the benchmark Buy & Hold strategy (B&H), however this same strategy presents stability during bearish periods of the market, on the contrary to the B&H benchmark, which present poor results in these periods.

An author that was also previously mentioned, Juszczuk et al. [8], explored a fuzzy approach (based in the technical analysis) as well, applying it to the FOREX market. In this work the author also "fuzzifies" the values of the indicators, but studies variations in the fuzzy information of the indicators from $[t-1]$ and to $[t]$ too. With values obtained from the membership functions a dominance-based algorithm was developed, implementing a trading system that uses 3 to 6 of the indicators. This trading system was tested against an implementation of the same trading system using only crisp logic. The proposed trading system achieves a maximum of average accuracy of approximately, 70%, which corresponds to the fuzzy logic approach using 6 indicators and the highest number of test samples, while, for the same number of indicators and number of test samples, the approach that uses crisp logic only, achieves approximately 50% of average accuracy.

Bahrepour et al. [27], also implements a fuzzy approach, in which, instead of using fuzzy information generated by the technical indicators computed, focus only in information generated by "fuzzing" time series. The algorithm proposed in this work, uses self-organizing maps coupled with the fuzzy information, to predict prices of currency-pair. The proposed model is evaluated by comparing the standard deviation and mean (from MSE and MAPE error rates) of this approach and comparing them to two earlier studies, which this works surpasses, providing more accurate predictions.

2.5.3 Genetic Algorithm Approaches

In the literature presented, several works employ GAs with several different strategies in mind, with the main objective of finding an optimal solution. The majority of the genetic algorithms studied use trading rules based in TIs with the objective of finding the best forecasting model.

The most common method to generate a population in the different GA approaches present in the literature is to randomly generate each individual with interval constraints depending on the chromosome structure implemented. These structures are, however different among the approaches studied, with some implementing combinations between the technical indicators that the individuals uses, like the work of Gorgulho et al. [4], in which the combinations are made by generating weights for each of trading rules. Other works lean to an implementation that, besides including which of the trading rules are to be used, also comprises the parameters for the computation of each indicator, having the ability of choosing which is the best settings for a technical indicator in a given scenario.

Another common factor among the implementations studied is the population's fitness evaluation, which several of the authors base on the profit generated by each of the observed individuals. For example, both Hirabayashi et al. [5] and Ozturk et al. [16] use the net profit obtained by each individual as a fitness function, while Almeida et al. [6], Gorgulho et al. [4] and Silva et al. [26] use the previously mentioned ROI (equation 28) instead. While also focusing on the profit generated by an individual, Fernandez et al. [25], computes the fitness of each individual taking into account the maximum achievable profit. Another selection method that should be highlighted is the one explained by professor Winston in his lecture on GA, which uses a ranking system using a fitness function and the dissimilarity of the individuals, giving an higher probability of selection to individuals that have higher fitness values and that present more differences to the individuals already selected.

The genetic operators used by the literature demonstrate the plethora of viable options that exist. In terms of selection, Gorgulho et al. [4] applies the "Truncation Selection Method", in which all the

individuals are sorted using the fitness function, with the worst ending up discarded, and the best used in the crossover. Other method of selection is the one applied by Almeida et al. [6] and Hirabayashi et al. [5], the ”Tournament Selection Method”, in which a tournament is simulated between the population, with the more fit individuals having a higher probability to be selected for crossover, than weaker individuals.

A concept that is also used in the literature as a method of selection, namely by Ozturk et al. [16] and Hirabayashi [5], is the concept of ”elitism”. This method selects automatically the elites (individuals with the highest fitness value) for the population of the next generation. The percentage of each generation that is chosen as an elite is normally very low, with both of the works mentioned using 1%.

The GA implemented in previous researches also present operators that try to maintain the population diverse. A standard mutation operator, for example, is implemented by Gorgulho et al. [4], Ozturk et al. [16], Hirabayashi et al. [5] and Fernandez et al. [25], in which the probability of mutation (usually under 10%) defines how probable it is to replace a value inside an individual, by a random one inside the constraints of the chromosome. The implementation done by Bernardo et al. [6] uses a higher rate of mutation (20%), and instead of using a random value to replace the gene, uses in its place a value obtained through a Gaussian Distribution.

There are also several implementations of the crossover operator in the literature, with both Hirabayashi et al. [5] and Almeida et al. [6] using the ”Two-Cut-Point Crossover” method previously mentioned and also described in Figure 4b. In the work of Gorgulho et al. [4], a comparison is made between three crossover methods, namely ”Single Arithmetic Recombination”, ”Whole Arithmetic Recombination” and ”One-Cut Point” method, in which the conclusion was that the ”One-Cut Point” methodology presented the best results for the chromosome used. Another method that, as previously mentioned, helps in the maintenance of a population’s diversity is the concept of ”immigrants” used by Fernandez et al. [25] and Hirabayashi et al.[5], with the latter replacing 30% of the previous generation by randomly generated individuals.

2.5.3.1 Adaptive Mechanisms

As previously stated, adaptive mechanisms are needed so that genetic algorithms can readjust to the frequent changes of the financial markets. With that in mind Almeida et al. [6] implemented the concepts of ”hyper-mutation”, which doubles the normal mutation rate heavily increasing the gene diversion among the population, and ”hyper-selection”, which raises the selection pressure having the contrary effect in the population. In the scope of the algorithm, the use of these methods, respectively, widens and narrows the solution search space. In this same work the concept of ”associative memory” is also evidenced, due to the hybrid approach in which the SVM provides information about the market (environment), which is saved alongside the best individuals of each population, providing a solution where the GA uses this information to adapt to each market.

Finally it is suggested by Yang [28] and, more recently, Almeida [6], that an evaluation based on certain dynamic criterions must be made in order to study the behavior of the implemented dynamic approach to the fluctuations of the FOREX market. The criterions chosen for the study were:

- Predictability,
- Cyclicity,
- Visbility,
- Time-Linkage,
- Stability,

- Diversity Measures,
- Adaptability.

2.5.4 Chapter Conclusions

This chapter provided a background to the Foreign Exchange market, whose environment was used to implement an architecture that forecasts it. This architecture was idealised to utilise a Genetic Algorithm coupled with the technical analysis of this market, as well a fuzzy logic approach. A background to these machine learning techniques is also provided in this chapter, as well as a description and analysis of several comparable frameworks, implemented in previous works.

Table 2 presents the most relevant literature in the context of the implemented framework.

Table 2: Review of the state of the art.

Work,Year	Heuristic	Approaches	TI	Financial Applications	Benchmark
[6], 2018	SVM-GA	SVM, Dynamic GA, Technical Analysis	RSI, ROC, MACD, MA and EMA	FOREX Market EUR/USD	ROI, Precision, Recall and Accuracy comparing with Static GA, Random Walk, B&H and S&H
[5], 2009	GA	GA, Technical Analysis	RSI, Rising (Falling) rate, Percentage Difference of MA and RSI of EWMA	FOREX Market USD/JPY, EUR/JPY and AUD/JPY	Percentage Profit, Comparison with NN
[16], 2016	GA	GA, Technical Analysis	24 TI - RSI, MACD, etc.	FOREX Market EUR/USD, GBP/USD	Net Profit, Average Profit per trade
[4], 2011	GA	GA, Technical Analysis, Portfolio Composition	EMA, Hull Moving Average, ROC, RSI, MACD, True Strength Index and On Balance Volume	Stock Market	ROI; Comparison with B&H and Random Walk
[25], 2008	GA	Dynamic GA, Technical Analysis	RSI, EMA, MACD and WMA	Stock Market	Mean profit and Standard deviation compared with B&H and Random Walk
[26], 2014	GA	GA, Fundamental Analysis, Technical Analysis Portfolio Composition	X	Stock Market	Return and Variance compared with SP&500
[9], 2015	Fuzzy Logic	Fuzzy Logic, Technical Analysis	Average Directional Movement Index, RSI and custom-made MACD	Stock Market	Net Profit, Comparison with B&H
[8], 2020	Fuzzy Logic	Fuzzy Logic, Dominance-based algorithm, Technical Analysis	Commodity Channel Index, Stochastic Oscillator, RSI DeMarker indicator, Bulls indicator and Moving Average of Oscillator	FOREX Market 26 currency pairs	Efficiency compared with crisp approach
[27], 2011	Fuzzy Logic	Fuzzy time series	X	FOREX Market USD/EUR, USD/GBP and EUR/GBP	Standard deviation and Mean compared with two earlier studies

Chapter 3

Model Architecture

In this chapter a solution for the problem at hand is unveiled and proposed. Firstly, a brief overview of the implemented algorithm and the model's architecture are presented. This first segment is then followed by several sections detailing each components and their purpose.

3.1 Overview

As stated in the introductory section, the main objective of this thesis is to create a model that predicts the FOREX market effectively, and study the viability of this method coupled with a fuzzy approach. With that objective in mind, several algorithms already applied to the subject by previous researches were considered as a starting point. Among them were algorithms such as Random Forests, Neural Networks and Genetic Algorithms. Due to its previous successes in the area, a Technical Analysis based Dynamic Genetic Algorithm was chosen as the fundamental idea.

A simple overview of the implemented algorithm is represented in Figure 5, which highlights several key parts of the model.

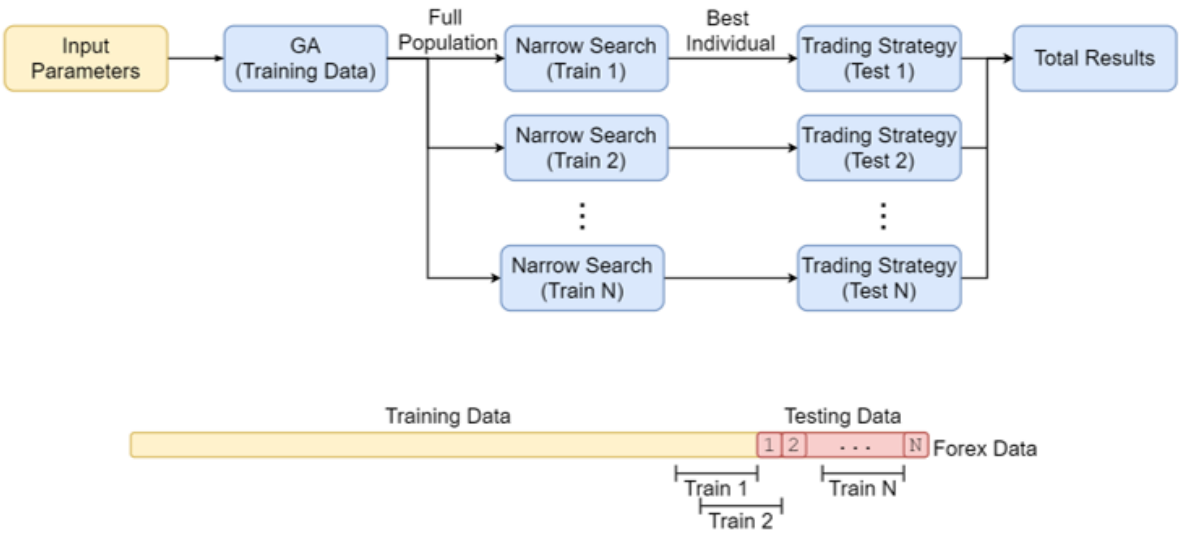


Figure 5: Simplified overview of the implemented model.

The overall idea of the model is to implement the ideas of MIT's Professor Winston [24], by first creating a diverse population that covers all the solution space, using a traditional GA. After this population is trained, a follow up training takes place (Narrow Search), in which a smaller training data is used to

shrink the population's diversity so that the resulting population converges to a solution used to predict the next market's movements. Each of these components deserve its own section, where they will be succinctly explored, while trying not to loose focus of the overall algorithm.

Before diving into the first component of the algorithm, the overall architecture of the model should also be introduced. This architecture can be divided in three major layers, the *User Layer*, the *Data Layer* and the *GA Layer*, represented in Figure 6. The first layer consists of the parameters given by the user to the model, such as the settings of the trading rules, and of the Genetic Algorithm. These inputs are mostly used in the initialisation of several objects.

The *Data Layer* contains the modules responsible for the trading rule computation and the generation of the buy and sell signals. This layer also carries out the simulations of the created trading strategies in the real world constraints. Finally, the *GA Layer* contains all the operations concerning the Genetic Algorithm, specifically the population creation, and the diversity mechanisms.

It should be noted that the two last layers are dependent of the *User Layer*, since the first step in the model is to load the necessary data and parameters. On the other hand, an interdependence between the *Data Layer* and the *GA Layer* is evident, since the computation of the technical indicators use the individual's chromosome values, and the GA utilizes simulation results to compare individuals.

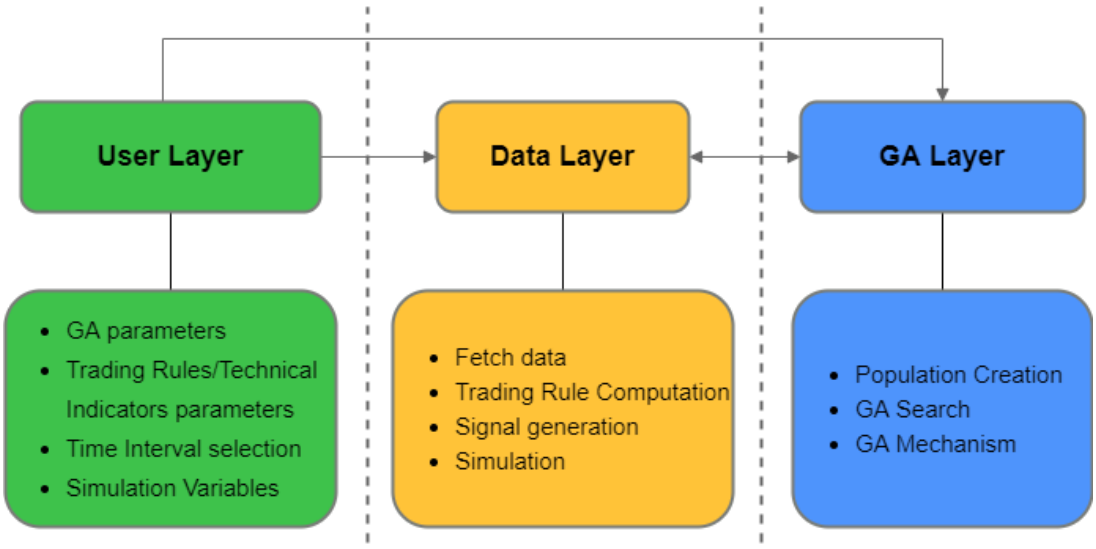


Figure 6: Overall architecture.

3.2 Inputs

Since the algorithm is based on the Technical Analysis of the FOREX market, the currency pair historical prices and the Trading Rule settings are essential inputs for the algorithm. Besides these inputs, the model would also need parameters for the Genetic Algorithm, and settings for the simulation module. From all the inputs only the *Historical Data* will be explored in this section, while the remaining will be introduced in the sections about the algorithm components that make use of them.

3.2.1 Historical Data Import

As stated previously, importing data is one of the first steps in every algorithm. As a result, the currency pair's historical data import is also the starting point of the implemented model, which requires a file with all the historical data and time intervals that are going to be used.

This data consists in the hourly historic data feed of the EUR/USD currency pair from the year 2014 through 2020. Each of the periods represents a row of data which is characterised by a set of features (columns). The historical data used throughout the entire project was obtained through the DukasCopy website [29].

The first feature present in this file is the ”Gmt time”, which is a descriptor that states the time and date of the given period. The other features that characterise each period are the prices of the currency pair in the beginning (”Open”) and the end (”Close”) of the period, as well as the highest (”High”) and the lowest (”Low”) prices that existed during that period. The number of traded units of the currency pair during this a given period (”Volume”) is the last feature to be present in the imported data. All the information described previously is organised in a data frame as the one represented in Figure 7.

Gmt time	Open	High	Low	Close	Volume
01.01.2014 22:00:00.000	1.37553	1.37652	1.3739	1.37585	1088.61
01.01.2014 23:00:00.000	1.37599	1.3772	1.3759	1.37653	1636.28
02.01.2014 00:00:00.000	1.37653	1.37691	1.37585	1.37592	1794.35
02.01.2014 01:00:00.000	1.3759	1.37653	1.37507	1.3763	1841.79
02.01.2014 02:00:00.000	1.37629	1.37674	1.37598	1.37662	1610.1
02.01.2014 03:00:00.000	1.37662	1.37746	1.37632	1.3766	1765.99
02.01.2014 04:00:00.000	1.37661	1.37661	1.37439	1.3746	2003.66
02.01.2014 05:00:00.000	1.3746	1.37581	1.37445	1.37573	1970.32

Figure 7: Slice of the data used during the project.

This data is imported from the *.csv* file using the *Python* library [30], which generates a complete data frame with all the features. The next step is to use the time intervals input to trim the complete data to the data that will indeed be used. The outcome is a data frame with the historical data of a currency pair that will be used to compute trading rules and to simulate individuals during the training component.

A final remark to make is that only the periods during which the market is active and being updated, are present in the imported data. Therefore, periods that occur during holidays and weekends are absent from the data frame, as it can be observed in the Figure 7, which evidences the first prices of the year 2014.

3.3 Genetic Algorithm

During this section the structure of the Genetic Algorithm will be described thoroughly, particularly, how a population is created, how its trained, and how its tested. The inputs that each of these operations require will also be highlighted.

3.3.1 Population Creation

As already stated, the first step in every Genetic Algorithm is to create the population. To generate the individuals that compose the population, first the user must choose which trading rules to use. After importing the trading rule settings, chromosomes are generated, which are then used, in conjunction with the data frame described in section 3.2.1, to create individuals. A simplified flowchart of the process is represented in Figure 8.

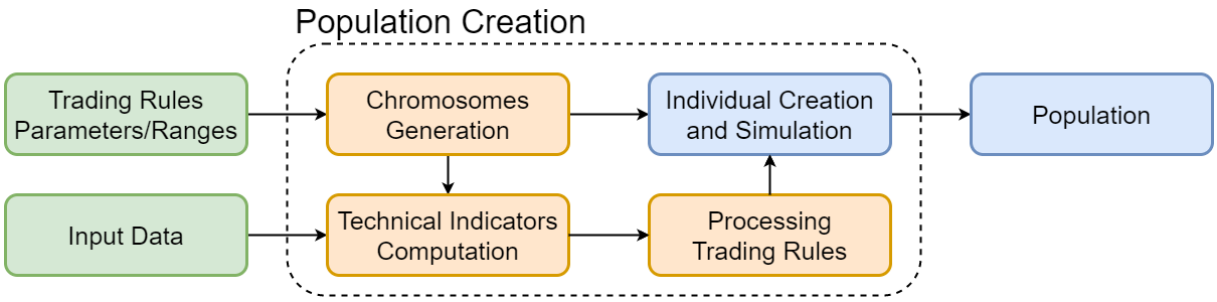


Figure 8: Process of creating the population.

3.3.1.1 Trading Rules: Description

Accordingly, in this section each of the implemented trading rules will be covered, highlighting the necessary variables in their computation. It should also be mentioned that the calculations of all the technical indicators mentioned during the section on Technical Analysis (2.2.2), was achieved using the *Python* library *ta* [31] and its respective documentation [32]. The calculations of said technical indicators will also be briefly explored, in spite of their implementation being achieved by the author of the aforementioned software.

a) RSI Rule

The RSI rule, as its name suggests, is a simple implementation of the technical indicator. This index relies on the absolute values of the rising and falling widths to detect if a market is overbought or oversold. As such the price values at market's close and a number of samples to use are needed, as suggested by the aforementioned documentation [32].

Therefore this technical rule is easily implemented using three variables, the number of past samples used, and the oversold limit variable (*floor*) and the overbought limit variable (*ceiling*) limits, which generate a buy and sell signal respectively.

b) ROC Rule

Again, as its name suggests, the ROC rule is a simple implementation of the TI with the same name. This indicator measures the ratio between two currency values separated by N periods. The values of ROC are then processed by a trading rule, generating a buy/sell signal. This trading rule is featured in the researches by Almeida et al. [6] and Gorgulho et al. [4], and states that buy and sell signals should be generated when the ROC values are, respectively, positive and negative.

c) EMA Rule

The EMA rule's implementation uses the price value of the current period and compares it with the Exponentially Moving Average value of the last N periods, using the following computation:

$$EMA_{rule}(t) = \frac{Close(t) - EMA_N(t)}{Close(t)} * 100. \tag{29}$$

The previous equation results in positive and negative percentages, which represent a buy and sell signal, respectively.

d) 3-EMA Rule

As the name suggests, to compute the 3-EMA Rule three different Exponentially Moving Averages are required. Each of the EMA are computed using a different N, hence existing a short EMA (lowest N), a medium EMA and a long EMA (highest N).

The investigated trading rule [33] states that a buy signal should be employed when the fast EMA crosses from below to above the medium EMA, while both of the averages and the price of the currency are above the slow EMA. It also states that when the fast EMA crosses from above to below the medium EMA, while both of the averages and the price of the currency are below the slow EMA, a sell signal should be generated.

Although the aforementioned trading rule states that the fast EMA has to cross the medium EMA to generate buy and sell signals, this characteristic was overlooked in its implementation. As such, the buy and sell signals are generated following the simple method represented in Algorithm 1, which consist in a series of *IF* statements comparing the values of the EMAs and the currency price. This algorithm uses the 3_EMA_{rule} indicator, which is a custom indicator that utilizes the values of the short and medium EMA to produce positive and negative percentages, as evidenced by equation 30.

Algorithm 1: 3-EMA rule implementation

```
for t = 1, 2, ... do
    if 3_EMA_rule(t) > 0 then
        if EMA_short(t), EMA_medium(t), Close(t) > EMA_long(t) then
            Signal(t) = 1;
    else if 3_EMA_rule(t) < 0 then
        if EMA_short(t), EMA_medium(t), Close(t) < EMA_long(t) then
            Signal(t) = -1;
```

$$3\_EMA_{rule}(t) = \frac{EMA_{fast}(t) - EMA_{medium}(t)}{EMA_{fast}(t)} * 100. \tag{30}$$

e) MACD Rule

As stated in section 2.2.2.1, the crossing between the “MACD line” and the “MACD signal” can be used to generate trading signals. The implemented trading rule is featured in the research of Almeida et al. [6], which uses an histogram between the two lines [34], computed as:

$$MACD_{hist}(t) = MACD_{line}(t) - MACD_{signal}(t). \tag{31}$$

This difference results in positive and negative values, which are used in the generation of buy and sell signals, respectively.

Finally, it should be noted that, since the Moving Average Convergence/Divergence indicator uses the difference between a short EMA and a long EMA, and an Exponentially Moving Average of the resulting signal, three averages of different periods are used. Therefore, the variables of this trading rule are the three distinct Ns.

f) Bollinger Rule

The Bollinger trading rule was implemented using the *%b* indicator as a guideline. As stated previously, this indicator is derived from the Bollinger Bands, using the formula 12. Through the analysis of this formula, some conclusions about the *%b* values can be drawn:

- $\%b = 1$ - the last price is at the upper band (market overbought).
- $\%b = 0.5$ - the last price is at the middle band.
- $\%b = 0$ - the last price is at the lower band (market oversold).

The architecture of this trading rule operates very similarly to the RSI rule, as such, buy and sell signals are generated when the indicator is less than the *floor* and when it surpasses the *ceiling*, respectively.

Overall, the Bollinger trading rule's implementation uses three variables, the number of periods used (N), and the oversold (*floor*) and overbought (*ceiling*) limits.

g) MFI Rule

The MFI rule was also implemented following the guidelines of the RSI rule, using the MFI indicator. Hence, after computing the indicator's values, the rule implemented generates a buy signal if the indicator is lower than the oversold limit, and a sell signal if the indicator is above the overbought limit.

As a whole, the architecture of the MFI rule uses, the number of periods used (N), and a *floor* and a *ceiling*, following the membership function of the RSI and Bollinger rules.

h) Bollinger-MFI Rule

In a previous research, Ozturk et al. [16] explored the trading rule "*%b-MFI*" based in using the Bollinger and MFI indicators together.

This rule combines the logic of the previous two trading rules, the Bollinger and the MFI. As such, when both the indicators are higher than their respective *ceiling* a sell signal is generated, and when both the indicators are lower than the respective *floor* the trading rule generates a buy signal.

Overall this trading rule's implementation requires values for five variables. The number of periods used (N), two *floor* values, and two *ceiling* values.

i) ADX Rule

As stated by Naranjo et al. [9], the Average Directional Movement Index quantifies the trend strength, regardless of the direction, which makes it impossible to generate a trading rule using solely this index. As such, the ADX rule was implemented using the ADX and the SMA_{rule}, a custom indicator, to generate buy and sell signals. The latter is computed by the equation:

$$SMA_{rule}(t) = \frac{Close(t) - SMA_N(t)}{Close(t)} * 100, \tag{32}$$

and returns positive and negative percentages.

In essence, the implementation of this rule generates either a buy or a sell signal, depending if the SMA_{rule} is positive or negative, respectively. However the signal is only created if the ADX value exceeds the *ceiling* value. As a whole, the ADX rule uses two sample periods (one for the ADX and the other for SMA), and a *ceiling* value, which makes up a total of three variables.

j) Stochastic Rule

The final trading rule implemented in the scope of the project was the Stochastic rule. This rule follows the framework of several previously described rules, namely the RSI, the MFI and the Bollinger. As such, this rule utilizes a technical indicator (, in this case the Stochastic Oscillator (%D),) to generate buy and sell signals when its values are below the *floor*, and when they surpass the *ceiling*, respectively.

In general, three variables are used in the implementation of the rule, two offsets and the number of periods used in the computation of the Stochastic Oscillator.

3.3.1.2 Trading Rules: Review

In summary, to compute each of the implemented trading rules the *Data Layer* requires the input data present in the data frame described in section 3.2.1, such as the *Close*, *Low* and *High* price values, and certain variables exclusive to each rule.

Every of the aforementioned rules and their several components are succinctly displayed in Table 3. It is important to note that several custom indicators (, namely EMA_{rule}, 3_EMA_{rule} SMA_{rule},) were computed with the objective of facilitating the implementation of the fuzzy approach, which will be explored later on.

Table 3: Review of all the implemented trading rules.

Trading Rule	**Used Indicators**	**Input Data**	**Variables**	**Signal**	
				Sell	Buy
RSI	RSI	Close	N $ceiling$ $floor$	RSI $> ceiling$	RSI $< floor$
ROC	ROC	Close	N	ROC < 0	ROC >0
EMA	EMA$_{rule}$	Close	N	EMA$_{rule} < 0$	EMA$_{rule} > 0$
3-EMA	EMA$_{short}$ EMA$_{medium}$ EMA$_{long}$ 3_EMA$_{rule}$	Close	N_{short} N_{medium} N_{long}	3_EMA$_{rule} > 0$ $\wedge$ EMA$_{long} < Min_{var}$	3_EMA$_{rule} < 0$ $\wedge$ EMA$_{long} > Max_{var}$
MACD	MACD	Close	N_{signal} N_{short} N_{long}	MACD < 0	MACD > 0
Bollinger	%b	Close	N_{short} $ceiling$ $floor$	%b $> ceiling$	%b $< floor$
MFI	MFI	Close, High, Low and Volume	N_{short} $ceiling$ $floor$	MFI $> ceiling$	MFI $< floor$
Bollinger-MFI	%b-MFI	Close, High, Low and Volume	N_{short} $ceiling_{MFI}$ $floor_{MFI}$ $ceiling_{\%b}$ $floor_{\%b}$	%b $> ceiling_{\%b}$ $\wedge$ MFI $> ceiling_{MFI}$	%b $< floor_{\%b}$ $\wedge$ MFI $< floor_{MFI}$
ADX	ADX SMA$_{rule}$	Close, High and Low	N_{ADX} N_{SMA} $ceiling$	ADX $> ceiling$ $\wedge$ SMA$_{rule} < 0$	ADX $> ceiling$ $\wedge$ SMA$_{rule} > 0$

Table 3 continued from previous page

Stochastic	%D	Close, High and Low	N_{short} $ceiling$ $floor$	%D $>$ $ceiling$	%D $<$ $floor$

Note: $Min_{var} = Min(EMA_{short}, EMA_{medium}, Close)$

$Max_{var} = Max(EMA_{short}, EMA_{medium}, Close)$

3.3.1.3 Chromosome Generation

To design the chromosome's structure, some inspiration was drawn from the architecture of the work from Almeida et al. [6], in which chromosomes are composed by one weight and one parameter, for each trading rule implemented.

In this work however, chromosomes are created with one weight and several parameters (variables) for each trading rule. The general structure of the chromosomes generated during this work is represented in Figure 9. From this figure one can conclude that the chromosome is split into several segments, one for each trading rule, which may not have the same number of parameters.

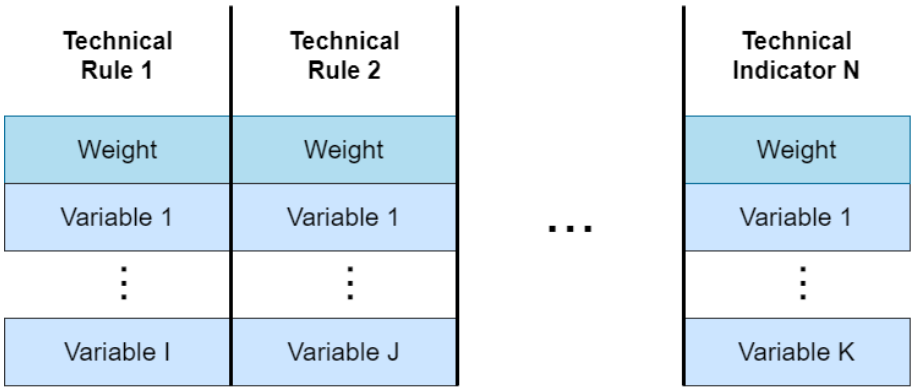


Figure 9: Representation of the general structure of this architecture's chromosome.

Technically speaking, the design represented in the previous figure was implemented in *python* using a simple list of arrays, in which each array has the values that are pertinent for one of the chosen trading rules. For instance, a chromosome's structure in a model which only features four trading rules (RSI, EMA, Bollinger and Stochastic rules), is:

$$chromosome = \{[weight, N, ceiling, floor], [weight, N], [weight, N, ceiling, floor], [weight, N, ceiling, floor]\}.$$

This example is also represented in Figure 10.

RSI Rule	EMA Rule	Bollinger Rule	Stochastic Rule
Weight	Weight	Weight	Weight
N	N	N	N
ceiling		*ceiling*	*ceiling*
floor		*floor*	floor

Figure 10: Example of a chromosome in a model with RSI, EMA, Bollinger and Stochastic trading rules.

As stated previously, the arrays, that compose the list, represent each of the selected trading rules, and have to be generated using a guideline for the parameters that describe the computation of the trading rule. For this reason, each trading rule has a predefined range for each of the parameters, which are represented in Table 4. These predefined ranges are employed in conjunction with the standard *python* library, *random*, to generate the arrays, which are then appended to the list. This way, the values of the chromosomes follow the predefined guidelines, and are randomly generated. It should also be mentioned that the predefined range for the weight parameter is a float value between 0 and 1 ($[0, 1]$).

Table 4: Ranges of the variables of each trading rule.

Trading Rule	Variables	Value Range	Trading Rule	Variables	Value Range
RSI	N	[5, 30]	MFI	N	[15, 60]
	$ceiling$	[65, 75]		$ceiling$	[75, 85]
	$floor$	[5, 30]		$floor$	[15, 25]
ROC	N	[15, 120]	Bollinger-MFI	N	[15, 60]
EMA	N	[10, 150]		$ceiling_{MFI}$	[75, 85]
3-EMA	N_{short}	[5, 15]		$floor_{MFI}$	[15, 25]
	N_{medium}	[15, 35]		$ceiling_{\%b}$	[85, 100]
	N_{long}	[40, 60]		$floor_{\%b}$	[0, 15]
MACD	N_{signal}	[5, 15]	ADX	N_{ADX}	[10, 25]
	N_{short}	[10, 25]		N_{SMA}	[20, 60]
	N_{long}	[20, 40]		$ceiling$	[20, 35]
Bollinger	N	[15, 60]	Stochastic	N	[10, 50]
	$ceiling$	[85, 100]		$ceiling$	[75, 85]
	$floor$	[0, 15]		$floor$	[15, 25]

To conclude this section it must be said that, the reasons that led to the development of this method to implement the chromosome's structure were, mainly, the ease of creation and manipulation, which are critical due to the fact that several of the GA operators, described in section 2.4.1, are performed using the individual's chromosomes, such as the mutation and crossover operators.

3.3.1.4 Individual Creation

After generating a number of chromosomes equal to the size of the population intended, the implemented architecture creates an individual for each of the chromosomes. In this architecture, an individual

possesses three main components, which are the chromosome that characterises it, a data frame which details how the individual performs in the training data, and finally a fitness score that allows the individual to be compared with others. Since the chromosomes have already been created, the first component of each individual already exist.

The second step is to create the data frame. This data frame was constructed to contain the trading signals of each trading rule, a general trading signal, and the simulation values of the general trading signal in the training data. To compute the trading signals of the selected trading rules, the model first uses a few of the parameters present in the chromosome and the data frame with the input data (section 3.2.1) to compute the technical indicators (section 2.2.2.1), necessary to process the trading rules selected, which were described in detail in section 3.3.1.1. The selected trading rules use as inputs the indicators, as well as the remaining parameters and the input data, to generate a trading signal composed by buy signals (1), sell signals (-1) and neutral signals (0), referent to the training period.

Each trading rule produces a signal, which are then combined to produce the general trading signal. The previously mentioned weight parameter enters in effect at this stage, being the main component in the combination of the trading signals, giving emphasis to some trading rules to the detriment of others. This follows the guideline in equation 33, which generates an array with positive and negative float values. This types of values are however incompatible with the previous notion of buy and sell signal, which are symbolised by the number 1 and -1.

$$general\_signal = \sum_{i=0}^{N} signal_i \cdot weight_i \tag{33}$$

Therefore this general signal has to be normalised prior to simulation. To achieve this, two methods were implemented.

The first method consists in simply generate a ternary signal using the values -1, 0 and 1, symbolising sell, get even and buy respectively. To achieve this, the a simple *for* cycle, described by the pseudo-code in 2, was devised. This cycle generates a trading signal from the *general_signal* created previously.

Algorithm 2: Method 1 to normalise the general trading signal.

$buy = 1/3 * max(general\_signal)$
$sell = 1/3 * min(general\_signal)$

for *i* **in** *general_signal* **do**
  if $i > buy$ **then**
    $i = 1$
  else if $i < sell$ **then**
    $i = -1$
  else
    $i = 0$

However, a second method was devised by following the logic present in algorithm 3. As it can be deduced, the implemented architecture processes the signal array obtained previously into a quinary trading signal composed by integer values which may symbolise light/heavy buy or sell positions, or even a neutral position.

After obtaining the normalised trading signal, whose function is to state the trading strategy of its individual for the period at hand, the objective becomes to simulate the strategy represented by it and evaluate it. For this reason the simulation module in the *Data Layer* was implemented, which receives as input a data frame, generated using the *pandas* library, containing the necessary data of the time period

Algorithm 3: Method 2 to normalise the general trading signal.

$buy_1 = 1/3 * max(general\_signal)$
$buy_2 = 2/3 * max(general\_signal)$
$sell_1 = 1/3 * min(general\_signal)$
$sell_2 = 2/3 * min(general\_signal)$

for *i* **in** *general_signal* **do**
  if $i > buy_2$ **then**
    $i = 2$
  else if $buy_2 > i > buy_1$ **then**
    $i = 1$
  else if $i < sell_2$ **then**
    $i = -2$
  else if $sell_2 < i < sell_1$ **then**
    $i = -1$
  else
    $i = 0$

to simulate. This data consists in the dates, the trading signal and the Close values of the considered currency pair. The use of the Close price is an approximation, since the decision to buy or sell in real time would take seconds, which would be enough for the price to change.

To this effect, the data frame described previously in section 3.2.1 is utilized as a stepping stone, since it has the necessary data, missing only the trading signal. To prepare the data frame, to serve as input, the implementation utilizes the *pandas* library to shave the unnecessary data and to append the general trading signal.

3.3.1.5 Simulation of Individuals

The final step to create an individual in this architecture is to test its trading strategy using the simulation module. This module follows the simplified logic present in the algorithm 4, in which positions are opened and closed using the trading signal of the current and the past period as conditions.

While the process of opening and closing positions may appear trivial, that could not be further from the truth. Several functionalities are ”hidden” behind the functions *update_position()*, *close_position()* and *open_position()*, which are present in the previous pseudo-code, and don’t receive the emphasis that they should.

Among these functionalities, one should point out the tracking of profit and money spent, which allows the user to better analyse the trading strategy’s performance. This tracking is achieved in grand part thanks to equations 34 (, for buy position,) and 35 (, for sell position), which compute how much an open position is worth in a given period.

$$current\_profit = leverage * initial\_investment * (current\_value - entry\_value) \tag{34}$$

$$current\_profit = leverage * initial\_investment * (entry\_value - current\_value) \tag{35}$$

Features such as the stop loss and stop gain should also be highlighted, since their function is to close positions that are either no longer profitable, or have achieved the profit targeted. These features were implemented using two input factors, one for each feature, which are then multiplied by the money

Algorithm 4: Pseudo-code of the procedure of normalising the general trading signal.

```
position = closed
past_signal = 0
for signal in trading_signal do
    if signal! = 0 then
        if position == open and signal == past_signal then
            update_position()
        else
            if position == open then
                close_position()
            open_position()
            position = open
    else
        if position == open then
            close_position()
            position = closed
    past_signal = signal
```

initially invested in the current open position. If the value of position eventually reaches any of the results of the multiplication, this position is closed and a new one is opened.

The last functionality implemented simulates a Double Down trading strategy. To achieve this, the trading signal automatically doubles down the value invested if the position has become unprofitable. This feature works by closing the current active position and doubling the money invested initially. However this event only takes place once the current position is worth less than 85% of what was originally invested in it. The new position is then only closed if the stop loss or gain criteria is met or when the next signal represents a position contrary to the one its on.

Finally it should be pointed out that, during the simulation, the values used to produce the statistical results are also computed, namely the confusion matrix values for both the trading signal and for the positions.

3.3.1.6 Statistics and Fitness

As stated before, during the individual's simulation the confusion matrices for the trading signal and the positions are created. The label values for the trading signal were computed by following the subsequent logic:

- True Positive - $previous\_signal > 0 \quad \wedge \quad current\_value > previous\_value$;
- False Positive - $previous\_signal > 0 \quad \wedge \quad previous\_value > current\_value$;
- True Negative - $previous\_signal < 0 \quad \wedge \quad previous\_value > current\_value$;
- False Negative - $previous\_signal < 0 \quad \wedge \quad current\_value > previous\_value$.

While the confusion matrix values regarding the positions are computed by following different guidelines:

- True Positive - Buy positions with profit;
- False Positive - Buy positions with loss;

- True Negative - Sell positions with profit;
- False Negative - Sell positions with loss.

Using the values of the confusion matrices described above, several indicators used in the posterior analysis of each individual's performance, such as the sensitivity (36), the specificity (37, the precision (38), accuracy (39) and the F1 score (40).

$$sensitivity = \frac{TP}{TP + FN} \tag{36}$$

$$specificity = \frac{TN}{TN + FP} \tag{37}$$

$$precision = \frac{TP}{TP + FP} \tag{38}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{39}$$

$$f1\_score = 2 \cdot \frac{precision \cdot sensitivity}{precision + sensitivity} \tag{40}$$

The final step of the creation of each individual is the computation of a fitness score, so that the individuals can be compared between themselves. In the implemented architecture it was decided to use two fitness components multiplied by each other. The first component is the Return on Investment (equation 28), which, as stated in the Background section 2.4.1, is a very standard fitness indicator that measures the profit during a certain period, taking into account the investments made. The second component is the accuracy computed using the labels regarding the profit/loss of the positions. The computation of the complete fitness indicator follows equation 41.

$$fitness\_score = ROI \cdot accuracy\_positions \tag{41}$$

The fitness score was implemented using both components instead of a single one with the objective of giving the best score to individuals that have a good balance between generating a high profit margin and having a good percentage of profitable positions.

3.3.2 Population Training (Wide Search)

After creating the population, the next step in any Genetic Algorithm is to train it. As such the next section presents the full implementation of the training process, which is represented in the flowchart present in Figure 11. In this figure all the main functionalities are highlighted, as well as the inputs of the process, both of which are described in the following sections.

An observation of note is that it is somewhat deducible that the generic GA training, present in 3, was used as a guideline to the implementation of the training process.

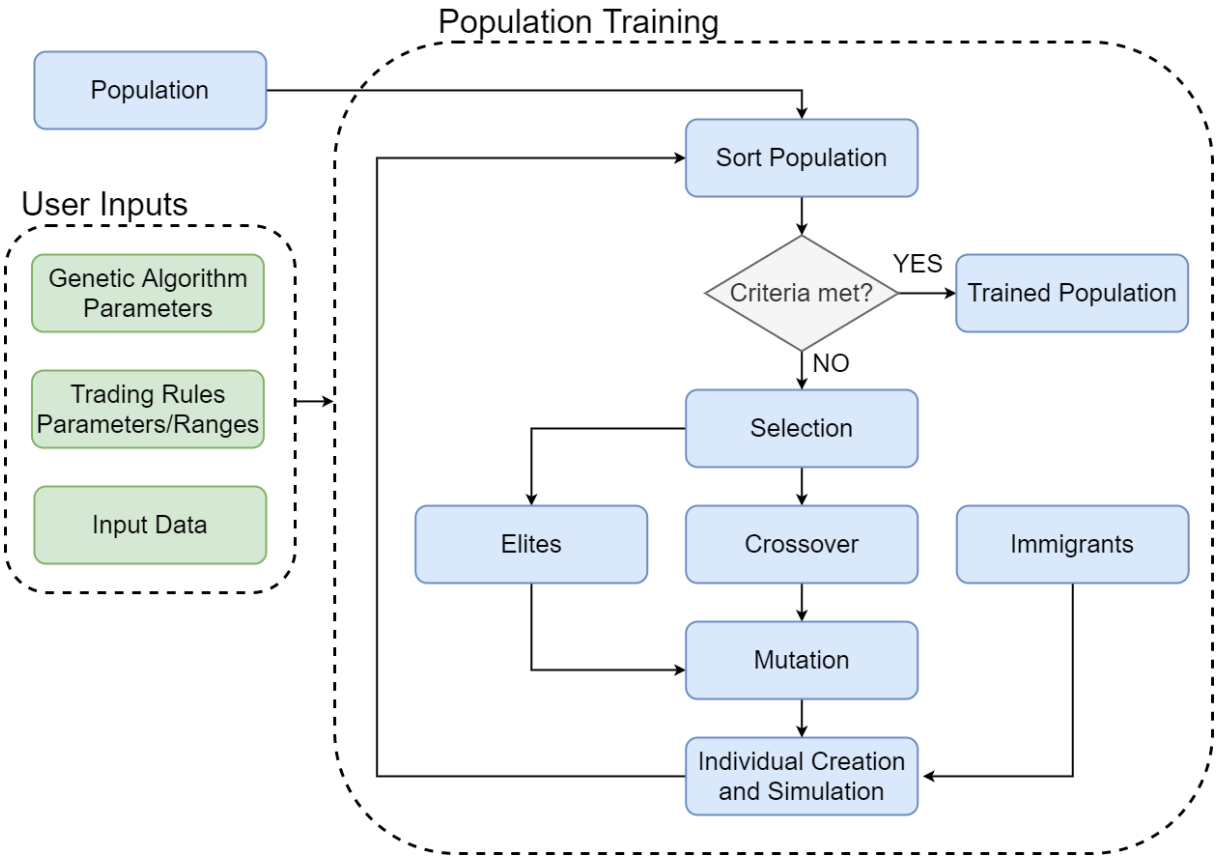


Figure 11: Representation of the Population Training.

3.3.2.1 Sort Population and Selection

As stated in section 2.4.1, the first step in a traditional GA is to evaluate every individuals' fitness, and rank them. In the case of the implemented architecture, this method is replaced by *Sort Population*, which as its name suggests, sorts the population created previously by the fitness values of its individuals (computed previously in the individual simulation). As input this process receives a list of individuals, which in the first iteration comes directly from the *Population Creation* process, as opposite to the following iterations, in which this list of individuals is created from the chromosomes that result from the GA operators.

After sorting the population, a *Selection* process takes place, in which the chromosomes of the top individuals, are extracted and two lists are made with them, as Figure 12 demonstrates. The first, the *elite* list, is composed by the chromosomes from the individuals with the best fitness values, in this case the top 1-10% individuals.

The second list is the *crossover* list, which is composed by the top 50-60% individuals from the population. This list also contains the elites from the first list, and is used to perform the crossover operation described in the next section.

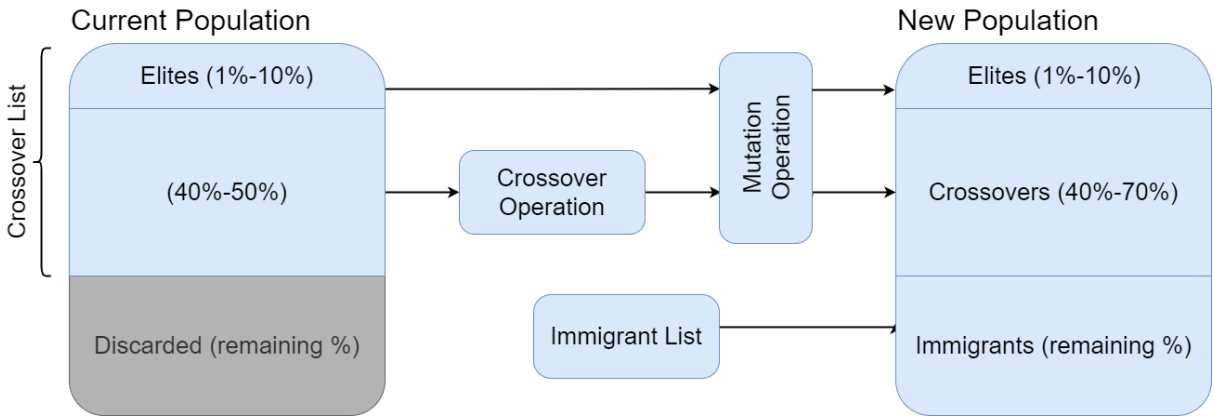


Figure 12: Representation of the selection process.

The final list of chromosomes resulting from the GA operators, is composed by the elites' chromosomes, the chromosomes generated from the crossover operation and immigrant chromosomes generated using the process described in section 3.3.1.3, following the proportions shown in Figure 12.

3.3.2.2 Crossover

The implemented crossover process consists in a standard two-cut crossover, with some extraordinary exceptions, when crossover is performed using the one-cut method. Both of these processes applied to the current architecture are represented in Figures 13 and 14, and were explained previously in section 2.4.1, which is the reason why their inner workings won't be further explored in this section, however, the procedure that selects which method is used deserves some insight

The first step to this process is to select two random chromosomes from the *crossover* list previously created (previous section). In the unlikely case that the process selects the same chromosome twice, the offspring is a copy of the chromosome, becoming a rare case of elitism.

Nevertheless, the most likely case is that the selected chromosomes are from distinct individuals, in which case the implemented architecture follows the procedure represented in algorithm 5. From the logic present in this algorithm, one can perceive that the first step is to generate two random numbers between 1 and the number of gaps between the chromosome (number of rules minus one). The next step depends of the values generated, if both are values are equal, the one-cut method is employed, whereas if they are different the implementation performs the two-cut method.

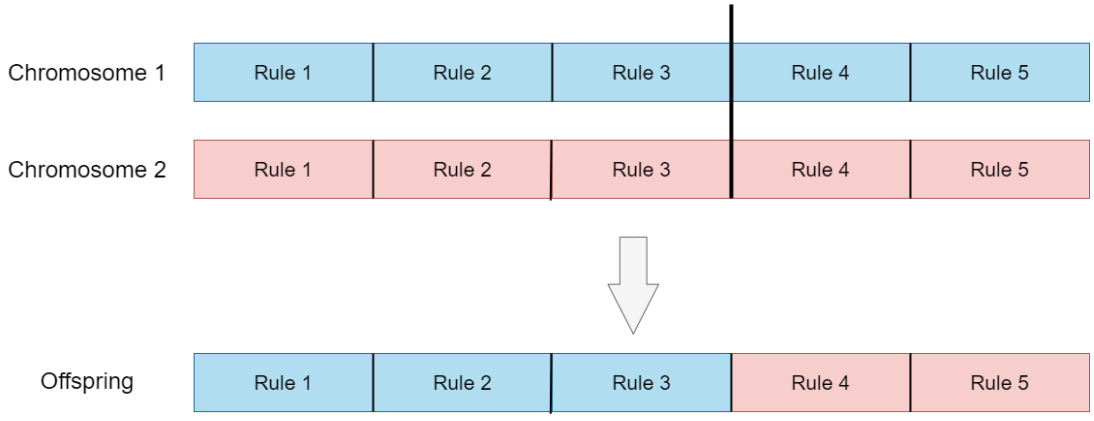


Figure 13: Representation of the one-cut crossover, in the context of the implementation.

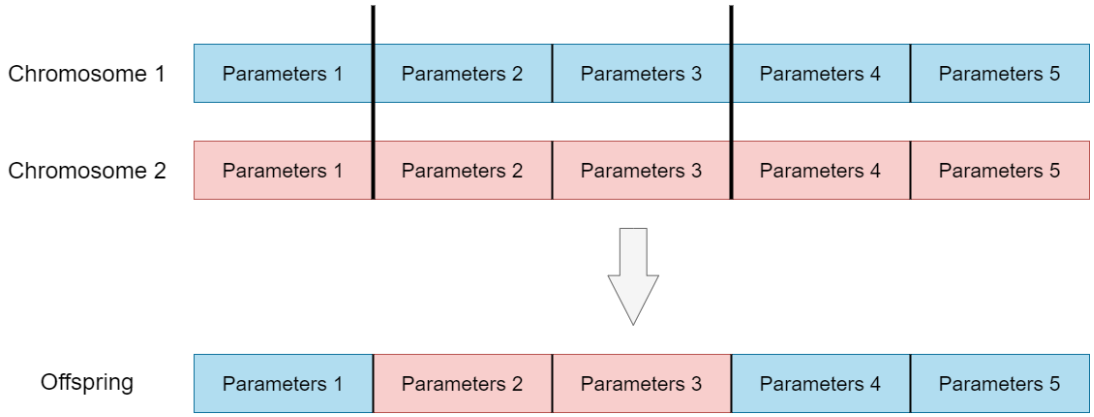


Figure 14: Representation of the two-cut crossover, in the context of the implementation.

A final remark to make about the crossover process, is that each crossover operation generates a single offspring, a choice in design that was made to improve the diversity of the population in the next generation.

Algorithm 5: Pseudo-code of the selection process between the crossover methods

```
max_rand = length(parent_1) - 1
cut_1 = random(1, max_rand)
cut_2 = random(1, max_rand)
if cut_1 == cut_2 then
    offspring = one_cut(parent_1, parent_2, cut_1)
else
    offspring = two_cut(parent_1, parent_2, cut_1, cut_2)
```

3.3.2.3 Mutation

The final GA operator implemented was the *mutation* process. This process receives as input a list including the chromosomes of the selected elites and the offspring created through crossover, and the ranges of the parameter necessary to compute the selected trading rules (Table 4). These ranges are necessary to the mutation process, since the mutated alleles still need to follow the guidelines stated before. Additionally, the mutation process requires as input a value for mutation rate, which corresponds to the probability of a mutation happening.

The mutation process implemented in the scope of the project, in case of occurrence, changes the parameter in which it occurs by a single value, as demonstrated in algorithm 6. This occurs with every parameter except the weight parameter, since all the parameters are integers, while the weight is a float variable. To accommodate such particularity, the weight mutation process is computed replacing the 1 by 0.05, and the *range_max* and *range_min* by 1 and 0, respectively.

To finalize the section on Population Training, it should be highlighted that the list of chromosomes that the mutation process outputs is then appended to the list of immigrants, which are, as stated previously, chromosomes generated from scratch using the method described in section 3.3.1.3. This list now contains the chromosomes of the next generation. The next step, as highlighted by Figure 11, is to create and simulate individuals using and to sort the population again.

Algorithm 6: Pseudo-code detailing the logic of the mutation process.

```
for parameter, parameter_range in chromosome, range_list do
    probability = random_probability()
    if probability < mutation_rate then
        probability = random_probability()
        if probability > 50 then
            parameter += 1
            if parameter > range.max then
                parameter = range.max
        else
            parameter -= 1
            if parameter > range.min then
                parameter = range.min
```

3.3.3 Population Testing

In this section is present the description of the testing methods used to obtain the testing trading strategies, which are the strategies executed in test data, producing the results present in the next chapter.

3.3.3.1 Static GA vs Dynamic GA

One of the major objectives of this work was to reproduce a Genetic Algorithm that optimizes the solution to the problem at hand dynamically. As such, a solution was envisioned in which a population would be traditionally trained with a given amount of training data, in which the diversity mechanisms would be tuned to generate the most diverse population. The model would subsequently retrain this population using a much smaller sample size and number of generations, narrowing the diversity of the original population, and use the best of the resulting individuals to predict the next few periods. The originally trained population would then be used in the retraining process several times before a reset happens and a new population is created. The implementation of this process, dubbed Narrow Search method, is further explained in its own section (3.3.3.3).

In Figure 15 a simplified representation of the differences between the static solution (right), and the dynamic solution (left), both of which were implemented. As it can be observed the static solution consists of the best individual of the traditional GA being simply tested, however the dynamic uses the full traditionally trained population in the Narrow Search method.

3.3.3.2 Simple Testing

The first method to obtain a test trading strategy is to select the chromosome of the fittest individual of a trained population, and to create and simulate an individual with the same chromosome in data from a test period.

This test was implemented using several mechanisms used to create the population, such as the creation of a data frame with the necessary test data (section 3.2.1), and individual creation and simulation (sections 3.3.1.4 and 3.3.1.5).

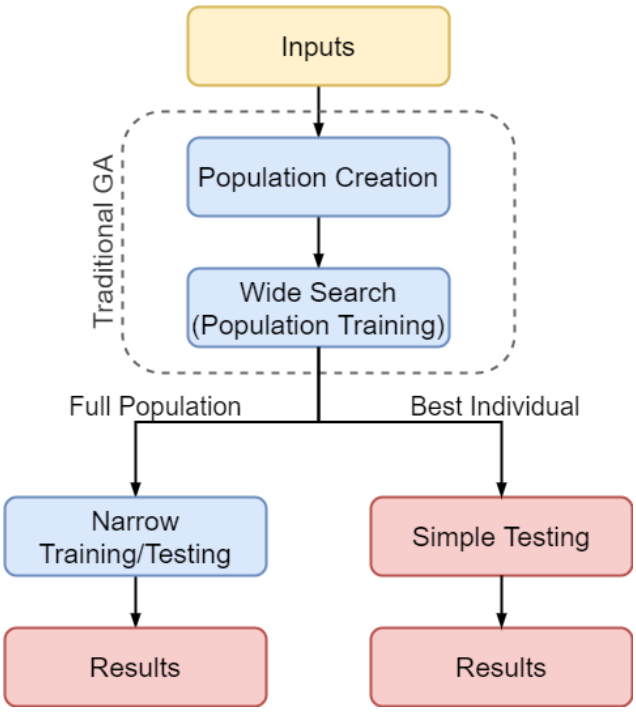


Figure 15: Simple representation between Simple Test (right) and Narrow Search (left).

3.3.3.3 Narrow Search

The second method is more complex, since there several steps before arriving at a final trading strategy. A simplified representation of the method is presented in Figure 16. In contrast with the first method of testing, it is observable that the second one has more inputs, adding every individual of the originally trained population, a series of training periods and a set of Genetic Algorithm parameters, to the input list of the previous method.

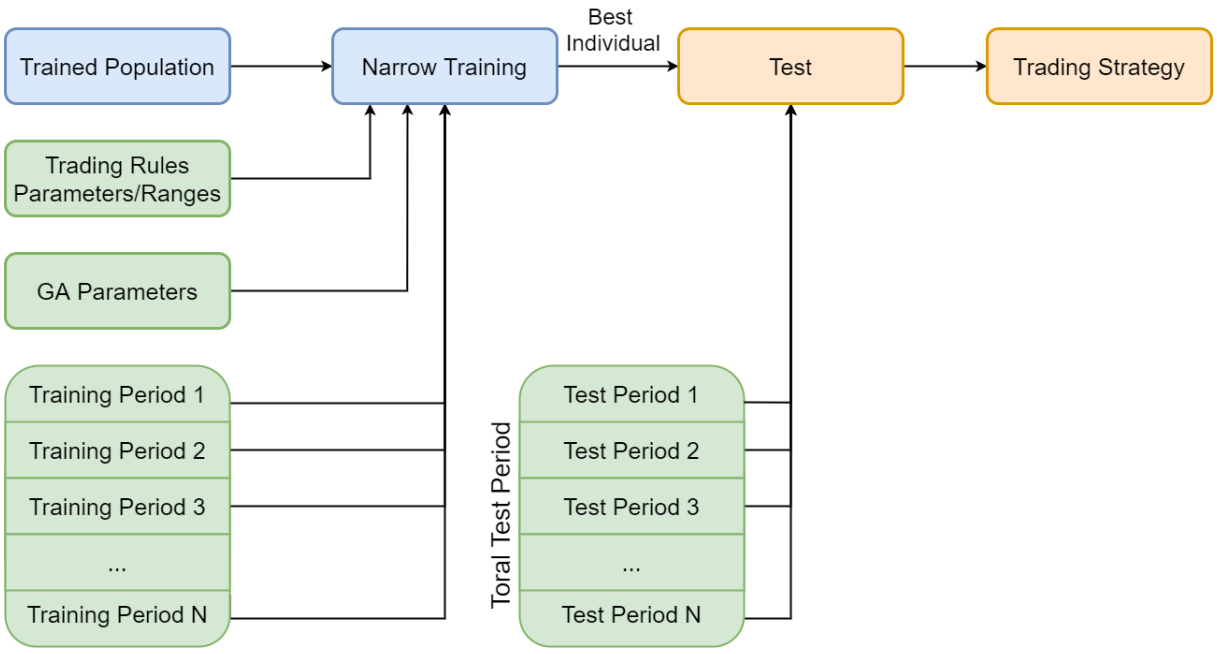


Figure 16: Representation of the Narrow Search method.

The first step of the implemented procedure is to segment the total testing period into several shorter

periods (hours/days), and to associate each of them to a shorter training period (weeks/months) that directly precedes them. An example of this would be to train the population in the period between 1/1/2015-28/2/2015 and generate a trading signal to the whole day of 1/3/2015.

After creating these time periods, a narrow training is performed with the previously trained population using one of the training periods. It should be noted that, from this point forward the first training is going to be called *wide* training and the training which occurs in this method of testing will be called *narrow* training, since the two types of training aim to achieve different things. The wide training aims to create the most diverse population, having a high number of generations and high GA parameters that increase the population diversity. Conversely, the narrow training aims to decrease the population's diversity, with fewer generations and low GA parameters. However, despite their different goals, both of the training methods share the implementation described previously in section 3.3.2.

The final step of this testing method is to use the best individual of the population that results from the narrow training and generate a trading signal for the testing period associated with the reduced training period. The whole process is repeated one time for each of the testing periods, resulting in a series of trading signals, that are consequently appended and simulated as a whole.

A noteworthy observation is the fact that the population used as input in every narrow training is always the one that results from the wide training.

3.3.4 Overall Algorithm Parameters

To close out this section, a summary of the implemented algorithm's input parameters is presented, in Table 5. Each of the parameters present in this table are stated before each case study, with many of them being constant throughout the next chapter.

Table 5: Summary of the algorithm parameters.

Data Input	**Wide Training**	**Narrow Training**	**Simulation**
Training Data	Population Size	Generations	Leverage
Testing Data	Generations	Elites	Investment
Trading Rule List	Elites	Crossover List	Initial Money
	Crossover List	Crossover Generated	Stop Gain
	Crossover Generated	Mutation Rate	Stop Loss
	Mutation Rate	Training Period Size	Trading Signal Method
		Testing Period Size	Double Down

3.4 Fuzzy Implementation

One of the points of interest in this work is the comparison between a crisp logic model and a fuzzy logic model, the latter of which being the theme of the current section.

Firstly, it should be emphasised that both of the implemented models are almost identical. The only difference between both models occurs during the computation of the trading signals of each selected trading rule in the *Individual Creation* (3.3.1.4), in which the crisp model generates a ternary signal (either -1, 0 or 1), while the fuzzy implementation generates a signal with float values between -1 and 1. Figure 17 highlights in red the difference between creating a population using the crisp model (present in Figure 8), and the fuzzy implementation represented in this figure.

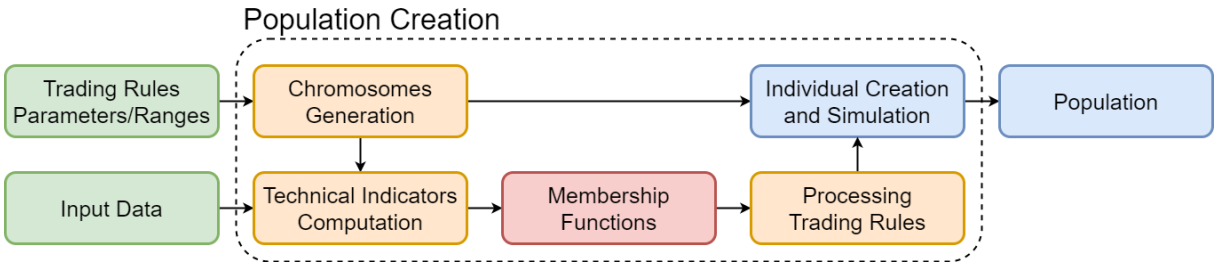


Figure 17: Fuzzy implementation of the process of creating a population.

It is evident that this implementation relies heavily in custom made membership functions, which receive as input the value of a given indicator and generate as output a float signal value. After generating a signal for each of the selected trading rules the implementation follows the same process as in the crisp model, creating a general trading signal using the same method, represented by the pseudo-code in algorithm 2 or 3. It should also be stated that, the trading signals that are generated during the training and testing processes of the algorithm, are also processed through these membership functions.

The logic behind this approach is to generate weaker signals during periods in which the used indicator marginally fulfils the trading rule. On the other hand, signals close to 1 or -1 should be employed an indicator fulfils the trading rule by a considerable margin. Using this logic, it should be possible for the trading rules to assign how strong a trend is.

3.4.1 Membership Functions

In this section the main types of membership functions that were used to implement the trading rule's fuzzy logic approach will be analysed, while also highlighting any variables used.

To start, it should be stated that every membership function is composed by one or a combination of two *S-functions*, which were implemented using the python module *Scikit-Fuzzy* [35].

3.4.1.1 Membership Function 1

The first membership function presented in this section concerns the RSI, Bollinger, MFI and Stochastic trading rules. In the crisp version of these trading rules the variables *floor* and *ceiling*, are used as the limits to generate buy and sell signals, respectively.

As such, the fuzzy logic implementation uses the values of these variables as the transition points of the membership function, as represented in Figure 18, where point *a* is the *floor* variable, and point *b* is the *ceiling* variable. As it can be observed by the representation of this membership function, its structure is composed by the sum of two *S-functions*, with each being responsible by one transition.

It should be stated that, the fuzzy implementation of the Bollinger-MFI trading rule was achieved through two membership functions with this same layout, one for each of the indicators used (%b and MFI). However, the signal value generated by each of these functions ranges only between -0,5 and 0,5 , instead of the normal range ($[-1, 1]$). The values of these ranges are then added up, which results in the trading signal of this rule.

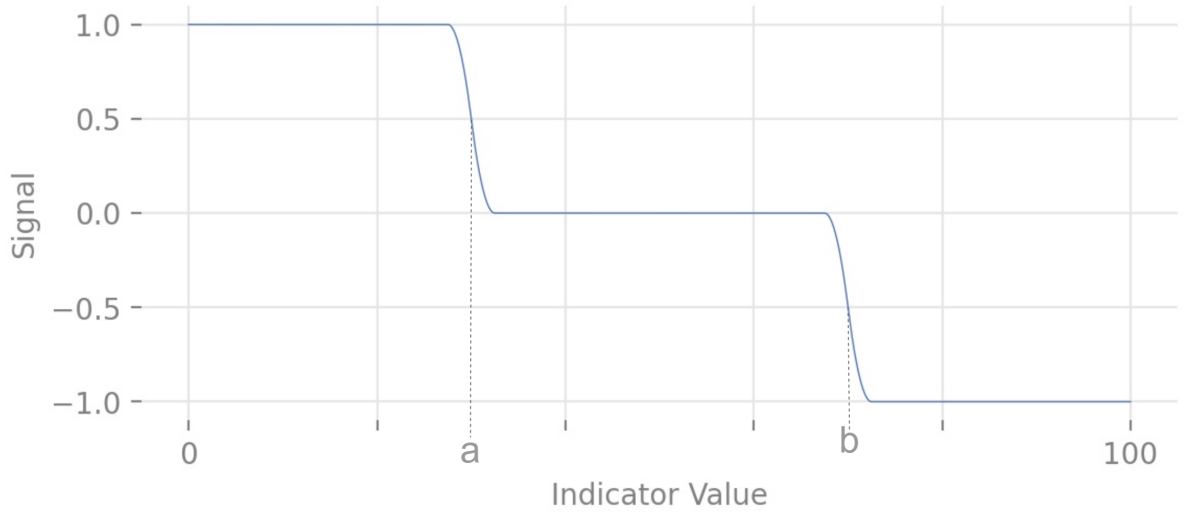


Figure 18: Example of a membership function of type 1.

3.4.1.2 Membership Function 2

The second membership function is used in the computation of the fuzzy implementation of the ROC, EMA, MACD and 3-EMA trading rules. The crisp version of these rules is characterized by resorting to a single indicator, which, if negative, produce a sell signal, and, if positive, a buy signal.

As such, the obvious point to start the transitions of the membership function would the origin point. It is also clear that, in this membership function, if the value of the indicator progresses through the negative values, the corresponding signal values will decrease until they reach -1, while, if the indicator's values progress through positive values, the signal value will increase, reaching the maximum 1. This behaviour is represented in the representation of this membership function in Figure 19, which as the previous membership function is by the sum of two *S-functions*.

Despite the fact that all the aforementioned trading rules use the layout of this membership function, there is some discrepancies between them. For example the EMA and the 3-EMA use the custom made indicators present in equations 29 and 30 in section 3.3.1.1, which return positive and negative percentages, while the ROC and MACD rules utilise indicators that often return infinitesimal values. Since these trading rules do not have *ceiling* and *floor* variables, a guideline for the transitions had to be created. To accommodate the different values that each indicator may return each trading rule's membership function was created using the values provided in Table 6.

Table 6: Behaviour of each trading rule to the membership function 2.

Trading Rule	Indicator Value	Signal value
ROC	> 0.5	1
	< -0.5	-1
EMA	> 3.5	1
	< -3.5	-1
MACD	> 0.2	1
	< -0.2	-1
3-EMA	> 3.5	1
	< -3.5	-1

A final remark to make is that, while the fuzzy implementation of the 3-EMA trading rule utilises the membership function to produce values for the trading signal based in the 3_EMA_{rule}, the remaining logic present in algorithm 1 is also followed. As such, trading signals different than 0 are only produced when the conditions present in this algorithm are fulfilled.

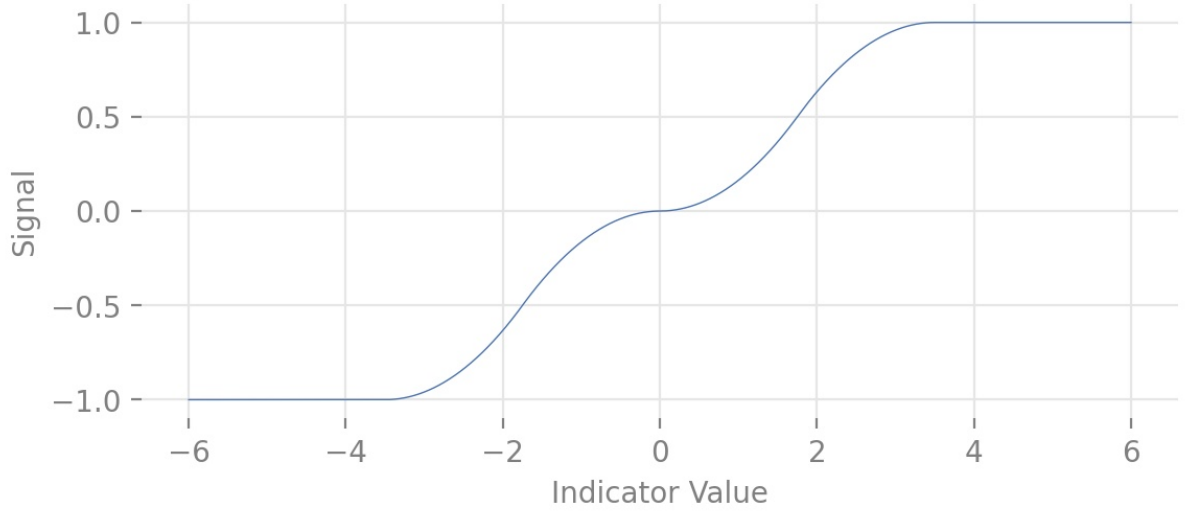


Figure 19: Example of a membership function of type 2.

3.4.1.3 Membership function 3

The third and final membership function is only employed by the fuzzy version of the ADX trading rule. This membership function utilizes a single *S-function* in its structure and utilizes the variable *a* to produce the only transition, as represented by Figure 20. This variable corresponds to the *ceiling* variable employed by the crisp approach of the ADX rule.

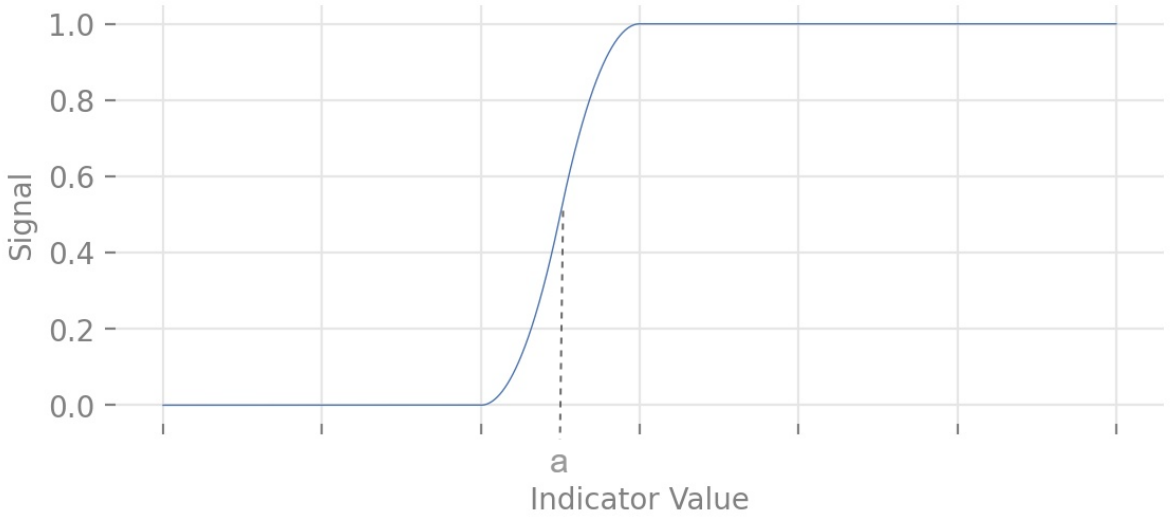


Figure 20: Example of a membership function of type 3.

In summary, the fuzzy implementation of the ADX trading rule uses this membership function to produce a trading signal between 0 and 1 based on the value of its main technical indicator the Average Directional Movement Index. However, by using its other technical indicator (the SMA_{rule}), the values of

the trading signal are changed to a sell or buy position, following the logic described in section 3.3.1.1. For instance, if the membership produces a signal of 0,2 and the SMA_{rule} is negative, the final trading signal will be -0,2.

3.5 Chapter Conclusions

This section describes summarily the work done in this chapter. The overall architecture was thoroughly described, dividing the different layers into functional modules, in order to make it easier to develop and implement the solutions presented. This chapter also provided the frameworks and python libraries that were used.

Chapter 4

System Validation

During this chapter different evaluation strategies and metrics are explored. The different case studies are also described, while highlighting each of the different approaches used, as well as the experiments regarding the investment rules. The algorithm is put against some proposed benchmarks (Buy and Hold, Sell and Hold and Random Walk) and against a Static GA using the most common parameters seen in the literature.

4.1 Evaluation Metrics

In the following section the evaluation metrics used to assess the quality and effectiveness of each strategy, are presented. These metrics consist in the *Accuracy*, the *Drawdown* and *Return on Investment*, which have been proved very useful to evaluate investment strategies throughout the literature.

4.1.1 Return on Investment

The Return on Investment is one of the most common metrics to evaluate trading strategies, which is noticeable by its use in several researches in financial markets. One of the reasons for its high usage is its intuitiveness, since it can be defined as the mean profit that results from an investment strategy, as evidenced by equation 42.

$$ROI = \frac{Returns - Initial\ Investment}{Initial\ Investment} \tag{42}$$

To better understand the use of this metric two simple examples are presented in Table 7, in which one can observe and compare the results of two trading strategies.

Table 7: Two examples of the Return on Investment.

	FX Market	**Initial Investment**	**Return**	**ROI [%]**
Trading Strategy 1	EUR/USD	100000 €	150000 €	50
Trading Strategy 2	EUR/USD	100000 €	80000 €	-20

However, it should be stated that the ROI does not take into account exposure risks or accuracy, representing only the monetary gains of a trading model. For this reason one or more metrics should be used to compare trading strategies, as stated during the implementation of a fitness score previously in section 3.3.1.6.

4.1.2 Accuracy

Previously, during the implementation of the fitness score (, in section 3.3.1.6), the accuracy of the trading positions was used in combination with the ROI, to assess the efficiency of a trading strategy. As such, along with the ROI, the position accuracy was chosen as a metric to evaluate and compare investment strategies during this section.

As stated previously, the position accuracy measures the percentage of profitable positions, following equation 43, which is a simplified version of equation 39.

$$Position\ Accuracy = \frac{Profitable\ Positions}{Profitable\ Positions + Unprofitable\ Positions} \tag{43}$$

To better help understand the functioning of the position accuracy metric a simple example is provided in Table 8.

Table 8: Example of the metric position accuracy.

Profitable		**Unprofitable**		**Accuracy [%]**
Buy Positions	Sell Positions	Buy Positions	Sell Positions	
50	30	40	10	61.54

In the context of this framework, results that obtain an average accuracy of around 55% will be considered the standard, while results with 57% or more will be considered remarkable. This criteria stands on the fact that results that present high accuracy will not always be the more profitable, but rather the ones that obtained a higher ratio of profitable positions. As such, it should be stated that a good average accuracy is a necessary, but not sufficient , criteria in the analysis of the results.

4.1.3 Drawdown

The final metric employed to compare the results of different investment strategies was the Drawdown. This metric is commonly used to assess the risk associated with the applied investment strategy. The computation of this metric can be achieved by calculating the difference between the highest local maximum and the succeeding lowest local minimum. A graphical aid is provided in Figure 21 to better understand the this non-trivial computation.

Finally, is should be stated that a good trade-off between all the three metrics would be ideal, more emphasis will be given to the ROI since its most common amongst the literature.

4.2 Case Studies

This section describes the results of several case studies, in which a variety of changes and configurations analysed and compared, as well as the investment strategies used.

4.2.1 Methodologies

The implemented system's validation can be described by four different case studies. However in each of the case studies some inputs and configurations will be kept constant.

Such is the case of the input data was used. As stated in section 3.2.1, the data utilized to train and test the trading system throughout the validation process, consists in Foreign Exchange's historic data feed of the EUR/USD currency pair. From the total data that spans from the year 2014 until 2020,

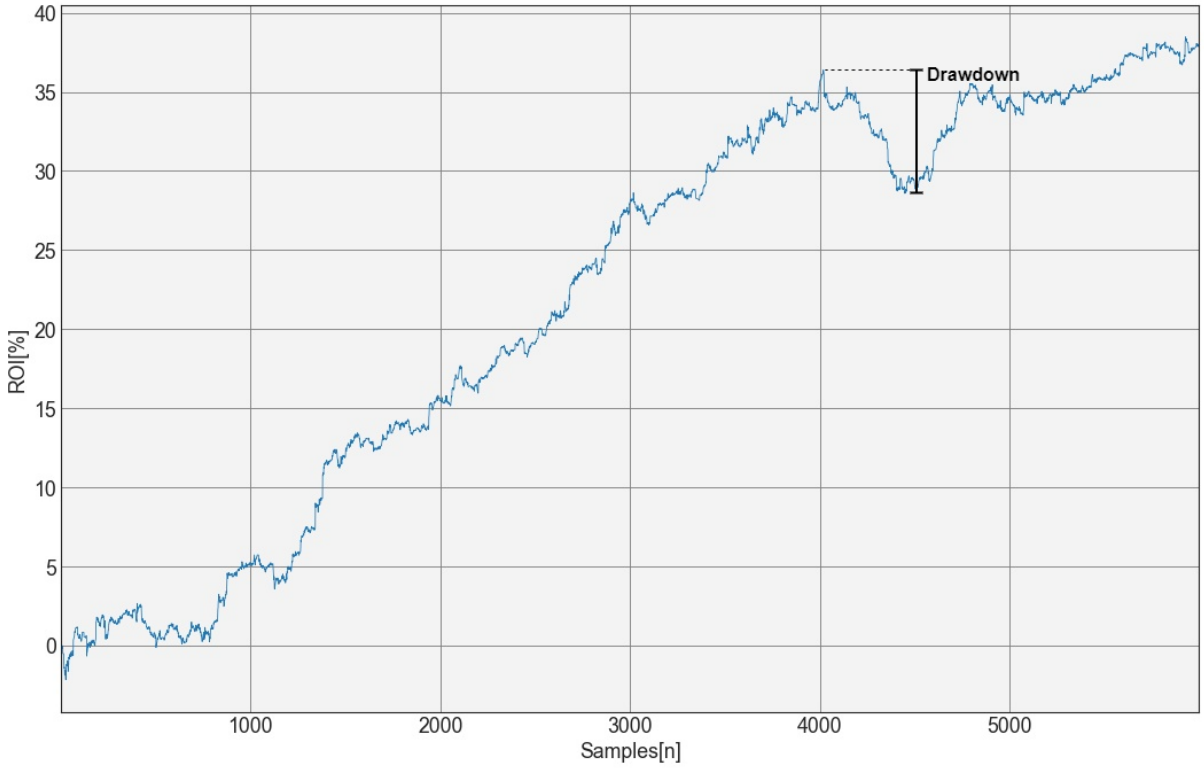


Figure 21: Example of the drawdown of standard ROI evolution graph.

three periods were selected, and segmented into training and testing data. As evidenced by Table 9, the three testing periods are consecutive and not overlapped, providing three time periods with different characteristics.

Table 9: Training and testing data time periods.

	Training Period		**Testing Period**	
	Begin	End	Begin	End
1^{st} Period	1/7/2014	31/12/2018	1/1/2019	30/6/2019
2^{st} Period	1/1/2015	30/6/2019	1/7/2019	31/12/2019
3^{st} Period	1/7/2015	31/12/2019	1/1/2020	30/6/2020

In addition to the input data, each of the case studies also uses the same input parameters used to configure the Wide training component of the Genetic Algorithm, described in section 3.3.2. As stated previously this training component precedes the testing process, which may follow the guidelines present in section 3.3.3.2 to generate the trading strategy of a static GA, used as a comparison object throughout the Validation section. It may also follow the guidelines of the Narrow Search method described in section 3.3.3.3, to produce a trading strategy using a Dynamic approach.

In Table 10 each of training parameters are presented. These parameters were selected through the analysis of the experiments described in the literature, that had been performed in this same market with a comparable framework.

Table 10: Configuration of the parameters of the Wide Search (GA training).

Parameters	Value
Population Size	100
Generations	100
Elites	4
Crossover List	50
Crossover Generated	50
Mutation Rate	10

The input parameters utilized to simulate trading strategies (as explained in section 3.3.1.5) were also invariable throughout the Validation process (except in Case Study C), both during the training are the testing of the algorithm. These parameters, which are present in Table 11, provide a constant strategical behaviour throughout the executions, which allows comparisons between the machine learning methods to be made, without the trading strategy being a factor.

Table 11: Default simulation parameters.

Parameters	Values
Initial Investment	100 000€
Default Position	5000€
Leverage	20
Stop Gain Factor	5
Stop Loss Factor	0.2
Trading Signal	Quinary (Method 2)
Double Down	False

To finalize, it should be pointed out that, throughout the analysis of the case studies, several benchmarks will be present, namely the B&H and S&H strategies, and, as stated above, a trading strategy generated by a static GA.

4.2.2 Case Study A

The first study case presented aims to evaluate the results of the Narrow Search method, and to observe how this method fares against the selected benchmark strategies. To achieve that, the case study focuses in utilizing different configurations for the input parameters of this method, in order to maximize the performance of this method.

The baseline configuration for this method is presented in Table 12. It is possible to observe that each of the parameter values of every population diversity setting is tuned down, when compared with the configuration of the Wide Search presented previously. Evidence of this is the low mutation rate, and the high number of elite individuals and individuals generated by crossover. The number of generations is also tuned down. The objective of this tuning is to not overtrain the population during the testing phase of the algorithm.

Several variations were selected to be offered as subjects of comparison to the baseline configuration. Among these configurations, there are variations selected to study the implemented method's behaviour when it processes different sample sizes to train and test samples. These variations were obtained by doubling/halving the training period size and doubling the testing period size, and both at the same time. Other variations, such as increasing the number of generations, increasing the mutation

Table 12: Parameters of the baseline configuration.

Parameters	Value
Generations	2
Elites	15
Crossover List	50
Crossover Generated	60
Mutation Rate	2%
Training Period Size	2 Months (2*24*30 samples)
Testing Period Size	6 Samples

rate and reducing the number of elites, were selected to study the effects of increasing the diversity factor in the testing method. The full set of configurations is presented in Table 13, which also evidences which parameters are altered.

Table 13: Variations of the baseline configuration tested in the Narrow Search method.

Configuration Name	Changed Parameters	New Values
1/2*TS	Train Period Size	1 Month (30*24 Samples)
3TS	Train Period Size	3 Months (90*24 Samples)
2*ts	Test Period Size	12 Samples
1/2*TS-2*ts	Train Period Size Test Period Size	1 Month (30*24 Samples) 12 Samples
3TS-2*ts	Train Period Size Test Period Size	3 Months (60*24 Samples) 12 Samples
5Gen	Generations	5
10Mut	Mutation Rate	10 %
30Elites	Elites	30
5Elites	Elites	5

It should also be mentioned that during this case study the only trading rules employed are presented in Table 14. These trading rules were selected due to being used with success in literature with comparable frameworks, such as Hirabayashi et al. [5], Almeida et al. [6] and Gorgulho et al. [4].

Table 14: Trading rules and ranges used in case study A.

Trading Rule	Variables	Value Range
RSI	N	[5, 30]
	$ceiling$	[65, 75]
	$floor$	[5, 30]
ROC	N	[15, 120]
EMA	N	[10, 150]
MACD	N_{signal}	[5, 15]
	N_{short}	[10, 25]
	N_{long}	[20, 40]

4.2.2.1 Parameter Evaluation

To evaluate the different configurations in this case study, the Proposed Dynamic model was executed 10 times in each of the periods presented in Table 9, which resulted in 30 executions per configuration.

Figure 22 shows an histogram of the ROI obtained through the executions of algorithm using the configurations generated using variations for the number of training and testing samples employed in the Narrow Search method, as well as the *Base* configuration.

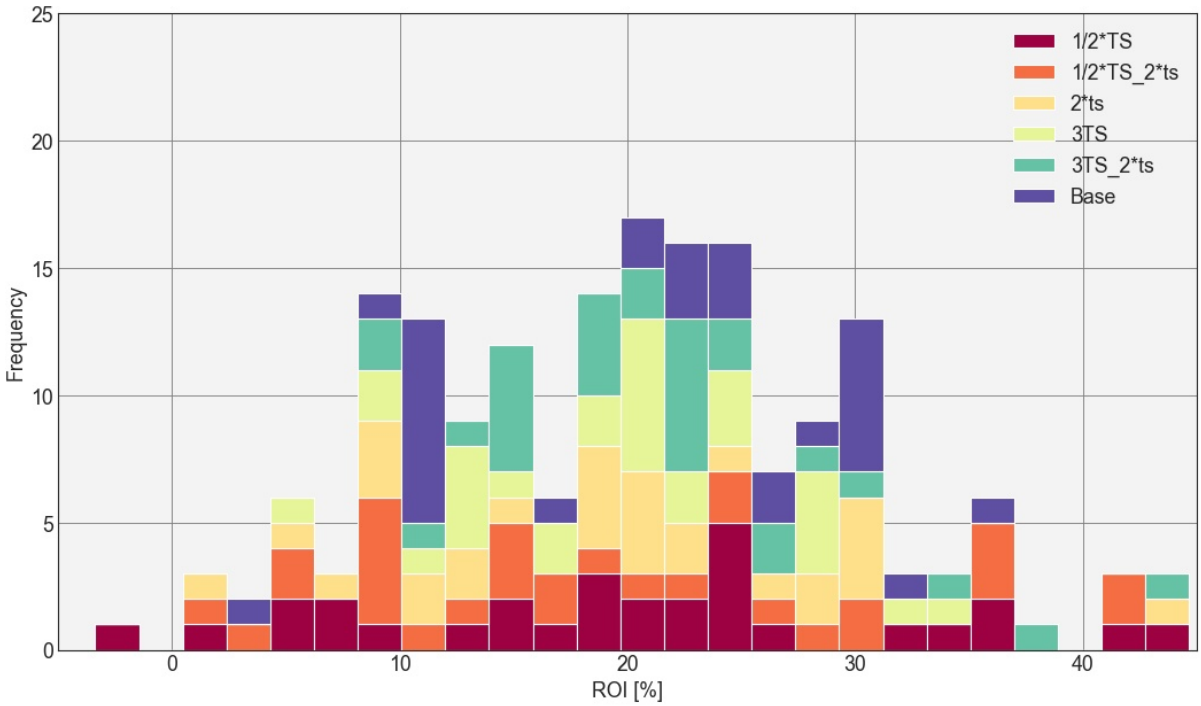


Figure 22: Histogram of the configurations in which the number of samples is altered.

From the results obtained by varying the sample sizes, it is possible to conclude that the configuration *3TS-2*ts* achieved an average ROI value higher than the other configurations in the 15 runs executed. This configuration scored 21.24%, against the 20.14%, 19.99%, 19.42% and 19.30%, of the *½*TS*, *3TS*, *2*ts* and *½*TS_2*ts* configurations, respectively. The *Base* configuration also performed worse than the *3TS-2*ts* configuration, having obtained an average ROI of 20.96%.

Since the final objective of an algorithm such as this would be to invest in a market in real time, the algorithm's total computation time is a also very important constraint. Hence, although the *3TS-2*ts* configuration presents marginally better results, it should be highlighted that its computation time is one of the highest among the tested configurations. Through the analysis of the structure of the testing method previously explained in section 3.3.3.3, it is expectable that the computation time would be proportional to the training sample size, since the quantity of technical indicators to be computed increases with the size of the training period. It is also expectable that the computation time would be inversely proportional to the test sample size, since higher test sample sizes mean less training processes and, consequently, less computation time.

As such, the configuration that presents the shorter running time is the configuration with the lowest training sample size and highest testing sample size, which in the case of the study is the *½*TS_2*ts* configuration. These expectations became reality with the tests of this configuration obtaining the lowest computation time. The aforementioned changes to the periods sample sizes may cause the computation time of the Narrow Search method to increase by 30% to 80%.

Although the *½*TS_2*ts* configuration presents the fastest time between these configurations, it is also one of the configurations with the worst ROI results, even if it is just marginally. Therefore it should be stated the user should strive for a configuration that presents a good trade of between run time and results, such as the base or the *½*TS* configurations.

The histogram displayed in Figure 23 presents the ROI achieved by the configurations in which the diversity parameters, such as the number of generations and elites used, as well as the mutation rate.

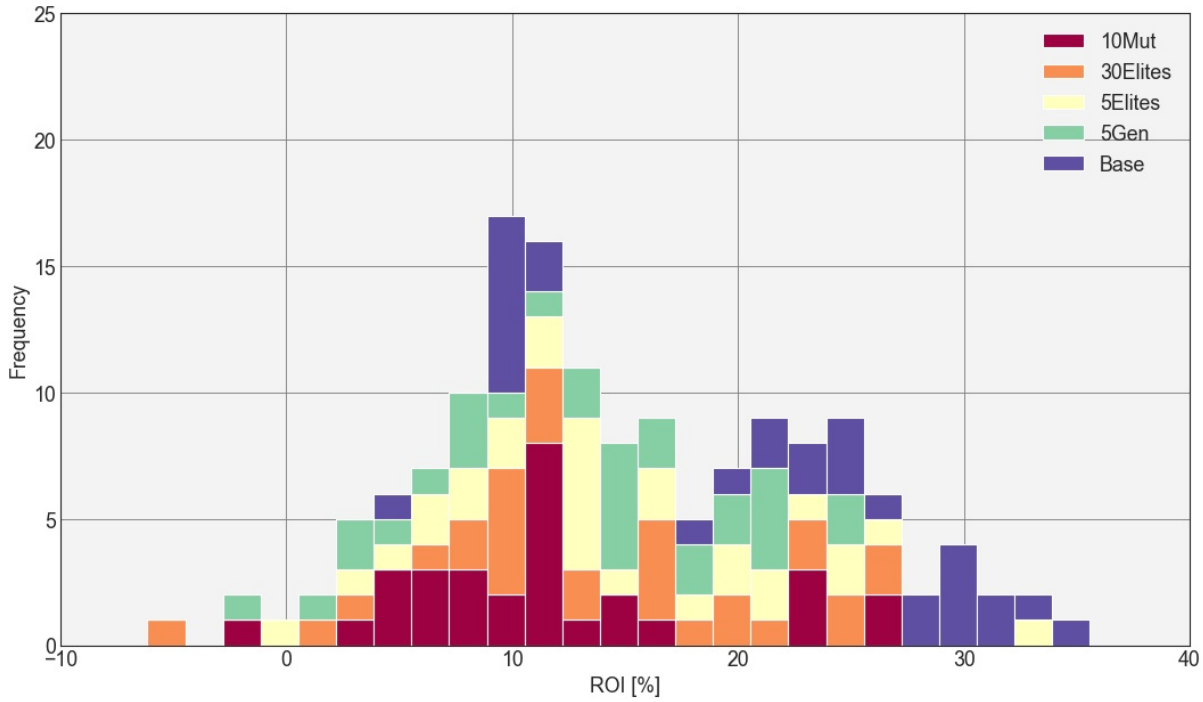


Figure 23: Histogram of the configurations in which the diversity parameters are altered.

The results obtained by using the configurations in which the diversity parameters are altered, offer several conclusions. First of all, it is possible to observe that every configuration in which the diversity parameters were modified attained worse results than the baseline configuration. The *5Gen*, *10Mut*, *30Elites* and *5Elites* obtained as an average ROI, respectively, 13.50%, 11.50%, 14.25% and 14.46%, against the 20.96% obtained by the base configuration.

It is possible that the reason for the bad performance of the *5Gen* configuration is due to the over-training of the population to the training data during the Narrow Search method.

On the other hand it is evident that increasing the mutation rate to 10% or decreasing the number of elite individuals (which increases the number of random immigrants) increases the diversity of the population so much that, some main features of the original population (wide search) seem to be discarded. The *30Elites* configuration's reason for the subpar results is the opposite, since more elite individuals means less random immigrants, and consequently, and less diversity. Since the diversity of this method is reduced so much, the result seems to be a population that maintains all the features of the wide search and doesn't obtain new ones during the Narrow Search method.

It should also be stated that, since the *5Gen* configuration utilizes 5 as the number of generations in each iteration of the Narrow Search method, its total execution time is more than double that of the *Base* configuration, which utilizes 2 as the generations parameter.

In Table 25, present in the Appendix A, the periodic results of each of the tested configurations are displayed. From the average ROI values of each period, we can conclude that the Narrow Search method performed the worst in the second period (1/7/2019-31/12/2019), in which none of the configurations reached values higher than 15%. This value is surpassed by almost every configuration in the both of the other periods (1/1/2019-30/6/2019 and 1/1/2020-30/6/2019). A possible explanation for this phenomenon would be that the used trading rules or input parameters are not ideal for the second period. It is also possible that this period was less eventful than the other two, having longer sideways

periods.

It should also be highlighted that, the periodic results obtained for the third testing period present a considerable drop off in the accuracy values, throughout every configuration. These results are particularly bad for every configuration in which the diversity parameters were altered, since each obtained an average accuracy of around 50%.

The overall results of this case study are displayed in Table 15. From the results present in this table, it is possible to conclude that the *3TS-2*ts* configuration presents better results in each of the evaluation metrics, presenting the best average ROI, accuracy and drawdown, of every configuration for the Narrow Search method.

This table also displays the standard deviation value for the ROI values obtained using each of the tested configurations, which evaluates how spread out the ROI values are. As such, it can be concluded that, in addition to presenting the best average ROI, the *3TS-2*ts* configuration also presents one of the best standard deviation values, meaning the that ROI values obtained using this configuration are not a spread out as other configurations.

Table 15: Overall results of all configurations tested in Case Study A.

Metric	Base	1/2*TS	3TS	1/2*TS_2*ts	2*ts	3TS_2*ts	5Gen	5Mut	30Elites	5Elites	Static
Drawdown [%]	4.31	4.46	4.76	4.40	4.00	3.64	5.59	6.24	5.75	5.36	4.21
Accuracy [%]	57.23	56.42	57.80	57.05	58.60	59.45	54.26	53.46	54.24	54.44	63.39
Min ROI [%]	4.20	-3.37	5.63	1.09	1.79	9.15	-1.40	-2.62	-6.15	-0.17	-8.97
Max ROI [%]	35.58	43.72	34.60	42.08	44.66	43.84	24.34	26.50	26.37	32.82	8.73
Avg. ROI [%]	20.96	20.14	19.99	19.42	19.30	21.24	13.50	11.50	14.25	14.46	2.57
Std. Dev. [%]	8.86	11.53	7.04	11.37	9.37	7.87	6.96	6.98	7.77	7.43	4.05

It is also possible to conclude that every tested configuration, in which the number of samples of training and testing are modified, presents an average ROI of around 20%, and an accuracy of around 57%. However any configuration in which the diversity parameters were altered present an average ROI of around 14% and an accuracy of around 54%, which make them less viable for the Narrow Search method.

These results are a testament to the robustness and effectiveness of the Narrow Search method, which present a better ROI values than the Static GA, independently of the both of configuration used.

4.2.2.2 Conclusions of Case Study A

This section concludes Case Study A, from which it is possible to conclude that the configuration *3TS-2*ts* outperforms every other tested configuration, including the *Base* configuration, proving it to be a solid solution. However, it is important to highlight its long computation time, when compared with configurations such as *½*TS_2*ts*.

However, the main conclusion to take away is that the Narrow Search method presents better average ROI than the Static Genetic Algorithm, independently of the configuration used. It should also be stated that, if the only metric used was the position accuracy, the Static GA would be most viable solution, since it obtain the best average accuracy, with 63.39%.

As a final note, future users of similar algorithms should be aware that increasing the number of generations utilized in the Narrow Search method will proportionally increase the computation time. The same can be said of using large sample in the training component of this method, as well as using small sample sizes in its testing component. As such, even if configurations of this type produce better results, its application to real time trading should be analysed further.

4.2.3 Case Study B

The objective of this case study is to test the implemented algorithm with different sets of trading rules and, as in the previous case study, compare the results obtained using the Narrow Search method with the results from the Static GA. As such, every other test condition and parameter should be kept constant, so that the only changing variable would be the sets of trading rules.

To achieve that, firstly, it should be stated that the input parameters described during the Methodology section (4.2.1), such as the input data (Table 9), the training parameters (Table 10) and the simulation parameters (Table 11), were employed in every execution in this case study.

Additionally the Narrow Search method's parameters used in this case study were also kept constant throughout every execution of the algorithm, respecting the values displayed in Table 16. This configuration, along with others, was previously analysed and tested during Case Study A (section 4.2.2), achieving solid results and a reasonable computation time. For these reasons the *Base* configuration of the previous case study was selected.

Table 16: Parameters of the baseline configuration.

Parameters	**Value**
Generations	2
Elites	15
Crossover List	50
Crossover Generated	60
Mutation Rate	2%
Training Period Size	2 Months (2*24*30 samples)
Testing Period Size	6 Samples

To study the impact that different trading rules have in the algorithm' efficacy several sets of trading rules were selected, which are present in Table 17. For each set of trading rule two criteria were employed. Firstly, a the minimum of four trading rules in each set was selected as the norm. Secondly, there must be a balance between rules that use trend following indicators and rules that use momentum oscillator indicators.

Table 17: Sets of trading rules used during Case Study B.

Set 1	**Set 2**	**Set 3**	**Set 4**
RSI	3-EMA	RSI	RSI
ROC	Bollinger	Bollinger	ROC
EMA	MFI	ADX	EMA
MACD	MFI-Bollinger	Stochastic	MACD
			Bollinger
			Stochastic

As stated previously, several authors have already successfully implemented models by using the trading rules that make up the first set. Consequently, this set was previously employed in Case Study A, and represents a baseline for this case study. This set is composed by two momentum based rules such as the RSI and the ROC rules, and a trend following rule such as the EMA. Additionally, this set also contains the MACD trading rule, which is both a trend following and momentum-based rule.

The second set comprises the Bollinger, the MFI, the Bollinger-MFI and the 3-EMA trading rules. This set, as the previous one, presents a mix between a trend following trading rules, namely the Bollinger and the 3-EMA rules, and trading rules that try to predict momentum oscillations, such as the MFI. Since the Bollinger-MFI trading rule is a combination between the Bollinger and the MFI rules, it is both a trend following and momentum-based rule, similar to the MACD.

The remaining trading rules, that is, the ADX and the Stochastic rules, along with two rules featured in the first two sets, compose the third set. Since the ADX is a trend following rule and Stochastic is momentum based rule, the RSI and the Bollinger trading rules were selected to create a balanced set between a trend following and momentum oscillator rules.

A final set made up of more that four trading rules was selected in order to test how the implemented model handles a higher number of trading rules. However only the trading rules that present a simpler computation were selected, since the higher number of trading rules will most likely increase the model's overall computation time. As such the RSI, the ROC, the EMA, the MACD, the Bollinger and the the Stochastic rules were selected to compose the final set. It should be stated that this set comprises three rules that use trend-following indicators and two that use momentum oscillators and one rule that uses a mix of the two (MACD).

To conclude it must be mentioned that the predefined ranges of each of the trading rules employed during this case study are present in Table 4.

4.2.3.1 Trading Rule Sets Evaluation

In this section the performance of each trading rule set is analysed, while using both the Dynamic GA and the Static GA. To draw conclusions from this case study, the algorithm was executed using each of the rule set 10 times for each testing period, resulting in a total of 30 executions per set in each GA.

In Figure 24 a histogram presents the ROI obtained by each trading rule while utilising the Dynamic GA approach. From these results it is possible to state that the best training rule sets were sets 1 and 4, obtaining an average ROI of 20.96% and 20.91%, respectively, against the 17,04% and 13.36 % obtained by sets 2 and 3.

It is also a fact that both rule sets 2 and 3 possess trading rules that require multiple complex computations, such as the MFI or the ADX rules, which would predictably increase the model's total computation time. This event came to reality, with the model that utilizes these sets taking twice as long to finish than the models that used sets 1 and 4.

In Table 26, which is present in Appendix B, are displayed the periodic results of each of the tested rule sets. The obtained average ROI values point, once again, to the fact that the implemented Dynamic approach performs worse in the executions of the second testing periods present in Table 9 (1/7/2019-31/12/2019). Since the Dynamic approach presents a decline in the average ROI, independently of which rule set is used, it is possible to conclude that the trading rules utilised are not the cause for this decline. The instability of rule set 3 is also apparent from the results, since it was obtained an average ROI of 21.71% in the first testing period, and declined to values under 10% in both of the remaining periods.

As in case study A, it is possible to observe a considerable drop off in the Dynamic approach's average accuracy in the third testing period, across every trading rule set's results.

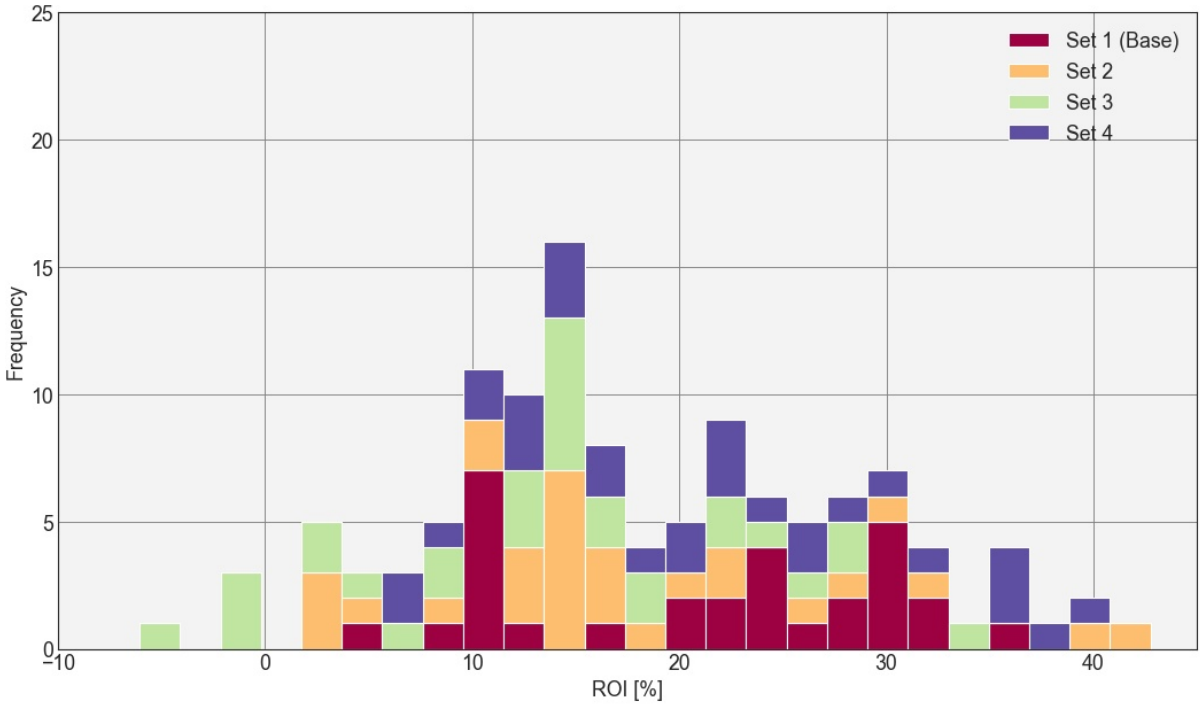


Figure 24: Histogram of the ROIs obtained by each trading rule set, using the Dynamic implementation of the GA.

Overall, it is possible to observe that the Dynamic implementation of the model presents better average ROI than the Static approach in every period, independently of which set of rules is used.

Table 18, which presents the overall results of both the Static and Dynamic approaches for all the trading rule set utilised. From the Dynamic approach results, the underperformance of rule set 3, when compared with the other rule sets, is evident, presenting the lowest average ROI, and one of the highest drawdown values.

Conversely, rule sets 1 and 4 present themselves as viable trading rule sets to use coupled with the Dynamic Approach, obtaining consistently profitable results throughout every period, with low drawdown values, comparatively. It should also be mentioned that each of the trading rule set present an above average position accuracy.

This table also corroborates the periodic results presented previously, with the Dynamic approach outperforming the Static GA in every metric, independently of the trading rule set used, except in accuracy.

4.2.3.2 Conclusions of Case Study B

Overall it can be concluded that the approach that obtained the best results in this case study was the Dynamic GA while utilizing rule set 1, which presents the best average ROI, with rule set 2 coming in close second. The metrics presented in Table 18 also confirm that, among the Dynamic approaches, this is the most consistent, presenting the lowest drawdown and standard deviation values, while also presenting a reliable accuracy value of 57.23%.

It should also be concluded that, once again the Dynamic approaches of the implementation presented more lucrative results than the Static counterparts. However it should also be observed that, if the result analysis was solely based on the accuracy values, once again the Static GA would outperform the Dynamic approach in every rule set.

Table 18: Overall results of all the sets tested in Case Study B.

Metric	Dynamic				Static GA			
	Set 1 (Base)	**Set 2**	**Set 3**	**Set 4**	**Set 1 (Base)**	**Set 2**	**Set 3**	**Set 4**
Drawdown [%]	4.31	4.38	4.44	4.09	4.21	4.75	3.54	3.41
Accuracy [%]	57.23	60.10	59.43	55.73	63.39	65.10	67.14	61.24
Min ROI [%]	4.20	1.99	-6.06	5.84	-8.97	-4.08	0.52	-7.38
Max ROI [%]	35.58	42.76	34.19	40.01	8.73	6.91	12.61	8.16
Avg. ROI [%]	20.96	17.04	13.36	20.91	2.57	1.27	4.57	2.04
Std. Dev. [%]	8.86	9.84	9.47	9.77	4.05	3.14	2.79	2.99

4.2.4 Case Study C

In this case study the methods used to generate trading signals, previously described in section 3.3.1.4 will be analysed and validated. The methods are explained by the pseudo-code present in algorithm 2 (method 1) and 3 (method 2), and their purpose is to normalise the original trading signal to a ternary and a quinary signal, respectively. This case study will also be put up to the test the Double Down investment strategy implemented as described in section 3.3.1.5.

Each of the methods were tested using input parameters already employed during the successful validation of previous models. A component of these inputs is the historic FOREX data of the time periods displayed in Table 9, which are utilised as the training and testing data. Along with the time periods, the Wide Search parameters presented in Table 10 were also utilised in every execution of each of the methods that are analysed in this case study.

Along with these, this case study the Narrow Search method also utilised the parameters of the baseline configuration of case study A (section 4.2.2) present in Table 16, which, as demonstrated previously, proved to be a reliable configuration for this method.

Identically, a set of trading rules was also selected to be used in the validation procedure in this case study. This set comprises the RSI, ROC, the EMA and the MACD trading rules, which, as stated previously, were successfully utilized by several authors in their frameworks. This set, which is present in Table 17 under the name Set 1 (base), was also the set that presented the best results in Case Study B (section 4.2.3), whose aim was to analyse the performance of several trading rules.

As stated previously, the subject of analysis in this case study is the investment strategy generation methods, which are reflected by the input simulation parameters of the model. As such, the first step was to select the parameters present in Table 19 as baseline for the other subjects. This set of parameters was selected as a baseline, due to the fact that they served as the default parameters throughout all the validation section of this work.

Table 19: Configuration 1 for the simulation parameters.

Parameters	Values
Initial Investment	100 000€
Default Position	5000€
Leverage	20
Stop Gain Factor	5
Stop Loss Factor	0.2
Trading Signal	Quinary (Method 2)
Double Down	False

To serve as objects of comparison, two different configurations for the simulation parameters were selected. The first replaces the method of normalising the trading signal to a simpler method, that generates a ternary signal instead of a quinary signal. The second configuration utilises this same method of normalising the trading signal, and adds the the Double Down strategy. Both of these configurations are displayed in Table 20.

Table 20: Configurations 2 and 3 for the simulation parameters.

Parmeters	Values	
	Configuration 2	Configuration 3
Initial Investment	100 000€	100 000€
Default Position	5000€	5000€
Leverage	20	20
Stop Gain Factor	5	5
Stop Loss Factor	0.2	0.2
Trading Signal	Ternary (Method 1)	Ternary (Method 1)
Double Down	False	True

4.2.4.1 Trading Signal Method Evaluation

In this section the performance of each method of generating a trading signal is analysed, while using both the Dynamic GA approach and the Static GA approach. In this case study each of the subject configurations were executed 10 times for each time period, resulting in a total of 30 executions per configuration in each GA.

In Figure 25 it is represented a histogram which displays the ROI valuess obtained by the Dynamic approach using each of the aforementioned configurations for the simulation parameters. From this results it is possible to conclude that configuration 1 presents the best results of all three configurations by quite some margin. This is attested by the difference between the values of the average ROI obtained, in which configuration 1 obtained 20.96%, against the 6.49% and 5.21%, obtained by configurations 2 and 3, respectively.

The periodic results present in Table 27, featured in Appendix C, also corroborate the ROI values present in the previous histogram, with the Dynamic approach that utilises configuration 1 obtaining the best average ROI in every period. From these results it possible to, once again, observer a decline of the Dynamic approach during the second testing period, independently of the configuration employed.

It is also noticeable that both configurations 2 and 3, coupled with the Dynamic approach obtain their best average ROI during the third testing period. However it is also be observed that the accuracy values

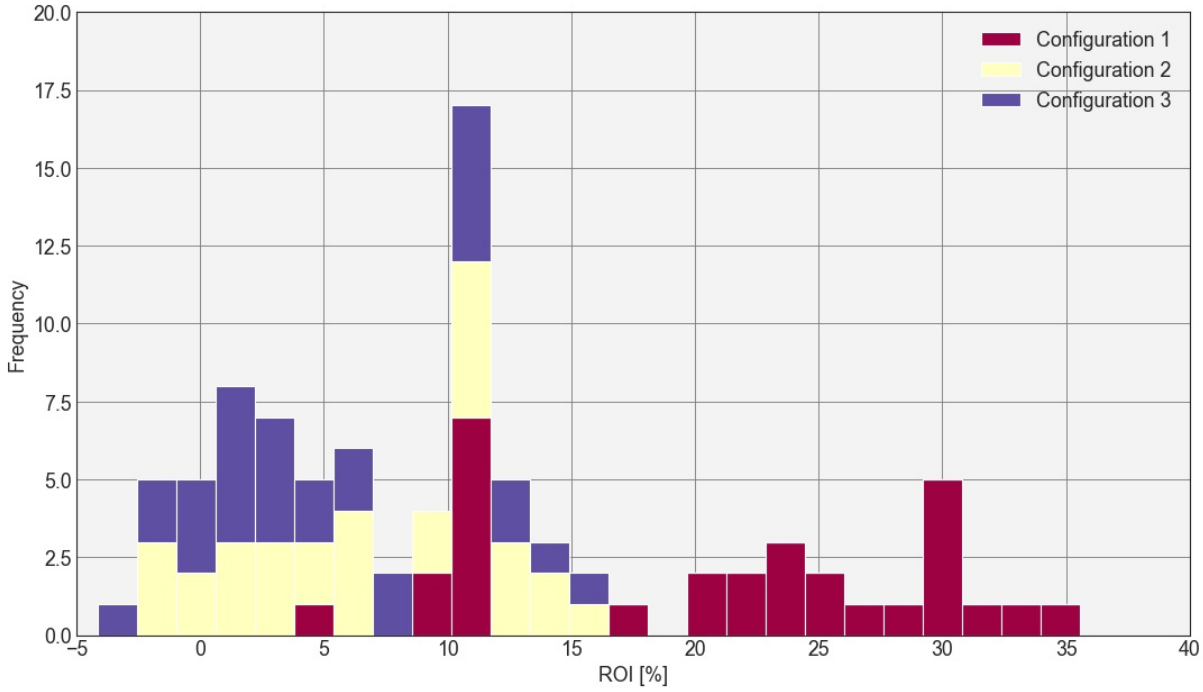


Figure 25: Histogram of the ROIs obtained by each trading rule set, using the Dynamic implementation of the GA.

obtained are subpar throughout almost every period.

These conclusion are confirmed by the overall results displayed in Table 21, from which it is perceivable that the Dynamic approach coupled with configuration 1 outperforms every tested combination in average ROI, while also presenting a satisfactory accuracy. It should be stated that, in spite of their unsatisfactory results, configurations 2 and 3 coupled with the Dynamic approach still performed better than if coupled with the Static approach, obtaining higher average ROI.

Table 21: Overall results of all the sets tested in Case Study C.

Metric	**Configuration 1**		**Configuration 2**		**Configuration 3 (DD)**	
	Dynamic	**Static**	**Dynamic**	**Static**	**Dynamic**	**Static**
Drawdown [%]	4.31	4.21	3.46	3.72	3.94	4.28
Accuracy [%]	57.23	63.39	53.63	58.09	53.29	59.14
Min ROI [%]	4.20	-8.97	-2.19	-5.98	-4.15	-7.96
Max ROI [%]	35.58	8.73	15.39	4.75	15.48	3.63
Avg. ROI [%]	20.96	2.57	6.49	0.39	5.21	-0.13
Std. Dev. [%]	8.86	4.05	5.23	2.20	5.30	2.68

4.2.4.2 Conclusions of Case Study C

The main take away from this case study would be that, the only acceptable method to use in combination with the implemented approach would be the one that generates a quinary signal (configuration 1), without the Double Down Strategy. This configuration, coupled with the Dynamic approach, ob-

tained the best results in this case study, presenting the best average ROI, and adequate values for the accuracy and drawdown.

Once again, the Dynamic approaches of the implementation presented more lucrative results than the Static counterparts. However, it should also be observed that, the Dynamic approach coupled with configuration 2 or 3, presented low accuracy values, making them inadequate simulation configurations.

4.2.5 Case Study D

This final case study aims to validate the proposed fuzzy implementation coupled either with the Dynamic GA or the Static GA, and to compare it with the crisp approaches of both algorithms. To start, it should be stated that each of the default input settings are present in the Methodology Section (4.2.1), such as the training and testing periods utilized (Table 9), the Wide Search parameters (Table 10) and the investment parameters (Table 11).

To test the fuzzy component coupled with the Dynamic GA, which utilizes the Narrow Search method as its testing module, one of the previously studied configurations was selected. The used configuration, which is present in Table 16, is similar to the configuration utilized as baseline in Case Study A (section 4.2.2) and similar to the configuration used throughout Case Study B (section 4.2.3) and Case Study C (section 4.2.4).

Two trading rule sets were selected in the validation process of this Case Study. Table 22 presents both of the sets, whose performance was previously analysed during Case Study B (section 4.2.3). The first set comprises the RSI, ROC, the EMA and the MACD, which, as stated previously, were successfully utilized by several authors in their frameworks. The second selected set add the Bollinger and the Stochastic rules to the first set, composing a total of six trading rules. The reason for the selection of these sets was the solid results presented in the previous case studies and their reduced overall computation time. The second set's purpose is also to analyse how the fuzzy implementation fares with an increased number of trading rules. To finalize, it should be stated that the selected trading rules' parameter ranges follow the guidelines present in Table 4, and the membership functions utilised in the fuzzy component of the model are present in section 3.4.1.

Table 22: Sets of trading rules used during Case Study D.

Set 1	**Set 2**
RSI	RSI
ROC	ROC
EMA	EMA
MACD	MACD
	Bollinger
	Stochastic

4.2.5.1 Crisp vs Fuzzy Evaluation

This section features an analysis of the results obtained by the Dynamic and Static GAs while employing a Crisp or a Fuzzy approaches using each of the aforementioned rule sets. Each of the combinations was executed 10 times for each of the time periods stated previously, resulting in a total of 30 executions per combination.

Figure 26 displays a histogram in which are represented the ROI values obtained by the crisp and fuzzy approaches of the Dynamic GA, coupled with each of the aforementioned trading rule sets. From these it might be concluded that the best results were obtained by utilizing a fuzzy approach in conjunction with the rule set 2.

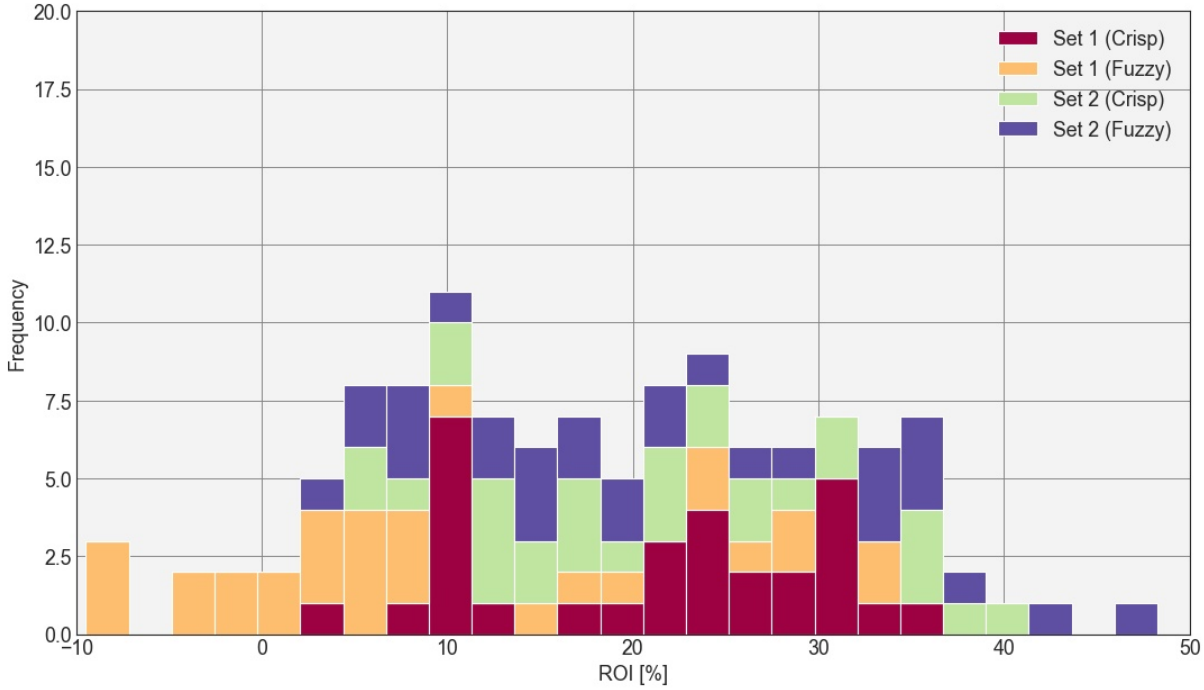


Figure 26: Histogram of the ROIs obtained by each trading rule set, using the Dynamic implementation of the GA.

It can also be observed that the crisp approach's presents consistent results independently of the trading rule set used, in contrast with the fuzzy approach's results, which demonstrate a great disparity between employing the fuzzy approach coupled with rule set 1 or employing it with rule set 2.

This disparity is also evident in the periodic results displayed in Table 28, present in Appendix D. While every Dynamic approach, independently of the rules used, presents an average ROI of above 20%, for the first testing period, the same is not true for the other periods. The results from the second testing period, once again, demonstrate a considerable drop off in every Dynamic approach, with the fuzzy approach that employed rule set 1 averaging a negative ROI during this period. The results of this approach improved in terms of average ROI in the third period, but it accuracy dropped below 50 %, which reflects the incapacity of this combination to obtain consistent profit in this period.

Overall, both variations of the crisp approach as well as the fuzzy approach that employed rule set 2 presented positive results in every testing period, despite the drop off of the ROI values characteristic of the second period and the accuracy drop off in third testing period.

These conclusions are also corroborated by the results presented in Table 23, which displays the overall results obtained by both the Static and Dynamic models when coupled either with the crisp or the fuzzy approach. From the results of the Dynamic approach, it is easily observable that using the crisp approach, independently of the rule set used, presents very similar profits and accuracy to the fuzzy approach coupled with rule set 2, with the latter being slightly more profitable. The results obtained by this fuzzy approach are, however, associated with a higher standard deviation, which evidences a higher level of result dispersion than the crisp approaches.

It should also be mentioned that, the results obtained for the Static GA coupled with fuzzy membership functions produced results slightly more profitable than the same algorithm in the its crisp version.

Table 23: Overall results of all the sets tested in Case Study D.

Metric	Set 1 (Crisp)		Set 1 (Fuzzy)		Set 2 (Crisp)		Set 2 (Fuzzy)	
	Dynamic	Static	Dynamic	Static	Dynamic	Static	Dynamic	Static
Drawdown [%]	4.31	4.21	3.41	3.36	4.09	3.41	4.80	3.08
Accuracy [%]	57.23	63.39	54.12	65.01	55.73	61.24	57.00	62.75
Min ROI [%]	4.20	-8.97	-7.38	-4.15	5.84	-7.38	2.32	-0.36
Max ROI [%]	35.58	8.73	34.27	14.37	40.01	8.16	48.25	10.24
Avg. ROI [%]	20.96	2.57	2.04	3.72	20.91	2.04	21.59	3.38
Std. Dev. [%]	8.86	4.05	12.49	3.62	9.77	2.99	12.16	2.63

Generally, the Dynamic GA coupled with the fuzzy approach present promising results with rule set 2, despite its higher result dispersion. In contrast, this same approach while, employing rule set 1, produced appalling results across all metrics, with a substandard accuracy, low average ROI and profit. The reason for the discrepancy of the results of the same approach while using the same rule sets, may be the erroneous tuning of one of the membership functions utilised by one of the rules that comprise rule set 1. Such problem would also affect the model when employing rule set 2, since all membership functions utilised by rule set 1 are also shared with rule set 2. However, an error like this would particularly affect the model when employing rule set 1, since it presents a smaller number of trading rules, which means that each one is more impactful to the model's outcome.

Finally the computation duration of both the crisp and the fuzzy approaches should be addressed. As discussed previously in case study B (section 4.2.3), the difference between the total computation times of rule sets 1 and 2 (, sets 1/baseline and 4 in case study B,) in a crisp approach is none existent, taking about of half the time that the other rule sets studied in that case study. Nonetheless, this total computation time quadruples when the fuzzy approach is employed with rule sets 1 and 2. The complex method of processing technical indicator values through the membership functions is one of the main causes for this long computation time. The other main reason is the fact that the implemented model creates a membership function each time a trading rule's signal is generated, which, as it can be guessed, happens several hundred of times each generation.

4.2.5.2 Conclusions of Case Study D

This section concludes Case Study D, from which several conclusions about the viability of a fuzzy approach can be pointed out.

The main conclusion to take away from this case study would be that the implemented fuzzy approach does not seem viable to implement in a real world setting. Its large computation time and the poor results of this approach coupled with rule set 1 are the main key contributors to this conclusion.

On the other hand, the same approach, while employing rule set 2, obtained slightly more profitable results than its crisp counterpart. In Figure 27, the average ROI results of each of the Dynamic approaches are represented throughout the testing periods. As highlighted previously, the fuzzy approach coupled rule set 1 present similar results to the other approaches in the first testing period (0-3000 samples), and a decline in performance in the succeeding testing periods.

These results and the fact that the Static GA in conjunction with the fuzzy membership functions present slightly more profitable results than the their crisp counterparts, are promising indicators that an optimised version of this approach may be implementable.

Figure 27: Evolution of ROI obtained by the crisp and fuzzy approaches throughout the totality of the testing period.

4.2.6 Dynamic vs Static

This section presents a review of the performance results of the Dynamic and the Crisp approach throughout the validation process.

4.2.6.1 Dynamic Approach: Dynamic Criteria Analysis

Before analysing the results, it should be proven that the Dynamic framework complies with the financial environment through a series of dynamic criteria, referred in the Background chapter of this work (2).

a) Predictability

Throughout this work no predictive algorithms (such as Hidden Markov Chains or the Random Forests) were employed. However, it is a fact that no considerable losses were presented in all the executions of the proposed dynamic framework, with the majority of the results proving themselves to be profitable and with an average position accuracy of around 57-58%. By utilising the Wide Search method to generate a population trained through a period of 3.5 years and the Narrow Search to train it again in a shorter period directly preceding the testing period, the proposed model turned out to be quite adaptable to the changes in the financial environment, despite it not containing any ”true” predictive methods.

b) Cyclicity

The Dynamic framework takes advantage of the cyclicity of the FOREX market, by saving a population that was trained in a long data period, which saves the general behaviour of this market. By retraining this population in a shorter training period preceding the testing period, the originally saved population is adapted to the current behavior of the market. After a certain number of retrains a new is generated through the the Wide search method again. This combination makes the adaptation process of the original population to the cyclic changes that occur in the market possible, independently if the market is in a downtrend, an uptrend or in a sideways trend.

c) Visibility

In the context of this framework, changes in the market could be detected through a fundamental analysis of the market, such as changes caused by geopolitical factors such as important news or press conferences from important politicians or central banks from the world. However, since the Proposed Framework's objective is to generate trading strategies only from the technical analysis standpoint, the sudden changes created by the geopolitical factors are not taken into account. This framework does not present a module that detect changes in the market per se, fundamental or technical, but is always able to react to them by re-adapting itself to the current behaviour of the market every n samples.

As for changes in the performance of the GA, the Proposed System presents several metrics that measure the fitness of the selected trading strategy in the training periods, namely the ROI and te position accuracy. The Stop Gain and Stop Loss features also allow the proposed system to adapt the selected trading strategy to sudden changes that may indicate great profits or losses, and to respectively, collect the profits or limit the loss.

d) Time-Linkage

As stated by Almeida et al. [6], it is possible to assume that this problematic does not present a time-linkage characteristic. As it was concluded in the cited work, the FOREX market, due to its massive volume, is independent of a single investor, as long as the positions made by him are on the level of the ones simulated in this work. As such the Proposed Framework would not have any influence in the environment.

The only theoretical way of this framework (or any comparable to it) influence the FOREX behaviour would be if a high number of investor would be running the same model with the same configurations and obtaining the same results, which is not realistic.

e) Stability-Robustness

Throughout all the Validation section the Proposed Dynamic Framework was tested in various testing periods, while using several configurations for different components of the model. The results of the tests demonstrate the robustness and stability of the implemented framework, with the majority of them producing profitable and accurate results.

From this section it is possible to conclude that the implemented model is not fully dynamic since, instead of detecting changes in the market environment and reacting to them, the model adapts itself to the markets behaviour in predetermined intervals of time. However, it should be stated that the implemented framework does present dynamic properties, missing only a change detection functionality.

Solutions to achieve a fully dynamic framework will be further discussed in the final remarks of this work.

4.2.6.2 Dynamic vs Static: Overall Results

This section presents an overall summary of the results obtained in the most pertinent case studies (A and B) to analyse the performance of the implemented Dynamic approach against its Static counterpart.

Throughout the validation process several input configurations were tested in several modules of the implemented framework. To start, in Case Study A (section 4.2.2), several configuration parameters concerning the Narrow Search were tested and analysed. The results obtained from the different configuration employed demonstrated that, the implemented method presents a high level of stability and robustness, achieving profitable results independently of the configuration used. It is fair to state that some configurations did present substandard accuracy results, however it should be stated that the worse results obtained still present an average ROI four times higher than the ROI obtained by the Static GA implementation presented in this same case study.

In Figure 28 the several results obtained throughout the three testing periods are represented. Among them there are the best and average results obtained by the baseline configuration and the configuration that obtained the better results. Along with these, the best results obtained by the Static GA are also present. As evidenced by the ROI evolution throughout the three periods of each approach it is easily observable that, although the Static GA approach presents steady profits, these are outmatched by the steady profits of the Dynamic approach.

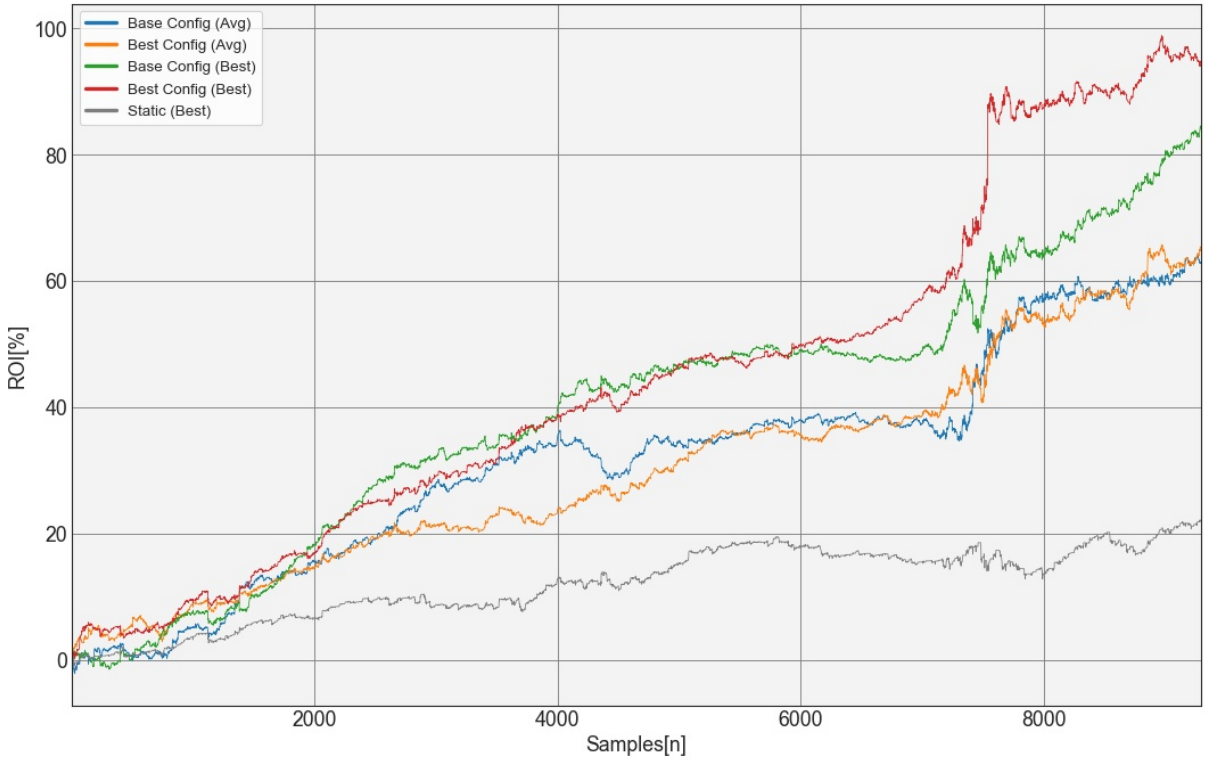


Figure 28: Evolution of ROI obtained by the relevant configurations of case study A throughout the totality of the testing period.

In Case Study B (section 4.2.3) the focus point turns to the trading rules utilised in the approach. The results obtained by using different sets of trading rules demonstrate once more the robustness of the Dynamic implementation, with every approach obtaining profitable results, independently of the set employed. The impact of using different trading rules in the Static approach was also studied, with it also obtaining profitable results, although with a much smaller ROI values. Figure 29 presents the average evolution of the ROI of the Dynamic approach coupled with rule sets 1 and 3 and the

best results of their Static counterparts. This figure evidences a significant drop off when the Dynamic approach employs rule set 3 instead of rule set 1, during the second and third testing periods (3000-9000 samples). However, the Static approach presents slightly better results when employs rule set 3. Despite the fact that, rule set 3 causes a significant decline in the profitability of the Dynamic and a slight increase in the profitability of the Static approach, the Dynamic framework still presents more profitable results throughout all the testing periods.

This figure adds strength to the conclusion of case study A, demonstrating how robust the implementation of the Dynamic approach is in dealing with different trading rule sets.

Figure 29: Evolution of ROI obtained by the relevant rule sets of case study B throughout the totality of the testing period.

4.2.6.3 Dynamic vs Static: Comparison with Benchmarks and State of the art

To finalise this chapter, a comparison between the implemented Dynamic and Static approaches against the benchmarks is provided.

Figure 30 presents the average and best results obtained by the of the Dynamic approach employing the baseline configurations used throughout the Validation Section. This figure also displays the best results obtained by the Static GA and the results obtained by the benchmark strategies Buy and Hold and Sell and Hold in the same testing periods.

The results achieved by both the Static and Dynamic approaches compared to the ones obtained through the use of the benchmark strategies are a testament to the consistency and profitability of the frameworks implemented throughout the work, as demonstrated by this figure. A detailed overview of these results is present in Table 24, from which it is possible to observe that after the totality of the testing period, the both implemented approaches (Dynamic and Static) produced more profits than any of the benchmarks, while achieving considerably lower drawdowns.

Table 24: Results of the baseline approaches compared with the benchmark strategies

Metric	Baseline (Avg)	Baseline (Best)	Static (Best)	Buy and Hold	Sell and Hold
ROI (%)	62.77	84.35	22.13	-10.10	10.10
Drawdown (%)	5.72	5.30	5.59	40.63	25.24

Figure 30: Evolution of ROI obtained by baseline implementation, the static implementation and the benchmarks throughout the totality of the testing period.

The works of Hirabayashi et al. [5] and Almeida et al.[6], present themselves as perfect objects of comparison with the implemented framework, due to the fact that both also apply a variation of a Genetic Algorithm approach to the FOREX market, and present their results with some of the evaluation metrics used in this work. When compared with the leverage approach presented by Hirabayashi et al. [5], which presents a maximum ROI of around 40% across a year of data, the implemented Dynamic framework average results present a ROI value inline with these results, while the best results present about 10% more. The implemented framework also presents a steadier behaviour with relatively low drawdowns, which is a quality missing in the work of Hirabayashi et al. [5].

It should however be stated that, although both works are aimed at the FOREX market, Hirabayashi et al. [5] presents results concerning the major currency pairs envolving JPY, while this work is focused in the EUR/USD currency pair. On the other hand Almeida et al. [6] presents a prototypical dynamic approach, and focuses its testing in the same currency pair as the this work.

The Proposed Dynamic GA approach by Almeida et al. [6] also presents an average ROI of around 43% across a year and 2 months of testing data, which is slightly better than the results obtained by this work's results. However, this ROI value is associated with an higher value of drawdown, presenting a value of 9% against the 5% obtained in this work.

These works highlight the successful used of the Evolutionary Computation in the FX market.

It should also be stated that the works by Gorgulho et al. [4], Fernandez et al. [25] and Silva et al.

[26] present works with comparable frameworks to the one implemented during this work, but applied them at the Stock market instead of the FOREX market. When compared with the results obtained by the implemented framework, these works present lower profit values across a larger testing period, which can be related to the different environment to which they are associated.

4.3 Chapter Conclusions

This chapter analysed four different Case Studies aimed to test the performance of the implemented approaches using different configurations for the input parameters. Each case study presented a set of results from which several conclusions were drawn. Throughout the Validation process it was proven that the implemented Dynamic approach presents a more profitable results than the Static approach, while maintaining a reduced exposure.

The results obtained in Case Studies A and B (respectively sections 4.2.2 and 4.2.3), highlight the robustness of the Dynamic approach to different parameter configurations for the Narrow Search method and different sets of trading rules.

Case Study C (4.2.4) tests different methods of generating investment strategies (trading signals), and concludes that only method 2 (algorithm 3) is viable to use in the context of this framework.

In Case Study D (4.2.5) an analysis of the performances of the implemented fuzzy approach and its crisp counterpart is present. During this case study it was concluded that the current implementation of fuzzy approach does not appear to be viable to real time trading, due to its instability and long computation time. However, it is stated that some the tests done to the Static approach and one of the Dynamic approaches present promising results.

Finally, an overview of the results obtained by the Static and Dynamic approaches is present at the end of this chapter, in which several benchmarks and works from the literature are used as objects of comparison.

Chapter 5

Conclusions

This final chapter addresses the final conclusions obtained during this work, and proposes further tests and improvements to the implemented architecture.

5.1 Conclusions

In this work a Dynamic approach that combines traditional GA with the proprieties of the Narrow Search method is proposed implemented. This model is based in the technical analysis principals and was tested in the FOREX market environment. This architecture was also coupled with a fuzzy logic approach to accomplish the objective of analysing the viability of an architecture such as this in a financial market such as the Foreign Exchange.

Throughout its testing several conclusions were drawn about the performance of the implemented approaches. First the robustness and efficiency of the Dynamic implementation were overwhelmingly positive, exceeding the profits of its main benchmark, a Traditional Static GA. It is important to mention that these profits are associated with low exposure values, presenting itself as relatively safe investment strategy generator.

This robustness and stability presented themselves throughout every case study in which the impact of parameter configurations and the use of trading rules was studied. This and the fact that this approach preformed better than several benchmarks and comparable frameworks from the state of the art indicate the accomplishment of one of the main objectives of this thesis.

The third and final conclusion of this work is that the fuzzy implementation coupled with the Dynamic approach created in context of this thesis does not present itself as viable to real time testing due to its long computation times and high exposure values, when compared with its crisp counterpart. This goes without saying that, in several executions this approach presented itself more profitable than the crisp version. These are very promising results which indicate that a profitable and stable fuzzy implementation coupled with the Dynamic approach may be possible. It should also be mentioned that, since this implementation presents several performance metrics, it is expectable that authors use it as an object of comparison for future works within the same scope.

To finalise it should be stated that a User Interface for the implemented architecture was roughly drafted and implemented. However this component was not addressed during this thesis due to it being incomplete and not adding any substantial to the results of the work.

5.2 Future Work

Despite the accomplishment of the proposed objectives, this thesis does have room for improvements and would benefit of further testing in some areas the implementation. For future works future directions or improvements are suggested:

- Further testing with the implemented architecture is required. Such tests should include use several configuration to test the wide search method to improve the population's diversity prior to testing. Tests to the fuzzy approach should also be executed, such as, tune the trading rules' membership functions and test other types of membership function, such as the gaussian function.
- Other tests that should prove quite useful, would be to test several investment strategies related to the leverage values employed. A pretty straightforward solution would be to include the leverage value in the chromosome, as evidenced in the work of Almeida et al. [6].
- Improvements to make the Dynamic approach fully dynamic should also be pursued. Methods that allow the algorithm to automatically detect environment changes before retraining the population should be further looked upon.
- Complete and improve the User Interface so that simulating a given configuration and environment become a more intuitive process.
- The major area that should deserve some attention would be to apply this architecture to a real time trading environment in which the real world constraints can be simulated.

Bibliography

[1] M. Burgin and E. Eberbach, "Evolutionary turing in the context of evolutionary machines," 04 2013.

[2] K. Tang, K. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 22–37, 1996.

[3] M. Paulinas and A. Ušinskas, "A survey of genetic algorithms applications for image enhancement and segmentation," *Information Technology and control*, vol. 36, no. 3, 2007.

[4] A. Gorgulho, R. Neves, and N. Horta, "Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition," *Expert Systems with Applications*, may 2011. [Online]. Available: https://doi.org/10.1016%2Fj.eswa.2011.04.216

[5] A. Hirabayashi, C. Aranha, and H. Iba, "Optimization of the trading rule in foreign exchange using genetic algorithm," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 1529–1536. [Online]. Available: https://doi.org/10.1145/1569901.1570106

[6] B. Jubert de Almeida, R. Ferreira Neves, and N. Horta, "Combining support vector machine with genetic algorithms to optimize investments in forex markets with high leverage," *Applied Soft Computing*, vol. 64, pp. 596 – 613, 2018. [Online]. Available: https://doi.org/10.1016/j.asoc.2017.12.047

[7] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S001999586590241X

[8] P. Juszczuk and L. Kruś, "Soft multicriteria computing supporting decisions on the forex market," *Applied Soft Computing*, vol. 96, p. 106654, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494620305925

[9] R. Naranjo, A. Meco, J. Arroyo, and M. Santos, "An intelligent trading system with fuzzy rules and fuzzy capital management," *International Journal of Intelligent Systems*, vol. 30, no. 8, pp. 963–983, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/int.21734

[10] M. Galant and M. Brian Dolan, *CURRENCY TRADING FOR DUMMIES*. Wiley India Pvt. Limited, 2007. [Online]. Available: https://books.google.pt/books?id=QKmR2LEazJQC

[11] "Foreign exchange turnover in april 2019," "[accessed 2-January-2021]". [Online]. Available: https://www.bis.org/statistics/rpfx19_fx.htm

[12] A. Petropoulos, S. P. Chatzis, V. Siakoulis, and N. Vlachogiannakis, "A stacked generalization system for automated forex portfolio trading," *Expert Systems with Applications*, vol. 90, pp. 290 – 302, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417305493

[13] D. Pradeepkumar and V. Ravi, “Soft computing hybrids for forex rate prediction: A comprehensive review,” *Computers Operations Research*, vol. 99, pp. 262 – 284, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054818301436

[14] M. D. Archer, *Getting Started in Currency Trading: Winning in Today’s Hottest Marketplace*, ser. Getting Started In. Wiley, 2008. [Online]. Available: https://books.google.pt/books?id=ei2_RD704qwC

[15] O. Momoh, “How leverage works in the forex market,” ”[accessed 25-November-2020]”. [Online]. Available: https://www.investopedia.com/ask/answers/06/forexleverage.asp

[16] M. Ozturk, I. H. Toroslu, and G. Fidan, “Heuristic based trading system on forex data using technical indicator rules,” *Applied Soft Computing*, vol. 43, pp. 170 – 186, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494616300369

[17] J. Fernando, “Moving average convergence divergence,” ”[accessed 12-December-2020]”. [Online]. Available: https://www.investopedia.com/terms/m/macd.asp

[18] J. Bollinger, *Bollinger on Bollinger Bands*. McGraw-Hill Education, 2002. [Online]. Available: https://books.google.pt/books?id=RxkVMFg-KIIC

[19] “%b indicator),” ”[accessed 27-September-2021]”. [Online]. Available: https://school.stockcharts.com/doku.php?id=technical_indicators:bollinger_band_perce

[20] J. Wilder, *New Concepts in Technical Trading Systems*. Trend Research, 1978. [Online]. Available: https://books.google.pt/books?id=WesJAQAAMAAJ

[21] “Average directional index (adx),” ”[accessed 13-September-2021]”. [Online]. Available: https://school.stockcharts.com/doku.php?id=technical_indicators:average_directional_index_adx

[22] “Stochastic oscillator,” ”[accessed 14-September-2021]”. [Online]. Available: https://school.stockcharts.com/doku.php?id=technical_indicators:stochastic_oscillator_fast_slow_and_full

[23] L. Martí, “Algoritmos genéticos,” 2014, ”[accessed 27-November-2020]”. [Online]. Available: http://lmarti.com/genetic-algorithms-algoritmos-geneticos-eng1456

[24] P. H. Winston, “Lecture 13: Learning: Genetic algorithms,” ”[accessed 14-December-2020]”. [Online]. Available: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/lecture-videos/lecture-13-learning-genetic-algorithms/

[25] P. Fernández, D. J. Bodas Sagi, F. Soltero, and I. Hidalgo, “Technical market indicators optimization using evolutionary algorithms,” 01 2008, pp. 1851–1858.

[26] A. Silva, R. Neves, and N. Horta, “A hybrid approach to portfolio composition based on fundamental and technical indicators,” *Expert Systems with Applications*, vol. 42, no. 4, pp. 2036 – 2048, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417414006113

[27] M. Bahrepour, M.-R. Akbarzadeh-T., M. Yaghoobi, and M.-B. Naghibi-S., “An adaptive ordered fuzzy time series with application to FOREX,” *Expert Systems with Applications*, vol. 38, no. 1, pp. 475–485, jan 2011. [Online]. Available: https://doi.org/10.1016%2Fj.eswa.2010.06.087

[28] S. Yang, “Genetic Algorithms with Memory- and Elitism-Based Immigrants in Dynamic Environments,” *Evolutionary Computation*, vol. 16, no. 3, pp. 385–416, 09 2008. [Online]. Available: https://doi.org/10.1162/evco.2008.16.3.385

[29] “Dukascopy-historical data feed),” ”[accessed 5-May-2021]”. [Online]. Available: https://www.dukascopy.com/swiss/english/marketwatch/historical/

[30] “Python data analysis library — pandas: Python data analysis library.” [Online]. Available: https://pandas.pydata.org/

[31] “Technical analysis library,” ”[accessed 14-September-2021]”. [Online]. Available: https://github.com/bukosabino/ta

[32] “Technical analysis library in python’s documentation,” ”[accessed 14-September-2021]”. [Online]. Available: https://technical-analysis-library-in-python.readthedocs.io/

[33] N. Lioudis, “Moving average strategies for forex trading,” ”[accessed 25-September-2021]”. [Online]. Available: https://www.investopedia.com/ask/answers/122314/how-do-i-use-moving-average-ma-create-forex-trading-strategy.asp

[34] “Macd (moving average convergence/divergence oscillator),” ”[accessed 27-September-2021]”. [Online]. Available: https://school.stockcharts.com/doku.php?id=technical_indicators:moving_average_convergence_divergence_macd

[35] *Fuzzy Membership Function Generators*, ”[accessed 20-December-2020]”. [Online]. Available: https://pythonhosted.org/scikit-fuzzy/api/skfuzzy.membership.html

Appendix A

Case Study A: Periodic Results

Table 25: Periods results of all configurations tested in Case Study A.

	Metric	Base	1/2*TS	3TS	1/2*TS_2*ts	2*ts	3TS_2*ts	5Gen	5Mut	30Elites	5Elites	Static
1^{st} Period	**Drawdown** [%]	3.12	2.90	3.95	2.88	2.54	2.52	3.53	5.00	4.79	4.59	3.92
	Accuracy [%]	62.11	60.11	61.28	61.22	64.27	62.48	56.91	55.89	55.87	56.06	57.57
	Min ROI [%]	8.99	12.43	8.29	9.02	18.37	14.51	2.78	5.02	-6.15	2.77	-7.37
	Max ROI [%]	32.63	27.27	34.60	24.45	30.88	29.47	24.34	25.86	25.26	24.35	8.73
	Avg. ROI [%]	26.39	22.37	22.64	17.76	23.13	21.05	16.08	13.28	14.65	13.85	3.26
	Std. Dev. [%]	6.78	4.03	7.92	5.31	4.62	4.38	6.68	6.20	9.13	6.98	5.09
2^{nd} Period	**Drawdown** [%]	4.22	4.43	3.89	5.77	4.15	3.42	6.46	5.81	5.86	5.29	2.72
	Accuracy [%]	56.38	55.16	57.62	56.28	56.20	59.89	56.15	56.38	56.73	57.23	65.56
	Min ROI [%]	4.20	-3.37	5.63	1.09	1.79	9.15	-1.40	3.98	1.66	-0.17	0.77
	Max ROI [%]	17.22	22.53	20.53	14.03	19.26	21.78	15.28	16.06	22.04	19.56	8.10
	Avg. ROI [%]	10.57	8.78	14.39	8.27	9.46	14.65	6.92	8.80	9.57	10.56	3.82
	Std. Dev. [%]	2.94	7.47	4.37	3.86	4.35	4.11	5.09	4.07	5.34	5.05	1.99
3^{rd} Period	**Drawdown** [%]	5.58	6.04	6.43	4.55	5.30	4.99	6.79	7.90	6.60	6.21	5.99
	Accuracy [%]	53.21	54.00	54.50	53.65	55.34	56.00	49.72	48.12	50.12	50.02	67.03
	Min ROI [%]	21.03	14.31	15.62	15.31	12.86	18.02	13.73	-2.62	10.57	7.56	-8.97
	Max ROI [%]	35.58	43.72	28.89	42.08	44.66	43.84	22.05	26.50	26.37	32.82	5.60
	Avg. ROI [%]	25.91	29.28	22.94	32.24	25.30	28.02	17.50	12.42	18.53	18.96	0.63
	Std. Dev. [%]	4.37	10.45	4.45	7.32	8.71	7.75	2.98	8.94	5.36	7.46	3.68

Appendix B

Case Study B: Periodic Results

Table 26: Periodic results of all configurations tested in Case Study B.

	Metric	Dynamic				Static GA			
		Set 1 (Base)	Set 2	Set 3	Set 4	Set 1 (Base)	Set 2	Set 3	Set 4
1^{st} Period	**Drawdown** [%]	3.12	2.93	2.01	2.96	3.92	3.66	2.25	3.08
	Accuracy [%]	62.11	61.43	64.17	58.76	57.57	63.30	66.03	58.48
	Min ROI [%]	8.99	7.92	13.50	14.25	-7.37	0.75	2.33	-7.38
	Max ROI [%]	32.63	21.37	34.19	40.01	8.73	6.83	12.61	7.59
	Avg. ROI [%]	26.39	15.08	21.71	25.46	3.26	3.60	6.81	1.90
	Std. Dev. [%]	6.78	3.44	6.80	7.85	5.09	2.01	3.44	4.25
2^{nd} Period	**Drawdown** [%]	4.22	4.34	3.94	4.23	2.72	3.84	3.05	2.46
	Accuracy [%]	56.38	59.23	59.34	55.64	65.56	67.85	69.65	62.70
	Min ROI [%]	4.20	1.99	-2.08	5.84	0.77	-3.25	1.39	-0.09
	Max ROI [%]	17.22	14.29	17.85	16.06	8.10	1.49	5.34	4.29
	Avg. ROI [%]	10.57	8.84	9.38	11.11	3.82	-1.09	4.24	1.15
	Std. Dev. [%]	2.94	4.93	6.56	3.32	1.99	1.28	1.17	1.14
3^{rd} Period	**Drawdown** [%]	5.58	5.86	7.37	5.09	5.99	6.74	5.31	4.68
	Accuracy [%]	53.21	59.63	54.78	52.80	67.03	64.15	65.73	62.54
	Min ROI [%]	21.03	15.12	-6.06	10.09	-8.97	-4.08	0.52	0.00

Table 26 continued from previous page

Max ROI [%]	35.58	42.76	21.94	37.69	5.60	6.91	4.31	8.16
Avg. ROI [%]	25.91	27.20	8.99	26.16	0.63	1.30	2.66	3.08
Std. Dev. [%]	4.37	8.93	8.67	8.33	3.68	3.59	1.19	2.36

Appendix C

Case Study C: Periodic Results

Table 27: Periodic results of all configurations tested in Case Study C.

	Metric	Configuration 1		Configuration 2		Configuration 3 (DD)	
		Dynamic	Static	Dynamic	Static	Dynamic	Static
1^{st} Period	**Drawdown** [%]	3.12	3.92	2.84	3.14	3.43	3.41
	Accuracy [%]	62.11	57.57	57.48	54.89	54.81	56.29
	Min ROI [%]	8.99	-7.37	-2.19	-3.33	-1.59	-3.06
	Max ROI [%]	32.63	8.73	14.83	3.81	13.21	3.19
	Avg. ROI [%]	26.39	3.26	6.14	-0.28	3.78	-0.23
	Std. Dev. [%]	6.78	5.09	4.87	2.07	4.06	1.73
2^{nd} Period	**Drawdown** [%]	4.22	2.72	4.14	2.34	4.69	2.20
	Accuracy [%]	56.38	65.56	53.44	58.75	53.67	63.46
	Min ROI [%]	4.20	0.77	-1.56	-0.15	-4.15	-0.19
	Max ROI [%]	17.22	8.10	6.13	4.75	2.77	3.63
	Avg. ROI [%]	10.57	3.82	1.92	1.80	0.67	1.84
	Std. Dev. [%]	2.94	1.99	2.44	1.28	2.12	1.10
3^{rd} Period	**Drawdown** [%]	5.58	5.99	3.42	5.70	3.69	7.23
	Accuracy [%]	53.21	67.03	49.98	60.64	51.41	57.68
	Min ROI [%]	21.03	-8.97	5.40	-5.98	7.98	-7.96

Table 27 continued from previous page

Max ROI [%]	35.58	5.60	15.39	3.98	15.48	3.23
Avg. ROI [%]	25.91	0.63	11.41	-0.36	11.18	-2.01
Std. Dev. [%]	4.37	3.68	2.67	2.38	2.21	3.14

Appendix D

Case Study D: Periodic Results

Table 28: Periodic results of all configurations tested in Case Study D.

	Metric	Set 1 (Crisp)		Set 1 (Fuzzy)		Set 2 (Crisp)		Set 2 (Fuzzy)	
		Dynamic	Static	Dynamic	Static	Dynamic	Static	Dynamic	Static
1^{st} Period	**Drawdown** [%]	3.12	3.92	3.45	2.28	2.96	3.08	2.26	1.79
	Accuracy [%]	62.11	57.57	59.90	62.87	58.76	58.48	61.87	65.15
	Min ROI [%]	8.99	-7.37	2.35	-4.15	14.25	-7.38	28.16	1.16
	Max ROI [%]	32.63	8.73	34.27	14.37	40.01	7.59	48.25	9.08
	Avg. ROI [%]	26.39	3.26	22.54	5.50	25.46	1.90	36.44	5.42
	Std. Dev. [%]	6.78	5.09	8.91	5.17	7.85	4.25	5.26	2.16
2^{nd} Period	**Drawdown** [%]	4.22	2.72	10.99	3.10	4.23	2.46	4.32	2.76
	Accuracy [%]	56.38	65.56	54.09	67.54	55.64	62.70	57.63	61.36
	Min ROI [%]	4.20	0.77	-9.48	2.76	5.84	-0.09	5.90	-0.36
	Max ROI [%]	17.22	8.10	5.69	6.26	16.06	4.29	23.98	5.04
	Avg. ROI [%]	10.57	3.82	-2.46	3.95	11.11	1.15	13.76	1.76
	Std. Dev. [%]	2.94	1.99	5.06	1.12	3.32	1.14	5.17	1.62
3^{rd} Period	**Drawdown** [%]	5.58	5.99	10.84	4.71	5.09	4.68	7.83	4.70
	Accuracy [%]	53.21	67.03	48.36	64.64	52.80	62.54	51.51	61.75
	Min ROI [%]	21.03	-8.97	-0.15	-1.98	10.09	0.00	2.32	0.49

Table 28 continued from previous page

Max ROI [%]	35.58	5.60	9.32	5.17	37.69	8.16	26.43	10.24
Avg. ROI [%]	25.91	0.63	5.58	1.71	26.16	3.08	14.57	2.96
Std. Dev. [%]	4.37	3.68	2.67	1.99	8.33	2.36	7.61	2.56