TÉCNICO LISBOA

Portfolio Composition Based on High-Ranking Hedge Funds Using Dynamic Algorithms

João Gustavo Guerreiro Costa da Silva Carita

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor

Prof. Rui Fuentecilla Maia Ferreira Neves

Prof. Nuno Cavaco Gomes Horta

Examination Committee

Chairperson: Prof. António Manuel Raminhos Cordeiro Grilo

Supervisor: Prof. Rui Fuentecilla Maia Ferreira Neves

Members of Committee: Prof. Nuno Filipe Valentim Roma

December 2017

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

The work presented proposes a system that optimizes a portfolio of stocks based on the hedge funds' assets, strategies and by tracking their past performances using Dynamic strategies in Genetic Algorithms. Most works that developed systems for the stock market use technical and fundamental indicators as features, but in this work, the hedge fund public data was used to model the behaviour of these all-star teams of financial experts. Being the stock market search space time-variant, the system developed is going to exploit a set of Dynamic strategies based on state-of-the-art approaches, such as *Temporal Seeding*, *Direct Memory*, *Immigrant* manipulation and a custom *Hyper-state*. The proposed architecture used information from 12th of February of 2014 until 12th of February of 2016. The proposed system outperformed the benchmark indexes for an entire quarter, obtaining an average *Return On Investment* of 5.7% and a *Sharpe Ratio* of 3.26.

Keywords

Hedge Funds, Portfolio, Genetic Algorithms, Dynamic Systems, Financial Markets

Resumo

O trabalho proposto apresenta um sistema que optimiza um portfolio de acções baseando-se em informações de fundos de investimentos, recorrendo aos seus activos, estratégias e desempenhos passados utilizando estratégias Dinâmicas associadas a Algoritmos Genéticos. A maioria dos trabalhos aplicados a mercados financeiros recorrem a dados baseados em indicadores técnicos e fundamentais, mas no contexto deste trabalho foi utilizada informação pública de fundos de investimento para modelar o comportamento destas equipas de especialistas financeiros. Tendo em conta que o mercado de acções é um sistema variante no tempo, o sistema desenvolvido vai explorar um conjunto de estratégias dinâmicas baseadas no estado da arte, tais como *Temporal Seeding,* Memória Directa, manipulação de imigrantes e um *Hyper-state* personalizado. A arquitectura proposta utilizou informação financeira de 12 de Fevereiro de 2014 até 12 de Fevereiro de 2016. O sistema proposto superou os *benchmarks* indexes selecionados durante um trimestre inteiro, obtendo um Retorno de 5.7% e *Sharpe Ratio* de 3.26.

Palavras-Chave

Fundos de Investimento, Portfolio, Algoritmos Genéticos, Sistemas Dinâmicos, Mercados Financeiros

Table of Contents

List of Figures

List of Tables

List of Acronyms

B&H	Buy & Hold
CHF	Swiss Franc
CZK	Czech Koruna
D/E	Debt to Shareholders Equity Ratio
DAX 30	Deutscher Aktienindex
DJIA	Dow Jones Industrial Average
DOP	Dynamic Optimization Problem
DY	Dividend Yield
EP	Evolutionary Programming
EPS	Earnings Per Share
EUR	Euro
FTSE 100	Financial Times Stock Exchange 100
GA	Genetic Algorithm
GBP	Great Britain Pound
GPM	Gross Profit Margin
HUF	Hungarian Forint
KOSPI 200	Korean Composite Stock Price Index
MC	Market Capitalization
MODistGA	Multi-Objective Distributed Genetic Algorithm
MOGA	Multi-Objective Genetic Algorithm
NASDAQ	National Association of Securities Dealers Automated Quotations
NIKKEI	Nihon Keizai Shimbun 225 Stock Average
ROE	Return On Equity
RSI	Relative Strength Index
S&P 500	Standard & Poor’s 500
SE	Shareholders’ Equity
SO	Stochastic Oscillator
USD	United States Dollar

List of Symbols

c_j	Crossover unary transformation on an individual j.
$f_{min,w}$	Lowest observed fitness in last w generation.
$\tilde{f}$	Average fitness in current population.
H_n	Highest price of n previous trading sessions.
L_n	Lowest price of n previous trading sessions.
m_i	Mutation unary transformation on an individual i.
x_i^t	Individual i at time t.
x_p	Individual x gene p.
y_p	Individual y gene p.
σ_f	Standard deviation of the fitness in the current population.
$\%D$	3 period Moving Average of %K.
$\%K$	Stochastic Oscillator parameter.
c	Constant.
C	Most recent closing price.
$F(t)$	Set of viable solutions.
$f(x)$	Fitness function of individual x.
H	Search Space.
k	Constant.
n	Number of previous trading sessions.
$N(0,\sigma)$	Gaussian distribution with average zero and standard deviation σ.
$o(z,t)$	Objective function.
$P(t)$	Population of individuals at time t.
Pr	Probability of reproduction.
Q	Normalization factor.
RS	Relative Strength.
S	Data-Structure.
t	Time instant.
T	Constant.
w	Scaling window.
x	Individual
y	Individual
z	Possible solution.

α	Number of reproductions by the least fit individual.
β	Number of reproductions by the fittest individual.
γ	Rank of the individual's fitness inside a population.
λ	Population size.
σ	Standard deviation.
τ	Total time.

Chapter 1

Introduction

This introductory chapter provides a synopsis of the Thesis. Section 1.1 provide a general scope and a brief historical contextualization. Sections 1.2 and 1.3 give details about Thesis motivation and objectives. Finally, section 1.4 concludes chapter 1 by presenting this work structure.

1.1 Evolutionary Computation and Finance: Overview

The Evolutionary computation field is defined by a group of algorithms called evolutionary algorithms, that are inspired in several biological mechanisms of evolution. This research field offers several approaches that will be explained later in section 2.2.

This thesis will implement a Genetic Algorithm. In this kind of algorithms, the search task assigned is done using procedures that are inspired by natural evolution. These so-called procedures are commonly called Genetic Algorithms and rest on concepts that are familiar with ideas like individuals, mating, chromosome, gene mutation, fitness and natural selection. All these concepts were described by Charles Darwin in his work *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life* published at 1872. In his work, Darwin defend the principle of Evolution through natural selection. He states firstly that each individual tends to pass on its traits to its offspring, secondly that nature produces individuals with different traits, thirdly that the fittest individuals – those with most favourable traits – tend to have more offspring than do those with unfavourable traits, thus driving the population as a whole toward favourable traits and finally that over long periods, variation can accumulate, producing new species whose traits make them especially suited to a particular ecological niche. Genetic algorithms rests in the principles above to obtain a heuristic solution to the given optimization and search problem.

Genetic algorithms were first introduced by John Holland in 1960s, being the first to explicitly propose crossover and other recombination operators. Subsequently this kind of algorithms were developed by Holland and his students and colleagues at the University of Michigan in the 1960s and 1970s. The earlier research and papers created both by Holland and his colleagues at the University of Michigan laid the foundation for Holland's book called *Adaptation in Natural and Artificial Systems* where he presents the concept of adaptive digital systems using mutation, selection, crossover, simulating processes of biological evolution, as a problem-solving strategy [1].

John Holland's original objective, in comparison with evolutionary programming, was to develop a formal study of the phenomenon associated with adaption as it happens in nature and create ways in which the mechanism of natural adaptation might be imported into computer systems, instead of simply design algorithms to solve specific problems. In his works, he presented the genetic algorithms as an abstraction of biological evolution mechanism and gave a theoretical foundation for adaption under the genetic algorithms paradigm.

Holland's original genetic algorithm is a design where one population of "chromosomes" moves into a new population by applying "natural selection" with genetic-inspired methods of mutation, crossover and inversion. Each one of these chromosomes consists of "genes", being this gene an instance of a specific "allele" – number of alternative forms of the same gene. The natural selection operator chooses the best chromosomes in the overall population that will be granted to reproduce, in a way that the fitter chromosomes produce more offspring when compared with the less fit ones. The crossover operator is responsible for the exchange of subparts of two chromosomes, to mimic the recombination between two

single-chromosome organisms. Mutation in Holland's context is compelled to randomly change the allele values of some locations in the chromosome, and inversion just reverses the order of a contiguous section of the chromosome, rearranging the order in which genes are presented. Holland's theoretical work on genetic algorithms where a population experiences the effects of crossover, mutation and inversion was revolutionary and composes a notable innovation and put genetic algorithms on a firm theoretical footing.

These earlier works established a solid base and spread the interest in evolutionary computation. In the 1980s, genetic algorithms were already applied to a wide range of fields, from mathematical problems to tangible engineering issues like pattern recognition and classification [2].

In the beginning, genetic algorithms usage was merely theoretical, yet, as research continued to spread, this kind of algorithms converged toward the commercial sector, being fed by the exponential growth of computing power and the creation of the Internet. Today, evolutionary computation became a leading-edge field, and genetic algorithms are a well-used design to everyday interest problems to a wide range of issues, such as, stock market prediction and portfolio composition, microchip design, molecular biology or aerospace engineering. The implementation of the evolutionary paradigm affected almost every field and is starting to change the world as research goes by. All the innovation bought by genetic algorithms comes from nothing more than Charles Darwin's simple and mighty principle: that the random chance of variation, combined with the law of selection, is a powerful technique for problem-solving and have nearly unlimited applications.

As stated in the paragraph above, these evolutionary-motivated algorithms have many applications in the computational finance field like portfolio composition, stock market prediction or to balance technical rules for stock picking. The work developed in this thesis focus on a portfolio composition and optimization problem using genetic algorithms.

1.2 Motivation

The Stock Exchange Market is a mechanism that allows shares, bonds and other kinds of securities to be traded between private investors and institutions. The financial markets didn't remain untouched after the artificial intelligence rise that we faced in the most recent years, becoming a common tool for predicting outcomes, and started to be vastly used by institutions, such as banks and private funds.

The time series generated every second by the financial markets show that the systems to tackle in this thesis' work are populated with noise and are non-stationary, thus being hard to study and predict. This noisy, non-stationary structure presented on stocks' prices is the reason why many investors and researchers use machine learning, to discover the underlying arrangement, present in the price signals from the stock market.

To date, there are several studies that develop investment strategies by balancing technical indicators,

such as relative strength index, moving averages, rate of change and many others. However, not much attention has been given to other financial behaviours and macroeconomic factors. Under this context, alternative investment strategies are less studied when compared to the regular ones which mainly use technical indicators.

One alternative, sometimes neglected, source of information about financial markets are the assets held by the biggest hedge funds in the world. Information on purchase or sale orders of shares by hedge fund managers allow to sometimes predict the evolution of future prices of that asset. When the shares are undervalued there is a tendency of increase in purchases, and when they are overvalued increased buyback. Having this into account, our main goal is to use hedge fund behaviours with techniques of intelligent computing to optimize investment strategies.

This thesis motivation is to achieve a machine learning model that understands and profits from the investment strategies applied by different hedge funds, thus providing portfolios of assets that are considered viable in the future.

1.3 Objectives

In this thesis context, it is expected to produce an application capable of predicting profitable and risk-controlled portfolios of stocks based both on assets controlled by the biggest American Hedge Funds and the past prices of the different stocks. The application developed in the context of this thesis, must have the power to do predictions, adopting soft computing procedures.

Applying Genetic Algorithms, our objective is to develop an investment strategy that can pick a weighted group of stocks from many available financial assets, to minimize the risk and maximize the expected return, being this work a resource allocation problem with competing objectives such as risk and expected return. The model optimizes both stocks and their weights relative to each other.

In a more concise and organized manner, the main objectives of this thesis are:

- Retrieve and parse all the Hedge Fund's information publicly available;
- Analyse and Simulate all Hedge Fund's portfolios;
- Use Hedge Fund's behaviour and stock's past prices to create portfolios with good performance in the future;
- Explore new approaches and topologies of Genetic Algorithms in a portfolio optimization process;
- Explore different strategies of Dynamic Genetic Algorithms;
- Study the model's performance at different types of markets, and inspect the model's performance versus the Standard & Poor's 500.

The thesis' presents the following contributions:

- Develop a stochastic combinatorial model using Dynamic Genetic Algorithms, that returns a portfolio of stocks and a valorization rule that translates into the stock weights;
- Assemble a Genetic Algorithm that applies dynamic strategies that invoke a hyper-state when certain conditions are met, while taking advantage of memory to preserve good solutions and temporal seeding for a better tune of our population of portfolios;
- Propose a stochastic system based on fundamental indicators, that mimics a Hedge Fund portfolio using stock sectors and industries to balance our portfolio exposure to risk;
- Develop a scrapping system able to automatically retrieve hedge fund information.

1.4 Thesis Structure

This work is divided into five main chapters:

- **Chapter 1 Introduction –** gives an overview about the task at hand with a brief historical contextualization, motivation, objectives and contributions;
- **Chapter 2 Background –** focus on the theoretical foundations of all the basic concepts that compose this work, more precisely, financial markets and genetic algorithms. Ultimately, it is presented the state of the art of the problem under study, where is given an overview about other methodologies.
- **Chapter 3 Proposed Architecture and Implementation –** shows the adopted methodologies to successfully conclude this thesis' work, as well as the software's architecture and the computational improvements made to boost our application.
- **Chapter 4 System Validation –** introduce a solid validation procedure to assess the solution performance.
- **Chapter 5 Conclusions –** encapsulates the work developed and concludes about the strategy used and the overall performance. Finally, are presented the future work propositions.

Chapter 2

Background

This chapter presents an overview about all the theoretical concepts that compose the work developed. It starts by giving a brief explanation about financial markets and the underlying mechanisms involved, after that will be provided a deep explanation about genetic algorithms and dynamic optimization problems. Finally, is given the state of the art research, where will be explored the works on portfolio composition using genetic algorithms and dynamic optimization problems using evolutionary strategies.

2.1 Financial markets

2.1.1 Introduction

In a market economy, the allocation of economic resources is the outcome of many private decisions. Thus, prices become the signals operating in a market economy that direct economic resources towards their best use. An economy can be divided into two kinds of markets - the product market (manufactured goods and services) and the factor market (labour and capital) [3].

This thesis will focus on a segment of the factor market called financial market, which is a market where financial assets are traded. Nevertheless, the existence of a financial market is not a fundamental condition for the creation and exchange of a financial asset, however in most world economies this kind of asset is always created and traded in some type of financial market.

Technological evolution increased the integration and efficiency of the financial market. Advances in telecommunication systems allowed to connect market participants around the world, letting financial assets to be traded within seconds. These advances combined with the advances in computer technology granted investors with systems that allow real-time transmission of information on financial assets price and other key elements to many participants in many places. This system enabled investors to monitor global markets and at the same time the opportunity to rapidly analyse this data through computer programs to predict outcomes and risks. After this an investor, can instantly place a buy/sell order in the market from almost everywhere [3].

2.1.2 Financial assets

An asset is any resource that has value in a trade and can be tangible or intangible. A tangible asset price always depends on physical properties, such as currencies, real estate, equipment or buildings. Intangible assets, on contrary, represent legal claims on future benefits, so their value does not carry any correlation with form, physical, or otherwise, in which these claims are recorded.

Financial assets do not necessarily have physical worth, so this kind of asset is considered as an intangible asset. Financial assets derive its value from a contractual claim, being the typical benefit or value a claim to future cash. Financial assets can be, for example, bank deposits, bonds or stocks.

2.1.3 Hedge funds

The term “hedge fund” defines a group of different investment vehicles that share similar characteristics. Hedge fund term includes any type of pooled investment vehicle that is privately organized, managed by professional investment managers and not widely accessible to the public, i.e., the average investor.

The main financiers of this kind of investment vehicle are institutional investors and wealthy individuals. The hedge fund manager generally has a share in the funds they manage [4].

As the hedge funds became bigger and bigger, the U.S. Securities and Exchange Commission forced these investment vehicles, with over $100 million in marketable securities, to report all their holdings quarterly by submitting the form 13F [5].

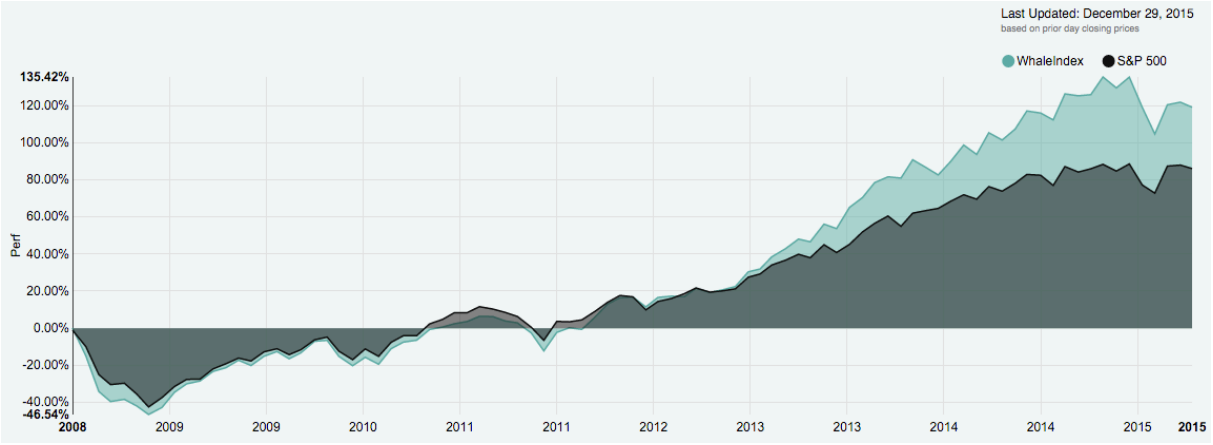

Figure 1. Whale Index Vs S&P 500 [6]

This kind of funds usually use strategies that take more risk than usual when compared with mutual funds, although their profits usually outperform benchmark indexes such as Standard & Poor's 500 (S&P 500) stock index. The figure 1 shows the whale index, which tracks all the S&P 500 stocks owned by leading hedge fund managers and compares it with S&P 500. To achieve this kind of results, hedge funds often use strategies like short-selling, arbitrage and hedging.

Hedge funds are active market participants that provide benefits to the financial markets by improving liquidity and efficiency, being able to play an important role in financial innovation and the reallocation of financial risk. Nonetheless, hedge funds, like other large highly leveraged financial institutions, have the power to manipulate the markets, disrupting the functioning of financial markets, mainly in vulnerable economies.

2.1.4 Technical Analysis

Technical analysis is a market forecast tool that visually studies a financial asset past prices. This kind of analysis mainly uses historical asset price data to predict future market trends. Technical analysis doctrine supporters believe that when prices and other technical indicators are plotted as a function of time, certain patterns that have predictive power can be observed.

As said above, technical indicators are the core of this forecast approach. A technical indicator is a series of data points that are calculated by applying a formula to the price or volume data of a specific financial asset. Some technical indicators used to estimate market trends are shown in the table 1 below.

Table 1. Technical Indicators

TECHNICAL INDICATOR	FORMULAE
RELATIVE STRENGH INDEX	$RSI = 100 - \frac{100}{1 + RS}$, $RS = \frac{Avg.\ of\ x\ days' up\ closes}{Avg.\ of\ x\ days' down\ closes}$
STOCHASTIC OSCILLATOR (SO)	$\%K = 100 \times \frac{C - L_n}{H_n - L_n}$ $n = days\ period\ considered$ $C = most\ recent\ closing\ price$ $L_n = lowest\ price\ of\ n\ previous\ trading\ sessions$ $H_n = highest\ price\ of\ n\ previous\ trading\ sessions$ $\%D = 3\ period\ Moving\ Average\ of\ \%K$

This doctrine advocate that the evolution of prices over time contains specific patterns that can be used to predict the future events on financial markets.

The relative strength index (RSI) is a technical momentum indicator that measure the strength or weakness of a financial asset to determine if is over-bought or over-sold. RSI allows an investor to track an asset losses and gains over a period of time, giving indications about the right moment to buy or sell. RSI is a momentum oscillator, since it measures an asset directional price movements. This technical indicator ranges from 0 to 100, if an asset shows a value higher or equal than 70 is considered over-bought and considered to be overvalued, if shows a value lower or equal than 30 is considered over-sold and is likely to be undervalued.

The stochastic oscillator measures an asset closing price in comparison to its price range over a given period of time and is used to help investors to define the best time to buy or sell. The stochastic oscillator is composed by two parameters, %K and %D, and both have a range of 0 to 100. If %D is higher than %K the security is thought to be over-bought, if %D is lower than %K the asset is thought to be over-sold. The sell signal is generated when %K and %D are both above 80 and a buy signal is generated when both are below 20.

2.1.5 Fundamental Analysis

Fundamental analysis approach consists on the study of the underlying forces that manipulate the economy, industry groups or companies. The main objective of this kind of approach is to forecast and profit from future price changes. This forecast approach states that markets, in some moment, may misprice a financial asset but, as the time goes by, the market will eventually acknowledge the security "true" price and the asset will increase its value. The investor, using fundamental analysis, should exploit this situation by buying an asset that is mispriced and sell when the asset reaches its "true" price, making profit.

As explained before, in this section, fundamental analysis is based upon three different kinds of analysis. At first, there is the company layer analysis where we analyse financial data (for example financial

statements which are composed by cash flow statement, balance sheet and income statement), direct and indirect competitors or the management decisions through history. Afterwards, lies the industry layer, where should be studied market shares, laws and regulation, industry growth achieved in the past years or supply and demand. At last, economic level analysis will study macro-economic factors to predict an economy growth, unemployment, inflation or purchasing power of the average consumer.

By implementing a strategy using fundamental analysis, one must combine these three layers of analysis to derive an asset "true" value or to forecast an asset performance in the future. This kind of approach as proven to be worthy as a tool to forecast a stock performance and is supported by Wall Street gurus such as Warren Buffet [7] or Benjamin Graham [8].

Fundamentalists can follow two distinct approaches, a top-down approach or a bottom-up approach. The top-down investor employs a strategy that will start with a specific economy and after that will decrease the level of analysis to industry and finally company. The bottom-up follows the same logic but uses the reverse order, starting with company analysis, then industry and at last macro-economic analysis.

a) Economic Analysis

Companies and industries both belong to a specific economical tissue. Taking this into account, is possible to extrapolate that the macro-economic factors will impact an economy industrial sectors and subsequently his own companies as a result.

A detailed economic analysis can be used to hypothesize an economy trend which will reflect an industry growth or decline, depending on the conclusions. Economists have developed several economic indicators that ultimately can be used as a definition of an economy actual state and predict the forthcoming events.

All economic data needed is usually reported by the governments and independent institutions. Although, economic data about the same matters is frequently ambiguous since most of the information disclosure is usually influenced by political reasons. The ideal investing strategy must be capable to pick neutral data in order to reach real conclusions that will reflect an economy "true" shape.

Economic analysis has become more complex due to globalization increase and technologies development. As markets and economies shift towards a more interconnected state, economic inquiry turn into a more complex task because more variables are needed to be considered to fully understand a specific economical tissue.

b) Industry Analysis

Economic growth or decline will define the trend that a specific industry group will follow. When the economy grows, most industry sectors benefit and grow. When economy fall, most industry groups deteriorate and plunge. If an economical forecast points that economy will grow, certain groups will most likely benefit more when compared to others, so an investor must choose the best industry sectors to

invest.

In industry analysis, an investor must compare the industry sectors against each other to conclude which sectors will perform better. By doing this, an investor can narrow alternatives by choosing sectors that are best suited to thrive from current or future economical mediums. A fundamentalist can also select the portfolio composition based on the overall performance of that sector against all the others, i.e., if energy and financial sectors performed better than the rest, an investor should have more stocks from both this sectors compared to the rest. To evaluate an industry sector potential, an investor should consider different factors, like growth rate, regulation, syndicates, market size, consumers or the importance of that sector to the overall economy.

In the market, commonly, stocks move as a group, so more important than have the perfect stock is to be in the perfect sector. Choosing the right company is important, but its industry group is likely to influence just as much, or more its value. The following charts illustrates the 5-year performance of several industrial sectors distinct from each other [7].

In a 5-year run there are sectors that clearly outperform the rest, showing a steady growth as time goes by, such as health care, consumer discretionary and technology. Other sectors show poor results like energy, materials and utilities. This kind of industrial analysis can be important when choosing each sector weight on a portfolio.

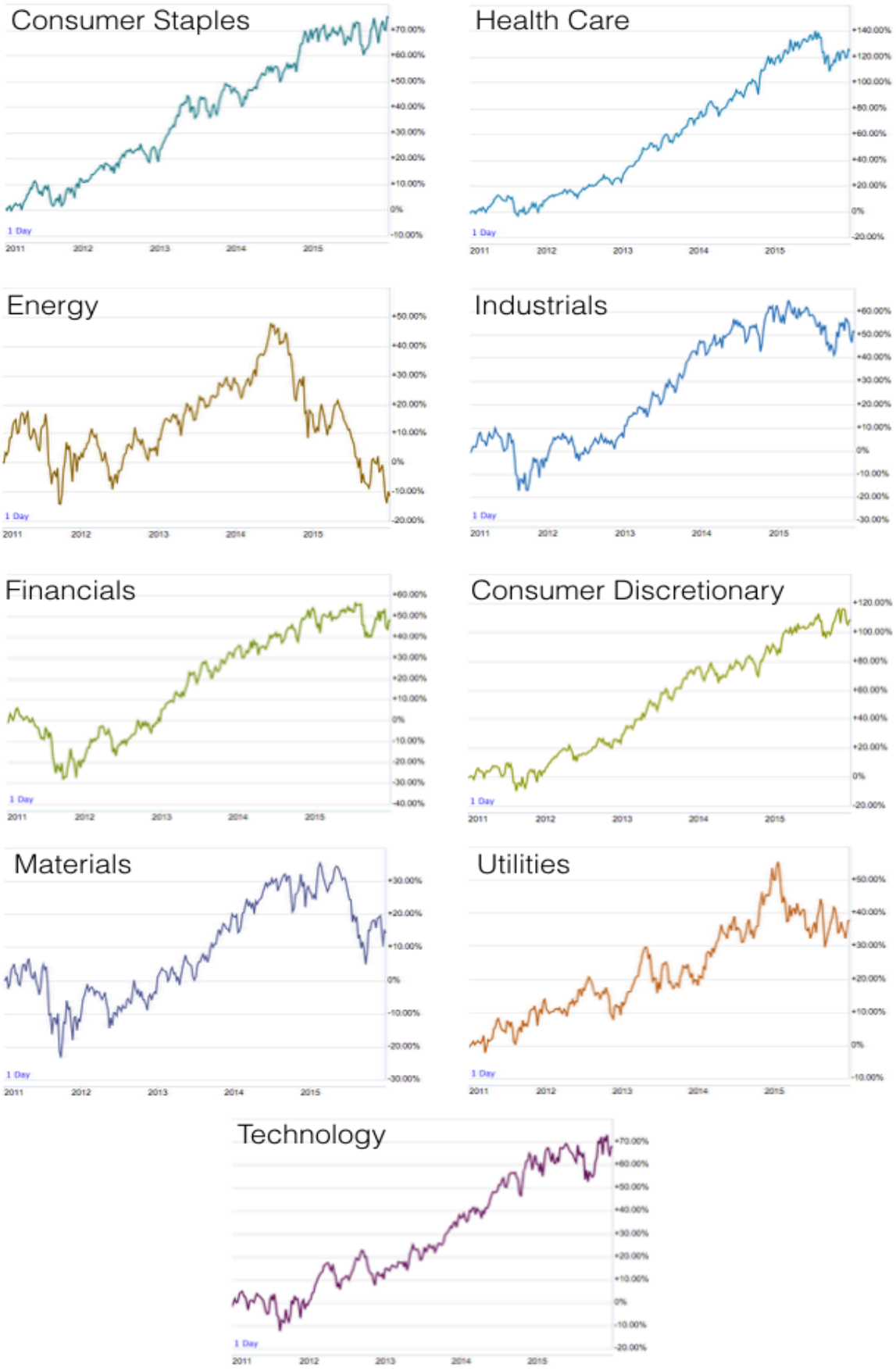

Figure 2. 5-year returns of S&P 500 stocks per sector [43]

c) Company Analysis

After choosing an industry group to invest, an investor must choose companies that are leaders and *avant-garde*. This companies must have a solid market share and a competitive advantage in order to thrive above all the others, so the task at hand is to find the right fundamental indicators to identify these characteristics without error. To fulfil the expressed above, an investor should analyse a company true capability and resources using the right fundamental indicators, in a way that in the end he chooses the best companies with a strong potential to create and maintain a true competitive advantage [8].

To achieve a substantial study about a corporation we should take into account more subjective parameters, i.e. business plan and management personnel. After this, we should proceed to the most important step of all, the financial analysis of the company at study.

The financial analysis will evaluate the financial health of a specific corporation by studying the financial statement. The financial statement is divided into three parts called cash flow statement, balance sheet and income statement, and all the public companies are obliged to publish them every quarter. These statements give us information about a company's assets, gross profit margin, debt, dividends, revenues, taxes, patents and many other data that is used as input values to the fundamental indicators that allow us to classify a company profit, efficiency, liquidity or actual growth [7]. The table 2 below enumerates a few fundamental indicators used by investors.

Table 2. Fundamental Indicators

FUNDAMENTAL INDICATOR	FORMULAE	DESCRIPTION & STRATEGY
RETURN ON EQUITY (ROE)	$ROE = \frac{Net\ Income}{Shareholder's\ Equity}$	Measures company's profitability by showing how much profit a firm generates with the shareholders' money.
GROSS PROFIT MARGIN (GPM)	$GPM = \frac{(Revenue - Cost\ of\ Goods\ Sold)}{Revenue}$	Metric to assess a company's financial health by revealing the money left over from revenues after paying the costs of good sold. It should not fluctuate, unless the industry is suffering drastic changes.
SHAREHOLDERS' EQUITY (SE)	$SE = Assets - Liabilities$	Represents the amount by which a company is financed through shares sold.
DEBT TO SHAREHOLDERS' EQUITY RATIO (D/E)	$D/E = \frac{Total\ Liabilities}{Shareholders'Equity}$	Debt ratio that measures a corporation financial leverage and indicates how much debt a company is using to finance its assets per share. High D/E indicates that a company has been aggressive in financing its growth with debt.
DIVIDEND YIELD (DY)	$DY = \frac{Annual\ Dividends\ per\ Share}{Price\ per\ Share}$	Ratio that estimates how much a corporation pays in dividends each year relative to it's share price.
EARNING PER SHARE (EPS)	$EPS = \frac{Net\ Income - Dividends\ on\ Preferred\ Stocks}{Average\ Outstanding\ Common\ Shares}$	The quantity of a company's profit allocated to each share. It might be used as an indicator of a company's profitability.
MARKET CAPITALIZATION (MC)	$MC = Company's\ Shares \times Market\ Price$	The total market value of all company's shares. The investor might use it to determine a company size, and oppose sales or total assets figures.

2.2 Evolutionary computation – Genetic Algorithms

2.2.1 Introduction

Computational systems based on evolutionary computation handle a population of potential and apply processes of selection based on the adaption of the individual to the problem at hand, as well as many other operations depending on the algorithm applied. Many evolutionary approaches have been proposed to different problems, like ant colony optimization, artificial immune systems, genetic algorithms, swarm intelligence and many others. All these approaches were developed independently but share the same principle and base structure, as seen in the pseudo code shown below [9].

```
Procedure evolution algorithm
begin
    t ← 0
    initialize P(t)
    evaluate P(t)
    while (not termination-condition) do
    begin
        t ← t + 1
        select P(t) from P(t − 1)
        alter P(t)
        evaluate P(t)
    end
end
```

Figure 3. Pseudo code of an evolutionary algorithm

As seen in the figure above, an evolutionary algorithm controls a population of individuals represented by $P(t) = \{x_1^t, \dots, x_n^t\}$ with n individuals at a specific iteration t. Each one of the individuals x_i^t represent a candidate solution to the problem and is implemented as some data structure S. Each one of the solutions $x_i^t \in P(t)$ is evaluated by a specific measure of "quality" or "fitness" that classifies that individual potential as solution to the given problem. Next, a new population $P(t+1)$ is conceived by selection of the fittest individuals among population $P(t)$, this stage is called selection. After, there is the alter step that by means of "genetic" operators are going to form new solutions among the population $P(t+1)$ by applying two unary transformations, m_i and c_j. The first transformation, m_i, create a new individual by applying a short change in a single individual just like mutations, i.e. $m_i : S \to S$. The second transformation is going to perform a high order unary transformation, called crossover, by combining segments of two or more individuals $x_i^t \in P(t+1)$, thus creating a new one with combined characteristics, i.e. $c_j : S \times \dots \times S \to S$. After some generations, the algorithm converges and eventually

reaches the stop condition. In the end, the final population should provide an acceptable solution among all individuals to the task at hand [9].

The evolutionary approaches diverge from each other in many aspects, such as data structures used to represent individuals, genetic operators applied, initial population generation procedures and methods to choose the individuals to the next generation. Nonetheless, all these approaches have a common basic principal, i.e., a population of individuals suffers some transformations, and, at the same time these individuals compete for survival.

2.2.2 Standard Genetic Algorithm

Genetic Algorithms (GA) were inspired on the theory of evolution described by Charles Darwin, and uses abstractions like natural selection, genetics, mutation and many others. Using a GA approach, an individual of the population is represented by a single chromosome that contains the codification (genotype) of a possible solution to the problem (phenotype). In this context, the data structure most commonly used to represent a chromosome is an array, where each attribute inside is going to be a gene.

The evolution process executed by the GA is a search algorithm for potential solutions inside a specific search space, therefore is essential to find an equilibrium point between two conflicting objectives: choosing the best solutions and at the same time keeping the diversity, in order to search the whole space of possible solutions (exploitation vs exploration). GA become a satisfactory solution to achieve this kind of equilibrium, showing that is possible to successfully explore both good solutions and the search space at the same time. GAs, despite having nondeterministic steps, are not purely random search algorithms, because they combine random variations like mutation and crossover with a fitness evaluation of the individual, and maintain a population of other possible solutions.

A GA search is going to be multi-directional by maintaining a population of multiple solutions and encouraging the information trade between them to achieve better solutions. In each generation, the best solutions are encouraged to reproduce (trade information) and generate a new offspring of candidate solutions, however, this kind of algorithm continues to discard possible solutions with worst performance. To make this evaluation of the best or worst solutions, a GA makes use of a specific fitness function that emulates the pressure made by the environment upon the individuals of a population.

The genetic algorithm structure is similar to the pseudo code presented in figure 3, and can be described:

- At each iteration t, a genetic algorithm maintains a population $P(t) = \{x_1^t, \dots, x_n^t\}$;
- Each individual x_i^t inside the population $P(t)$ is a possible solution and is evaluated by a fitness function that classify him by a measure of fitness;
- A new population is created (iteration $t + 1$) by indulging the fittest individuals;
- Some individuals of the new population are transformed by means of crossover and mutation, forming new potential solutions;
- This algorithm repeats until several pre-determined iterations is completed or until a certain level

of optimization is reached.

A genetic algorithm must have the following characteristics implemented: A genetic representation to the solutions for the given problem; A procedure to create an initial population of potential solutions; A rating function that emulates a specific environment, ranking solutions in means of “fitness”; Genetic operators responsible to change the composition of the population’s children; and well defined values for key parameters of the GA, for example, population size, probabilities of applying a determined genetic operator, and many others depending on the implementation and features of the GA.

a) Chromosome Encoding

Each individual from a population is represented by a specific chromosome that represents a potential solution to the given problem. The classic GA, proposed by John Holland, encodes the chromosomes using binary arrangements, which are supported by the fundamental theorem of genetic algorithms. In short, the fundamental theorem of genetic algorithms states that short, low-order schemata with above-average fitness increase exponentially in successive generations, being a schema a template that classify a subset of strings with correlation at certain string positions.

Holland states that this kind of implementation can be used to achieve good performances while maximizing the implicit parallelism inherent to the GA, and proves that the binary alphabet can maximize this parallelism.

However, in various practical applications a binary encoding can reach unsatisfactory performances. This conclusion is supported by Michalewicz [9], stating that a binary representation can lead to a poor performance when applied to numeric problems in high dimensions and that require a high precision solution. So, he proposes the usage of float values in these cases.

We can conclude that the task of choosing the encoding applied is a critical step in any work related with GAs, because it will affect the convergence of the genetic algorithm to possible solutions. The structure of the chromosome must represent all the solutions inside a search space and must be simple at the same time, that’s why most of the works developed using GAs use arrays as a data structures.

b) Initial Population

The initial population given to the GA can be generated in multiple ways. The most broadly used implementation is to randomly generate the initial population by creating each chromosome of the population with random values. Although, if there are any prior knowledge about the desired solution this should be used in the initial population generation.

c) Genetic Operators

After the selection has been made and chosen the fit individuals of a specific population, we must randomly change these individuals, in order to find fittest ones in the next generation. The most commonly used methods to achieve the expressed above are mutation and crossover.

c.1) Crossover

The crossover method is inspired in the sexual reproduction concept, where is observed the recombination phenomenon between two distinct chromosomes. In this method, two individuals are chosen from a set of fit individuals to produce offspring by swapping segments of their code, emulating the effect of having artificial "offspring" that are the result of the recombination of the parent's code material.

The most used method to implement crossover is the single-point crossover. In this crossover method, a point of exchange is set at a given/random location in both individuals' chromosomes. Consequently, the first individual will contribute with all the code until that point and the other contributes with all his code from after that point thus producing a new individual. This method can be seen and better understood in the figure 4.

Parent X	010101	010	Offspring 1	010101	001
Parent Y	110011	001	Offspring 2	110011	010

Figure 4. Single-point crossover example

As seen in figure 4, if we have two parents with configuration $\{x_1, \dots, x_n\}$ and another with configuration $\{y_1, \dots, y_n\}$ and we define $p \in [1, 2, \dots, n]$, the offspring becomes $\{x_1, \dots, x_p\}.\{y_p, \dots, y_n\}$ and $\{y_1, \dots, y_p\}.\{x_p, \dots, x_n\}$.

Another method commonly used is the multi-point crossover. Multi-point crossover is similar to the single-point method, but instead of having only one point of crossover, it will use more than one. An example of this process can be seen in figure 5.

Parent X	010	101	010	Offspring 1	010	011	010
Parent Y	110	011	001	Offspring 2	110	101	001

Figure 5. Multi-point crossover example with two points

In this implementation, two parents with configuration $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_n\}$, with $p_1 < \cdots < p_k$ and $k \in [2, \dots, n]$, generates $\{x_1, \dots, x_{p_1}\}.\{y_{p_1+1}, \dots, y_{p_2}\}.\{x_{p_2+1}, \dots, x_{p_3}\}$ and $\{y_1, \dots, y_{p_1}\}.\{x_{p_1+1}, \dots, x_{p_2}\}.\{y_{p_2+1}, \dots, y_{p_3}\}$, as seen in figure 5.

Finally, there is the uniform crossover [10], in which the value at every location of the offspring is either the value of one parent's chromosome at that same location or the value of the other parent's chromosome at that location, chosen with a 50/50 probability.

Parent X	010 101 010	Offspring 1	010 001 011
Parent Y	110 011 001	Offspring 2	110 111 000

Figure 6. Uniform crossover example

The crossover operators shown before can be used in chromosomes using float encoding. However, there are operators specifically designed to these cases. One example is arithmetic crossover, that uses

a linear combination of both parents' chromosomes. In arithmetic crossover, considering that we have two parents x and y, then the offspring is going to become $x' = ax + (1 - a)y$ and $y' = (1 - a)x + ay$, where $a \in [0,1]$ is randomly chosen.

c.2) Mutation

The mutation operator of a GA randomly modifies one or more genes of a chromosome. The probability of mutation in a gene is called, in GA context, rate of mutation and, usually, has a small probability of occurrence. The basic idea behind mutation is to find new solutions inside the whole search space without compromising the progress already made by GA's successive searches.

In a binary encoding context, the mutation operator simply changes the value of a gene inside a chromosome by switching his bit value from one to zero or vice-versa.

In a floating-point context, the mutation can be made using different methods like uniform mutation, Gaussian mutation, and many others. In the uniform mutation, the method randomly pick a gene inside the chromosome and randomly modifies this gene with a new value (this value must be inside this gene range of accepted values). In the Gaussian mutation method, every gene inside a specific chromosome is changed by adding noise that follows a Gaussian distribution, as seen in the equation below:

$$x' = x + N(0, \sigma), \quad (1)$$

Where $N(0, \sigma)$ is an array of Gaussian random variables, with average zero and standard deviation σ.

Other mutation method similar to Gaussian mutation is called Cauchy mutation. They are both quite similar but the second uses a Cauchy distribution instead. This new approach performs better when the distance to the global optimum is far away. Another approach would be using both mutation distributions by generating two offspring, one using a Gaussian distribution and another using Cauchy distribution, keeping the best one.

c.3) Selection

The selection step of the GA is responsible for selecting the individuals to be copied over into the next generation. There are many techniques to perform selection, many of them are mutually exclusive but some of them can be combined to achieve better performances. The selection methods most commonly used are:

- **Elitist selection:** The fittest members of each population are selected and always pass to the next generation. This kind of approach is usually combined with another by using some form of elitism. In this hybrid approach of the elitist selection, only the best chromosome or just a few of the best chromosomes from each generation are copied to next generation, to ensure that the best solutions are kept and not lost by chance.
- **Tournament selection:** All the individuals of a population are grouped randomly into subgroups of individuals. Inside this subgroup, the individuals compete against each other and the fittest survives, passing to the next generation and being able to reproduce.

- **Scaling selection:** This method transform the raw fitness scores returned by the implemented fitness function into a more suitable range of values. Then uses the scaled fitness values of each individual to select the ones that pass to the next generation. This method is helpful when selecting individuals in a relatively high fitness population, because as the average fitness of the population increases, the strength of the selective pressure also increases and the fitness function will become more and more discriminating. Several forms of scaling can be implemented, and can be seen in table 1. In the following table $f(x)$ represents the fitness function.

Table 3. Scaling selection types

SCALING TYPE	FORMULAE
SIMPLE SCALING	$\tilde{f}(x) \coloneqq f(x) - f_{min,w}$
	w, scaling window $f_{min,w}$, lowest observed fitness in last w generations
SIGMA SCALING	$\tilde{f}(x) \coloneqq min\{0, f(x) - (\bar{f} - c.\sigma_f)\}$
	c, constant $\bar{f}$, average fitness in current population σ_f, standard deviation of the fitness in the current population
POWER SCALING	$\tilde{f}(x) \coloneqq f(x)^k$
	$k > 0$
EXPONENTIAL SCALING	$\tilde{f}(x) \coloneqq e^{\frac{f(x)}{T}}$
	$T > 0$

- **Rank selection:** Every individual inside a specific population is ranked according to its fitness. Then selection is made based on this ranking. The advantage of this approach is that the most fit individuals will be prevented from gaining early dominance over the rest. This measure increases the population diversity. The following table shows different forms of assigning survival probabilities according to the individual's rank inside the population. In the following table consider a population with size λ, each individual with rank γ, $0 \leq \gamma \leq \lambda - 1$ and $\sum_{\gamma=0}^{\lambda-1} Pr_{type} = 1$, being $\alpha + \beta = 2$ where $1 \leq \beta \leq 2$. The β represents the number of reproductions made by the fittest individual and α the number of times the least fit individual reproduced. Finally, Q represents a normalization factor and $0 < \alpha < \beta$.

Table 4. Ranking selection types

RANKING TYPE	FORMULAE
LINEAR RANKING	$Pr_{linear}[\gamma] \coloneqq \frac{\alpha + (\beta - \alpha)\frac{\gamma}{\lambda - 1}}{\lambda}$
POWER RANKING	$Pr_{power}[\gamma] \coloneqq \frac{\alpha + (\beta - \alpha)\left(\frac{\gamma}{\lambda - 1}\right)^{k}}{Q}$
GEOMETRIC RANKING	$Pr_{geometric}[\gamma] \coloneqq \frac{\alpha + (1 - \alpha)^{\lambda - 1 - \gamma}}{Q}$
EXPONENTIAL RANKING	$Pr_{exponential}[\gamma] \coloneqq \frac{1 - e^{-\gamma}}{Q}$

- **Fitness proportional selection:** This method for selection generates a survival probability that is proportional to the fitness amount by which the individual is greater or lesser than the rest of individuals' fitness. With this method, the chance of survival becomes proportional to the individual's fitness. The probability of survival is given by the following function:

  $$\Pr[x] = \frac{f(x)}{\sum_{y \in P} f(y)}, \quad (2)$$

  Where x represents a specific individual, P represents the overall population, y represents another individual inside the population and $f(x)$ represents the fitness function.
- **Steady-state selection:** In this approach, some individuals are selected to reproduce, all the resulting offspring replaces other individuals inside the population that were ignored by the selection process.
- **Generational selection:** This selection procedure selects individuals to perform crossover, forming the new generation only by using the resulting offspring.

2.3 Dynamic Optimization Problems

2.3.1 Introduction

Programming has become a widely-used tool to solve every kind of problem that mankind faces nowadays. That led to new kinds of problems for computer science to solve, mainly problems derived from fields like engineering, medicine and finance. These fields gave origin to new complex problems never approached in computer science.

The problems described above, usually have great uncertainty associated, caused by time-varying systems with constantly changing variables, coefficients, behavior and goals. The demand for new software that solve this kind of problems catalyzed intelligent systems programming and many researches started addressing this using different perspectives and methodologies. One approach to

solve this cases is called dynamic optimization problem (DOP) where some facts change over the course of execution. DOPs are often characterized by [11]:

$$DOP = \begin{Bmatrix} optimize\ o(z,t) \\ s.t.\quad z \in F(t) \subseteq H, t \in \tau \end{Bmatrix}, \tag{3}$$

Where $H \in \mathbb{R}^n$, being H the search space for the give problem, t represents time, $o: H \times T \rightarrow \mathbb{R}$ is the objective function that gives a numerical value ($o(z,t) \in \mathbb{R}$) to each possible solution ($z \in H$) at a time t and finally, $F(t)$ is the set of viable solutions $z \in F(t) \subseteq H$ at a given time t.

In less mathematical terms, a DOP is a system where the objectives or constraints can change over time, so DOPs objective is not only to find a local optimum but to track it on the solution space trough time. Evolutionary programming, or more specifically GAs, have emerged has one possible approach to DOP, but it is required to apply some enhancements to the original concept. The figure 7 represents the challenge described above, being this problem a moving optimal solution through time.

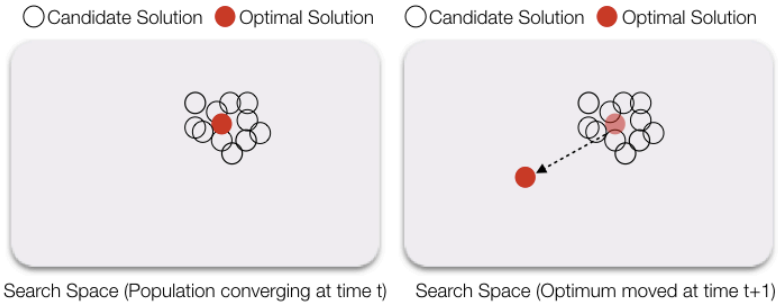

Figure 7. Moving optimal solution

2.3.2 Dynamic Genetic Algorithms

For a GA to be used as an efficient tool to solve DOPs some changes are needed. The typical approaches for DOPs' solving using GA involve modifications by manipulating memory, diversity, multi-population, adaptation and prediction.

a) Memory Approaches

The memory approach for DOP solving will store and reuse useful information to find the moving optimal solutions in our search space. In figure 8, we have an example of a cyclic DOP where the optimal solution will change cyclically. A memory approach would be useful to solve a cyclic DOP by storing all the possible states of the optimal solution trough the cycle and using that information every time there is an optimal solution change.

There are two main classes regarding memory approaches, implicit memory approach and explicit memory approach, where the first uses redundant representations and the second uses extra space to store information.

In the implicit memory approach, GAs store partial good solutions (partial chromosome structure) inside an individual to be reused later. Typical examples of an implicit memory approach are multiploidy and dominance, and dualism mechanism.

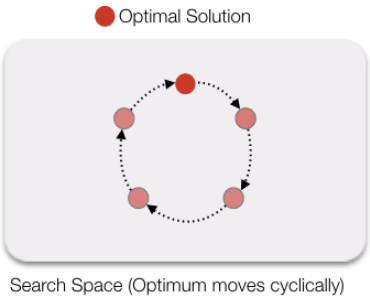

Figure 8. Cyclic DOP

In the explicit memory approaches, GAs will store, at an extra memory, fit individuals from a specific environment (search space) and makes use of them whenever this specific environment returns, or at a completely new environment. This information can be stored by two different strategies: direct memory or associative memory. For the direct memory strategy, good individuals are stored inside an external memory and reused whenever the search space changes. The associative approach stores the individual and environmental information, by finding good individuals inside the population and associating them with the current environmental condition. After finding the best individuals and associate them with the environment information, they are stored inside an auxiliary memory. Both approaches can be seen in figure 9.

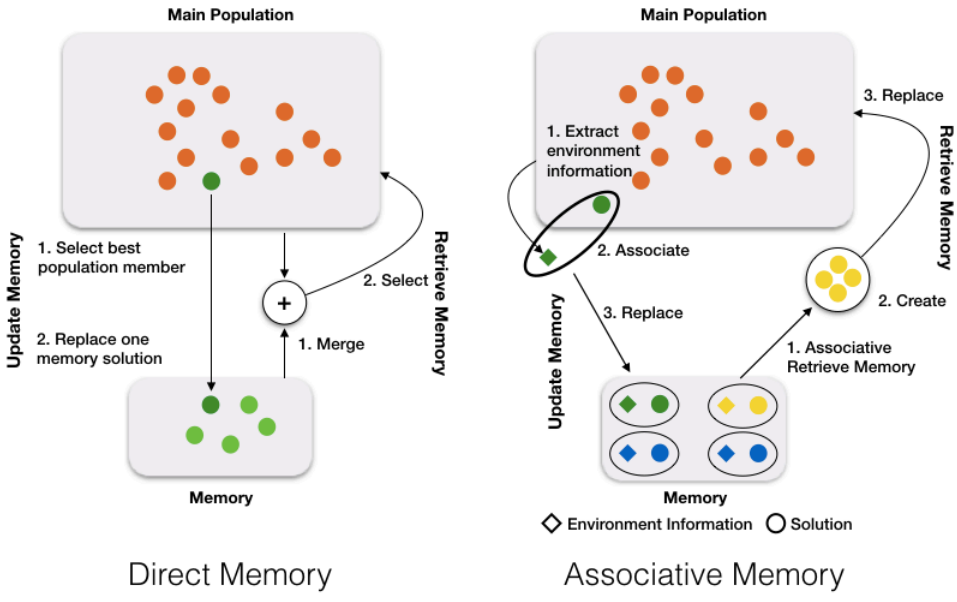

Figure 9. Direct memory vs Associative memory

b) Diversity Approaches

The genetic algorithm using a diversity approach to solve DOPs will exploit a population diversity to keep track of the moving optimum by having individuals spread across the search space. There are two main approaches: a reactive approach and a proactive approach. The reactive strategy will react to an environmental change by activating a diversity increasing mechanism. An example of this algorithm is random immigrants.

The proactive strategy maintains the population diversity every generation, despite the environmental changes that may occur. This kind of approach may imply that the selection pressure will be reduced in order to keep the diversity over the population.

The diversity approach can be implemented using random immigrants, memory-based immigrants or hybrid immigrants. All these implementations can be used in both proactive and reactive mechanisms. Figure 10 shows a representation of a random immigrant's strategy, in which each generation inserts some random individuals, called immigrants, to maintain the diversity. With this approach, if the optimum moves, all the random immigrants nearby will react and pull the population nearby.

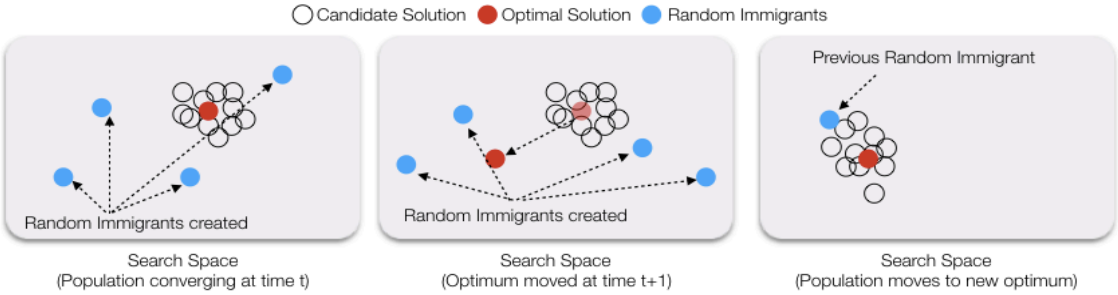

Figure 10. Random immigrants approach

In the memory-based approach, memory will be used to guide the immigrants towards the new optimum. Figure 11 shows a representation of this kind of strategy. In memory-based diversity, every generation the memory is re-evaluated, then the best memory point is chosen as the base for the new immigration wave. After, an immigrant wave will be generated by mutating the base individual with a specific probability, thus generating a new sub-population of immigrants. Finally, the worst individuals inside the main population are replaced by these memory-based immigrants.

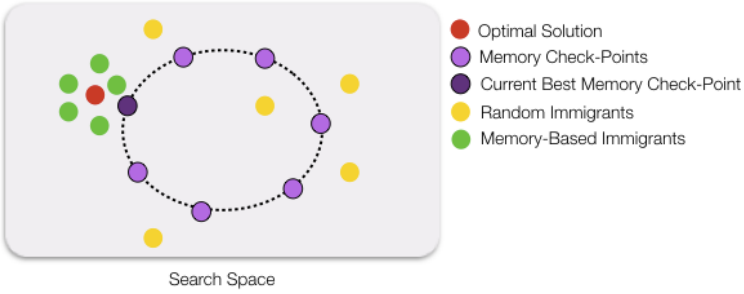

Figure 11. Memory-based vs Random Immigrants

Hybrid immigrants' strategy is a combination of several other strategies like elitism, dualism and

randomization. In this kind of approach, the immigrant sub-population is generated by a combination of these three principles, firstly the algorithm chooses the best individuals among the population and mutates them to just apply minor changes in the chromosomes, this is the elitism principle. Next happens the dualism stage in which the best chromosomes' genes are inverted and mutated, generating a sub-population of opposing solutions with significant changes. After, happens the random stage where a sub-population of random individuals is generated to address the search space changes. In the end, all these three sub-populations are combined and merged with the main population.

c) Multi-Population Approaches

This kind of strategy uses multiple populations during the run-time and defends that each population should be responsible for a specific zone, that reveals promising solutions, of the search space. This mechanism can be used using self-organizing scouts' method or shifting balance method. The self-organizing scouts' method was proposed by Branke [13] and the main population, also called parent population, will explore the search space. After, if certain conditions are met, this population is split into child populations that will search certain limited sub-zones of the search space, as showed in figure 12.

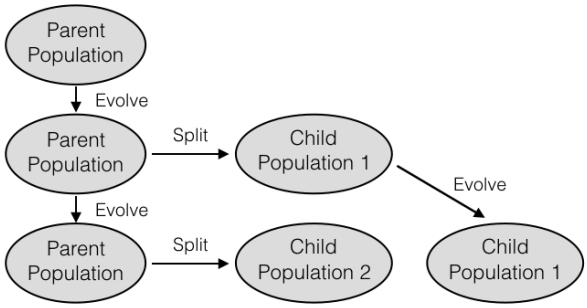

Figure 12. Self-organizing Scouts approach

Another approach using Multi-Population is called shifting balance, this kind of implementation has a core population that will have several colonies composed by different populations. The core population will exploit a specific promising area of the search space while the colonies explore the whole search space. Whenever the core population needs to shift his search to find a moving optimal point the fittest individuals of the colonies will migrate into the core population. This approach can be s

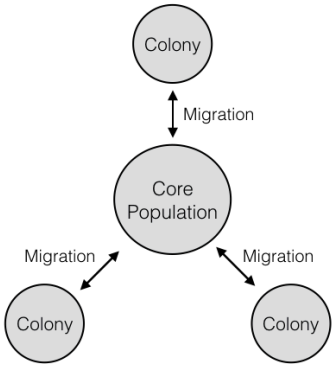

Figure 13. Shifting Balance approach

een in figure 13.

d) Adaptive Approaches

The adaptive strategies using GAs will exploit the genetic operators or parameters during the algorithm's run-time. This king of approaches has specific triggers that will activate a hyper state where certain processes will increase their pressure on the system, urging the population to change drastically.

e) Predictive Approaches

Most DOPs have observed patterns that can be predicted. This kind of problems can be implemented using predictive triggers that will predict the position of the optimal solution after a change, and the new resulting environment.

2.4 State of the Art

This chapter will give an insight on what has been developed in the field in the recent years. The first chapter will focus on approaches that used genetic algorithms for portfolio composition problems. The second chapter will target DOPs works using Evolutionary Programming (EP) and none of the approaches was directly applied to financial computation in any of the examples.

2.4.1 Works on portfolio composition using Genetic Algorithms

Works developed on stock portfolio composition have showed encouraging results by using genetic algorithms to optimize both fundamental and technical analysis of the market. The works listed below use genetic algorithms for this process, but other implementations have been suggested using other evolutionary programming algorithms, like neural networks, swarm optimization, ant and bee colonies, and many other approaches.

One of the first works involving GAs and stock portfolio composition and evaluation was made by Liad Wagman [15]. In *Stock Portfolio Evaluation: An Application of Genetic-Programming-Based Technical Analysis* developed by Liad Wagman (2003) a portfolio selection software was developed using technical analysis indicators as a strategy that generates buy and sell signals for different stocks. In this approach a population of individuals, where each individual represents a specific portfolio, will pass through the regular genetic operators like selection, crossover and mutation. One genetic operator was added called shuffle, this genetic operator will select two random stocks from an individual and randomly modifies their percentage shares, being shuffle a variation of the mutation process for a specific propose. The results were promising and the returns were 3% higher that the interest rate of that year. These promising results, with such a simple strategy, laid the foundation for portfolio composition using GAs problems.

Tomassini et al. (1995) developed a portfolio selection distributed genetic algorithm [16]. In their work, they implemented a genetic algorithm with multiple equal populations, called islands, that will exchange groups of individuals in a synchronous manner. Then, they compare their approach with a standard sequential genetic algorithm and conclude that the distributed GA will give satisfactory solutions in a much shorter time. Their approach used a multi-objective goal for the genetic algorithm, where risk should be minimized and return maximized. Both objectives were parameterized into one goal that needs a coefficient that will define the overall risk that a user wants to take. This work focus much more on the computational complexity and parallelization strategies of the problem than on the portfolio selection goal and subjacent returns. The authors state that the returns of the portfolios generated by the genetic algorithm are satisfactory but no precise values were given.

Kyong et al. (2005) focus their work on using a genetic algorithm to create a portfolio for index fund management. Index fund portfolios aim to match the performance of a chosen benchmark index like S&P 500 or Financial Times Stock Exchange 100 (FTSE 100) [17]. In this strategy, the selection of stocks used fundamental variables like standard error of the portfolio beta, average trading amount and market capitalization. The genetic algorithm applied is simple and uses the standard genetic operators and is compared with a non-optimized algorithm for different market types (bull market, bear market and flat market). The results showed that the GA optimized algorithm have a better performance over the conventional algorithm without GA optimization, being the GA optimized algorithm quite effective when market volatility has increased.

Aranha & Iba (2007) developed a portfolio selection system using a genetic algorithm that considers the modelling cost and uses a seeding and multi-objective strategy [18]. The costs associated with optimizing a specific portfolio are not considered in most approaches because only the return and risk are considered, neglecting a real-world constraint called the transaction costs associated with the buying/selling of securities in the stock exchange. To consider these costs of transaction they used an indirect approach, where, instead of discounting this cost on the overall return, they define a third goal that examine the distance between the actual solution and the desired solution by using Euclidian distances between portfolios, so the minimum distance between portfolios results in minimized transaction costs. In this approach, we have three concurrent objectives, minimize the Euclidian distance between portfolios, maximize the return and minimize the overall risk of the portfolio. To achieve this, they implemented two different fitness functions, one that minimizes the transaction costs and another that will maximize return and minimize risk called Sharpe ratio. At each generation, one of the two fitness functions can be used to evaluate the population's individuals and both have a probability of usage. The GA implementation used the usual genetic operators and added a new one called seeding stage. In the GA implementation made by the authors, the standard genetic algorithm runs several times and the previous run will pass on the best solutions (seeding stage), so they enter as immigrants into the new initial population, this procedure is repeated n times and a final population of solutions will be given as an output in the end. In the figure 14 we can see a description of this seeding method. The results obtained were compared with a regular GA implementation and showed that the seeding GA has better results when compared with a standard implementation without this feature. The returns obtained were

3.12% compared with the 1.7% obtained by the NASDAQ index in the same period. The returns on the NIKKEI for the seeding GA were like the benchmark index, authors suggested that this situation was caused by the bubble that happened on the Japanese market, leading to high volatility, thus concluding

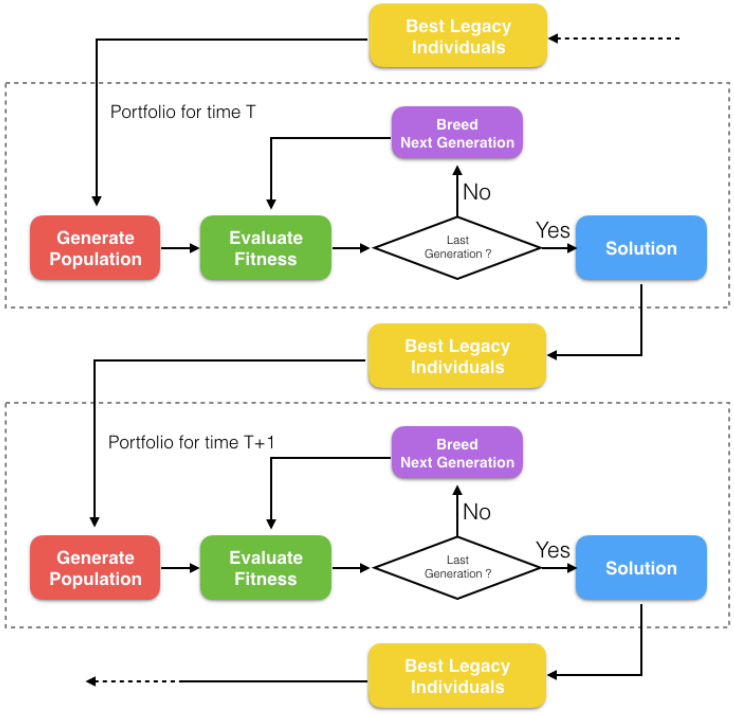

Figure 14. Description of the seeding method

that the seeding genetic algorithm has difficulty when leading with these extreme movements.

Gorgulho et al. (2009) proposed a GA that will optimize an investment strategy based on technical indicators by optimizing the weights of each one of them in the strategy for an efficient portfolio composition [19]. The GA proposed follows the standard structure and uses the standard genetic operators and a new that will mimic an artificial immune system with the objective of handling the infeasible individuals on the population that might be created by the crossover and mutation. This immune system will store the infeasible individuals inside a separated population and they will suffer the same effects of evolution as the main population. The main goal of this approach is to store relevant information that might be contained inside the infeasible individuals. This strategy was compared with other strategies like Buy & Hold (B&H) and “Random Walk”, showing far better results, with an average return of 3.46% compared with 0.49% from B&H and 0.9% from the random approach.

Simões et al. (2010) proposed a system that using GAs will formulate a technical indicators strategy, then compares the results with a B&H, Moving Averages Crossover and Moving Average Derivative strategies [20]. In this work, a standard genetic algorithm was used with the standard genetic operators. The crossover operator used was a one-point crossover and the selection procedure was a roulette-wheel selection. The fitness function used was the Sharpe ratio, being this approach multi-objective, i.e., minimizing the risk and maximizing the return. The strategy applied by Simões was a combination of Simple Moving Averages Crossover and Moving Averages Derivative, being this strategy optimized by the GA. The results were promising, presenting an average return on investment of 9.0%, thus

beating all other strategies.

Gorgulho et al. (2011) proposed a system that develop a technical analysis strategy for stock picking and portfolio composition using a GA optimization [21]. The strategy created was compared with other approaches such as B&H and “Random Walk”, as well as the benchmark index. The GA system applied used a selection system that first choose the best individuals for selection and then applied a Roulette-Wheel procedure to select the breeders. The crossover system used was a standard one-point crossover method. The mutation method applied would randomly generate a value for each gene selected inside the chromosome, although Gorgulho used an elitist approach by keeping the best individual of the population away from the mutation process. Finally, the fitness function used was the Return on Investment, being the goal of the GA to maximize this function. The results were positive, being the “Random Walk” and B&H outperformed by the GA strategy using technical indicators.

Silva et al. (2015) approach uses technical and fundamental indicators and a group of real world constraints to make the strategy closer to the real financial world [22]. The implementation proposed by Silva used a multi-objective goal, i.e., minimizing the risk and maximizing the return. Three chromosomes were developed to simulate different investment models with different real-world constraints like transaction costs, long only investments and quantity constraint for each asset inside a portfolio. This approach shown promising results and with solutions that outperformed the benchmark index S&P 500. The fundamental approach applied in this work gave a higher precision to the Multi-Objective Evolutionary Algorithm (MOEA), showing that these indicators can be important when formulating an investment strategy. The MOEA applied was a SPEA II inspired on the works of Zitzler & Thiele (1998).

Michalak et al. (2012) strategy uses a Multi-Objective optimization to find portfolios that outperform investment funds while showing an identical behaviour [23]. The crossover used was a standard single-point crossover. The mutation procedure selects a random individual and then adds a random value with uniform probability to a gene inside that chromosome. The select procedure is based on the tournament-selection approach with slight changes from the NSGA-II algorithm proposed by Deb et al. (2000) [24]. Another two genetic operators were added, called Division Mutation and Local Search operator. This approach composes portfolios of stocks and currencies. The strategy proposed is Multi-Objective, using as objectives the investment return, value-at-risk (measure of risk) and mean squared error (measure of difference between generated portfolios and investment funds actual portfolios). The proposed algorithm could find portfolios similar to investment funds’ portfolios but with higher returns and lesser risk.

2.4.2 Works on Dynamic Optimization Problems using Evolutionary Programming

Dynamic Optimization Problems have become an important research field in recent years and many works have been developed to optimize the already existing solutions. Many approaches have been proposed based on abstractions of mechanisms used by actual living beings. Many of the approaches

referred in this chapter of state of the art were already explained in chapter 2.3.2 of this work.

One typical approach for DOP using EP makes use of memory systems to track the moving optimum inside a specific search space. Memory approaches are divided into two sub-groups, implicit and explicit. Implicit approaches include examples as Multiploidy and Dominance and Dualism Mechanisms.

Inspired by diploid and multiploid organisms, Multiploidy and Dominance schemes were proposed to improve the performance of solving systems for DOPs, at first was proposed a system that used a triallelic dominance scheme to extended a simple haploid GA into a diploid GA by Goldberg & Smith (1987) [25]. After, was proposed by Ng & Wong (1995) [26] a diploid scheme with four possible combinations per gene for GA implementations. Another approach proposed three years later added a dominance scheme with additive diploid mechanism by Lewis et al. (1998) [27]. In figure 15 can be seen an example of a dominance and diploid scheme, the dominance schemes presented come from the works of Lewis and Ng & Wong.

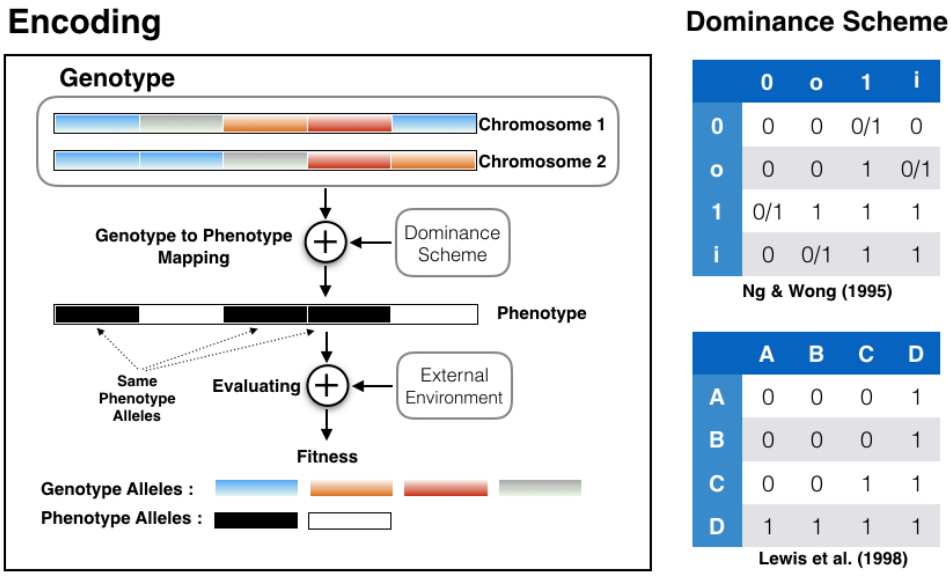

Figure 15. Diploid and Dominance GA

Another implicit memory approach is dualism, where every individual has two chromosomes (diploid), being the chromosomes the inverse of each other. Several approaches have been developed, for example, the primal-dualism genetic algorithm [28] and the dual population-based incremental learning [29].

Explicit memory approaches include direct and associative memory. The direct memory approach was proposed by Branke (1999) [13]. The associative approach named population-based incremental learning using an associative memory scheme was developed by Yang & Yao (2008) [30].

Diversity approaches for DOPs using EP includes approaches like random immigrants, memory-based immigrants [31] and finally hybrid immigrants approach [32].

Multi-Population approaches for DOPs include two main approaches, self-organizing scouts [33] and shifting balance [34].

A distinct strategy proposed is adaptive, and examples of this mechanism's implementations are hyper-mutation, hyper-selection and hyper-learning. In hyper-mutation, the mutation rate will be increased to galvanize the population diversity [35]. In hyper-selection, the selection pressure will be increased, so the population convergence will increase to specific optimum [36]. Finally, the adaptive hyper-learning strategy will raise the learning capability for the population-based incremental learning [37]. There are other implementations available to this approach, as well as combinations of multiple implementations.

Ultimately, there are the predictive approaches for DOPs solving. One of the first works developed was a Kalman-Extended GA [38] which uses a Kalman filter over the fitness values of each individual of the population. These Kalman filter will discover when is needed to create an individual or when is needed to re-classify an individual. Next, there is the Van Hemert et al. (2001) [39] strategy, which used a learning module to estimate the future state of the environment. This is implemented by using two populations, one that performs a search at the current environment and another that, by using the best individuals from the past, will predict the future optimum. Other approach by Bosman [40] combined EP with machine learning algorithms and statistical techniques to predict the future optimums for a specific environment. At last, there are the works of Simões & Costa [41] [42] that made use of linear regression strategies and Markov chains to predict changes in environments.

WORK	DATE	HEURISTIC	STRATEGY	FITNESS FUNCTION	BENCHMARKS	FINANCIAL APPLICATION	ASSETS/PERIOD	RETURNS
[15]	2003	GA	Standard Genetic Algorithm w/ shuffle operator.	6 technical analysis indicators combined, being the risk and return the fitness parameters.	Interest rate of 4.5%	Portfolio Composition Stock Market	DJIA: 1979-1980	7.5%
[16]	1995	MODistGA	Multi-population w/ synchronous immigration; Multi-objective	Lambda Trade-off Function	-	Portfolio Composition	-	-
[17]	2005	GA	Standard Genetic Algorithm	Standard Error of Portfolio Beta; Average Trading Amount; Market Capitalization	KOSPI 200 Conventional Algorithm	Portfolio Composition for Index Funds	KOSPI 200: 1999-2001	-
[18]	2007	MOGA	Multi-Objective; Seeding	Euclidian Distance; Sharpe Ratio	NASDAQ NIKKEI Standard GA	Portfolio Composition Stock Market	NIKKEI: 1998-2006 NASDAQ: 2000-2006	NIKKEI: 1.49% NASDAQ: 3.14%
[19]	2009	GA	Artificial Immune System	Return on Investment	B&H Random Walk	Portfolio Composition Stock Market	S&P 500: 2005-2007	Avg. 3.46%
[20]	2010	MOGA	Multi-Objective	Sharpe Ratio	B&H Random Walk	Portfolio Composition Stock Market	S&P 500, FTSE 100, DAX 30, NIKKEI 225: 1993-2009	Avg. 9.0%
[21]	2011	GA	Artificial Immune System	Return on Investment	B&H Random Walk	Portfolio Composition Stock Market	DJIA: 2003-2009	Best. 62.95%
[22]	2015	MOGA	Strength Pareto Evolutionary Algorithm; Multi-Objective	Return on Investment; Variance of returns in a portfolio	S&P 500	Portfolio Composition Stock Market	S&P 500: 2010-2014	Best. 50.24%
[23]	2012	MOGA	Non-dominated Sorting Genetic Algorithm; Multi-Objective; Division Mutation; Local Search	Return on Investment; Value-at-Risk; Mean Square Error	18 Polish Investment Funds	Portfolio Composition for Investment Funds	Warsaw Stock Exchange & FOREX (CHF, CZK, EUR, GBP, HUF, USD)	Positive

Table 5. State of the Art on Portfolio Composition using GAs

Chapter 3

Architecture and Implementation

This chapter is divided into two parts, the **Overall System Architecture** and the **Architecture Implementation**. The first part of this chapter discusses our architecture by explaining the overall objectives and procedures of each one of the layers that composes our system. The second part of this chapter gives a further insight about each layer by specifying what strategies were developed, which libraries were used during implementation and what information is used as input and output.

3.1 Overall System Architecture

The main objective of this sub-chapter is to present the overall architecture plan that was developed during this work. For a better understanding, our architecture was separated into different layers to give a better intuition about the overall problem and all the working parts and their relations. The proposed architecture used was distributed into the *Mining Layer*, *Database Layer*, *Genetic Algorithm Layer* and *Validation Layer*. In the figure 16 is demonstrated the overall architecture plan divided into well-defined layers.

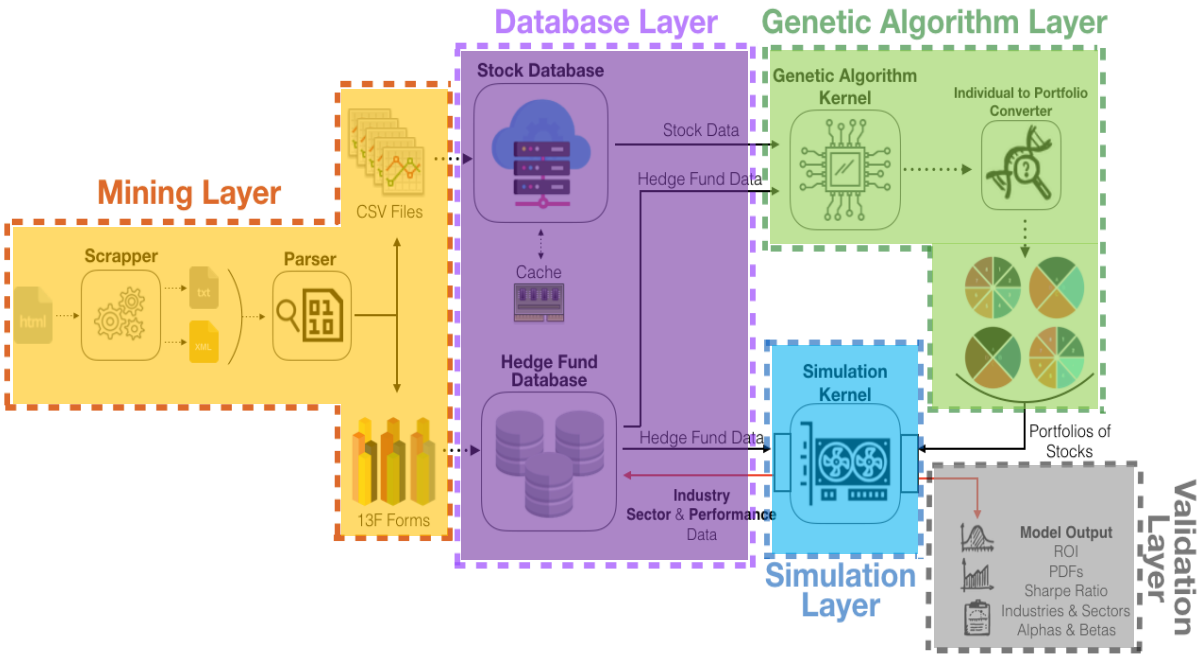

Figure 16. Overall System Architecture

To achieve a better level of implementation a layer abstraction was deployed. In more general terms, the end-to-end description of this system starts in the *Mining Layer*, where different internet sources are compressed into useable data and promptly stored at the designated file system inside the *Database Layer*. Some data about the hedge fund's performance needs to be calculated, being the *Simulation Layer* the responsible for this computation, returning all the data back to the *Database Layer* for storage. After achieving all the pre-processing necessary, our data is sent accordingly for the desired optimization into the *Genetic Algorithm Layer.* At this stage, the genetic kernel will receive both stock and hedge fund data to start the optimization process, where a population of portfolios of stocks is optimized, based by a set of rules, in an incremental manner for several pre-defined iterations until the best performing portion is proposed as solution for the problem at hand. The proposed solutions are tested, once again at the *Simulation Layer*, and all the outputted results sent into the *Validation Layer*, where the solutions are evaluated and the overall system performance is addressed by a set of benchmarks.

The layer abstraction paradigm will improve the overall performance of each layer by assigning specific objectives to each one, that way, all the code refactoring and optimization problem will be easier and faster. In the following sub-chapters (3.1.1–3.1.5) we will give a better insight about each layer architecture.

3.1.1 Mining Layer

The Mining Layer is responsible for the data acquisition mechanism, this mechanism uses internet servers as sources for data like 13F Forms or stock prices. This process of getting all the raw data via requests to remote servers in the internet is made by a *Scrapper*, able to get the index of all formularies and immediately starts getting all the raw information using simple requests. Then, all the data is stored into a temporary place for further pre-processing.

The temporary raw data is then sent to the *Parser*, able to read the downloaded information and returns the relevant data for our study in a new format. This format can be read by the libraries later described in the implementation at chapter 3.2.1. All the parsed data is sent into the *Database Layer* and promptly stored.

3.1.2 Database Layer

The Database Layer is divided into two separated branches, each one of them representing different kinds of data.

The first branch is responsible for the storage and handling of all stock prices for usage. When the optimization process is called, a new stock database instance is launched, this instance is responsible for handling all data, using specific data structures, for the *Genetic Algorithm Layer*, *Simulation Layer* and *Validation Layer*. Since the analysis focus on 4000 different stocks, each one of them with daily data since 1998, a memory and time efficient system is needed. For this propose a caching mechanism was applied, being the needed stocks kept in memory, thus occupying less memory by not allocating them all, and providing a faster access by making less requests to the disk by exploitation of the faster Random Access Memory. The stock database also provides some functionalities to help implementation given an input stock, such as, intra-day price variations, return on investment during a specific interval and standard deviations of the intra-day price variations.

The second branch is responsible for the management of the hedge funds data. The data about the different hedge funds needs to be handled by a specific instance that is created every time an optimization procedure is called in the *Genetic Algorithm Layer*. Before being launched, the hedge funds database instance needs to know which hedges funds to load in order to operate. This information must be defined by the user for the portfolio optimization. This branch is responsible to provide each hedge fund popularity and each stock sector and industry popularity among each hedge fund (these parameters are explained in the table 7 from sub-chapter 3.2.3a). All the parameters returned by this instance are based in the 13F Forms information and need to be calculated beforehand at the *Simulation Layer*.

This layer is then composed by two different systems, one is responsible to handle all the stock data and another is responsible to handle the hedge fund data. Both systems have separated file systems to store the database needed with the information organized.

3.1.3 Genetic Algorithm Layer

The *Genetic Algorithm Layer* is responsible for the generation of groups of portfolios based on data about the stock prices and the hedge funds' own portfolios. In sum, this layer main goal is optimization, in our context, portfolio optimization by selecting the best possible stocks with the best possible weight. The genetic algorithm kernel is where the optimization happens, recurring to strategies as the ones discussed in the state of the art.

This layer has a Genetic Algorithm Kernel module where the optimization phase truly happens. This kernel is composed by a population of portfolios, each portfolio is represented by an array of floats, each float might represent a specific stock or a specific rule that specifies each stock weight. After running this Genetic Algorithm Kernel, the best portfolios among the population are returned. The best portfolios, after the optimization process, are then converted to a format that can be analysed by our *Simulation Layer*.

The strategies applied and all other details about our Genetic Algorithm will be explained at chapter 3.2.3 from Architecture Implementation.

3.1.4 Simulation Algorithm Layer

The *Simulation Layer* is responsible for the simulation of portfolios of stocks to determine their performance and other relevant data like sector or industry allocation. This layer can be divided into two distinct branches, the Hedge Fund Simulator and the Model Portfolio's Simulator.

The Hedge Fund Simulator is responsible to simulate each hedge fund portfolio and then calculates relevant information that will be used during our optimization phase, so this branch is responsible for all the pre-processing needed for the 13F Forms. After this simulation, all the data is sent back to the *Database Layer.*

The second branch, called Model Portfolio's Simulator and is responsible for the simulation of the portfolios returned by our *Genetic Algorithm Layer*. After simulating each portfolio for a defined interval of time, all the performance data from each portfolio is sent to the *Validation Layer* for further processing.

3.1.5 Validation Layer

The *Validation Layer* is responsible for our model performance estimation. This layer receives, from the *Simulation Layer,* the performance metrics achieved by our model portfolios' during simulation, and calculates a group of performance indicators suited for our problem, such as return on investment or *Sharpe Ratio*. This final layer will be able to generate the output needed to evaluate our model's overall

performance and compares this information with other benchmark values.

3.2 Architecture Implementation

The *Architecture Implementation* section will portray the implementation developed in detail, during this thesis, for every layer that composes this architecture.

All the programs developed for this thesis used Python programming language with version 3.5 [44] and Spyder [45] as an *Integrated Development Environment* for Python.

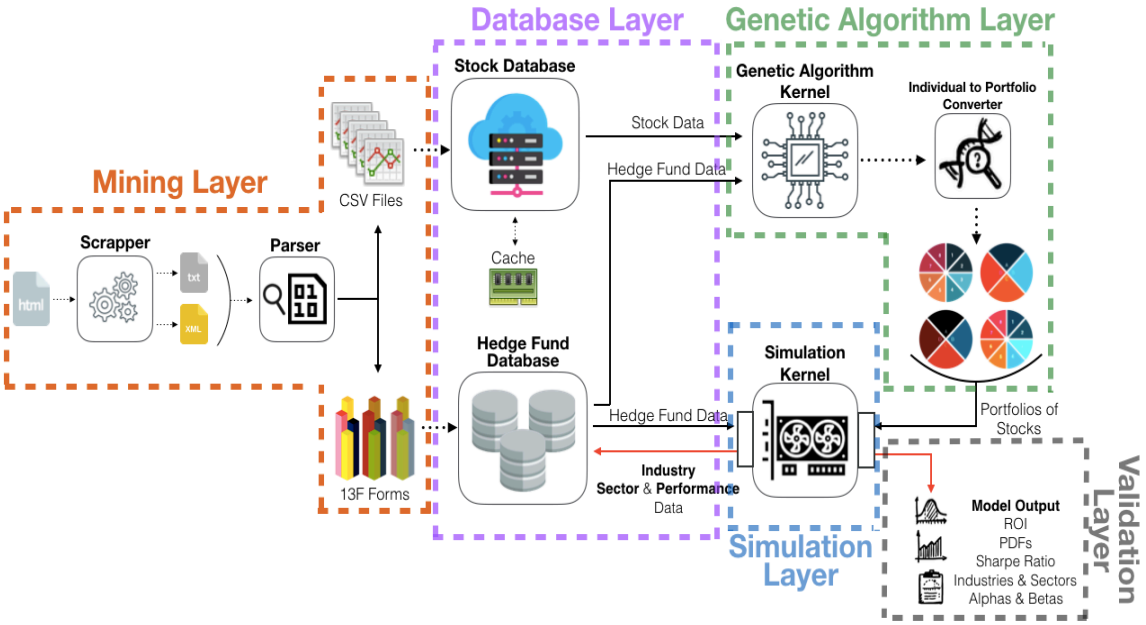

Figure 17. Specified Overall System Architecture

3.2.1 Mining Layer Implementation

The *Mining Layer* handles two distinct types of data, the stock data and the hedge fund data. Both data come from different internet sources, thus having different formats and different parsers.

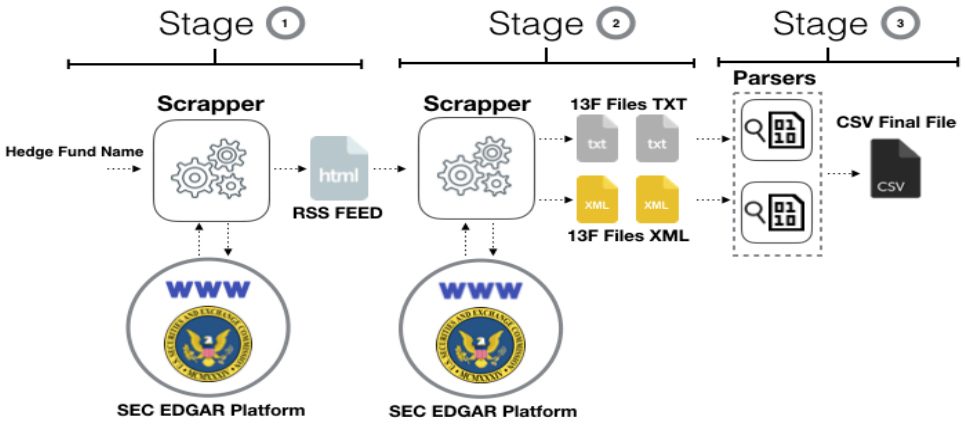

Figure 18. Mining Layer for Hedge Fund data

a) Hedge Fund Data

The Hedge Fund data is composed by the 13F formularies of each hedge fund considered. Each year has four 13F formularies and each one of these formularies represents a specific quarter of a specific hedge fund, in sum each 13F formulary represents a hedge fund portfolio of stocks. The information gathered starts at the fourth quarter of 1998 and ends at the first quarter of 2016.

For the Hedge Fund data acquisition, this system uses a *Scrapper* that exploits the Edgar Platform [46] made available by the *U.S. Securities and Exchange Commission*. This *Scrapper* starts by searching for a specific Hedge Fund, if the Hedge Fund is found, then a RSS Feed file is downloaded. This process can be seen at the **stage 1** of the figure 18.

The RSS Feed uses the *Extensible Markup Language* (XML) format and contains a list of links for every 13F submission made by that Hedge Fund. After acquiring this list with all 13F formularies links, the *Scrapper* module starts requesting to the Edgar Platform each one of the files listed. After this step, we have in our local machine every 13F formulary ready for the *Parser*. The *Parser* will be responsible to convert the recently acquired 13F data into a more suitable format for further usage. The 13F formularies are composed by two files, the first is the header of the submission and contains information about the hedge fund, relevant dates and other information, such as the hedge fund manager name. The second file is the hedge fund asset list that represents that specific quarter portfolio. The 13F formularies, depending on their date, can have two different formats, XML or Text (TXT). The TXT format is present in all the files that came before 2013 and XML is used in all the files that were released after 2013. This

step can be seen at the **stage 2** of the figure 18.

XML File

```
<informationTable xmlns="http://www.sec.gov/edgar/document/thirteenf/informationtable">
    <infoTable>
      <nameOfIssuer>ALPHABET INC</nameOfIssuer>
      <titleOfClass>CAP STK CL C</titleOfClass>
      <cusip>02079K107</cusip>
      <value>8927</value>
      <shrsOrPrnAmt>
        <sshPrnamt>11983</sshPrnamt>
        <sshPrnamtType>SH</sshPrnamtType>
      </shrsOrPrnAmt>
      <investmentDiscretion>SOLE</investmentDiscretion>
      <votingAuthority>
        <Sole>11983</Sole>
        <Shared>0</Shared>
        <None>0</None>
      </votingAuthority>
    </infoTable>
    <infoTable>
      <nameOfIssuer>AMAZON COM INC</nameOfIssuer>
      <titleOfClass>COM</titleOfClass>
      <cusip>023135106</cusip>
      <value>59364</value>
      <shrsOrPrnAmt>
        <sshPrnamt>100000</sshPrnamt>
        <sshPrnamtType>SH</sshPrnamtType>
      </shrsOrPrnAmt>
      <investmentDiscretion>SOLE</investmentDiscretion>
      <votingAuthority>
        <Sole>100000</Sole>
        <Shared>0</Shared>
        <None>0</None>
      </votingAuthority>
    </infoTable>
</informationTable>
```

TXT File

```
<TABLE>
                                SCGE MANAGEMENT, LP
                    Form 13F INFORMATION TABLE AS OF DECEMBER 31, 2011

                                                                                                VOTING AUTHORITY
                                                VALUE X  SHARES/  SH/  PUT/  INVSTMT  OTHER
NAME OF ISSUER                  TITLE OF CLASS  CUSIP    $1000)  PRN AMT  PRN  CALL  DISCRETN  MANAGERS  SOLE    SHARED  NONE
--------------                  --------------  -----    ------  -------  ---  ----  --------  --------  ----    ------  ----

<S>                             <C>             <C>      <C>     <C>      <C>  <C>   <C>       <C>       <C>     <C>     <C>
21VIANET GROUP INC              Sponsored ADR   90138A103  6,545   715,300 SH       SOLE                715,300      0    0
ACME PACKET                     COM             004764106 13,137   425,000 SH       SOLE                425,000      0    0
ARCHIPELAGO LEARNING INC        COM             03956P102 11,266 1,165,000 SH       SOLE              1,165,000      0    0
EZCHIP SEMICONDUCTOR LTD        ORD             M4146Y108 16,998   600,000 SH       SOLE                600,000      0    0
GOOGLE INC                      CLA A           38259P508 20,992    32,500 SH       SOLE                 32,500      0    0
HISOFT TECHNOLOGY INTERNATIONAL LTD SPONSORED ADR 43358R108 12,448 1,348,600 SH       SOLE              1,348,600      0    0
HOMEAWAY INC                    COM             43739Q100  2,858   122,937 SH       SOLE                122,937      0    0
ISOFTSTONE HOLDINGS LTD         SPONSORED ADR   46489B108 15,922 1,819,700 SH       SOLE              1,819,700      0    0
LINKEDIN CORP                   COM CL A        53578A108  7,876   125,000 SH       SOLE                125,000      0    0
NETAPP INC                      COM             64110D104 12,695   350,000 SH       SOLE                350,000      0    0
PEGASYSTEMS INC                 COM             705573103  8,820   300,000 SH       SOLE                300,000      0    0
QUALCOMM INC                    COM             747525103 13,675   250,000 SH       SOLE                250,000      0    0
RIVERBED TECH INC               COM             768573107 12,338   525,000 SH       SOLE                525,000      0    0
RPX CORP                        COM             74972G103  6,325   500,000 SH       SOLE                500,000      0    0
TAL EDUCATION GROUP             ADS REPSTG COM  874080104  6,162   618,064 SH       SOLE                618,064      0    0
UNIVERSAL DISPLAY CORP          COM             913477105 11,007   300,000 SH       SOLE                300,000      0    0
VERISIGN INC                    COM             92345Y106 25,004   700,000 SH       SOLE                700,000      0    0
YANDEX NV                       SHS CLASS A     N97284108  9,228   468,410 SH       SOLE                468,410      0    0
YOUKU.COM INC                   SPONSORED ADR   98742U100  3,291   210,000 SH       SOLE                210,000      0    0
</TABLE>
```

Figure 19. Input Samples in XML and TXT formats

The situation described above shows that two different *sub-Parsers* are needed, one for the TXT format and another for the XML format, and both return the same output format, being this output sent as a *Comma Separated Values* (CSV) file to the *Database Layer* after parsing. The input files carry several parameters, being only some of them relevant for our study. In our context, the parameters required are *Title of Class*, *CUSIP, Value* and *Shares*, and two examples for each format can be seen above in the figure 19. The table 6 below enumerates the input parameters used by our system. This step can be seen at the **stage 3** of the figure 18.

As said above, the output file is a CSV with five features and fully represent each 13F formulary in the following steps of our architecture. Figure 20 shows the file format for all 13F formularies in CSV. The *CUSIP*, *VALUE* and *SHARES* is kept for safety in further calculations, the other two features are *TICKER*, which represents the stock, and finally *P_WEIGHT*, which represents the weight of an asset in the overall portfolio and must be a real value between 0 and 1.

```
ticker,cusip,value,shares,p_weight
QTM,747906204,5625,500000,0.08551881413911061
WM,94106L109,28744,2100000,0.43700494108703913
THC,88033G100,19181,825000,0.2916153553781832
VTR,92276F100,7496,2262800,0.11396427213987077
PKG,695156109,4729,450400,0.07189661725579627
```

Figure 20. Output format for CSV files

A Python library called *BeatifulSoup* [47] was used to process the RSS Feed File, and both TXT and XML files from each 13F formulary.

Table 6. Valuation Rule Parameters

INPUT FILE PARAMETER	DESCRIPTION	USAGE
TITLE OF CLASS	Represents the class of the security owned by the Hedge Fund, they can be stocks, short positions, etc.	This parameter is used to filter only stock assets.
CUSIP	CUSIP means Committee on Uniform Securities Identification Procedures, and it is formed by a unique set of nine characters that represent a specific company and type of title.	The CUSIP is used to get the Ticker (Stock identifier) for that asset.
VALUE	Value in thousands of dollars for a specific security.	The value is used to calculate the portfolio weight by dividing the asset value by the total assets value.
SHARES	*Total number of shares owned for a specific security.*	This value is stored for safety reasons, i.e., might be used if necessary for further calculations.

b) Stock Data

The stock data is composed by a set of files, each one of them represents a specific stock price daily since December of 1998 until May of 2016.

All the data acquired was retrieved from the Yahoo Finance Platform [48] by using a custom library made for Python called *Yahoo-finance* [49]. This library, given a specific ticker and a time interval, returns all the daily prices between that interval in a *Python Dictionary* (which is a basic data type in Python). This dictionary is then saved as CSV file, being each line the stock price at a given day.

This module is called for every stock ticker present in our 13F formularies and returns all the stocks that exists in the Yahoo Platform and sends everything to the *Database Layer*.

3.2.2 Database Layer Implementation

The *Database Layer* is divided into two separate categories, the *Stock Database* and the *Hedge Fund* Database. Both databases were implemented as *Classes* and can be instantiated by any other layer at runtime if needed. These databases provide a group of behaviours that uses the data and returns specific computations needed for the *Optimization Layer*.

Both databases are prepared for concurrency because both are used for reading activities only (after the pre-processing), thus they can be instantiated multiple times and used by different layers. This way our data becomes immutable and can be accessed by multiple threads without the need to implement a locking system to manage the access for a parallel computation paradigm. Both databases used *Pandas* library [50] to load all CSV files into *DataFrames* for Python routines. The *DataFrame* data structure is an optimized implementation, thus providing faster access to our data in RAM memory, more

reliability when compared to other Python basic types, such as *Lists* or *Dictionaries* and provides a set of built-in methods for data manipulation and selection.

a) Stock Database

This database is responsible for handling all the stock price data. Considering that our system uses 4000 stocks, being each stock represented by a *Pandas DataFrame*, when using a regular computer, we would expect our memory to be unable to handle this much data in RAM. To solve this problem a caching mechanism was implemented to achieve a better running performance by micro-managing the stock data that gets loaded/cleared into RAM.

During our computation at the *Optimization Layer*, the GA sometimes needs other information related to the stock prices, such as:

- Stock's return on investment and standard deviation for each day for a given interval of time;
- Stock's overall return on investment for a given interval of time.

To address both behaviours, above described, a group of methods was developed, thus returning any information about the stock prices that might be requested.

b) Hedge Fund Database

This database is responsible for handling all the Hedge Fund data that is relevant for our study. Taking into consideration that our system uses information about 29 different Hedge Funds with multiple 13F formularies each one of them, thus requiring a custom data structure. The data structure implemented within this *Database Class* is a *Dictionary* (Python basic type) of hedge funds, where each hedge fund has his own *Dictionary* of formularies, being each 13F formulary a *Pandas*' *DataFrame* data structure.

During the optimization process that happens at the GA Kernel, other relevant information about hedge fund's assets is required, like:

- **Stock popularity** among hedge funds;
- **Stock's sector popularity** among hedge funds;
- **Stock's industry popularity** among hedge funds;
- **Hedge fund popularity** associated to a specific stock.
- **Stock exposure variation** associated to a specific stock and two consecutive 13F formularies.

These five parameters need to be calculated before any optimization starts, being the hedge fund data sent into the *Simulation Layer* for that purpose. After the pre-processing gets accomplished, all the data is returned and promptly stored and will be returned, during optimization, if requested. All parameters are further explained at the next chapter (3.2.3.1).

3.2.3 Genetic Algorithm Layer Implementation

The *Genetic Algorithm Layer* is responsible for the generation of a set of possible solutions based on hedge fund information and past stock prices that are returned by the *Database Layer*. For the Genetic Algorithm Kernel implementation, a Python library called *DEAP* [51] was used, this library enables concurrency, enhances our algorithm robustness and allows us to use out-of-box implementations of different GA strategies. Before explaining all the constituent parts of the developed Genetic Algorithm, it is important to list all the strategies implemented in this work. The table 7 bellow shows all the approaches developed.

Table 7. Strategies Applied in the Genetic Algorithm

METHOD	TYPE OF APPROACH	STRATEGY
SELECTION	Stationary	• Tournament Selection • Elitist Selection
MUTATION	Stationary	• Tournament Selection • Elitist Selection
CROSSOVER	Stationary	• Single-Point Crossover
DIVERSITY	Dynamic	• Elitist Immigrants • Random Immigrants
ADAPTIVE	Dynamic	• Hyper-Mutation • Hyper-Selection
MEMORY	Dynamic	• Direct Memory
TEMPORAL	Dynamic	• Temporal Seeding

a) Chromosome Structure

The chromosome can be considered the basic unit of our Genetic Algorithm, because it represents a possible solution inside our population and the function parameters that we want to optimize. Taking this into consideration, becomes empirical that our chromosome structure needs to be well-defined, to

Figure 21. Chromosome Representation

suitably describe our optimization problem.

Each individual has its own chromosome, and each chromosome is represented by a *List* (Python basic type) of floats (genes), being each float between 0 and 1. When the Genetic Algorithm starts its optimization process, each individual is initialized by giving random set of values to each gene. The chromosome representation can be seen in the figure 21.

The chromosome structure is inspired in a regular portfolio, thus having a set of stocks and for each stock a specific weight. The above chromosome follows this paradigm and is divided into two sub-sets, the first set represents which stocks to pick for our portfolio (red box in figure 21) and another is responsible for the *Valuation Rule* (green boxes in figure 21) that defines the weight of each individual stock based on hedge fund information. In the end, both problems are optimized by a defined evaluation function and a set of stocks and the corresponding weights of each one of them is returned.

Before explaining how the *Valuation Rule* maps the weights for each stock, it is necessary to understand each one of the five members that compose this rule. The *Valuation Rule* uses as input five different parameters, such as, *Stock Popularity* (StP), *Sector Popularity* (SeP), *Industry Popularity* (IP), *Hedge Fund Popularity* (HFP) and *Stock Exposure Variation* (SEV). The *Valuation Rule* parameters are explained in the table 8.

Table 8. Valuation Rule Parameters

VALUATION RULE PARAMETER	DESCRIPTION	FORMULAE
STOCK POPULARITY	Average weight of a specific stock in all the HF's portfolios where exists.	$\mathrm{StP} = \frac{1}{N}\sum_{i=0}^{N} HF_{i_{weight}}(stock)$
SECTOR POPULARITY	Average weight of a specific stock sector in all the HF's portfolios.	$\mathrm{SeP} = \frac{1}{N}\sum_{i=0}^{N} HF_{i_{sector}}(stock)$
INDUSTRY POPULARITY	Average weight of a specific stock industry in all the HF's portfolios.	$\mathrm{IP} = \frac{1}{N}\sum_{i=0}^{N} HF_{i_{industry}}(stock)$
HEDGE FUND POPULARITY	Each HF has a Popularity parameter between 0 and 1 and its related to each HF performance. The best HF has a value of 1 and the worst a value of 0, all other HFs use values inside this interval according to their performance rank. The HF Popularity parameter is the average value of all HF popularities for the HFs where the stock exists.	$\mathrm{HFP} = \frac{1}{N}\sum_{i=0}^{N} HF_{i_{performance}}$
STOCK EXPOSURE VARIATION	Average variation of a specific stock weight between two consecutive quarters for the HF portfolios where that stock exists.	$SEV = \frac{1}{N}\sum_{i=0}^{N} \Delta HF_{i_{weight}}(stock, Quarter_t, Quarter_{t-1})$

The chromosome structure implemented tries to find a specific set of stocks and weights for the *Valuation Rule* that maximizes a fitness function. In this context, the chromosome presented is the genotype of the individual and the final portfolio with every weight defined is the phenotype for this individual.

The above statement raises the following question: *"How do we get a phenotype from a genotype like our chromosome?"*. Firstly, the stocks picked are quite easy to convert, each one of them is a value from 0 to 1 that is mapped to an integer from 0 to 4000, each integer represents a stock ID that is transformed into a ticker. For example, a *Stock gene* has a value of 0.1 represents the ID 432, then converted to the corresponding ticker, for instance, *GOOGL*. The final gene, called *Number of Stocks*, limits the maximum number of stocks in our portfolio and can vary from 0 to 1, then mapped into an interval from 8 to 12. So, in summary, the stock genes first need to be converted to their corresponding ticker, and the number of stocks is defined by the last gene, called *Number of Stocks*.

Secondly, the stocks weights must be defined, and each weight is based in the *Valuation Rule* optimized during our GA. The Valuation Rule can be defined by:

$$\mathrm{VR(stock)} = \mathrm{w_{StP}}\times\mathrm{StP(stock)} + \mathrm{w_{SeP}}\times\mathrm{SeP(stock)} + \mathrm{w_{IP}}\times\mathrm{IP(stock)} + \mathrm{w_{HFP}}\times\mathrm{HFP(stock)} + \mathrm{w_{SEV}}\times\mathrm{SEV(stock)} \qquad (4)$$

Where $\mathrm{w_{StP}}$, $\mathrm{w_{SeP}}$, $\mathrm{w_{IP}}$, $\mathrm{w_{HFP}}$ and $\mathrm{w_{SEV}}$ are defined from 0 to 1, all these parameters are optimized during our GA and come directly from our possible solution chromosome. The functions StP, SeP, IP, HFP and SEV receive a specific stock and return a value from 0 to 1 based on the Hedge Fund data. These functions are calculated during the pre-processing and are stored inside the *Hedge Fund Database*. So, the *Valuation Rule* will return a value from 0 to 1 for each stock considered, then all the *Valuation Rule* values obtained are normalized and finally each stock weight is derived. The stocks weights calculated must respect the following conditions:

$$\begin{cases} 0 < w_i < 1, \;\; i \in 0,1,\dots,N \\ \sum_{0}^{N} w_i = 1 \end{cases} \qquad (5)$$

Where N is the number of stocks considered, and is defined by the last gene of the chromosome.

After the two conversion processes above described, a phenotype solution emerges from a different genotype representation in the form of a regular portfolio that is composed by a set of stocks $\{stock_0, \dots, stock_N\}$ and a set of stock weights $\{w_0, \dots, w_N\}$, both ready to be studied in matters of performance.

b) Fitness Function

The fitness function implemented is called *Sharpe Ratio* and is defined by the following equation:

$$\mathrm{S_r} = \frac{\overline{r_p} - r_f}{\sigma_p} \qquad (6)$$

Where, $\overline{r_p}$ is the mean portfolio return, r_f is the return of a risk-free investment and σ_p is the standard-deviation of between all stocks contained in the portfolio. The risk-free investment considered for this

work were the U.S.A. treasury bonds yield, that is widely considered a safe asset with a return on investment value of 0.01.

The *Sharpe Ratio* evaluates both risk and return, being the main goal to maximize the return on investment and minimize the risk associated. This ratio is used to specify how much excess of return you are receiving for the higher volatility endured for holding riskier stocks. The *Sharpe Ratio* works as a multi-objective fitness function, where the return on investment must be maximized and the risk minimized, thus generating portfolios with positive returns and stocks with low volatility, i.e., semi-steady growth.

The *Sharpe Ratio* can return any real number, but values higher than one are usually considered good, a value between 0 and 1 is considered satisfactory and all the negative values are considered unsatisfactory, because it means the portfolio is losing money.

c) Genetic Algorithm Operators

As explained before, every Genetic Algorithm implementation needs to support three different operators, such as *Selection, Crossover* and *Mutation*. The three operators used are explained bellow.

c.1) Selection

The model developed in this thesis context uses two different selection methods for two non-independent sub-sets of the population. In the chapter 3.2.3.6, after all the GA parts are well-described, an overall structure of the GA Kernel is showed for a better intuition. Two distinct GA selection operators were used, the *Tournament Selection* and the *Elitist Selection*.

The *Tournament Selection* will randomly group every individual into groups of 3, inside each sub-group the individuals will compete against each other in terms of fitness and the fittest of each sub-set survives, thus passing its traits to the next generation.

The Elitist Selection will select a defined portion of the fittest individuals from a population and always passes them to the next generation. This selection method was implemented to use *Elitist Immigrants*, passing the best solutions to the next generation after some mutation. This strategy is explained later in the *Dynamic Enhancement* sub-chapter (3.2.3 d).

c.2) Crossover

The crossover method used was the *Single-point* crossover. This method selects a point at a random location at both parents, then each parent contributes with one part of the chromosome to generate two new individuals that represent an offspring between both parents. This method can be better understood in chapter 2.2.2 c1.

The *Two-point* crossover was discarded because it would generate an offspring with individuals composed by 3 different segments, thus mixing our individuals in a way that would inflict the convergence of our model.

c.3) Mutation

The population of individuals has a chance to be mutated in every generation of our GA, if an individual is chosen for mutation a sub-set of genes will be randomly picked using a uniform distribution since every gene has equal probability of being mutated.

If one of those picked genes is a stock gene, then a uniform distribution is used to pick a value from 0 to 1, if the picked gene is a *Valuation Rule* gene then a Gaussian distribution is called with the gene own value as mean and standard deviation of 0.2, thus returning values which will be 99.9% of the times between three standard deviations of the mean.

```
for each Gene in individual:
    if Random Probability <= Mutation Probability:
        if Gene is Stock Gene:
            Gene = new Gene from Uniform Distribution
        if Gene is Valuation Rule Gene:
            Gene = new Gene from Gaussian Distribution
```

Figure 22. Pseudo-code of the Mutation Operator

The stock genes use a uniform distribution because our objective is to randomly pick a new stock and every stock must have an equal probability of being picked. The *Valuation Rule* genes use a Gaussian distribution because our objective is to pick a new value between 0 and 1 that has a high probability to be close to the past value, this method doesn't destroy the past information kept by the gene because it'll produce a similar value with high likelihood, this way our genes will try a new solution that only has a slight variation, thus optimizing the *Valuation Rule* genes in an incremental manner. The figure 22 shows the mutation procedure pseudo-code for our individuals.

d) Dynamic Enhancement

Being the task at hand composed by a DOP, to enhance our GA performance some Dynamic Genetic Algorithm's strategies were used to explore memory, diversity and adaption in our GA. In this work five different strategies were applied and are the following:

- **Random Immigrants** – The Random Immigrants are randomly generated individuals that join our population every generation. In this kind of procedure, the worst 30% solutions in the population are replaced by randomly generated individuals called Random Immigrants. This proactive method enhances the diversity of our population every generation by doing a global search of the search space.
- **Elitist Immigrants** – The Elitist Immigrants are individuals based on the best individuals of the past generation. The approach used selects the best individuals among the population, then it applies mutation over them and they are injected into the next generation. This strategy is useful to maintain good solutions while exploring the rest of the search space.
- **Hyper-Mutation** – This dynamic procedure exploits the mutation probability to introduce more

diversity in the overall population. If the fitness of the best individual in the overall population remains constant for 3 generations, then this procedure is activated by doubling the mutation probability for another 3 generations. This strategy enhances our GA adaptability by improving the population diversity, i.e., this mechanism can detect if the overall population shows a high density of individuals close to local optimums (low-diversity) and by increasing the mutation probability tries to find new solutions in the search space by improving the population diversity.

- **Hyper-Selection** – This method exploits the tournament size of the *Tournament Selection* to create a higher selection pressure. This method is activated after the Hyper-Mutation state for a single generation, because by increasing our selection pressure we want a rise in the overall quality of the potential solutions. This method is used after the Hyper-Mutation procedure to fine-tune the recently discovered solutions by over-mutation.
- **Direct Memory** – The Direct Memory approach uses an auxiliary memory to store individuals based on their fitness for later use. In this strategy, the best individuals of each generation are selected and if some individual, already inside the memory, has a lower fitness, then its replaced by a new one from the present population. In the end, we have a memory with the best individuals among all the generations, ready to be injected in the next temporal iteration of the *Temporal Seeding* process that is explained in the next sub-chapter 3.2.3. e.

e) Temporal Seeding

The *Temporal Seeding* approach implemented for this thesis is based on the work developed by Aranha & Iba [18]. This technique splits the time-series dataset into even parts, then each part is optimized by a different GA that uses the best individuals, from the previous time-series chunk, as immigrants into the new initial population. This procedure is repeated for every time-series spilt made, in order. This process was explained in the chapter 2.4.1 and the figure 14 is provided for further intuition. In our problem context, a year dataset is divided into four different quarters, each quarter is optimized by a different GA that uses the best solutions of the past quarter (*Seeding*) on the initial population.

Citing Iba & Aranha [18], *"In practice this technique generates a bias in the evolutionary search towards the region of the search space that contained the solution in the previous time period."*, so the seeding immigrants might have a higher fitness when compared to the newly generated individuals, thus they will have a higher probability of passing their traits into the next generation and influences the population to keep searching in the area of the search space related to the best individuals of the previous execution. According to the same study cited above, the number of individuals that are transferred to the next generation doesn't really influence the performance of this technique, being influenced only by the existence or not of Seeding Immigrants.

f) Overall Kernel Architecture

Now that all the parts of the *Genetic Algorithm Layer* were explained, the overall architecture of his GA Kernel must be explained. The structure of the GA Kernel can be seen in figure 23 and receives a year as input, this year is then divided into 4 equal parts, each one of them represents a specific quarter.

Each quarter is optimized by a GA that receives the *Seeding Immigrants* of the previous quarter as input and outputs the top 10 individuals to be used as input in the next quarter. The first quarter starts with no

Seeding Immigrants, being the population fully random.

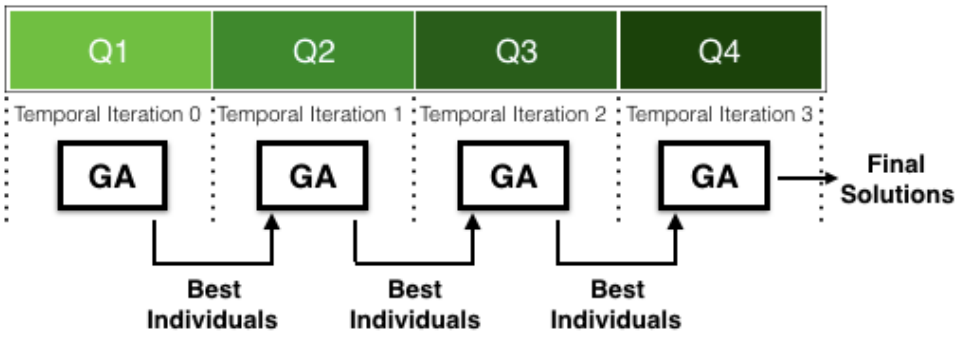

Figure 23. Genetic Algorithm Kernel Architecture

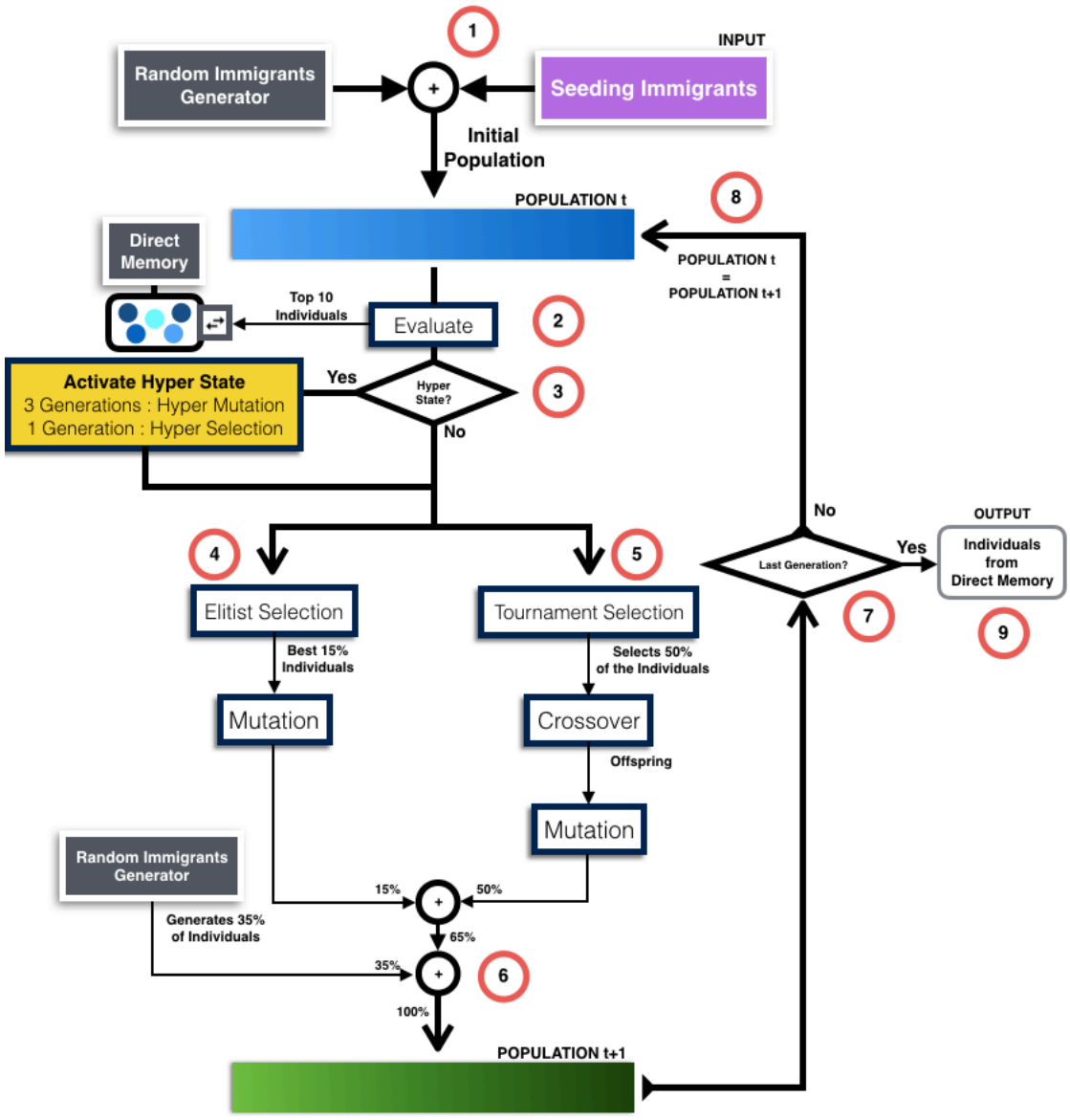

Figure 24. Implemented Genetic Algorithm Flowchart

All the GA modules present in the figure 23 follow the same architecture, but each GA focus on a different time period. For a better intuition, the figure 24 flowchart was divided into several steps. Each step accomplishes the following:

1. This is the first step of our GA, in this procedure the *Seeding Immigrants* are processed and inserted into the initial population with a set of randomly generated individuals. If no *Seeding Immigrants* are given, then the initial population is totally randomly generated.

2. The population at iteration *t* is evaluated in terms of their *Sharpe Ratio*, then the best 10 individuals are sent to the Direct Memory. If the Direct Memory is empty, they are automatically stored. If the Direct Memory already has 10 individuals from the previous generation, then we merge both groups and the best 10 are kept while the other 10 are discarded.

3. The third step is a conditional state, if the best fitness among the population has remained the same for 3 generations then the Hyper State is activated. This state doubles the mutation probability of every mutation operator for 3 consecutive generations. After this 3 generations, the mutation probability is scaled back to normal and for another generation the tournament size is doubled, thus inciting the selection pressure. In more general terms, this Hyper State is responsible for the growth of diversity to search new areas of the search space, after this process, our population must converge to the local optimums found, achieving this by increasing the selection pressure.

4. The step 4 is responsible for the creation of the *Elitist Immigrants* that will be part of the next generation. First, through *Elitist Selection*, the best 15% individuals of the overall population are selected and then mutated. By the end of this procedure the *Elitist Immigrants* sub-set is ready to be included into the next generation.

5. This step will generate an offspring sub-set based on the current population. From the overall population of individuals only 50% will be selected to pass its traits to the next generation. The surviving half of the population will perform crossover and mutation, thus generating a final offspring that will join the population of the next generation.

6. The sixth step of our flowchart starts by generating a set of *Random Immigrants* for the next generation. These newly formed random individuals are then merged with the *Elitist Immigrants* of the step 4 and the offspring generated individuals from step 5. In the end a new population of individuals is formed based on the previous population. The population for *t+1* is composed by 15% *Elitist Immigrants*, 35% *Random Immigrants* and 50% offspring individuals based on parents from the population *t*.

7. This step is a conditional one, if the actual generation is equal to the number of pre-defined generations, then our GA accomplished the optimization process for the given number of generations and goes directly to the step 9. If the generation is not the last the step 8 is reached.

8. At step 8, the old population is switched by the newly formed population, i.e. the population *t* becomes the population *t+1* and the optimization procedure continues through steps 2,3,4,5 and 6 until the step 7 condition is fulfilled.

9. Finally, the step 9 is reached, meaning that our optimization problem was accomplished, thus returning the individuals present inside our *Direct Memory* as output. The output individuals are the best solutions present during all the GA generations. These individuals might be used as a final output of our model or as *Seeding Immigrants* to the next temporal iteration seen in the figure 23.

g) Implementation Enhancements

Considering that our GA model is quite computational expensive, some implementation strategies were applied to increase the computational performance. After analysing the implicit computational complexity of each part of our GA, it was concluded that the bottleneck was the fitness calculation for each solution in the population. The implementation enhancement made use of only the first part of the *MapReduce* paradigm [52] and it was used to optimize the fitness bottleneck, i.e. the fitness calculation for each individual in the population.

The *MapReduce* is a programming model that distributes a specific computational job to several workers. This approach is divided into two distinct tasks, the first is called *Map* and is responsible for splitting the overall task into sub-tasks and maps each one of these segments to a specific worker. The *Reduce* phase devises some function that will process each sub-task and output a value.

Our problem only uses the *Map* stage, in this stage the population is divided into equal sub-sets of individuals, each sub-set is then assigned to a different worker. The worker is responsible to calculate the fitness score of each individual by using the *Sharpe Ratio* as processing function. In the end, each sub-set of individuals is returned and merged, thus concluding the fitness calculation process with a speedup of approximately the number of workers assigned to the task. The implementation automatically detects how many workers are available and uses the number of workers minus one as the number of available computational nodes (workers). The following figure illustrates the process described.

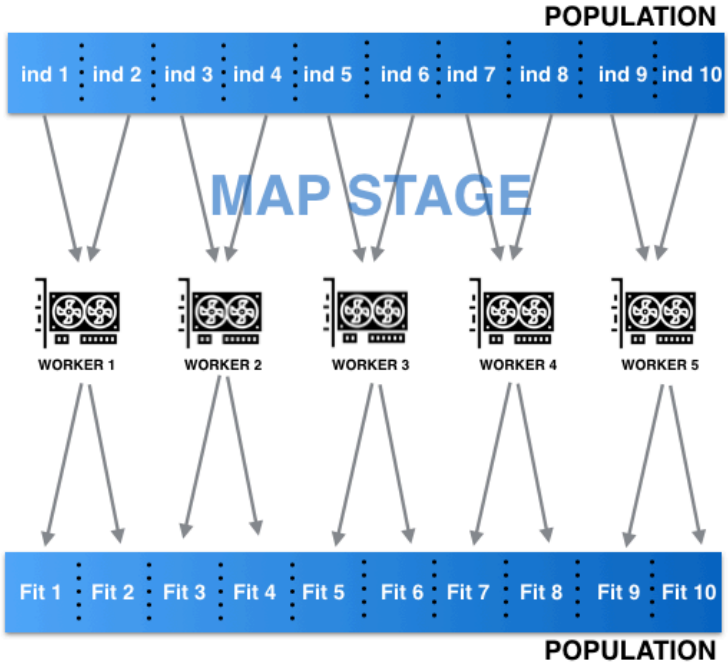

Figure 25. MapReduce for Fitness Calculation

3.2.4 Simulation and Validation Layer Implementation

The *Simulation Layer* is responsible for the Hedge Fund Simulator and the Model Portfolio's Simulator. Both simulators were implemented as standard *Python Class* and for all calculations needed the *Numpy* library was used [53]. The *Validation Layer* was implemented as a standard *Python Class*, using *Numpy* library [53] for calculations and *Matplotlib* library [54] for all the plotting needed for the analysis of the model performance.

Chapter 4

System Validation

The System Validation chapter explains and discusses different case studies, designed to test the solution proposed in this thesis. Each case study is analysed by a set of evaluation metrics and confronted with different benchmark performances.

4.1 Evaluation Metrics

To successfully address the quality of our approaches, they must be evaluated by mutual evaluation metrics. The performance metrics used in this system, to compare the quality of distinct approaches, were the *Sharpe Ratio* and *Return on Investment*.

4.1.1 Sharpe Ratio

As explained before, the *Sharpe Ratio* is a performance measure used to address a portfolio profitability and associated risk. In simple terms, the *Sharpe Ratio* measures a portfolio's risk-adjusted returns, i.e. for each unit of risk taken how much return was accomplished. The *Sharpe Ratio* is defined by the following equation:

$$Sr = \frac{\overline{r_p} - r_f}{\sigma_p} \qquad (7)$$

Where, $\overline{r_p}$ is the mean portfolio return, r_f is the return of a risk-free investment and σ_p is the standard-deviation of between all stocks contained in the portfolio.

This measure was used since it relates the return of an investment with the associated risk, thus being a better performance measure when compared to approaches that only use the return or volatility to estimate some portfolio performance.

4.1.2 Return on Investment

Return on Investment is a widely-used metric to evaluate an asset or group of assets performance because of its simplicity and adaptability. This performance indicator measures the efficiency of an asset by evaluating his return relative to the asset acquisition cost. The ROI expression is defined by the following equation:

$$ROI = \frac{r_{asset} - c_{asset}}{c_{asset}} \qquad (8)$$

Where r_{asset} is the return of the asset and c_{asset} is the cost of acquisition for that same asset. This expression can return any real number. A negative value means an asset has a lower value than when it was acquired and a positive value means an increase in valuation when compared to the original cost.

4.2 Case Studies

4.2.1 Methodologies

This thesis results will be compared with two benchmark indexes. The first index will be the Standard & Poor's Index, a benchmark that tracks the 500 biggest companies in the United States of America. The second benchmark index is going to be the overall return of all hedge funds' portfolios combined. Hedge funds became a famous investment vehicle in contemporary finance, being known by their risky and profitable trades with some of them always beating the market with all-star managers and top analyst. The proposed solution expects to mimic the overall returns of all hedge funds, win and losses combined, meaning that the strategy developed by the system can rival high-ranking hedge fund managers. The most optimistic results that this work can crave for would be outperform both benchmark indexes, proving to be a solid investment strategy for financial institution that usually invest in financial assets.

The case studies were divided into three distinct parts, each one designed to validate the proposed solution in different terms. The three case studies created are:

- Case Study A – The hedge funds are tested to check whether they are profitable entities or not and an approximation of the average return of all hedge funds is proposed.

- Case Study B – The portfolio composition problem using HF information is tested with two different approaches of Genetic Algorithm, one static and another dynamic, to study whether the dynamic approach is able to enhance a Static GA.

- Case Study C – The proposed system is exhaustively tested in different quarters to study its behaviour in other kinds of market to validate the model as an investment tool.

4.2.2 Case Study A – Hedge Fund Profitability and Whale Index Estimation

This case study is divided into two different objectives, **Hedge Fund profitability** and **Whale Index estimation**.

The study about Hedge Fund profitability, i.e. the returns on investment of each Hedge Fund, was done to validate our Hedge Fund Dataset as a reliable source of information and to prove that a model based on this data might be able to profit as much or more, depending on the optimization degree achieved.

In chapter 2.1.3 figure 1 showed the Whale Index, which follows all the S&P 500 stocks owned by leading hedge fund managers and compares it with S&P 500 Index. Citing the Whale Index creators: *"The Whale Index tracks the highest-conviction stocks held by leading hedge fund managers. It is a long-only index that tracks the 100 most commonly held stocks disclosed on the quarterly 13F regulatory filings of consistently successful managers. The index is equal-weighted and rebalanced on a quarterly*

basis 46 days after the end of each quarter." [55]. Thus, becomes obvious that Whale Index would be a great benchmark for a model in our thesis context, but since it's not publicly available an approximation for this index must be found. So, the main goal of the Whale Index estimation segment is to create an approximate index that effectively reflects the overall Hedge Fund performance.

a) Hedge Fund Profitability Analysis

Our first goal is to successfully analyze all Hedge Funds in terms of *Return on Investment* and then look at the average return of all the HF combined. The following simulation was done using 29 different HFs and only the top 5 performers are showed. The parameters used in the simulation are presented in the table 9. The period of study is composed by 5 different years, from 2010 to 2015 and all the results are compared to the S&P500 index.

Table 9. Case Study A Configuration

PARAMETERS	VALUE
STUDY PERIOD	01/11/2010 – 01/11/2015
NUMBER OF HF	29
BENCHMARK	S&P500

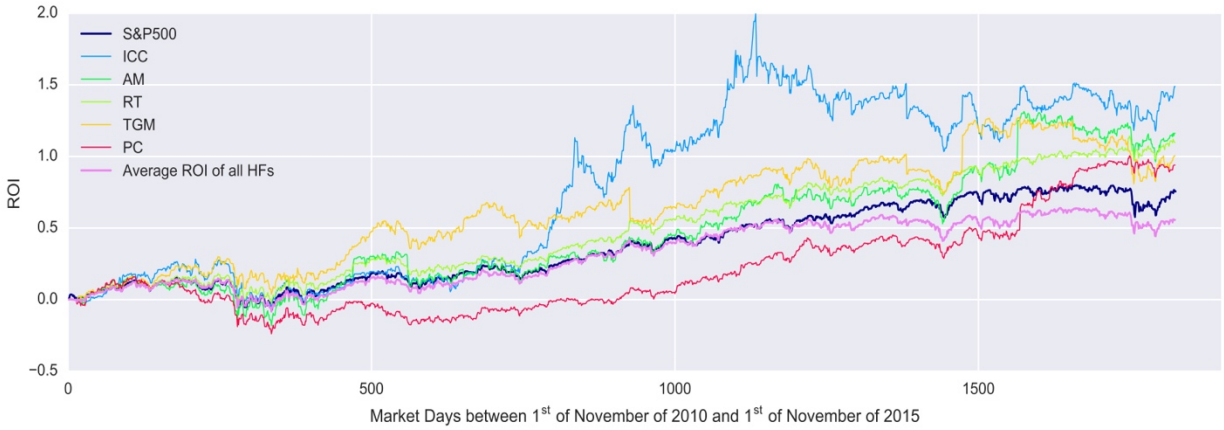

Figure 26. Profitability analysis of the top 5 performing Hedge Funds vs Average ROI of all HFs vs S&P 500

Figure 26, presents the *Return on Investment* of the best 5 Hedge Funds, as well as our benchmark S&P500 and the average *Return on Investment* of all 29 HFs.

During the chosen 5-year period the top 5 HF showed an outstanding performance, beating the S&P500 (dark blue line) and the average of all the HFs combined. The best HF (light blue line), in terms of ROI, considered was the famous Icahn Carl C (ICC) with a ROI of approximately 150% during the 5-year period.

The average ROI of all Hedge Funds (pink line) was approximately the same as the S&P500 index (dark blue line), until the latter half of 2013 is reached and the overall HF performance drops to values which are always lower than our S&P500 benchmark. The average ROI of all Hedge Funds is always correlated with the S&P500 benchmark, as expected given the diversity of stocks held by all the HFs combined, thus being able to successfully track the S&P500 index.

This study showed that some HFs presented a better performance when compared to the S&P500 index and that the overall HF ROI is correlated with this index. The overall HF ROI was the same as the S&P500 index until the latter half of 2013 is reached and the performance drops.

b) Whale Index Estimation

Since the Whale Index is not publicly available an approximated version was needed. The original Whale Index is calculated using only the 100 most common stocks and using different HFs each quarter. Our proposed approximation is the average ROI for all the HF considered in our thesis. By doing this, our approximation is calculated using a fixed set of 29 HFs and considering all stocks held. The figure bellow shows our Whale Index Approximation (WIA) vs the S&P500 index.

Figure 27. Whale Index Approximation

As expected, the WIA estimated shows high correlation with the S&P500, which is expected by analysing the original Whale Index shown in the figure 1. The WIA is outperformed by the S&P500, which was unexpected when compared with the original Whale Index (who beats the S&P500). This can be explained because the original Whale Index changes the hedge fund set based on their performances and only considers the 100 biggest stock investments, thus achieving better results.

Since the goal is to compare the proposed solution performance with the average performance achieved by all the hedge funds considered, the WIA is presented as a better alternative by being a better estimate of the real hedge fund performance, while the Whale Index might be considered more of a strategy. The WIA was chosen since it follows a theory supported by some of the biggest experts in the HF market, that suggests hedge funds as a whole are starting to fail against the S&P500 benchmark [56].

4.2.3 Case Study B – Static Genetic Algorithm vs Dynamic Genetic Algorithm

In order to deliver a successful solution, it is necessary to better understand Genetic Algorithms in our problem context. To achieve the above stated, case study B was designed. In this case study, the main goal is to compare a static and a dynamic approach in terms of performance to the problem at hand. This case study is divided into three different stages, the first studies the results of the Static Genetic Algorithm, then the second discusses the results of the Dynamic Genetic Algorithm and the final part compares them both in terms of performance for the task defined. The configurations for this case study can be seen at table 10.

Table 10. Case Study B Configuration

PARAMETERS	VALUE
MARKET	Stock Market (4000 stocks)
TRAINING PERIOD	12/02/2015 – 12/02/2016
TEST PERIOD	12/02/2016 – 10/05/2016
NUMBER OF EXECUTIONS	100
NUMBER OF GENERATIONS	10
POPULATION SIZE	100

Since the stock market is considered a dynamic system, it is expected that the dynamic approach outperforms the static one. This case study is directly related to one of the main objectives of this thesis, which is to study Dynamic approaches in Genetic Algorithms to enhance the performance for dynamic optimization problems.

Table 11. Static and Dynamic GA configurations

OPERATORS	STATIC GA	DYNAMIC GA
SELECTION	Tournament Selection	Tournament Selection
MUTATION	Gaussian + Uniform Function	Gaussian + Uniform Function
CROSSOVER	Single Point	Single Point
ELITISM IMMIGRANTS	No	Yes
RANDOM IMMIGRANTS	No	Yes
HYPER-MUTATION	No	Yes
HYPER-SELECTION	No	Yes
DIRECT MEMORY	Yes	Yes
TEMPORAL SEEDING	No	Yes

This case study is going to compare two different Genetic Algorithm topologies, the first is the standard Genetic Algorithm (static approach) and the second is a Dynamic Genetic Algorithm. Table 11 shows the topological differences between both approaches.

a) Static Genetic Algorithm Approach

The following sub-section characterize our static approach of the GA to be compared with a dynamic approach later in this work. The main goal is to build up a vanilla Genetic Algorithm with the standard operators as shown in table 11. The vanilla GA will be executed without any dynamic enhancements and respecting the rules defined in table 10.

Table 12. Static Genetic Algorithm Configuration

OPERATORS	METHOD	HYPERPARAMETERS
SELECTION	Tournament	**Tournament size:** 3
MUTATION	Gaussian + Uniform	**Mutation Rate:** 20% **Gaussian:** • Mean: Gene Value; • Std. Deviation: 0.2
CROSSOVER	Single Point	**Cut-Point:** Randomly selected.
DIRECT MEMORY	Yes	**Top 10** individuals.

The table 12 above specifies how each operator was implemented and what hyperparameters were defined for this specific approach. The **selection** and **crossover** operators are completely standard and are well explain at the chapter 2 of this work.

Each individual has a probability of 20% of passing through the **mutation** operator. The mutation operator implemented supports two different sub-mutation procedures, one for the stock genes and other for the *Valuation Rule* genes. For the stock genes, a mutation by uniform distribution was implemented and for the *Valuation Rule* genes, a Gaussian distribution mutation was implemented, using the gene value as the mean and 0.2 of standard deviation. Each gene has a 20% probability of being mutated and during a mutation operation more than one gene can be modified. If the Gaussian distribution returns a value higher than one, then one is assumed, the same happens for values lower than zero. For further information check the sub-chapter 3.2.3 c3. The **Direct Memory** was used to save the best solutions of all generations, like a *Hall of Fame*, being the best 10% individuals returned in the end and evaluated during test phase.

The Static Genetic Algorithm creates a set of solutions based on the HF and stock information from 12/02/2015 to 12/02/2016, being this period the training period of our model. The best 10% solutions are stored inside the Direct Memory and are returned in the end for the test phase, which will evaluate the performance of each solution from 12/02/2016 to 10/05/2016. The Static GA was tested on 100 executions, which returned 1000 possible portfolios as solutions for the task at hand. The 1000 proposed solutions are evaluated during the test phase and two histograms were created, the first histogram

shows the 1000 solutions density in terms of *Return on Investment* and the second histogram shows the same in terms of *Sharpe Ratio*. To get an approximation of the true distribution plotted in figures 28 and 29, a *Kernel Density Estimation* (KDE) was used in both histograms. In simple terms, KDE is a non-parametric tool to estimate the probability density function of some random variable of interest. The

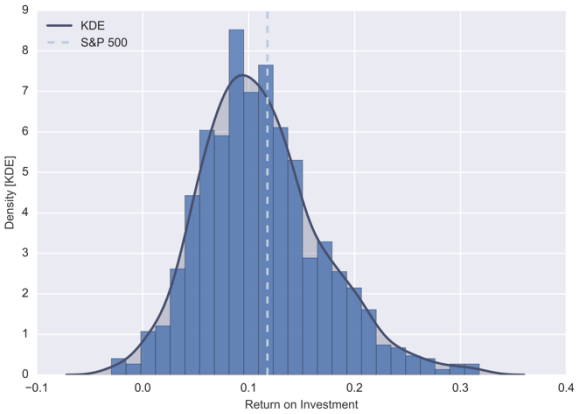

Figure 28. Histogram of ROI for the Static GA

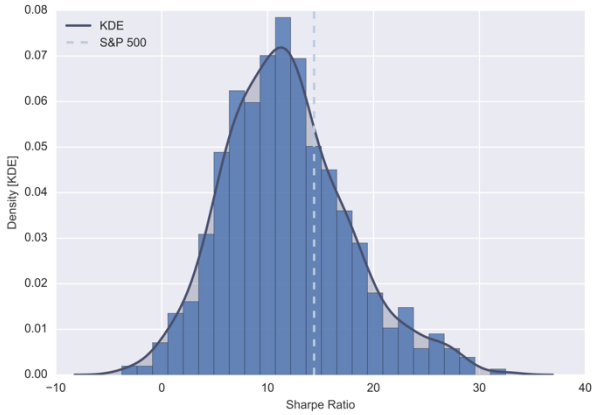

Figure 29. Histogram of Sharpe Ratio for the Static GA

implementation of KDE used is part of the *Seaborn Python* library [57].

The figure 28 shows a histogram of the *Return on Investment* for all the Static GA's proposed portfolios and the S&P 500 ROI on the same test period, which is 0.1178. As one can see, the ROI of the Static GA follows a Normal Distribution with mean 0.1119 and standard deviation 0.0573, which tells us that 68% of our model's ROI will be within one standard deviation from the mean, i.e., 68% of the portfolios proposed by the static model will be between a ROI of 0.0546 and 0.1693.

The figure 29 shows a histogram of the *Sharpe Ratio*'s for all the Static GA's proposed portfolios and the S&P 500 *Sharpe Ratio* in the test period, which is 14.3905. Looking at the figure 29, it's apparent that the *Sharpe Ratio* of the portfolios proposed by our system can be approximated to a Normal Distribution with mean 11.7658 and standard deviation 5.8752. Being our results approximated to a Normal Distribution, 68% of the portfolios returned by the Static GA will be between 5.8905 and 17.6411

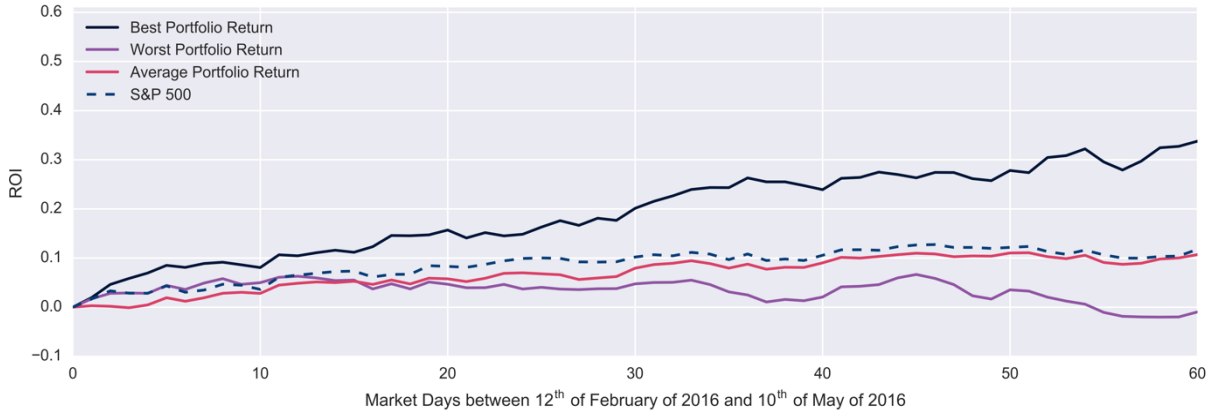

Figure 30. ROI progression for the Best, Worst and Average portfolios from the Static GA

in terms of *Sharpe Ratio*.

Figure 30 shows the evolution of the *Return on Investment* during the test period for the Best and Worst portfolios and the average of all the portfolios proposed by the Static Genetic Algorithm. During the test stage, the average of all portfolios shows a similar behaviour to the S&P500 index. The best portfolio had a ROI of 0.3378 while the worst one had a ROI of -0.0096, and the average of all portfolios had a performance of 0.1119. All the results were summarized in the table 13.

Table 13. Static GA Results

		AVERAGE SOLUTION	BEST SOLUTION	WORST SOLUTION
STATIC GA MODEL	**ROI**	0.1119	0.3378	-0.0096
	Sharpe Ratio	11.7658	32.4844	-3.7733
S&P 500	**ROI**	0.1178		
	Sharpe Ratio	14.3905		

The results obtained after testing the Static GA's 1000 portfolios were quite positive, showing an output with low volatility and similar to our benchmark index (SP500) in matters of *Return on Investment*. In terms of *Sharpe Ratio*, the average solution returned by the static model is worse when compared to the S&P 500, thus showing that in average the static portfolios are riskier than the benchmark index. Although the Static Genetic Algorithm is an elementary topology, the results were promising by showing that on average the static GA will converge to solutions which can successfully track the S&P 500 index using a maximum of 12 stocks against the 500 selected by the index. The extreme portfolios, i.e. worst and best, showed that the range of ROI for a portfolio is frequently positive and does not lose money for the investor.

The static approach proves to be a good solution for the task at hand by successfully track, on average, the S&P 500 index, which is better than the Whale Index Approximation, thus showing that the static solution can outperform the average of all Hedge Funds combined. The Vanilla Genetic Algorithm is presented as a good baseline for the optimization problem and a dynamic approach might achieve better results by understanding the underlying dynamics of the stock market system.

b) Dynamic Genetic Algorithm Approach

The sub-section here presented will analyse the performance of the GA in our portfolio composition problem when using dynamic enhancements. The goal is to create a Dynamic Genetic Algorithm with the operators presented in table 14 and test the results in terms of *Return on Investment* and *Sharpe Ratio*. The following test will respect the constraints defined above in table 10.

Table 14. Dynamic Genetic Algorithm Configuration

PARAMETERS	METHOD	HYPERPARAMETERS
SELECTION	Tournament	**Tournament size:** 3
MUTATION	Gaussian + Uniform	**Mutation Rate:** 20% **Gaussian:** • Mean: Gene Value; • Std. Deviation: 0.2
CROSSOVER	Single Point	**Cut-Point:** Randomly selected. **Crossover Probability:** 40%
DIRECT MEMORY	-	**Top 10** individuals.
ELITISM IMMIGRANTS	-	**Top 15%** individuals.
RANDOM IMMIGRANTS	-	**35%** of the next generation individuals.
HYPER-MUTATION	Gaussian + Uniform	**Mutation Rate:** 40% **Gaussian:** • Mean: Gene Value; • Std. Deviation: 0.2
HYPER-SELECTION	Tournament	**Tournament size:** 6
TEMPORAL SEEDING	-	**Number of Temporal Splits:** 4

The dynamic approach engineered for this case study is the main model developed for the thesis and all the operators involved were explained thoroughly on chapter 3.2.3. The Dynamic GA is trained with information about the stocks and HF from 12/02/2015 to 12/02/2016 and the best 10% individuals were tested from 12/02/2016 to 10/05/2016. The Dynamic GA was executed 100 times and 1000 portfolios of stocks were returned as potential solutions for the task at hand. After testing all portfolios in terms of *Return on Investment* and *Sharpe Ratio*, two histograms were made using KDE as explained in the previous sub-chapter 4.2.3. a. The histograms presented in the figures 31 and 32 show the density of portfolios in terms of ROI and *Sharpe Ratio*.

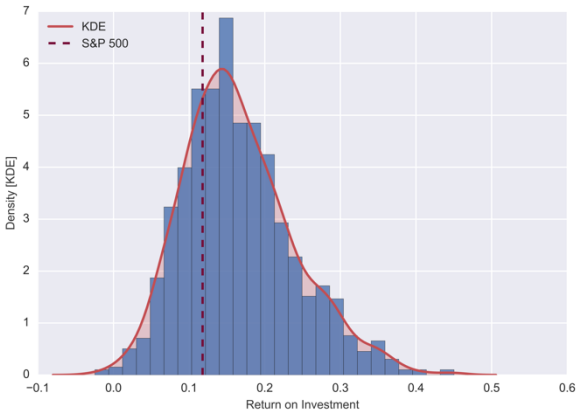

Figure 31. Histogram of ROI for the Dynamic GA

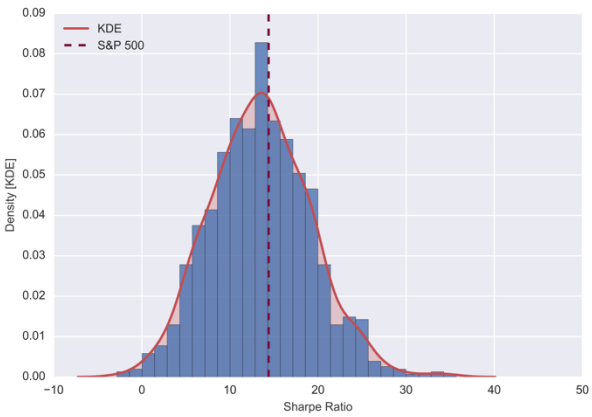

Figure 32. Histogram of Sharpe Ratio for the Dynamic GA

The figure 31 presents a histogram of the *Return on Investment* for all the Dynamic GA's proposed

portfolios and the S&P 500 ROI of 0.1178 on the same test period. Accordingly, the ROI of the Dynamic GA comply with a Normal Distribution with mean 0.1654 and standard deviation 0.0729, which tells us that 68% of our model's ROI will be within one standard deviation from the mean, i.e., 68% of the portfolios proposed by the dynamic model will be between a ROI of 0.0924 and 0.2384.

The figure 32 demonstrates a histogram of the *Sharpe Ratio* for all the Dynamic GA's proposed portfolios and the S&P 500 *Sharpe Ratio* of 14.3905 on the same test period. As shown, the *Sharpe Ratio* of the Dynamic GA is approximately a Normal Distribution with mean 13.5048 and standard deviation 5.6526. Thus, 68% of our model's *Sharpe Ratio* will be within one standard deviation from the mean, i.e., 68% of the portfolios proposed by the dynamic GA will be between a *Sharpe Ratio* of 7.8522 and 19.1575.

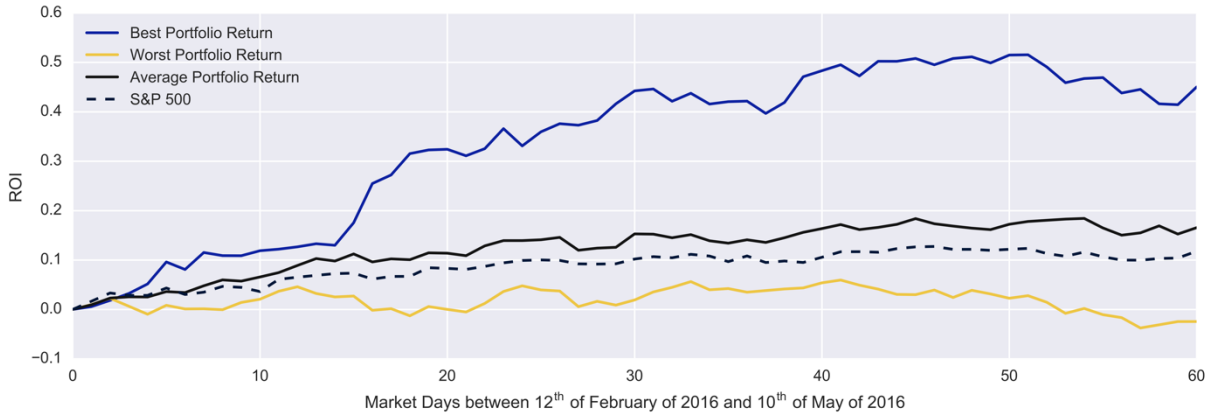

Figure 33. ROI progression for the Best, Worst and Average portfolios from the Dynamic GA

The figure 33 presents the variation of the *Return on Investment*, at the test phase defined, for the best, worst and average portfolios returned by our dynamic approach during this case study. After looking at the figure above, becomes clear that the average portfolio ROI outperforms the benchmark index while following the same volatility. The best and worst portfolios had a *Return on Investment* of 0.1654 and -0.0245, respectively, while the average of all portfolios combined had a performance of 0.1654.

All the results obtained were summarized in the following table 15.

Table 15. Dynamic GA Results

		AVERAGE SOLUTION	BEST SOLUTION	WORST SOLUTION
DYNAMIC GA MODEL	**ROI**	0.1654	0.4502	-0.0245
	Sharpe Ratio	13.5048	35.6875	-2.8302
S&P 500	**ROI**	0.1178		
	Sharpe Ratio	14.3905		

The results obtained for the dynamic strategy were positive since our average portfolio surpasses the benchmark index defined for this case study (SP500). The results obtained showed that an approach

using dynamic strategies is well suited for the stock market, since it can successfully beat the benchmark index for the test period in terms of *Return on Investment* while maintaining, on average, the same *Sharpe Ratio*, thus showing that the proposed model maintains the same risk as the index with higher returns.

The dynamic approaches implemented enhanced the system performance, mainly in terms of memory, diversity and adaptability.

In terms of **memory**, our approach proven to be valuable preserving good solutions from all generations, thus protecting unique individuals. The above statement is relevant when using a system that triggers noise like the one implemented, by means of hyper-mutation for example, which sometimes is necessary to explore completely new segments of the overall search-space. Another memory enhancer would be the temporal seeding, which gives more importance to more recent quarters while preserving optimized genes from previous runs in previous quarters.

Regarding **adaptability**, the proposed system improved by applying a set of hyper operators that will increase specific hyperparameters to enhance the system's response to a dynamic system. In other words, the hyper-mutation operator will increase the search pressure by exploring the search space more intensely via increased mutation probability and after, the resulting population will be affected by an extra selective pressure to fine tune the newly discovered solutions in the sample space.

In matters of **diversity**, the dynamic strategies applied showed an improvement in terms of gene diversity. By applying a hyper state to the population of individuals, the individuals will increase its diversity, thus exploring the search space more efficiently and finding new local optimums. The random immigrants and hyper state mechanisms are responsible for injecting noise to the system in order to stimulate the diversity among the population thus achieving a better search of the sample space.

The dynamic strategies applied to this model effectively raise the performance of the GA by considering the time-variant behaviour of the stock market and hedge funds. The dynamic GA presented successfully beats the SP500 index in the test period defined in terms of profitability, which is a promising result for the topology developed. Although, in terms of risk by unit of ROI, i.e. *Sharpe Ratio*, the Dynamic approach is outperformed by the SP500 index by a small margin.

The case study C in the next chapter will analyse this approach in different training and test periods to better address the dynamic GA performance.

c) Comparing Dynamic and Static Approaches

The following section completes the case study B by comparing the results obtained for the Static and Dynamic GAs formerly explained in the past sections of this case study. The static and dynamic approaches will be compared in terms of ROI and *Sharpe Ratio* with each other and with the two benchmark indexes selected, the S&P 500 index and the *Whale Index Approximation*.

In terms of *Return on Investment*, the dynamic approach outperformed the static in both best and average portfolios, regarding the worst portfolio both strattegies achieved approximate results. The ROI

in both cases showed low volatility, presenting a steady growth correlated with the S&P 500 index. The Dynamic GA outperformed the best portfolio by 11.24**%** and the average portfolio by 5.35% when compared to the Static GA, but the static strategy had a better worst portfolio by 1.49%.

In respect to the *Sharpe Ratio*, the dynamic approach presented better results when compared to the static counterpart. The Dynamic GA surpassed the Static GA in the best, average and worst portfolios, thus showing less risk for a higher return. The Dynamic GA exceeded the best portfolio by 3.2030 points, the average portfolio by 1.7389 points and the worst portfolio by 0.9430 points when compared to the Static GA in terms of *Sharpe Ratio*.

The results of both static and dynamic approaches can be seen in table 16.

Table 16. Dynamic and Static GA Results

		AVERAGE SOLUTION	BEST SOLUTION	WORST SOLUTION
STATIC GA MODEL	**ROI**	0.1119	0.3378	**-0.0096**
	Sharpe Ratio	11.7658	32.4844	-3.7733
DYNAMIC GA MODEL	**ROI**	**0.1654**	**0.4502**	-0.0245
	Sharpe Ratio	**13.5048**	**35.6875**	**-2.8302**

Regarding the benchmarks, the S&P 500 index in terms of ROI surpassed, by a small portion, the static approach on average, being outperformed by the dynamic approach by 4.76%. The S&P500 surpassed both GAs in matters of *Sharpe Ratio*, thus showing less risk per unit of ROI. Although, S&P500 only beats the dynamic strategy by 0.8856 points, showing that de Dynamic GA can achieve better results in terms of profit in average with a relatively small decay of risk when compared to the index in discussion.

The WIA in terms of ROI is surpassed by both approaches, showing that the Genetic Algorithm as a model can outperform the overall portfolio of all the hedge funds considered combined. The figure 34

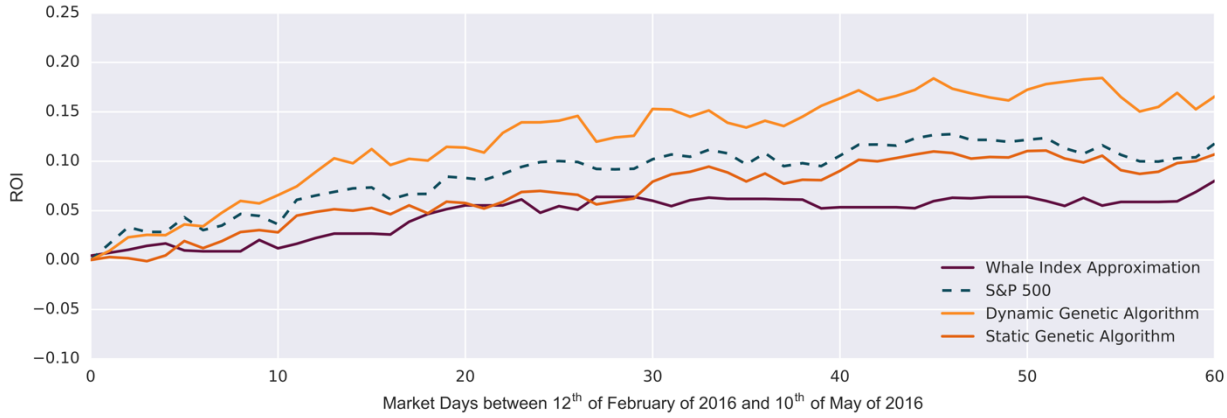

Figure 34. ROI progression for WIA, SP500, Average Dynamic GA and Average Static GA

shows the WIA against each approach during the test period.

The case study B developed proved that the dynamic approach can surpass the Static Genetic Algorithm exploiting diversity, memory and adaptability by means of dynamic enhancement, finding better solutions throughout the search procedure of the sample space. The dynamic model proved to be the most profitable model on average against all benchmarks in the test period with little difference in terms of risk against the S&P 500 index. The next case study will focus on the dynamic approach here presented and explores the Dynamic Genetic Algorithm further in other time periods.

4.2.4 Case Study C – Proposed Model Performance

The proposed model is a Dynamic Enhanced Genetic Algorithm as explained in chapter 3. The specific configuration of the Dynamic GA implemented during this case study is the same as case study B and can be seen in table 14. The Dynamic Genetic Algorithm tries to explore the adaptability, diversity and memory of the static counterpart to achieve a better performance in matters of fitness.

The following experiment is structured into four different pairs of train and test periods, the train periods are 12 months while the test periods are only 3 months. The train and test pairs are generated by a sliding window as shown in figure 34.

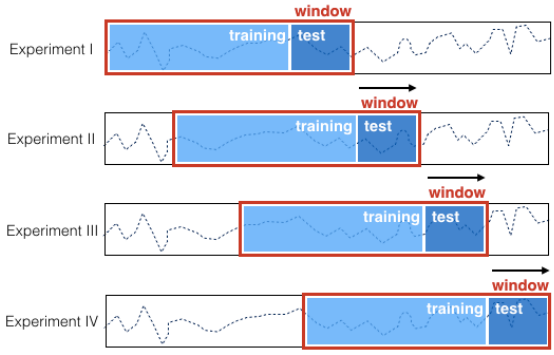

Figure 35. Sliding Window Scheme

The four experiments follow different pairs of train and test created according to the figure 35 and each experiment configuration can be seen in the following table 17. Each experiment is composed by a training set with 4000 stocks available, doing 100 executions which returned the top 10 portfolios. Each execution is composed by a population of 100 possible solutions, optimized during 10 generations. After all the executions, the 1000 possible portfolios will be analysed in terms of ROI and *Sharpe Ratio* to validate the proposed system performance.

Table 17. Dynamic GA Train/Test Configuration

EXPERIMENT	TRAINING PERIOD	TEST PERIOD
EXPERIMENT I	12/02/2014 – 12/02/2015	12/02/2015 – 12/05/2015
EXPERIMENT II	12/05/2014 – 12/05/2015	12/05/2015 – 12/08/2015
EXPERIMENT III	12/08/2014 – 12/08/2015	12/08/2015 – 12/11/2015
EXPERIMENT IV	12/11/2014 – 12/11/2015	12/11/2015 – 12/02/2016

The figure 36 presents the S&P 500 index evolution and the *Whale Index Approximation* during all the test periods considered for this case study experiments. Regarding the S&P 500, the year proposed for our system validation is in the overall period quite bearish (downtrend), showing that the market ended up, in the proposed period, with an overall ROI of -10.9%. In terms of WIA, the year proposed follows the S&P 500 index but with a slightly worse performance, which had an overall ROI of -15.17%. The year selected for testing the proposed system shows a downtrend behaviour, thus being harder for our model to thrive because only long positions were considered in this study, leaving short selling for further

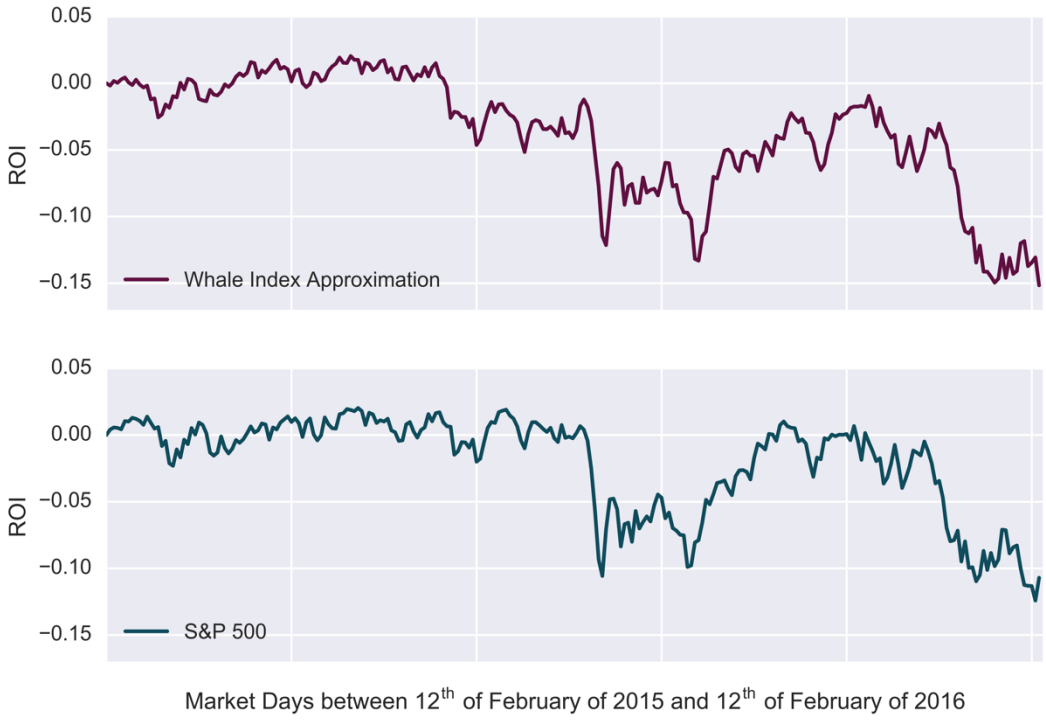

Figure 36. SP500 and WIA Return on Investment between 12/02/2015 and 12/02/2016

work.

The case study C will present the four experiments, one for each test period, with a performance analysis about them. After studying each test period, an analysis will be performed about the overall performance of the proposed system, being all market conditions and test periods considered. In the end, the

proposed system will be compared with the State-of-the-Art on portfolio composition.

a) Experiment I

The following experiment uses a Dynamic Genetic Algorithm trained to generate 10 possible solutions, each one representing a specific portfolio that will be tested during the test period, this procedure will be repeated 100 times and 1000 portfolios will be tested. The Dynamic GA was trained from 12th of February of 2014 to 12th of February of 2015 and was tested from 12th of February of 2015 to 12th of May of 2015.

The objective of this experiment is to analyse the proposed system performance during the selected test period in terms of ROI and *Sharpe Ratio* and compare the results obtained with our two selected benchmarks, S&P 500 index and WIA. In the end, a brief final commentary will be made about the performance in the test period.

The 1000 solutions collected were evaluated in terms of ROI and *Sharpe Ratio* for the test period, and two histograms were plotted based on that information. The SP500 ROI for the test period was traced over our distributions for reference and the KDE was traced to show an approximation of the true distribution of our data. The histograms can be seen in figures 37 and 38.

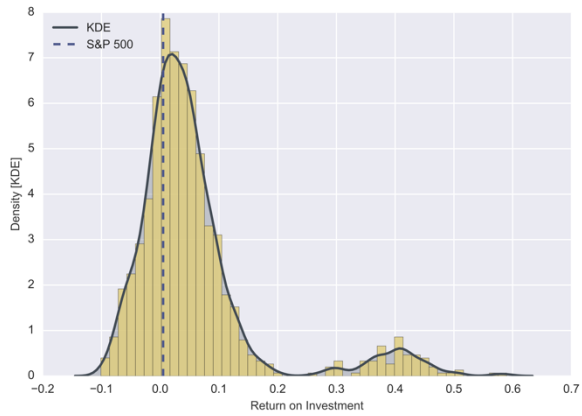

Figure 37. Histogram of ROI for Experiment I

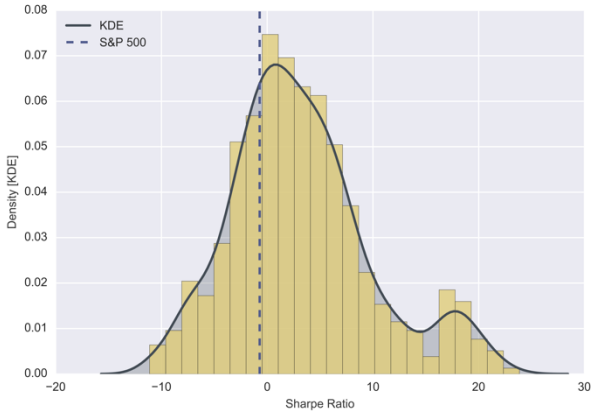

Figure 38. Histogram of Sharpe Ratio for Experiment I

This experiment results show a multimodal distribution in terms of ROI, composed by two Normal distributions with different means and standard deviations.

The figure 37 presents a histogram of the *Return on Investment* for all the proposed portfolios and the S&P 500 ROI of 0.0051 on the same test period. Accordingly, the ROI of the Dynamic GA follows a multimodal distribution with two Normal Distributions. The first Normal Distribution with mean 0.03 and standard deviation 0.0518, tells us that 68% of our model's ROI will be within one standard deviation from the mean, i.e., 68% of the portfolios proposed by the system will be between a ROI of -0.0218 and 0.0818. The second Normal Distribution with mean 0.3962 and standard deviation 0.0640, tells us that 68% of our model's ROI will be within one standard deviation from the mean, i.e., 68% of the portfolios

proposed will be between a ROI of 0.3322 and 0.4603. The overall ROI mean was 0.0571.

The figure 38 shows the histogram of the *Sharpe Ratio* for all the proposed portfolios for the given test period, and the S&P 500 *Sharpe Ratio* of -0.7127 on the same test period. As shown, the *Sharpe Ratio* is approximately a Normal Distribution with mean 3.2558 and standard deviation 6.6772. For the Normal distribution, 68% of our portfolio's *Sharpe Ratio* will be within one standard deviation from the mean, i.e., 68% of the portfolios proposed by the dynamic GA will be between a *Sharpe Ratio* of -3.4214 and 9.9330.

The figure 39 shows the ROI evolution during the test period for the best and worst portfolios, the average of all portfolios combined and the S&P 500 benchmark. The average portfolio returned by the proposed model outperformed the S&P 500 during the period of analysis. Since the WIA had a ROI of

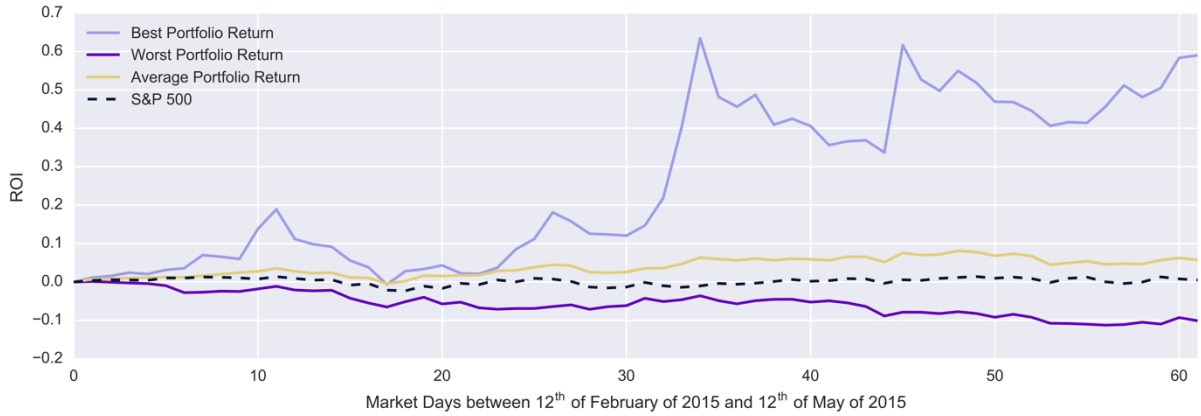

Figure 39. ROI progression for SP500 and Average, Best and Worst portfolios for Experiment I

0.0126, our model outperforms the average ROI of all hedge funds for the given test period.

For the experiment I, the proposed system had a positive performance by outperforming both benchmarks. The average of all solutions shows low volatility with steady growth and lower risk when compared to the S&P 500 by examining them both under the *Sharpe Ratio*.

b) Experiment II

Regarding experiment II, the proposed system was trained from 12th of May of 2014 to 12th of May of 2015 and was tested from 12th of May of 2015 to 12th of August of 2015. The proposed system simulates 100 different Dynamic GAs that output 10 possible solutions each for the given test period, thus ending with 1000 possible portfolios to examine.

The main goal of the second experiment is to address the proposed system results during the selected test period in terms of ROI and *Sharpe Ratio* and compare the system performance against the S&P 500 index and WIA benchmarks. At last, a conclusion about the experiment performance will be made.

The portfolios obtained after the 100 executions were tested in the test period in matters of ROI and *Sharpe Ratio*. The figures 40 and 41 show two histograms with the density distribution of our data with the SP500 index in the test period as a reference.

Figure 40 shows the histogram of the ROI with a bell shape, like a Normal distribution. The Normal distribution present has mean -0.0495 and standard deviation 0.0585, and 68% of our model's ROI will be within one standard deviation from the mean, i.e., 68% of the portfolios proposed by the system will be between a ROI of -0.1081 and 0.0089. The S&P 500 ROI in the same period was -0.0062.

Figure 41 presents the histogram of the Sharpe Ratio during the test period defined. As can be seen, the data follows a Normal distribution with mean -5.3750 and 5.312 standard deviation. In terms of Sharpe Ratio, one can assume that 68% of the portfolios will be between -10.6871 and -0.0628, i.e. one standard deviation from the mean value. The S&P 500 Sharpe Ratio in the same period was -2.3991.

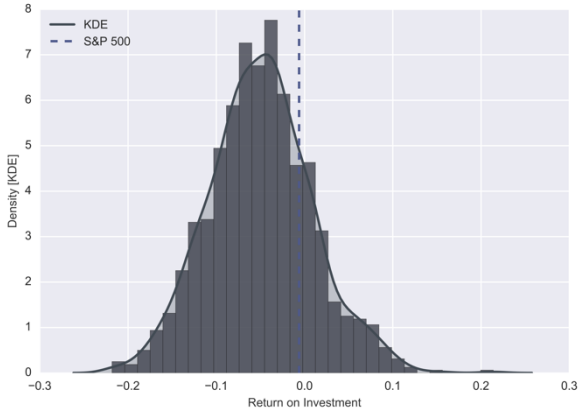

Figure 40. Histogram of ROI for Experiment II

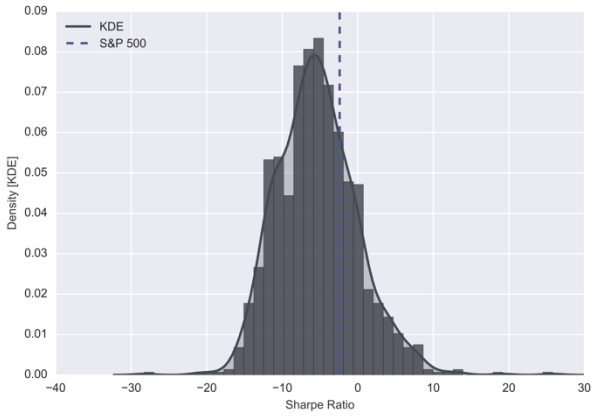

Figure 41. Histogram of Sharpe Ratio for Experiment II

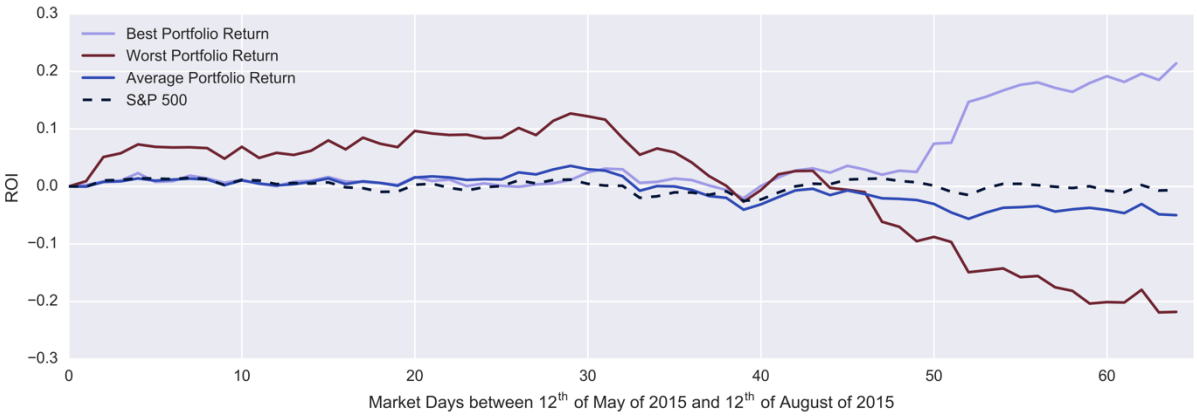

Figure 42. ROI progression for SP500 and Average, Best and Worst portfolios for Experiment II

Figure 42 shows the ROI progression along the test period, during this period the average portfolio was surpassed by the S&P 500 index. In this experiment, the average ROI was negative, changing to this situation on the last month of simulation and the average portfolio showed a riskier behaviour than the S&P 500 index. The best portfolio had a ROI of 0.2139 and the worst group of stocks had a ROI of -0.2181. The WIA for the test period was -0.0451, slightly better than the average portfolio.

c) *Experiment III*

The experiment III trained the proposed system from 12th of August of 2014 to 12th of August of 2015 and tested the obtained model between 12th of August of 2015 and 12th of November of 2015. The proposed system will be executed 100 times and 1000 portfolios will be generated and evaluated in matters of ROI and *Sharpe Ratio* and finally compared with the two benchmarks selected. At the end of this third experiment, a brief commentary about the overall performance will be made.

The possible solutions obtained were evaluated in terms of ROI and *Sharpe Ratio* for the test period,

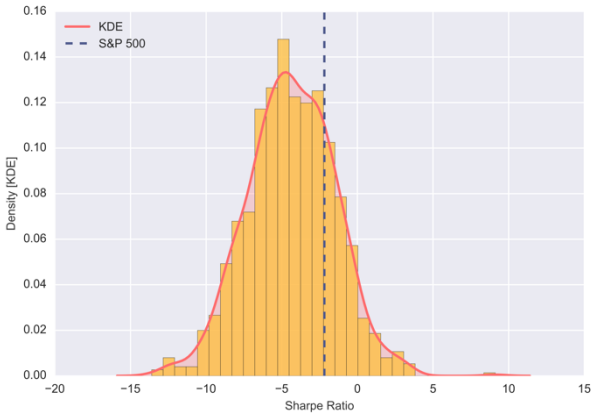

Figure 43. Histogram of Sharpe Ratio for Experiment III

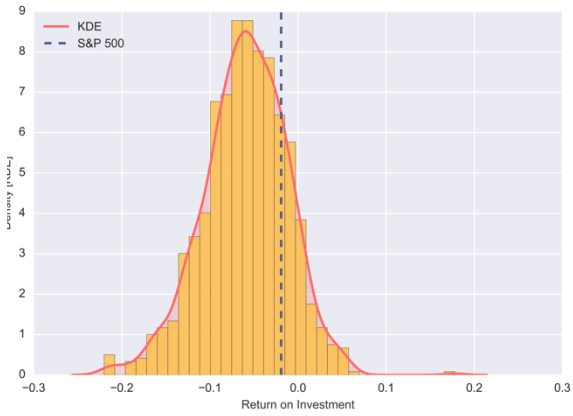

Figure 44. Histogram of ROI for Experiment III

and two histograms were plotted with the SP500 ROI for the test period and an approximation of the distribution of our data using KDE. The histograms can be seen in figures 43 and 44.

The figure 44 shows a histogram of the *Return on Investment* for all the experiment portfolios and the S&P 500 ROI on the same test period, which is -0.0192. As one can see, the ROI of the experiment follows a Normal Distribution with mean -0.0597 and standard deviation 0.0474, which tells us that 68% of our model's ROI will be within one standard deviation from the mean, i.e., 68% of the portfolios proposed by the static model will be between a ROI of -0.1072 and -0.0123.

The figure 43 shows a histogram of the *Sharpe Ratio*'s for all the portfolios and the S&P 500 *Sharpe Ratio* in the test period, which is -2.1833. Looking at the figure 43, it's apparent that the *Sharpe Ratio* of the portfolios proposed by our system can be approximated to a Normal Distribution with mean -4.3629 and standard deviation 2.8645. Being our results approximated to a Normal Distribution, 68% of the portfolios returned by the Static GA will be between -7.2275 and -1.4984 in terms of *Sharpe Ratio.*

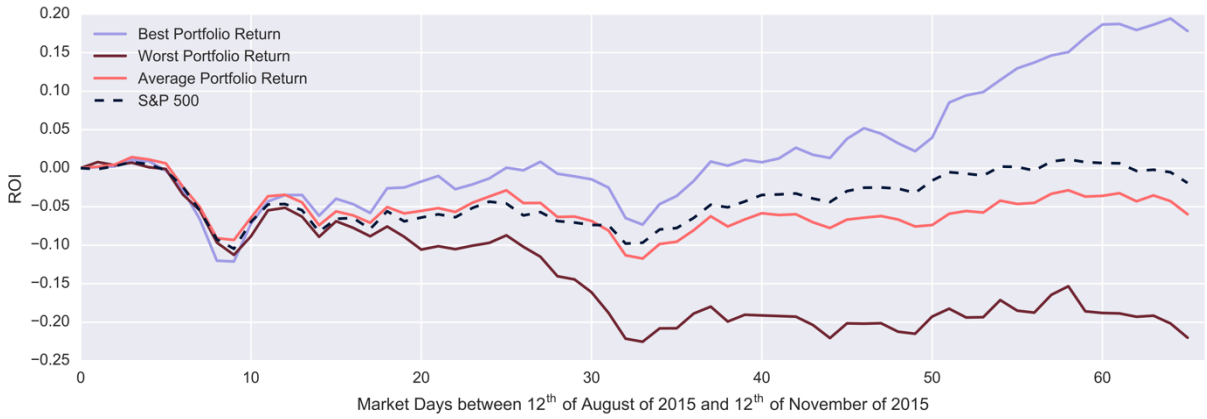

Figure 45. ROI progression for SP500 and Average, Best and Worst portfolios for Experiment III

The figure 45 shows the evolution of ROI during the test period for the best, worst and average approaches and the SP500 benchmark index. The S&P 500 index surpassed the average return of all portfolios combined. The best portfolio achieved a ROI of 0.1780 and the worst had a ROI of -0.2201. The proposed system outperformed the WIA by a small margin, being the WIA for the given period of -0.0621.

For the experiment III, the proposed system was outperformed on average by the S&P 500 during the test period, showing a relatively good performance until the second half of the test period is reached and the algorithm suffers a slight drawdown. Since the available model only uses long positions, it becomes quite hard to endure a good performance when the overall market has a downtrend in the test quarter. This situation could be handled by developing a system to define which stocks to long and which to short in a further work perspective.

d) Experiment IV

The remaining experiment trained the Dynamic GA from 12th of November of 2014 to 12th of November of 2015 and tested it from 12th of November of 2015 to 12th of February of 2016. The proposed system was executed 100 times and generated 1000 portfolios as possible solutions for the test period. These solutions were tested in matters of ROI and *Shape Ratio* and compared to the S&P 500 index and WIA.

In the end, a commentary about the performance of the proposed system during the test period will be made.

The following histograms are based on the ROI and *Sharpe Ratio* data of the 1000 portfolios generated by the proposed system during the test period of this experiment. The KDE was traced to show an approximation of the true distribution of the portfolios generated by the proposed system. Both histograms can be seen in the figures 46 and 47 bellow.

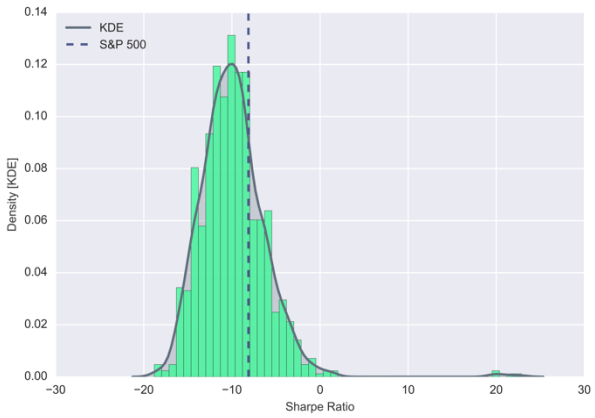

Figure 46. Histogram of Sharpe Ratio for Experiment IV

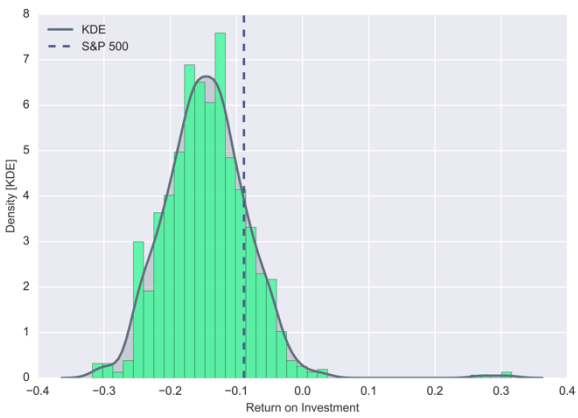

Figure 47. Histogram of ROI for Experiment IV

The histogram of ROI for experiment IV can be seen in figure 47. As one can see, the ROI of the proposed portfolios follows a Normal distribution with mean -0.1451 and standard deviation 0.0639. Considering that our model outputted portfolios follow a Normal distribution, 68% of the portfolios have a ROI from -0.2091 to -0.0812, i.e., 68% of the portfolios ROI will be between one standard deviation from the mean. The S&P 500 ROI for the same period was -0.0885.

Figure 46 presents the histogram of the Sharpe Ratio during the test period. The data follows a Normal distribution with mean -9.8511 and standard deviation 3.8381. Since in a Normal distribution 68% of the cases are one standard deviation far from the mean, in this experiment context, 68% of the portfolios generated by the proposed system are between -13.6891 and -6.0130. The overall *Sharpe Ratio* for the S&P 500 index in the same period was -8.1266.

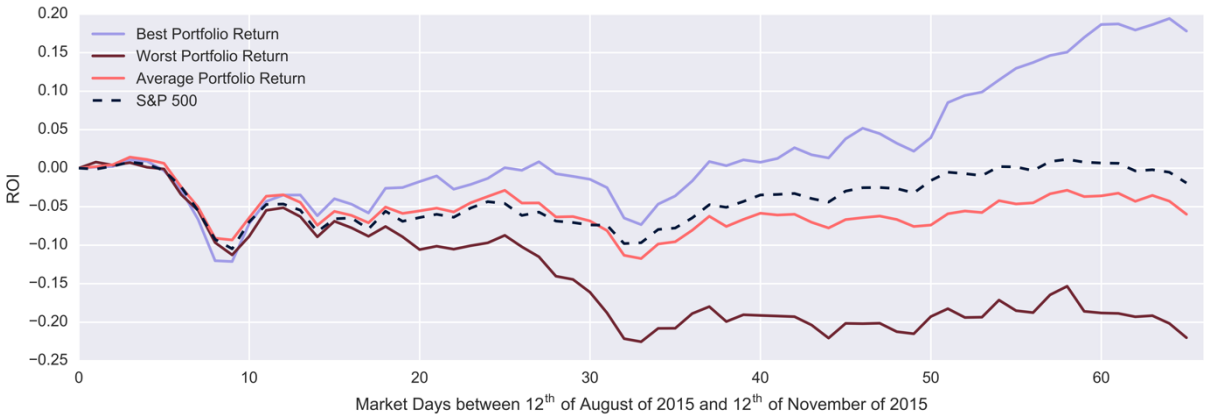

Figure 48. ROI progression for SP500 and Average, Best and Worst portfolios for Experiment IV

The overall market during the test period showed a downtrend behaviour, being both the S&P 500 and the average of the proposed solution's portfolios negative. The situation described before can be seen in figure 48, where the best, worst and average portfolios had, respectively, a ROI of 0.3158, -0.3173 and -0.1451. The S&P 500 surpassed our model for the given period, thus showing a better performance with approximately the same volatility. The WIA surpassed our approach by a small margin, showing a ROI for the given period of -0.1153.

The experiment IV test period was the hardest for our proposed system to track because of the accentuated drawdown that happens in the beginning of the period. Since the overall period is down trended and since our model only has long positions available, a performance like this was expected considering that the short selling counter-balance was disabled in our proposed system.

e) Overall Performance for the Proposed System

The system performance was tested during the years of 2015 and 2016. The tests of performance made to the system were in matter of ROI and *Sharpe Ratio* and all the tests were compared to our benchmarks, the S&P 500 index and the *Whale Index Approximation.* The table 18 summarizes the results obtained across the experiments and the S&P 500 and WIA values for the given test periods.

The test period selected, for all the experiments combined, is one year from 12th of February of 2015 until 12th of February of 2016. The year selected shows a negative trend, having a negative ROI for both benchmarks for the year tested. The results obtained by our model for the 4 different experiments can be seen at table 18.

The year period defined was divided into four different quarters. During the four experiments, only one quarter had a positive ROI, against the other 3 quarters with an overall negative ROI for our proposed system. The experiment II, III and IV all had negative returns, with -4.9%, -5.9% and -14.5% respectively, and the experiment I presented a good overall return of 5.7%. Regarding the year tested, the proposed system had an overall ROI of -19.6% against -10.9% from the SP500 and against -18.67% from the WIA. In terms of yearly ROI against the SP500 and WIA, the model was surpassed by both benchmarks, but the period is considered adverse, with at least 3 negative quarters in matters of SP500. In the other hand, the proposed system tries to beat the two benchmark indexes by using, at maximum, 12 stocks, against 500 stocks for the SP500 portfolio and 4000 stocks for the WIA portfolio, thus being both much less volatile in comparison and exposed to much less risk by taking into consideration that the assets are much sparser. In the other hand, the proposed system only uses long positions, being exposed to a down trended market without any short activities to manage the overall market fall.

Nevertheless, during a positive quarter for the American market, represented by the experiment I, the proposed system had a positive ROI, beating both benchmarks and shows that the proposed solution is well behaved during bullish markets (uptrend). The case study B corroborates this situation, since the overall ROI outperformed both benchmarks as well for that quarter.

In terms of *Sharpe Ratio*, the results showed a similar behaviour to ROI, being surpassed during all the negative quarters, but showing much better results during bullish markets, showing less risk for much

more *Return on Investment*.

Table 18. Proposed System Results in all the Experiments

		EXPERIMENT I			EXPERIMENT II		
		Average Solution	**Best Solution**	**Worst Solution**	**Average Solution**	**Best Solution**	**Worst Solution**
DYNAMIC GA MODEL	**ROI**	**5.7%**	58.9%	-10.1%	-4.9%	21.4%	-21.8%
	Sharpe Ratio	3.26	23.86	-11.12	-5.37	25.81	-28.29
S&P 500	**ROI**	0.5%			**-0.6%**		
	Sharpe Ratio	-0.71			-2.39		
WIA	**ROI**	1.26%			-5.0%		
		EXPERIMENT III			**EXPERIMENT IV**		
		Average Solution	**Best Solution**	**Worst Solution**	**Average Solution**	**Best Solution**	**Worst Solution**
DYNAMIC GA MODEL	**ROI**	-5.9%	17.8%	-22.0%	-14.5%	31.6%	-31.7%
	Sharpe Ratio	-4.36	9.16	-13.59	-9.85	22.86	-18.81
S&P 500	**ROI**	**-1.9%**			**-8.9%**		
	Sharpe Ratio	-2.18			-8.13		
WIA	**ROI**	-2.9%			-11.53%		

f) Proposed System and the State-of-the-Art

The works described at the state-of-the-art were all developed considering only the *Return on Investment* as a performance metric, thus being ignored the *Sharpe Ratio* for this sub-section sake. The assets considered for our proposed system are different from all the other works, since 4000 stocks were used in this study, while in the others only the stocks present in the major indexes like SP500 or Dow Jones Stock Index, to see more detailed information about the assets considered by other studies check table 5.

Since there is no defined dataset by the academic community to benchmark financial models and since the models are so dependent on the asset class under examination, it's quite hard to compare solutions in precise terms. Other problem associated with system comparisons in this financial paradigm is the fact that each researcher uses different time periods, which generates time-dependent benchmark studies.

The portfolio composition system proposed by Iba & Aranha [18] showed that a Dynamic approach offers better results when compared to the simples Genetic Algorithm, the same was studied in this work

context and the same conclusions were reached, that a dynamic approach exceeds the simple one by using non-stationary strattegies.

In matters of return on investment and risk, the comparison between this solution and the state-of-art offers no real value in matters of benchmark, since the time periods and assets are completely different between each one of the works, thus resulting in much different liquidity ratios and risk.

Chapter 5

Conclusions

The Conclusions chapter gives a final overview about the work developed during this thesis. The chapter is completed with future work suggestions about portfolio composition.

5.1 Conclusion

The main objective of the work developed during this thesis is to optimize a portfolio composition problem using evolutionary computation to achieve this optimization process. The optimization is made using Dynamic Genetic Algorithms with hedge fund and stock data to better understand the underlying strattegies performed by the different hedge funds. The proposed system showed a good performance at specific types of market, thus showing that the Dynamic Genetic Algorithm might be a good baseline to a portfolio composition problem. The test period is composed by 4000 stocks from 12th February of 2015 to 10th of May of 2016. The overall test period was bearish (downtrend) so the proposed system had some problems in some down trended quarters.

To successfully address our systems performance, three case studies were developed. In section 4.2.2, hedge funds proved to be quite successful investment vehicles that can be profitable with the right optimization procedure. After that, a performance estimator for the overall hedge funds was made called *Whale Index Approximation*.

In the sub-section 4.2.3, the dynamic approach was compared against the static counterpart. Regarding the test period considered, the dynamic approach showed, on average, a better performance than the Static Genetic Algorithm, this proving that a dynamic approach can better explore the sample space in matters of diversity, memory and adaptability, thus achieving better results by doing a more efficient search of the sample space considered.

The sub-section 4.2.4 tested the proposed system during a year, by creating four different test periods. The periods obtained showed an overall negative trend which was reflected in our system performance, since the model developed only has access to long market positions, thus being unable to counter balance the overall downtrend with some short positions. Regarding the four quarters used in test, only one had a positive overall SP500 ROI, being this quarter outperformed by our proposed architecture. All the other quarters were bearish, being both indexes able to outperform the proposed system.

After scrutinizing the results obtained during the different case studies performed, the proposed system showed that can outperform both the S&P 500 and the WIA in both profit and risk if the overall period has a bullish trend (uptrend), then our algorithm is able to exploit the market and achieve better results than our benchmarks. During bearish markets, and since the proposed system has no short selling mechanisms, the overall performance is worse than both indexes during mainly down trended markets. The proposed solution showed an average quarterly ROI of -0.612% and a *Sharpe Ratio* of -0.564, the best quarter had an average ROI during the test of 16.5% and the worst quarter had an average ROI of -14.5%.

The dynamic approaches applied improved the overall performance of the Static Genetic Algorithm by being more efficient at exploiting features like adaptability, memory and diversity, being able to react to

sudden changes in the market by activating different states that adjust the Genetic Algorithm parameters to more suitable ones.

5.2 Future Works

For a portfolios composition problem, there are many ways to approach the problem to successfully create a viable model. Although the Evolutionary Computation model developed showed promising results, different models combined with different abstractions of the problem might achieve better results. For further work, is suggested to explore some of the following ideas:

- Approach the portfolio composition problem by using a different set of rules to define the resulting portfolios;
- Explore unsupervised methods to be able to extract features from the 13F formularies;
- Explore deep learning approaches, such as *Long-Short Term Memory* approach, to make a better usage of the time-series information about each stock;
- Use fundamental and Technical Indicators to improve the approach by adding more information to the model;
- Explore new dynamic approaches presented on this thesis, like predictive approaches using Kalman filters;
- Explore the most used supervised models, like Gradient Boosting Trees, Support Vector Machines or Logistic Regression.

References

[1] J.H. Holland*, Adaptation in Natural and Artificial Systems*, Reprint edition, MIT Press, 1975.

[2] P.H. Winston, *Artificial Intelligence*, 3rd edition, Addison-Wesley, 1992, pp. 505-528.

[3] F.J. Fabozzi, F. Modigliani and F.J. Jones. *Foundations of Financial Markets and Institutions*, 4th edition, Prentice Hall, 2010.

[4] Report of The President's Working Group on Financial Markets, *Hedge Funds, Leverage, and Lessons of Long-Term Capital Management*, 1999.

[5] (2015, Dec.) *Form 13-Reports Filed by Institutional Investment Managers*, U.S. Securities and Exchange Commission [Online]. http://www.sec.gov/answers/form13f.htm

[6] (2015, Dec.) WhaleIndex Portfolio [Online]. http://whalewisdom.com/whaleindex

[7] M. Buffet, D. Clark, *Warren Buffet and the Interpretation of Financial Statements*, 1st edition, Scribner, 2008.

[8] B. Graham, J. Zweig, *The Intelligent Investor*, Rev Sub edition, HarperBusiness, 2006.

[9] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd edition, Springer, 1992.

[10] G. Syswerda, *"Uniform Crossover in Genetic Algorithms"*, in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 2-9, San Francisco, USA, 1989.

[11] C. Cruz, J.R. González, D.A. Pelta, Optimization in *"Dynamic environments: a survey on problems, methods and measures"*, in *Soft Computing,* vol. 15, 2011, pp. 1427-1448.

[12] S. Yang, *"Evolutionary Computation for Dynamic Optimization Problems"*, in *Tutorial: Evolutionary Computing for Dynamic Optimization Problems,* GECCO'15, 2015.

[13] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems", in CEC'99, 1999, pp. 1875–1882

[14] L. Bottou, Large-Scale Machine Learning with Stochastic Gradient Descent. In Proceedings of COMPSTAT'2010, pages 177-186, 2010.

[15] L. Wagman, "*Stock Portfolio Evaluation: An Application of Genetic-Programming-Based Technical Analysis"*, in *Genetic Algorithms and Genetic Programming at Stanford*, 2003, pp. 213-220.

[16] A. Loraschi, A. Tettamanzi, M. Tomassini, P. Verda, "*Distributed Genetic Algorithms with an*

Application to Portfolio Selection", in *Artificial Neural Nets and Genetic Algorithms*, Springer, 1995, pp. 384-387.

[17] J.O. Kyong, Y.K. Tae, M. Sungky, *"Using Genetic Algorithm to Support Portfolio Optimization for Index Fund Management"*, in *Expert Systems with Applications*, vol. 28, Elsevier, 2005, pp. 371-379.

[18] C. Aranha, H. Iba, *"Modelling Cost into a Genetic Algorithm-based Portfolio Optimization System by Seeding and Objective Sharing"*, in *IEEE Congress on Evolutionary Computation*, IEEE, 2007, pp. 196-203.

[19] A. Gorgulho, R. Neves, N. Horta, *"Using GAs to Balance Technical Indicators on Stock Picking for Financial Portfolio Composition*", in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, ACM, 2009, pp. 2041-2046.

[20] A. Simões, R. Neves, N. Horta, *"Evolutionary Computing Applied to Stock Market using Technical Indicators"*, 2010.

[21] A. Gorgulho, R. Neves, N. Horta, *"Applying a GA Kernel on optimizing technical analysis rules for stock picking and portfolio composition"*, in *Expert Systems with Applications*, vol. 38, Elsevier, 2011, pp. 14072-14085.

[22] A. Silva, R. Neves, N. Horta, *"A Hybrid Approach to Portfolio Composition based on Fundamental and Technical Indicators"*, in *Expert Systems with Applications,* vol. 42, Elsevier, 2015, pp. 2036-2048.

[23] K. Michalak, P. Filipiak, P. Lipinski, "*Evolutionary Approach to Multi-Objective Optimization of Portfolios that Reflect the Behaviour of Investment Funds*", in *Artificial Intellegence: Methodology, Systems and Applications*, vol. 7557, Springer, 2012, pp. 202-211.

[24] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, *"A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II"*, in: *PPSN VI, LNCS*, vol. 1917, Springer, 2000, pp. 849–858.

[25] D.E. Goldberg, R.E. Smith, "Nonstationary Function Optimization using Genetic Algorithms with Dominance and Diploidy", in Proceedings of Parallel Problem Solving from Nature Conference, 1987, pp. 145-154.

[26] K.P. Ng, K.C. Wong, "*A new diploid scheme and dominance change mechanism for non-stationary function optimization"*, in *ICGA 6*, 1995, pp. 159–166.

[27] E.H.J. Lewis, G. Ritchie, "A comparison of dominance mechanisms and simple mutation on non-stationary problems", in *PPSN V,* 1998, pp. 139–148.

[28] S. Yang, *"Non-stationary problem optimization using the primal-dual genetic algorithm"*, in *CEC'03,* 2003, pp. 2246–2253.

[29] S. Yang, X. Yao, "Experimental study on population-based incremental learning algorithms for dynamic optimization problems", in *Soft Computing,* vol. 9, Springer, 2005, pp. 815–834.

[30] S. Yang, X. Yao, "Population-based incremental learning with associative memory for dynamic

environments", in *IEEE Transactions on Evolutionary Computation*, vol. 12, IEEE, 2008, pp. 542–561.

[31] Yang S., *"Genetic Algorithms with Memory and Elitism based Immigrants in Dynamic Environments",* in *Evolutionary Computation*, vol. 16, MIT press, 2008, pp. 385-416.

[32] R. Tinos, S. Yang, "*A self-organizing random immigrants' genetic algorithm for dynamic optimization problems*", *in Genetic Programming and Evolvable Machines*, vol. 8, Springer, 2007, pp. 255-286.

[33] J. Branke, T. Kaußler, C. Smidt, H. Schmeck, "*A multi-population approach to dynamic optimization problems*", in *Evolutionary Design and Manufacture*, Springer London, 2000, pp. 299-307.

[34] F. Oppacher, M. Wineberg, "*The Shifting balance genetic algorithm: Improving the GA in a dynamic environment*", in *GECCO'99*, vol. 1, 1999, pp. 504–510.

[35] H.G. Cobb, J.J. Grefenstette, *"Genetic algorithms for tracking changing environments"*, in *Proc. ICGA*, 1993, pp. 523–530

[36] S. Yang, R. Tinos, "*Hyper-selection in dynamic environments*", in *CEC'08*, 2008, pp. 3185–3192

[37] S. Yang, H. Richter, *"Hyper-learning for population-based incremental learning in dynamic environments"*, in *CEC'09*, 2009, pp. 682–689

[38] P. D. Stroud, *"Kalman-extended Genetic Algorithm for Search in Nonstationary Environments with Noisy Fitness Evaluations*", in *IEEE Transactions on Evolutionary Computation*, vol. 5, 2001, pp. 66–77.

[39] J. van Hemert, C. Van Hoyweghen, E. Lukshandl and K. Verbeeck, "*A Futurist Approach to Dynamic Environments*", in *GECCO EvoDOP Workshop*, 2001, pp. 35–38.

[40] P. A. N. Bosman, H. La Poutre, "*Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case*", in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ACM, 2007, pp. 1165-1172.

[41] A. Simões, E. Costa, "*Using Linear Regression to Predict Changes in Evolutionary Algorithms dealing with Dynamic Environments*", in *Technical Report TR 2007/005, CISUC*, 2007.

[42] A. Simões, E. Costa, "*Improving prediction in evolutionary algorithms for dynamic environments*", in *GECCO'09*, 2009, pp. 875–882.

[43] (2015, Dec.) Sector Tracker [Online]. http://www.sectorspdr.com/sectorspdr/tools/sector-tracker/charting

[44] (2017, Jan.) Python 3.5 (Stable) [Online]. http://www.docs.python.org/3.5

[45] (2017, Jan.) Spyder IDE [Online]. http://github.com/spyder-ide/spyder

[46] (2017, Feb.) United States Securities and Exchange Commission - Edgar Platform [Online]. https://www.sec.gov/edgar/searchedgar/companysearch.html

[47] (2017, Jan.) Beautiful Soup 4.4.0 Library Documentation [Online]. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[48] (2017, Mar.) Yahoo Finance Platform [Online]. https://finance.yahoo.com/

[49] (2017, Jan.) Yahoo Finance Library for Python [Online]. https://pypi.python.org/pypi/yahoo-finance

[50] (2017, Jan.) Pandas Data Analysis Library for Python [Online]. http://pandas.pydata.org/

[51] (2017, Jan.) Distributed Evolutionary Algorithms in Python [Online]. https://github.com/DEAP/deap

[52] J. Dean, S. Ghemawat, "*MapReduce: Simplified Data Processing on Large Clusters*", in *Communications of the* ACM 51.1, 2008, pp. 107-113.

[53] (2017, Jan.) Numpy Library for Python [Online]. http://www.numpy.org/

[54] (2017, Jan.) Matplotlib Library for Python [Online]. http://matplotlib.org/

[55] (2017, Jan.) Whale Index Explanation [Online]. https://whalewisdom.com/whaleindex

[56] (2017, Feb.) C. Loomis, *"Warren Buffet Scorches the Hedge Funds"*, in Forbes [Online] http://fortune.com/2017/02/25/warren-buffett-scorches-the-hedge-funds/

[57] (2017, Jan.) Seaborn Library for Python [Online]. http://seaborn.pydata.org/