Practical Portfolio Optimisation using Genetic Algorithm (GA)

Thurka Puvanendran

Under Supervision Of Dr Cormac Lucas

A Dissertation in partial fulfillment of requirement for the degree of

Mathematics and Statistics with Management BSc

Department of Mathematics

College of Engineering, Design and Physical Sciences

Brunel University London

Academic Year 2022/23

Acknowledgements

I would like to express my gratitude to Dr Cormac Lucas for his invaluable guidance and support throughout the course of this project. Thank you for your expert knowledge and assistance.

Abstract

This dissertation focuses on using genetic algorithms (GA) for solving the portfolio selection optimisation problem, aiming to produce a practical method for choosing assets for a portfolio. The dissertation examines four markets: CAC 40, DAX 50, FTSE 100 and S&P 500. The goal is to create an optimal portfolio using an index tracker in the form of a cardinality-constrained optimisation model used to judge how well the GA performs. The cardinality-constrained index tracking problem is a known practical portfolio optimisation. It results in a combinatorial optimisation problem which is challenging to solve. The GAs developed as part of this paper will be compared with a benchmark which is a branch-and-bound solver. To obtain solutions from the branch-and-bound solver, various limits had to be applied to the solver, such as mixed integer solution limit or CPU time limit.

List of Tables

Contents

Chapter 1

Introduction

Portfolio optimisation is an important topic in many fields, such as finance and management, as it allows the creation of portfolios with the aim of maximising return and minimising risk. The outlook on optimisation has altered since Markowitz introduced the mean-variance model in 1952 [1]. Since then, several other models and methods for creating portfolios have been developed, with different solution methods such as genetic algorithm (GA), on which this paper will focus. As there are no definitive ways for creating a portfolio, and as genetic algorithms function well with optimisation problems, using GA is a popular method for solving such problems. It is based on Charles Darwin's theory of survival of the fittest [2]. Those more likely to survive are most likely to pass on their traits to the following generation. An index tracking model in the form of a cardinality optimisation model will be utilised to assess how effectively the GA performed in solving this problem. While the use of GA for portfolio selection is not novel, this research does something special by introducing intelligence to our population (choosing our assets in a specific, non-random way to lead to good value quicker and using the help of a CPLEX solver [3]).

Aims

- Develop a portfolio optimisation model that considers practical issues such as cardinality (i.e., the maximum number of assets in the portfolio) and threshold constraints (i.e., minimum or maximum weights for certain assets).
- Evaluate the effectiveness of the model in creating efficient portfolios in different markets.

Objectives

- Implement a genetic algorithm (GA) in which the portfolio closely follows the market based on historical data.
- Test the GA-based model on four different markets (CAC 40, DAX 40, FTSE 100, S&P 500)
- Fine-tune the GA code by adjusting parameters.

Chapter 2

Literature Review

2.1 Portfolio Optimisation

In this chapter, we introduce some financial definitions such as what an asset, portfolio, and capital market value is and review the existing literature on portfolio optimization, including Modern Portfolio Theory and the Capital Asset Pricing Model.

2.1.1 Asset

An asset is a resource that has economic value and can provide future benefits, such as generating revenue [4]. The different types of assets include current assets, fixed assets, financial assets, and intangible assets [4]. However, this project focuses on financial assets, representing investments that can be bought and sold on financial markets. Examples of financial assets include stocks, bonds, and mutual funds; their value can be influenced by market conditions, supply and demand, and other factors [4].

In particular, we are interested in the rate of return on an asset and the variance of the return. Formerly given, the price on the day i is x_i, and the next day, $i + 1$, is x_{i+1}. The rate of return of the asset is

$$r_i = \frac{x_i - x_{i+1}}{x_i},$$

and the variance,

$$\sigma^2 = \frac{\sum_i^n (r_i - \overline{r})}{n},$$

where n is the total number of days.

2.1.2 Portfolio

A portfolio is a group of financial assets held by investors. The main benefit of holding a portfolio is that it allows investors to minimize risk through diversification [5]. By investing in a collection of assets, investors can spread their risk across many different assets and potentially earn higher returns. Conversely, investing in only one individual asset carries more risk, as the performance of that single asset will solely influence the investors' returns.

2.1.3 Capital Market Value

The capital market value indicates the total market value of a company's stock compared to other companies in the market [6]. Furthermore, it shows how much a "company is worth on the open market" [6]. It does not only look at the company's present value but also the stock's future value, as all of that is represented in the current share price, which is used to calculate the capital market value [6]. The equation used to calculate the capital market value is:

$$\text{Market Capital Value} = \text{Current share price } \times \text{Total number of shares outstanding [7].}$$

If a company has a high capital market value, then it means that investing in them is quite safe as they tend to be large well-known companies in their respective industries with steady growth [6]. Companies with mid-capital market value are not as large as those with high capital value but show high growth potential [6]. Those with small capital market value are more easily affected by changes in the market due to limited growth as they usually represent small companies which are riskier to invest in [6].

2.1.4 Modern Portfolio Theory

Modern Portfolio Theory (MPT) is a theoretical framework that demonstrates how to efficiently build a portfolio by not looking at assets individually [1]. Instead, MPT directs us to consider the overall portfolio return and risk to obtain a higher return for a lower risk level [1]. It is based on Harry Markowitz's mathematical framework of the mean-variance model, published in 1952 in his paper ''portfolio selection'' [5, 1].

As stated in its name, the mean-variance model used the expected return to measure portfolio returns and variance to measure risk [8, 5]. The model states that the higher the risk, the higher the expected return [9]. A critical factor in this model is diversification. Markowitz tells us the importance of diversification, as it can help reduce risk by investing in a range of assets [5]. However, although it is possible to minimise the risk, eliminating it is impossible [5].

Another essential element is the efficient frontier, also known as the envelope curve [8]. The efficient frontier is a graphical representation of which portfolios are best for investing in a given level of return while minimising risk or vice versa [9, 10]. Rational investors are expected to invest on the line of the envelope curve because these portfolios are the most efficient [9]. The part of the curve where the investors are depends on how risk-averse they are [9, 10].

As the model is static [8], a problem with the efficient frontier is the constant need to rearrange the graph to determine the investor's optimal portfolio. In addition, the investor will have to constantly change the portfolio to meet this, which would increase transaction costs in the real world. Furthermore, the model assumes that all information is reflected in the asset value [8, 9, 5], and thus the portfolio return and risk are accurate; however, this is not true and is just an assumption.

While the model has unrealistic assumptions, modern portfolio theory provides a framework for building a successful portfolio by examining the relationship between risk and return and how to reduce that risk as it is an unwanted factor.

2.1.5 Capital Asset Pricing Model

The Capital Asset Pricing Model (CAPM) was developed by Sharpe and Lintner in 1964 and built on Harry Markowitz's portfolio theory [9, 8]. This financial model examines the relationship between systematic risk and the expected return on assets [11]. This model was revolutionary at the time, as it was made when quadratic optimisation problems were difficult to solve, as CAPM allows us to look at it from a more straightforward perspective. While Modern Portfolio Theory (MPT) considers both systematic and unsystematic risk and the return of portfolios, CAPM only considers the "systematic risk of individual securities to determine their fair price" [9]. This is because CAPM assumes a portfolio can be fully diversified, eliminating unsystematic risk [9].

Systematic risk affects the overall market, while unsystematic risk affects specific industries. This is why systematic risk can be minimised through diversification by investing in assets with a negative correlation such that when an asset's return increases, the other asset's return decreases [9].

The beta coefficient in CAPM is a crucial component reflecting how an asset is affected by market risk [9]. Specifically, the higher the beta value (indicating more systematic risk), the higher the expected return for that asset. This suggests that investors who bear market risk are rewarded with higher returns. This relationship shows that the market is efficient; thus, the market is the best investment. As a result, a proxy for tracking the market through index funds can be an efficient strategy for investors.

However, the market being efficient relies on the assumption CAPM has stated. The assumption affecting this is that the capital market is perfect [9]. This is not true, as for a capital market to be perfect would mean that taxes and transaction costs do not exist, but they do occur in the real world [9]. While the market is not perfect, there is empirical evidence that indicates that the market is highly efficient [9].

2.1.6 Index Tracking

There are two types of fund management: active and passive. Passive fund management is for risk-averse investors who conform to a set of constraints in order to attain a minimum level of return [12]. Active fund management is for investors willing to take risks to outperform the market; they use their expertise to determine which stocks to invest in to maximize return [12]. Index tracking is a type of passive fund management in which the goal is to be as close to a reference financial index (such as CAC, FTSE,...) as possible [12] using a subset of assets, rather than investing in every asset in the reference financial index, to avoid high transaction costs. This leads to a combinatorial optimisation problem and we now discuss one solution approach.

2.2 Genetic Algorithm

The Genetic Algorithm (GA) is a powerful method for solving optimisation problems. The underlying ideas regarding GAs were developed by John Henry Holland [13]. They are beneficial because they are population-based algorithms, meaning they have multiple candidate solutions [14]. Unlike a single-solution algorithm, a population-based algorithm is less likely to produce a solution stuck in a local optima because it works with multiple possible solutions, thereby increasing the likelihood of finding the optimum [2].

It is also a metaheuristic algorithm, and, like many others, it is inspired by biological processes [14]. More specifically, GA is based on Darwin's theory of evolution and mimics biological procedures such as natural selection and reproduction, allowing the user to achieve high-quality outputs when solving problems like optimisation problems [2].

It is also an iterative process; we hope to get closer to the optimal solution with each iteration. However, it is not guaranteed that the global optimum solution will be discovered because GAs are designed to find a reasonably good solution to the problem in a relatively short amount of time, as this method is typically used when no known analytical or specialised techniques are available [13].

2.2.1 Terminology

Chromosomes

Following the biology inspiration, individuals are usually referred to as chromosomes. The chromosomes are made up of genes; these have the traits that make up the characteristic of the chromosome.

Fitness value

The fitness value indicates how good our chromosome is; the higher the fitness value, the better the chromosome's traits. And if it has a high value, we want to breed it so those traits that give it a high value can be passed on to the next iteration to get closer to our optimal solution. The fitness value is determined by a function that we have defined and is what "we seek to optimise or the problem we attempt to solve."[2]

Encoding Schemes

Encoding schemes transform one piece of information into another; this is essential in GA because it allows us to implement crossover and mutation operators more quickly, saving us valuable computation time [14]. It is used to encode the chromosome. There are numerous encoding schemes, including octal, permutation, binary, and tree, but we will only be focusing on applying binary [14].

In binary, the chromosome will be represented as a bit string, so the genes will only take the value 1 or 0.

Biological Operators

There are three biological operators used in GA; these are selection, crossover, and mutation [2, 14]. Selection, also known as the reproduction operator [14], is the process of choosing the chromosomes that will be allowed to breed to produce offspring for the next generation. Its fitness value determines this; the higher the fitness value, the more likely the chromosome is selected to be a parent.

Crossover is the process of creating offspring; this is done by combining the parents' genetic information. The offspring can be formed by two or more parents, rather than just two [14].
Mutation is a random procedure that alters a portion of the offspring's genetic material. This is done to preserve diversity which is essential to avoid premature convergence [14].

2.2.2 Intelligent Population

An intelligent population is one in which we use specific knowledge and data to guide the selection of genetic information, which can lead to a faster and more efficient solution. However, there is a risk of introducing bias if the intelligence is not adequately designed or implemented. To mitigate this risk, limiting the portion of the population with intelligence and including a random selection of the genetic information can help ensure that the solution is more diverse and less prone to bias. This approach can balance the benefits of intelligent selection and the potential risks of introducing bias.

2.2.3 Procedure of GA

According to Darwin, individuals in a population differ from one another, and those who possess more adaptive traits will be more likely to survive and pass those traits on to their offspring [2].

Using this concept, GA works similarly. We select n individuals in a population based on their fitness value. These people are now considered the parents, and we use the crossover operator to generate the offspring. The mutation operator is then applied at random to the offspring. We continue this process for all individuals until the new generation has been created. We follow the same procedures with the new generation. We keep creating new generations until our stopping criteria are met. The stopping criteria are that the best values of the previous and the current generations do not differ, indicating that we have arrived at the best possible solution.

Schema Theorem

The schema theorem was introduced by Holland. A schema is a "pattern of gene values" [13] that represents a chromosome subset. According to this theorem, schemas are more likely to survive and be passed on to the next generation if the chromosome has a high fitness value [13]. Therefore, advantageous traits represented by successful schemas are more likely to be passed down to subsequent generations in the same way that beneficial characteristics are passed down through biological evolution.

Building Block Hypothesis

The building block hypothesis suggests that combining successful schemas that have helped increase the fitness value of a population can lead to even better overall fitness values [13]. Recombining these schemas in different ways makes obtaining more efficient and effective solutions possible.

2.2.4 Advantages of GA

Compared to traditional search algorithms, GA is less likely to become stuck in a local optimum because it can evaluate multiple candidate solutions and produce multiple good solutions. And so overall, GA has better global search capabilities [14], not only because it's a population-based algorithm but also because of the crossover and mutation operation, which ensure that all potential areas for optimal solutions are investigated [2].

Another advantage of GA is that it can handle "problems with complex mathematical representation" [2] as it will "only require the outcome of the fitness function" [2] unlike single-solution algorithm, which relies on other pieces of information like the derivative which could be hard to obtain [2].

Furthermore, it can withstand noisy behaviour due to GAs' "repetitive operation of reassembling and re-evaluating the individuals" [2].

2.2.5 Limitations of GA

Premature convergence is a significant issue in GA [14, 2]; if the individuals are not appropriately selected or diversity is not maintained because of the chosen operators, this results in premature convergence and prevents us from reaching our desired solution. For instance, a chromosome with a fitness value higher than the rest of the population would be duplicated to the point where it would dominate the population, bringing us to a local optimum [2].

Another issue with genetic algorithms is choosing the size of the initial population. It will be computationally intensive if the initial population size is too large. However, if the population is too small, we might develop a weak solution [14]. Furthermore, there is no right approach or method for determining the ideal population size [2].

The fitness function must also be carefully considered; if it is too lenient, the GA may have to undergo many iterations to find the optimal solution [14]. This is something we want to avoid since it will cause the computation to take longer than desired.

If the mutation is not considered, then "there will be no new information available for evolution" [14]; this means more areas of our solution space won't be explored. And if the crossover is not considered, the solution may be a local optimum [14]. These two operations must be cautiously considered to determine which variation best suits the problem; otherwise, the global optimum won't be reached.

2.3 Other Algorithms

GA is not the only metaheuristic algorithm for solving a cardinality-constrained problem in portfolio optimisation; it is quite popular to use these other algorithms. Additionally, some altered versions of the GAs can be used to solve such problems. In this section, we will be discussing those.

2.3.1 Ant-Colony Optimisation

Ant-colony optimisation (ACO) is a population-based metaheuristic algorithm [15]. As the name tells us, it is based on the lifestyle of ants, specifically how the ants forage their foods [15]. When ants go out of the nest to find food and do find a food source, they drop a pheromone trail [16]. That pheromone trail attracts the other ants to follow this trail [16]. As the ants follow this trail, the scent becomes more pungent [16]. The shortest path would have the most potent scents as it is more popular as it is more accessible to reach from the ant nest.

ACO mimics this concept of marking pheromones; in this case, ACO marks the best solution, not food [15]. The population comprises artificial ants that move through the search space and deposit pheromones in each solution they visit [15]. The stronger the scent of pheromones, the better the resolution. As more ants go to a specific solution, more ants will follow this motive; like in genetic algorithm, the population will converge to that particular value.

2.3.2 Particle Swarm Optimisation Algorithm

Particle Swarm Optimisation Algorithm (PSOA) was founded by Kennedy and Eberhart [15]. The social behaviour of a flock of birds and schools of fish inspires it [15]. It uses a random population known as a group of particles [15], so the population comprises of particles. The particles take up a point in the solution space. The particles are trying to find their optimal position in the solution space, and they can do that as they have ”memory” [15], which means they can retain information about their best-known position. Particles are able to move to their optimal position, and the overall population's best solution [15]. The best solution is the most particle in one spot of the search space.

2.3.3 Hybrid Geneetic Algorithms

Hybrid genetic algorithms, also known as memetic algorithms, are a recent development in which a genetic algorithm and another algorithm, such as the ones spoken above, are combined [15]. This is done to “improve the performance of the genetic search” [17], as the other algorithm can assist the GA performance by compensating the GA's weaknesses with its strengths [17]. For example, GA can quickly detect the region in which there is an optimal solution but finds it hard to localise it, but it can overcome this with the help of the local search method [17].

2.3.4 Multiple-Objective Genetic Algorithm

Multiple-objective Genetic Algorithm (MOGA) is an improved version of genetic algorithm as we have multiple objectives to meet [18]. This is great, especially when we have conflicting objective functions [18]. This method of GAs can be used in portfolio selection; rather than having one objective of just minimising risk, we can also have another objective where we maximise return [19]. This gives us more flexibility as we have a larger population with traits that we want our portfolio to have [19]; however, as the complexity of the GA increases along with the population size, finding a good solution will take much longer, even though that was an advantage of the genetic algorithm.

Chapter 3

Solving Approach

The aim of this project is to find the optimal cardinality-constrained index tracking portfolio. Due to the cardinality, it causes the LP model to become a combinatorial problem which is difficult to solve, so we look at genetic algorithms to solve it. We compare our results to the benchmark, which is a branch-and-bound solver [3, 20], to evaluate the performance of the genetic algorithm. To address this problem, we use AMPL, a software used to "develop and apply mathematical programming models," [21] as it is designed for optimisation problems.

3.1 Data

Four different markets were looked at CAC 40, DAX 40, FTSE 100, and S&P 500. The first two markets are small, as their purpose was to develop and validate the code. Because running the code is easier on a smaller dataset and less computationally expensive, checking for mistakes is easier by looking at these two markets. The other two are large markets where the application of GAs is important. As previously mentioned, this is a combinatorial problem. In addition to a large dataset, it is difficult to solve this problem without the aid of GAs or other metaheuristic algorithms.

When comparing our data to the benchmark, we expect to follow the benchmark with the CAC 40 and DAX 40 and possibly outperform the benchmark with the FTSE 100 and the S&P 500.

All the data was acquired from Refinitiv, a company that provides "financial markets data and infrastructure" [22]. The past two years were looked at, from 14/10/2020 to 14/10/2022, with the data being daily prices. The capital market value was also collected from Refinitiv; this was included as some intelligence when picking assets for the portfolios for the GA. This helps to run the code faster, as it does not rely on entirely randomly generated portfolios. The view here is that one would expect companies with a large capital market value to be in the best portfolios for the cardinality constrained optimisation problem.

3.2 The Index Tracking Problem

In this section, we will be stating the optimisation model for an index tracker, a form of passive fund management that aims to track the market index as closely as possible so we can create portfolios that closely mirror the market index while also achieving a reasonable expected return.

SETS:

$T = 1..N$	denotes time
$I = 1..A$	denotes assets
$I' = 1..F$	denotes assets forced in
$I'' = I - I'$	denotes assets that may or may not be in portfolio

DATA:

$r_{i,t}$	denotes return of asset i at time t
rm_t	denotes market return at time t
l	denotes minimum investment if an asset is to be held
U	denotes maximum investment if an asset is to be held
K	denotes maximum stock chosen
T'	denotes target return

VARIABLES:

$\omega_i \geq 0$	denotes weight in asset i, $i \in I$
$\delta_i = 0/1$	binary variable, indicating 1 if we have invested in asset i, 0 if we haven't, $i \in I''$
$u_t \geq 0$	amount over the index return t, $t \in T$

$o_t \geq 0$	amount under the index return t, $t \in T$
P_t	denotes portfolio return at time t, $t \in T$

MODEL:

$$
\begin{array}{llll}
\text{minimise:} & \sum_{t\in T}(o_t + u_t) & & \\
\text{subject to:} & \sum_{i\in I}\omega_i = 1 & & (1) \\
& \sum_{i\in I} r_{i,t}\cdot\omega_i = rm_t + o_t - u_t & t \in T & (2) \\
& l\delta_i \leq \omega_i \leq U\delta_i & i \in I'' & (3) \\
& l \leq \omega_i \leq U & i \in I' & (4) \\
& |I'| + \sum_{i\in I''}\delta_i \leq K & & (5) \\
& P_t = \sum_{i\in I} r_{i,t}\cdot\omega_i & t \in T & (6) \\
& \sum_{t\in T}\frac{P_t}{|T|} \geq T' & & (7)
\end{array}
$$

As previously stated, the optimisation model described above aims to track the index because the market is the most efficient investment. This is achieved by using the objective function to minimise the tracking error. This is the difference between the expected portfolio return and the market return, represented by o_t and u_t. The objective function serves as our fitness function; therefore, the smaller the tracking error, the more likely a given portfolio will be chosen as a parent.

We now discuss the constraints of the index tracking model. Constraint 1 is a budget constraint, and its purpose is to ensure that all money has been invested, where ω_i denotes the weight of asset i.

In an ideal world, the portfolio return would equal the market return; however, this is impossible and will make the optimisation problem infeasible; thus, we introduce constraint 2. Instead of having the portfolio return equal to the market return, the constraint states that the portfolio return is as close as possible to the market return by using the variables o_t and u_t, allowing the problem to be feasible.

Constraint 3 is the investment constraint; if we make an investment in asset i below $\geq l$, δ_i would be zero, and thus, $\omega_i = 0$.

Constraint 4 ensures that for all assets chosen to be in the portfolio, the weight of the assets must be between $l \leq \omega_i \leq U$.

Constraint 5 is the cardinality constraint, where K is the maximum allowed number of chosen stocks. This constraint makes sure that each portfolio does not have more than K different stocks. $|I'|$ represents the total number of assets in set I'.

Constraint 6 defines the portfolio return, P_t, at time t. It is somewhat redundant but is included in the model so the portfolio return can be easily accessed if needed.

The market return is not always positive. While we want the portfolio return to be as close to the market return as possible, we also do not want the expected portfolio return to be negative, so constraint 7 was added. This constraint ensures that the expected portfolio return is not negative by specifying the portfolio's minimum target return, T'.

When the solver generated the benchmark, the set I'' is the empty set as all assets in the market were included in calculating the benchmark hence set $I' = I$.

3.3 GA Pseudo-code

This section outlines the pseudocode for implementing the Genetic Algorithm in portfolio optimisation. The GA incorporates intelligence by first selecting $\frac{K}{2}$ stocks for the set I' and then calls the solver to complete the portfolio. The algorithm continues until a stable solution is reached by comparing the best solution from the current and last generation.

```
$P_p$ := sets of assets for portfolio $p$;
$C_a$ := capital market value for asset $a$ ;
$W_a$ := weight of asset $a$ ;
$E_p$ := tracking error for portfolio $p$;
$B_g$ := best solution for Generation $g$;
for $p = 1..P$ do
    if $p = 1$ then
        SORT ($C_a$) by capital market value;
        Let $I'$ = the $\frac{K}{2}$ assets of highest capital market value;
    else if $p = 2$ then
        SORT ($W_a$) by weight from the LP relaxed solution;
        Let $I'$ = the $\frac{K}{2}$ assets of highest weight;
    else
        Generate randomly $\frac{K}{2}$ assets Let $I' = \frac{K}{2}$ randomly generated assets;
    end
    $I'' = I - I'$;
    SOLVE;
    SAVE $E_p$;
    Let $P_p = I' \cup \{i \in I'' | \delta_i = 1\}$;
end
Let $g = 1$;
Let $B_1 = \min_{p \in P} E_p$;
repeat
    for $p = 2..P$ do
        Let $T = P_{p-1} \cup P_p$;
        Let $W_A = W_M + W_F, a \in T$;
        SORT ($W_A^T$) by weight;
        Let $I' = \frac{K}{2}$ assets of highest weight in $T$;
        SOLVE;
        SAVE $E_p$;
        Let $P_p = I' \cup \{i \in I'' | \delta_i = 1\}$;
        Let $g = g + 1$;
    end
    Let $B_g = \min_{p \in P} E_p$
until $B_g = B_{g-1}$ where $B_g$ is the best portfolio generation;
```

The above is the pseudocode representing the Genetic Algorithm section of the code.

The first for-loop represents how the parents (Generation 0) are formed. Only 12 parents were produced; therefore, $P = 12$. The GA does not produce the entire portfolio, but only half of it, set I', and the rest is chosen using the solver, hence why we have an intelligent population.

The first two portfolios, $p = 1$ and $p = 2$, are formed differently from the others. These two portfolios were produced by choosing assets based on a specific attribute. By doing so, we hoped to obtain very good portfolio values. With these good values, we can accelerate the process and find the best portfolio faster.

The first portfolio's set I' was chosen based on the assets with the highest capital market value. The second portfolio's set I' was chosen based on the assets with the highest weights in the relaxed LP solution to the problem. The relaxed LP is where we solve our index tracking problem but let $0 \geq \delta_i \geq 1$ rather than δ_i be binary. For the rest, $p = 3$ to $p = 12$, the portfolios set I' was formed with assets chosen randomly.

While we could have added more intelligence to the parent loop, this was done only for two portfolios, as we wanted to avoid duplicates in the earlier generations, maintain diversity, and avoid premature convergence, as chosen by these attributes will lead to similar portfolios.

The second for-loop generates the portfolio of the "children" by considering the combined weight of the "mother" and "father". We select the assets for I' based on the assets with the highest weights in the combined "mother" and "father" portfolios hence a vector, W; we create where $W_i = W_i^M + W_i^F$, W_i being the elements of vector W. We then used the solver to complete the portfolio, with a time limit set to prevent the solver from running indefinitely, causing the solver to stop at a suboptimal value usually.

The child loop is embedded in a repeat loop. This loop produces each generation of children until $B_g = B_{g-1}$ until the best solution of the current generation is equal to the best solution of the last generation. This means that our code has converged and hence, determined the optimal portfolio.

The portfolios were sorted according to their tracking error, which was computed using the optimization model outlined in section 3.2. The portfolios with the smallest tracking errors are deemed the most desirable and are chosen to breed the next generation. We also let the first child of the next generation to be the portfolio of the best parent.

Chapter 4

Experimentation

4.1 Experimentation: Changes

During the experimentation with the data, a few changes were made to the optimisation model and the genetic algorithm code. This was to improve the code to get better values (smaller index tracking errors) and to help the code run faster, as the code was extremely slow for the larger markets.

Index tracking model

The index tracking model was slightly altered to help reduce the code's time. In section 3.2, we mentioned constraint 6; this was to capture the portfolio return for each time period. As we stated before, this constraint was somewhat redundant, with its only purpose defining the portfolio return at time t. When experimenting, we realised knowing the portfolio return was not necessary, so we removed it. By doing this, the time is reduced as the code does not need to store the value of the portfolio return. However, we needed to make some changes to the model as this variable was used in constraint 7. To keep the code running, we just replaced the variable P_t with the definition of portfolio return.
Constraint 7 was:

$$\sum_{t \in T} \frac{P_t}{|T|} \geq T'.$$

And now becomes:

$$\sum_{t \in T, i \in I} \frac{\sum_{i \in I} r_{i,t} \cdot \omega_i}{|T|} \geq T'.$$

While AMPL did not have to calculate and store the portfolio return, this change did not help much in reducing the time of the code.

As removing the redundant constraint was not enough to optimise the algorithm, we then altered constraints 3 and 4. We updated the maximum weight, U, by setting it to the highest weight value among the parents' assets, specific to each market being run. This narrowed the range of possibilities, reducing the time to determine the optimal weight. This was done as the original maximum investment constraint was set to one, which was too high. Unfortunately, this also did not make much of a difference.

Furthermore, we hoped that when solving the relaxed problem, the solver would put some of the binaries at value one. When U is large in the relaxed solution, the binaries are always small, so the solver thinks they should first be tried at the value zero. By changing the range, the values of the relaxed binaries become larger, resulting in the binary sometimes being set to one.

Duplicates

A section of code was implemented to determine whether we have duplicate portfolios. To accomplish this, the mother and father's assets were compared; if they had the exact same assets, the child would be identical to the mother and father's portfolio. It was also checked whether the child portfolio contained the same assets from previous generations' portfolios. This approach saved computation time by skipping the calculation of the index tracking error for any duplicated portfolio. However, one limitation of this code is that it only considered the assets the genetic algorithm selected, not those the solver picked. Therefore, there is a possibility of false duplicates.

Quality of the Solution

At the start of the experimentation, we set the solution limit to 30 and the time limit to 1000 for the solver to solve. The high time and solution limit caused the code for the larger markets to run over three hours. To help alleviate this problem, we changed the solution limit to 20 and the time limit to 500. Thus, the code ran much faster; however, by doing this, there were a few drawbacks, which will be discussed in Section 4.2 by comparing the solution limit 20 and time limit 500 to solution limit 10 with time limit 500. The solution limit is the number of allowed bounds improvements in the solver. The time limit is the allowed CPU time that the solver can run for.

Child-loop

A different version of the child loop was implemented with the aim of improving the index-tracking error solution. We call the original child-loop, child-loop 1 and the new version child-loop 2. Whether child-loop 2 was better than child-loop 1 will be discussed in section 4.2. Below is the algorithm of child-loop 2. In child-loop 1, the child was formed by adding the weights of the

```
for $p = 2..P$ do
    Let $T = P_m \cup P_f$;
    $W'_i = max(w_{m,i} , w_{f,i}) \quad i \in T$;
    SORT $(W'_i)$ by weight;
    Let $I' = \frac{K}{2}$ assets of highest weight in $T$;
    SOLVE;
    SAVE $E_p$;
    SAVE $P_p = I' \cup \{i \in I'' | \delta_i = 1\}$;
end
```

mother and father together and then picking the assets with the largest combined weights to form half of the child, with the rest being determined by the solver. In child-loop 2, the father and mother are looked at together as an extensive set, and then $\frac{K}{2}$ of the largest weighted asset will be picked from that set and the rest by the solver to form the child.

4.2 Experimentation: Result and Analysis

We will now analyse the results obtained from running our code and evaluate its effectiveness in generating a good portfolio using the Genetic Algorithm (GA). To do this, we examine each market individually and compare the results for the two years. We also evaluate the impact of the two different child loops and the solution limit on the code's performance. Our approach involves comparing these changes to the results obtained for a solution limit of 20, child-loop 1, and year 1. This version of the code served as the control. Furthermore, we compare the results of our portfolio to a benchmark to determine whether it performs well relative to the market. Our analysis enables us to identify any similarities or differences in the portfolio's performance over time and assess the effectiveness of different parameters in generating good portfolios.

4.2.1 CAC 40

CAC 40 is the French index market with 40 assets, making it a small market index. Twenty assets were selected for each portfolio.

Comparing the years

Generation	Index Tracking Error (Year 1)	Index Tracking Error (Year 2)
Gen 0	0.000889017	0.00179318
Gen 1	0.000889017	0.00179318
CPU Time	2094.83	9.95312

Table 4.1: CAC: Year 1 vs Year 2

The index tracking error between the two years is not close in value, with a difference between the final value of 0.000904163, which one would not expect due to the timeline. This is most likely because the data were taken from 2020 to 2022 when the global pandemic occurred, so these values caused fluctuations in the market; hence, the index tracking error is quite different between the two years. The CPU time for year 2 is much shorter than that for year 1, which is unexpected.

Comparing the child-loop

Generation	Index Tracking Error (Child-loop 1)	Index Tracking Error (Child-loop 2)
Gen 0	0.000889017	0.000889017
Gen 1	0.000889017	0.000889017
CPU Time	2094.83	2216.7

Table 4.2: CAC: Child-loop 1 vs Child-loop 2

Child-loop 2 could not outperform, nor was it worse than child-loop 1. Both code iterations determined the solution after two generations, and they were equal to 0.000889017. The second child-loop took slightly longer, 121.87 longer, than the first child-loop.

Comparing the solution limit

Having a lower solution limit caused the GA to go through one extra generation as the population converges slower than a higher solution limit. However, we were still able to converge to the same value, 0.000889017. The CPU time difference between the two iterations was only 4.58 seconds, with the iteration with the lower solution limit being slightly faster.

Generation	Index Tracking Error (Solution Limit = 20)	Index Tracking Error (Solution Limit = 10)
Gen 0	0.000889017	0.000954795
Gen 1	0.000889017	0.000889017
Gen 2	-	0.000889017
CPU Time	2094.83	2090.25

Table 4.3: CAC: Solution limit 20 vs Solution limit 10

4.2.2 DAX 40

DAX 40 is a German market index and is another small market with 40 stocks. The stock Daimler Truck Holding E was removed, as we did not have complete data on those stocks from the past two years. Again the portfolio cardinality was set to be twenty.

Comparing the years

Generation	Index Tracking Error (Year 1)	Index Tracking Error (Year 2)
Gen 0	0.00174613	0.00429514
Gen 1	0.00174613	0.00429514
CPU Time	654.766	2.39062

Table 4.4: DAX: Year 1 vs Year 2

The values for Years 1 and 2 were very different, with the difference being 0.00254901. Again, this was probably affected by the pandemic. Both versions of the code took two generations to run. Noticeably, the CPU time of the Year 2 code was much shorter than that of Year 1, as was the case with CAC.

Comparing the child-loop

Generation	Index Tracking Error (Child-loop 1)	Index Tracking Error (Child-loop 2)
Gen 0	0.00174613	0.00174613
Gen 1	0.00174613	0.00174613
CPU Time	654.766	478.453

Table 4.5: DAX: Child-loop 1 vs Child-loop 2

There was no difference between using different child loops. Both converged in two generations and obtained the same index-tracking error of 0.00174613. However, the CPU time in child-loop 2 was shorter by 176.313.

Comparing the solution limit

Generation	Index Tracking Error (Solution Limit = 20)	Index Tracking Error (Solution Limit = 10)
Gen 0	0.00174613	0.00177718
Gen 1	0.00174613	0.00174613
Gen 2	-	0.00174613
CPU Time	654.766	298.109

Table 4.6: DAX: Solution limit 20 vs Solution limit 10

Both code versions converged to the same index tracking error, 0.00174613. However, the performance with the solution limit equal to 10 took three generations, as the outputs of the first generation were worse than that of the solution limit of 20. Hence, it took more generations for the population to converge to a good solution. On the other hand, while it took more generations, the CPU time was much shorter than the solution limit of 20.

4.2.3 FTSE 100

FTSE 100 is the UK market index and holds 100 stocks. When running the code, we removed the stocks Endeavour Mining (Lon) and Haleon as we had incomplete data for the two years, so we had 98 stocks in our data set. To build each portfolio, the portfolio cardinality was thirty.

Comparing the years

Generation	Index Tracking Error (Year 1)	Index Tracking Error (Year 2)
Gen 0	0.000552032	0.000528517
Gen 1	0.000547943	0.000526216
Gen 2	0.000547943	0.000526216
CPU Time	29656.8	41699

Table 4.7: FTSE: Year 1 vs Year 2

When comparing the best solution, there is a small difference of 0.000021727 between the two years. This is more in line with our expectation when comparing the two years as they are more similar in value than the smaller two markets. Interestingly, while running the code for the second year of the smaller indexes was quick compared to the first year, in this case, the computed time for the second year took twice as long as the first year. It is also interesting that it took only three generations to find the best solution, which shows that even though the solution and time limits are somewhat low, the GA still works well.

Comparing the child-loop

Generation	Index Tracking Error (Child-loop 1)	Index Tracking Error (Child-loop 2)
Gen 0	0.000552032	0.000552032
Gen 1	0.000547943	0.000547758
Gen 2	0.000547943	0.000547758
CPU Time	29656.8	27115.3

Table 4.8: FTSE: Child-loop 1 vs Child-loop 2

Both child-loop converged and found their best solution in three generations. However, child-loop 2 performed slightly better as the CPU time is shorter by 2541.5 and was still able to get a better index tracking error as it is smaller by 0.000021727. This means that child-loop 2 was able to find a better portfolio that more closely tracks the market index while using less computational resources compared to child-loop 1.

Comparing the solution limit

Generation	Index Tracking Error (Solution Limit = 20)	Index Tracking Error (Solution Limit = 10)
Gen 0	0.000552032	0.00062977
Gen 1	0.000547943	0.000549406
Gen 2	0.000547943	0.000547758
Gen 3	-	0.000547758
CPU Time	29656.8	17536.1

Table 4.9: FTSE: Solution limit 20 vs Solution limit 10

Unsurprisingly, the iteration with the lesser solution limit took one more generation to find the best solution, as the overall population was worse at the start. Decreasing the limit takes more generations for the population to converge to the best solution. However, while it took more generations, it was able to find a better solution, 0.000547758, in roughly half the CPU time of the higher solution limit.

4.2.4 S&P 500

S&P 500 is the USA market index; it has 503 stocks, but we removed Organon and Constellation Energy as we had incomplete data for the past two years. Due to a larger market, more assets were allowed, fifty assets to be specific, to build each portfolio since the cardinality was set at fifty.

Generation	Index Tracking Error (Year 1)	Index Tracking Error (Year 2)
Generation 0	0.000385007	0.000792847
Generation 1	0.000329491	0.000751972
Generation 2	0.000286854	0.000743443
Generation 3	0.000282461	0.000741659
Generation 4	0.000278454	0.000741659
Generation 5	0.000278454	-
CPU Time	29656.8	66381.4

Table 4.10: S&P: Year 1 vs Year 2

Comparing the years

As we stated before, we expect the year 1 and year 2 values to be similar; however, this is also not the case in this market. The difference between the two years is 0.000463205. The first year was completed in five generations, and the second year in four generations. The CPU time is quite interesting; in the smaller market, the CPU time was very short; however, in S&P, the CPU time of year 2 is twice as long as year 1.

Comparing the child-loop

Generation	Index Tracking Error (Child-loop 1)	Index Tracking Error (Child-loop 2)
Gen 0	0.000385007	0.000385007
Gen 1	0.000329491	0.000325309
Gen 2	0.000286854	0.000302768
Gen 3	0.000282461	0.000281021
Gen 4	0.000278454	0.000281021
Gen 5	0.000278454	-
CPU Time	92738.7	91472.4

Table 4.11: S&P: Child-loop 1 vs Child-loop 2

While child-loop 2 worked better in the FTSE, it performed worse in this market, as the best solution is 0.000281021, which is larger than that of child-loop 1, 0.000278454. However, it took slightly less CPU time and one less generation than child-loop 1.

Comparing the solution limit

While both iterations of the GA are close in value, the one with solution limit 10 did worse, which is to be expected given the solver's more lenient approach to asset selection. However, it was unexpected that they took the same number of generations to find their best solution as the population with the lower solution limit started with an overall higher index tracking error, so it was expected

Generation	Index Tracking Error (Solution Limit = 20)	Index Tracking Error (Solution Limit = 10)
Gen 0	0.000385007	0.000451963
Gen 1	0.000329491	0.000354975
Gen 2	0.000286854	0.000306018
Gen 3	0.000282461	0.000295534
Gen 4	0.000278454	0.000292391
Gen 5	0.000278454	0.000292391
CPU Time	92738.7	45158.2

Table 4.12: S&P: Solution limit 20 vs Solution limit 10

that the GA would take longer to converge to the best solution. Looking at the CPU, we see that the iteration with solution limit 10 converged to the best solution in half of the time of the iteration with solution limit 20. Regardless, the solution is worse.

4.2.5 Benchmark

This section will look at years one and two and compare them to the benchmark, calculated using a branch-and-bound solver with the stopping criteria, solution limit 30 and time limit 1000. As said in section 3.1, we aim to meet the benchmark for the smaller market and hopefully outperform the benchmark in the larger two markets. As we had many different values for year 1, the best value for each market will be compared to the benchmark.

Year 1

Market	Benchmark Value	CPU Time	GA	CPU Time
CAC	0.000889017	746.578	0.000889017	2094.83
DAX	0.001746129	84	0.00174613	654.766
FTSE	0.000559224	715.312	0.000547758	17536.1
S&P	0.000372042	4004.06	0.000278454	92738.7

Table 4.13: Benchmark vs Year 1 GA values

As expected, we could not outperform the benchmark but meet it with the smaller two markets, CAC and DAX. Comparing the GA value to the benchmark value, we outperformed it with a slight difference of 0.00001466. The GA value worked exceptionally well in S&P and is significantly smaller than the benchmark value, with a difference of 0.000093588. Comparing the CPU time, it is seen that the benchmark value is calculated much more quickly. In the benchmark and the GA, DAX took the shortest amount of time as it is the smallest market out of all four. Although, it is interesting that it is much faster than CAC as they have a similar market size. While the CPU time for FTSE and CAC took a similar time, FTSE took much longer in the GA as its market size is over twice that of CAC. As expected, S&P took the most time in the benchmark and the GA, as it is the biggest market looked at. The FTSE GA value was from the iteration of child-loop 1, solution limit of 10. For CAC, DAX and S&P, we used the value from child-loop 1, solution limit 20.

Year 2

Market	Benchmark Value	CPU Time	GA	CPU Time
CAC	0.00179318	0.21875	0.00179318	9.95312
DAX	0.00429729	0.21875	0.00429514	2.39062
FTSE	0.000599535	541.062	0.000526216	41699
S&P	0.0008246	2740.38	0.000741659	66381.4

Table 4.14: Benchmark vs Year 2 GA values

While the year 2 results differ greatly from the year 1 values, the year 2 values are similar to the benchmark, so we know an error does not cause the difference in the coding. Similar conclusions are reached by comparing the results of the year 2 GA and the benchmark. We again met the benchmark with the smaller market, CAC. However, DAX performed slightly better, with a difference of 0.00000215. The GA outperformed the benchmark in FTSE and S&P significantly, with the difference in FTSE being 0.000073319 and S&P 0.000082941. Looking at the CPU time, the GA in year 2 was quick, but the benchmark was calculated significantly quicker. The shortest time for the GA and benchmark was CAC and DAX, as they are the smallest market. As DAX is slightly smaller, in the GA, it was slightly quicker than CAC, even though they have identical values in the benchmark. As stated before, S&P took the longest as it is the largest market.

4.3 Experimentation: Discussion

The objective of our dissertation was to use GA to construct a portfolio that tracks the market index, which we consider an efficient form of investment according to our literature review. The results indicate that our GA code performed well, meeting the benchmark in smaller markets and outperforming it in larger markets.

The two years had completely different values, so our expectations were unmet. They should have been similar in value as the market should be alike in a similar timeline. This attribute is most likely due to the pandemic's impact on the market (as the data was taken from 14/10/2020 to 14/10/2020), resulting in high systematic risk in the first year, which decreased in the second year as the impact of the pandemic lessened. The CPU in the second year was also significantly faster. This could imply that the GA works effectively in a timeline without an event considerably impacting the market. However, the values of FTSE (Table 4.7) were similar, and the CPU time in year 2 was longer than in year 1, but we cannot determine the exact reason for these differences compared to the other markets.

In comparing the two different child loops, we found no difference between the solutions in smaller markets, but we observed varying results in larger markets. Child-loop 2 performed better in the FTSE (Table 4.8), whereas child-loop 1 performed better in the S&P (Table 4.11). Thus, both work well in finding reasonable solutions in the small markets, but which is better cannot be concluded due to the mixed results in the larger markets. However, child-loop 2 required less computational time to find the optimal solution in all markets except DAX (Table 4.5). Given that time was a major issue, this demonstrates that child-loop 2 was more effective than child-loop 1 in this aspect.

The special portfolios (portfolios 1 and 2) functioned well, consistently being among the ones with the lowest index tracking error, especially portfolio 2 (LP was relaxed), which was nearly always the best portfolio out of the parents. The first portfolio (assets chosen based on capital market value) did not work so well with the smaller markets but was still high enough that it was likely to be bred for the next generation; in the larger markets, portfolio 2 performed well, frequently ranking second in generation 0. Introducing these portfolios aided the GA in looking in the relevant search space, thus finding an appropriate solution faster.

While we obtained good results, the study had some limitations. For instance, the codes took over three hours to run for the larger market, so the time limit and solution limit had to be reduced to shorten computation times, causing us to lose on a potentially smaller index tracking error. This is uncharacteristic of GA, as this method is well-known for being quick at finding solutions. However, this might be an effect of working with a huge dataset as the computational time with the smaller data set was quick.

Adjusting the solution limit saved computation time. Nonetheless, it resulted in worse overall starting population values, requiring extra generations for the GA to converge to its optimal solution. More generations can result in premature convergence, as seen with S&P, as it signifies we are relaxing the strictness of our fitness value. However, in the FTSE (Table 4.9), the iteration with a lower solution limit outperformed the one with a higher limit.

When comparing our GA values to the benchmark, our expectations were met, meeting the benchmark in the smaller markets and outperforming in the larger markets. Except for the second year, when we beat the benchmark in the smaller market, DAX (Table 4.14), this is positive as it demonstrates that the GA is quite powerful. Overall, we successfully formed a genetic algorithm with an intelligent population to create a good portfolio.

Additionally, the GA code's lack of operation mutation may have caused the code to converge at a local optimum, resulting in worse values in some iterations as we cannot maintain the population's diversity, which leads to early convergence. To improve the GA performance, adding mutation to the GA code can help maintain diversity and prevent the code from converging at a local optimum.

As stated earlier, it needed to be more conclusive which child loop was better. To find out which is better, changing the solution and time limit is an option to see how the child loops affect population values—the child loop producing the smaller index tracking error being the better one. Also, combining the two child loops could lead to better values.

The larger markets produced better values than the benchmark because we had more assets. As stated in section 2.1.4, the more assets we have, the easier it is to diversify, therefore, minimising risk. Thus, following the market and getting a smaller index tracking error is easier. The problem is that practically most portfolios are about 20 to 30 assets [9, 23]. In S&P, we chose 50. It would be interesting to decrease the cardinality value and see if the GA still works as well.

The timeline chosen could have been better as it clearly affected the results of the GA, giving us some unexpected conclusions. A timeline with no sudden change in systematic risk should be chosen to overcome this limitation.

Chapter 5

Conclusion

The research aimed to create a portfolio using an optimisation model that tracks the index, with the help of genetic algorithms, to pick a set of assets that closely follows the market. Our analysis showed that these portfolios either met or outperformed the benchmark in all four markets, indicating the effectiveness of our approach in selecting a set of assets that closely follows the market.

Genetic algorithm was used because no perfect methods exist to form an optimal portfolio. Moreover, GA is remarkable because of its ability to search quickly through the search space and exploit an optimal solution, so the genetic algorithm cuts down time.

As expected, we were able to meet the smaller markets as its primary purpose was to test the codes due to their quick run time. In contrast, with the larger portfolios, we hoped to outperform the benchmark as we had a larger cardinality for the portfolios without having a subset similar in size to the market, so minimising systematic risk through diversification was more manageable. However, we did outperform in DAX in year 2, showing that our GA is quite robust. The comparison between year 1 and year 2 was unexpected as they differed significantly, but this could be due to the data being taken during the pandemic, affecting the market risk. Therefore, the market was not stable within these two years hence the difference.

The decision to utilise an index tracker was based on our CAPM study in section 2.1.5, which demonstrated the market's efficiency. Since an index tracker closely follows the market, it was deemed a fitting choice. However, our study should have delved deeper into the practical effectiveness of investing in the market beyond simply tracking the market index. For instance, the market index can sometimes yield negative returns. Although we had a constraint to prevent this for the expected return, we still need to ensure that our portfolios deliver satisfactory returns.

Moreover, while the intelligence in the population of our GA helped us lead to good values and shorten the CPU time, we need to understand how much our code relies on the solver. More assets could be picked by the GA instead of the solver to see how much is relied on by the GA.

To further improve our analysis, future studies could also explore the effectiveness of our GA against other GAs, such as MOGA and hybrid genetic algorithms or compare it with other portfolio optimisation methods. Alternatively, we could combine these methods to form an even more powerful GA, though adding more complexity can be computationally expensive.

Overall, using an intelligent population in our GA resulted in small index tracking errors, which is promising. However, further research is required to fully understand the implications of our results and identify opportunities to enhance the performance of our GA.

Bibliography

[1] Team TI. Modern portfolio theory: What MPT is and how investors use it. Investopedia; 2021. Visited on 2022-10-13. Available from: https://www.investopedia.com/terms/m/modernportfoliotheory.asp.

[2] Wirsansky E. Hands-on genetic algorithms with python. Packt; 2020.

[3] ;. Visited on 2023-03-18. Available from: https://www.ibm.com/docs/en/icos/12.10.0?topic=concepts-branch-cut-in-cplex.

[4] About Us;. Visited on 2023-03-08. Available from: https://www.investopedia.com/terms/a/asset.asp.

[5] Markowitz H. Portfolio Selection. The Journal of Finance. 1952;7(1):77-91. Available from: http://www.jstor.org/stable/2975974.

[6] What is market cap?;. Visited on 2023-02-26. Available from: https://www.fidelity.com/learning-center/trading-investing/fundamental-analysis/understanding-market-capitalization#:~:text=Market\%20cap\%E2\%80\%94or\%20market\%20capitalization,market\%20cap\%20of\%20\%241\%20billion.

[7] Fernando J. Market capitalization: How is it calculated and what does it tell investors?. Investopedia; 2023. Visited on 2023-02-15. Available from: https://www.investopedia.com/terms/m/marketcapitalization.asp.

[8] Elbannan MA. The capital asset pricing model: an overview of the theory. International Journal of Economics and Finance. 2015;7(1):216-28.

[9] Watson D, Head A. Corporate finance: principles and practice. Eighth ed. Harlow, Essex, England: Pearson Education Limited; 2018.

[10] Ganti A. Efficient frontier: What it is and how investors use it. Investopedia; 2022. Visited on 2022-10-13. Available from: `https://www.investopedia.com/terms/e/efficientfrontier.asp`.

[11] Kenton W. Capital Asset Pricing Model (CAPM) and assumptions explained. Investopedia; 2022. Visited on 2022-10-16. Available from: `https://www.investopedia.com/terms/c/capm.asp#:~:text=The\%20capital\%20asset\%20pricing\%20model\%20\%2D\%20or\%20CAPM\%20\%2D\%20is\%20a\%20financial,to\%20the\%20market\%20(beta).`

[12] Suárez RT. A hybrid optimization approach to index tracking. Annals of Operations Research. 2008.

[13] Beasley D, Bull DR, Martin RR. An overview of genetic algorithms: Part 1, fundamentals. University computing. 1993;15(2):56-69.

[14] Sourabh Katoch VK Sumit Singh Chauhan. A review on genetic algorithm: past, present, and future. Multimedia Tools and Application. 2020.

[15] Zanjirdar M. Overview of portfolio optimization models. Advances in mathematical finance and applications. 2020;5(4):419-35.

[16] Kolay S, Boulay R, d'Ettorre P. Regulation of ant foraging: A review of the role of information use and personality. Frontiers; 2020. Visited on 2023-03-06. Available from: `https://www.frontiersin.org/articles/10.3389/fpsyg.2020.00734/full`.

[17] El-Mihoub TA, Hopgood AA, Nolle L, Battersby A. Hybrid Genetic Algorithms: A Review. Eng Lett. 2006;13(2):124-37.

[18] Murata T, Ishibuchi H, et al. MOGA: multi-objective genetic algorithms. In: IEEE international conference on evolutionary computation. vol. 1. IEEE Piscataway, NJ, USA; 1995. p. 289-94.

[19] Bermúdez JD, Segura JV, Vercher E. A multi-objective genetic algorithm for cardinality constrained fuzzy portfolio selection. Fuzzy Sets and Systems. 2012;188(1):16-26.

[20] Cplex for AMPL;. Visited on 2023-03-18. Available from: `https://ampl.com/products/solvers/solvers-we-sell/cplex/`.

[21] Fourer R, Gay DM, Kernighan BW. AMPL: a modeling language for mathematical programming. 2nd ed. London;Pacific Grove, CA;: Thomson/Brooks/Cole; 2003.

[22] About Us;. Visited on 2023-03-03. Available from: `https://www.refinitiv.com/en/about-us#:~:text=Refinitiv\%2C\%20an\%20LSEG\%20`.

[23] Webb MS. Messy portfolios and the 'be busy' syndrome. Financial Times. 2014 May.