Portfolio Optimization Using Genetic Algorithm

Rethyam Gupta
Physics Department
IIT Guwahati
Guwahati, India
g.rethyam@iitg.ac.in

Dr. Prakash Kotecha
Chemical Department
IIT Guwahati
Guwahati, India
pkotecha@iitg.ac.in

Purvarth Raj Chaudhary
Chemistry Department
IIT Guwahati
Guwahati, India
c.purvarth@iitg.ac.in

***Abstract*—** Portfolio optimization is crucial for investors seeking to maximize their returns while minimizing risks. Genetic algorithms (GA) are powerful optimization tools that can solve this problem. This paper presents a GA-based approach to portfolio optimization that can handle many assets and constraints. Our method is designed to find the optimal combination of assets that maximizes the return on investment while adhering to certain constraints, such as the maximum allowable investment in any given asset or sector.

Keywords—Portfolio, Optimization, Genetic, Algorithm, Asset, Investment, Risk.

I. INTRODUCTION

Portfolio optimization is a fascinating area of study in financial mathematics. Since the introduction of Modern Portfolio Theory (MPT) by Harry Markowitz, numerous analytical and numerical methods have been explored to construct the best investment portfolio with a defined set of assets. Genetic algorithms (GA) provide a powerful means for optimal portfolio construction.

To tackle this optimization problem, Harry Markowitz developed a quantitative model in 1959, which is commonly known as the mean-variance model. The mean-variance model is typically used to minimize an objective function representing the portfolio variance (risk) for a given level of return or to maximize an objective function representing the portfolio return for a given level of risk.

A. Problem Statement

Suppose we have identified N financial assets we wish to invest in, such as stocks, funds, bonds, or ETFs. Each asset has a historical return, which refers to the relative price difference between two periods: days, weeks, or months. Our objective is to construct an investment portfolio comprising several assets and allocate a fraction x of total capital to each asset, known as the weight. The aim of portfolio optimization is to determine the optimal weights for each asset that maximizes returns and minimizes risk while adhering to certain constraints and then compare it to standard MPT.

B. Solving Approach

Our approach involves utilizing the Genetic Algorithm to identify the optimal weights for maximizing returns and minimizing risk. The Sharpe Ratio will serve as the fitness measure to assess the effectiveness of the approach

This flow chart on the right outlines the main algorithmic steps.

Image Ref- https://in.mathworks.com/help/gads/gaflowchart.png

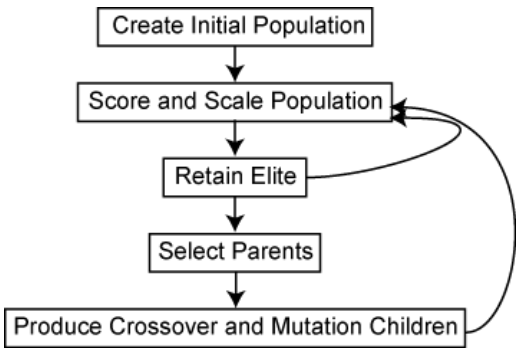
C. Why only GA and not PSO for the study

The effectiveness of Particle Swarm Optimization (PSO) in unconstrained portfolio optimization has been demonstrated in various studies. One such study conducted by Kamali in 2014 revealed that PSO outperformed Genetic Algorithms (GA) in terms of obtaining higher-quality portfolios in a shorter amount of time. In another study by Zhu et al. in 2011, PSO was found to be a superior algorithm compared to a GA and an industry tool for portfolio optimization. Additionally, PSO was shown to achieve better consistency and faster convergence towards an optimal solution for a behavioral portfolio model than a GA. Hence, we will focus on GA for this paper and will compare the Genetic Algorithm method to the classical Modern Portfolio Method.

II. BASIC GENETIC ALGORITHM

The Genetic Algorithm is a powerful optimization technique that can handle constrained and unconstrained optimization problems. It operates on the principle of natural selection, which underpins biological evolution. The algorithm iteratively modifies a population of individual solutions. At each iteration, it selects individuals from the current population to serve as parents and utilizes them to generate offspring for the next generation. This process continues until the population "evolves" towards an optimal solution.

The Genetic Algorithm can address a wide range of optimization problems that standard optimization algorithms may not be well-suited for. These include problems where the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear. Additionally, the algorithm can tackle problems in mixed-integer programming, where some components are constrained to be integer-valued.



A. Steps Involved in Genetic Algorithm

The Genetic Algorithm employs three primary rules at each iteration to generate the next generation from the current population:

Selection rules choose the individuals (i.e., parents) contributing to the population at the next iteration. The selection process is typically stochastic and may consider the scores of the individuals.

Crossover rules combine two parents to produce offspring for the next iteration.

Mutation rules introduce random alterations to the individual parents to generate offspring for the next iteration.

To learn more about how Genetic Algorithm works in detail, kindly refer to the link given –

https://in.mathworks.com/help/gads/how-the-genetic-algorithm-works.html

III. DATA AND DETAILED APPROACH

The dataset consists of monthly closing stock prices for six companies: MRF, Infosys, Reliance, Coal India, SBI, and Titan. The data spans a period of four years, providing a comprehensive picture of the performance of these companies in the stock market over a significant time frame. This data can be utilized for various financial analysis tasks, including portfolio optimization and risk assessment.

The date can be downloaded by the following link –

https://drive.google.com/drive/folders/1qtsgKYbJDYVOvJ1j58lYwq2tyk5IueIQ?usp=sharing

A. Detailed Process

Firstly, the data for the monthly closing stock values of MRF, Infosys, Reliance, Coal India, SBI, and Titan are combined into a single data frame. Historical returns for different periods ranging from 3 months to 36 months are then calculated for each of the stocks. The next step is to define the genetic algorithm terms - Gene and Chromosome. A gene is a numerical value that indicates the proportion of total capital allocated to a particular stock, while a chromosome is a one-dimensional collection of genes that represents the proportion of total capital allocated to each stock. It is crucial to maintain the sum of all genes within a chromosome to be equal to one.

Once the concepts of Gene and Chromosome have been established, the subsequent stage involves creating an Initial Population, which is a two-dimensional collection of randomly generated chromosomes. The fitness function is then computed, typically through the Sharpe ratio (S) calculation. The Sharpe ratio is a measure that quantifies the portfolio's performance by maximizing return and minimizing risk simultaneously. It is computed as the difference between the portfolio return and the risk-free rate, divided by the standard deviation of the returns. The fitness function uses the mean portfolio return, risk-free rate, and the standard deviation of the portfolio return to calculate the Sharpe ratio.

The next step is to select the Elite Population, which filters the elite chromosomes with the highest returns, as calculated in the fitness function. Mutation and crossover are two essential steps in the genetic algorithm. A mutation function randomly selects two numbers between 0 and 5 and swaps those elements in a chromosome. The heuristic crossover, also known as blend crossover, utilizes the fitness scores of two parental chromosomes to determine the direction. Offspring are created based on a random number, β, between 0 and 1.

Finally, the Next Generation function is defined, which performs mutation, mating, or crossover based on probability and builds a new generation of chromosomes. The entire process is iterated until there is no change in the maximum returns or for a fixed number of iterations.

The process with Equations –

1) Combine the data into a single table and calculate the historical returns for different time periods for each stock.

2) A gene refers to the portion of the overall capital allocated to a particular stock, while a chromosome is a collection of genes that signify the proportions of the total capital assigned to each stock. It is a prerequisite for the summation of all genes in a chromosome to be equal to 1.

3) To ensure that the sum of six random numbers is equal to one, one can first generate the random numbers and then calculate a factor by dividing one by the sum of the random numbers. The generated random numbers can then be multiplied by this factor, which will result in the sum of the random numbers being equal to one.

4) Generate an initial population of chromosomes randomly. The Sharpe ratio is a fitness function that measures the portfolio's performance by maximizing returns and minimizing risk simultaneously. It is calculated as:

   S = (Mean Portfolio Return - Risk-free Rate) / Standard Deviation of Portfolio Return.

   $S = (\mu - r)/\sigma$

   The symbol μ represents the mean return of a portfolio for a specific time frame, while r denotes the risk-free rate for that same period. Lastly, σ refers to the standard deviation of the portfolio's returns during that specified time frame.

   Basic Equations related to Portfolio Optimization-

   Mean portfolio return - Mean Return * Fractions of Total Capital (Chromosome).

   Risk-free rate - 0.0697 (Market Risk-Free Rate)

   The standard deviation of portfolio return - (Chromosome * Standard deviation) ** 2 + Covariance * Respective weights in the chromosome.

5) Select the elite population by filtering the chromosomes with the highest returns based on the fitness function. The mutation is a function that

randomly selects two elements in a chromosome between 0 and 5 and swaps them.

6) Heuristic crossover or Blend Crossover is a method that uses the fitness values of the two parent chromosomes to determine the direction of the search.

The offspring are created using the equation:

Offspring A = Best Parent + β * (Best Parent - Worst Parent)

Offspring B = Worst Parent - β * (Best Parent - Worst Parent)

7) Where β is a random number between 0 and 1, the next generation is a function that creates a new generation of chromosomes by performing mutation, mating, or crossover based on probability.

8) Iterate the entire process until there is no further improvement in the maximum returns or for a fixed number of iterations.

IV. PORTFOLIO SELECTION MODEL

We are examining a portfolio comprising of n weights (W_1,W_2, …., W_n), where W_i represents the proportion of total invested wealth by the investors in asset i, r_i is a random variable that indicates the expected return of asset i, and n denotes the total number of assets in the portfolio. The portfolio's return, r_p, is determined by the following equation:

$$rp = W1r1 + W2r2 + \cdots + Wnrn = \sum_i^n Wiri \quad (1)$$

$$E(rp) = E[\sum_i^n Wiri\,] = \sum_i^n WiE(ri) \quad (2)$$

We let, $E(rp) = R \; and \; E(ri) = \mu i$

$$R = \sum_i^n Wi\mu i \quad (3)$$

$$\sigma^2 p = E[(rp - \mu p)^2] \quad (4)$$

$$\sigma^2 p = \sum_i^n Wi^2\sigma^2 + 2\sum_i^{n-1}\sum_{j=i+1}^n WiWj\sigma ij \quad (5)$$

But, var $(ri) = \sigma ii$

$$Var(P) = \sum_{i=1}^n \sum_{j=1}^n WiWj\sigma ij \quad (6)$$

Now, The Sharpe Ratio serves as the fitness measure to assess the effectiveness of the approach as discussed in the detailed process section, where the constraints include budget constraint, cardinality constraint, and floor and ceiling constraints. The budget constraint ensures that the investor allocates the entire budget to a portfolio. The cardinality constraint ensures that a specific number of assets, K, are included in the portfolio. The floor constraint, which is a lower-bound constraint, helps avoid administrative costs for very small holdings. In contrast, the ceiling constraint, an upper-bound constraint, avoids excessive exposure to a particular asset.

$$\sum_i^n Wi = 1 \quad (7)$$

Here, *Wi* is the weight assigned to *ith* asset.

$$\sum Zi = K \quad (8)$$

Here, *K* is the sum of the total number of assets available. In our case, the value of *K* is 6.

$$0 < Wi < \{0.25, 0.35, 0.45\} \quad (9)$$

Here *Wi* is the weight assigned to the *ith* asset, and we have set an upper and a lower bound. The lower bound is set to 0 for obvious reasons, whereas we have kept a floating upper bound to make a case study of the effect of the upper bound on the results.

A. Assets available for Portfolio Selection

We randomly chose six companies from the National stock exchange (NSE) market (MRF, Infosys, Reliance, Coal India, SBI, and Titan). We then gathered the historical prices of these companies for four years, from 2015 to 2018, and calculated their historical returns for the 3, 6,12, 24, and 36-month period. The calculation for a stock's overall return involves adding the price increase with any dividends distributed and then dividing it by the stock's initial price. We assessed the portfolio variance, mean, and standard deviation using the annual average return data provided.

Historical Return Data for the available stocks in months.

Mon	MRF	INFY	REL	COAL	SBI	TITAN
3	0.114515	0.039664	-0.030947	0.211564	0.13151	0.296599
6	0.125163	0.011212	0.011417	0.194062	-0.0179	0.368094
12	0.275866	-0.178478	0.129783	0.330899	0.010911	0.562537
24	0.792712	0.084237	0.274461	0.255179	-0.26591	0.44833
36	0.974847	0.266844	0.069614	0.399938	-0.35878	0.447535

Mean and Standard Deviation of Historical Returns.

STOCKS	MEAN	COVARIANCE
MRF	0.456621	0.400340
INFOSYS	0.044696	0.159583
RELIANCE	0.090865	0.119189
COAL	0.278328	0.086091
SBI	-0.100047	0.204405
TITAN	0.424619	0.099615

The Mean and Standard Deviation of stock prices remain unchanged for the "Gene" and the "Chromosome" factors and only depend on the monthly closing price data available.

Next, we will show the covariance matrix of returns of the chosen assets, and then we will assess the results over a different set of conditions.

Covariance Matrix of Historical Returns.

	MRF	INFY	REL	COAL	SBI	TITAN
MRF	0.000000	0.045393	0.027916	0.024127	-0.079078	0.014362
INFY	0.045393	0.000000	-0.000718	0.004381	-0.023178	-0.005554
REL	0.027916	-0.000718	0.000000	0.002510	-0.013841	0.007330
COAL	0.024127	0.004381	0.002510	0.000000	-0.011042	0.005700
SBI	-0.079078	-0.023178	-0.013841	-0.011042	0.000000	-0.007211
TITAN	0.014362	-0.005554	0.007330	0.005700	-0.007211	0.000000

V. ASSESSING RESULTS FOR GA ON DIFF. PARAMETERS

Genetic Algorithms (GA) can be a powerful tool for identifying optimal investment portfolios in portfolio optimization. However, the effectiveness of the GA can depend on various parameters, such as the population size, mutation rate, and crossover rate. It is, therefore, crucial to assess the results of the GA on different parameter settings to identify the most effective configuration for a given portfolio optimization problem. For example, a larger population size may be required for more extensive portfolios, while a higher mutation rate may be necessary for portfolios with more complex constraints.

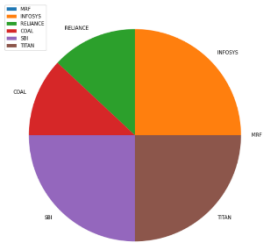
In the first study round, we kept the population size, mutation rate, and crossover rate as 100, 0.4, and 0.6, respectively. We specifically varied other parameters like upper bound on an individual weight constraint and the number of iterations. We will assess the effect of the before-mentioned parameters in the 2nd round of study.

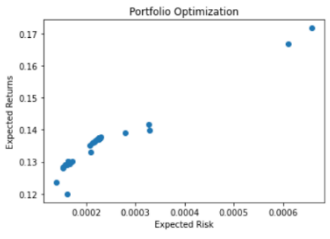
Note – The results in the upcoming section depend on the randomly generated gene, alpha, and beta values, which may vary from time to time. Hence, the effects observed in the report may not be observed again.

A. *Varying the maximum weight per asset*

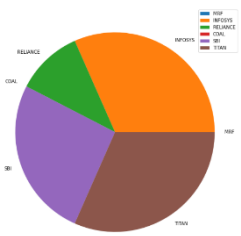
We varied only the upper bound of the maximum possible weight allotted to an asset. Initially, it was set to 25% of the overall portfolio, and then we studied it at 35% and 45% levels to understand the effect of diversification on an optimization problem.

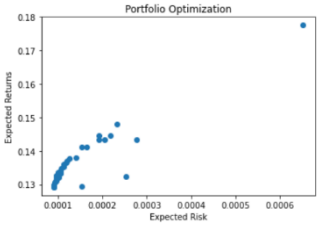
25% Study - Expected returns of 13.74636320442024 % with a risk of 0.00022742572753497916



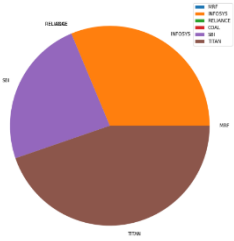


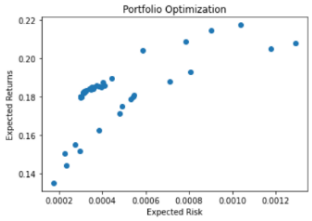
35% Study - Expected returns of 13.225226255278447 % with a risk of 9.573475690171154e-05





45% Study - Expected returns of 17.99140312833723 % with a risk of 0.00030152507352092413





Thus, it is immediately noticeable as we permit the maximum weight per stock. Few of the available stocks are assigned zero weight per the given constraints to provide the maximum returns with minimum risk involved. We can also notice that as we increase the maximum weight per stock, the value of expected return increases, and the risk involved increases.

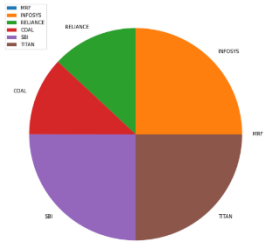
Moving forward, we will only use a 25% weight allotment per stock for maximum portfolio diversification and minimum risk involved.

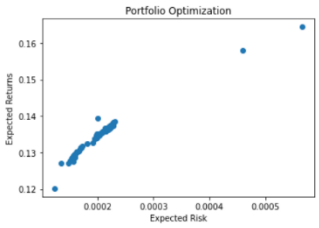
B. *Varying Population for Optimization*

Now, we will vary the population size to understand the effect of population variation on a portfolio optimization problem. Initially, we had a population of 100, but we will increase it to 500 with a population gap of 100 to study the effects.

Population Size = 200

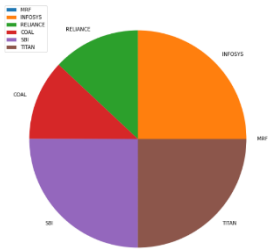
Expected returns of 13.745726348461783 % with a risk of 0.00022731441721664594

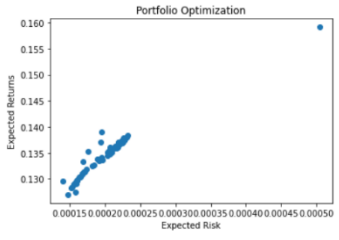




Population Size = 300

Expected returns of 13.74740786334796 % with a risk of 0.00022726051611534313





The effect of population variation was studied till the Population size reached 500, starting from 100, and the impact noticed was minimal. It can be ignored in our case of an unconstrained portfolio optimization problem.

C. *Varying Mutation and Crossover Rate*

In a Genetic Algorithm (GA) portfolio optimization problem, the mutation and crossover rate determine the extent of exploration and exploitation of the search space. These parameters play a critical role in shaping the diversity of the population and ultimately determining the quality of the solutions found.

Varying the mutation and crossover rate is essential for several reasons. Firstly, if the mutation rate is set too high, there is a risk of premature convergence, which means the algorithm may get stuck in a local optimum and fail to find the global optimal solution. On the other hand, if the mutation rate is too low, the search may not explore enough solution space, leading to a limited set of solutions.

Similarly, suppose the crossover rate is set too high. In that case, the population may quickly converge to a limited set of solutions, reducing diversity and preventing the algorithm from exploring other regions of the search space. On the other hand, if the crossover rate is set too low, the algorithm may fail to take advantage of the reasonable solutions that are already present in the population.

Therefore, varying the mutation and crossover rates during optimization is essential to maintain an appropriate balance between exploration and exploitation. This can be achieved by adjusting the mutation and crossover rates based on the algorithm's performance at each generation, aiming to achieve a diverse and high-quality population of solutions that represents the entire search space.

Initial conditions –

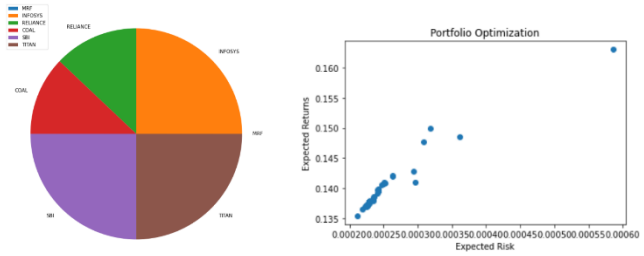
<u>Mutation probability = 0.4 and Crossover probability = 0.6</u>

We generated all the possible cases of pair of Mutation and Crossover rates and determined the best case for our problem.

For initial condition result – maximum weight variation result 1.

Mutation probability = 0.3 and Crossover probability = 0.7

Expected returns of 13.7710108495711 % with a risk of 0.000229173389885178



<u>Mutation probability = 0.2 and Crossover probability = 0.8</u>

Expected returns of 0.1374846286246595 with a risk of 0.00022756556016249267

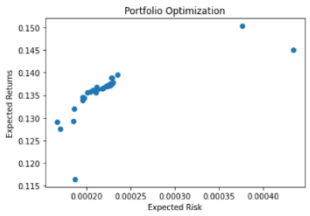
<u>Mutation probability = 0.1 and Crossover probability = 0.9</u>

Expected returns of 0.1374687033541035 with a risk of 0.00022745969357219377

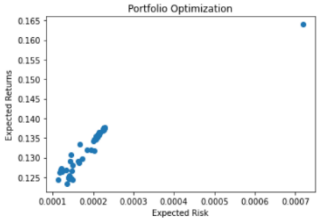
When the mutation probability is set too low below 0.3, we can see that the search is not exploring enough space.

<u>Mutation probability = 0.9 and Crossover probability = 0.1</u>

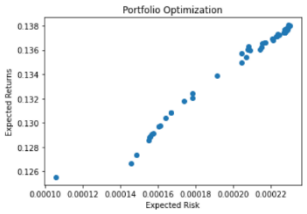
Expected returns of 13.74656823475979 % with a risk of 0.0002274395007818104



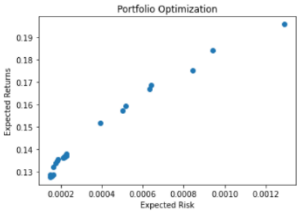
1. Pm = 0.9 & Pc = 0.1



2. Pm = 0.8 & Pc = 0.2



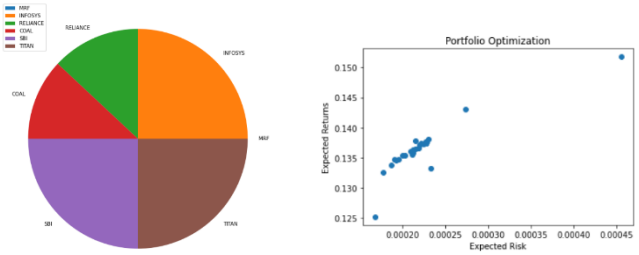
3. Pm = 0.7 & Pc = 0.3



4.. Pm = 0.6 & Pc = 0.4

<u>Mutation probability = 0.5 and Crossover probability = 0.5</u>

Expected returns of 13.746565506476635 % with a risk of 0.00022743931764917808



If changing the mutation and crossover probabilities does not result in any noticeable change in the fitness function of a Genetic Algorithm (GA), the algorithm may have reached the global optimum. However, this assumption should be made with caution, as the lack of improvement in the fitness function could also be due to a variety of other factors, such as a limited number of iterations, inadequate population size, or poor choice of selection or crossover methods.

In our problem, we have already verified the issue of iteration and inadequate population. We received similar results from both studies, but we have only used heuristic methodology for crossover and mutation. We will study arithmetic crossover in the next section and determine whether the obtained result from the heuristic crossover is the global optimum.

It is possible that there may be multiple optimal solutions in the search space, and the algorithm may have converged to one of them, but not necessarily the global optimum. Therefore, it is important to continue exploring the solution space using different parameter settings and techniques to ensure that the best possible solutions are found.

VI. ARITHMETIC CROSSOVER IN GENETIC ALGORITHM

Arithmetic crossover involves the creation of two offspring, represented as one-dimensional arrays, from two-parent arrays. The offspring arrays are created using the following equations:

Offspring A = α * Parent1 + (1 - α) * Parent2

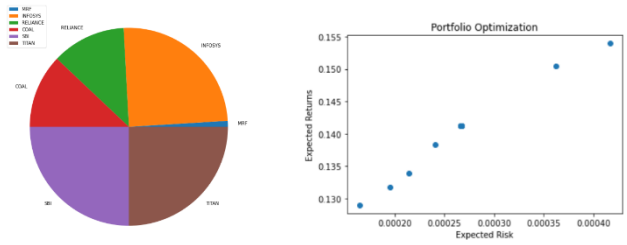
Offspring B = (1 - α) * Parent1 + α * Parent2

Here, the value of α is a randomly generated number between 0 and 1. The input to this process is two-parent arrays, and the output is two offspring arrays, each represented as a one-dimensional array.

As we noticed from the study with the Heuristic approach, the effect of the population was minimal on the result, and the impact of maximum weight per asset constraint was as expected, so for the arithmetic case, we will only study the effect of crossover and mutation probability and compare the two methods.

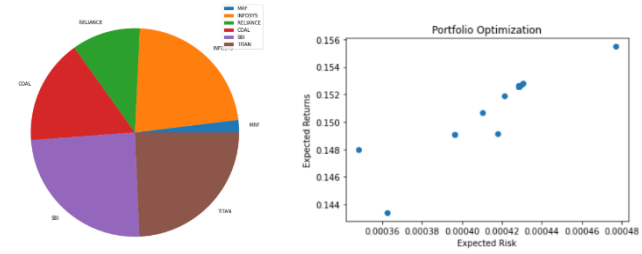
Mutation probability = 0.5 and Crossover probability = 0.5

Expected returns of 14.121406502636747 % with a risk of 0.0002667285639127272



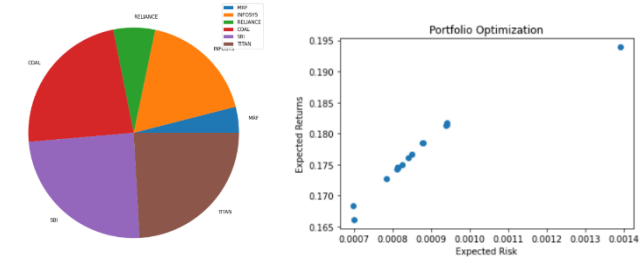
Mutation probability = 0.4 and Crossover probability = 0.6

Expected returns of 15.260398394608693 % with a risk of 0.0004285360534640407



Mutation probability = 0.6 and Crossover probability = 0.4

Expected returns of 17.457091965468567 % with a risk of 0.0008149555905863498



Comparing the above three results with the results of the heuristic method under the same parameters, one can conclude that the heuristic method is only approaching a local optimization point and not the global one. We can see that under the constraint of Pc = 0.5 and Pm = 0.5.

Heuristic Values - Expected returns of 13.746565506476635 % with a risk of 0.00022743931764917808

Arithmetic Values - Expected returns of 14.1214065026367 % with a risk of 0.0002667285639127272

Arithmetic Crossover is better than Heuristic Crossover for portfolio optimization problems for several reasons.

Firstly, Arithmetic Crossover produces children that are always a convex combination of the parent portfolios. In other words, the resulting child portfolio is guaranteed to lie within the range defined by the parent portfolios. This is an essential feature for portfolio optimization because it ensures that the resulting portfolios are always within the problem's constraints, such as the budget constraint or the limit on the number of assets.

On the other hand, Heuristic Crossover does not guarantee that the resulting portfolios will be within the problem's constraints. The heuristic method may produce a child portfolio that violates the constraints, leading to an invalid solution that cannot be used.

Secondly, Arithmetic Crossover generates more diverse and less correlated children than their parents. This diversity helps to prevent premature convergence and increases the likelihood of finding the global optimum. In contrast, Heuristic Crossover may produce highly correlated children with their parents, leading to a limited set of solutions.

Lastly, Arithmetic Crossover is a widely used and well-established method in portfolio optimization, with numerous studies demonstrating its effectiveness. In contrast, Heuristic Crossover used earlier in the paper is a relatively new and untested method, with limited research supporting its efficacy.

VII. COMPARISON WITH MODERN PORTFOLIO THEORY

Genetic Algorithm (GA) and Mean-Variance Portfolio Optimization (MVPO) are methods for constructing a portfolio of assets that maximizes the expected return for a given level of risk. However, GA and MVPO differ in their approach to the problem, and there are situations where one method may be more effective than the other.

MVPO assumes that asset returns are normally distributed and that investors are risk-averse and seek to maximize their utility function, which depends on both the expected return and variance of the portfolio. The optimization process involves finding the set of portfolio weights that minimize the portfolio variance for a given expected return or maximize the expected return for a given level of portfolio variance.

The advantage of GA over MVPO is that GA is more flexible and can handle non-normal and nonlinear relationships between assets. GA can also incorporate additional constraints and objective functions. Furthermore, GA can find global optima, while MVPO can only find local optima, which may lead to suboptimal solutions.

However, GA has some disadvantages compared to MVPO. First, GA is computationally expensive and requires many simulations to converge to a satisfactory solution. Second, GA is less transparent than MVPO and may produce difficult results to interpret and validate. Finally, GA requires careful tuning of the algorithm parameters to obtain good performance, which may be time-consuming and error-prone.

In conclusion, GA can be a powerful tool for portfolio optimization when the problem is complex. MVPO may be sufficient when the problem is more straightforward and can be approximated by the normal distribution assumption. The

choice between the two methods depends on the problem's specific characteristics and the investor's preferences.

A. Comparing Expected Returns and Markowitz Bullet

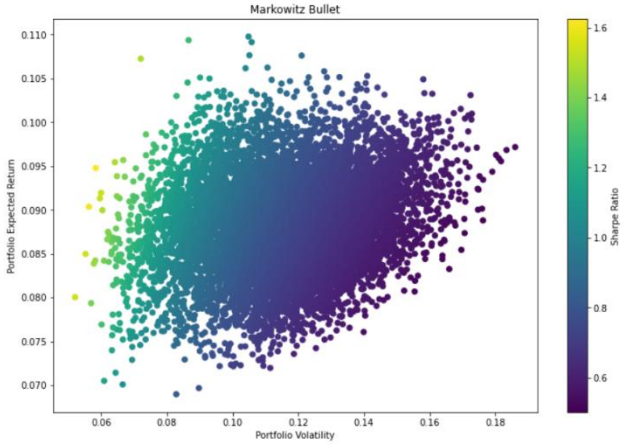
We have compared the standard mean-variance portfolio optimization with heuristic and arithmetic genetic algorithms based on the expected return factor for a given risk. The results show that the GA is a better method to work with a complex portfolio optimization problem because of the constraints involved.

As expected, it is because GA can incorporate additional factors such as weight constraint, objective function constraint, and other limitations discussed in the previous sections. In terms of expected returns, both methods aim to maximize the portfolio's expected return. However, GA may be more effective in generating higher expected returns since it can incorporate additional constraints and objective functions that may impact expected returns.

Regarding risk factors, MVPO minimizes portfolio variance, which measures risk. At the same time, GA considers risk as a constraint and aims to maximize returns subject to the risk constraint. This means that GA may generate portfolios with higher risk than MVPO but potentially higher expected returns.

We generated many random portfolios to generate a Markowitz bullet to display all possible portfolios under the modern portfolio theory with expected return on Y-axis and Volatility on X-axis.

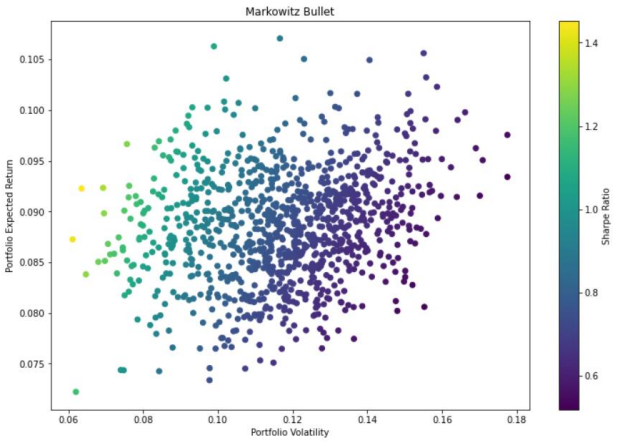
Markowitz Bullet - The Markowitz Bullet is a graphical representation of the set of efficient portfolios that can be generated through MVPO. Efficient portfolios offer the highest expected return for a given level of risk or the lowest risk for a given level of expected return.



Portfolio Expected Return: 8.833333333333333%

Portfolio Variance: 0.0004946111111111108

The code uses around 10000 iterations to generate the bullet-shaped figure with all the possible portfolios. The covariance matrix was provided to the code directly from the generated one in the genetic algorithm code. We can see that even the mean-variance portfolio optimization generates a portfolio with around 12 to 13 % returns, like the arithmetic genetic algorithm, which shows it is a possible portfolio. Still, both methods use different approaches to optimize a portfolio leading to other choices for the optimal portfolio.



The above figure only uses around 1000 iterations to generate the Markowitz bullet. The figure shows the optimized portfolio in yellow at the left center, with the maximum return for the minimum risk. There are also a few possible portfolios in green that have a higher return with very little difference in the risk value, but it is up to the algorithm to select the one with the yellow. One can manually choose other possible portfolios from the generated pool according to the investment needs.

VIII. Conclusion

In this paper, we compared the performance of a Genetic Algorithm (GA) with Heuristic and Arithmetic Crossover and Mean-Variance Optimization (MVO) for a Portfolio Optimization Problem. The aim was to find an optimal portfolio of stocks with maximum return and minimum risk using historical financial data.

Our experiments revealed that Arithmetic Crossover performed the best among the tested algorithms in generating more efficient portfolios with higher returns and lower risk. However, the results suggest that GA with Heuristic Crossover and MVO are promising approaches for solving portfolio optimization problems.

While our study shows that Arithmetic Crossover outperformed the other algorithms in this specific dataset, it is essential to note that the performance of the algorithms may vary in different universes or datasets. Further investigation is required to evaluate these algorithms' robustness and performance on various datasets.

Overall, our study contributes to the ongoing research on portfolio optimization and highlights the potential of different optimization techniques. The findings suggest that practitioners should carefully consider and test the algorithm on various datasets before making investment decisions.

Acknowledgment

The path to achievement involves groundwork, diligence, and taking lessons from missteps. We express our sincere appreciation to Dr. Prakash Kotecha for bestowing upon us the chance to work and acquire knowledge under his tutelage. We are lucky to have received his expert supervision, valuable counsel, and guidance throughout the project, enabling us to learn and shape the work we presented. His leadership, unwavering support, and recommendations were instrumental in making this project a reality.

REFERENCES

[1] T.J. Chang, N. Meade, J. Beasley, and Y. Sharaiha, Heuristics for cardinality constrained portfolio optimization, Computers and Operations Research, (27) (2000), 1271 - 1302.

[2] H. Markowitz, Analysis in portfolio choice and capital markets Oxford, Basil Blackwell, 1987.

[3] A.R. Pandari, A. Azar, and A.R. Shavazi, Genetic algorithms for portfolio selection problems with nonlinear objectives, African Journal of Business Management, 6(20) (2012), 6209-6216.

[4] Felix Roudier, Portfolio optimization and genetic algorithms, Master's thesis, Department of Management, Technology and Economics, Swiss Federal Institute of Technology (ETM), Zurich, 2006.

[5] P. Sinha, A. Chandwani, and T. Sinha, Algorithm of constructing an optimum portfolio of stocks using genetic algorithm MPRA Paper, 48204, (July 2013).

[6] S.M. Wang, J.C. Chen, H.M. Wee, and K.J. Wang, Nonlinear stochastic optimization using genetic algorithm for portfolio selection, International Journal of Operations Research, 3(1) (2006), 16 - 22.