Genetic Algorithms: Portfolio Investment

Computational Intelligence for Optimization

Group 8:
Ana Gonçalinho 20191210
Henry Tirla 20221016
Pedro Rodrigues 20220639

Link to repository: https://github.com/AnaGoncalinho/PortfolioInvestmentProblem

Portfolio Investment Problem

This problem involves finding the best combination of different assets in order to maximize the overall return of investment and at the same time minimize the risk involved with the investments. Due to the complexity of performing maximization and minimization approaches at the same time, we decided to take an approach of incorporating this multi-objective approach into the fitness function. We chose this idea because there were several objectives to be met in this problem, such as the number of investments that can be made and the maximum budget.

Research Question: *Given a set of investments (with expected return and risk), what is the combination of investments that optimize the return and the risk simultaneously?*

We established some restrictions during the development of this project:

- Each investment can only be made once, there cannot be repeated investments in the final set.
- The investor might have a maximum budget to spend on the investments. If the budget is exceeded, the solution is not viable. If there is no budget, every investment amount is accepted.
- The investor has a predetermined number of investment he/she wants to do. The number of investments in the final solution must be equal to the number of investments to be done.

Representation

Every individual is a set of 10 investments (it can be changed for future research). We chose 10 investments per individual because it is not too big, not requiring much computational effort, but at the same time it is a size that allowed us to understand if the operators were working correctly and how they were performing. Since we want to provide the best portfolio possible, we had to have a final solution as a set of investments, and not only one investment.

However, not every set of 10 investments represents a valid portfolio, since, as explained above, there are some requirements that must be met for the portfolio to be accepted. We decided to deal with this issue by penalizing invalid solutions in the fitness function. When the requirements for a portfolio aren't met, the fitness value for that individual will be highly penalized.

To initialize our population, we created 3 different datasets, to store each variable (investment, accumulative percentage return and risk). The first variable was created randomly within a range from 0 to 1000, representing the financial amount the investor must put in that investment. The second variable has values between 0 and 1, representing the percentage to add to the investment made. The final variable was created with values between 0 and 10, representing the risk associated with a specific investment. The values in position 1 in all of the variables represent the same investment, and so on. In total, for the purpose of testing our Genetic Algorithm, the population was created randomly, it isn't real data.

Fitness Function

As explained before, the portfolio investment problem involves finding the set of investments that maximize the return and at the same time minimizes the risk associated with the total investments. We are interested in a portfolio with maximum return and minimum risk.

The fitness function for this project is given by the Sharpe Ratio, that evaluates the investment performance based on risk-free return. This measure subtracts return of portfolio with risk-free rate and then divides the result by the standard deviation of the portfolio's excess return. This measure typically deals with time series data, which was not used in this project.

So, our fitness function was defined as the difference between the return of the portfolio and the risk involved in the portfolio. Since we want to obtain the maximum return and the minimum risk, the bigger the difference, the more likely we are to achieve those results. We also tried to divide this difference by the standard deviation of the returns, but the results obtained were worse than the ones obtained, since the second fitness function obtained values very close to each other.

Since our fitness function represented the difference between returns and risks, the bigger the fitness, the better the portfolio. However, there are other factor that influence the fitness of a portfolio, such as:

- Is the portfolio only composed of unique investments?
- Is the portfolio exceeding the maximum budget?

When one of the requirements isn't met, the fitness function is set to the difference between the risks and the returns, reaching negative values, making the portfolio less probable to survive. We knew that this was quite a simple fitness function, but since it provided us with good final results, we decided to keep it this way.

Selection Methods

Three selection methods were used for the development of this project. Fitness Proportional and Tournament selections were lectured, and the code was provided during classes, and we also implemented ranking selection.

Crossover Operators

Since our solutions all present the same size and the restrictions imposed in the beginning didn't lead to a huge number of invalid individuals, we were able to adopt the crossover methods learned through classes. We used single point, cycle, and arithmetic crossovers. We also tried using PMX crossover, but the results we were obtaining were quite strange and didn't provide good results, so we decided to drop it from the analysis.

Mutation Operators

For the mutation of the individuals, we also used the operators given in classes, with the exact same code provided in practical classes. All mutation operators seemed to work as expected and provided expected results. The mutation operators used in this project were swap mutation, inversion mutation, and scramble mutation.

Elitism

In order to compare even more options regarding the results provided by the different configurations, we decided to implement elitism and understand if its inclusion or exclusion impacted our Genetic Algorithm.

In our project, elitism was the removal of the worst individual from the new population, if its fitness was worse (in our case lower) than the fitness of the best individual of the current population.

Final Results

After the implementation of all methods are operators defined above, we tested the performance of every possible combination. Each combination was composed of one selection method, one crossover operator, one mutation operator and the presence of elitism. For all of the 54 combinations (3 selection x 3 crossover x 3 mutation x 2 elitism) we ran 100 iterations for 50 generations. We had to set other parameters for the Population evolution:

- Population size: 50
- Probability of crossover: 0.90 (90%)
- Probability of mutation: 0.05 (5%)
- Solution size: 10

For the performance evaluation we used several approaches. We started by computing the average fitness for all of the combinations, in order to find the combinations that tend to return solutions with higher fitness. For this evaluation we can see that the best configuration was composed of:

- Ranking Selection
- Arithmetic Crossover
- Swap Mutation
- Elitism

This was the configuration with the best average fitness score, but the configurations in the following ranking places had very similar performances. Due to that, we cannot conclude with 100% certainty that this is the best configuration for our problem. But, from the analyses of all configurations we can see some patterns regarding the best results, while for the worst doesn't seem to have any pattern.

For the best configurations, the methods and operators tend to be:

- Ranking Selection
- Single Point Crossover

Regarding the convergence of our Genetic Algorithm. All of the tested configurations led to a convergence in the results, so, no configuration negatively affected the convergence of our Genetic Algorithm.

As mentioned before, we implemented elitism in our project, that basically transfers the best individuals of a generation automatically to the next generation, without going through crossover or mutation. We analyzed the impact of implementing elitism, but there wasn't any pattern. The top three configurations use elitism, but the following four don't use elitism. On the other hand, the worse configuration doesn't use elitism, but the following four use. Having said this, we believe that there is no evident pattern that leads us to say that our Genetic Algorithm benefits a lot with the implementation of elitism.

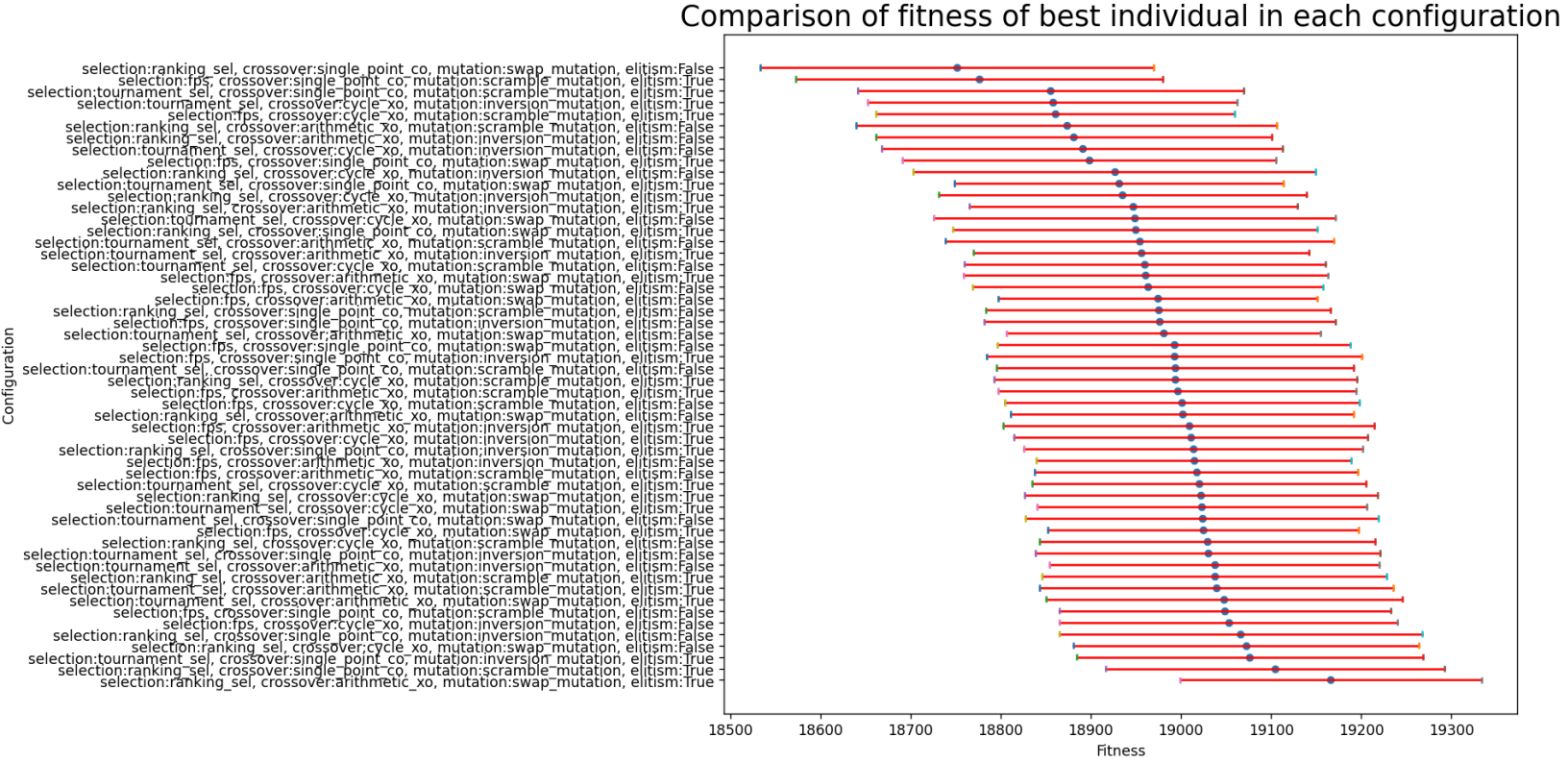

Figure 1 - Mean Fitness for each Configuration with Standard Deviation

As we can see if the figure above, the best configuration is composed of ranking selection, arithmetic crossover, swap mutation and elitism. We can see that there is not statistically significance that tells with 100% accuracy that the model in the bottom as the best fitness, but we can see with statistical significance that the last model is better than the first one, since there is no overlap between the mean with the standard deviations.

Implementation

During the development of this project, we always took into consideration that we should develop an abstract solution, able to work for both maximization and minimization problems. For that purpose, while defining the population and the selection methods, we always defined paths for both maximization and minimization problems. Regarding the fitness function, we believe that it doesn't need to be changed, since the minimization of this problem involves minimizing the investment.

Future Research

We believe that we found good results, since we achieved quite high fitness results. High fitness results mean high investment returns and low risks involved. We believe that the fitness and respective results could be improved regarding the budget the investor has, since a higher budget allows a higher investment leading to a very probable high return.

Of course, as we know, nothing is perfect, meaning that neither is our solution nor results obtained. Improvements that could be made in future works would be the type of data. As mentioned before in this document, this type of problem usually deals with time series data, with several variables regarding return measures and risk involved. We suggest, for future research, implementing that type of data, using real data, not randomly generated, and the usage of the Sharpe Ratio as the fitness function, since it is the one used in higher complexity problems.

Division of Labor

Ana Gonçalinho did the report and the additional Python files (additional to the ones provided in classes). Henry Tirla helped in the implementation of the functions (get_fitness, portfolio_return, portfolio_investment and portfolio_risk). Pedro only helped to choose the first topic of this project, that was posteriorly changed.

Other visualizations are available with the code provided.