# Task-2

## Introduction

For this problem we'll have 20 stocks which include their stock data from Januray 1st 2018 to December 31st 2022. This include high, low, volume, opening and closing. By this data one can look at how each stock has acted for a certain time. In this task we'll use Evolutionary Programming(EP) and Evolutionary Strategies(ES), with some variation, and in total six different version.

## Methodology

The first version is about a basic version of EP. The EP algorithm evolves a population of candidate solutions by generating off spring through mutation. Each generation, the population is ranked by fitness, and the top candidates are selected to survive and reproduce.

The second version is also about EP, but here we implemented a more complex version of it. For Self-Adaptive Mutation, Mutation rates adapt dynamically to improve the balance between exploration and exploitation. For elitism, the top-performing individuals in each generation are retained in the population to prevent loss of optimal solutions. For selection mechanisms, Offspring are selected based on a probability distribution proportional to fitness, favoring high-fitness solutions while maintaining diversity.

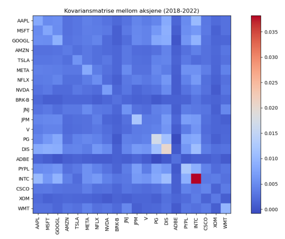
The third one, is the first one in ES, and is a basic implementation of this. For mutation, Gaussian mutation is applied to each candidate solution to explore the search space. For selection, each generation, the top-performing individuals are selected to form the population for the next generation, promoting gradual improvement in fitness.

The fourth one, is an advanced version of ES. For self-adaptive mutation rates, each candidate solution has an adaptive mutation rate that evolves alongside the portfolio weights. For recombination, candidate solutions are recombined from selected parent weights and mutation parameters, promoting exploration of the search space.

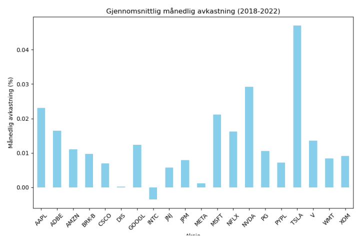
The fifth version is also about ES, and with an $\mu + \lambda$ extension. For population parameters, parents ($\mu$) and offspring ($\lambda$) combine for generation selection. For mutation and selection, offspring are generated with adaptive mutation, and the best individuals from parents and offspring are elected for each new generation.

The sixth version is also about, and also with an $\mu$ and $\lambda$ extension, but the algorithm is more concern with that next generation is picked based on offspring population. For population parameters, parents ($\mu$) generate offspring ($\lambda$), but only the top $\mu$ offspring advance. For mutation and selection, mutation is applied to generate offspring, and the best $\mu$ offspring replace the parent generation.

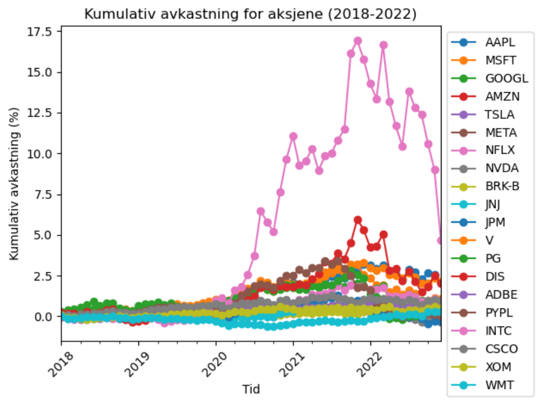
## Results



(a) Covariance



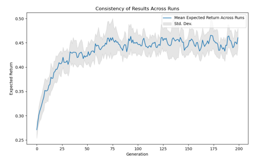
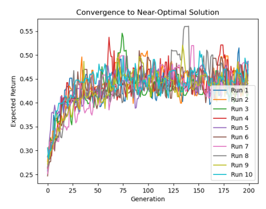
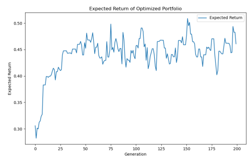
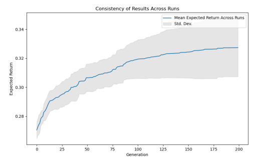
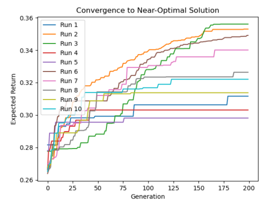
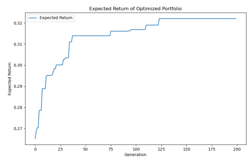
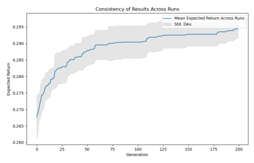
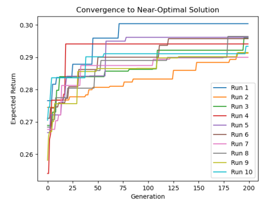
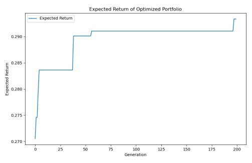
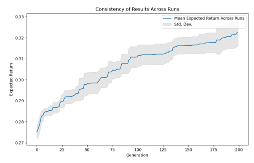
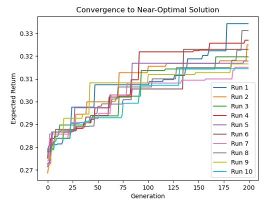
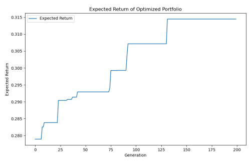
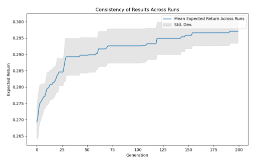
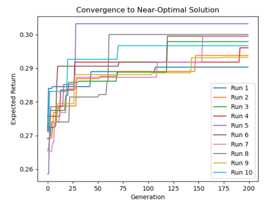
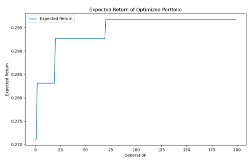
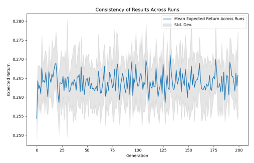
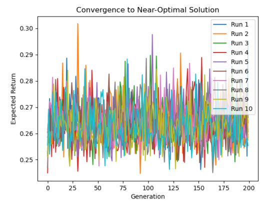
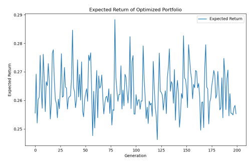
(b) Average Return



(c) Cumulative Return

Figure 1: Figures for the prelimenary development

Here one can see a covariance matrix, average monthly return and cumulative return. Noticeably, the stock INTC, is not having any return in the period. Another insight is that NFLX is doing quite well in the second half of the period. The next results is about optimizing several algorithms to maximize return.

| Version | Consistency of results | Convergence to near-optimal | Expected return |
|---|---|---|---|
| V. 1 |  |  |  |
| V. 2 |  |  |  |
| V. 3 |  |  |  |
| V. 4 |  |  |  |
| V. 5 |  |  |  |
| V. 6 |  |  |  |

In the continuation of this section we'll present the results from our evolutionary programming optimization project for portfolio management. In this project, we implemented six different versions of an evolutionary algorithm to maximize the Sharpe ratio of a portfolio. The goal was to find the best portfolio allocation strategy for a set of 20 popular stocks, balancing returns against risk. Each version uses different parameters, and we evaluated them based on three metrics: Consistency of Results, Convergence to Near-Optimal Solutions, and Expected Return. We'll explain what each metric shows and discuss the performance of each version.

The first metric, Consistency of Results, measures the average expected return across multiple runs, with the shaded area representing standard deviation. This helps us understand how stable each version's results are. For example, in Version 1, the consistency plot shows moderate variability, meaning the results are somewhat stable across runs, but with some fluctuation. Versions 2, 3, and 4, on the other hand, show a more steady progression with less variability, indicating high stability in results across different runs. Version 5 starts to show a bit more variability, while Version 6 has significant fluctuations, meaning its results are not as consistent.

Next, we'll explain Convergence to Near-Optimal Solutions. This metric tracks the best return across generations for each run, showing how quickly each version finds high-return portfolios. For Version 1, we see a steady convergence to high returns, which suggests this version finds good solutions consistently. Versions 2, 3, and 4 are even better in this aspect. They show smooth, steady convergence across runs, meaning these versions reliably find high-return solutions over generations. However, Version 5 has a slower and less regular path to the optimal solution, and Version 6 shows highly irregular paths, indicating it struggles with convergence.

Finally, the Expected Return plot shows the progression of the best portfolio in a single run for each version. This metric demonstrates how each version performs over time. Versions 2, 3, and 4 have a steady increase in expected returns, which means these versions not only converge consistently but also keep improving over generations. In contrast, Version 5 has a stable, but somewhat less consistent pattern, while Version 6 shows erratic results, with frequent ups and downs in expected return. This instability in Version 6 indicates that its configuration might be suboptimal for this type of optimization.

## Conclusion

In summary, the project tested six versions of an evolutionary programming algorithm for optimizing a stock portfolio, aiming to maximize the Sharpe ratio. Versions 2, 3, and 4 emerged as the top performers, consistently demonstrating high stability across multiple runs, smooth convergence to high-return solutions, and reliable improvement over generations. These versions produced portfolios that balanced risk and return effectively, achieving the objectives of the optimization. Conversely, Versions 5 and 6 showed higher variability, irregular convergence, and less reliable results, suggesting they may be less suitable without further tuning.

To end the conclusion, I will add some ideas for future work. Firstly, to fine-tuning parameters like mutation rate, population size, and selection strategy could improve the less stable versions. A systematic parameter optimization approach, such as grid search or random search, could identify configurations that enhance stability and convergence.

Secondly, adding constraints such as maximum asset weight or sector-based limits could create portfolios that are more practical and aligned with real-world investment policies, making the model applicable to diverse investment strategies.

Thirdly, expanding the dataset to include additional stocks, international assets, or different time periods could reveal how robust the algorithm is under different market conditions. This would help evaluate the generalizability of the best-performing configurations.

Fourthly, combining evolutionary programming with other optimization methods, such as genetic algorithms or particle swarm optimization, could yield even better performance, leveraging strengths from multiple techniques to improve convergence speed and solution quality.

Lastly, exploring alternative risk-adjusted performance metrics, like the Sortino ratio (which considers downside risk only) or conditional value at risk (CVaR), could provide insights into how the algorithm manages different types of risk and whether it can achieve more targeted risk-return balances.

These improvements could enhance the algorithm's robustness, make it more applicable to real-world portfolio management, and ensure it remains effective under diverse market conditions. Future work along these lines could yield a more versatile and reliable portfolio optimization tool.