

Beetle Antennae Search Algorithm and Beetle Swarm Antennae Search Algorithm for Portfolio Optimization Application

Linwei Zhu

2014969

Project Dissertation



Swansea University
Prifysgol Abertawe


Department of Computer Science
Adran Gyfrifidureg

28th April 2023

Declaration


Statement 1

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed  Linwei Zhu (2014969)
Date 28/04/2023 Linwei Zhu (2014969)

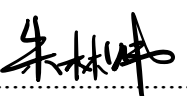
Statement 2

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by citations giving explicit references. A bibliography is appended.

Signed  Linwei Zhu (2014969)
Date 28/04/2023 Linwei Zhu (2014969)

Statement 3

The University's ethical procedures have been followed and, where appropriate, ethical approval has been granted.

Signed  Linwei Zhu (2014969)
Date 28/04/2023 Linwei Zhu (2014969)

Abstract

Portfolio optimization is a critical component in financial investment decision-making, aiming to allocate assets in a way that maximizes returns while minimizing risk. This study investigates the application of two innovative optimization algorithms, the Beetle Antennae Search (BAS) and Beetle Swarm Antennae Search (BSAS) algorithms, to portfolio optimization problems, and compares their performance with the widely-used Sequential Least Squares Quadratic Programming (SLSQP) algorithm. Through a series of experiments, the BAS and BSAS algorithms were assessed in terms of their optimization results (Sharpe ratio), computation time, and algorithmic stability. Although both BAS and BSAS algorithms demonstrated potential in portfolio optimization and exhibited considerable performance in optimization results and computational efficiency, they did not surpass the SLSQP algorithm in terms of the Sharpe ratio. However, the BSAS algorithm displayed better stability compared to the BAS algorithm. Future research directions include refining the algorithms, incorporating additional optimization algorithms for comparison, introducing new risk-adjusted metrics, testing in real market conditions, and expanding to other asset classes. This study contributes to the understanding of the applicability and performance of the BAS and BSAS algorithms in portfolio optimization, providing valuable insights for investment strategy formulation and practical applications.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Zhan Li, for his continuous guidance, patience, and expertise throughout the entire process of my graduation project. His insights and suggestions have been invaluable in addressing technical challenges and improving the quality of my work.

I would also like to thank Swansea University for providing me with access to the necessary literature resources through the iFind system, which greatly contributed to the development of my project.

My heartfelt appreciation goes to my girlfriend, who has been a constant source of emotional support and encouragement during the research and writing of my thesis. Her presence has made the journey less lonely, and her motivation has kept me going even when I felt tired and overwhelmed.

I am grateful to Yahoo Finance for providing the free API that enabled me to obtain essential data for my project. Their service has been instrumental in advancing my research and ensuring its success.

Lastly, I would like to express my gratitude to Dr. Jiang for publishing the beetle antennae search algorithm in 2017, which provided me with the opportunity to utilize and explore its potential in my work.

Table of Contents

1	<i>Introduction</i>	1
2	<i>Background</i>	1
2.1	Importance of portfolio optimization	1
2.2	Existing optimization algorithms	2
2.3	Basic principles of beetle antennae search algorithm	2
2.4	Applications of beetle antennae search algorithm in other fields	2
2.5	Where the data comes from	3
3	<i>Sharp ratio</i>	4
3.1	Sharp ratio	4
3.2	Maximizing sharpe ratio	4
4	<i>Algorithm description</i>	4
4.1	Beetle antennae search algorithm (BAS)	5
4.2	Beetle swarm antennae search algorithm(BSAS)	6
5	<i>Processing the data</i>	6
5.1	Selecting and obtaining portfolio data	6
5.2	Sharp ratio	7
6	<i>Code listings</i>	8
6.1	Sharpe ratio function implementation	8
6.2	Beetle antennae search algorithm implementation	8
6.3	Beetle swarm antennae search algorithm implementation	10
6.4	Sequential least squares quadratic programming implementation (SLSQP)	11
7	<i>Algorithm applications</i>	12
8	<i>Results</i>	14
8.1	Impact of Algorithm Parameters on Performance	14
8.2	Comparison of algorithms	18
8.3	Comparative of Algorithmic Stability	21
9	<i>Potential Improvements and Future Work</i>	22
9.1	Algorithm enhancement	22
9.2	Introducing additional optimization algorithms	22
9.3	Incorporating new metrics	22
9.4	Real market testing	22
9.5	Adapting to multiple asset classes	22
9.6	Testing under various market conditions	22

10	<i>Conclusions</i>	23
11	<i>Bibliography</i>	24

1 Introduction

In recent years, the field of machine learning and optimization algorithms has made significant advancements, providing novel approaches to problem-solving in various industries, including finance. With the rise in global economic uncertainty and inflation, investors are becoming increasingly concerned about their asset allocation and seeking more effective ways to optimize their investment portfolios. In light of these concerns, this study aims to explore the application of relatively new optimization algorithms, the Beetle Antennae Search (BAS) algorithm and its variant, the Beetle Swarm Antennae Search (BSAS) algorithm, in the realm of stock investment portfolio optimization.

The research objectives of this study are to develop and implement the BAS and BSAS algorithms for portfolio optimization and compare their performance with existing algorithms such as SLSQP. We focus on the constituent stocks of the Dow Jones Industrial Average, NASDAQ, and S&P 500 indices in the US stock market. By employing the BAS and BSAS algorithms, we aim to investigate their potential advantages in terms of runtime and optimization results.

The research questions we seek to address in this study include:

- (1) How do the BAS and BSAS algorithms perform in the context of portfolio optimization?
- (2) How do they compare to the SLSQP algorithm in terms of performance and efficiency?

This research is significant as it explores new optimization algorithms that could provide more effective solutions for portfolio optimization, potentially aiding investors in achieving better investment returns in a challenging economic climate.

The methodology employed in this study involves designing and implementing the BAS and BSAS algorithms, finding suitable methods for evaluating investment portfolios, such as the Sharpe ratio, and comparing their performance with the SLSQP algorithm. Additionally, we will tackle the challenges of seeking trustworthy data providers and integrating the data for analysis.

The structure of the paper is as follows: background and literature review, data sources and processing, algorithm principles and implementation, experimental design and result analysis, and conclusion, including potential improvements and future work. Through this study, we hope to contribute to the field of portfolio optimization by providing insights into the potential of BAS and BSAS algorithms, as well as offering investors an effective tool for optimizing their investment portfolios.

2 Background

2.1 Importance of portfolio optimization

Portfolio optimization is essential for investors because it allows them to construct a diversified portfolio that can yield higher returns without incurring additional risk. As

Markowitz [1] demonstrated in 1952, a well-diversified portfolio can be less volatile than the total sum of its individual components. By optimizing the portfolio, investors can effectively reduce risk and increase return.

2.2 Existing optimization algorithms

Several existing optimization algorithms, such as the Sequential Least Squares Quadratic Programming (SLSQP) and Particle Swarm Optimization (PSO), have been employed to solve portfolio optimization problems. SLSQP is a stepwise approximation algorithm for nonlinear programming that constructs a quadratic programming model to solve constrained nonlinear optimization problems. It is suitable for optimization problems with equality and inequality constraints and can accurately find optimal solutions but may get stuck in local optima. On the other hand, PSO is a stochastic search algorithm that simulates the movement of particles in a search space to find the global optimum. It does not require the calculation of derivatives or quadratic programming subproblems and is suitable for nonlinear optimization problems. However, it may also get stuck in local optima.

2.3 Basic principles of beetle antennae search algorithm

The Beetle Antennae Search (BAS) algorithm is a nature-inspired optimization algorithm based on the foraging behaviour of beetles. It simulates the process of beetles searching for food using their antennae to detect their surroundings. The core idea of the BAS algorithm is to perform both local and global search within the search space to find the optimal solution.

During the execution of the algorithm, the search space, initial beetle position, and other parameters are initialized. Next, local search is performed using two antennae (left and right) based on the current beetle position. The direction and step size of the antennae search are determined by specific rules. According to the antennae search results, the position with the best fitness value among the left antenna, right antenna, and current position is selected as the new beetle position. The beetle position and global optimum are continuously updated until the preset number of iterations is reached, at which point the algorithm terminates and outputs the global optimum.

A variant of the Beetle Antennae Search algorithm is the Beetle Swarm Antennae Search (BSAS) algorithm. In BSAS, the step size of the antennae search is variable rather than fixed. As the number of iterations increases, the step size gradually decreases, allowing the algorithm to search more accurately for the optimal solution. This adaptive step size strategy helps improve the search performance of the algorithm in complex optimization problems.

2.4 Applications of beetle antennae search algorithm in other fields

In this section, we will discuss the applications of the beetle antennae search (BAS) algorithm in various fields, highlighting its versatility and potential for solving complex optimization problems.

Fault Diagnosis of Mine Fan Bearing: Che et al. [2] proposed a method that combines rough set attribute reduction with BAS to optimize the BP neural network for diagnosing motor bearing faults in mine fans. Their method achieves a 90% accuracy in fault diagnosis and

demonstrates faster convergence speed and shorter operation time compared to other optimization algorithms.

Path Planning for Vehicles: Liang et al. [3] proposed an improved BAS algorithm for vehicle path planning by introducing a map safety threshold, the addition of virtual target points, and smoothing the path using B-spline. The improved algorithm showed better performance than existing path planning algorithms in simulation tests.

Global Path Planning for Mobile Robots: Lin et al. [4] presented a path planning algorithm based on the BAS that effectively improves the speed, obstacle avoidance, and environmental adaptability of mobile robot path planning. Their BAS-based algorithm demonstrated superior performance in searching for paths and avoiding obstacles compared to other algorithms, even in complex environments.

These applications demonstrate the adaptability and potential of the BAS algorithm in addressing various optimization problems in diverse fields, such as fault diagnosis, path planning, and mobile robot navigation. By exploring these applications, we can better understand the potential of the BAS algorithm in solving complex optimization problems.

2.5 Where the data comes from

In this research, the data is obtained from the Yahoo Finance API, which is a widely used and reliable source of financial data. The choice of using Yahoo Finance API is due to its extensive data coverage, stability, ease of use, and free access within certain limits.

To gather data using the Yahoo Finance API, a request is constructed in Python by importing the Yahoo Finance API library. The request includes the required data type, time range, and stock symbols. A variable is then created to store the retrieved data.

The stock samples are selected from the constituents of the S&P 500, NASDAQ, and Dow Jones indices. These stocks are chosen because they represent top companies from various industries, which are already filtered by the index creators. This selection method reduces the need for extensive research on individual companies while minimizing the risk associated with smaller, potentially more volatile stocks.

The time range for data collection is not based on any specific standard or method but is chosen to ensure a sufficient time span for analyzing long-term stock performance. A broader time range helps in better understanding the overall trends in the stock market.

The representativeness of the chosen data is ensured by selecting stocks from well-established indices. These indices are widely recognized and followed in the financial world, which provides credibility to the data and ensures that the research results are representative of overall market trends.

3 Sharp ratio

3.1 Sharp ratio

The Sharpe ratio, devised by Nobel laureate William F. Sharpe [5] in 1966, serves as a metric for evaluating risk-adjusted returns, enabling investors to appraise the performance of their investments while taking risk into account. The formula for calculating the Sharpe ratio is as follows:

$$\text{Sharpe ratio} = \frac{(r_x - R_f)}{\sigma}$$

Where r_x represents the average rate of return on the investment, R_f denotes the optimal risk-free rate of return, and σ signifies the standard deviation of r_x , which reflects the risk inherent in the investment. A higher Sharpe ratio suggests a greater excess return relative to the increased volatility associated with the investment.

In the realm of investment, a Sharpe ratio of 1 is deemed acceptable or favourable, whereas values below 1 are considered sub-optimal, and values exceeding 1, approaching 2 or 3, are regarded as highly exceptional. The eminent standing and broad acceptance of the Sharpe ratio render it a suitable instrument for investors to assess and compare diverse investment prospects.

3.2 Maximizing Sharpe ratio

With regard to maximizing the Sharpe ratio, let us hypothesize that an investor aims to allocate assets in such a manner that the portfolio attains the highest possible or maximal Sharpe ratio. Let P represent a portfolio comprising assets A_1, A_2, \dots, A_n , with $\mu_1, \mu_2, \dots, \mu_n$ constituting the asset returns and W_1, W_2, \dots, W_n as the corresponding weights. The portfolio return r is ascertained by a weighted summation of individual asset returns, expressed as $\sum(W_i \cdot \mu_i)$, while the risk is given by

$$\sqrt{\sum \sum W_i \cdot W_j \cdot \sigma_{ij}}$$

Assuming that the investor imposes only fundamental constraints on the portfolio, the mathematical model for Sharpe ratio-based portfolio optimization is formulated by maximizing the objective function $(r_x - R_f) / \sigma$, subject to the basic constraints. The numerator of the objective function embodies the excess returns of the investment over that of a risk-free asset R_f , and the denominator represents the investment's risk. The goal is to maximize the Sharpe ratio while maintaining a fully invested portfolio.

4 Algorithm description

In this section, we will elaborate on the BAS and its variant BSAS nature.

4.1 Beetle antennae search algorithm (BAS)

Introduced in 2017, Jiang and Li [6] first presented the Beetle Antennae Search (BAS) algorithm represents a cutting-edge, bio-inspired optimization technique that mimics the foraging behaviour of beetles. This method is characterized by its simplicity, low computational complexity, and rapid convergence. Distinguished from traditional gradient-based optimization approaches such as gradient descent and Newton's method, BAS achieves global optimization without necessitating intricate gradient computations. Moreover, it only involves a single individual, rendering it more computationally efficient than heuristic algorithms like simulated annealing and genetic algorithms.

The fundamental principle underlying the BAS algorithm is that a beetle, during the foraging process, lacks knowledge of the precise location of its food source. In order to locate the source, the beetle employs its antennae to detect the concentration of food scent in its surroundings. By comparing the scent concentrations perceived by the left and right antennae, the beetle can ascertain the direction in which it should proceed. The algorithm successively refines the search direction until a predetermined level of precision is attained, culminating in the identification of the optimal solution.

Consider an optimization problem to understand the working of BAS,

$$\min_x F(x) \in R.$$

The Function $F(x)$ is objective function to optimize, The beetle has two tentacles, which means there are two position vectors x_{left} and x_{right} .

$$x_{left} = x_m + d_m \times b$$

$$x_{right} = x_m - d_m \times b$$

Where x_m is the centroid position of the beetle, d_m is the distance between two antennae, b is a random directional vector.

Update rules for x

$$x_{m'} = x_m + \delta_m \times b \times \text{sign}(f(x_{right}) - f(x_{left}))$$

$f(x_{right}) - f(x_{left})$ will determine the next step of x_m , δ_m is step distance in m .

The update rule for the step length δ and the distance d between the two antennae s is as follows.

$$d_{m'} = d_e \times d_m + 0.001$$

$$\delta_{m'} = \delta_e \times \delta_m$$

d_e and δ_e respectively represent the decreasing factors of d and δ .

During the optimization process of the BAS algorithm, only a single beetle is involved rather than a population. Consequently, the algorithm requires fewer parameters, simpler code, and

significantly lower computational resources than other methods, resulting in reduced complexity. However, this also leads to a loss of population diversity, and the algorithm's singularity makes it prone to becoming trapped in local optima.

4.2 Beetle swarm antennae search algorithm(BSAS)

As the convergence results of the BAS algorithm largely depend on the random direction of the beetle during each iteration, this may lead to different optimization outcomes for distinct random seeds. As previously mentioned, the simplicity of BAS parameters and the use of only one beetle for function optimization make it prone to getting trapped in local optima. Therefore, this section will introduce a variant of the BAS algorithm, the BSAS algorithm, which aims to address these issues.

The BSAS algorithm employed in this study is inspired by the Beetle Swarm Antennae Search (BSAS) algorithm proposed by Jiangyu Wang in 2018 [7]. In the original BSAS algorithm, the parameter p is used to control the randomness in the search process. However, in our implementation, we have adjusted the definition of parameter p to enhance the practicality and effectiveness of the algorithm. Through this improvement, we have successfully applied the BSAS algorithm to solve optimization problems in portfolio selection, achieving better performance.

The BSAS algorithm initializes a swarm of beetles and iteratively updates their positions by exploring the search space using the `beetle_antenna_search` function. During each iteration, it evaluates the new positions to find the best solution for the given objective function (in this case, portfolio optimization). The algorithm maintains a record of the best position found and the corresponding best objective function value, ultimately returning them after the specified number of iterations.

5 Processing the data

5.1 Selecting and obtaining portfolio data

In this study, the selection of investment projects is focused on stocks from the three major US stock indices: the NASDAQ Composite Index, the S&P 500 Index, and the Dow Jones Industrial Average. As previously mentioned, choosing stocks from these major indices aligns with the objective of optimizing an investment portfolio with lower investment risk. For the experimental design, nearly 200 stocks were selected from these three indices to form an investment project pool, allowing the algorithm to optimize a better investment portfolio among these 200 projects.

For the acquisition of stock data, the Yahoo Finance API is utilized, as it is a free, highly credible, and widely recognized source. The deployment and usage of the API are both swift and convenient, enabling readers to easily deploy the code and verify its correctness after reviewing this research. One of the objectives of this study is to assist investors in better optimizing their investment portfolios through the application of the BAS and BSAS algorithms in the financial investment domain. Considering that the code may be utilized by numerous investors, simplicity and ease of deployment and usage are important factors taken into account in this research.

5.1.1 Yahoo finance API

For the acquisition of stock data, the Yahoo Finance API is utilized, as it is a free, highly credible, and widely recognized source. The deployment and usage of the API are both swift and convenient, enabling readers to easily deploy the code and verify its correctness after reviewing this research. One of the objectives of this study is to assist investors in better optimizing their investment portfolios through the application of the BAS and BSAS algorithms in the financial investment domain. Considering that the code may be utilized by numerous investors, simplicity and ease of deployment and usage are important factors taken into account in this research.

5.2 Sharp ratio

After obtaining the data from the Yahoo Finance API, it is essential to pre-process the data to facilitate the subsequent calculation of the Sharpe ratio by the algorithms. The following steps outline the data pre-processing procedures:

5.2.1 Data cleaning

Initially, inspect the data for missing values, outliers, or inconsistencies. If any issues are present, perform imputation, correction, or deletion as necessary. This step ensures data quality and provides a reliable foundation for further analysis.

5.2.2 Data transformation

Convert the closing price data into daily returns by calculating the percentage change between each day's closing price and the previous day's closing price. Daily returns serve as a key indicator for assessing stock volatility and portfolio risk.

5.2.3 Calculate the mean daily return and standard deviation

These two statistics will be utilized to compute the Sharpe ratio. The mean daily return represents the expected return of a stock, while the standard deviation denotes its volatility or risk.

5.2.4 Compute the covariance matrix

This crucial step evaluates the correlation between stocks within the investment portfolio. The covariance matrix aids in understanding the interrelationships among different stocks and diversifying risk when constructing an investment portfolio.

5.2.5 Data standardization

Before executing optimization algorithms, standardizing the data to a uniform scale improves computational efficiency and prevents issues such as vanishing or exploding gradients.

Upon completing these pre-processing steps, the data will provide the necessary input for subsequent algorithms to calculate the Sharpe ratio.

6 Code listings

6.1 Sharpe ratio function implementation

```
def sharpe_ratio_function(weights, mean_returns, cov_matrix,
rf):

    portfolio_return = np.dot(weights, mean_returns) - rf

    portfolio_volatility = np.sqrt(np.dot(weights.T,
np.dot(cov_matrix, weights)))

    sharpe_ratio = portfolio_return / portfolio_volatility

    return -sharpe_ratio
```

Listings 6.1.1: An extract implementation of Sharpe ratio function

In this function, “sharpe_ratio_function”, we calculate the Sharpe ratio for a given investment portfolio. The function takes four inputs: weights, “mean_returns”, “cov_matrix”, and “rf”.

Weights represent the proportion of each asset in the portfolio, mean_returns is a vector containing the mean daily returns of each asset, “cov_matrix” is the covariance matrix for the returns, and rf is the risk-free rate.

The function first computes the expected portfolio return by taking the dot product of the weights and the mean returns, then subtracting the risk-free rate. This calculation represents the excess return of the portfolio over the risk-free rate.

Next, the function calculates the portfolio volatility by taking the square root of the dot product of the weights transposed, the covariance matrix, and the weights. This represents the overall risk of the portfolio.

Finally, the Sharpe ratio is computed by dividing the portfolio return by the portfolio volatility. The result is then negated to conform with optimization algorithms that minimize the objective function. The Sharpe ratio measures the risk-adjusted performance of a portfolio, with higher values indicating better risk-adjusted returns. This function is an essential component for optimizing the investment portfolio and achieving the optimal trade-off between risk and return.

6.2 Beetle antennae search algorithm implementation

```
def beetle_antenna_search(f, dim, min_, max_, n_iterations,
mean_returns, cov_matrix, rf):

    x = np.abs(np.random.randn(dim))

    x /= np.sum(x)

    step = 1
```

```

fleft = float('inf')
fright = float('inf')

for i in range(n_iterations):

    dir = np.random.randn(dim)

    dir = dir / np.linalg.norm(dir, ord=1, axis=0)

    step = 0.95 * step

    d0 = step / 2

    xl = x + d0 * dir / 2
    xr = x - d0 * dir / 2

    fleft = f(xl, mean_returns, cov_matrix, rf)
    fright = f(xr, mean_returns, cov_matrix, rf)

    tx = x - step * dir * np.sign(fleft - fright)

    tx = np.abs(tx)

    tx /= np.sum(tx)

    if np.any(np.abs(tx) > max_) or np.any(np.abs(tx) <
min_):

        continue

    else:

        x = tx

return x, min(fleft, fright)

```

Listings 6.2.1: An extract implementation of beetle antennae search algorithm.

In this function, “beetle_antenna_search”, we implement the Beetle Antennae Search (BAS) algorithm for portfolio optimization. The function takes seven inputs: “f”, “dim”, “min_”, “max_”, “n_iterations”, “mean_returns”, “cov_matrix”, and “rf”.

f is the objective function to be optimized (in this case, the “sharpe_ratio_function”), dim is the dimensionality of the search space (equal to the number of assets in the portfolio), “min_” and “max_” are the minimum and maximum bounds for the weights, “n_iterations” is the number of iterations for the algorithm, “mean_returns” is a vector containing the mean daily returns of each asset, “cov_matrix” is the covariance matrix for the returns, and rf is the risk-free rate.

The function initializes a random weight vector x , which represents the proportion of each asset in the portfolio, and normalizes it so that the sum of weights is equal to 1. It then initializes the step size and sets the initial values for $fleft$ and $fright$ to positive infinity.

Within the main loop of the algorithm, which iterates $n_iterations$ times, the function generates a random search direction dir , which is normalized using the L1 norm. The step size is gradually reduced by multiplying it by a decay factor of 0.95. The function then calculates two new candidate solutions x_l and x_r , representing the left and right antennae of the beetle, by moving from the current position x in the positive and negative directions of dir , respectively.

The objective function values $fleft$ and $fright$ are computed for the two candidate solutions. Based on the comparison of these values, the function updates the current position x by moving in the direction of the better candidate solution. The updated position tx is then normalized so that the sum of weights remains equal to 1.

The function checks whether the updated position tx satisfies the minimum and maximum constraints for the weights. If the constraints are satisfied, the position x is updated with the new position tx . The loop continues until all iterations are completed.

After the algorithm has finished, the function returns the optimized weight vector x and the minimum value of the objective function, which represents the optimal Sharpe ratio for the portfolio. This implementation of the BAS algorithm allows for the efficient optimization of investment portfolios, taking into account both risk and return.

6.3 Beetle swarm antennae search algorithm implementation

```
def beetle_swarm_search(f, swarm_size, dim, min_, max_,
n_iterations, mean_returns, cov_matrix, rf):

    beetles = [np.random.rand(dim) for _ in range(swarm_size)]

    for beetle in beetles:

        beetle /= np.sum(beetle)

    best_position = None

    best_value = float('inf')

    for i in range(n_iterations):

        new_positions = []

        for beetle in beetles:

            new_position, new_value = beetle_antenna_search(f,
dim, min_, max_, 500, mean_returns, cov_matrix, rf)

            if new_value < best_value:

                best_position = new_position
```



```

        best_value = new_value

        new_positions.append(new_position)

    beetles = new_positions

    return best_position, best_value

```

Listings 6.3.1: An extract implementation of beetle swarm antennae search algorithm.

In this section of the code, we define the Beetle Swarm Search (BSAS) algorithm, which is an optimization technique applied to portfolio optimization. The BSAS algorithm works by simulating the behaviour of a swarm of beetles, each representing a candidate solution (i.e., a portfolio with a specific set of weights) in the search space.

The algorithm starts by initializing a swarm of beetles with random weights, and then normalizes each beetle's weights to ensure that the sum of the weights is equal to 1. The “best_position” and “best_value” variables are initialized to keep track of the globally optimal solution found by the swarm.

The main loop of the algorithm iterates for a specified number of iterations. In each iteration, a “new_positions” list is created to store the updated positions (i.e., weight vectors) for each beetle in the swarm. The algorithm then iterates through each beetle and applies the Beetle Antenna Search (BAS) function to find the new position and objective function value for the current beetle. If a better solution is found, the global best_position and best_value variables are updated accordingly.

At the end of each iteration, the swarm is updated with the new_positions list, which contains the updated weight vectors for each beetle. Finally, the algorithm returns the “best_position” (i.e., the optimal weight vector) and the “best_value” (i.e., the minimum objective function value) found by the swarm.

6.4 Sequential least squares quadratic programming implementation (SLSQP)

```

result_slsqp=
sco.minimize(sharpe_ratio_function, initial_weights, args=(mean_
returns, cov_matrix, rf), method='SLSQP', bounds=bounds,
constraints=cons)

```

Listings 6.4.1: An extract implementation of slsqp algorithm

The SLSQP optimization is performed using the `sco.minimize()` function from the SciPy library, which takes the following parameters:

6.4.1 Sharpe ratio function

The objective function to minimize, which calculates the negative Sharpe ratio for a given set of weights.

6.4.2 Initial weights

The starting point for the optimization algorithm, which is the initial weight vector for the assets.

6.4.3 Args

A tuple containing additional arguments to be passed to the objective function, including the `mean_returns`, `cov_matrix`, and `rf`.

6.4.4 Method

The optimization algorithm to use, in this case 'SLSQP'.

6.4.5 Bounds

The boundary conditions for the optimization problem, which are set to ensure that the weights of the assets remain between 0 and 1.

6.4.6 Constraints

The constraint conditions for the optimization problem, which are set to ensure that the sum of the weights is equal to 1.

7 Algorithm applications

In this thesis, we apply the SLSQP algorithm, BAS algorithm, and BSAS algorithm to the portfolio optimization problem. First, we import the necessary libraries, such as NumPy, Pandas, SciPy, Matplotlib, and Seaborn, which provide support for data processing, algorithm implementation, and result visualization. Next, we use the Yahoo Finance API to select some representative stocks from the constituents of the three major US stock indices to ensure a diverse stock pool. Then, we obtain the historical data of the selected stocks within a specific time range, providing the foundation for subsequent portfolio optimization analysis.

After obtaining the stock data, we preprocess the data, including calculating the logarithmic returns for each stock and the covariance matrix between stocks. The preprocessed data is used for the following portfolio optimization analysis.

Upon completing data preprocessing, we separately construct the code implementations for the SLSQP algorithm, BAS algorithm, and BSAS algorithm. The specific implementation details of these three algorithms have been introduced in the previous chapter and will not be repeated here. In the process of implementing these algorithms, we use the Sharpe ratio function as the optimization objective and include relevant constraints in the algorithms.

Next, we configure the parameters of the SLSQP, BAS, and BSAS algorithms in detail to implement portfolio optimization. When applying the SLSQP algorithm, we first set the weight boundaries (bounds) and constraint conditions (cons). The weight boundaries are a list of tuples, where each tuple represents the minimum and maximum values of the

corresponding asset weight. In this example, we set the weight range for all assets to be between 0 and 1. The constraint conditions include a restriction that the sum of asset weights should be 1. We define this constraint in dictionary form and pass it into the `sco.minimize` function when calling it.

```
bounds = [(0, 1) for _ in range(n_assets)]  
  
cons = ({'type': 'eq', 'fun': lambda weights: np.sum(weights)  
- 1})
```

Listings 7.1: An extract implementation of SLSQP algorithm variable

For the BAS algorithm, we need to set parameters such as the number of iterations “`n_iterations`”. In this example, we set the number of iterations to 100, meaning the algorithm will run 100 rounds to find the optimal solution. Additionally, we set the minimum “`min_`” and maximum “`max_`” values of the weight range to 0 and 1, respectively, to restrict the weight values generated during optimization.

```
n_iterations = 100  
  
min_ = 0  
  
max_ = 1
```

Listings 7.2: An extract implementation of BAS algorithm variable

When applying the BSAS algorithm, we first determine the swarm size “`swarm_size`”. In this example, we set the swarm size to 30, indicating that 30 beetle individuals will search for the optimal solution in each iteration.

```
n_iterations = 100  
  
min_ = 0  
  
max_ = 1  
  
swarm_size = 30
```

Listings 7.3: An extract implementation of BSAS algorithm variable

We define the “`run_and_compare_algorithms`” function to run these three algorithms multiple times and compare their performance. In this function, we record the start and end times of each run to calculate the running time of the algorithms when solving the portfolio optimization problem. We also record the Sharpe ratio obtained from each run to evaluate the optimization performance of the algorithms.

When calling the “`run_and_compare_algorithms`” function, we set the number of runs (`runs`) to 20, indicating that we will perform 20 independent runs for each algorithm to evaluate their performance. Through this function, we can call the SLSQP, BAS, and BSAS algorithms to solve the portfolio optimization problem and compare the results.

To compare the efficiency and effectiveness of these algorithms, we plot box plots of their time consumption and obtained Sharpe ratios at different running times. By analyzing the box plots, we can observe the distribution of time consumption and Sharpe ratios of the SLSQP

8 Results

8.1 Impact of Algorithm Parameters on Performance

In this section, we investigate the effects of modifying the parameters of BAS and BSAS algorithms on their performance in the context of portfolio optimization. Specifically, we modify the 'n_iterations' parameter for BAS and both 'n_iterations' and 'swarm' parameters for BSAS to analyze how these changes impact the effectiveness and efficiency of the algorithms when solving the optimization problem.

8.1.1 BAS

In this experiment, we investigated the impact of the 'n_iterations' parameter of the BAS algorithm on its performance in portfolio optimization problems. To evaluate the effects of parameter variations, we set up three groups of different 'n_iterations' values, namely 10, 100, and 1000. For each parameter setting, we repeated the portfolio optimization experiment 20 times and collected the runtime and Sharpe ratio data for each trial.

	Sharpe ratio		
	Max	Min	Mean
BAS_10	1.273779	0.757916	1.023304
BAS_100	1.775802	1.472403	1.648170
BAS_1000	1.814676	1.573991	1.680960

Table 8.1.1: Reporting on BAS algorithm changing iteration optimisation results.

	Time(s)		
	Max	Min	Mean
BAS_10	0.001023	0.000000	0.000401

BAS_100	0.004999	0.002999	0.003952
BAS_1000	0.042105	0.033518	0.036702

Table 8.1.2: Results of the time taken by the BAS algorithm changing iteration.

The experimental results are as follows:

When $n_iterations = 10$, the maximum Sharpe ratio is 1.273779, the minimum is 0.757916, and the average is 1.023304. The maximum runtime is 0.001023 seconds, the minimum is 0.000000 seconds, and the average is 0.000401 seconds.

When $n_iterations = 100$, the maximum Sharpe ratio is 1.775802, the minimum is 1.472403, and the average is 1.648170. The maximum runtime is 0.004999 seconds, the minimum is 0.002999 seconds, and the average is 0.003952 seconds.

When $n_iterations = 1000$, the maximum Sharpe ratio is 1.814676, the minimum is 1.573991, and the average is 1.680960. The maximum runtime is 0.042105 seconds, the minimum is 0.033518 seconds, and the average is 0.036702 seconds.

According to the experimental results, it can be observed that as the 'n_iterations' parameter increases, the Sharpe ratio performance of the BAS algorithm in portfolio optimization problems improves. However, at the same time, the runtime of the algorithm also increases accordingly. Therefore, in practical applications, we need to strike a balance between computational efficiency and optimization performance in order to select an appropriate 'n_iterations' parameter value according to specific requirements.

8.1.2 BSAS swarm size

In this experiment, we investigated the impact of the 'swarm' parameter of the BSAS algorithm on its performance in portfolio optimization problems. To evaluate the effects of parameter variations, we set up three groups of different 'swarm' values, namely 1, 2, and 4. For each parameter setting, we repeated the portfolio optimization experiment 20 times and collected the runtime and Sharpe ratio data for each trial.

	Sharpe ratio		
	Max	Min	Mean
BSAS_swarm1	1.785653	1.423228	1.648704
BSAS_swarm2	1.776848	1.610179	1.706578
BSAS_swarm3	1.823446	1.696668	1.748038

Table 8.1.3: Reporting on BSAS algorithm changing swarm size optimisation results.

	Time(s)		
	Max	Min	Mean
BSAS_swarm1	0.018275	0.007999	0.010146
BSAS_swarm2	0.024002	0.015996	0.018373
BSAS_swarm3	0.039507	0.034004	0.036021

Table 8.1.4: Results of the time taken by the BSAS algorithm changing swarm size.

Based on the experimental results, we can observe the performance of the BSAS algorithm in terms of Sharpe ratio and running time under different swarm sizes. The following is a summary of the experimental results:

For BSAS-swarm1, with a swarm size of 1, the maximum Sharpe ratio is 1.785653, the minimum Sharpe ratio is 1.423228, and the average Sharpe ratio is 1.648704. In terms of running time, the maximum time is 0.018275 seconds, the minimum time is 0.007999 seconds, and the average time is 0.010146 seconds.

For BSAS-swarm2, with a swarm size of 2, the maximum Sharpe ratio is 1.776848, the minimum Sharpe ratio is 1.610179, and the average Sharpe ratio is 1.706578. In terms of running time, the maximum time is 0.024002 seconds, the minimum time is 0.015996 seconds, and the average time is 0.018373 seconds.

For BSAS-swarm3, with a swarm size of 4, the maximum Sharpe ratio is 1.823446, the minimum Sharpe ratio is 1.696668, and the average Sharpe ratio is 1.748038. In terms of running time, the maximum time is 0.039507 seconds, the minimum time is 0.034004 seconds, and the average time is 0.

In this study, we investigated the impact of changing the swarm size of the BSAS algorithm on its performance. Experimental results show that as the swarm size increases, the maximum, minimum, and average Sharpe Ratios obtained also improve. This suggests that, in this particular case, increasing the swarm size helps the algorithm find better portfolio weights, thereby improving the Sharpe Ratio. However, it should be noted that as the swarm size increases, the computation time required by the algorithm also increases.

8.1.3 BSAS iteration times

	Sharpe ratio		
	Max	Min	Mean

BSAS_ieter1	1.769530	1.542687	1.682373
BSAS_ieter2	1.773328	1.635620	1.704576
BSAS_ieter3	1.805363	1.682665	1.742041

Table 8.1.5: Reporting on BSAS algorithm changing iteration optimisation results.

	Time(s)		
	Max	Min	Mean
BSAS_ieter1	0.007993	0.007993	0.009592
BSAS_ieter2	0.021990	0.016994	0.018228
BSAS_ieter3	0.038627	0.033531	0.035643

Table 8.1.4: Results of the time taken by the BSAS algorithm changing iteration.

In this experiment, we adjusted the iteration parameter of the BSAS algorithm to observe its impact on the algorithm's performance. The results indicate that while increasing the number of iterations leads to a limited improvement in Sharpe values, it significantly increases the computation time.

Specifically, by comparing the results of BSAS-iter1, BSAS-iter2, and BSAS-iter3, we can observe that the maximum, minimum, and mean Sharpe values all increase with the number of iterations. However, this improvement is relatively small, while the computation time increases significantly.

For instance, when increasing the number of iterations from BSAS-iter1 to BSAS-iter2, the mean Sharpe value only increases from 1.682373 to 1.704576, while the computation time increases from 0.009592 seconds to 0.018228 seconds. Likewise, when the number of iterations increases from BSAS-iter2 to BSAS-iter3, the mean Sharpe value increases from 1.704576 to 1.742041, while the computation time increases from 0.018228 seconds to 0.035643 seconds.

In conclusion, although increasing the number of iterations can improve the Sharpe values of the BSAS algorithm to a certain extent, it also substantially increases the computation time.

8.2 Comparison of algorithms

In this study, we applied the BAS and BSAS algorithms to portfolio optimization problems and compared them with the SLSQP algorithm. We designed two sets of experiments to evaluate the performance of these three algorithms in different scenarios.

8.2.1 First set of experiments

All three algorithms select the same investment portfolio and then optimize this portfolio. Finally, the output time and optimization results of the three algorithms are compared.

	Sharpe ratio		
	Max	Min	Mean
SLSQP	2.101288	2.101288	2.101288
BAS	1.808900	1.581206	1.693330
BSAS	1.903300	1.514977	1.735242

Table 8.2.2: Reporting on SLSQP BAS and BSAS algorithm optimisation results(same portfolio).

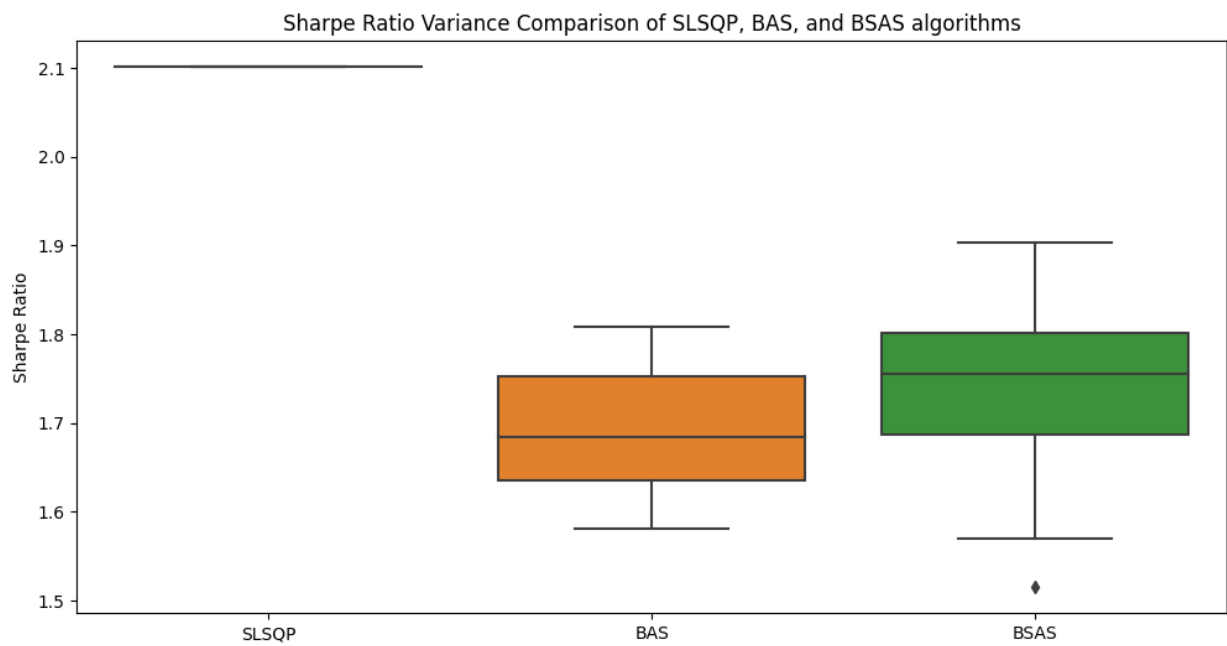


Figure 8.2.3: Box-plot of three algorithms' optimisation results(same portfolio).

	Time(s)		
	Max	Min	Mean
SLSQP	0.008001	0.005001	0.006328
BAS	0.004996	0.002944	0.003698
BSAS	0.012545	0.007999	0.009128

Table 8.2.2: Reporting on SLSQP BAS and BSAS algorithm optimisation time taken(same portfolio).

In the first experiment, when all three algorithms were applied to the same investment portfolio, the SLSQP algorithm achieved the highest Sharpe ratio with a mean value of 2.101288. However, the BAS algorithm showed the shortest computation time with a mean value of 0.003698. The BSAS algorithm demonstrated an intermediate performance in terms of the Sharpe ratio with a mean value of 1.735242 and a slightly longer computation time, with a mean value of 0.009128.

8.2.2 Second set of experiments

The three algorithms independently select investment portfolios and optimize them. Finally, the computation time and optimization results of the three algorithms are compared as well.

	Sharpe ratio		
	Max	Min	Mean
SLSQP	2.101288	2.101288	2.101288
BAS	1.808900	1.581206	1.693330
BSAS	1.903300	1.514977	1.735242

Table 8.2.3: Reporting on SLSQP BAS and BSAS algorithm optimisation results(different portfolio).

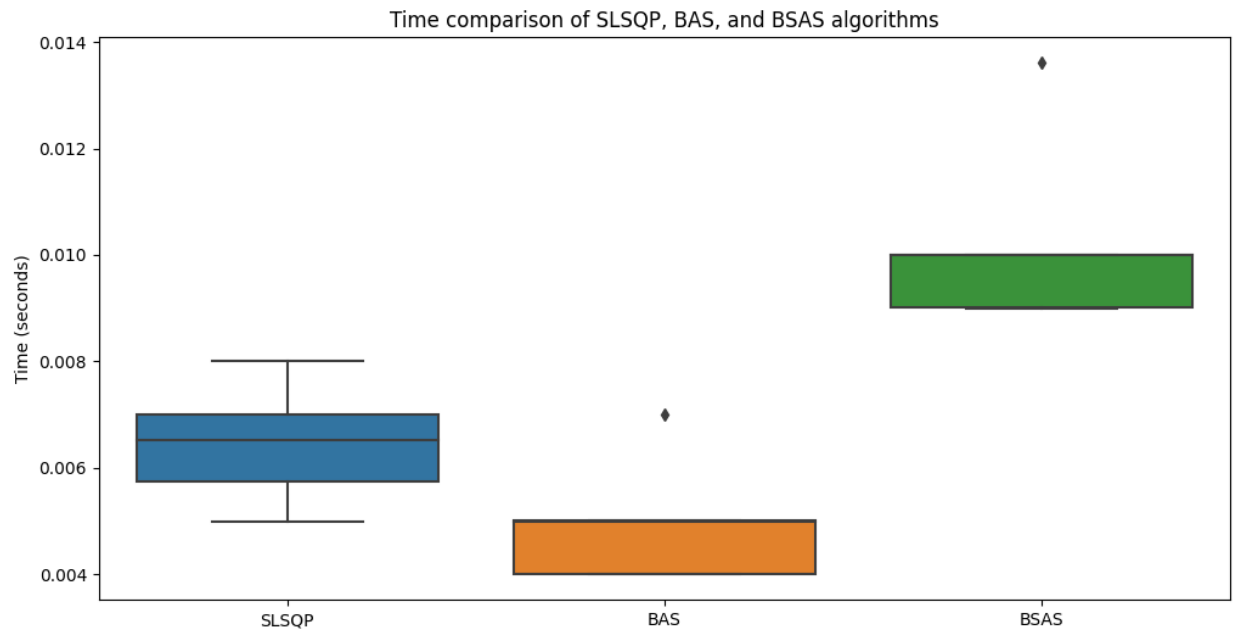


Figure 8.2.2: Box-plot of three algorithms' optimisation results(different portfolio).

	Time(s)		
	Max	Min	Mean
SLSQP	0.008001	0.005001	0.006328
BAS	0.004996	0.002944	0.003698
BSAS	0.012545	0.007999	0.009128

Table 8.2.4: Reporting on SLSQP BAS and BSAS algorithm optimisation time taken(same portfolio).

In the second experiment, where the algorithms independently selected the investment portfolios, the SLSQP algorithm still achieved the highest maximum Sharpe ratio of 2.101288 but with a lower mean value of 1.442959. The BAS algorithm had a mean Sharpe ratio of 1.193889 and a mean computation time of 0.004725, while the BSAS algorithm had a slightly lower mean Sharpe ratio of 1.168315 and a longer computation time with a mean value of 0.009810.

8.2.3 Conclusions

Through comparative experiments, we draw the following conclusions: BAS and BSAS algorithms have potential in portfolio optimization problems. In different scenarios, both algorithms show considerable performance in optimization results and computational efficiency. In particular, the BSAS algorithm, as a variant of the BAS algorithm, has a certain advantage over the BAS algorithm in terms of Sharpe ratio, but has a slight disadvantage in

terms of computation time. However, compared to the SLSQP algorithm, both BAS and BSAS algorithms have a certain gap in the Sharpe ratio, but perform well in computation time.

8.3 Comparative of Algorithmic Stability

In this study, we assessed the stability of the algorithms in order to better understand their performance in practical applications. The following approach was adopted to achieve this objective.

First, both the BAS and BSAS algorithms were run three times, each with the same parameter settings. This allowed for a comparison of the performance of the two algorithms under identical conditions. Subsequently, the results of each run, including the Sharpe ratios and running times, were collected and used for subsequent stability analysis.

	Sharpe ratio	Time(s)
	Mean Standard Deviation	Mean Standard Deviation
BAS	8.163101%	0.057644%
BSAS	5.450409%	0.097775%

Table 8.3.1: Reporting on BAS and BSAS algorithm stability.

To quantitatively reflect the stability of the algorithms, the standard deviation was employed as a metric. For each algorithm's run results (Sharpe ratios and running times), the standard deviation of the three runs was calculated separately. Then, the average standard deviation of the Sharpe ratios and running times was computed to obtain a more comprehensive measure of stability.

According to the analysis, the BSAS algorithm exhibits better stability in terms of Sharpe ratios (average standard deviation of 5.4504%) compared to the BAS algorithm (average standard deviation of 8.1631%). Regarding running times, both algorithms exhibit fluctuations of less than 1%, specifically 0.0978% (BSAS) and 0.0576% (BAS), indicating that their impact on running time stability is minimal in practical applications.

In conclusion, the BSAS algorithm demonstrates superior stability in terms of Sharpe ratios compared to the BAS algorithm, while both exhibit negligible fluctuations in running times. Consequently, the BSAS algorithm is recommended for investment strategy analysis when considering stability.

9 Potential Improvements and Future Work

In light of the findings in this result, we observed that the performance of the proposed BAS and BSAS algorithms did not surpass the SLSQP algorithm. This observation serves as a catalyst for further improvement and exploration. To advance the current research and enhance the project, we propose the following directions for future work:

9.1 Algorithm enhancement

Investigate potential modifications and enhancements to the BAS and BSAS algorithms by refining their structure or incorporating features from other optimization algorithms. These improvements could boost their performance, narrowing or even surpassing the gap between them and other mainstream optimization algorithms.

9.2 Introducing additional optimization algorithms

The current project utilizes the SLSQP algorithm for comparison with the BAS and BSAS algorithms. Future research could incorporate other optimization algorithms, such as Particle Swarm Optimization and Genetic Algorithm, in order to provide a more comprehensive comparison with BAS and BSAS.

9.3 Incorporating new metrics

The study's primary focus lies in the Sharpe ratio and runtime. Future endeavours could introduce other risk-adjusted metrics, such as the Sortino ratio and Information ratio, enabling a more multi-dimensional evaluation of the algorithms' performance.

9.4 Real market testing

The research conducted thus far has been primarily based on historical data analysis. Applying the developed algorithms to actual market conditions in future studies will help determine their performance in real-world environments.

9.5 Adapting to multiple asset classes

The current project's main focus is stock portfolio optimization. Expanding the research scope to encompass other asset classes, such as bonds, commodities, and foreign exchange, will enhance the algorithms' versatility.

9.6 Testing under various market conditions

It is essential to assess the algorithms' performance in different market environments, such as bull markets, bear markets, and sideways markets. This will provide valuable insights into their stability and adaptability under a range of market situations.

10 Conclusions

In the present research, we re-examined the significance of portfolio optimization problems and investigated new optimization algorithms. Our objective was to apply relatively innovative optimization algorithms, namely, the beetle antennae search (BAS) algorithm and the beetle swarm antennae search (BSAS) algorithm, to portfolio optimization problems.

We studied and implemented the BAS and BSAS algorithms based on the available research and literature on these techniques. A comparison was conducted between these algorithms and existing optimization methods, such as the SLSQP algorithm, in the context of portfolio optimization.

Our experimental results discussed the performance of BAS and BSAS algorithms in terms of the Sharpe ratio and runtime and compared them with the SLSQP algorithm. The research contributes to the portfolio optimization problem by providing new solutions using the BAS and BSAS algorithms, which hold significant importance for investment strategy formulation and practical application.

However, the study also identified that the BAS and BSAS algorithms constructed in this research might not yet demonstrate optimal performance. Further investigation is needed to confirm whether the BAS and BSAS algorithms are indeed inferior to mainstream optimization algorithms like SLSQP.

For future work and improvement, several directions can be explored, such as optimizing algorithm parameters, introducing more optimization algorithms for comparison, considering additional risk-adjusted indicators, testing in real market conditions, adapting to various asset classes, and evaluating performance in different market environments.

11 Bibliography

- [1] H. Markowitz, "Portfolio Selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, Mar. 1952, doi: <https://doi.org/10.2307/2975974>.
- [2] X. Q. Che, H. He, C. T. Liu, H. Sun, and L. Bian, "Fault Diagnosis of Mine Fan Bearing Based on Beetle Antennae Search," *IOP conference series*, vol. 354, no. 1, pp. 012092–012092, Oct. 2019, doi: <https://doi.org/10.1088/1755-1315/354/1/012092>.
- [3] Q. Liang, H. Zhou, Y. Yin, and W. Xiong, "An improved beetle antennae search path planning algorithm for vehicles," *PLOS ONE*, vol. 17, no. 9, p. e0274646, Sep. 2022, doi: <https://doi.org/10.1371/journal.pone.0274646>.
- [4] X.-M. LIN, Y.-L. WANG, Y.-F. LIU, and J.-X. LU, "Research on Global Path Planning Method of Mobile Robot Based on BAS," *Journal of Physics: Conference Series*, vol. 1284, no. 1, p. 012014, Aug. 2019, doi: <https://doi.org/10.1088/1742-6596/1284/1/012014>.
- [5] W. F. Sharpe, "Mutual Fund Performance," *The Journal of Business*, vol. 39, no. 1, pp. 119–138, 1966.
- [6] X. Jiang and S. Li, "BAS: Beetle Antennae Search Algorithm for Optimization Problems," *International Journal of Robotics and Control*, vol. 1, no. 1, p. 1, Apr. 2018, doi: <https://doi.org/10.5430/ijrc.v1n1p1>.
- [7] J. Wang and H. Chen, "BSAS:Beetle Swarm Antennae Search Algorithm for Optimization Problems," Jul. 2018.

A. Implementation of main algorithm

```
import pandas_datareader.data as web
import datetime as dt
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from yahoo_fin.stock_info import get_data
from numpy import random
from scipy.optimize import minimize
from scipy.stats import shapiro
from tqdm import tqdm
import seaborn as sns
import scipy.optimize as sco
import time
import numpy as np

# Define function to get stock's adjusted close price
def ticker_adjclose(sd,ed,ticker):
    ticker_adj = get_data(ticker, start_date=sd, end_date=ed,
index_as_date = True, interval="1d")['adjclose']

    data = {ticker+'_adjclose':ticker_adj}

    ticker_adjclose = pd.DataFrame(data)

    return ticker_adjclose

# Define function to calculate stock's normalized return
def ticker_normedR(ticker_name,ticker_adjclose=[]):
```

```

        ticker_adjclose[ticker_name+'_Normed_Return'] =
ticker_adjclose[ticker_name+'_adjclose']/ticker_adjclose.iloc[
0][ticker_name+'_adjclose']

# Define function to get stock's adjusted close price
def get_ticker_adjclose(money,start_time,end_time,ticker=[]):
    ticker_num=len(ticker)

    ticker_name =ticker

    hole_data = pd.DataFrame()

    a=0

    allo = get_allo(ticker_num)

    while a < ticker_num:

        ticker_data=ticker_adjclose(start_time,end_time,ticker_name[a])

        ticker_normedreturn=
ticker_normedR(ticker_name[a],ticker_data)

        allocation (allo[a],ticker_data,ticker_name[a])

        positionValue(money,ticker_data,ticker_name[a])

        hole_data=pd.concat([hole_data,ticker_data],axis=1, join =
'outer')

        a += 1

    return hole_data

# Define function to get allocation weight of each stock
def allocation (allocation,stock_data,ticker_name):

    stock_data[ticker_name+'_Allocation'] =
stock_data[ticker_name+'_Normed_Return'] *allocation

# Define function to calculate position value of each stock
def positionValue (money,stock_data,ticker_name):

```



```

        stock_data[ticker_name+'_Position_Value'] =
stock_data[ticker_name+'_Allocation'] *money

# Define function to generate random allocation weights
def get_allo(num):

    random_1= np.random.random(num)

    random_2= sum(random_1)

    a = 0

    allo = []

    while a < len(random_1):

        allo.insert(a, random_1[a]/random_2)

        a +=1

    return allo

# Define function to simulate Sharpe ratio
def
sharpeSimulation(simulateNum,money,ticker=[],tickerName=[]):

    loop= 0

    ticker_num=len(tickerName)

    ticker_cal= ticker.copy()

    sharpe_Simulate = pd.DataFrame({tickerName[0]: []},)

    while loop < simulateNum:

        portfolio_val = pd.DataFrame()

        allo = get_allo(ticker_num)

        a=0

        while a<ticker_num:

```

```

allocation (allo[a],ticker_cal,tickerName[a])

positionValue (money,ticker_cal,tickerName[a])

                                portfolio_val =
pd.concat([portfolio_val,ticker_cal[tickerName[a]+'_Position_V
alue']],axis=1, join = 'outer')

if loop==0 :

    sharpe_Simulate[tickerName[a]]=0

if loop ==0 and a==ticker_num-1:

    sharpe_Simulate['Final_Total']=0

    sharpe_Simulate['ASR']=0

    sharpe_Simulate['annual Return mean']=0

    sharpe_Simulate['annual Return st']=0

a +=1

portfolio_val.columns = tickerName

portfolio_val['Total'] = portfolio_val.sum(axis=1)

                                portfolio_val['Daily Return'] =
portfolio_val['Total'].pct_change(1)

# Calculate Sharpe ratio and annualized Sharpe ratio

    sharpe_ratio = portfolio_val['Daily Return'].mean() /
portfolio_val['Daily Return'].std()

ASR = (252**0.5) * sharpe_ratio

# Add portfolio total value, annualized Sharpe ratio,
annual return mean, and standard deviation to allo list

```

```

    allo.append(portfolio_val['Total'].iloc[-1])

    allo.append(ASR)

    allo.append(portfolio_val['Daily Return'].mean())

    allo.append(portfolio_val['Daily Return'].std())

    # Add allo list data to sharpe_Simulate DataFrame

    sharpe_Simulate.loc[loop]=allo

    loop +=1

    return sharpe_Simulate,portfolio_val

# Define function to calculate daily return for each stock
def get_daily_return(ticker_name=[],ticker_list=[]):

    ticker_num=len(ticker_name)

    ticker_listc=ticker_list

    daily_return = pd.DataFrame()

    i=0

    while i < ticker_num:

        daily_return[ticker_name[i]] =
ticker_listc[ticker_name[i]+'_Normed_Return'].pct_change(1)

        i +=1

    return daily_return

# Define stock list and date range

ticker_list1=['MMM', 'AOS', 'ABT', 'ABBV', 'ACN', 'ATVI',
'ADM', 'ADBE', 'ADP', 'AAP', 'AES', 'AFL', 'A', 'APD', 'AKAM',
'ALK', 'ALB', 'ARE', 'ALGN', 'ALLE', 'LNT', 'ALL', 'GOOGL',
'GOOG', 'MO', 'AMCR', 'AMD', 'AEE', 'AAL', 'AEP', 'AXP',
'AIG', 'AMT', 'AWK', 'AMP', 'ABC', 'AME', 'AMGN', 'APH',
'ADI', 'ANSS', 'AON', 'APA', 'AMAT', 'APTV', 'ACGL', 'ANET',
'AJG', 'AIZ', 'T', 'ATO', 'ADSK', 'AZO', 'AVB', 'AVY', 'BKR',

```

```

'BALL', 'BAC', 'BBWI', 'ASML', 'KO', 'NVDA', 'TSM', 'META',
'JPM', 'WMT', 'TGT', 'MSFT', 'AMZN', 'AAPL', 'TSLA', 'IBM',
'ORCL', 'JPM-PC', 'BACHY', 'RY', 'HSBC', 'CICHF', 'XOM',
'CVX', 'SHEL', 'RYDAF', 'COP', 'TTE', 'UNH', 'JNJ', 'LLY',
'PFE', 'AZN']

start_date = "01/01/2020"

end_date = "03/30/2023"

# Get adjusted close price data for each stock

ticker1=get_ticker_adjclose(10000,start_date,end_date,ticker_list1)

# Calculate daily return for each stock

daily_returns = get_daily_return(ticker_list1, ticker1)

# Calculate annualized return mean and covariance matrix

mus = (1 + daily_returns.mean())**252 - 1

cov = daily_returns.cov() * 252

# Assuming you already have daily return data (daily_returns)
and calculated stock return (mus) and covariance matrix (cov)

n_assets = 20

```

Listings A.1: A full implementation of data acquisition and processing.

```

# Define a function to calculate the Sharpe ratio

def sharpe_ratio_function(weights, mean_returns, cov_matrix,
rf):

    # Calculate the portfolio return and volatility

    portfolio_return = np.dot(weights, mean_returns) - rf

    portfolio_volatility = np.sqrt(np.dot(weights.T,
np.dot(cov_matrix, weights)))

    # Compute the Sharpe ratio

    sharpe_ratio = portfolio_return / portfolio_volatility

```

```

        return -sharpe_ratio

# Define constraint function
def constraint_function(weights):
    return np.sum(weights) - 1

# Define the Beetle Antenna Search (bsas) algorithm for
portfolio optimization
def beetle_antenna_search(f, dim, min_, max_, n_iterations,
mean_returns, cov_matrix, rf):
    # Initialization

    x = np.abs(np.random.randn(dim)) # Initialize a random
weight vector

    x /= np.sum(x) # Normalize the weight vector

    step = 1 # Initialize the step size

    fleft = float('inf')
    fright = float('inf')

    # Main loop for optimization
    for i in range(n_iterations):
        # Generate a random search direction and normalize it
        dir = np.random.randn(dim)
        dir = dir / np.linalg.norm(dir, ord=1, axis=0)

        step = 0.95 * step # Gradually reduce the step size

        # Compute the new candidate solutions
        d0 = step / 2

```

```

    x1 = x + d0 * dir / 2

    xr = x - d0 * dir / 2

    # Calculate the objective function values for the
candidate solutions

    fleft = f(x1, mean_returns, cov_matrix, rf)

    fright = f(xr, mean_returns, cov_matrix, rf)

    # Update the current position based on the comparison
of the objective function values

    tx = x - step * dir * np.sign(fleft - fright)

    # Normalize the updated position

    tx = np.abs(tx)

    tx /= np.sum(tx)

    # Check if the updated position satisfies the
constraints and update it accordingly

    if np.any(np.abs(tx) > max_) or np.any(np.abs(tx) <
min_):

        continue

    else:

        x = tx

    # Return the optimized weight vector and the minimum value
of the objective function

    return x, min(fleft, fright)

# Define the Beetle Swarm Search (BSAS) algorithm for
portfolio optimization

```

```

def beetle_swarm_search(f, swarm_size, dim, min_, max_,
n_iterations, mean_returns, cov_matrix, rf):

    # Initialize a swarm of beetles with random weights

    beetles = [np.random.rand(dim) for _ in range(swarm_size)]

    # Normalize the weights of each beetle in the swarm

    for beetle in beetles:

        beetle /= np.sum(beetle)

    # Initialize the best position and best value to track the
    global minimum

    best_position = None

    best_value = float('inf')

    # Main loop for swarm search optimization

    for i in range(n_iterations):

        # Initialize a list to store the new positions for
        each beetle in the swarm

        new_positions = []

        # Iterate through each beetle in the swarm

        for beetle in beetles:

            # Apply the bsas algorithm to find the new
            position and value for the current beetle

            new_position, new_value = beetle_antenna_search(f,
dim, min_, max_, 250, mean_returns, cov_matrix, rf)

            # Update the global best position and value if a
            better solution is found

            if new_value < best_value:

```

```

        best_position = new_position

        best_value = new_value

        # Add the new position to the list of
new_positions

        new_positions.append(new_position)

        # Update the swarm with the new position

        beetles = new_positions

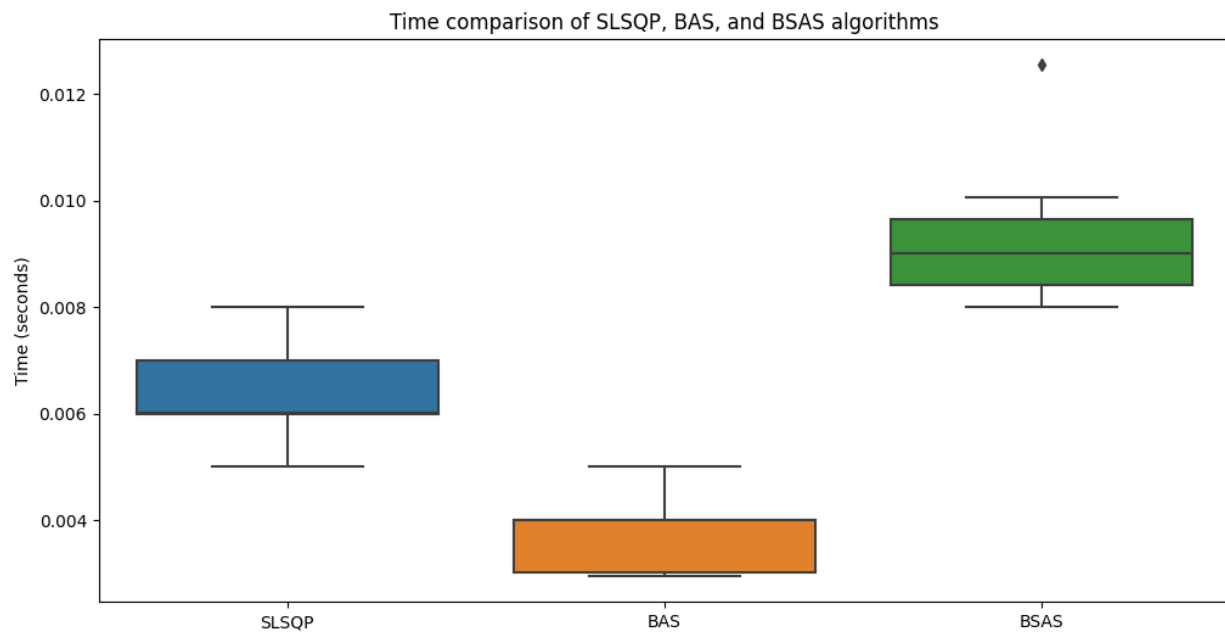
        # Return the optimized weight vector and the minimum value
of the objective function

        return best_position, best_value

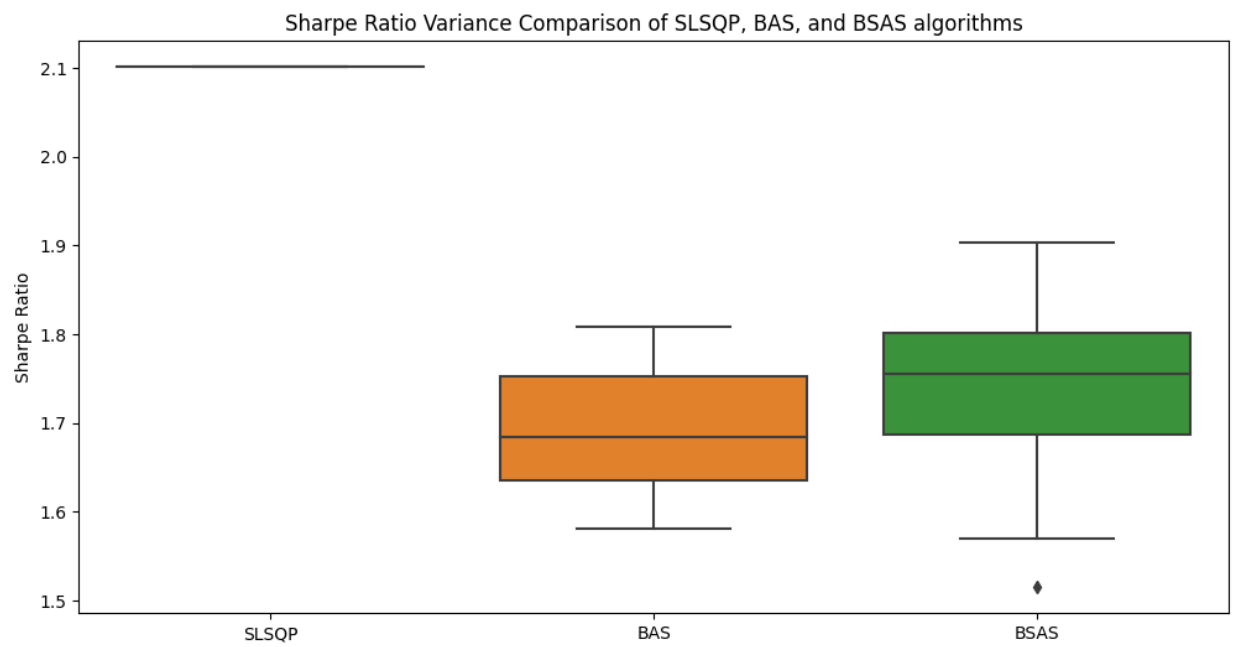
```

Listings A.2: A full implementation of BAS and BSAS algorithms

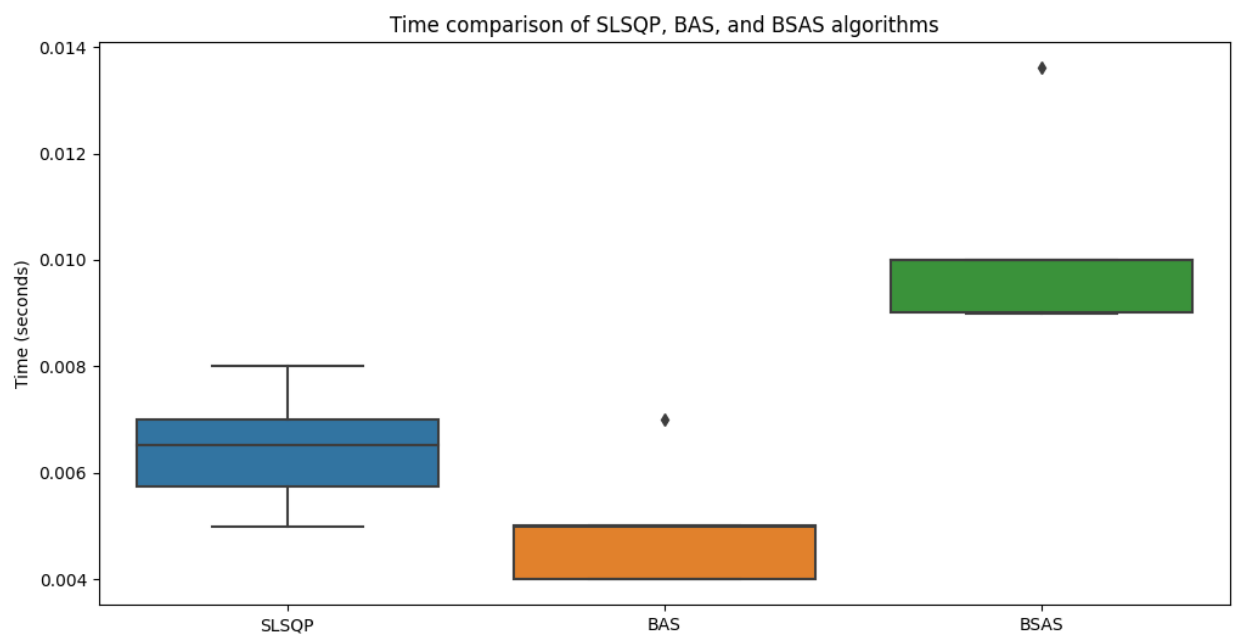
B. Figure



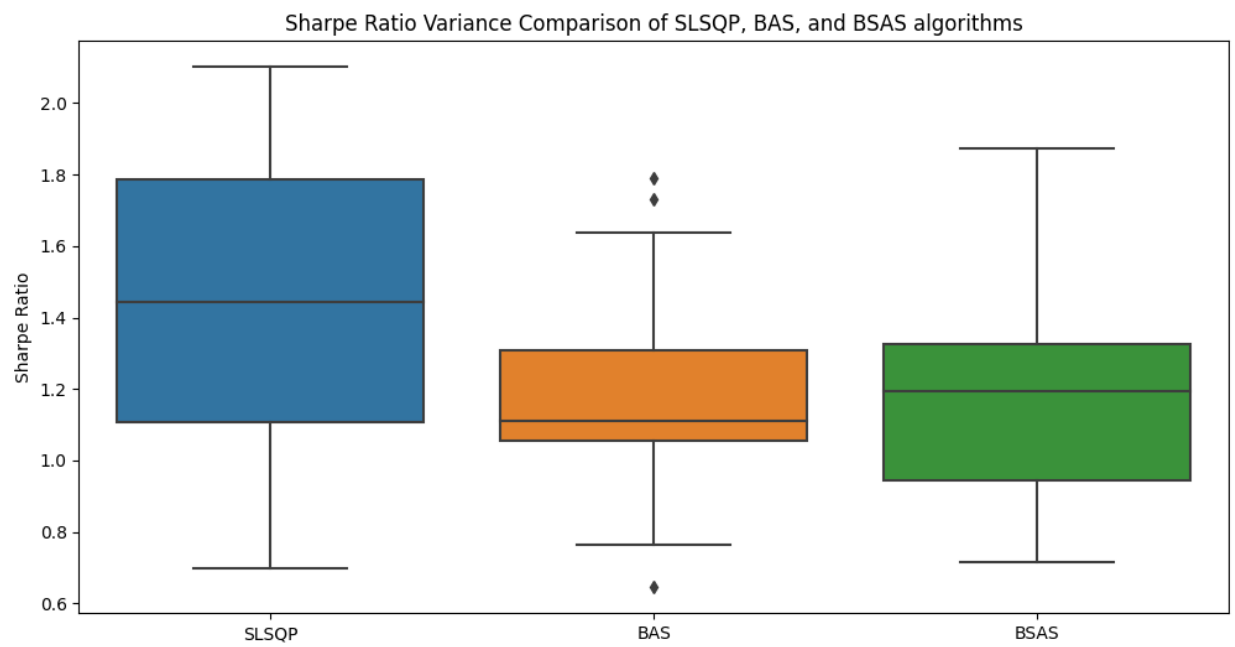
Figures B.1: Time comparison of three algorithms(same portfolio)



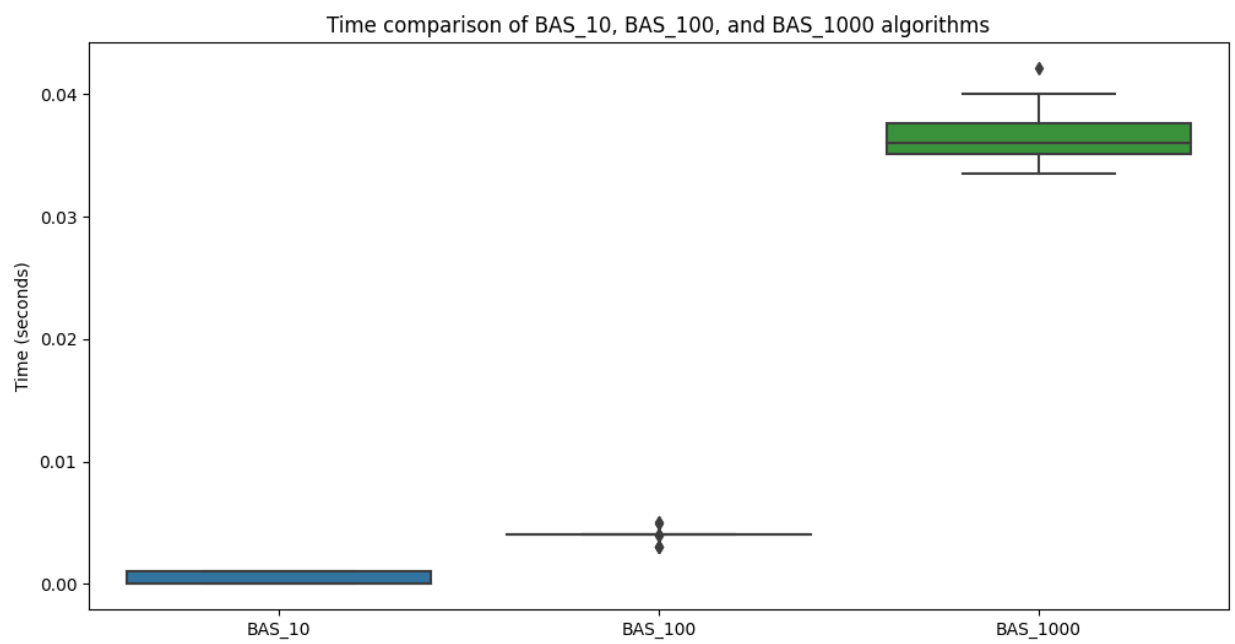
Figures B.2: Sharpe ratio comparison of three algorithms(same portfolio)



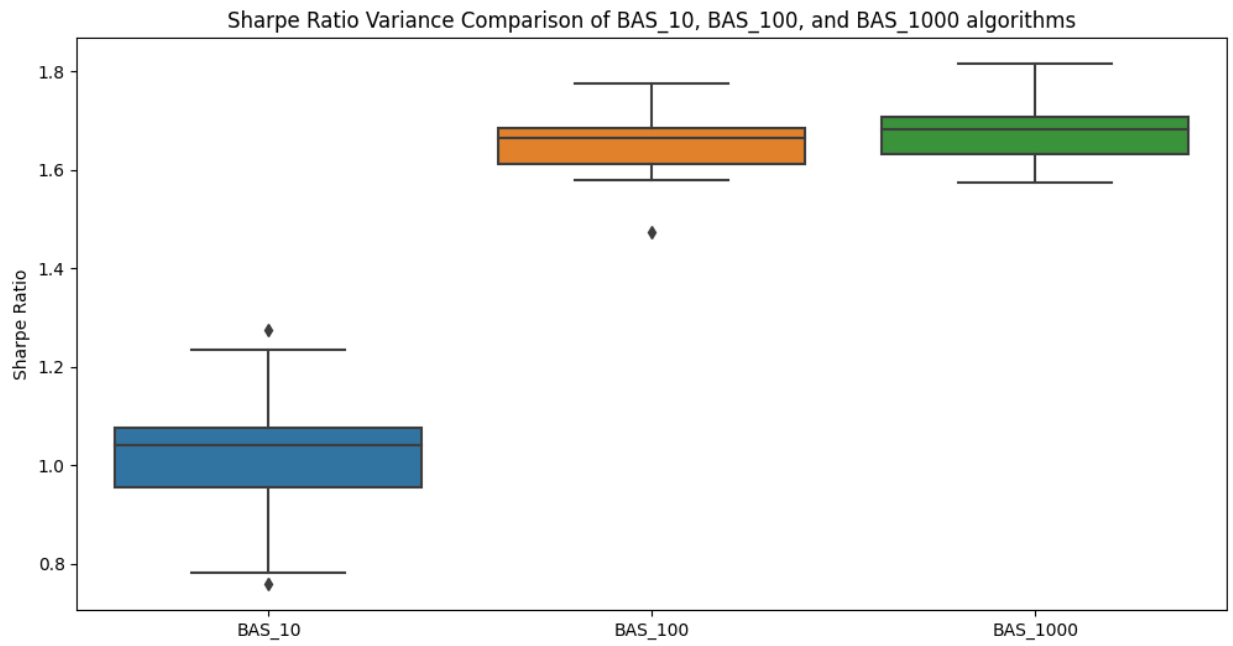
Figures B.3: Time comparison of three algorithms(different portfolio)



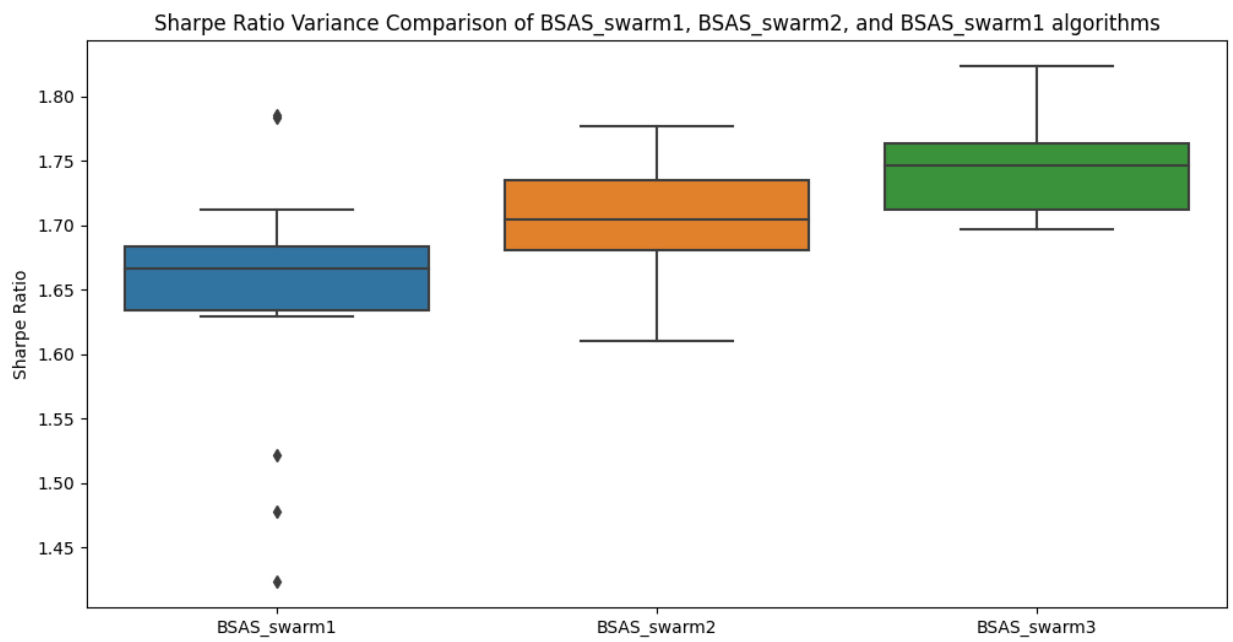
Figures B.4: Sharpe ratio comparison of three algorithms(different portfolio)



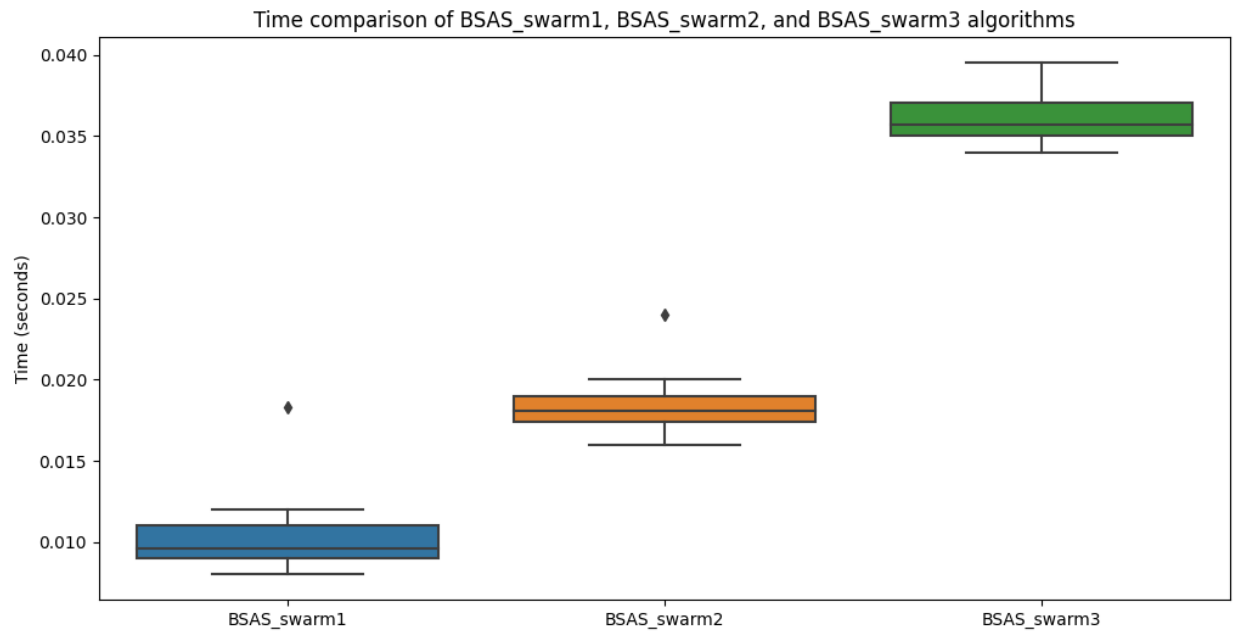
Figures B.5: Time comparison of BAS algorithm in different iteration



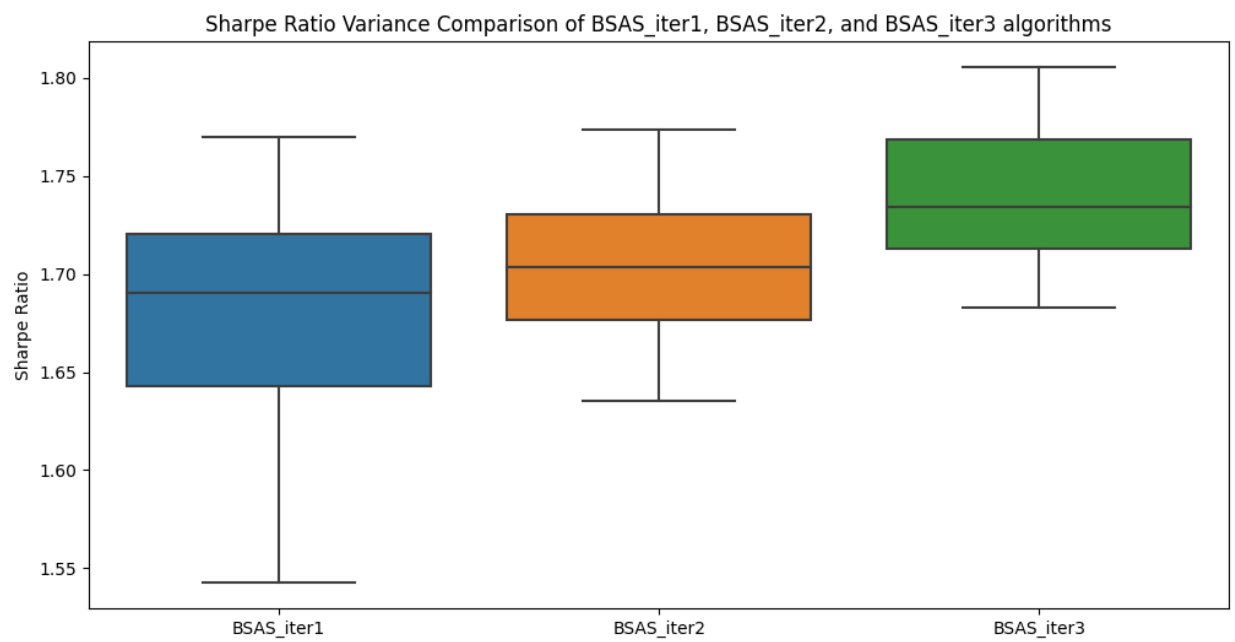
Figures B.6: Sharpe ratio comparison of BAS algorithm in different iteration



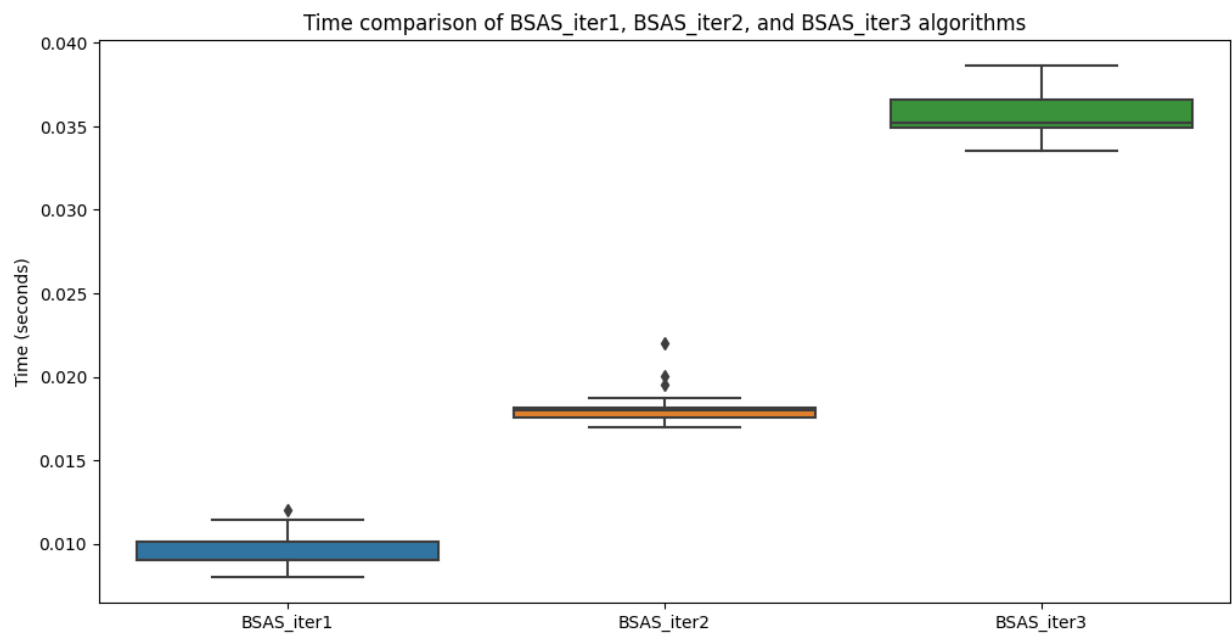
Figures B.7: Sharpe ratio comparison of BSAS algorithm in different swarm



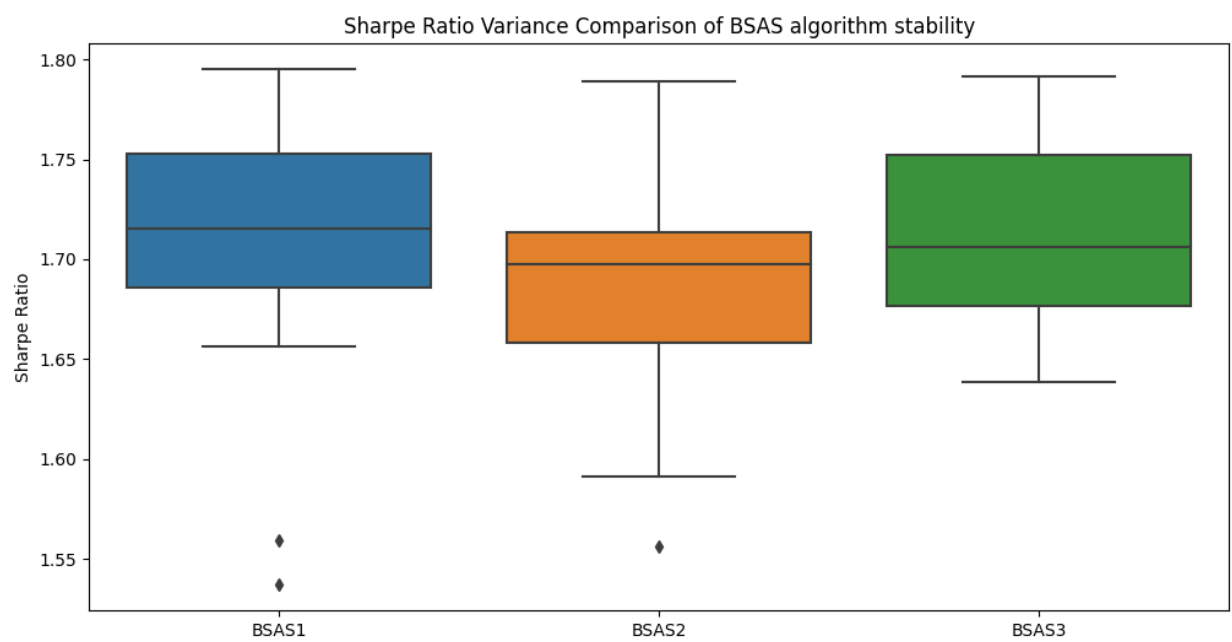
Figures B.8: Time comparison of BSAS algorithm in different swarm



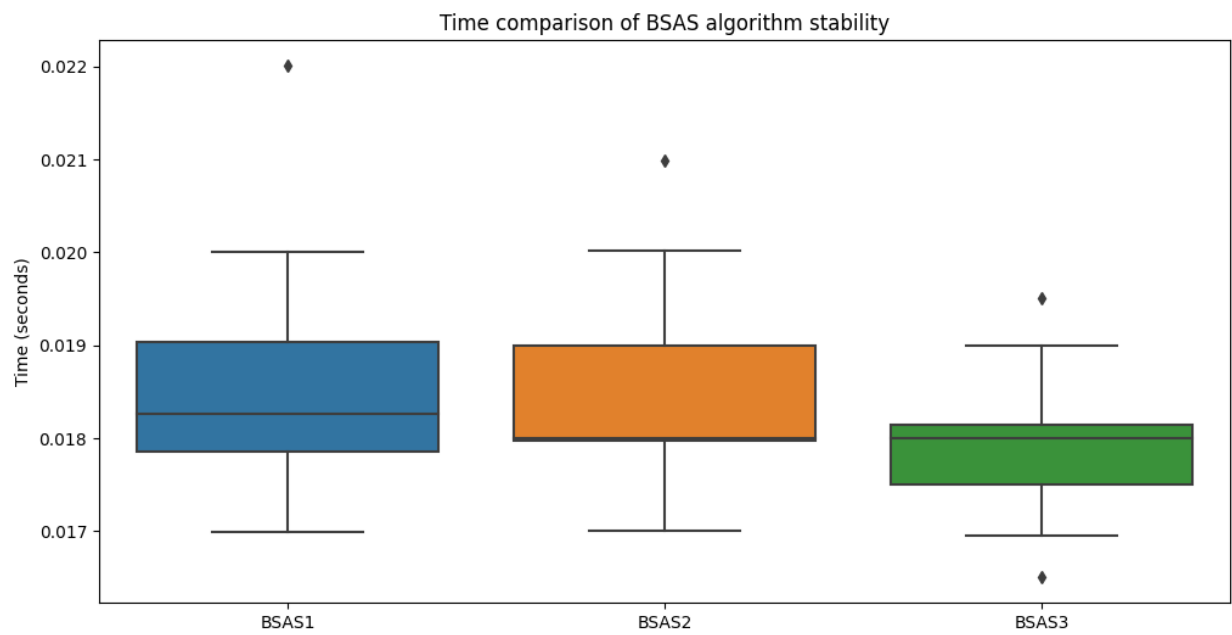
Figures B.9: Sharpe ratio comparison of BSAS algorithm in different iteration



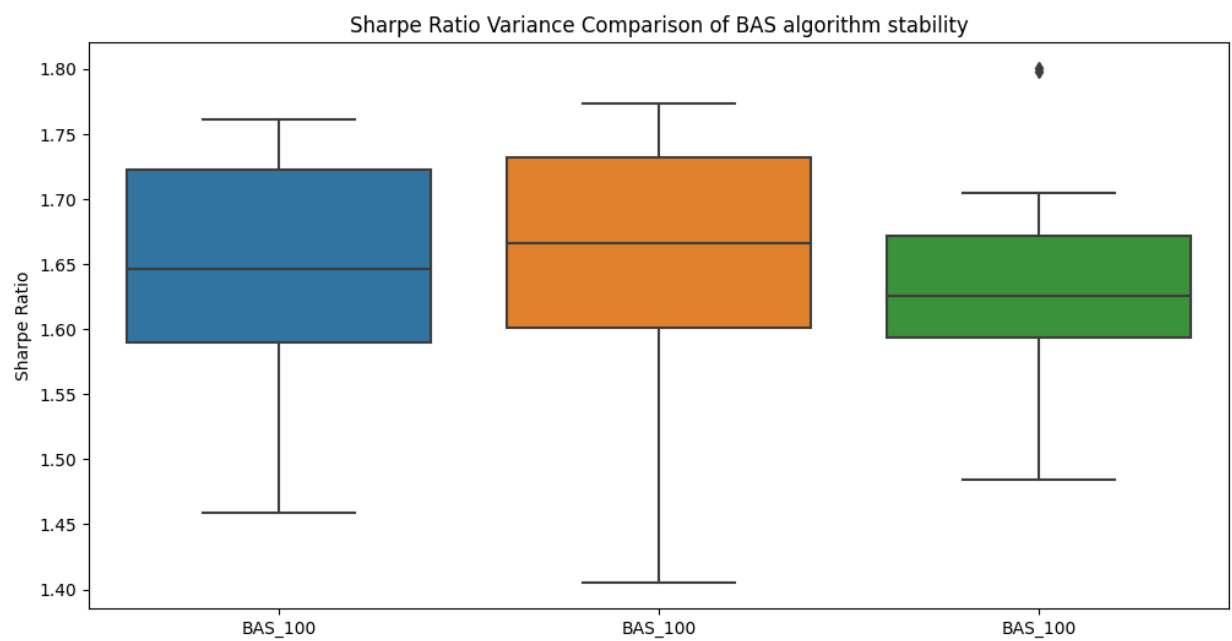
Figures B.10: Time comparison of BSAS algorithm in different iteration



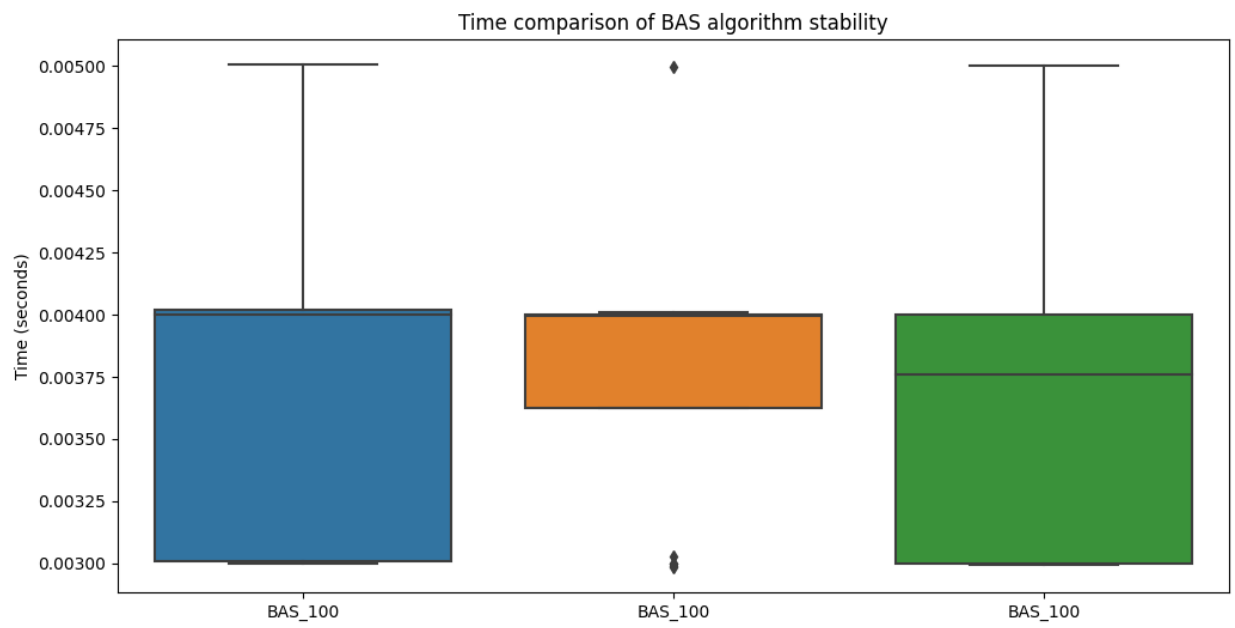
Figures B.11: Sharpe ratio comparison of BSAS algorithm stability



Figures B.12: Time comparison of BSAS algorithm stability



Figures B.13: Sharpe ratio comparison of BAS algorithm stability



Figures B.14: Time comparison of BAS algorithm stability