Methods for Portfolio Optimization

Giovanni Piva[†], Roberto Vicentini[‡]

***Abstract*—This report explores portfolio optimization using different variations of the Frank-Wolfe (FW) algorithm within the framework of Markowitz portfolio theory. The methods analyzed include the Frank-Wolfe algorithm, Pairwise Frank-Wolfe and Away-Step Frank-Wolfe. We will examine the application of these FW algorithms to several stock indices, such as FTSE MIB, EuroStoxx50, and FTSE100. The performance of each algorithm will be evaluated based on metrics like loss, risk, return, duality gap, and execution time, providing practical insights into their effectiveness in financial portfolio management.**

I. Introduction

This project focuses on the Frank-Wolfe method, an optimization algorithm used to solve convex programming problems. We will begin by discussing the theoretical aspects, presenting all the algorithms that will be used and highlighting the differences between them. Following this, we will introduce the datasets employed in the experimental phase and present the results obtained with each algorithm.

In general, our optimization problem uses the Frank-Wolfe algorithm to solve problems of the form:

$$\min_{x \in D} f(x)$$

where D is a convex set and $f(x)$ is a convex function.

The Frank-Wolfe algorithm, also known as the conditional gradient method, is particularly suited for large-scale optimization problems due to its iterative approach that leverages linear optimization at each step. The variations of the Frank-Wolfe algorithm that we will explore include the Frank-Wolfe algorithm, Pairwise Frank-Wolfe and Away-Step Frank-Wolfe. Each of these variations offers distinct strategies for improving convergence rates and handling different types of constraints within the optimization problem.

[†]Department of Mathematics, University of Padova
roberto.vicentini@studenti.unipd.it
giovanni.piva.4@studenti.unipd.it

II. Loss Function

In this section, we present the loss function used for the portfolio optimization problem. This function is formulated to balance the trade-off between risk and return, following the principles of Markowitz mean-variance optimization.

We have n available assets. We denote by w_i the quantity of money invested in the i-th asset during the considered period and by r_i the returns on the i-th asset. We have two different constraints. The first one is non-negativity for the variables (i.e., $w_i \geq 0$). It basically means that short selling (selling an asset that we still don't own) is not allowed. We then have the budget constraint:

$$\sum_{i=1}^{n} w_i = B \tag{1}$$

where the total amount of money invested needs to be equal to the budget B (which can be simply set to 1). Consider a stochastic model for the returns: $\mathbf{r} \in \mathbb{R}^n$ is a randomly generated vector with mean $\overline{r}$ and covariance Σ. Thus the expected return will be:

$$\overline{r}^{\intercal}\mathbf{w} \tag{2}$$

and the variance (risk) is:

$$\mathbf{w}^{\intercal}\Sigma\mathbf{w} \tag{3}$$

The classic portfolio problem, described by Markowitz (1952), is a convex quadratic programming problem:

$$\begin{aligned} \underset{w \in \mathbb{R}^n}{\text{minimize}} \quad & \gamma\, \mathbf{w}^\mathsf{T} \Sigma \mathbf{w} - \bar{r}^\mathsf{T} \mathbf{w} \\ \text{subject to} \quad & \mathbf{e}^\mathsf{T} \mathbf{w} = 1 \\ & \mathbf{w} \geq 0 \end{aligned} \tag{4}$$

where $\gamma > 0$ is the risk-aversion parameter. The goal is to find the set of assets that minimizes the variance (risk associated with the given portfolio) while maximizing the expected return. We obviously need to satisfy the budget and non-negativity constraints.

III. Frank Wolfe

In this section, we present the implementation of the line search variant of the Frank-Wolfe algorithm used for portfolio optimization. The goal of this algorithm is to find the optimal portfolio weights that minimize the loss function, which balances the trade-off between risk and return.

The main steps of the algorithm are as follows:

1) **Initialization**:

$$\mathbf{w} \leftarrow \frac{1}{n}\mathbf{e} \tag{5}$$

where n is the number of assets and $\mathbf{e}$ is a vector of ones.

2) **Gradient Calculation**:

$$\nabla f(\mathbf{w}) = 2\gamma \Sigma \mathbf{w} - \bar{r}^T \tag{6}$$

where Σ is the covariance matrix, $\bar{r}$ is the vector of expected returns, and γ is the risk-aversion parameter.

3) **Direction Finding**:

$$s_i = \begin{cases} 1 & \text{if } i = \arg\min \nabla f(\mathbf{w}) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

4) **Line Search**:

$$\gamma^* = \arg \min_{\gamma \in [0,1]} f((1-\gamma)\mathbf{w} + \gamma \mathbf{s}) \tag{8}$$

This is done by evaluating the loss function at a range of γ values and selecting the one that gives the minimum loss.

5) **Update Weights**:

$$\mathbf{w} \leftarrow (1-\gamma^*)\mathbf{w} + \gamma^* \mathbf{s} \tag{9}$$

6) **Convergence Check**: Check if:

$$\|\nabla f(\mathbf{w})\| < \epsilon \text{ or Duality Gap} < \epsilon \tag{10}$$

If true, terminate the algorithm.

The algorithm iterates through these steps until convergence or until a predefined number of iterations is reached. The resulting optimal weights are then used to construct the efficient portfolio.

Algorithm 1 Frank-Wolfe Algorithm

```
1: Initialize $\mathbf{w} \leftarrow \frac{1}{n}\mathbf{e}$
2: for each iteration do
3:     Compute gradient $\nabla f(\mathbf{w})$
4:     Determine direction s such that $s_i = 1$ for $\arg\min \nabla f(\mathbf{w})$, $0$ otherwise
5:     Perform line search to find $\gamma^*$
6:     Update weights: $\mathbf{w} \leftarrow (1-\gamma^*)\mathbf{w} + \gamma^*\mathbf{s}$
7:     Project $\mathbf{w}$ to ensure non-negativity and normalization
8:     if $\|\nabla f(\mathbf{w})\| < \epsilon$ or DGap $< \epsilon$ then
9:         Break
10:    end if
11: end for
12: return optimal weights $\mathbf{w}$
```

IV. Pairwise Frank Wolfe

The pairwise variant of the Frank-Wolfe algorithm differs from the standard Frank-Wolfe algorithm primarily in the way it determines the direction for updating the portfolio weights. The other steps, such as initialization, gradient calculation, line search, projection, and convergence check, remain the same as described previously.

The main differences in the Pairwise Frank-Wolfe algorithm are as follows:

1) **Direction Finding**: Instead of selecting a single direction, the pairwise variant finds

two indices: one with the minimum gradient and one with the maximum gradient:

$$i = \arg\min \nabla f(\mathbf{w}), \quad j = \arg\max \nabla f(\mathbf{w})$$

The direction is then defined as:

$$\mathbf{s} = \mathbf{e}_i - \mathbf{e}_j$$

where $\mathbf{e}_i$ and $\mathbf{e}_j$ are the standard basis vectors.

2) **Weight Update**: The weights are updated using the direction s and the optimal step size γ found via line search:

$$\mathbf{w} \leftarrow \mathbf{w} + \gamma^* \mathbf{s}$$

Here is the simplified pseudo codes:

Algorithm 2 Pairwise Frank-Wolfe Algorithm

```
1: Initialize $\mathbf{w} \leftarrow \frac{1}{n}\mathbf{e}$
2: for each iteration do
3:     Compute gradient $\nabla f(\mathbf{w})$
4:     Find indices $i = \arg\min \nabla f(\mathbf{w})$ and $j = \arg\max \nabla f(\mathbf{w})$
5:     Determine direction $\mathbf{s} = \mathbf{e}_i - \mathbf{e}_j$
6:     Perform line search to find $\gamma^*$
7:     Update weights: $\mathbf{w} \leftarrow \mathbf{w} + \gamma^* \mathbf{s}$
8:     Project w to ensure non-negativity and normalization
9:     if $\|\nabla f(\mathbf{w})\| < \epsilon$ or DGap $< \epsilon$ then
10:        Break
11:    end if
12: end for
13: return optimal weights w
```

V. Away-Step Frank-Wolfe

The Away-Step Frank-Wolfe variant enhances the standard Frank-Wolfe algorithm by introducing an "away" direction to improve convergence, especially when the current point is near the boundary of the feasible region. The main differences between the standard Frank-Wolfe algorithm and the Away-Step Frank-Wolfe algorithm are as follows:

1. **Away Direction**: In addition to the Frank-Wolfe direction, an away direction is computed. The away direction is determined by finding the vertex of the current active set that has the maximum gradient:

$$j = \arg\max(\nabla f(w) \cdot V), \quad v_j = V[j]$$

$$d_{away} = w - v_j$$

2. **Optimal Direction and Step Size**: The algorithm chooses between the Frank-Wolfe direction and the away direction based on which reduces the objective function the most. The step size is also adjusted accordingly:

```
1: if $\nabla f(w) \cdot d_{fw} < \nabla f(w) \cdot d_{away}$ then
2:     $direction \leftarrow d_{fw}$
3:     $step\_size \leftarrow 1.0$
4: else
5:     $direction \leftarrow -d_{away}$
6:     $max\_step\_size \leftarrow \min\left(\frac{w}{w - v_j}\right)$
7:     $step\_size \leftarrow \min(1.0, max\_step\_size)$
8: end if
```

In the next page (for a matter of space and clarity) the simplified pseudo code.

Algorithm 3 Away Step Frank-Wolfe Algorithm

```
1: Initialize $\mathbf{w} \leftarrow \frac{1}{n}\mathbf{e}$
2: for each iteration do
3:     Compute gradient $\nabla f(\mathbf{w})$
4:     $\mathbf{s}$ = linearMinimizationOracle($\nabla f(\mathbf{w})$)
5:     $\mathbf{d}_{\text{fw}} = \mathbf{s} - \mathbf{w}$
6:     $j = \arg\max \mathbf{V} \cdot \nabla f(\mathbf{w})$
7:     $\mathbf{v}_j = \mathbf{V}[j]$
8:     $\mathbf{d}_{\text{away}} = \mathbf{w} - \mathbf{v}_j$
9:     if $\nabla f(\mathbf{w}) \cdot \mathbf{d}_{\text{fw}} < \nabla f(\mathbf{w}) \cdot \mathbf{d}_{\text{away}}$ then
10:        direction = $\mathbf{d}_{\text{fw}}$
11:        step_size = $1.0$
12:    else
13:        direction = $-\mathbf{d}_{\text{away}}$
14:        max_step_size = $\min(\mathbf{w}/(\mathbf{w} - \mathbf{v}_j))$
15:        step_size = $\min(1.0, \text{max\_step\_size})$
16:    end if
17:    Perform line search to find $\gamma^*$
18:    $\gamma = \min(\gamma^*, \text{step\_size})$
19:    Update weights: $\mathbf{w} \leftarrow \mathbf{w} + \gamma\mathbf{s}$
20:    Project $\mathbf{w}$ to ensure non-negativity and normalization
21:    if direction = $\mathbf{d}_{\text{fw}}$ then
22:        $V \leftarrow V \cup \{\mathbf{s}\}$
23:    else
24:        if step_size = max_step_size and $|V| > 1$ then
25:            Remove vertex: $V \leftarrow V \setminus \{\mathbf{v}_j\}$
26:        end if
27:    end if
28:    if $\|\nabla f(\mathbf{w})\| < \epsilon$ or DGap $< \epsilon$ then
29:        Break
30:    end if
31: end for
32: return optimal weights $\mathbf{w}$
```

VI. Experiment

In this section, we evaluate the algorithms on real-world financial data.

A. Dataset

In this section, we describe the datasets used for our portfolio optimization problem. We have three datasets with weekly returns for different stock indices: FTSE MIB, Euro Stoxx 50, and FTSE 100. Each dataset includes both the index returns and the individual stock returns, and is composed of two files:

- **IndRet**: This file contains a column vector of weekly returns for the specific stock index (e.g., FTSE MIB, Euro Stoxx 50, FTSE 100) from January 22, 2007, to May 6, 2013.
- **RR**: This file contains a matrix where each column represents the weekly returns of individual stocks within the specific index.

All datasets include weekly returns calculated from adjusted prices that include dividends. Stocks with more than two consecutive missing values were removed, while individual missing values were interpolated. This ensures the quality and completeness of the data for analysis. Each dataset covers the period from January 22, 2007, to May 6, 2013, providing a robust timeframe for evaluating the performance of different portfolio optimization algorithms.

The datasets offer a comprehensive basis for analyzing stock returns and applying various portfolio optimization algorithms. We will use these datasets separately to compare the performance of the Frank-Wolfe algorithm, Pairwise Frank-Wolfe, and Away-Step Frank-Wolfe. By doing so, we aim to evaluate the efficiency of these algorithms in optimizing a financial portfolio. In our final considerations, we will take into account the loss, risk, and return obtained by the various algorithms.

B. Algorithms Results

In this section, we will analyze the results of the three algorithms (Frank-Wolfe, Pairwise Frank-Wolfe and Away-Step Frank-Wolfe) applied to the three datasets (FTSE MIB, Euro Stoxx 50 and FTSE 100) with a fixed risk aversion of 6.

*1) **Loss Analysis**:* First, we will analyze the loss for each algorithm on the three different datasets. The loss function measures the performance of the portfolio optimization, balancing the trade-off between risk and return.

We plot the loss for each algorithm across the three datasets. This helps us see which algorithm performs better in terms of minimizing the loss.

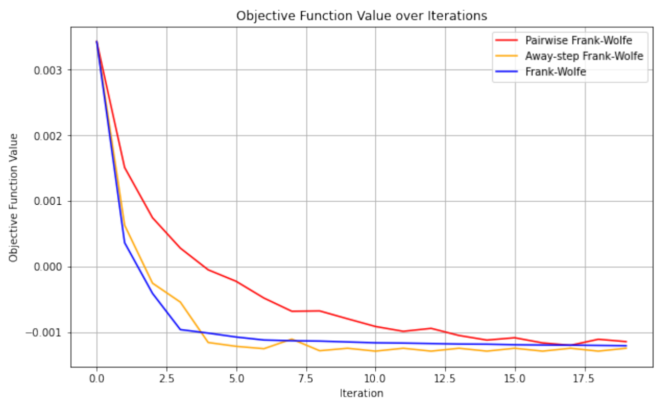

Fig. 1: Euro Stoxx 50

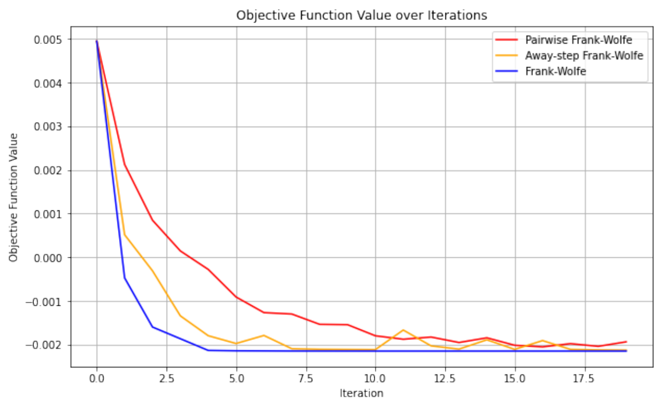

Fig. 2: Ftse MIB

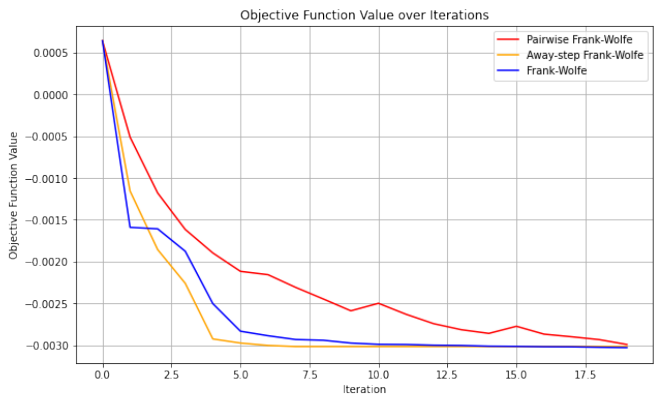

Fig. 3: Ftse 100

The Standard Frank-Wolfe algorithm consistently converges faster and reaches a lower loss value compared to the Pairwise and Away-step variations. In terms of performance, the Pairwise Frank-Wolfe algorithm performs better than the Away-step Frank-Wolfe but not as well as the Standard Frank-Wolfe in minimizing the loss. Additionally, the performance trends of the algorithms are consistent across the three datasets.

Overall, the Standard Frank-Wolfe algorithm appears to be the most efficient and effective in minimizing the loss across all datasets.

*2) **Duality Gap Analysis**:* Next, we will analyze the duality gap for each algorithm on the three datasets. The duality gap indicates how close the current solution is to the optimal solution. A smaller duality gap means the solution is closer to optimal.

We plot the duality gap for each algorithm across the three datasets. This comparison shows the convergence behavior of the algorithms.

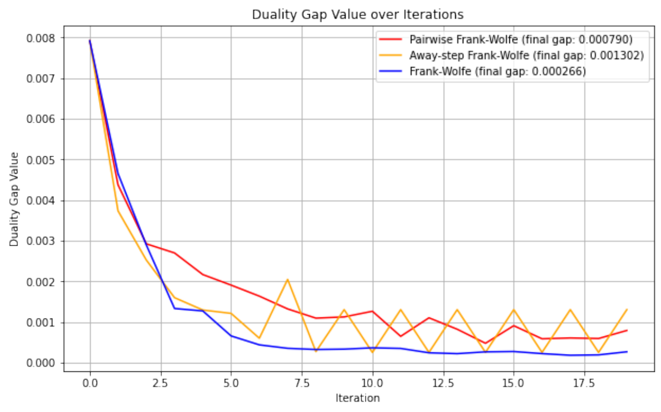

Fig. 4: Euro Stoxx 50

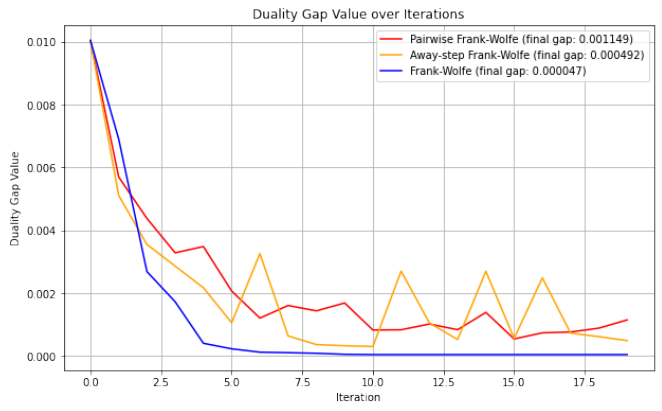

Fig. 5: Ftse MIB

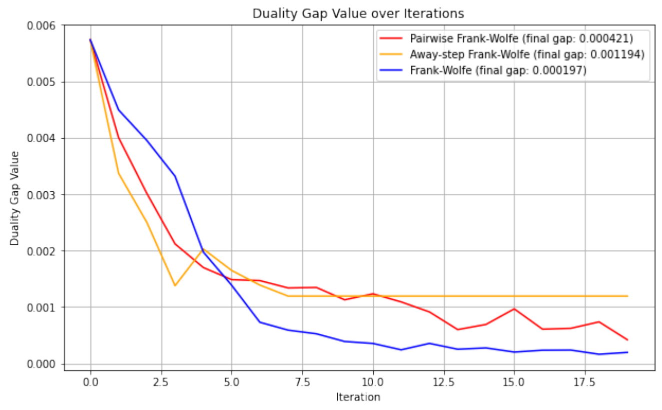

Fig. 6: Ftse 100

The Standard Frank-Wolfe algorithm consistently achieves the lowest final duality gap across all datasets, demonstrating superior convergence properties. Although the Pairwise Frank-Wolfe algorithm shows better results than the Away-step Frank-Wolfe, it does not match the effectiveness of the Standard Frank-Wolfe.

This pattern holds true across all three datasets, confirming the robustness and efficiency of the Standard Frank-Wolfe algorithm in reducing the duality gap.

*3) **Execution Time Analysis**:* We will also analyze the execution time for each algorithm on the three datasets. The execution time is important to understand the computational efficiency of the algorithms.

We create a table showing the execution time for each algorithm on the three datasets. This helps us identify which algorithm is the fastest.

Dataset	FW	PFW	AFW
Euro Stoxx 50	35ms	33ms	32ms
FTSE MIB	40ms	41ms	35ms
FTSE 100	30ms	32ms	28ms

TABLE 1: Execution Time for Each Algorithm on the Three Datasets

The Away-step Frank-Wolfe algorithm has the fastest execution time across all three datasets, making it the most efficient in terms of computation time. The Frank-Wolfe and Pairwise Frank-Wolfe algorithms have slightly higher execution times, but the differences are relatively small. This indicates that while AFW is the quickest, all three algorithms have comparable execution times and are efficient in their computation.

*4) **Risk and Return Analysis**:* We will analyze the risk and return for each algorithm on the three datasets. Risk is measured as the portfolio variance, and return is the expected return of the portfolio.

We compare the optimal risk and weekly return for each algorithm across the three datasets. This helps us understand the trade-offs between risk and return for each algorithm.

Dataset	FW	PFW	AFW
Euro Stoxx 50	1.89	1.98	1.89
FTSE MIB	1.73	1.82	1.68
FTSE 100	1.68	1.83	1.79

TABLE 2: Percentage Risk for Each Algorithm on the Three Datasets

Dataset	FW	PFW	AFW
Euro Stoxx 50	0.34	0.35	0.34
FTSE MIB	0.40	0.41	0.38
FTSE 100	0.47	0.49	0.49

TABLE 3: Weekly Percentage Return Time for Each Algorithm on the Three Datasets

The Away-step Frank-Wolfe algorithm generally achieves the lowest percentage risk across all datasets, indicating a more stable portfolio. However, the weekly percentage return is quite similar across all three algorithms, with some variations depending on the dataset. This shows that while AFW is effective in minimizing risk, all three algorithms perform similarly in terms of return, making them reliable options for portfolio optimization.

*5) **Efficient Frontier**:* Finally, we will plot the efficient frontier for each dataset with the three algorithms. The efficient frontier shows the set of optimal portfolios that offer the highest expected return for a given level of risk.

We plot the efficient frontier for each dataset using the three algorithms. This visualization shows how the different algorithms perform in terms of portfolio optimization.

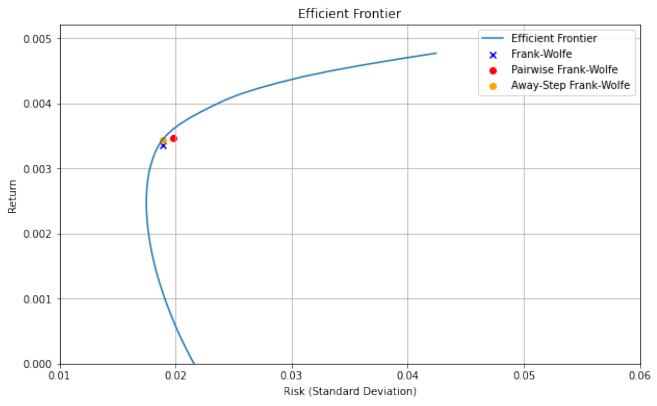

Fig. 7: Euro Stoxx 50

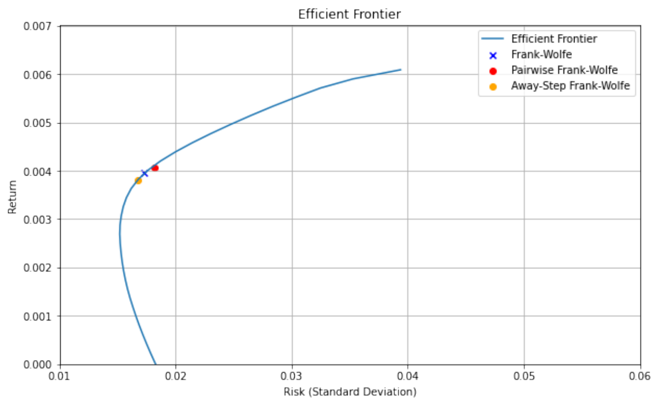

Fig. 8: Ftse MIB

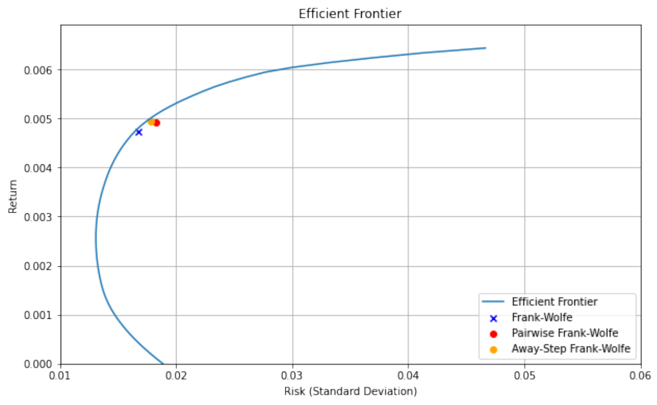

Fig. 9: Ftse 100

The Frank-Wolfe algorithm generally places the optimized portfolio closest to the efficient frontier across all datasets, indicating the best performance in terms of achieving the optimal risk-return trade-off. The Pairwise Frank-Wolfe and Away-step Frank-Wolfe algorithms also perform well but are slightly less efficient in some cases.

VII. Concluding Remarks

In this paper, we explored the application of the Frank-Wolfe algorithm and its variants, Pairwise Frank-Wolfe and Away-Step Frank-Wolfe, to the problem of portfolio optimization within the framework of Markowitz portfolio theory. Our goal was to assess the effectiveness of these algorithms in optimizing financial portfolios based on metrics such as loss, risk, return, duality gap, and execution time. The results demonstrated that

the standard Frank-Wolfe algorithm consistently outperformed the other variants in terms of minimizing loss and reducing the duality gap across all datasets. This algorithm also showed robust convergence properties, making it a reliable choice for practical applications in portfolio management. However, the Away-Step Frank-Wolfe algorithm exhibited the fastest execution times, indicating its computational efficiency, while still providing competitive performance in risk minimization and return optimization. The Pairwise Frank-Wolfe algorithm, though not as efficient as the standard Frank-Wolfe, still offered significant improvements over the Away-Step Frank-Wolfe in terms of loss minimization and duality gap reduction. All three algorithms demonstrated their utility in constructing optimized portfolios that align closely with the efficient frontier, ensuring an optimal balance between risk and return.

Additionally, we simulated investments using the weights derived from the algorithms and found that our optimized portfolios outperformed the benchmark indices over the same period. This further validates the practical benefits of these optimization techniques in real-world scenarios.

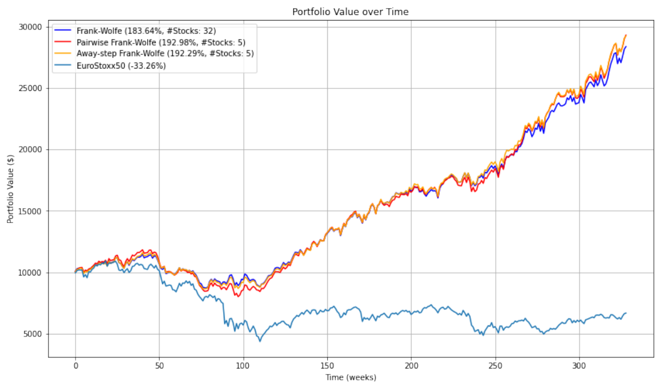

Fig. 10: Euro Stoxx 50

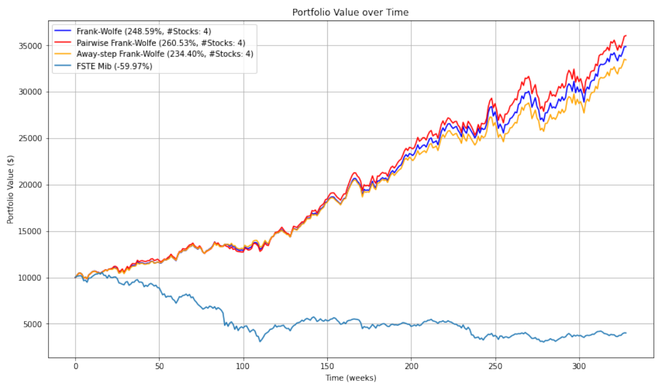

Fig. 11: Ftse MIB

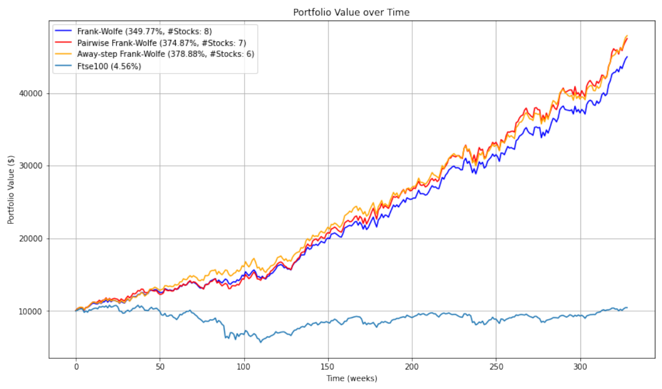

Fig. 12: Ftse 100

Overall, our analysis confirms the viability of the Frank-Wolfe algorithm and its variants as powerful tools for portfolio optimization. Future work could explore further enhancements and adaptations of these algorithms to handle more complex financial models and larger datasets, potentially incorporating advanced machine learning techniques to improve their performance and scalability. By leveraging these optimization techniques, financial analysts and portfolio managers can achieve better risk management and higher returns, ultimately leading to more effective and informed investment strategies.

VIII. References

Martin Jaggi. *Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization*. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013.

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, Patrick Pletscher. *Block-Coordinate Frank-Wolfe Optimization for Structural SVMs*. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013.

Julien Mairal. *Optimization with First-Order Surrogate Functions*. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013.