Università di Padova
Dipartmento di Matematica

FIRST ORDER OPTIMIZATION METHODS: a Comparative Analysis of the Projected Gradient Method and the Frank-Wolfe Algorithm on Portfolio Optimization

Optimization for Data Science

Contributors: Lazzari Tommaso
Ludergnani Brenno
Movila Dumitru
Safa Nasser

Date: September 2025

Table of Contents

1 Introduction

Portfolio Optimization is a typical optimization problem well suited to test constrained first-order optimization methods. This paper evaluates the efficiency of the Frank-Wolfe Algorithm (and its variants) and the Projected Gradient Method on the Markowitz Portfolio Optimization problem. The document presents a general overview and theoretical analysis of the Frank-Wolfe Algorithm, its variants, the Pairwise Frank-Wolfe and the Away-Step Frank-Wolfe, and the Projected Gradient Method. The study aims to compare the performances of these algorithms through empirical studies using four different datasets. The paper emphasizes the ability of Frank-Wolfe algorithms to effectively handle the structured constraints appearing in machine learning applications, its good scalability and the crucial property that maintains its iterates as a convex combination of only few atoms, enabling sparse and low- ranking solutions. However, the classical Frank-Wolfe convergence rate is known to be slow (sublinear) when the solution lies at the boundary. Thus, the paper also explores the Pairwise and Away-Step variants, highlighted for their enhanced convergence properties and robustness, making them valuable extensions of the classical Frank-Wolfe algorithm.

2 Markowitz Portfolio Optimization Problem

Markowitz Portfolio Optimization Problem, introduced by Harry Markowitz in 1952 [6], formalizes the trade-off between risk and return in portfolio selection by modelling risk as the variance of portfolio returns and expected performance as the mean return. The main insight is that diversification can reduce overall portfolio risk without necessarily lowering expected return.

2.1 Problem Formulation

Assuming we have n available assets, we call x_i the amount of money invested on the i-th asset during the considered period of time and r_i the return on the i-th asset. We can easily set the following constraints:

- *Non negativity for the variables* (i.e. $x_i \geq 0 \forall i \in \{1, \ldots, n\}$): meaning that short selling is not allowed;
- *Budget constraint* (i.e. $\sum x_i = B$): the total amount of money invested cannot exceed the budget B (B can simply be set to 1).

The classic Portfolio Optimization Problem described by Markowitz can be expressed as a quadratic programming problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & x^T \Sigma x - \eta \hat{r}^T x \\ x_i \geq 0 \forall i \\ \textstyle\sum_{i=1}^n x_i = B \end{aligned}$$

3 Frank-Wolfe Algorithm

Consider a general constrained convex optimization problem of the form

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

we assume that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and continuously differentiable, and that the domain C is a compact, convex subset of any vector space. For such optimization problems one of the simplest and earliest known iterative optimizer is the Conditional Gradient Method also called Classical Frank-Wolfe algorithm. It was first introduced by Marguerite Frank and Philip Wolfe in 1956 [5]. The algorithm iteratively leverages a linear approximation of the objective function to generate a sequence of feasible iterates, avoiding costly projection while mantaining convergence guarantees. The "projection-free" peculiarity has renewed interest in the method, especially in large-scale data science applications where projection operations can be computationally expensive.

3.1 Algorithm Description

At each iteration k, the algorithm solves the subproblem:

$$s^{(k)} = \arg\min_{s \in C} \langle \nabla f(x^{(k)}), s \rangle$$

finding the new vertex s that minimizes the inner product with the gradient. The new iterate is computed as:

$$x^{(k+1)} = (1 - \gamma_k)\, x^{(k)} + \gamma_k\, s^{(k)}$$

This is a convex combination of the two points, since the stepsize $\gamma_k \in [0, 1]$. The latter can be set adopting different strategies, such as the exact line search or predefined schemes such as the diminishing step strategy. The strcuture of the Classical Frank-Wolfe algorithm is the following:

1. **Initialization**: choose the starting point $x^{(0)} \in C$.
2. **FW direction**: solve the linear sub-problem to identify the FW atom $s^{(k)}$ and the descent direction $d^{(k)} = x^{(k)} - s^{(k)}$.
3. **Step-Size Determination**: find the step-size γ_k suitable for the iteration k.
4. **Iterate Update**
5. **Termination**: check the convergence criteria and terminate if satisfied; otherwise, repeat from step 2.

Algorithm 1 Classical Frank-Wolfe Algorithm

1: **Input:** Convex function f, compact convex domain C.
2: **Initialize:** $x^{(0)} \in C$.
3: **for** $k = 1, 2, \ldots$ **do**
4: $\quad s^{(k)} = \arg\min_{s \in C} \langle \nabla f(x^{(k)}), s \rangle$
5: $\quad$ Choose step size γ_k.
6: $\quad x^{(k+1)} = (1 - \gamma_k)x^{(k)} + \gamma_k s^{(k)}$
7: **end for**

3.2 Unit Simplex Case

In the context of the portfolio optimization problem, we consider the special case in which the domain C is represented by the unit simplex $\Delta_{n-1} \subset \mathbb{R}^n$. In this case the linear sub-problem is computationally cheap, since$\hat{x}_k = e_{i_k}$ where $i_k = \arg\min_i \nabla_i f(x^{(k)})$.

$$C = \{x \in \mathbb{R}^n : e^T x = 1, x \geq 0\}$$

Algorithm 2 Classical Frank-Wolfe Algorithm. Unit simplex case.

1: **Input:** Convex function f, compact convex domain C.
2: **Initialize:** $x^{(0)} \in C$.
3: **for** $i = 1 \ldots$ **do**
4: $\quad$ Compute the linear subproblem: $s^{(k)} = e_{i_k} = \arg\min_{s \in C} \nabla f(x^{(k)})$
5: $\quad$ Choose the step size γ_k.
6: $\quad$ Update the solution $x^{(k+1)} = (1-\gamma_k)x^{(k)} + \gamma_k s^{(k)}$
7: **end for**

3.3 Frank-Wolfe Convergence Analysis

The Frank-Wolfe method is guaranteed to converge under general convexity assumptions, achieving a rate of $O(1/k)$ when the objective function is smooth and convex. This baseline rate can be enhanced under stronger assumptions, such as strong convexity or when the optimum lies in the relative interior of the feasible region. In fact, for strongly convex problems, the method can even attain linear convergence given suitable conditions. The algorithm is also robust in the presence of inexact oracles, where the linear subproblems are solved only approximately. Even in this setting, convergence guarantees are preserved, which makes the method practical in applications where computing exact solutions is too costly. A central strength of the FW procedure lies in the fact that it preserves feasibility at every step. This feature is especially advantageous in structured optimization problems, where enforcing feasibility is often difficult. Another important property is its affine invariance: the algorithm's performance does not depend on affine transformations of the feasible domain. This ensures that scaling or translating the problem has no negative effect on its efficiency. The convergence of the FW algorithm is formally established in **Theorems 1 and 2** where the rate depends on the curvature constant C_f and on the precision δ used in solving the linear sub-problem. The method guarantees both primal and primal-dual convergence, which underlines its reliability for constrained convex optimization. Specifically, the primal convergence theorem bounds the primal error, ensuring its decrease at a rate of $O(1/k)$, while the primal-dual convergence theorem provides an equivalent rate for the duality gap. These results are particularly significant in scenarios where exact projections are computationally demanding or impractical, highlighting the FW algorithm's usefulness in large-scale optimization tasks.

Theorem 1. ***(Primal Convergence*** *For each $k \geq 1$, the iterate $x^{(k)}$ of Frank-Wolfe algorithm and its variants, satisfies:*

$$f(x^{(k)}) - f(x^*) \leq \frac{2C_f}{k+2}(1+\delta)$$

where $x^* \in C$ is an optimal point, $x^{(k)}$ is the current solution, $\delta \leq 0$ is the accuracy to which the internal linear subproblems are solved and C_f is the curvature constant of the objective function f.

Duality Gap For any constraint convex optimization problem of the form:

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

and a feasible point $x \in C$ we define the following simple surrogate of duality gap:

$$g(x) := \max_{s \in C} \langle x - s, \nabla f(x) \rangle$$

Convexity of f implies that the linearization $f(x) + \langle s - x, \nabla f(x) \rangle$ always lieas below the graphic of the function f. This immediatly gives a crucial property of the duality gap, as being the certificate of the current approximation quality, i.e. $g(x) \geq f(x) - f(x^*)$.

Theorem 2. ***(Primal-Dual Convergence)*** *If Frank-Wolfe algorithm or any of its variants is run for $K \geq 2$ iterations, then the algorithm has an iterate $x^{(\hat{k})}$ with $1 \leq \hat{k} \leq K$ with duality gap bounded by:*

$$g_{\hat{k}} := g(x^{(\hat{k})}) \leq \frac{2\beta C_f}{K+2}(1+\delta)$$

3.4 Frank-Wolfe variants

To improve its efficiency and broaden its applicability to diverse problem settings, many extension of the FW method have been introduced:

- **Diminishing Step Variant**: A predefined decreasing sequence step size choosing rule, which usually is of the form

$$\gamma_k = \frac{2}{k+2}$$

This schedule ensures that $\gamma_k \to 0$ as $k \to \infty$, while still being large enough in early iterations.

- **Line-Search Variant**: Performs a line search to find the optimal step size that minimizes $f(x^{(k)} + \gamma_k(s - x^{(k)}))$. So that:

$$\gamma_k = \arg\min_{\gamma \in [0,1]} f(x^{(k)} + \gamma(s - x^{(k)}))$$

  Selecting the most suitable γ_k at every iteration ensures faster convergence and more efficient steps.

- **Away-Step Frank-Wolfe (AFW)**: This variant introduces the "away-step", thus the name, which allows the method to step away from an active vertex of the feasible set and mitigate the zig-zagging effect that often arises near the boundary of the feasible region. By doing so, it improves convergence behaviour, particulary in problems where the feasible domain contains sharp vertices.

- **PairWise Frank-Wolfe (PFW)**: This variant improves descent directions by combining pairs of feasible points, which leads to faster convergence through more refined updates. Instead of relying solely on the current iterate and a new vertex, it performs pairwise moves between the present point and previosuly selected vertices, offering an effective strategy to reduce the zig-zagging phenomenon.

4 Away-Step Frank-Wolfe

The Away-Step Frank-Wolfe (AFW) is a variant of the FW algorithm aiming to enhance convergence rate in optimization problems over polytopes. It was specifically designed to address the zig-zagging behaviour of the classical FW when the optimal solution of the problem is close to the boundary of the feasible set. In the AFW the algorithm keeps track of the visited vertices by storing them in the active set $S^{(k)}$, which represents the subset of vertices we can represent the current solution $x^{(k)}$ as a convex combination of. In the AFW algorithm not only we compute the classical descent direction using the output of the LMO to produce $d_{FW}^{(k)} = s^{(k)} - x^{(k)}$, but we also compute the away step direction $d_A^{(k)} = x^{(k)} - v^{(k)}$, where $v^{(k)}$ is the "worst vertex" belonging to the active set, obtained as:

$$v_A^{(k)} = argmax_{v \in S^{(k)}} \langle \nabla f(x^{(k)}), v \rangle$$

The AFW can decide either to use the classical descent direction $d_{FW}^{(k)}$ or to use the away-step direction $d_A^{(k)}$, based on which direction would bring the best improvement in the objective function with respect to $\nabla f(x^{(k)})$. We observe that during each classical FW step, a vertex may or may not be added to the active set, this means $|S^{(k+1)}| \geq |S^{(k)}|$. While during each away-step the size of the active set can either decrease or remain the same, $|S^{(k+1)}| \leq |S^{(k)}|$. Assuming that $d_A^{(k)}$ is chosen at step k, than the following iterate is defined as:

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \alpha_k d_A^{(k)} \\ &= x^{(k)} + \alpha_k (x^{(k)} - v^{(k)}) \\ &= (1 + \alpha_k) \sum_{u \in S^{(k)}} \gamma_u^{(k)} u - \alpha_k v^{(k)} \end{aligned}$$

where α_k is the step size and every $\gamma_u^{(k)}$ is the coefficient of vertex $u \in S^{(k)}$ in the convex combination of $x^{(k)}$. Since $v^{(k)} \in S^{(k)}$, this vertex is included in the convex combination describing the current solution. Then if

$$(1 + \alpha_k)\gamma_u^{(k)} - \alpha_k = 0 \iff \alpha_k = \frac{\gamma_u^{(k)}}{1 - \gamma_u^{(k)}}$$

the vertex $v^{(k)}$ is eliminated from the description of $x^{(k)}$, hence from the active set. Thus we should obtain $|S^{(k+1)}| < |S^{(k)}|$. We would call this a drop-step. Otherwise, the weight of $v^{(k)}$ in the convex combination would be modified by the factor $-\alpha_k$ while the active set remains the same $|S^{(k+1)}| = |S^{(k)}|$.

4.1 Algorithm Description

The AFW algorithm is characterized by the following steps:

1. **Initialization**: Start with an initial point $x^{(0)} \in C$ and initialize the active set $S^{(0)} = \{x^{(0)}\}$.

2. **FW Direction**: Solve the linear subproblem to find the FW direction $d_{FW}^{(k)}$.

3. **Away-Step Direction**: Solve the linear subproblem to find the away-step vertex $v^{(k)}$ and define $d_A^{(k)} = x^{(k)} - v^{(k)}$.

4. **Direction choice**: choose the direction leading to the best improvement between $d_{FW}^{(k)}$ and $d_A^{(k)}$.

5. **Step-Size Determination**: Compute the step-size γ_k.

6. **Iterate Update**

7. **Termination**: check the convergence criteria and stop if satisfied; otherwise repeat from step 2.

Algorithm 3 Away-Step Frank-Wolfe Algorithm.

Input: Convex function f, compact convex domain C.
Initialize: $x^{(0)} \in C$ and $S^{(0)} = \{x^{(0)}\}$.
for $i = 1 \dots$ **do**
  Compute the linear subproblem: $\hat{x}_{FW}^{(k)} = \arg\min_{s \in C} \langle \nabla f(x^{(k)}), s \rangle$
  if $\hat{x}_{FW}^{(k)}$ satisfies some conditions **then**
    Stop.
  else
    Set $\hat{x}_A^{(k)} = \arg\max_{x \in S^{(k)}} \nabla f(x^{(k)})^t (x - x^{(k)})$
    Set $d_{FW}^{(k)} = \hat{x}_{FW}^{(k)} - x^{(k)}$ and $d_A^{(k)} = x^{(k)} - \hat{x}_A^{(k)}$
    if $\nabla f(x^{(k)})^t d_{FW}^{(k)} \leq \nabla f(x^{(k)})^t d_A^{(k)}$ **then**
      set $d^{(k)} = d_{FW}^{(k)}$ and $\bar{\alpha} = 1$
    else
      set $d^{(k)} = d_A^{(k)}$ and $\bar{\alpha} = \max\{\beta \mid x^{(k)} + \beta d_A^{(k)} \in C\}$
    end if
    set $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$ with $\alpha_k \in (0, \bar{\alpha})$
    Compute $S^{(k+1)}$ set of the current used vertices.
  end if
end for

4.2 Away-Step Frank-Wolfe Convergence Analysis

The AFW algorithm enjoys stronger convergence guarantees compared to the classical FW method, particularly when the solution lieas on the boundary of the feasible set. The introduction of away-steps enables the method to achieve linear convergence under suitable conditions. Specifically, when the objective function is strongly convex and the feasible domain is a polytope, AFW ensures that the primal suboptimality decreases at a geometric rate. This behaviour is explained through the notion of a geometric strong convexity constant, which couples the condition number of the objective function with a polytope-dependent constant capturing the geomtry of the feasible set. Formally, under strong convexity one obtains a bound of the form:

$$f(x^{(k)}) - f(x^*) \leq (f(x^{(0)}) - f(x^*)) \exp(-\rho k)$$

with $\rho > 0$, k number of iterations and x^* the optimal point. Particularly ρ is determined by the curvature of the function and the characteristics of the feasible set. A crucial element of this analysis is the subdivision between "good steps" that guarantee a sufficient decrease in the objective function and "bad steps" represented by drop-steps in which we just eliminate a "bad vertex" from the current solution description. Since the number of "bad steps" performed untile step k is upper bounded by $k/2$, progresses are consistently made and AFW still recovers $O(1/k)$ convergence rate. These results highlight AFW as a robust and theoretically wel-grounded variant, especially effective for structured optimization tasks where the solution lies on the boundary of the feasible region.

5 PairWise Frank-Wolfe

The Pairwise Frank-Wolfe (PFW) algorithm is a refinement of the classical Frank-Wolfe method designed to overcome some of its inherent drawbacks. The PFW-variant improves convergence by incorporating more sophisticated update steps, thereby increasing both the efficiency and the practical applicability of FW. PFW exploits the geometry of the feasible set by working with convex combinations of the vertices involved. Unlike the classical approach, which only moves towards a new vertex, PFW considers both the current iterate and atoms from the active set, striking a balance between exploring new directions and correcting past choices. This strategy not only accelerates convergence but also promotes sparsity in the active set, which is particularly beneficial for problems where sparse solutions are desired.

5.1 Algorithm Description

The PFW method extends the classical FW algorithm by introducing pairwise moves. Instead of advancing solely toward the current linear minimizer, it also allows movement away from vertices already included in the active set. This mechanism reduces the zig-zagging effect and leads to faster convergence. The Pairwise Frank-Wolfe Algorithm is characterized by the following steps:

1. **Initialization**: Start with an initial point $x^{(0)} \in C$ and initialize the active set $S^{(0)} = \{x^{(0)}\}$.

2. **FW Direction**: Solve the linear subproblem to find the FW direction $d_{FW}^{(k)}$.

3. **Away-Step Direction**: Solve the linear subproblem to find the away-step vertex $v^{(k)}$ and define $d_A^{(k)} = x^{(k)} - v^{(k)}$.

4. **PairWise Direction**: Compute the pairwise direction as $d_{PFW}^{(k)} = s^{(k)} - v^{(k)}$.

5. **Step-Size Determination**: Compute the step-size γ_k.

6. **Iterate Update**

7. **Termination**: check the convergence criteria and stop if satisfied; otherwise repeat from step 2.

Algorithm 4 Pairwise Frank-Wolfe Algorithm.

Input: Convex function f, compact convex domain C.
Initialize: $x^{(0)} \in C$ and $S = \{x^{(0)}\}$.
for $i = 1 \dots$ **do**
    Solve for $s^{(k)}$ the linear subproblem:

$$s^{(k)} = \arg\min_{s \in C} \langle \nabla f(x^{(k)}), s \rangle$$

    Identify $v^{(k)} \in S^{(k)}$ such that:

$$v^{(k)} = \arg\max_{v \in S^{(k)}} \langle \nabla f(x^{(k)}), v \rangle$$

    Compute $d^{(k)} = s^{(k)} - v^{(k)}$.
    Choose γ_k that minimizes $f(x^{(k)} + \gamma_k d^{(k)})$.
    Update $x^{(k)}$ and $S^{(k)}$.
end for

5.2 PairWise Frank-Wolfe Convergence Analysis

The PFWmethod offers improved convergence behavior compared to the classical FW algorithm. By using pairwise steps, it can move from suboptimal vertices more effectively, thereby advancing towards the optimal solution in fewer iterations. Under general convexity assumptions, the PFW algorithm maintains the O(1/k) convergence rate of the original FW method, but in practice it often converges more quickly thanks to more effective direction updates. In the case of strongly convex functions, the PFW algorithm achieves linear convergence, which substantially enhances its efficiency relative to the classical approach. This faster rate is particularly valuable in scenarios that demand rapid convergence, such as real-time applications. Moreover, the PFW method is resilient when working with inexact oracles: it preserves theoretical convergence guarantees even if the linear subproblems are solved only approximately. This property is especially important in settings with limited computational resources, where exact solutions are too costly to obtain.

6 Projected Gradient

The Projected Gradient Method (PG) is a fundamental tool in convex optimization, particularly effective for solving constrained problems. The well-known Gradient Method cannot be applied to constrained optimization problems, in fact, its iterates might not belong to the feasible set. Therefore, PG method at each iteration, performs a gradient descent update followed by a projection onto the feasible region, guaranteeing that all iterates satisfy the imposed constraints.

6.1 Algorithm Description

The PG approach is especially useful when the search space is confined to a feasible region $C \subseteq \mathbb{R}^n$. Its objective is to minimize a differentiable function $f(x)$ over the convex set C. To achieve this, the algorithm repeatedly takes a gradient descent step and then projects the result back onto C, ensuring that the constraints are satisfied. The iterative update rule is given by:

$$x^{(k+1)} = \rho_C(x^{(k)} - \gamma_k \nabla f(x^{(k)}))$$

where ρ_C denotes the projection operator onto C and γ_k is the step-size. The projection guarantees that each new iterate $x^{(k+1)}$ remains within the feasible region, which is essential to maintain the validity of the solution under the constraints of the problem.

Algorithm 5 Projected Gradient Algorithm.

Input: Convex function f, compact convex domain C.
Initialize: $x^{(0)} \in C$.
for $i = 1 \dots$ **do**
    Compute the gradient descent step:

$$y^{(k+1)} = x^{(k)} - \gamma_k \nabla f(x^{(k)})$$

    Project onto the feasible set:

$$x^{(k+1)} = \rho_C(y^{(k+1)})$$

    Check the stopping criteria.
end for

6.2 Projected Gradient Convergence Analysis

The PG method has a strong theoretical foundation ensuring its convergence. For convex objective functions, its convergence rate is comparable to standard gradient descent, with the rate primarily determined by assumptions such as strong convexity and Lipschitz continuity of the gradient. Specifically, for a function that is L-smooth and μ-strongly convex, the PG method achieves a linear rate of convergence. This means that the distance from the current iterate to the optimal solution decreases at a geometric rate with each iteration. This can be expressed formally by the following inequality:

$$||x^{(k+1)} - x^*|| \leq \rho ||x^{(k)} - x^*||$$

where x^* is the optimal solution and $\rho \in (0, 1)$ is the convergence factor. Observe that ρ is determined by the properties of the function.

7 Data

The different algorithms have been tested on four different real world portfo lios, downloaded from Real-

world datasets for portfolio selection and solutions of some stochastic dominance portfolio models- ScienceDirect [4]. The datasets contain weekly return values of the FTSE100, Dow Jones, NASDAQ100 and SP500 indexes, adjusted for dividends and for stock splits, which are cleaned from errors as much as possible.

8 Results

The empirical analysis confirms that all tested algorithms successfully converged to nearly identical optimal portfolio solutions across the four datasets, with negligible differences in final objective values and duality gaps. However, the computational efficiency varied significantly. The classical Frank-Wolfe algorithm with line search exhibited solid reliability but required substantially more iterations than its variants. In contrast, the Pairwise and Away-Step Frank-Wolfe methods consistently achieved convergence within a handful of iterations and with minimal runtime, highlighting their practical advantage in large-scale applications. The Projected Gradient method, while theoretically strong, demonstrated slower convergence and higher computational costs due to the projection step, particularly evident in the FTSE100 and SP500 datasets. Overall, the results illustrate the superior scalability and efficiency of Frank-Wolfe variants, with Pairwise and Away-Step methods standing out as the most effective approaches for portfolio optimization problems constrained to the simplex.

References

[1] I. M. Bromze, F. Rinaldi, and D. Zeffiro. Frank–Wolfe and friends: A journey into projection-free first-order optimization methods. *Optimization Letters*, 2021.

[2] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank Wolfe optimization variants. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[3] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[4] R. Bruni, F. Cesarone, A. Scozzari, and F. Tardella. Real-world datasets for portfolio selection and solutions of some stochastic dominance portfolio models. *Omega*, 2015.

[5] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[6] H. Markowitz. Portfolio Selection. *The Journal of Finance*, 7(1):77–91, 1952.

9 Appendix

9.1 Tables of Results

In the following tables are displayed the observed metrics of the performances of the five different tested algorithms on the four real world financial datasets. In each table, for each algorithm we report

- the final objective function value;
- the final duality gap value;
- the support size of the solution;
- the CPU runtime required to converge (in seconds)
- the number of iterations required to converge (note that 1000 is the maximum number of iterations set for the experiments)

With **(ls)** we indicate that the line-search method has been implemented, while **(dim)** refers to the diminishing step size strategy

Table 1: Algorithms' Performace Results on FTSE100 dataset

Algorithm	Final Objective	Final Gap	Support Size	Runtime(s)	Iterations
Frank-Wolfe (ls)	-0.005234	0.000012	5	0.034966	397
Pairwise FW (ls)	-0.005240	0.000025	4	0.001667	10
Away-Step FW (ls)	-0.005240	0.000012	4	0.001174	11
Projected Gradient	-0.005240	0.000012	4	1.184409	1000
Frank-Wolfe (dim)	-0.005240	0.000015	4	0.007068	185

Table 2: Algorithms' Performace Results on NASDAQ100 dataset

Algorithm	Final Objective	Final Gap	Support Size	Runtime(s)	Iterations
Frank-Wolfe (ls)	-0.007463	0.000017	6	0.022593	402
Pairwise FW (ls)	-0.007470	0.000017	5	0.001398	13
Away-Step FW (ls)	-0.007470	0.000010	5	0.000954	12
Projected Gradient	-0.007470	0.000010	5	0.541565	682
Frank-Wolfe (dim)	-0.007470	0.000037	5	0.008402	238

Table 3: Algorithms' Performance Results on DOWJONES dataset

Algorithm	Final Objective	Final Gap	Support Size	Runtime(s)	Iterations
Frank-Wolfe (ls)	-0.004179	0.000011	5	0.013576	205
Pairwise FW (ls)	-0.004183	0.000021	4	0.001297	12
Away-Step FW (ls)	-0.004183	0.000017	4	0.000692	8
Projected Gradient	-0.004183	0.000024	4	0.739824	1000
Frank-Wolfe (dim)	-0.004183	0.000072	4	0.002197	73

Table 4: Algorithms' Performance Results on SP500 dataset

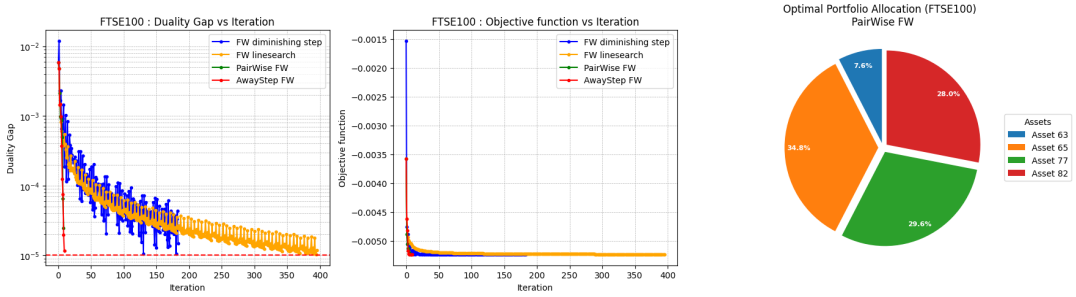
Algorithm	Final Objective	Final Gap	Support Size	Runtime(s)	Iterations
Frank-Wolfe (ls)	-0.007487	0.000016	6	0.160002	402
Pairwise FW (ls)	-0.007494	0.000045	5	0.004553	10
Away-Step FW (ls)	-0.007494	0.000059	5	0.004416	11
Projected Gradient	-0.007494	0.000010	5	2.366967	686
Frank-Wolfe (dim)	-0.007494	0.000043	5	0.068565	233

9.2 Graphical Representation

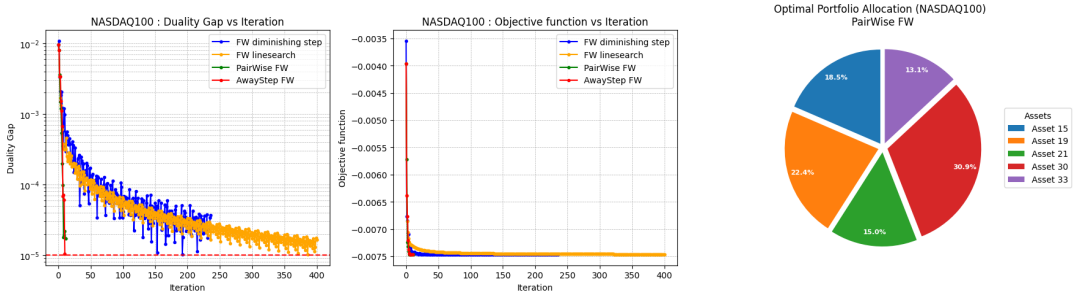
In the following graphs, for each one of the four different datasets we show:

- A plot of the duality gap for the tested algorithms, where the y-axis is on a logarithmic scale and represents the value of the duality gap, while on the x-axis we find the iteration number;
- A convergence plot of the objective function. The function's value is reported on the y-axis, while the iteration number is on the x-axis;
- The obtained composition of the optimal portfolio with the PWF algorithm

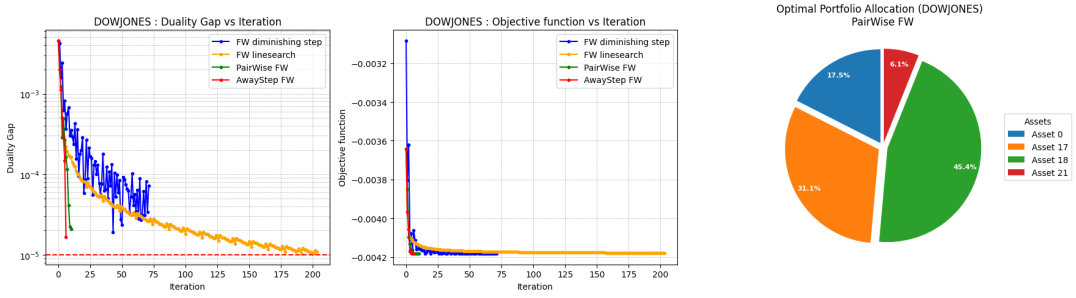
Graphs 1A, 1B, 1C: Algorithms' Convergence Analysis on FTSE100



Graphs 2A, 2B, 2C: Algorithms' Convergence Analysis on NASDAQ100



Graphs 3A, 3B, 3C: Algorithms' Convergence Analysis on DOWJONES100



Graphs 4A, 4B, 4C: Algorithms' Convergence Analysis on SP500