University of Padova

Department of Data Science

ODS PROJECT:
Optimization Methods for Clustering
(Group 25)

Authors:

Alessandro Giuffrè [1], Vishwa Mittar[2], Jaime Candau Otero[3], Alessandro Di Frenna [4]

[1] Matricola: 2090827 (alessandro.giuffre@studenti.unipd.it)
[2] Matricola: 2141871 (vishwa.mittar@studenti.unipd.it)
[3] Matricola: 2132765 (jaime.candauotero@studenti.unipd.it)
[4] Matricola: 2160704 (alessandro.difrenna@studenti.unipd.it)

Contents

1 Introduction

The Frank-Wolfe algorithm, also known as the conditional gradient method, is a classical first-order optimization algorithm for solving constrained smooth convex optimization problems of the form:

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

where f is a continuously differentiable convex function and $\mathcal{C}$ is a compact convex set.

What distinguishes Frank-Wolfe from other optimization methods is its **projection-free** nature: instead of projecting onto the feasible set $\mathcal{C}$, it iteratively solves a much simpler **linear minimization problem** over $\mathcal{C}$:

$$s_t = \arg\min_{s \in \mathcal{C}} \langle \nabla f(x_t), s \rangle,$$

and then updates the current iterate as a convex combination:

$$x_{t+1} = (1 - \gamma_t) x_t + \gamma_t s_t,$$

where $\gamma_t \in [0, 1]$ is a step-size.

This approach makes the algorithm especially appealing when projection onto $\mathcal{C}$ is computationally expensive or complicated, but linear minimization over $\mathcal{C}$ is easy. Because of this, Frank-Wolfe has been widely used in machine learning, signal processing, and computational geometry, especially when working with structured constraint sets like the simplex or the nuclear norm ball.

In addition to its practical simplicity, the Frank-Wolfe algorithm also guarantees convergence at a rate of $O(1/t)$ for convex problems, which is optimal among projection-free first-order methods. Over the years, several variants have been proposed to accelerate convergence or handle non-convex objectives.

2 Frank-Wolf: Mathematical Formulation and Algorithm

2.1 Problem Setting

The Frank-Wolfe algorithm solves constrained optimization problems of the form:

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex and differentiable function, and $\mathcal{D} \subset \mathbb{R}^n$ is a compact convex feasible set.

2.2 Algorithm Description

At each iteration, the algorithm linearizes the function f at the current iterate x_t and moves towards a minimizer of the linear approximation:

1. Compute the gradient: $\nabla f(x_t)$.
2. Solve the Linear Minimization Oracle (LMO):
$$s_t := \arg\min_{s \in \mathcal{D}} \langle s, \nabla f(x_t) \rangle.$$
3. Update the solution:
$$x_{t+1} := (1 - \gamma_t) x_t + \gamma_t s_t,$$
where $\gamma_t \in [0, 1]$ is the step-size.

2.3 Pseudocode (Algorithm 2 from FW Survey)

Algorithm 1 Frank-Wolfe Algorithm (from FW Survey)

Input: Convex set $\mathcal{D}$, differentiable function f, initial point $x_0 \in \mathcal{D}$, tolerance ε
for $t = 0, 1, 2, \ldots$ **do**
    Compute gradient $\nabla f(x_t)$
    Find FW vertex: $s_t \in \arg\min_{s \in \mathcal{D}} \langle s, \nabla f(x_t) \rangle$
    Compute FW gap: $g_t = \langle x_t - s_t, \nabla f(x_t) \rangle$
    if $g_t \leq \varepsilon$ **then**
        stop and return x_t
    end if
    Choose step-size $\gamma_t \in [0, 1]$ (e.g., exact line search or fixed)
    Update: $x_{t+1} := (1 - \gamma_t) x_t + \gamma_t s_t$
end for

3 Variants of the Frank-Wolfe Algorithm

3.1 Motivation

While the classical Frank-Wolfe (FW) method is attractive due to its projection-free nature, it can suffer from slow convergence, especially when the optimal solution lies on the boundary of the feasible region. This motivated the development of several enhanced variants, all aiming to accelerate convergence and improve practical performance.

3.2 Overview of Main Variants

1. Away-Step Frank-Wolfe (AFW) This variant allows moving away from current atoms (points in the active set) that contribute poorly to descent. It helps mitigate the "zig-zagging" behavior when the solution lies on the boundary. The algorithm introduces away directions that remove weight from bad atoms. Under certain conditions, AFW achieves linear convergence.

2. Pairwise Frank-Wolfe (PFW) Instead of simply moving toward a new FW atom, PFW shifts weight from an existing atom v_t to a new one s_t. This leads to sparse updates and is particularly effective when the solution is sparse, as in many machine learning applications.

3. Fully-Corrective Frank-Wolfe (FCFW) After each FW step, this variant re-optimizes over the convex hull of all previously selected atoms (the correction set). Though computationally expensive, it ensures maximal progress at each iteration and is related to Matching Pursuit algorithms.

4. Wolfe's Min-Norm Point (MNP) A generalization of FCFW, MNP uses a correction procedure that maintains affine independence and solves internal projection steps to maximize descent. It is well suited for polytope distance and norm-based problems.

3.3 Comparison Table

Variant	Removes Mass?	Sparse Updates	Linear Convergence	Efficient on Sparse Problems
Frank-Wolfe (FW)	No	Yes	No	Yes
Away-Step (AFW)	Yes	Yes	Yes	Yes
Pairwise (PFW)	Yes	Yes	Yes	Very Good
Fully-Corrective (FCFW)	Yes	No	Strong	No
Min-Norm Point (MNP)	Yes	No	Strong	No

Table 1: Summary of main Frank-Wolfe algorithm variants.

3.4 Duality Gap and Line Search

In our project, we adopted the Frank-Wolfe gap (also referred to as the duality gap) as a natural and practical stopping criterion. For a given iterate x_t, the gap is computed as:

$$g_t := \langle x_t - s_t, \nabla f(x_t) \rangle,$$

where s_t is the solution of the linear subproblem at iteration t. This quantity provides an upper bound on the distance from optimality:

$$g_t \geq f(x_t) - f(x^*),$$

and thus certifies how close the current solution is to optimality.

In our implementation, we used the Frank-Wolfe gap to determine convergence: once the gap falls below a tolerance threshold ε, the algorithm stops. This made the method both efficient and easy to monitor during testing.

Moreover, instead of using a fixed step-size, we implemented a **line search strategy** at each iteration. Specifically, we selected the value of $\gamma_t \in [0, 1]$ that minimizes the objective function along the line between x_t and s_t:

$$\gamma_t := \arg\min_{\gamma \in [0,1]} f\left(x_t + \gamma(s_t - x_t)\right).$$

This approach allowed us to improve convergence in practice and adapt the step size dynamically across different problem instances.

4 The Maximum Clique Problem and its Continuous Formulation

4.1 Problem Description

In our project, the optimization objective is to find cliques in undirected graphs. Formally, given a graph $G = (V, E)$, a *clique* is a subset of nodes $C \subseteq V$ such that all pairs of nodes in C are connected by an edge. A clique is *maximal* if it cannot be extended by adding any other vertex, and it is *maximum* if it has the largest possible number of vertices.

The classical **Maximum Clique Problem (MCP)** can be expressed as:

$$\text{maximize } |C| \quad \text{such that } C \text{ is a clique in } G.$$

This problem is known to be NP-hard and has many practical applications in social network analysis, computational biology, and scheduling.

4.2 Regularized Formulation and Practical Benefits

In our project, we use a regularized version of this formulation to avoid undesirable local optima. The regularized problem becomes:

$$\max_{x \in \Delta} \; x^\top A x + \Phi(x),$$

where $\Phi(x)$ is a convex penalty (regularization) function. Properly chosen regularizers ensure that every local maximizer corresponds to a maximal clique, even though the problem remains non-convex.

The advantage of this approach is that we can now apply first-order optimization methods (like the Frank-Wolfe algorithm and its variants) to solve the problem efficiently over the continuous domain, while still recovering high-quality cliques.

4.3 Why Solving MCP Matters

Solving the maximum clique problem has a wide range of applications. For example:

- In social networks, it helps detect tightly-knit communities.
- In bioinformatics, it can be used to identify highly connected protein complexes.
- In scheduling and resource allocation, maximal cliques represent mutually compatible assignments.

These practical motivations, combined with the rich mathematical structure of the problem, make the MCP a central benchmark for combinatorial optimization algorithms.

4.4 Projected Gradient Descent

In the PGD algorithm, each update step moves the solution vector in the direction of the gradient. However, since the problem is constrained to the standard simplex (i.e., the solution must be non-negative and sum to 1), each update must be projected back onto the simplex to ensure feasibility.

To achieve this, we use the project to simplex() function, which performs an efficient projection of a real-valued vector $v \in \mathbb{R}^n$ onto the probability simplex:

$$\Delta = \left\{ x \in \mathbb{R}^n \;\middle|\; x_i \geq 0,\ \sum_{i=1}^{n} x_i = 1 \right\}$$

This projection algorithm, based on sorting and thresholding, ensures that:

- All components remain non-negative.
- The sum of the components is exactly 1.
- The Euclidean distance between the input and the projected vector is minimized.

This step is crucial for PGD to operate correctly on the continuous relaxation of the Max-Clique problem, as it enforces the same constraint structure used in the Frank-Wolfe methods, without relying on a Linear Minimization Oracle (LMO).

5 Experimental Results

In this section, we report the results obtained for each dataset using the three algorithms implemented: Frank-Wolfe (FW), Away-Step Frank-Wolfe (AFW), and Pairwise Frank-Wolfe (PFW). For each dataset, we include visualizations of the final values of the variable x and comment briefly on the performance of each method.

5.1 brock200_2.txt

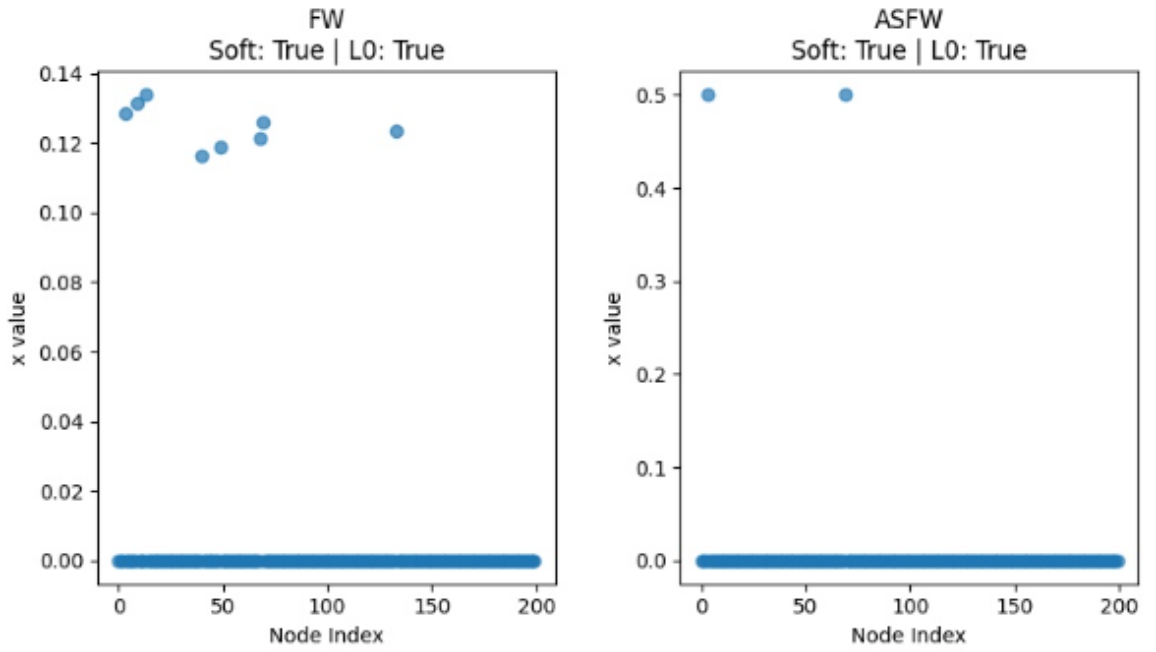

Figure 1: Brock200

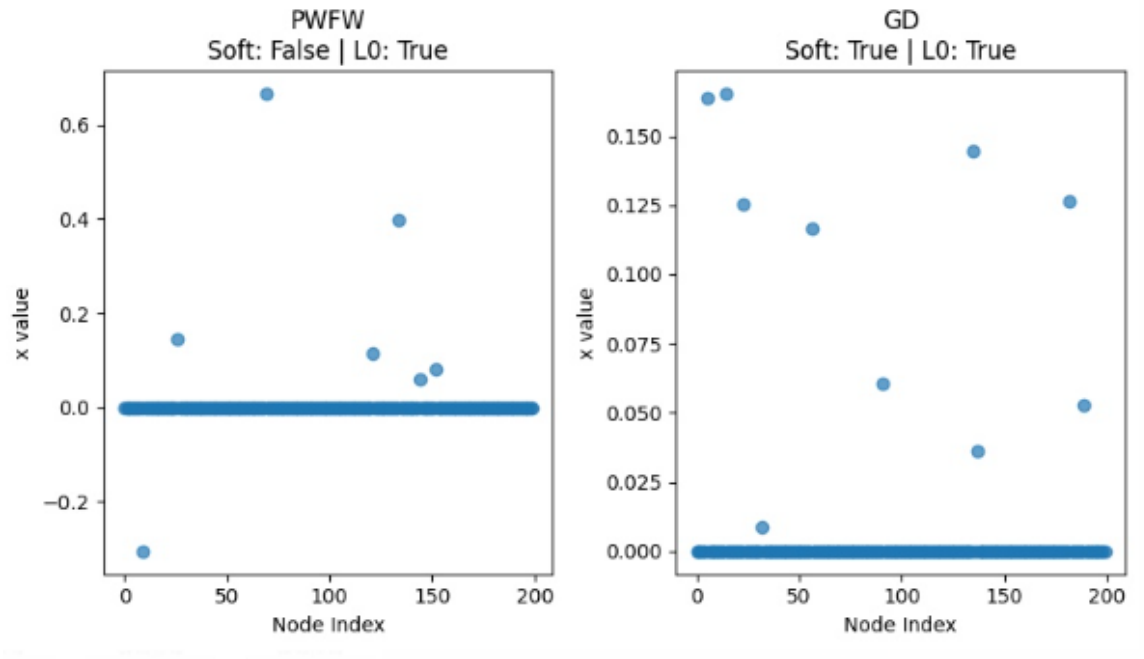

Figure 2: Brock200

Solution structure analysis; In the brock200 dataset, the FW algorithm returned a solution with several small-weight components, indicating a diffuse but richer clique. AFW, on the other hand, selected only two nodes with value 0.5, clearly showing its tendency to collapse into minimal active sets. PFW produced more scattered values, some of which were negative, suggesting instability. GD displayed noisy behavior with small weights distributed across many nodes, without any clear clique structure. Among all, FW provided the most coherent and interpretable solution.

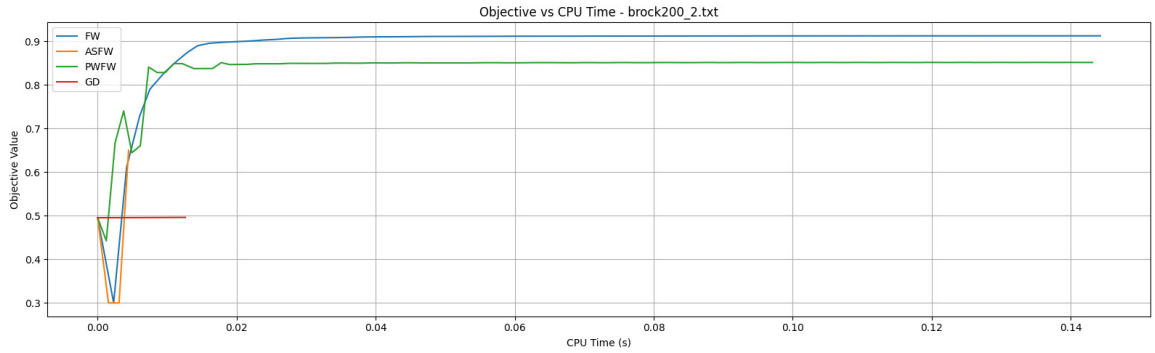

Figure 3: Computation Time

Convergence analysis; In the brock200_2 dataset, Frank-Wolfe (FW) achieved the highest objective value in the shortest amount of time, stabilizing in under 0.02 seconds. Pairwise Frank-Wolfe (PFW) also reached a good value but took slightly longer and showed small oscillations during

early iterations. Away-Step FW (AFW) displayed rapid early movement but plateaued early at a poor value, suggesting it failed to make meaningful progress. Gradient Descent (GD) remained flat at a low objective level, confirming it was the least effective method in this instance.

5.2 brock800_2.txt

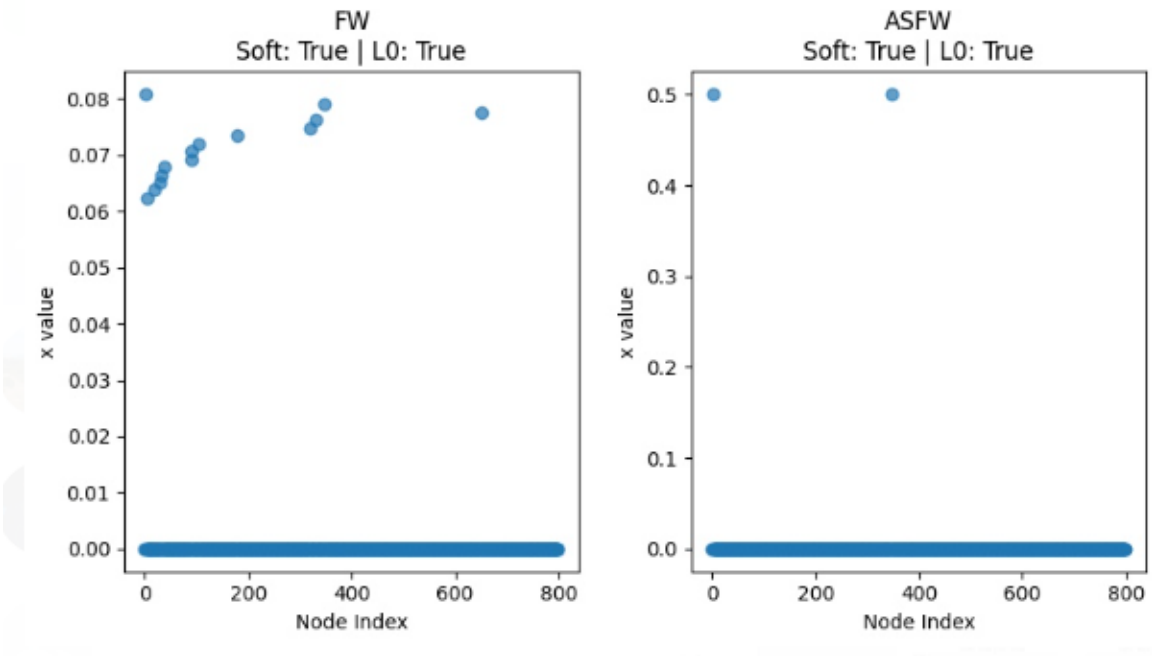

Figure 4: Brock800

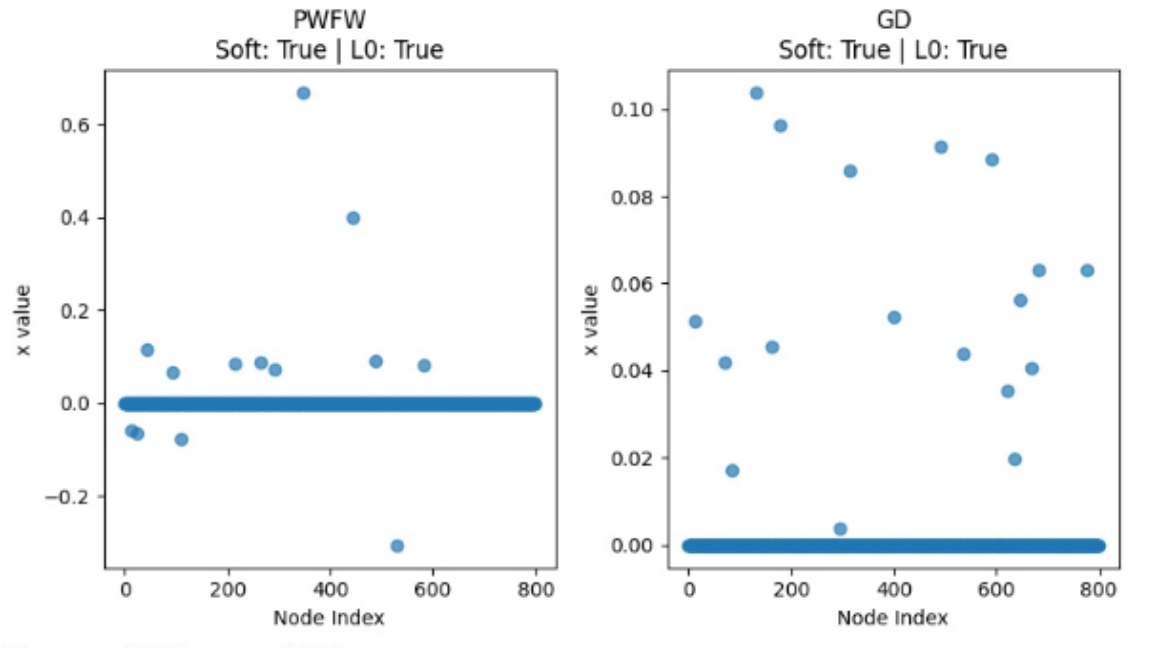

Figure 5: Brock800

Solution structure analysis; In the brock800 instance, FW again generated a sparse solution with more non-zero entries and smoother values, indicating a well-formed large clique. AFW once more converged to only two nodes, with all others zeroed out. PFW showed more scattered activity than AFW but also produced negative components. GD gave a noisy and irregular solution, where no meaningful clique pattern emerged. This further confirms that FW scales better and retains solution interpretability on larger graphs.

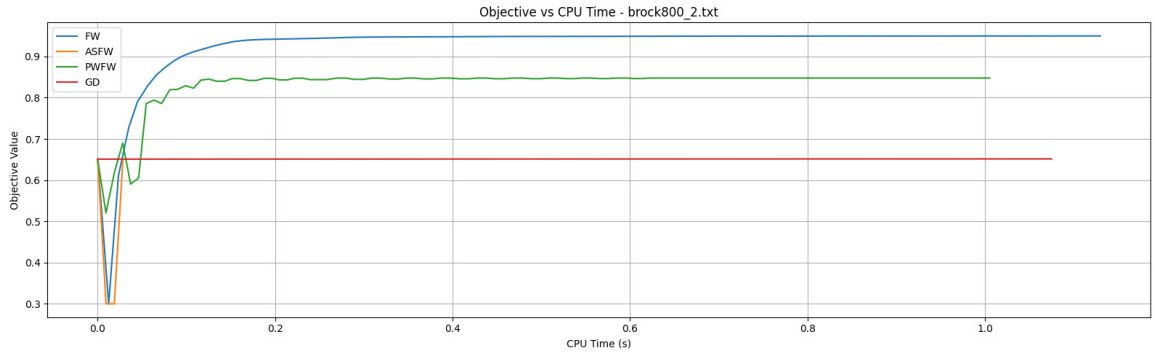

Figure 6: Computation Time

Convergence analysis; For the larger brock800_2 dataset, FW once again showed the best convergence, reaching the top objective level smoothly within approximately 0.2 seconds. PFW demonstrated similar behavior, though convergence was slower and subject to more oscillation.

AFW stagnated early, never recovering beyond its initial steps, while GD flattened at a significantly lower level and did not improve. These results highlight the robustness of FW even in large-scale graphs.

5.3 C125.9.txt

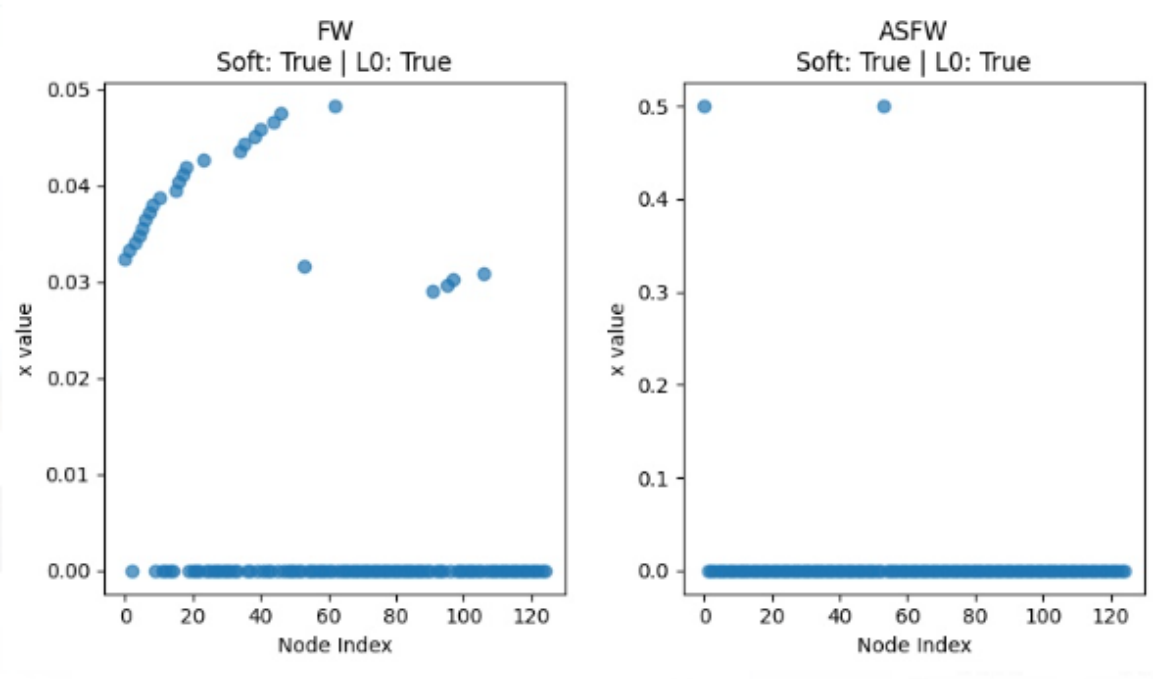

Figure 7: C125.9

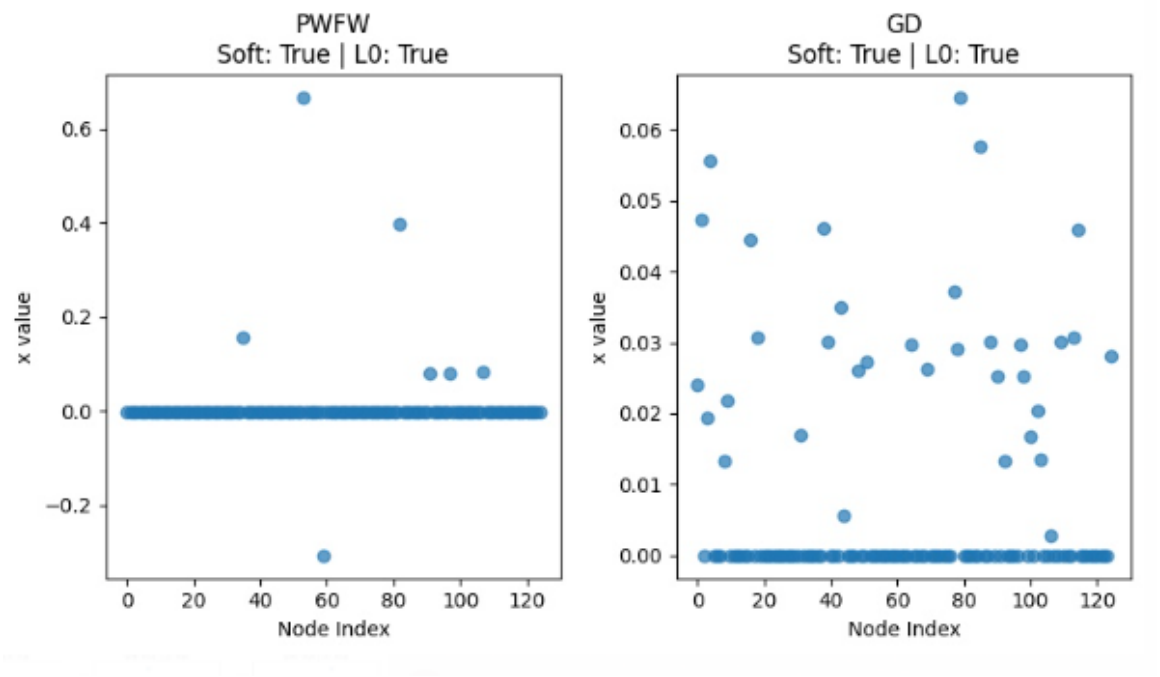

Figure 8: C125.9

Solution structure analysis; On the smaller C125.9 graph, FW successfully identified a denser structure with many low-weight non-zero components, suggesting the presence of a large clique. AFW maintained its pattern of selecting exactly two active nodes, a poor representation of clique structure. PFW gave more variety in node values but again included outliers and negatives. GD resulted in scattered, low-magnitude weights, again lacking any clear selection pattern. Once again, FW proved the most consistent across different scales.

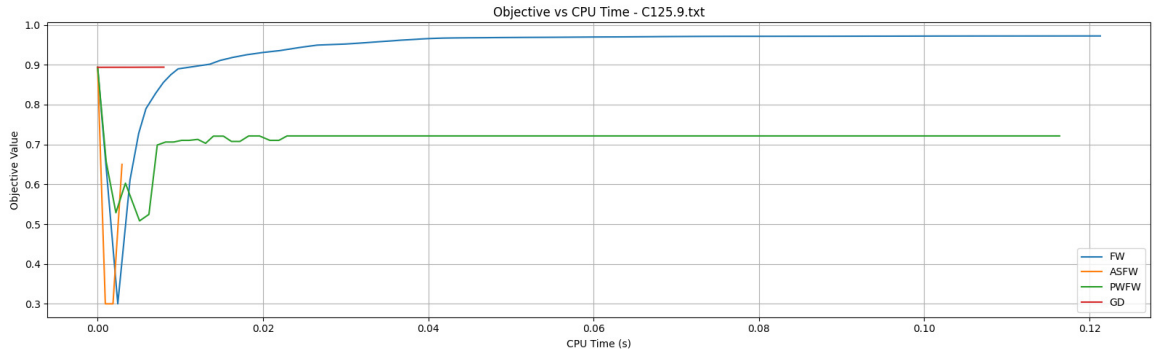

Figure 9: Computation Time

Convergence analysis; In the C125.9 instance, FW maintained its strong performance, reaching the optimal region rapidly and without instability. PFW followed a similar path but required

more time and presented more fluctuations. AFW again plateaued early at a suboptimal level. Interestingly, GD performed slightly better in this instance than in the others, yet still remained inferior to FW and PFW in both value and speed. The clear winner in convergence speed and solution quality was again FW.

6 Conclusion

In this project, we implemented and tested several variants of the Frank-Wolfe algorithm , namely the standard version, Away-Step FW, and Pairwise FW to solve the continuous relaxation of the Maximum Clique Problem. We also integrated a Projected Gradient Descent (PGD) method as a baseline for comparison.

From our experiments across different datasets, we observed that the standard Frank-Wolfe consistently provided the most stable and interpretable solutions. It converged quickly, even for larger graphs, and often returned meaningful clique structures. On the other hand, AFW tended to collapse into minimal active sets with limited node selection, while PFW showed intermediate performance with slightly better sparsity but some instability. PGD, although stable, failed to reach high-quality solutions in most cases.

Overall, our implementation confirms the effectiveness of the Frank-Wolfe framework for solving structured optimization problems over the simplex. In future work, we could explore additional regularization techniques, tune the step-size policies more aggressively, or incorporate gradient-based methods like PGD into hybrid strategies. Testing on larger and more diverse graphs would also help assess the generalization of our findings.