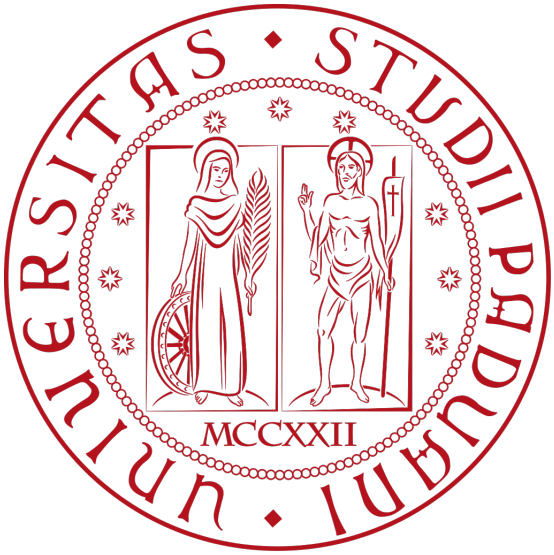

Università degli Studi di Padova

Optimization for Data Science

Optimization Methods for Clustering:

Ellada Aslanova, 2009473
Auriane Mahfouz, 2072042
Oday Najad, 2107495

Department of Mathematics

June, 2024

Contents

1 Abstract

This report explores the implementation and evaluation of Frank-Wolfe algorithm and its variants for solving the Maximum Clique problem in clustering optimization. Specifically, we develop and implement the Frank-Wolfe, Pairwise Frank-Wolfe, and Away Step Frank-Wolfe algorithms, incorporating tools such as line search and Linear Minimization Oracle (LMO) to enhance efficiency. These algorithms are tested on the ℓ_2-regularized and ℓ_0-regularized max-clique problems using three datasets from the DIMACS Instances. The performance and results of these algorithms are analyzed through various plots, showing how effectively they work for clustering problems.

2 Introduction

The Maximum Clique Problem (Hungerford and Rinaldi (2019)), a fundamental optimization problem, has applications in various fields such as networks analysis, bioinformatics, and clustering. It consists of finding the largest clique i.e. the complete subgraph of a graph that contains the maximum number of nodes. This problem is an NP-hard problem and solving it is computationally expensive.

In this report, we are going to address the Maximum Clique problem by implementing the Frank-Wolfe algorithm and its variants: Pairwise Frank-Wolfe, and Away Step Frank-Wolfe. The Frank-Wolfe algorithm has gained popularity in handling constrained optimization problems, but with a slow convergence rate. Therefore, we also introduce its variants in order to improve the convergence rates and hence, efficiency.

We focus on two regularized versions of the Maximum Clique problem: the ℓ_2-regularized max-clique problem and the ℓ_0-regularized max-clique problem. Regularization introduces additional constraints that improve the robustness of the solutions by adding a penalty term, promoting sparsity and leading to solutions with fewer non-zero elements.

In order to test the performance of these algorithms, we choose three datasets from the DIMACS instances which presents a benchmark for optimization problems.

This report shows the implementation of the algorithms and the helper functions and presents the results and performance of each of our methods. We aim to solve the max-clique problem using Frank-Wolfe variants and different regularization techniques.

2.1 Formulation

The Maximum Clique Problem can be based on solving the continuous quadratic programming formulation which is:

$$\begin{aligned} &\max \quad x^T A x \\ &\text{subject to} \quad x \in \Delta, \end{aligned} \tag{1}$$

with Δ the n-dimensional simplex:

$$\Delta := \{x \in \mathbb{R}^n : x \geq 0 \text{ and } \mathbf{1}^T x = 1\},$$

and $A = (a_{ij})_{i,j \in \mathcal{V}}$ denotes the adjacency matrix for the graph G defined by

$$a_{ij} = \begin{cases} 1, & (i,j) \in \mathcal{E} \\ 0, & (i,j) \notin \mathcal{E} \end{cases} \quad \forall i, j \in \mathcal{V}.$$

Where $\mathcal{E}$ is the set edges of the graph.
For any nonempty set $S \subseteq \mathcal{V}$, we let $x(S) \in \Delta$ denote the corresponding characteristic vector (defined by $x(S)_i = \frac{1}{|S|}$ whenever $i \in S$ and $x(S)_j = 0$ otherwise).

In our formulation, in order to have the problem smooth and convex enough, we want to add a penalization term Phi in the objective function:

$$\begin{aligned} &\max \quad x^T A x + \Phi \\ &\text{subject to} \quad x \in \Delta, \end{aligned} \tag{1}$$

3 Material and methods

3.1 Linear Minimization Oracle

The Linear Minimization Oracle (Lacoste-Julien (2015)) is a pivotal component of the Frank-Wolfe algorithm, designed to identify the direction that minimizes the linear approximation of the objective function within the problem constraints. This oracle facilitates progress towards an optimal or near-optimal solution by iteratively updating the current solution based on the linearized objective function.

In the Frank-Wolfe algorithm, at each iteration t, the objective function $f(x)$ is approximated linearly around the current iterate x^t:

$$g(x) = \nabla f(x^t)^T(x - x^t) + f(x^t)$$

where $\nabla f(x^t)$ denotes the gradient of f evaluated at x^t.

The Linear Minimization Oracle then solves the following subproblem to find the optimal direction s^t:

$$s^t = \arg\min_{s \in \mathcal{C}} \nabla f(x^t)^T s$$

Here, $\mathcal{C}$ represents the feasible set or constraint set defined by the problem.

The main goal of the Linear Minimization Oracle is to identify s^t such that:

$$\nabla f(x^t)^T s^t = \min_{s \in \mathcal{C}} \nabla f(x^t)^T s$$

This ensures s^t is the direction of steepest descent in the linearized function $g(x)$ within the constraints of the problem.

Once s^t is determined, the algorithm updates the current iterate x^t using a step size γ^t:

$$x^{t+1} = x^t + \gamma^t s^t$$

where γ^t is chosen to ensure sufficient decrease in the objective function $f(x)$, often determined through a line search or other step size determination methods.

By iteratively applying the Linear Minimization Oracle, the Frank-Wolfe algorithm converges towards an optimal or near-optimal solution that satisfies both the problem constraints and the optimization objective. This iterative process ensures that the algorithm progresses efficiently towards a solution that balances optimization and constraint satisfaction.

3.1.1 Unit Simplex

For the specific Maximum Clique Problem, the constraint is the unit simplex which presents useful characteristics that allows simpler computation for finding the search direction which generally is:

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

. In fact, the search direction can be simplified into

$$s^t = e_{i_k}, \text{ with } i_k = \text{argmin}_i \nabla f(x_k)$$

3.2 Line Search

Line search is a fundamental technique in optimization that aims to determine the optimal step size along a given search direction. This method plays a crucial role in accelerating convergence towards the optimal solution by ensuring that each step taken in the iterative optimization process effectively minimizes the objective function.

Given a current iterate x^t and a search direction s^t, the objective is to find an optimal step size γ^t that minimizes the objective function $f(x)$ along the direction s^t:

$$\gamma^t = \arg\min_{\gamma \geq 0} f(x^t + \gamma s^t)$$

The goal is to find γ^t such that $f(x^t + \gamma^t s^t)$ is minimized.

The line search process (Bomze, Rinaldi, and Zeffiro (2021)) typically involves evaluating $f(x^t + \gamma s^t)$ for different values of γ and selecting the value that provides the greatest reduction in the objective function. Common methods for conducting line search include:

- **Exact Line Search**: This method evaluates the objective function at a finite set of points and selects the exact γ^t that minimizes $f(x^t + \gamma s^t)$.
- **Backtracking Line Search**: Here, an initial step size γ is chosen, and then the step size is reduced iteratively until a sufficient decrease in the objective function is achieved, often using a sufficient decrease condition (e.g., Armijo condition) to determine the reduction in f.
- **Interpolation-based Line Search**: This approach uses interpolation techniques (e.g., quadratic or cubic interpolation) to estimate the optimal step size based on function evaluations.

The choice of line search method depends on the specific characteristics of the objective function and the constraints of the optimization problem. By carefully selecting the step size γ^t, line search ensures that the optimization algorithm avoids overshooting or undershooting the optimal solution, thereby improving the overall efficiency and convergence rate.

Effective line search techniques contribute significantly to the convergence properties of optimization algorithms by enabling them to progress efficiently towards the optimal solution. By dynamically adjusting the step size based on the local behavior of the objective function, line search methods facilitate rapid convergence while maintaining stability and robustness in the optimization process.

3.3 Frank-Wolfe Algorithm

In our project, we implement the classic Frank-Wolfe algorithm (Jaggi (2013)), which is well-suited for minimizing convex functions under linear constraints. This iterative optimization method starts with an initial solution vector and updates it incrementally to approach the optimal solution.

Algorithm Overview:

Algorithm 1 Frank-Wolfe Algorithm

Input: Initial point x^0, objective function f, feasible region $\mathcal{C}$, termination criterion.
Output: Optimal solution x^*.
Initialize $t \leftarrow 0$.
while not converged **do**
    Compute gradient: $\nabla f(x^t)$.
    Determine search direction: $s^t = \arg\min_{s \in \mathcal{C}} \langle \nabla f(x^t), s \rangle$.
    Perform line search to find step size γ^t: $\gamma^t = \arg\min_{\gamma \in [0,1]} f(x^t + \gamma s^t)$.
    Update solution: $x^{t+1} = x^t + \gamma^t s^t$.
    $t \leftarrow t + 1$.
end while
return x^*.

Mathematical Formulation:

The Frank-Wolfe algorithm minimizes a convex function $f(x)$ subject to linear constraints represented by $\mathcal{C}$. At each iteration t, it computes the gradient $\nabla f(x^t)$ and finds the direction s^t that minimizes the linear approximation of f over $\mathcal{C}$. The step size γ^t is determined via line search to ensure sufficient progress towards minimizing f.

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

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

The solution update rule is given by:

$$x^{t+1} = x^t + \gamma^t s^t$$

Key Features and Advantages:

- **Constraint Handling:** Frank-Wolfe efficiently handles problems with linear constraints by iteratively updating the solution vector within the feasible region.
- **Linear Minimization Oracle:** The method's effectiveness relies on the linear minimization oracle, which identifies the optimal direction for progress at each iteration.

- **Convergence Properties:** Although Frank-Wolfe may have slower convergence compared to methods like gradient descent, it is advantageous in scenarios where the feasible region is complex or the objective function is nonsmooth.

Implementation Considerations:

- **Gradient Computation:** Efficient computation of the objective function's gradient $\nabla f(x)$ is crucial for the algorithm's performance.
- **Line Search Technique:** Choosing an appropriate line search strategy helps in determining the step size γ^t that balances progress and stability during each iteration.

3.4 Pairwise Frank-Wolfe Algorithm

In our project, we explore the Pairwise Frank-Wolfe algorithm (Lacoste-Julien (2015)), an extension of the classic Frank-Wolfe method designed to accelerate convergence by considering pairs of directions in each iteration. This approach aims to distribute the weight between two directions, potentially improving the exploration of the feasible region and enhancing convergence rates.

Algorithm Overview:

Algorithm 2 Pairwise Frank-Wolfe Algorithm

Input: Initial point x^0, objective function f, feasible region $\mathcal{C}$, termination criterion.
Output: Optimal solution x^*.
Initialize $t \leftarrow 0$.
while not converged **do**
    Compute gradient: $\nabla f(x^t)$.
    Determine pair of directions: $(s_1^t, s_2^t) = \arg\min_{s_1, s_2 \in \mathcal{C}} \langle \nabla f(x^t), s_1 + s_2 \rangle$.
    Perform line search to find step size γ^t: $\gamma^t = \arg\min_{\gamma \in [0,1]} f(x^t + \gamma(s_1^t + s_2^t))$.
    Update solution: $x^{t+1} = x^t + \gamma^t(s_1^t + s_2^t)$.
    $t \leftarrow t + 1$.
end while
return x^*.

Mathematical Formulation:

The Pairwise Frank-Wolfe algorithm seeks to minimize a convex function $f(x)$ under linear constraints represented by $\mathcal{C}$. At each iteration t, it computes the gradient $\nabla f(x^t)$ and identifies a pair of directions (s_1^t, s_2^t) that jointly minimize the linear approximation of f over $\mathcal{C}$. The step size γ^t is determined via line search to ensure effective progress towards minimizing f.

$$(s_1^t, s_2^t) = \arg \min_{s_1, s_2 \in \mathcal{C}} \langle \nabla f(x^t), s_1 + s_2 \rangle$$

$$\gamma^t = \arg \min_{\gamma \in [0,1]} f(x^t + \gamma(s_1^t + s_2^t))$$

The solution update rule is given by:

$$x^{t+1} = x^t + \gamma^t(s_1^t + s_2^t)$$

Key Features and Advantages:

- **Enhanced Convergence:** Pairwise direction selection can improve convergence rates by more effectively exploring the feasible region.
- **Balanced Adjustment:** Distributing the weight between two directions helps in balancing adjustments and potentially speeding up convergence.
- **Suitability:** Particularly useful in scenarios where the objective function exhibits nonsmooth behavior or the feasible region is complex.

Implementation Considerations:

- **Direction Pair Selection:** Efficiently determining the pair of directions (s_1^t, s_2^t) that minimizes the linear approximation is critical for algorithm performance.
- **Line Search Technique:** Similar to the classic Frank-Wolfe algorithm, an appropriate line search strategy helps in determining the step size γ^t that balances progress and stability during each iteration.

3.5 Away Step Frank-Wolfe Algorithm

In our project, we explore the Away Step Frank-Wolfe algorithm (Lacoste-Julien (2015)), a variant of the classic Frank-Wolfe method designed to improve convergence rates, especially in scenarios where the optimal solution lies on the boundary of the feasible region. This algorithm introduces "away steps" that allow the method to move away from previously chosen vertices, potentially accelerating convergence by exploring new directions efficiently.

Algorithm Overview:

Algorithm 3 Away Step Frank-Wolfe Algorithm

Input: Initial point x^0, objective function f, feasible region $\mathcal{C}$, termination criterion.
Output: Optimal solution x^*.
Initialize $t \leftarrow 0$.
while not converged **do**
    Compute gradient: $\nabla f(x^t)$.
    Determine away direction: $s^t = \arg\max_{s \in \mathcal{C}} \langle \nabla f(x^t), s \rangle$.
    Perform line search to find step size γ^t: $\gamma^t = \arg\min_{\gamma \in [0,1]} f(x^t + \gamma s^t)$.
    Update solution: $x^{t+1} = x^t + \gamma^t s^t$.
    $t \leftarrow t + 1$.
end while
return x^*.

Mathematical Formulation:

The Away Step Frank-Wolfe algorithm aims to minimize a convex function $f(x)$ under linear constraints represented by $\mathcal{C}$. At each iteration t, it computes the gradient $\nabla f(x^t)$ and identifies an "away" direction s^t that maximizes the linear approximation of f over $\mathcal{C}$. The step size γ^t is determined via line search to ensure effective progress towards minimizing f.

$$s^t = \arg\max_{s \in \mathcal{C}} \langle \nabla f(x^t), s \rangle$$

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

The solution update rule is given by:

$$x^{t+1} = x^t + \gamma^t s^t$$

Key Features and Advantages:

- **Improved Convergence:** Away steps enable the algorithm to explore new directions efficiently, particularly beneficial when the optimal solution lies on the boundary of the feasible region.
- **Avoiding Local Minima:** Moving away from previously chosen vertices helps in escaping local minima and potentially finding a global optimum.
- **Robustness:** The algorithm's ability to dynamically adjust directions based on gradient information enhances robustness across different problem landscapes.

Implementation Considerations:

- **Direction Selection:** Efficiently determining the away direction s^t that maximizes the linear approximation is crucial for algorithm efficiency.
- **Line Search Technique:** Similar to other variants of the Frank-Wolfe algorithm, an appropriate line search strategy helps in determining the step size γ^t that balances progress and stability during each iteration.

3.6 Regularization

In our project, we evaluate the three algorithms using objective functions designed for finding the maximum clique in a graph, incorporating ℓ_2 and ℓ_0 regularization terms.

3.6.1 ℓ_2-Regularization

ℓ_2-regularization, also known as ridge regularization, adds a penalty proportional to the square of the solution's magnitude. This regularization technique is effective in promoting smoother solutions and reducing overfitting by penalizing large coefficient values.

Mathematical Formulation:

The objective function with ℓ_2-regularization is formulated as:

$$\min_{x \in \mathbb{R}^n} \left\{ f(x) + \lambda \|x\|_2^2 \right\}$$

where $f(x)$ is the original objective function, $\|x\|_2^2$ represents the ℓ_2 norm squared of x, and $\lambda \geq 0$ is the regularization parameter controlling the strength of regularization.

Algorithmic Implementation:

For each iteration of the algorithms (Frank-Wolfe, Pairwise Frank-Wolfe, and Away Step Frank-Wolfe), the ℓ_2-regularization term is incorporated into the gradient computation step. the gradient $\nabla f(x^t)$ is adjusted to include the gradient of the ℓ_2-regularization term:

$$\nabla f(x^t) + 2\lambda x^t$$

This adjustment ensures that each algorithm iteration moves towards a solution that balances the original objective $f(x)$ with the ℓ_2-regularization penalty, promoting smoother and more stable solutions. Following the (Hungerford and Rinaldi (2019)), we used $\lambda = \frac{1}{2}$.

3.6.2 ℓ_0-Regularization

ℓ_0-regularization, also known as sparsity regularization, adds a penalty based on the number of non-zero elements in the solution vector x. This technique encourages sparsity in the solution, effectively promoting solutions with fewer non-zero elements.

Mathematical Formulation:

In the classical theory, the objective function with ℓ_0-regularization is formulated as:

$$\min_{x \in \mathbb{R}^n} \{ f(x) + \lambda \|x\|_0 \}$$

where $\|x\|_0$ denotes the ℓ_0 norm of x, representing the count of non-zero elements, and $\lambda \geq 0$ is the regularization parameter.

Algorithmic Implementation:

In the algorithmic implementation of ℓ_0-regularization, the ℓ_0 norm is directly integrated into the objective function evaluation and gradient computation steps. During each iteration of the algorithms, the gradient $\nabla f(x^t)$ is adjusted to incorporate the effect of ℓ_0-regularization. We used the formula in Hungerford and Rinaldi (2019) for the penalization term, which is:

$$\nabla f(x^t) + \nabla(\Phi_2(x))$$

where Φ_2 is:

$$\Phi_2(\mathbf{x}) := \alpha_2 \sum_{i=1}^{n} \left(e^{-\beta x_i} - 1\right)$$

with $\beta > 0$ and $0 < \alpha_2 < \frac{2}{\beta^2}$ Thus, the corresponding gradient is:

$$\nabla(\Phi_2(x)) := -\alpha_2 \beta \sum_{i=1}^{n} \left(e^{-\beta x_i}\right)$$

4 Results

4.1 Datasets

In this study, we utilized several graph instances from the DIMACS benchmark (Johnson and Trick (1996)) to evaluate the performance of different variants of the Frank-Wolfe algorithm in solving the Maximum Clique problem. Specifically, we used instances from three families: `C`, `DSJC`, and `gen`. From the `C` family, we included the instance `C125.9`, from the `DSJC` family, we selected `DSJC500_5`, and lastly, from the `gen` family, we included `gen200_p0.9_44`. These instances provide a diverse set of graph structures, allowing us to comprehensively test and compare the algorithms' effectiveness and efficiency in identifying the maximum cliques under different regularization conditions (ℓ_2 and ℓ_0).

4.2 ℓ_2-regularized Max Clique Problem

The ℓ_2-regularized Max Clique Problem is tested on the different variants of the Frank-Wolfe algorithm by comparing the objective value that needs to be maximized in our case. The goal is to find the largest possible clique in the given graph, so the objective function would be maximized to find the maximum clique size.

In cases where ℓ_2 regularization is introduced, the objective function incorporates penalties that discourage large norms. Thus, the primary objective is typically to maximize the clique size while accounting for these penalties.

For ℓ_2 regularization, the objective function can be defined as:

$$\text{Objective}(x) = \text{score}(x) - \Phi_B(x)$$

Where:

- $\Phi_B(x) = \frac{1}{2}\|x\|_2^2$,
- x is the binary vector indicating the presence of vertices in the clique.
- $\text{score}(x) = x^\top A x$ represents the score of the clique in terms of the adjacency matrix A.
- $\|x\|_2^2 = \sum_{i=1}^{n} x_i^2$ is the squared Euclidean norm of x.
- $\lambda = \frac{1}{2}$ is the regularization parameter controlling the importance of the regularization term relative to the score.

4.3 ℓ_0-regularized Max clique Problem

Instead, for ℓ_0 regularization, the objective function can be defined as:

$$\text{Objective}(x) = \text{score}(x) - \Phi_2(x)$$

Where:

- $\Phi_2(x) = \alpha_2 \sum_{i=1}^{n} \left(e^{-\beta x_i} - 1\right), \quad \text{with } \beta > 0 \text{ and } 0 < \alpha_2 < \frac{2}{\beta^2}$

Φ_2 is a widely recognized approximation of the subsequent nonsmooth function (Bradley, Mangasarian, and Rosen (1998)):

$$\Phi(x) = -\alpha_2 \|x\|_0$$

4.4 Other Regularization term

Following the results obtained with l_2 and l_0 regularization, where we observed a significant difference in performance when utilizing different regularization methods, we are going to introduce one other regularization term (Hungerford and Rinaldi (2019)) to provide an indication of the potential impact that different regularization terms can have on the performance of a local optimization algorithm.

$$\Phi_1(x) := \alpha_1 \|x + \epsilon 1\|_{p'}^{p'}, \quad \text{with } \epsilon > 0,\ p > 2, \text{ and } 0 < \alpha_1 < \frac{2}{p(p-1)(1+\epsilon)^{p-2}},$$

Bomze (1997) presented the two-norm regularization function Φ_B, and Φ_1 is Φ_B's generalization to p-norms, where $p > 2$.

We compared the average time as well as the mean and variance and displayed the results in table1. The mean and variance in the table refer to the objective value obtained from the optimization algorithms using the different regularization methods. The mean represents the average value of the objective function across multiple runs of the algorithm. It gives an indication of the typical performance achieved by the algorithm. On the other hand, the variance measures the variability or spread of the objective function values across multiple runs. A lower variance indicates that the algorithm produces more consistent results, whereas a higher variance suggests greater fluctuation in the performance.

Performance Comparison Across Algorithms

Vanilla Algorithm

Generally, the time taken for the vanilla algorithm to run varied between the various datasets. We notice that, with respect to all the regularization, the mean shows slighter difference. We can say that it is less fluctuating than other variances of the frank-wolfe algorithm. However, the variance shows fluctuating values with respect to different regularization terms.

Pairwise Algorithm

The pairwise algorithm exhibits better performance in terms of speed, primarily because it employs optimization techniques that reduce the search space by considering pairs of elements at each step. This method is efficient in pruning less promising areas of the search space early on, leading to faster convergence. Moreover, the pairwise algorithm often achieves superior mean results as it efficiently navigates through the search space to find larger cliques. The improvement in speed does not come at the expense of accuracy, making this algorithm a good choice for some datasets.

Away-Step Algorithm

The away step algorithm balances well between the vanilla and pairwise algorithms. It combines elements of both approaches, ensuring a comprehensive search while employing the "away-step" strategy to enhance efficiency and speed. The away step method is particularly effective in dealing with the datasets where a purely pairwise or vanilla approach might either miss potential solutions or take too long to compute. By dynamically adjusting the search strategy, the away step algorithm achieves a balance between computational efficiency and solution quality.

Impact of Regularization Methods

Phi Function: Φ_B

The Φ_B regularization method is designed to provide a balanced approach, offering stable outcomes across different algorithms and datasets. It achieves this by maintaining a consistent level of regularization that prevents overfitting while allowing for flexibility in the search process. This balance is crucial in ensuring that the results are not only accurate but also reliable, with lower variance indicating stable performance across multiple runs.

Phi Function: Φ_1

The Φ_1 regularization can be thought as a generalization of the Φ_B, which lead to improvements in mean results by more pruning the search space. However, this approach resulted in higher variance, indicating less consistency across different runs. The Φ_1 method can be beneficial in scenarios where a slight increase in mean results is critical, but the trade-off is a higher risk of fluctuating outcomes.

Phi Function: Φ_2

The Φ_2 regularization method offers a middle ground between Φ_B and Φ_1. It provides reasonable mean results (often lower than the two other techniques) and variance, indicating a balanced approach to regularization. The Φ_2 method is particularly effective in scenarios where both stability and performance are required.

Specific Observations

- **C125.9 Dataset** The pairwise algorithm combined with Φ_1 regularization yields the highest mean result, but with significant variance, indicating less consistency. A good trade-off would be using the away-step algorithm with with Φ_b regularization.
- **DSJC500_5 Dataset** Results are generally close across all algorithms and regularisation terms, with Φ_2 showing a slight edge in terms of mean result and stability.
- **gen200_p0.9_44 Dataset** The pairwise algorithm with Φ_1 regularization yields the highest mean result, but again with high variance.

Table 1: Performance of Different Algorithms on Various Datasets

Dataset	**Algorithm**	**Phi Function**	**Average Time (seconds)**	**Mean Result**	**Variance of Result**
C125.9	vanilla	Φ_B	0.187934	25.652989	0.908371
C125.9	vanilla	Φ_1	0.185410	25.623109	0.856739
C125.9	vanilla	Φ_2	0.091454	25.463214	0.850371
C125.9	pairwise	Φ_B	0.086164	30.268428	2.517514
C125.9	pairwise	Φ_1	0.151100	33.501179	4.261678
C125.9	pairwise	Φ_2	0.101407	29.758731	2.341759
C125.9	away_step	Φ_B	0.182695	29.997595	1.846690
C125.9	away_step	Φ_1	0.194318	33.362757	4.411222
C125.9	away_step	Φ_2	0.161942	29.889105	2.055538
DSJC500_5	vanilla	Φ_B	0.779853	9.490004	0.589896
DSJC500_5	vanilla	Φ_1	0.826810	9.460003	0.648397
DSJC500_5	vanilla	Φ_2	0.795718	9.330004	0.801097
DSJC500_5	pairwise	Φ_B	0.776382	9.384991	0.635891
DSJC500_5	pairwise	Φ_1	0.830917	9.429996	0.685087
DSJC500_5	pairwise	Φ_2	0.812712	9.470244	0.569077
DSJC500_5	away_step	Φ_B	0.902718	9.345201	0.684737
DSJC500_5	away_step	Φ_1	0.867187	9.549992	0.887471
DSJC500_5	away_step	Φ_2	0.917916	9.220246	0.571582
gen200_p0.9_44	vanilla	Φ_B	0.143782	29.953233	0.907039
gen200_p0.9_44	vanilla	Φ_1	0.200097	30.103389	0.688870
gen200_p0.9_44	vanilla	Φ_2	0.168479	30.053155	1.066935
gen200_p0.9_44	pairwise	Φ_B	0.168916	32.057090	1.833864
gen200_p0.9_44	pairwise	Φ_1	0.253009	35.955442	6.400198
gen200_p0.9_44	pairwise	Φ_2	0.189518	31.733379	2.215303
gen200_p0.9_44	away_step	Φ_B	0.243835	32.137788	2.555683
gen200_p0.9_44	away_step	Φ_1	0.298131	35.956403	5.214817
gen200_p0.9_44	away_step	Φ_2	0.259239	31.764255	2.364221

4.4.1 Plots

The following plots allow us to better visualize table 1 by comparing the performance of the different Φ functions on each algorithm for the three instances. Higher peaks indicate larger cliques.

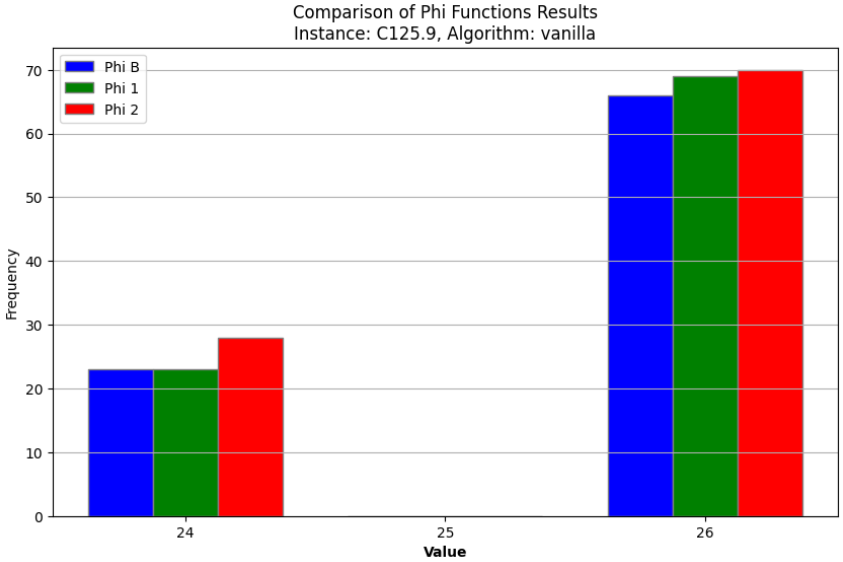

Figure 1: Comparison between the different regularization functions with Vanilla Frank-Wolfe on C125.9

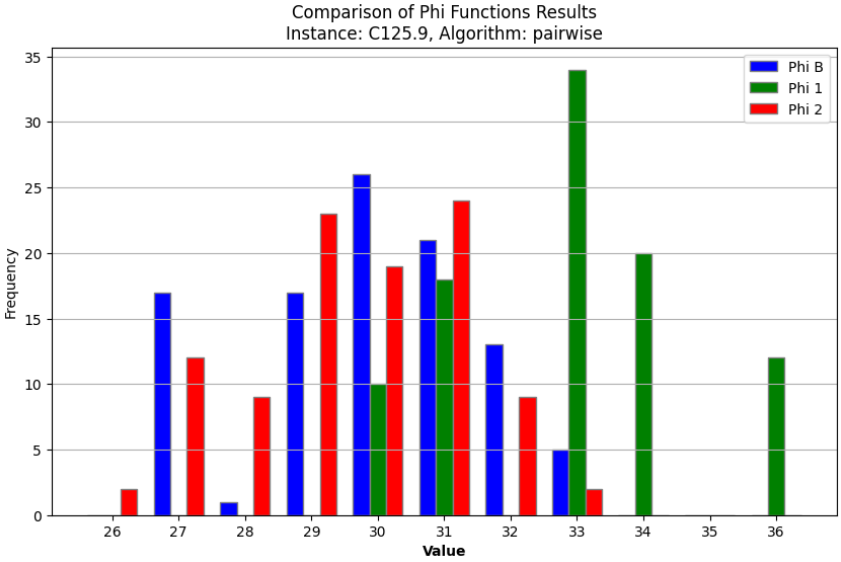

Figure 2: Comparison between the different regularization functions with Pairwise Frank-Wolfe on C125.9

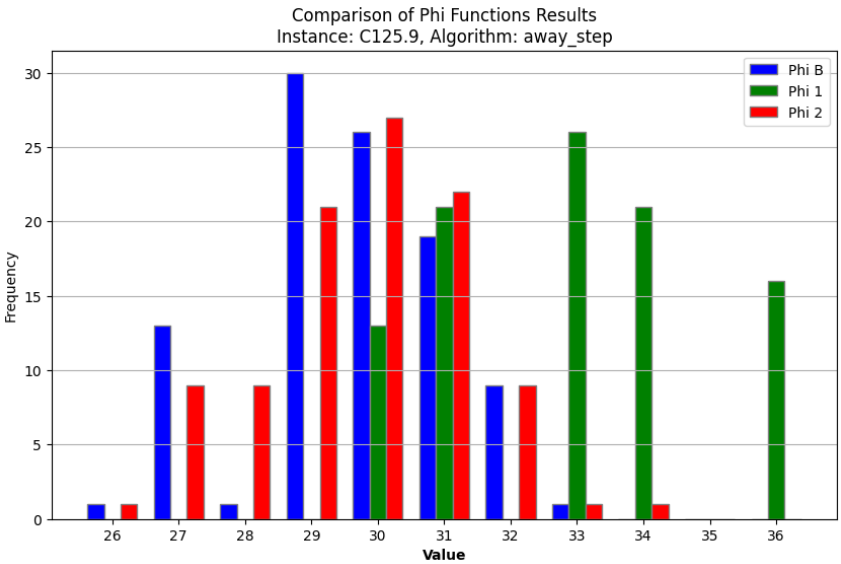

Figure 3: Comparison between the different regularization functions with Away-step Frank-Wolfe on C125.9

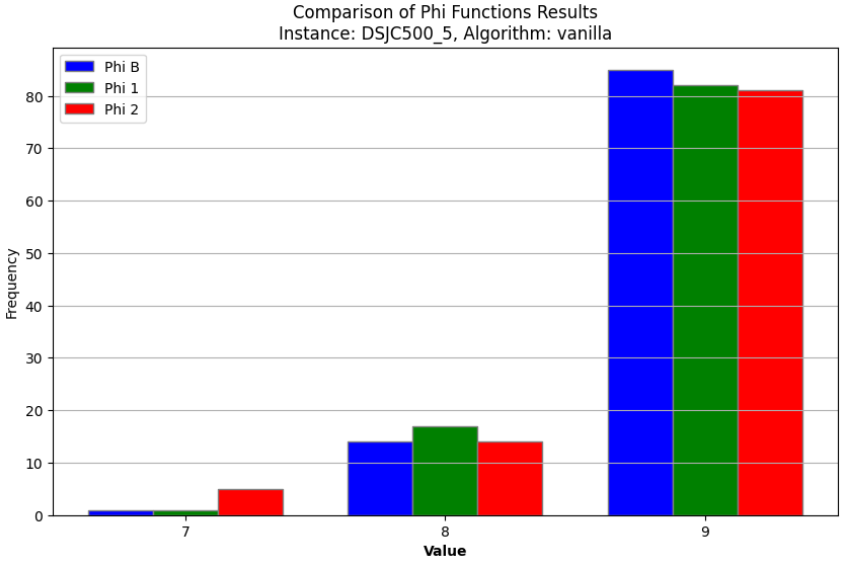

Figure 4: Comparison between the different regularization functions with Vanilla Frank-Wolfe on DSJC500_5

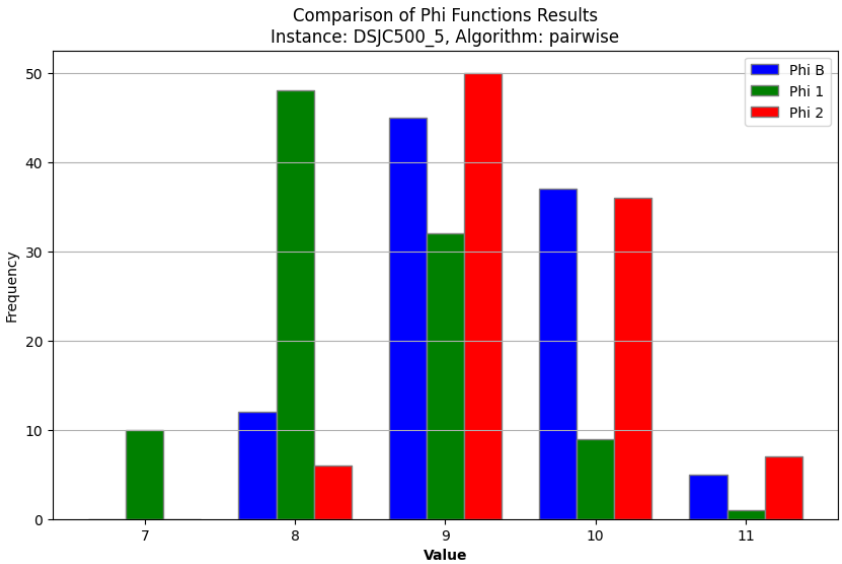

Figure 5: Comparison between the different regularization functions with Pairwise Frank-Wolfe on DSJC500_5

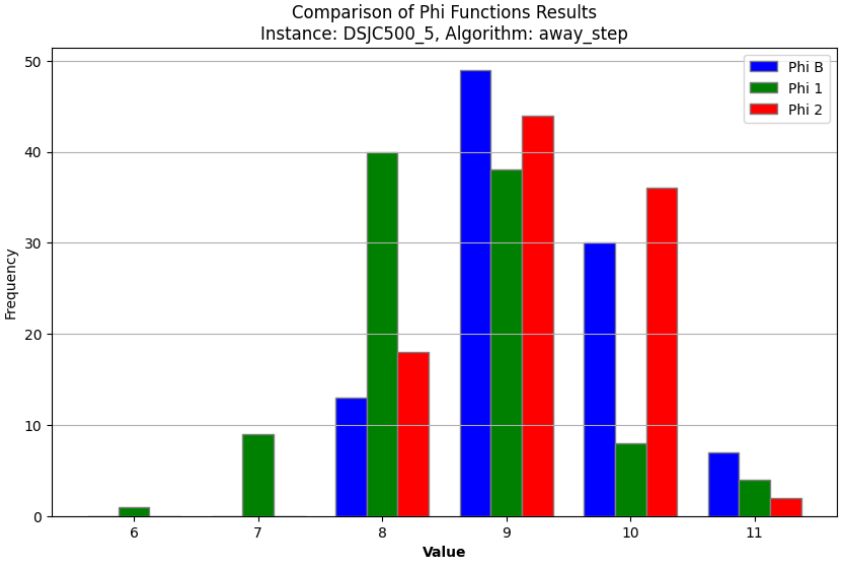

Figure 6: Comparison between the different regularization functions with Away-step Frank-Wolfe on `DSJC500_5`

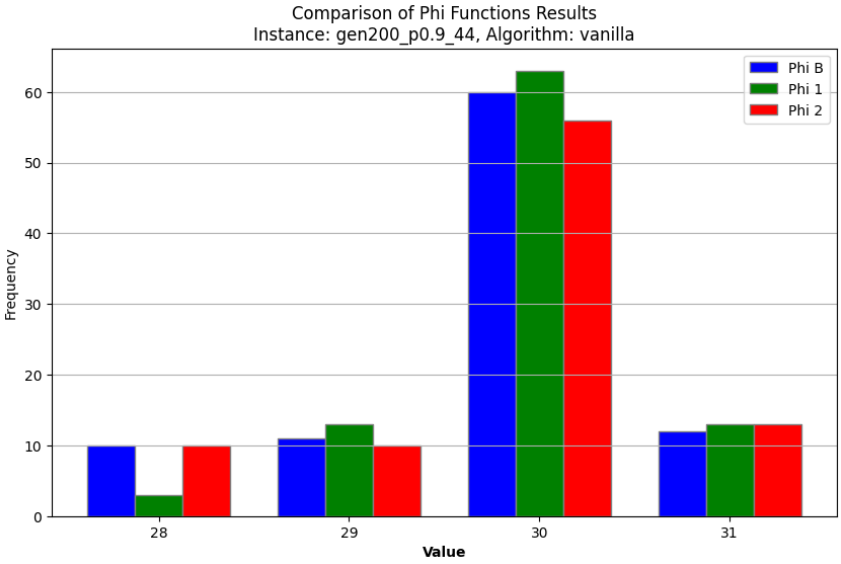

Figure 7: Comparison between the different regularization functions with Vanilla Frank-Wolfe on `gen200_p0.9_44`

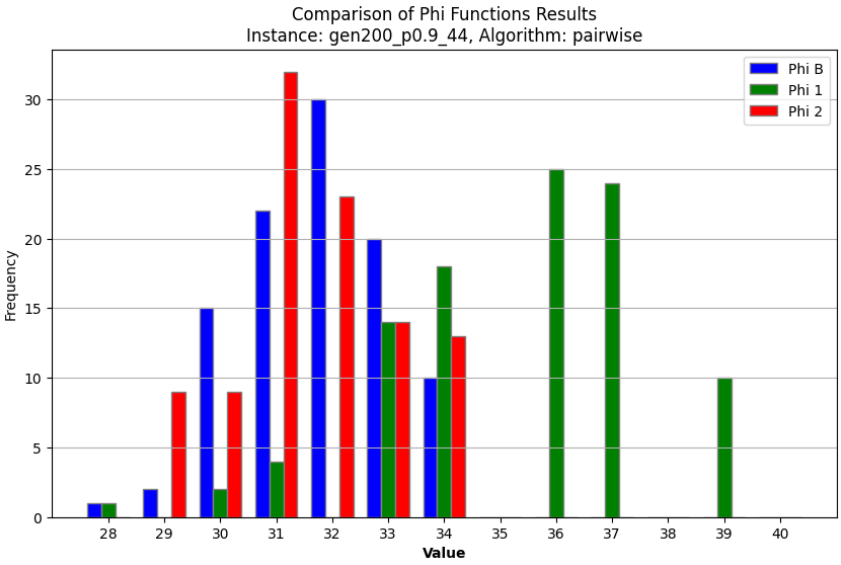

Figure 8: Comparison between the different regularization functions with Pairwise Frank-Wolfe on `gen200_p0.9_44`

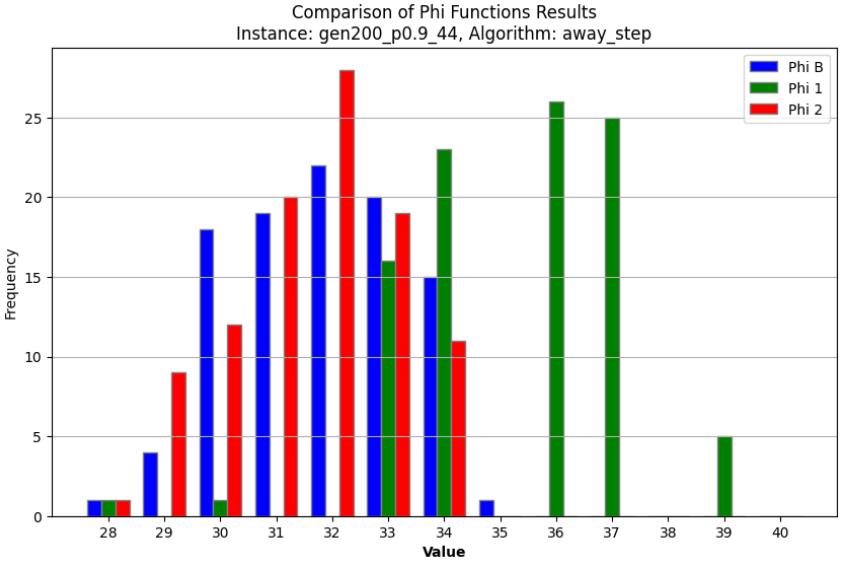

Figure 9: Comparison between the different regularization functions with Away-step Frank-Wolfe on `gen200_p0.9_44`

5 Conclusion

In this report, we have explored various extensions of the classic Frank-Wolfe algorithm to solve the Maximum Clique problem, focusing on the Pairwise Frank-Wolfe, Away Step Frank-Wolfe, and standard Frank-Wolfe algorithms. Through the introduction of Φ_B (ℓ_2-regularization), Φ_1, and Φ_2 ($\ell_0$regularization), we have examined the impact of different regularization methods on the algorithms' performance. Φ_1 often led to high mean results but with high variance, indicating potential for both high performance and instability. Φ_B provided balanced and consistent results, while Φ_2 offered a middle ground between performance and stability.

Future work could explore adaptive regularization techniques that dynamically adjust the regularization parameters based on the problem's evolving landscape, potentially offering even greater performance and stability. Additionally, extending these algorithms to handle even larger and more complex datasets could further validate their applicability and robustness in real-world scenarios.

References

[1] Bomze, I. M. (1997). Evolution towards the maximum clique. *Journal of Global Optimization*, *10*(2), 143-164.

[2] Bomze, I. M., Rinaldi, F., & Zeffiro, D. (2021). Frank–wolfe and friends: A journey into projection-free first-order optimization methods. *Journal of Global Optimization*. doi: 10 .1007/s10898-021-01090-0

[3] Bradley, P., Mangasarian, O., & Rosen, J. (1998). Parsimonious least norm approximation. *Computational Optimization and Applications*, *11*(1), 5-21.

[4] Hungerford, J. T., & Rinaldi, F. (2019). A general regularized continuous formulation for the maximum clique problem. *Mathematics of Operations Research*, *44*(4), 1287-1307. doi: 10.1287/moor.2018.0954

[5] Jaggi, M. (2013). Revisiting frank-wolfe: Projection-free sparse convex optimization. *arXiv preprint arXiv:1309.5438*.

[6] Johnson, D. S., & Trick, M. A. (1996). *Cliques, coloring, and satisfiability: Second dimacs implementation challenge, october 11–13, 1993* (Vol. 26). Providence, RI: American Mathematical Society.

[7] Lacoste-Julien, S. (2015). On the global linear convergence of frank-wolfe optimization variants. *arXiv preprint arXiv:1511.05932*.