DEPARTMENT OF SCIENCE
OPTIMIZATION FOR DATA SCIENCE
FINAL PROJECT

PADOVA
A.A. 24-25

CONTINUOUS OPTIMIZATION FOR CLIQUE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Contents

Riccardo Niccolò Agosti	riccardoniccolo.agosti@studenti.unipd.it	2142932
Andrea Calzarotto	andrea.calzarotto@studenti.unipd.it	2160876
Marco Augusto Carturan	marcoaugusto.carturan@studenti.unipd.it	2159630
Anthony Fantin	anthony.fantin@studenti.unipd.it	2160877

1 Introduction

The Maximum Clique Problem (MCP) is a fundamental challenge in graph theory and combinatorial optimization. Given a simple undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, a clique is a subset of vertices $C \subseteq V$ where every two distinct vertices are adjacent. The MCP seeks to find a clique of the maximum possible size in a given graph. This problem is well-known to be NP-hard and has a wide range of applications in fields such as bioinformatics, social network analysis, and scheduling [3].

1.1 Problem Formulation

A powerful approach to tackling the combinatorial nature of the MCP is to reformulate it as a continuous optimization problem. A well-known continuous formulation is the quadratic program proposed by Motzkin and Straus [3].

$$\max_{x \in \Delta} \quad x^T A x \tag{1}$$

where A is the adjacency matrix of the graph G, and Δ is the standard simplex in $\mathbb{R}^n$ (with $n = |V|$):

$$\Delta := \left\{ x \in \mathbb{R}^n \mid \sum_{i=1}^{n} x_i = 1,\ x_i \geq 0\ \forall i \in V \right\}$$

However, a known drawback of this classic formulation is the potential presence of "spurious" local maximizers that do not correspond to characteristic vectors of cliques in the graph [3]. This can cause optimization algorithms to terminate at points from which a valid clique cannot be easily recovered.

To address this issue, regularization techniques have been introduced. By adding a regularization term $\Phi(x)$ to the objective function, it is possible to establish a one-to-one correspondence between the local maximizers of the new problem and the maximal cliques of the graph. This project focuses on solving this general regularized formulation, as detailed in Hungerford and Rinaldi [3]:

$$\max_{x \in \Delta} \quad f(x) := x^T A x + \Phi(x) \tag{2}$$

Different choices for the regularization function $\Phi(x)$ lead to different problem characteristics. In this report, we specifically explore two variants also analyzed in Hungerford and Rinaldi [3]: an l_2-regularization term and a smooth concave approximation of the l_0-norm. The specific forms are:

$$\Phi_{l_2}(x) = \frac{1}{2}||x||_2^2 \tag{3}$$

$$\Phi_{l_0}(x) = \alpha_2 \sum_{i=1}^{n} (e^{-\beta x_i} - 1) \tag{4}$$

where α, α_2, β are positive scalar parameters.

The goal of this work is to implement and analyze the performance of several first-order optimization algorithms namely Frank-Wolfe, Pairwise Frank-Wolfe, Away-step Frank-Wolfe, and Projected Gradient Descent for solving these two variants of the regularized maximum clique problem on benchmark graph instances. The algorithms for this task are derived from foundational and modern works on Frank-Wolfe methods [1, 5, 4].

2 Optimization Algorithms and Step-Size

2.1 Step-Size Strategies

A crucial component of any iterative first-order optimization algorithm is the determination of the step-size, γ_k, at each iteration k. In this project, we implement and compare three distinct strategies.

Exact Line Search (Optimal Gamma)

This strategy aims to find the optimal step-size by solving a one-dimensional optimization subproblem at each iteration [1, p. 325]. For our maximization problem, this is formally defined as:

$$\gamma_k = \arg \max_{\gamma \in [0, \gamma_{\max}]} f(\mathrm{x}_k + \gamma \mathrm{d}_k)$$

This approach is explicitly described as a primary step-size strategy for Frank-Wolfe variants [5, p. 3].

Armijo Rule (Inexact Line Search)

This is a "backtracking" method that finds a "good enough" step-size without being strictly optimal. It starts with a large trial step and reduces it iteratively until a sufficient ascent condition is met [1, p. 325]. The condition is:

$$f(\mathrm{x}_k + \gamma \mathrm{d}_k) \geq f(\mathrm{x}_k) + c_1 \gamma \langle \nabla f(\mathrm{x}_k), \mathrm{d}_k \rangle$$

where c_1 is a small constant.

Fixed Step-Size

This is the simplest strategy where the step-size is a pre-defined constant (0.05 in our experiments). The effective step is constrained by the maximum allowable step for a given move, $\gamma_{\max}$:

$$\gamma_k = 0.05$$

Using a constant step-size with the classic Frank-Wolfe algorithm can lead to a slower theoretical convergence rate [1, p. 325].

2.2 Projected Gradient Method

The Projected Gradient (PG) method is a fundamental iterative algorithm for constrained optimization. At each iteration, it takes a step in the direction of the gradient and then projects the resulting point back onto the feasible set. The update rule is:

$$\mathrm{x}_{k+1} = P_\Delta(\mathrm{x}_k + \gamma_k \nabla f(\mathrm{x}_k))$$

where $P_\Delta(\cdot)$ is the Euclidean projection onto the simplex. This projection can be computed efficiently, with state-of-the-art algorithms having linear time complexity [2].

2.3 Frank-Wolfe Method

The Frank-Wolfe (FW) algorithm is a first-order optimization algorithm well-suited for problems where linear optimization over the feasible set is cheap [4]. It constructs a solution as a convex combination of a few vertices (atoms) of the feasible set [1, p. 314]. The update rule is:

1. Compute $\nabla f(x_k)$.
2. Solve the Linear Minimization Oracle (LMO): $s_k = \arg\max_{s \in \Delta} \langle s, \nabla f(x_k) \rangle$.
3. Update the solution: $x_{k+1} = (1 - \gamma_k)x_k + \gamma_k s_k$.

A standard choice for the step-size is a diminishing schedule, such as $\gamma_k = \frac{2}{k+2}$ [4, p. 1].

2.4 Pairwise Frank-Wolfe Method

The Pairwise Frank-Wolfe (PFW) method is an extension designed to accelerate convergence by moving "mass" directly between two vertices at each step [5, p. 4]. The procedure is:

- Compute $\nabla f(x_k)$.
- Identify the "FW vertex" s_k (best global vertex).
- Identify the "away vertex" v_k (worst active vertex).
- Form the pairwise direction $d_k = s_k - v_k$.
- Perform a line search to find the optimal $\gamma_k \in [0, \alpha_{v_k}]$, where α_{v_k} is the weight of the away vertex.
- Update the solution: $x_{k+1} = x_k + \gamma_k d_k$.

This contrasts with the classic FW, which shrinks all active weights at every iteration [5, p. 4].

2.5 Away-step Frank-Wolfe Method

The Away-step Frank-Wolfe (AFW) algorithm also aims to accelerate convergence by allowing steps that move "away" from existing active vertices [5, p. 3]. The procedure is as follows:

1. Compute $\nabla f(x_k)$.
2. Identify the FW vertex s_k and the away vertex v_k.
3. Calculate the potential progress for the FW direction ($d_{FW} = s_k - x_k$) and the away direction ($d_A = x_k - v_k$).
4. Choose the direction that offers greater progress.
5. Perform a line search along the chosen direction with an appropriate maximum step-size.
6. Update the solution: $x_{k+1} = x_k + \gamma_k d_k$.

This modification can lead to linear convergence for certain classes of problems [5, p. 1].

3 Datasets and Experiments

In this section, we describe the data we used and the structure of the computational experiments we performed to test our optimization methods.

3.1 Datasets

In order to do our tests, we used three publicly available graph datasets from the DIMACS challenge, as mentioned in [3]. These graphs are standard benchmarks for clique-finding algorithms and vary in size. The specific files we used were:

- `C250-9.mtx`: A graph with 250 nodes.
- `C500-9.mtx`: A graph with 500 nodes.
- `C2000-9.mtx`: A larger graph with 2000 nodes.

These graphs were loaded as adjacency matrices, which our algorithms used as input.

3.2 Experimental Setup

Our main goal was to compare the performance of the above mentioned optimization algorithms and settings. We designed an experiment that systematically tested many combinations of parameters.

The core components of our experiment were:

- Projected Gradient Descent (PGD)
- Frank-Wolfe (FW)
- Pairwise Frank-Wolfe (PFW)
- Away-Step Frank-Wolfe (AFW)

For each algorithm, we tried running it with two different types of regularization to see if it helped guide the search:

- L2 regularization (l_2): A standard technique that penalizes large values.
- L0 regularization (l_0): A penalty that encourages sparse solutions (i.e., solutions with many zeros).

Choosing the right step size (or learning rate) is critical for these algorithms. We tested three different approaches:

- Optimal Gamma (Line Search): At each step, we calculate the mathematically best step size.
- Armijo Rule: A popular method that finds a "good enough" step size without being as slow as an exact line search.
- Fixed Gamma: We just used a constant, small step size of 0.05.

Some algorithms, especially when using l_0 regularization, can get stuck in bad local optima. To deal with this, we used a multi-start strategy for the PFW and AFW algorithms. We ran them 5 times for each setting, starting from different random points, and kept only the best result found across those runs. For PGD and FW, we used a single, standard starting point.

We basically ran a large set of nested loops to test every possible combination: for each graph, we tested each algorithm with each regularization type and each step size strategy.

To provide a comprehensive comparison, we collected several key metrics from each experimental run:

- Clique Size: The number of vertices in the clique found by the algorithm. This is the primary measure of solution quality.
- Clique Validity: A boolean flag to confirm that the set of vertices returned is indeed a valid clique (i.e., all vertices are mutually connected).
- Runtime: The total wall-clock time in seconds required for the algorithm to converge, averaged over the multi-start runs. This measures computational efficiency.
- Loss: The final value of the objective function (negated, since we are maximizing). This allows for a direct comparison of the optimization performance.

These metrics were compiled into a summary table for analysis, allowing for a direct comparison of the performance of each algorithmic configuration across the different datasets.

4 Experimental Results and Analysis

This section presents the results obtained from the experimental evaluation of the four optimization algorithms on the benchmark DIMACS datasets [6]. We analyze and compare the performance of each algorithm and step-size strategy based on the key metrics defined in the experimental setup.

4.1 Analysis of Average Clique Size

A high-level overview of the algorithms performance is presented in Figure 1, which shows the average clique size found by each algorithm, aggregated across all datasets and grouped by the type of regularization used.

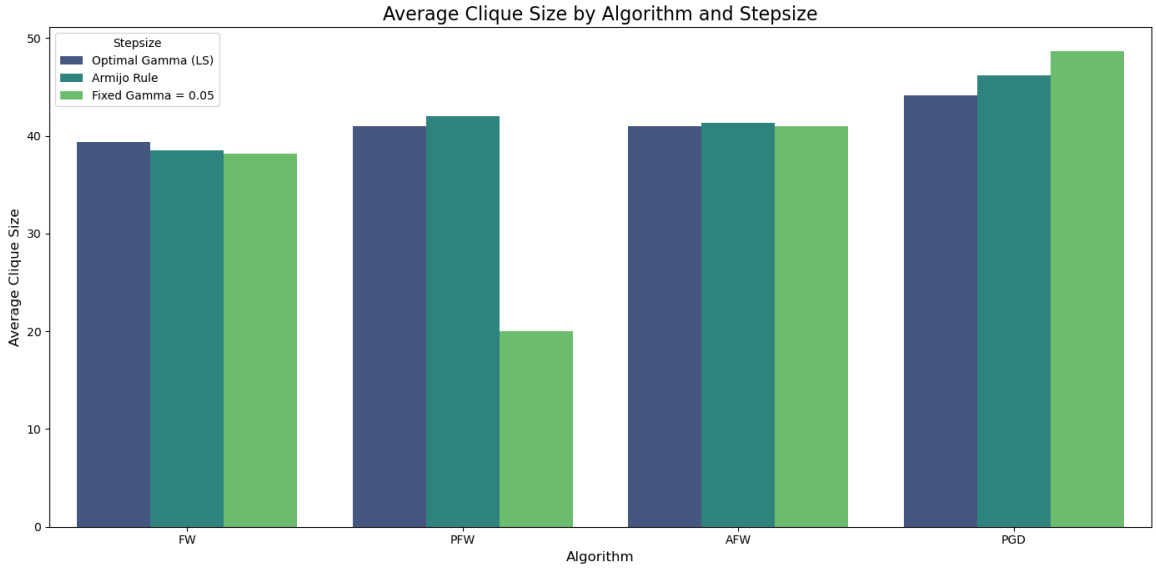

Figure 1: Average clique size found by each algorithm, grouped by regularization type.

PGD emerges as the overall best-performing algorithm. All three bars for PGD are taller than most other bars in the chart, indicating that this algorithm finds larger cliques on average in this specific context. FW and AFW show very similar and stable performance, with an average clique size hovering around 40-42, regardless of the stepsize strategy used. PFW exhibits the most extreme performance variability: it performs similarly to FW and AFW with adaptive stepsizes (Optimal Gamma and Armijo), but its performance collapses dramatically with a fixed stepsize.

For FW and AFW the choice of stepsize has a minimal impact infact the three strategies lead to nearly identical results and this suggests that these algorithms are robust and not particularly sensitive to the choice of stepsize in this experiment. For PFW this algorithm is extremely sensitive to the choice of stepsize, infact using a Fixed Gamma is clearly a poor choice, reducing the average clique size to about 20, which is less than half of what is achieved with other configurations, in contrast, the adaptive stepsizes (Optimal and Armijo) ensure excellent performance. For PGD an opposite trend is observed compared to PFW; while adaptive strategies provide good results, it is the fixed stepsize that achieves the absolute best result in the entire chart, reaching an average clique size of about 47.

The combination of the PGD algorithm with a fixed stepsize of 0.05 is the undisputed winner in this comparison, achieving the best performance. The Fixed Gamma strategy proves to be a high-risk, high-reward approach: it produced the best result (with PGD) but also the absolute worst one (with PFW), its effectiveness is critically dependent on the algorithm it is paired with.

The Optimal Gamma (LS) and Armijo Rule strategies prove to be safe and reliable choices infact they deliver consistently good or very good performance across all algorithms, avoiding catastrophic failures like the one seen with PFW.

4.2 Analysis of Computational Runtimes

The figure 2 bar chart illustrates the average runtime in seconds for four different algorithms (FW, PFW, AFW, and PGD), each tested with three distinct stepsize strategies.

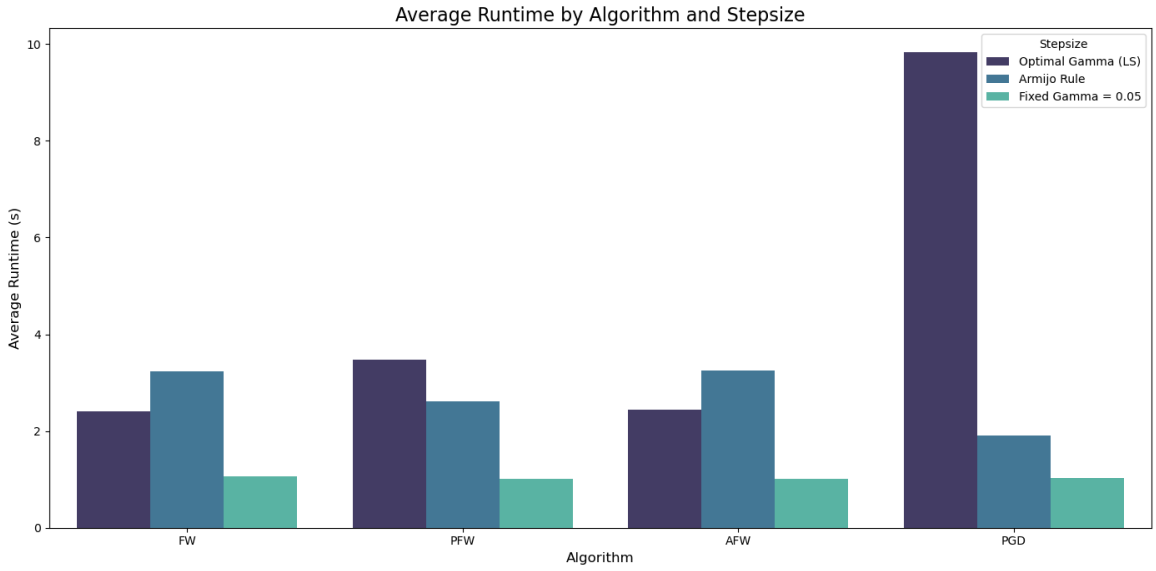

Figure 2: Average runtime (in seconds) by algorithm and step-size strategy, aggregated across all datasets and regularizations.

The most immediate and consistent trend visible in the chart is the exceptional speed of the Fixed Gamma $= 0.05$ strategy , this is expected, as using a fixed value completely avoids the computational overhead associated with calculating a new stepsize at each iteration.

The combination of the PGD algorithm with the Optimal Gamma (LS) strategy is exceptionally slow, with an average runtime of nearly 10 seconds. This is by far the highest cost of any configuration, making this specific pairing impractical for applications where speed is a factor.

Comparing the two adaptive strategies, the Armijo Rule and Optimal Gamma (LS), both consistently require more time than the fixed stepsize. For the Frank-Wolfe family of algorithms (FW, PFW, AFW), their runtimes are relatively comparable, generally falling within the 2-4 second range, however, for the PGD algorithm, the difference is stark: the Armijo Rule is significantly faster and more efficient than the extremely costly Optimal Gamma line search.

In conclusion, the graph clearly demonstrates a computational trade-off, while adaptive methods may offer benefits in terms of solution quality they come at the price of increased runtime. The Fixed Gamma strategy is the undisputed champion of efficiency, furthermore, the analysis highlights a significant risk associated with the Optimal Gamma (LS) strategy, as its computational cost can become prohibitively high for certain algorithms like PGD.

4.3 Analysis of Regularization Impact

To understand the effect of the chosen regularization schemes on solution quality, we analyzed the distribution of clique sizes found by the different algorithms under both l_2 and approximate l_0 regularization. The box plots in the figure illustrate the median, quartiles, and range of clique sizes, providing a detailed view of performance.

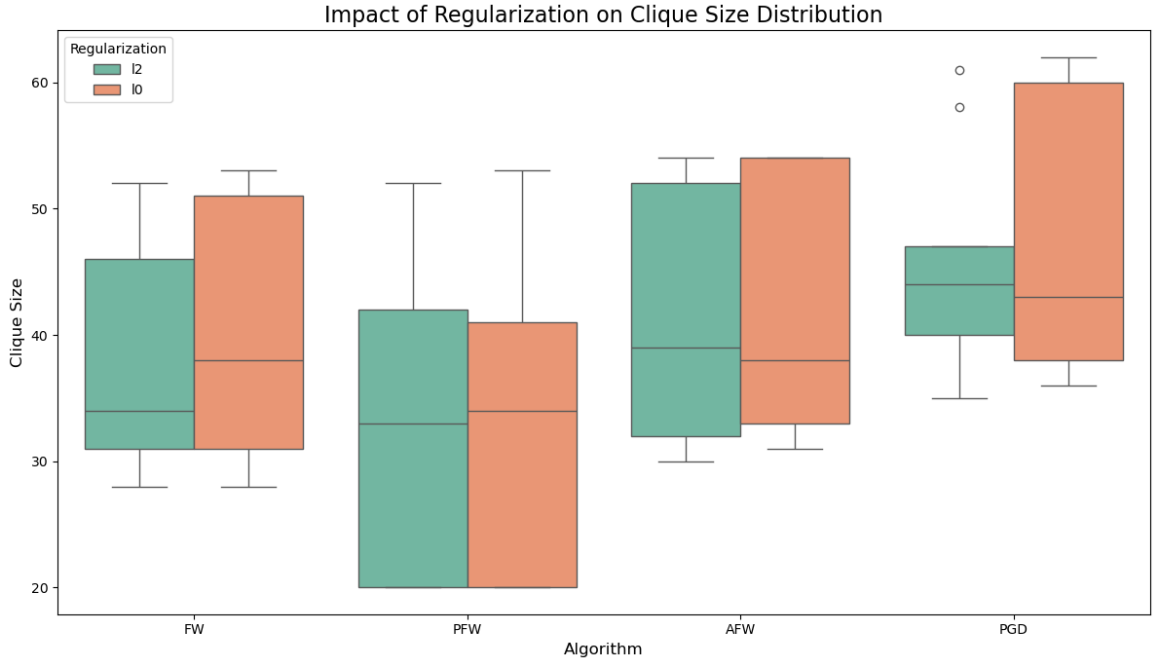

Figure 3: Distribution of valid clique sizes found by each algorithm, grouped by regularization type (l_2 vs. l_0).

For the FW and AFW algorithms, the l_0 regularization consistently leads to finding larger cliques compared to the l_2 regularization. For both of these methods, the entire distribution for the l_0 results is shifted upwards relative to the l_2 results, indicating superior performance across the board. The PFW algorithm, however, deviates from this pattern. Its performance under l_0 is highly variable, with a lower median and a much wider distribution compared to the more stable and predictable results from l_2 regularization.

The most striking result is observed with the PGD algorithm, where the data directly contradicts the initial text's claim. For PGD, l_0 regularization is overwhelmingly superior to l_2. The median clique size for l_0 is substantially higher (near 60) than for l_2 (around 45), and its distribution extends to much higher values, including outliers, establishing PGD with l_0 as the most effective combination shown.

This dominant performance of l_0 regularization for FW, AFW, and especially PGD, suggests that its sparsity-inducing nature is exceptionally well-suited for the maximum clique problem. The term $\Phi_{l_0}(\mathrm{x}) = \alpha_2 \sum(e^{-\beta x_i} - 1)$ effectively guides the optimization towards the inherently sparse vectors that characterize a clique, the l_2 regularization $\Phi_{l_2}(\mathrm{x}) = \frac{1}{2}\|\mathrm{x}\|_2^2$, which penalizes large weights rather than promoting true sparsity, appears to be a less effective structural prior for this combinatorial task.

4.4 Analysis of Step-Size Strategy Impact

The chart 4 provides a nuanced look at algorithm performance, exploring the combined impact of stepsize strategy and the type of regularization used l_0 versus l_2 on the average clique size found. The side-by-side comparison reveals that the relationship between an algorithm and its optimal stepsize is not fixed, but changes depending on the problem's formulation.

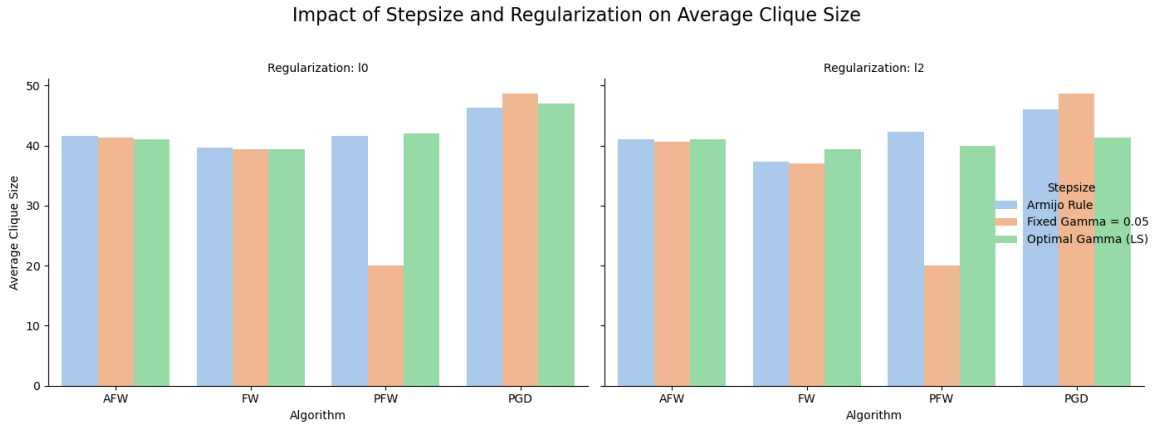

Figure 4: Distribution of valid clique sizes found by each algorithm, grouped by step-size strategy (Optimal Gamma vs. Armijo Rule).

The left panel, showing results with L0 regularization, confirms the key findings from the previous analyses; the PGD algorithm, when paired with a Fixed Gamma of 0.05, stands out as the top performer, achieving the largest average clique size. This panel also reiterates the critical weakness of the PFW algorithm, whose performance collapses when using a fixed stepsize, making it a high-risk choice. In contrast, the AFW and FW algorithms demonstrate consistent robustness, delivering stable and strong results across all stepsize strategies.

The right panel showing result with L2 regularization: the PFW algorithm continues to exhibit a dramatic performance failure when paired with a fixed stepsize, confirming its instability. Likewise, the FW and AFW algorithms maintain their characteristic robustness, as all three stepsize methods continue to produce stable and comparable results for them.

The PGD algorithm is capable of achieving the highest performance, but requires careful tuning that is sensitive to the problem's specifics like the regularization choice, the PFW algorithm is shown to be unreliable, while AFW and FW stand out as consistently robust and safe, albeit not peak-performing, options.

5 Convergence Analysis by Graph

In this section, we provide a detailed analysis of the objective function's convergence behavior. We will examine the results for each of the three graph instances C250, C500, and C2000 to highlight the consistency and scalability of the algorithms performance characteristics. Each analysis considers the interplay between the algorithm, the regularization scheme (l_2 vs. l_0), and the step-size rule (Optimal Gamma vs. Armijo Rule).

5.1 Analysis of Convergence on Graph C250-9

The convergence behavior for the smallest graph, C250-9, is presented in Figure 5.

First, the Projected Gradient Descent (PGD) algorithm shows exceptionally rapid convergence. In all four subplots, the PGD line (dashed red) rises steeply, achieving the majority of its final objective value within the first 15-20 iterations. It consistently reaches the highest plateau, confirming its effectiveness in finding high-quality solutions.

Second, the Frank-Wolfe variants (FW, PFW, AFW) exhibit a slower, more gradual convergence. Their performance curves are tightly clustered, especially under the Optimal Gamma (LS) step-size, indicating very similar iterative behavior.

Finally, the impact of the step-size rule is clear. The left column (Optimal Gamma LS) shows smooth, monotonic increases for all algorithms, as expected from an exact line search. In contrast, the right column (Armijo Rule) displays a characteristic "stair-step" pattern, particularly for FW and AFW. This reflects the non-monotonic nature of the inexact Armijo search, which prioritizes computational speed over guaranteed improvement at every single step.

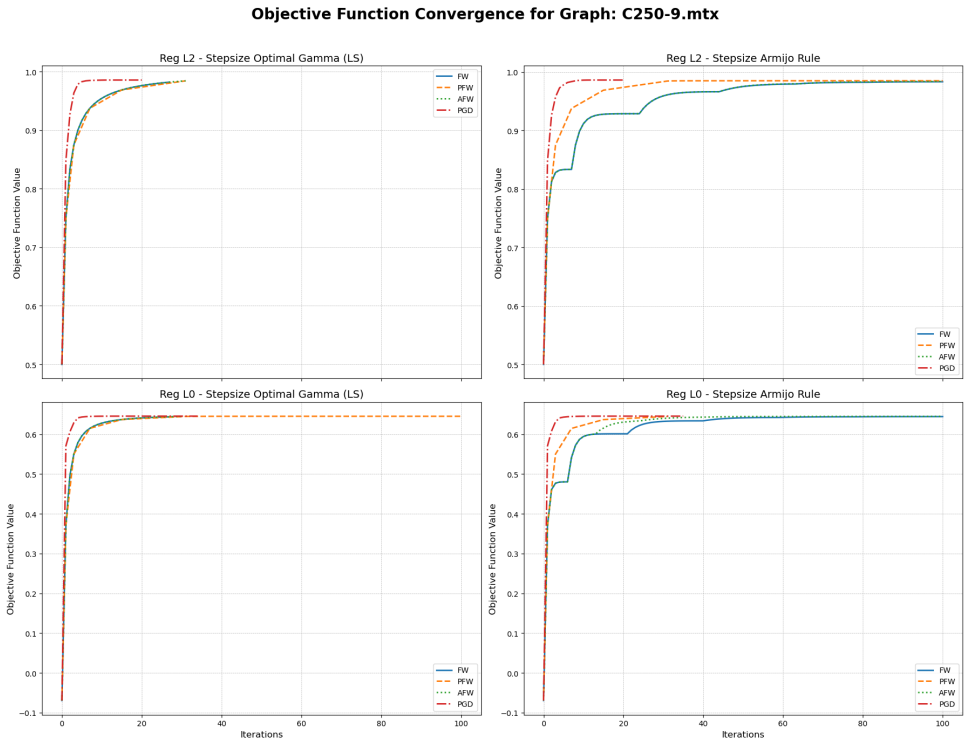

Figure 5: Objective function convergence on the C250-9 graph for different algorithms, regularizations, and step-size rules

5.2 Analysis of Convergence on Graph C500-9

Moving to the medium-sized graph, C500-9 (Figure 6), we observe a remarkable consistency with the patterns seen in the smaller instance. The fundamental behaviors of the algorithms are replicated, demonstrating their robustness.

Once again, the PGD algorithm demonstrates superior performance, characterized by a steep initial ascent and convergence to the highest final objective value across all four settings so its iterative efficiency remains a key advantage. The Frank-Wolfe algorithms again show a slower, clustered convergence path.

The distinction between the step-size rules also persists. The smooth convergence under Optimal Gamma (LS) contrasts sharply with the non-monotonic, stair-step progression under the Armijo Rule. This consistency suggests that these behaviors are inherent to the algorithms and step-size methods themselves, rather than being artifacts of a particular problem instance.

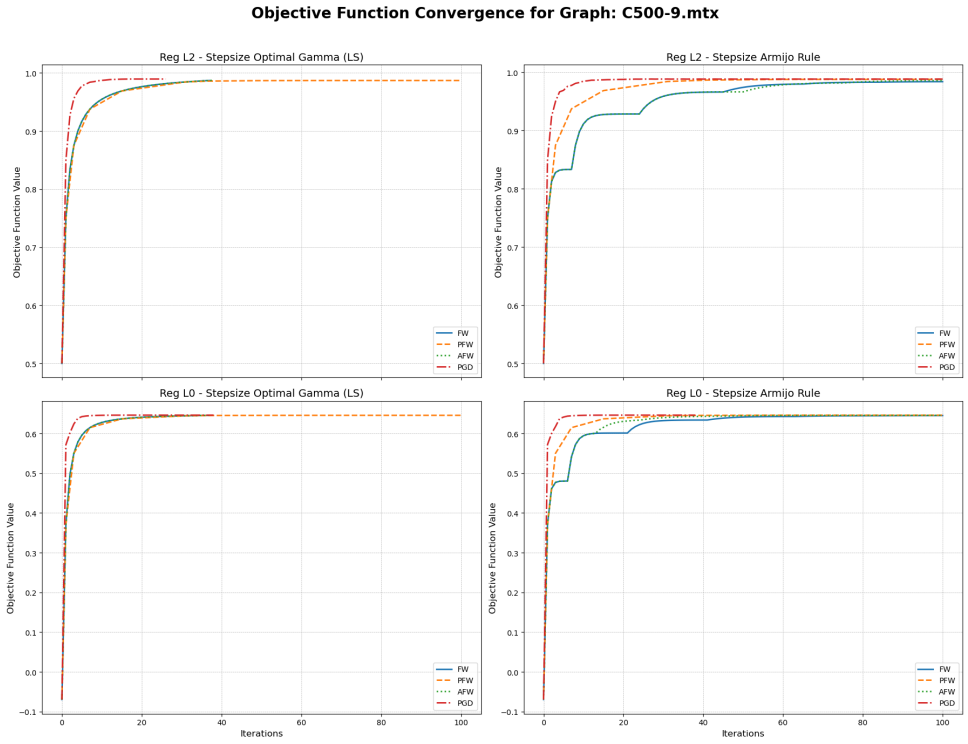

Figure 6: Objective function convergence on the C500-9 graph, confirming the patterns observed in the smaller instance.

5.3 Analysis of Convergence on Graph C2000-9

The analysis of the largest graph instance, C2000-9, presented in Figure 7, further solidifies these findings and demonstrates their scalability to larger and more big problems.

Despite the significant increase in problem size, the relative performance and convergence characteristics of the algorithms remain unchanged. PGD continues its reign as the most iteration-efficient algorithm, rapidly approaching a high-quality solution. The Frank-Wolfe variants maintain their slower, more deliberate convergence pattern. The visual distinction between the smooth convergence paths from the exact line search and the jagged paths from the Armijo rule is as clear as ever.

This consistency across scales is a crucial finding. It indicates that the observed convergence behaviors are fundamental properties of the algorithms' interaction with the problem's structure, allowing for predictable performance even on larger graphs.

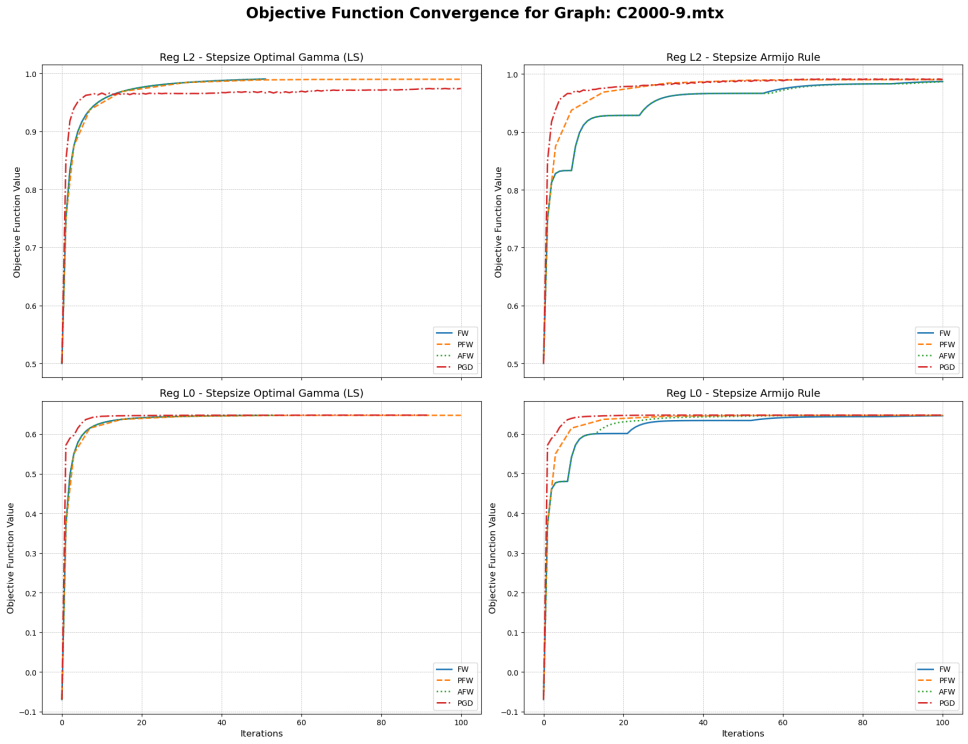

Figure 7: Objective function convergence on the C2000-9 graph, demonstrating the scalability of the observed behaviors.

5.4 Synthesis of Findings

Across all three graph instances, from C250 to C2000, a clear and consistent narrative emerges: the PGD algorithm is unequivocally the most efficient in terms of per-iteration convergence, finding high-quality solutions faster than any of the Frank-Wolfe variants. The Frank-Wolfe methods (FW, PFW, AFW) are effective but require more iterations to converge and exhibit very similar performance to one another.

The choice of step-size rule fundamentally impacts the path of convergence with Optimal Gamma (LS) providing smooth, monotonic increases and the Armijo Rule producing a faster but non-monotonic path, but has a lesser effect on the final converged value. The robustness of these patterns across graphs of varying sizes indicates that the observed performance characteristics are highly predictive of each algorithm's behavior on this class of problems.

6 Conclusion

The consistent patterns of convergence observed across different graph instances in Figures 5, 7 are not coincidental. They are direct consequences of the fundamental mathematical properties of the optimization algorithms and strategies employed. This section provides a deeper reflection on why these behaviors emerge, connecting the visual evidence to the underlying formulas.

6.1 The Dynamics of PGD versus Frank-Wolfe

The most significant observation was the superior per-iteration efficiency of Projected Gradient Descent (PGD) compared to the Frank-Wolfe (FW) variants. This can be explained by examining their respective update rules.

The PGD update at iteration k is given by:

$$\mathrm{x}_{k+1} = P_{\Delta}(\mathrm{x}_k + \gamma_k \nabla f(\mathrm{x}_k))$$

where we use a plus sign for maximization. Here, P_{Δ} is the projection onto the feasible set $\mathcal{C}$, and $\nabla f(\mathrm{x}_k)$ is the gradient of the objective function. The gradient provides the direction of steepest ascent in the unconstrained space. The PGD step is therefore an aggressive, direct move towards the optimum, only corrected at the end by a projection, this allows PGD to make substantial progress with each iteration, especially in the early stages, explaining its steep initial convergence curve.

In contrast, the Frank-Wolfe algorithm first solves a linear subproblem to find a search direction:

$$\mathrm{s}_k = \arg\max_{\mathrm{s}\in\mathcal{C}} \langle \mathrm{s}, \nabla f(\mathrm{x}_k)\rangle \tag{5}$$

The update is then a convex combination of the current point and this new vertex s_k:

$$\mathrm{x}_{k+1} = (1-\gamma_k)\mathrm{x}_k + \gamma_k \mathrm{s}_k = \mathrm{x}_k + \gamma_k(\mathrm{s}_k - \mathrm{x}_k) \tag{6}$$

The search direction $(\mathrm{s}_k - \mathrm{x}_k)$ is constrained to point towards a single vertex of the feasible set. This can be much less aligned with the true steepest ascent direction compared to the full gradient used by PGD. This restriction explains the slower, more gradual progress of the FW-family algorithms.

6.2 The Impact of Stepsize Rules on Convergence Paths

The difference in smoothness between the convergence paths generated by the Optimal Gamma (LS) and Armijo Rule strategies is also mathematically grounded.

An exact line search (LS) finds the optimal step-size γ_k by solving:

$$\gamma_k = \arg\max_{\gamma\in[0,1]} f(\mathrm{x}_k + \gamma \mathrm{d}_k) \tag{7}$$

where d_k is the search direction. By its very definition, this ensures that $f(\mathrm{x}_{k+1}) \geq f(\mathrm{x}_k)$, leading to the perfectly monotonic and smooth curves seen in the left column of the figures.

The Armijo Rule, however, is an inexact line search. It terminates as soon as a "sufficient ascent" condition is met. For maximization, this condition is:

$$f(\mathrm{x}_k + \gamma \mathrm{d}_k) \geq f(\mathrm{x}_k) + c\gamma\langle \nabla f(\mathrm{x}_k), \mathrm{d}_k\rangle \tag{8}$$

for some constant $c \in (0,1)$. The rule uses a backtracking procedure (starting with a large γ and decreasing it) and accepts the first value of γ that satisfies Equation 8. Because this step is not necessarily optimal, the algorithm might take a smaller step than possible.

In the subsequent iteration, a much better direction might become available, leading to a large jump in the objective function. This sequence of accepting "good enough" steps explains the characteristic "stair-step" pattern observed. It trades smoothness for computational efficiency by avoiding the costly subproblem of a full line search.

6.3 Synergy Between Regularization and Algorithm Choice

Finally, our earlier analysis showed that the approximate l_0 regularization was highly effective, particularly when combined with PGD. The objective function in this case is $f(\mathrm{x}) = g(\mathrm{x}) - \Phi_{l0}(\mathrm{x})$, where $g(\mathrm{x})$ is the original quadratic term and the penalty is:

$$\Phi_{l0}(\mathrm{x}) = \alpha_2 \sum_{i=1}^{n} (1 - e^{-\beta x_i}) \tag{9}$$

This term penalizes non-zero entries, directly encouraging the sparsity inherent to the characteristic vector of a clique. The gradient $\nabla f(\mathrm{x})$ therefore contains a strong signal that pushes components towards zero. PGD, with its direct use of the full gradient, is able to effectively leverage this signal to navigate towards sparse, high-quality solutions. The alignment between the problem's combinatorial structure, the regularizer's sparsity-inducing nature, and the algorithm's gradient-based mechanism creates a powerful and effective optimization process. The l_2 norm, in contrast, penalizes large weights rather than promoting true sparsity, making it a less suitable structural prior for this problem, as reflected in the results.

In conclusion, the observed performance is not arbitrary but a direct reflection of the mathematical underpinnings of each method. The success of the PGD algorithm with l_0 regularization stems from a deep synergy between its unconstrained gradient steps and a penalty function that accurately models the combinatorial nature of the maximum clique problem.

References

[1] Immanuel M. Bomze, Francesco Rinaldi, and Damiano Zeffiro. Frank-wolfe and friends: a journey into projection-free first-order optimization methods. 4OR, 19:313–345, 2021.

[2] Laurent Condat. Fast projection onto the simplex and the l_1 ball. Mathematical Programming, 158(1):575–585, 2016.

[3] James T Hungerford and Francesco Rinaldi. A general regularized continuous formulation for the maximum clique problem. Mathematics of Operations Research, 44(4):1161–1173, 2019.

[4] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In Proceedings of the 30th International Conference on Machine Learning (ICML), pages 427–435, 2013.

[5] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In Advances in Neural Information Processing Systems (NIPS), volume 28, pages 496–504, 2015.

[6] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In AAAI, 2015.