University of Padova
SCP7079229 - Optimization for Data Science 2023-2024

Optimization Methods for Clustering: Frank-Wolfe, Pairwise Frank-Wolfe, Away Step Frank-Wolfe algorithms

Group 14:
Dinara Kurmangaliyeva (ID: 2106044)
Akbota Norzhanova (ID: 2106050)

Table of Contents

Introduction

In this project, we implemented three algorithms: classic Frank-Wolfe algorithm and two of its adaptations, Pairwise Frank-Wolfe and Away Step Frank-Wolfe algorithm. We analyzed the performance of three Frank-Wolfe algorithms in L^2 regularized max-clique problem, comparing their resulting max-clique sizes with best-known solutions and comparing their convergence rates with respect to iterations and CPU-time.

Literature overview

The Frank-Wolfe method, also known as conditional gradient method or reduced gradient method, is an iterative first-order optimization algorithm originally proposed by Marguerite Frank and Philip Wolfe in 1956 [1] to solve quadratic programming problems with linear constraints. For our implementation of this algorithm and its variants, we reviewed and analyzed the relevant papers, which are discussed in the sections below.

Main papers

Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization (Jaggi, 2013) [2]

This paper provides an enhanced analysis of the traditional Frank-Wolfe algorithm. Jaggi introduces a primal-dual framework that improves convergence guarantees enabled by duality gap certificates, even with approximate linear subproblem solutions or inexact gradients. This work unifies various sparse greedy methods under the Frank-Wolfe paradigm, applicable to optimization over convex hulls of atomic sets like sparse vectors, low-rank matrices, and permutation matrices. This paper also includes new convergence analysis showing the algorithm achieves a small duality gap, optimizing the sparsity of solutions. The algorithm is generalized to handle approximate linear subproblem solutions and inexact gradients. Additionally, the Frank-Wolfe algorithm is applied to broader optimization problems, such as matrix factorization and trace norm minimization, with each iteration framed as a low-rank update.

On the Global Linear Convergence of Frank-Wolfe Optimization Variants (Lacoste-Julien, Jaggi, 2015) [3]

The paper by Jaggi and Lacoste-Julien advances the Frank-Wolfe algorithm by focusing on improved variants and their convergence properties. This paper also introduces Away-Step Frank-Wolfe (AFW) and Pairwise Frank-Wolfe (PFW) algorithms, which we used in implementation of our project. The Away-Steps Frank-Wolfe algorithm utilizes “away steps”, removing weight from previously selected atoms to mitigate the zigzagging effect, ensuring global linear convergence for strongly convex functions. The Pairwise Frank-Wolfe algorithm improves

convergence by adjusting the weights of two atoms simultaneously, further reducing the zigzagging effect.

Additionally, Jaggi and Lacoste-Julien also demonstrate that AFW, PFW, and two other Frank-Wolfe variants, Fully Corrective Frank-Wolfe and Min-Norm Point algorithms, achieve global linear convergence under conditions weaker than strong convexity. They introduce the concept of pyramidal width, a geometric measure crucial for determining convergence rates.

Frank–Wolfe and Friends: A Journey into Projection-Free First-Order Optimization Methods (Bomze et al., 2017) [4]

This paper by Bomze, Rinaldi, and Zeffiro provides an extensive review of the Frank-Wolfe algorithm and its variants, highlighting their theoretical foundations, practical applications, and recent advancements. Empirical results confirm that AFW, and PFW significantly outperform the classic Frank-Wolfe algorithm in terms of convergence speed and computational efficiency. It also provides insights on step size selection methods.

Supplementary papers

The paper titled "A General Regularized Continuous Formulation for the Maximum Clique Problem" by Hungerford and Rinaldi [5] introduces a new optimization framework for solving the Maximum Clique Problem (MCP). This framework is based on a regularization approach applied to the classic Motzkin-Straus formulation of the MCP. The authors focus on including a broad class of regularization terms to ensure the equivalence between the continuous regularized problem and the original MCP in both global and local contexts. They provide a detailed analysis of two specific regularizers and establish conditions that guarantee this equivalence.

Max-clique problem

Objective function and gradient definition

Given a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges, the maximum clique problem aims to identify the largest subset, or clique, $C \subseteq V$ such that every two distinct vertices in C are adjacent. In other words, for all $u, v \in C$, $(u, v) \in E$.

We use formulation of max-clique problem by Bomze [6] described in [5]:

$$\begin{aligned} &\max\ x^T A x + \alpha \|x\|_2^2 \\ &\text{subject to } x \in \Delta, \end{aligned}$$

where Δ is the n-dimensional simplex, A is the adjacency matrix for G, and x is the vector that signifies which vertices are included in the clique. The second term is a regularization term with $\alpha = \frac{1}{2}$.

By multiplying the objective function to -1 and substituting α, we attain the following minimization problem:

$$f(x) = -x^T Ax - \frac{1}{2}\|x\|_2^2$$

To navigate towards the optimal solution, our algorithms rely on the gradient of the objective function, which provides direction for the next step. The gradient is calculated as:

$$\nabla f(x) = -2Ax - x$$

Hence, this gradient vector will guide the algorithms' progression, optimizing towards potential maximal clique configurations by balancing the structural constraints of the graph and the regularization term.

Implementation of the algorithms

The Frank-Wolfe algorithm, also known as the conditional gradient method, is a widely used optimization technique for solving constrained convex optimization problems where the objective function is differentiable and the constraint set is convex. The algorithm minimizes a convex function $f(x)$ over a convex set C: $\min_{x \in C} f(x)$ where $f(x)$ is differentiable. The Frank-Wolfe algorithm iteratively updates an initial point $x^{(0)}$ by moving towards the direction of the negative gradient of the function at the current point, subject to a linear optimization problem over the constraint set.

Standard Frank-Wolfe algorithm

At each iteration t, the algorithm computes the gradient $\nabla f(x^{(t)})$ of the objective function at the current iterate $x^{(t)}$. Instead of moving directly towards the gradient descent direction as in traditional methods, the Standard Frank-Wolfe algorithm updates $x^{(t)}$ by moving towards a point s_t that minimizes the linearization of the objective function $f(x)$ over C: $s_t = arg\min_{s \in C} \langle \nabla f(x^{(t)}), s \rangle$. This step is known as the Linear Minimization Oracle (LMO).

As we are optimizing a max-clique problem and it is subject to $x \in \Delta$, our LMO will find point s_t by minimizing the following function: $s_t = arg\min_{s \in \Delta} (-2Ax - x)^T s$. Since s

is a vertex in simplex Δ, it can be one of the standard basis vectors e_i. Therefore, the problem can be reduced to finding the vertex that minimizes the dot product with the gradient.

The algorithm then updates the current solution $x^{(t)}$ using a step size γ_t:

$$x^{(t+1)} = x^{(t)} + \gamma_t(s_t - x^{(t)}).$$

The general schema of the classic Frank-Wolfe algorithm we implemented is reported in Algorithm 1:

Algorithm 1 Standard Frank-Wolfe: **`frank_wolfe`**($\mathcal{A}$, $x^{(0)}$, T, ϵ)

1: Let $x^{(0)} \in \mathcal{A}$
2: **for** $t = 0, \dots, T$ **do**
3: Let $g_t \coloneqq \nabla f\left(x^{(t)}\right)$
4: Let $s_t \coloneqq \mathrm{LMO}_{\mathcal{A}}(g_t)$
5: Let $\mathrm{d}_t^{FW} \coloneqq s_t - x^{(t)}$
6: Update $x^{(t+1)} = x^{(t)} + \gamma_t \mathrm{d}_t^{FW}$ (*with* $\gamma_t \in (0, 1]$ *suitably chosen stepsize*)
7: **If** $\|\mathrm{d}_t^{FW}\| < \epsilon$ **then return** $x^{(t+1)}$
8: **end for**

Away Steps Frank-Wolfe algorithm

The away steps variant of the Frank-Wolfe algorithm enhances the classic algorithm by incorporating additional steps to improve convergence and mitigate zigzagging.

Before diving into the algorithm of the AFW algorithm it is important to define the active set. The iterate $x^{(t)}$ can be represented as a sparse convex combination of at most $t + 1$ atoms $S^{(t)} \subseteq C$, which we write as:

$$x^{(t)} = \sum_{v \in S^{(t)}} \alpha_v^{(t)} v$$

The active set $S^{(t)}$contains the previously discovered search atoms s_r for $r < t$ that have non-zero weight $\alpha_{s_r}^{(t)} > 0$ in the expansion. It is important to notice that it is not necessary to track the active set $S^{(t)}$ in the classic Frank-Wolfe algorithm, but it is required for the improved variants of the algorithm to maintain $S^{(t)}$.

In each iteration t, besides computing the Frank-Wolfe (FW) direction as in classic algorithm, the AFW algorithm also identifies the away direction by finding the worst contributing atom from the active set $S^{(t)}$. The algorithm then selects between FW direction and the away direction based on a certain condition. If the FW direction is chosen, the parameter $\gamma_{max}$is set to 1. In contrast, if the away direction is chosen, the parameter $\gamma_{max}$is set to to $\alpha_{v_t}/(1 - \alpha_{v_t})$, where $\alpha_{v_t}$is the coefficient of the atom v_t in

the active set. The parameter γ_{max} is then used to calculate the step size γ_t in line-search procedure. Then, the current solution $x^{(t)}$ is updated using the calculated step size γ_t, following the same update rule as in the classic Frank-Wolfe algorithm. Finally, the active set $S^{(t)}$ is updated depending on the chosen direction d.

The general schema of the AFW algorithm is described in Algorithm 2:

Algorithm 2 Away Step Frank-Wolfe: **`away_step_frank_wolfe`**($\mathcal{A},\ x^{(0)}, \mathrm{T},\ \epsilon$)

Let $x^{(0)} \in \mathcal{A}$
Let $S^{(0)} \coloneqq \{x^{(0)}\}$
for $t = 0, \dots, T$ **do**
    Let $g_t \coloneqq \nabla f\left(x^{(t)}\right)$
    Let $s_t \coloneqq \mathrm{LMO}_{\mathcal{A}}(g_t)$
    Let $\mathrm{d}_t^{FW} \coloneqq s_t - x^{(t)}$
    Let $v_t \in \arg\max_{v \in S^{(t)}} \langle g_t,\ v \rangle$
    Let $\mathrm{d}_t^{A} \coloneqq x^{(t)} - v_t$
    if $\langle -g_t,\ \mathrm{d}_t^{FW} \rangle \leq \epsilon$ **then return** $x^{(t+1)}$
    if $\langle -g_t, \mathrm{d}_t^{FW} \rangle \geq \langle -g_t, \mathrm{d}_t^{A} \rangle$ **then**
        $d_t \coloneqq \mathrm{d}_t^{FW}$, and $\gamma_{\max} \coloneqq 1$
    else
        $d_t \coloneqq \mathrm{d}_t^{A}$, and $\gamma_{\max} \coloneqq \alpha_{v_t}/(1 - \alpha_{v_t})$
    end if
    Select γ_t using Armijo rule or Exact line search with bound $[0, \gamma_{\max}]$
    Update $x^{(t+1)} \coloneqq x^{(t)} + \gamma_t d_t$
    Update $\alpha_v^{(t+1)}$ for $v \in S^{(t)}$ depending on choice of d_t
    Update $S^{(t+1)} \coloneqq \{v \in \mathcal{A} \text{ s.t. } \alpha_v^{(t+1)} > 0\}$
end for

Pairwise Frank-Wolfe algorithm

Pairwise Frank-Wolfe algorithm also utilizes an active set maintained at each iteration, as described in AFW algorithm. However, in PFW, the weight is transferred between two atoms only: from v_t to s_t, while other weights remain unchanged. This step is called a pairwise FW step. In contrast, the classic FW algorithm shrinks all active weights at each iteration. Additionally, the PFW direction is determined by selecting s_t, which minimizes the linearization of the objective function $f(x)$ over C, and v_t, the worst contributing atom from the active set. The current solution $x^{(t)}$ is then updated using the calculated step size γ_t, following the same update rule as in the classic FW algorithm.

The general outline of PFW is provided in Algorithm 3:

Algorithm 3 Pairwise Frank-Wolfe: **`pairwise_frank_wolfe`**$(\mathcal{A},\ x^{(0)}, \mathrm{T},\ \epsilon)$

Let $x^{(0)} \in \mathcal{A}$
Let $S^{(0)} \coloneqq \{x^{(0)}\}$
for $t = 0, \ldots, T$ **do**
    Let $g_t \coloneqq \nabla f\left(x^{(t)}\right)$
    Let $s_t \coloneqq \mathrm{LMO}_{\mathcal{A}}(g_t)$
    Let $\mathrm{d}_t^{FW} \coloneqq s_t - x^{(t)}$
    Let $v_t \in \arg\max_{v \in S^{(t)}} \langle g_t,\ v \rangle$
    Let $\mathrm{d}_t^{A} \coloneqq x^{(t)} - v_t$
    if $\langle -g_t,\ \mathrm{d}_t^{FW} \rangle \leq \epsilon$ **then return** $x^{(t+1)}$
    Let $d_t = \mathrm{d}_t^{PFW} \coloneqq s_t - v_t$
    Let $\gamma_{\max} \coloneqq \alpha_{v_t}$
    Select γ_t using Armijo rule or Exact line search with bound $[0, \gamma_{\max}]$
    Update $x^{(t+1)} \coloneqq x^{(t)} + \gamma_t d_t$
    Update $\alpha_{v_t}^{(t+1)}$ and $\alpha_{s_t}^{(t+1)}$
    Update $S^{(t+1)} \coloneqq \{v \in \mathcal{A} \text{ s.t. } \alpha_v^{(t+1)} > 0\}$
end for

Step sizes

In our implementation of the Frank-Wolfe algorithm, we explored various step size methods to optimize convergence and performance. Below, we discuss the step size methods used in the project and the rationale behind their selection.

Diminishing Step Size

The simplest approach involves using a constant step size, typically defined as $\gamma_t = 2/(t+2)$. This method is straightforward to implement and ensures that the step size decreases over time, which is essential for convergence in many convex optimization problems. In our project, we used the following approach in the classic Frank-Wolfe algorithm, as described by Bomze and Rinaldi [4].

Exact Line Search

Additionally, we have integrated the exact line search method to determine the optimal step size during each iteration. Exact line search computes the optimal step size γ by precisely minimizing the objective function $f(x + \gamma d)$ over the interval $[0, \gamma_{max}]$. This method offers precision in step size determination, improved convergence rates, and stability, all while being easy to implement.

Armijo rule

Given the fact that exact line search can be computationally expensive, we also decided to apply Armijo rule for line search. Unlike exact line search, Armijo rule uses an iterative approach to find a suitable step size with fixed parameters $\delta \in (0,1)$ and $\omega \in (0, \frac{1}{2})$. This method ensures that the step size is chosen to satisfy the Armijo condition:

$$f(x^{(t)} + \gamma d_t) \leq f(x^{(t)}) + \omega\gamma\nabla f(x^{(t)})^T d_t,$$

where f is the objective function to be minimized, x is the current point in the optimization space, d is the search direction, $\nabla f(x)$ is the gradient of the objective function at the point x. The method tries steps $\gamma = \delta^m \gamma_{max}$ with $m \in \{0,1,2,\dots\}$, until the sufficient decrease in inequality above holds. In our experiments we used parameters $\delta = 0.7$ and $\omega = 0.1$, which ensured good performance.

Datasets

For this project, we conducted experiments using instances from the DIMACS benchmark [7]. We chose three databases from distinct families to evaluate performance of our algorithms on various instances. Specifically, we selected the datasets C125.9, DSJC500_5, and keller6.

The C125.9 dataset, part of the C family, is a randomly generated dataset consisting of 125 nodes and 6,963 edges. The DSJC500_5 dataset belongs to DSJC family of randomly generated datasets, containing 500 nodes and 125,248 edges. The keller6 is based on Keller's conjecture on tilings using hypercubes, featuring 3,361 nodes and 4,619,898 edges, and is part of the Keller family.

Overall, we are utilizing three datasets with a progressively increasing number of nodes and edges. This approach allows us to assess the efficiency of our algorithms as the complexity of the datasets increases.

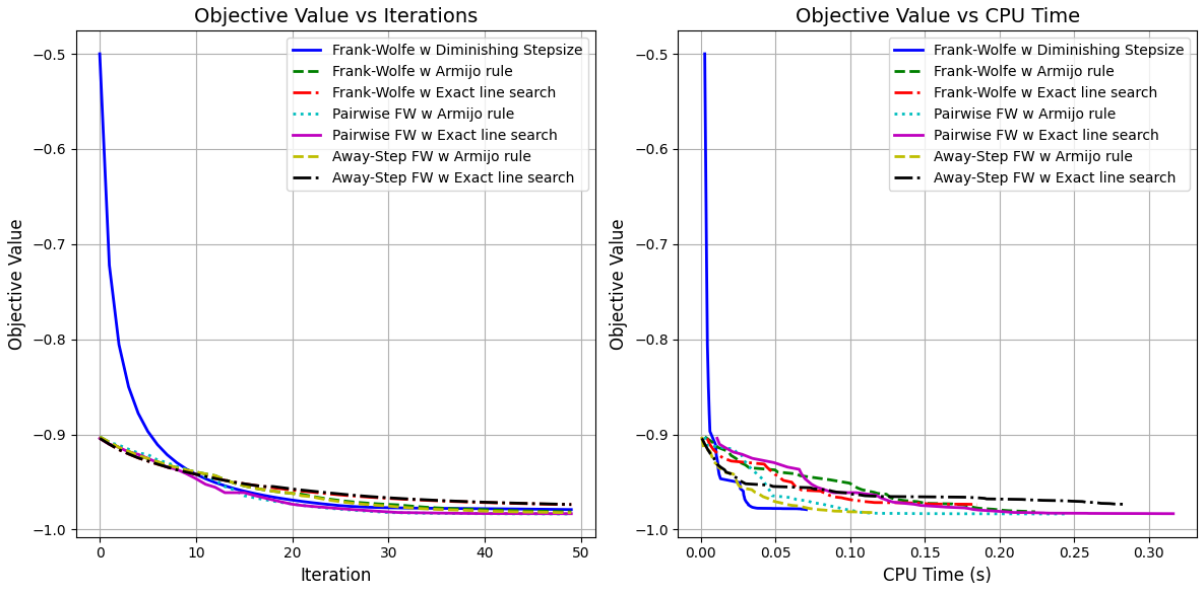
Results

We executed each algorithm for 50 iterations with a tolerance (ϵ) of $1.0 * 10^{-6}$. The maximum clique results of each algorithm and best known results are shown in the table below:

Algorithm	C125.9	DSJC500_5	Keller6
Frank-Wolfe with diminishing step size	26	10	31
Frank-Wolfe with Armijo rule	30	10	31
Frank-Wolfe with exact line search	30	10	31
Pairwise Frank-Wolfe with Armijo rule	31	10	33
Pairwise Frank-Wolfe with exact line search	30	9	33
Away Step Frank-Wolfe with Armijo rule	30	10	31
Away Step Frank-Wolfe with exact line search	30	10	31
Best known result	**34**	**13**	**59**

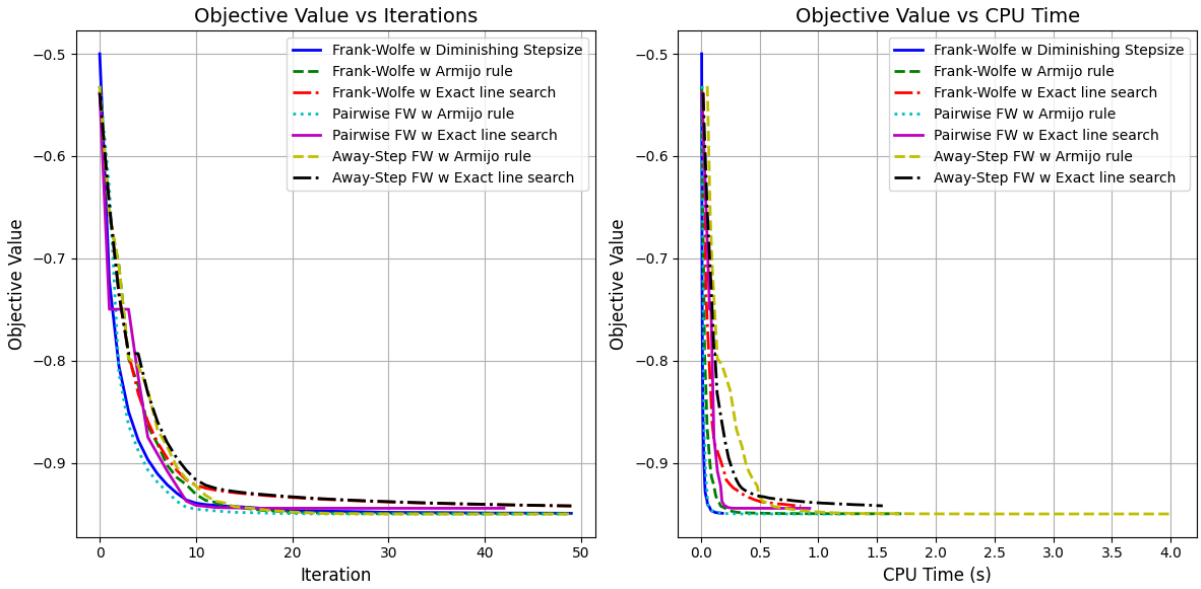
For each dataset, we present graphs that display the performance of each algorithm. These graphs compare the objective function value across iterations and CPU time.

C125.9 dataset



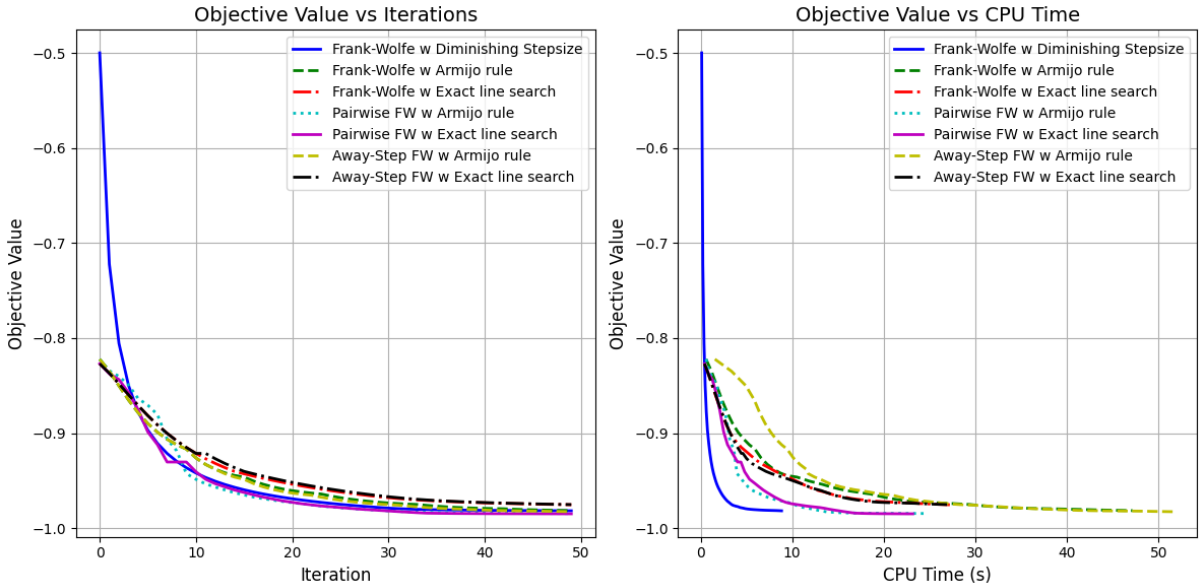
It can be noted that each algorithm reaches approximately the same objective value within each iteration, except for FW with diminishing step size. This indicates that while the computational efforts differ, the end objective value obtained by all the variants is similar. Unlike for FW with diminishing step size: the algorithm starts with a higher objective value and shows a rapid decrease in the first few iterations. FW with Diminishing Step Size, while simple, is less efficient compared to the other variants, requiring more iterations to converge (achieving similar objective value only after 10 iterations) and achieving the least number of max-cliques, being farthest away from the best known result. However, if minimizing CPU time is the priority, then AFW with Armijo rule might be a good choice.

DSJC500_5 dataset



For the DSJC500_5 instances, FW with diminishing step size and PFW with Armijo rule demonstrate the best performance in terms of the number of iterations, rapidly converging to lower objective values and producing a number of max-cliques similar to other algorithms. Therefore, FW with diminishing step size and PFW with Armijo rule achieve these values with fewer iterations, as well as CPU time, highlighting their efficiency.

Keller6 dataset



Again, all algorithms reach similar convergence over iterations, except for FW with diminishing step size. PFW with Armijo rule and exact line search also achieve the

largest max-clique size, compared to other algorithms. Although their results are not very close to the best known result of 59, their performance is similar to results in [5]. FW with diminishing step size requires more iterations to reach the objective value of other algorithms, but it requires less CPU time due to its simplicity. Both PFW algorithms are the next most efficient methods in terms of CPU time, while FW and AFW with Armijo rule are the least efficient in terms of CPU time, taking longer to achieve similar reductions in objective value.

Discussion

Despite differences in convergence rates and computational requirements, all algorithms reached similar objective values.

In our experiments we see that the PFW and AFW algorithms consistently outperform classic FW algorithms (except for diminishing step size in terms of CPU time) on max-clique problem. In terms of max-clique sizes, PFW with Armijo rule constantly produced the highest results in each of the datasets.

Conclusion

In this project, we implemented and compared the classical Frank-Wolfe algorithm and its two advanced variants, Pairwise Frank-Wolfe and Away-Step Frank-Wolfe, focusing on their performance in solving the L^2 regularized max-clique problem. Our findings indicate that while all three algorithms eventually reached similar objective values, there are notable differences in their efficiency and convergence behavior. The Pairwise Frank-Wolfe algorithm, particularly when combined with the Armijo rule for step size determination, demonstrated superior performance in terms of iterations and CPU time, and max-clique sizes across the datasets. This variant's ability to adjust weights between pairs of atoms effectively mitigates zigzagging and enhances convergence speed.

The Away-Step Frank-Wolfe algorithm also improved convergence by introducing away steps, which remove weight from previously selected atoms, thereby reducing the zigzagging effect and ensuring global linear convergence for strongly convex functions. This improvement, however, came at the cost of increased complexity in maintaining the active set.

Overall, our experiments confirmed that the advanced Frank-Wolfe variants, AFW and PFW, significantly outperform the classical FW algorithm in solving the max-clique problem.

Bibliography

[1] Frank, M., & Wolfe, P. (1956). "An algorithm for quadratic programming." *Naval Research Logistics Quarterly*, 3(1-2), 95-110. DOI: 10.1002/nav.3800030109

[2] Jaggi, M. (2013). "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization." In *Proceedings of the 30th International Conference on Machine Learning* (ICML-13), 427-435. URL: http://proceedings.mlr.press/v28/jaggi13.html

[3] Lacoste-Julien, S., & Jaggi, M. (2015). "On the Global Linear Convergence of Frank-Wolfe Optimization Variants." In *Advances in Neural Information Processing Systems* (NIPS 2015), 496-504. URL: https://papers.nips.cc/paper/2015/file/981a48cadea1f44684cd9b3f6c8109ab-Paper.pdf

[4] Bomze, I., Rinaldi, F., & Zeffiro, D. (2017). "Frank–Wolfe and Friends: A Journey into Projection-Free First-Order Optimization Methods." *Mathematical Programming*, 165(1), 233-292. DOI: 10.1007/s10107-016-0995-8

[5] Hungerford, K., & Rinaldi, F. (2019). "A General Regularized Continuous Formulation for the Maximum Clique Problem." *Mathematical Programming*, 174(1-2), 1-28. DOI: 10.1007/s10107-018-1300-3

[6] Bomze, I.M. (1997). "Evolution towards the Maximum Clique." *Journal of Global Optimization,* 10, 143–164. DOI: 10.1023/A:1008230200610

[7] DIMACS Benchmark Graphs. Center for Discrete Mathematics and Theoretical Computer Science. URL: http://dimacs.rutgers.edu/Challenges/