Frank Wolfe Variants for Cluster Detection on Hypergraphs ODS Project Report

Anna Badalyan[†]

***Abstract*—The project is aimed at analysing the performance of first order Frank-Wolfe (FW) variants, namely: *Classic FW*, *Away-Step FW*, *Pairwise FW* and *Blended Pairwise FW* in solving continuous formulation of the maximum clique problem on k-uniform hypergraphs. We show that the implemented variants outperfrom classic FW and show linear rate of convergence. In addition, we implement Short Step Chain (SSC) procedure which eliminates "bad" steps and further improves the convergence rates of FW variants.**

***Index Terms*—Frank Wolfe, Hypergraphs, Maximum clique**

I. Introduction

In this report we discuss the implementation of various variants of Frank-Wolfe (FW) algorithm to solve the maximum clique problem on k-uniform graphs. The goal is to compare the performance of FW variants in solving the given optimization problem.

The project is based on the following papers: [1] presents the classic FW algorithm and outlines a strong primal-dual convergence results. [2] shows linear convergence for Away-Step FW, Pairwise FW, Fully corrective FW and Wolfe's minimum norm point algorithm. The conditions of linear convergence are weaker than strong convexity of the objective function. [3] introduced a modification to the Pairwise FW which removes "bad" swap steps and improves the convergence rates up to a constant factor. [4] further improves the convergence rates by preventing "bad" steps in FW variants with the introduction of Short Step Chain (SSC) procedure.

The report is structured as follows: section II describes the maximum cluqie optimization problem and its continuous formulation in case of hypergraphs, section III gives a theoretical background on the various FW variants, section IV discusses the practical implementation of the FW variants and section V presents results of the numerical experiments.

II. Optimization problem

Maximum click problem for graphs is generally considered a combinatorial optimization problem; however, a connection between a global maxima of the Lagrangian of a graph and the maximum click number established by Motzkin and Straus helps to develop a continuous formulation of the problem. Rota Bulò and Pellilo [5] showed a generalization of the Motzkin-Straus theorem for k-uniform hypergraphs.

[†]Department of Mathematics, University of Padova, email: anna.badalyan@studenti.unipd.it

The Lagrangian of a k-graph with n vertices is defined as follows:

$$L_G(x) = \sum_{e \in E} \prod_{i \in e} x_i \tag{1}$$

Motzkin and Straus showed the connection between the maxima of the Lagrangian of the graph with n vertices over the standard simplex in $\mathbb{R}^n$ and clique number of the graph. x is a characteristic vector if it represents a subset S of vertices over a standard simplex

$$x_i^S = \frac{1_{i \in S}}{|S|}$$

Rota Bulò and Pellilo [5] propose the following continuous formulation of a maximual clique problem in k-graphs:

$$\begin{aligned} &\min \ \ h_{\overline{G}}(x) = L_{\overline{G}}(x) + \tau \sum_{i=1}^{n} x_i^k \\ &\text{subject to } x \in \Delta \end{aligned} \tag{2}$$

where $\tau \in \mathbb{R}$, $L_{\overline{G}}(x)$ is the Lagrangian of the complement of G and Δ is the standard simplex defined as

$$\Delta = \left\{ x \in \mathbb{R}^n : \forall i = 1...n, x_i \leq 0 \text{ and } \sum_{i=1}^{n} x_i = 1 \right\}$$

We can see that this is a constrained optimization problem over the standard simplex which can be efficiently solved by Frank-Wolfe algorithms.

III. Background

One of the simplest iterative first order optimization algorithms to solve the given problem is Frank Wolfe (FW). In this section we discuss theory related to FW algorithms. The algorithms are designed to solve the following type of optimization problems:

$$\min_{x \in \mathcal{D}} f(x), \tag{3}$$

where $\mathcal{D} = conv(\mathcal{A})$ is convex compact set, $\mathcal{A}$ is a finite set of vectors called atoms, and f is a convex, continuously differentiable function with Lipschitz continuous gradient $L > 0$.

At each iteration the FW algorithm minimizes the linear approximation of the objective function at the current point x_k

$$\min_{x \in \mathcal{D}} f(x_k) + \nabla f(x_k)^\top (x - x_k). \tag{4}$$

A. Definitions

The convergence results of FW algorithms in [1], [2] rely on several important concepts and we give the definitions here.

Primal-dual convergence analysis shown in [1] depends on the curvature constant C_f which measures the "non-linearity" of the objective function over the domain $\mathcal{D}$ defined as:

$$C_f := \sup_{\substack{x,s\in\mathcal{D},\\ \gamma\in[0,1],\\ y=x+\gamma(s-x)}} \frac{2}{\gamma^2}\left(f(y)-f(x)-\langle y-x,\nabla f(x)\rangle\right) \quad (5)$$

The linear convergence proofs for AFW and PFW in [2] depend on the *pyramidal width*. To define pyramidal width, [2] first give the definition of *pyramidal directional width* as:

$$PdirW(\mathcal{A},r,x)=\min_{\mathcal{S}\in\mathcal{S}_x}\max_{s\in\mathcal{A},v\in\mathcal{S}}\langle\frac{r}{\|r\|},s-v\rangle, \quad (6)$$

where $r := -\nabla f(x)$ is a negative gradient, $x\in\mathcal{D}$ a base point, $\mathcal{S}$ is a possible active set for x and $\mathcal{S}_x := \{\mathcal{S}|\mathcal{S}\subseteq\mathcal{A}\}$ such that x is a convex combination of all the elements in $\mathcal{S}$ with all coefficients being non-zero.

The pyramidal width of a set is then defined as the smallest pyramidal width of all its faces, that is:

$$PWidth(\mathcal{A}) := \min_{\substack{\mathcal{K}\in\text{faces}(\text{conv}(\mathcal{A}))\\ x\in\mathcal{K}\\ r\in\text{cone}(\mathcal{K}-x)\backslash\{0\}}} PdirW(\mathcal{K}\cup\mathcal{A},r,x) \quad (7)$$

For affine invariant formulation of convergence proofs, a slightly modified curvature constant C_f^A and a geometric strong convexity constant μ_f^A are introduced in [2]. C_f^A is a modification of curvature constant for any direction $s-v$ instead of the standard FW direction. It is defined as:

$$C_f^A := \sup_{\substack{x,s\in\mathcal{D},\\ \gamma\in[0,1],\\ y=x+\gamma(s-v)}} \frac{2}{\gamma^2}\left(f(y)-f(x)-\langle s-v,\nabla f(x)\rangle\right)$$

μ_f^A is a geometric strong convexity constant that depends not only on function f but also on domain $\mathcal{D}$. It is defined as:

$$\mu_f^A := \inf_{x\in\mathcal{M}} \inf_{\substack{x^*\in\mathcal{M},\text{s.t.}\\ \langle\nabla f(x),x^*-x\rangle<0}} \frac{2}{\gamma^A(x,x^*)^2}\left(f(x^*)-f(x)-\langle\nabla f(x),x^*-x\rangle\right), \quad (8)$$

B. Classic Frank Wolfe algorithm

The classic FW $\hat{x}_k$ finds a new descent direction as follows:

$$d_k = \hat{x}_k - x_k$$

and the new point becomes

$$x_{k+1} = x_k + \alpha_k d_k$$

with $\alpha\in(0,1]$ being a suitably chosen stepsize.

Thus, each point is a convex combination of previously chosen points, which means that the solution remains in the feasible region.

To measure the quality of the solution duality gap certificates are used. The gap is defined as follows:

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

By the weak duality result, in order to have the primal error less than ϵ it would be enough to have the $g(x)$ less than ϵ

$$f(x)-f(x*)<g(x)<\epsilon.$$

For the stepsize $\alpha=\frac{2}{k+1}$ the convergence rate is known to be $\mathcal{O}(\frac{1}{k})$ [6].

Theorem 1 in [1] states that for $k\geq 1$ iterates $x^{(k)}$ of the classic Frank Wolfe algorithm satisfy

$$f(x^{(k)})-f(x^*)\leq\frac{2C_f}{k+2}(1+\nu), \quad (9)$$

where $x^*\in\mathcal{D}$ is the optimal solution to the optimization problem, $\nu\geq 0$ accuracy of the solutions to the linear subproblems.

If the FW algorithm runs for $K\geq 2$ iterations Theorem 2 in Jaggi [1] states that there is an iterate $x^{(\hat{k})}, 1\leq\hat{k}\leq K$, with the duality gap bounded as:

$$g(x^{(\hat{k})})\leq\frac{2\beta C_f}{K+2}(1+\nu), \quad (10)$$

where $\beta=\frac{27}{8}$ and $\nu\geq 0$ accuracy of the solutions to the linear subproblems.

The primal-dual convergence results showed that the duality gap becomes small after many iterations even if the linear subproblems are solved only approximately.

The author also showed that the Frank Wolfe algorithm is invariant under affine transformations.

Frank Wolfe method is particularly suitable for continuous relaxations of NP hard combinatorial problems over the convex hull of atomic sets as the method can induce structured sparsity [1]. Problem 2 is of this type, therefore, FW algorithms are particularly suitable for it.

The main problem with the classic FW algorithm is the "zig-zaging" phenomenon as the convergence rate becomes sublinear when the optimal solution lies at the boundary of the domain $\mathcal{D}$. Thus, the algorithms starts to zig-zag between the vertices containing the optimal solution [2]. To address this problem several improvements to the classic FW algorithm have been introduced; Away-Step Frank-Wolfe, Pairwise Frank-Wolfe and Blended Pairwise Frank-Wolfe will be further discussed in this report.

C. Away-Step Frank-Wolfe (AFW)

The principal idea is to not only add new atoms at each iteration of the algorithm, but also remove the atoms that are bad. This helps to mitigate the "zig-zaging" problem. Removing "bad" atoms from the active set $S^{(t)}$ helps to reach linear convergence in case of strongly convex functions [2].

At each iteration in addition to the global FW direction, the algorithm computes the away step direction over the active set. The active set is defined as the set of vertices which form the convex combination of the current iterate. Maximizing over the active set gives us the direction which points away from the worst point in the active set. Then the algorithms chooses the best of the two directions that minimizes the linear approximation 2.

D. Pairwise Frank Wolfe (PFW)

The main idea behind this variant of FW is that we move the weight from the away atom to the global FW atom. That is, at each iteration only the weights of 2 atoms are swapped. In comparison, the classic FW shrinks the weights of all active atoms at each iteration. [2]

The authors in [2] proved a linear convergence rate for both AFW and PFW under weaker conditions than the strong convexity of the objective function, however, the convergence for PFW is loose because of the large possible number of "bad" swap steps, which is bound by $|\mathcal{A}|!$.

The convergence was prooved by first showing that the primal gap of these variants decreases geometrically at each step if the step is not a drop step for AFW or a swap step for PFW. Letting $h_k = f(x^{(k)}) - f(x^*)$ be the primal gap, we have for a μ strongly convex function over $\mathcal{D}$:

$$h_{k+1} \leq (1-\rho)h_k \quad \text{where} \quad \rho := \frac{\mu}{4L}\left(\frac{\delta}{D}\right)^2,$$

$D := \sup_{x,y\in\mathcal{D}}\|x-y\|$ is the diameter of the constrain set and $\delta = PWidth(\mathcal{A})$ is a pyramidal width.

Letting $n(k)$ be the number of good steps at the iteration k, we can lower bound it as $n(k) \geq k/2$ for AFW and $n(k) \geq k/(3|\mathcal{D}|!+1)$ for PFW. This gives the following linear convergence rate for all variants:

$$f(x^{(k)}) - f(x^*) \leq h_0 exp(-\rho n(k)) \tag{11}$$

The affine invariant version of the proof shows the following bounds:

$$h_{k+1} \leq (1-\rho)h_k$$

where

$$\rho := \frac{\mu_f^A}{4C_f^A} \quad \text{for AFW}$$

$$\rho := \min\left\{\frac{1}{2}, \frac{\mu_f^A}{C_f^A}\right\} \quad \text{for PFW}$$

where $\gamma^A(x,x^*)$ is a positive step-size quantity defined as:

$$\gamma^A(x,x^*) := \frac{\langle -\nabla f(x), x^* - x\rangle}{\langle -\nabla f(x), s_f(x) - v_f(x)\rangle},$$

with $s_f(x)$ standing for the standard FW atom, and $v_f(x)$ for the worst case away atom.

Thus, the suboptimality bounds for AFW and PFW are as follows:

$$f(x^{(k)}) - f(x^*) \leq \frac{4C}{n(k)+4} \quad \text{for} \quad n(k) \geq 1, \tag{12}$$

where $C = 2\mu_f^A + h_0$ for AFW, $C = C_f^A/2$ for PFW

E. Pairwise Frank Wolfe without swap steps (BPFW)

To eliminate the "bad" swap steps that don't guarantee sufficient progress in the PFW, Blended Pairwise Frank-Wolfe (BPFW) algorithm was presented in [3] . The algorithm checks if the local gap is larger than a global gap and performs the local pairwise update. Otherwise, the algorithm does a global FW step. This simple modification to PFW, ensures that the swap steps don't occur, thus improving the convergence by showing the same rates as PFW without swap steps.

The convergence rate of BPFW was shown to be:

$$f(x^{(k)}) - f(x^*) \leq \frac{4LD^2}{k}, \tag{13}$$

where $D := \sup_{x,y\in\mathcal{D}}\|x-y\|$ is the diameter of the constrain set.

F. Short Step Chain (SSC) procedure

To eliminate the distinction between "bad" steps and "good" steps in FW variants, Short Step Chain (SSC) procedure was introduced in [4]. The procedure performs steps of FW variants with the same value of the gradient until some conditions are satisfied. This particularly help to improve convergence speed when the gradient calculation is costly.

It was shown that the procedure gives improvements on previously mentioned convergence rates.

IV. Implementation

We implemented the following variants of the Frank Wolfe algorithms:

1) Classic Frank Wolfe algorithm, the algorithm 1 in [1]
2) Away Step Frank Wolfe algorithm, the algorithm 1 in [2]
3) Pairwise Frank Wolfe algorithm, the algorithm 2 in [2]
4) Blended Pairwise Conditional Gradient, the algorithm 1 in [3]

In this section we further discuss some of the algorithmic choices that were made.

A. Starting Point

The starting point was initialized at random. Each atom in the unit simplex was given a weight uniformly at random between (0, 1) with the weights being normalized afterwards.

B. LMO

In the case of the standard simplex, the solution to the linear minimization or maximization subproblems for FW variants is straightforward as it corresponds to the atom with the smallest or largest value of the gradient.

C. Stepsize

The following options for the stepsize strategy were implemented:

1) Fixed decreasing stepsize defined as:

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

where k is the number of iterations. The drawback of this stepsize strategy is that it can only be implemented with classic Frank Wolfe variant.

2) Armijo line search as defined in [7]. The stepsize γ is defined as follows:

$$\gamma = \delta^m \gamma_{max}$$

We start with $m = 0, 1, 2, ...$ until the inequality

$$f(x_k + \gamma d_k) \leq f(x_k) + \gamma\alpha\nabla f(x_k)^\top d_k$$

is satisfied. γ_{max} is the maximum allowed stepsize, $\delta \in (0, 1)$ and $\alpha \in (0, \frac{1}{2})$ are the pre-defined parameters. We chose $\delta = 0.7$ and $\alpha = 0.1$ for the experiments.

3) Decreasing Armijo line search. The procedure is the same as the classic line search, but but the value of γ_{max} is defined as the minimum between the largest allowed γ and the γ at previous iteration. This strategy was implemented to speed up the convergence of classic Frank Wolfe algorithm. As expected in increased the number of iterations required to converge, but significantly reduced the CPU time. We didn't continue with experiments with this step size though as it would make the comparison with other variants unfair.

4) Backtracking line search as presented in [8]. This procedure also allows to estimate the Lipschitz constant L. The gamma is estimated as follows:

$$\gamma = min\left\{\frac{g_k}{L\|d_k\|^2}, \gamma_{max}\right\}$$

where g_k is the gradient at point x_k, d_k descent direction, L the current estimate for the Lipschits constant. The value of L is increased by parameter $\tau > 1$ until the following condition is satisfied:

$$f(x_k + \gamma d_k) \leq f(x_k) - \gamma g_k + \frac{\gamma^2 L}{2}\|d_k\|^2 \qquad (14)$$

We run the experiments with parameter $\tau = 1.5$.

D. Stopping criteria

We use the duality gap certificate defined in [1] as a stopping criteria. The algorithm converges when the product between the global Frank Wolfe direction and the gradient becomes sufficiently small.

$$\langle x_k - s, \nabla f(x_k)\rangle \leq \epsilon$$

E. Short Step Chain procedure

The Sort Step Chain (SSC) procedure was implemented as described in [4]. The backtracking line search was used to estimate the stepsize and the Lipscitz constant L. To verify if the global minimum is reached we check if the condition 14 used in backtracking line search is satisfied with a small gamma. Replacing γ with $\gamma = \frac{g_k}{L\|d_k\|^2}$ in 14, we get the following inequality to check for global minimum:

$$f(x_k + \gamma d_k) \leq f(x_k) - \gamma g_k/2 \qquad (15)$$

	Instance	N	2-graph edges	4-graph edges	4-graph complement edges
0	johnson8-2-4	28	210	10185	10290
1	johnson8-4-4	70	1855	800275	116620
2	hamming6-2	64	1824	630016	5360
3	hamming6-4	64	704	61776	573600

TABLE 1: Instances used for experiments.

V. Numerical experiments

We have carried out numerical experiments to compare the performance of implemented methods on different instances. The results were tested on 4-uniform hypergraphs constructed from graphs by forming a hyperedge of 4 nodes if at least 4 connections exist in the original network. 4 graph instances *johnson8-2-4*, *johnson-8-4-4*, *hamming6-2*, *hamming6-4* from [9] were used. The choice of instances is explained by performance consideration. As shown in 1, although the number of nodes and the number of edges in the original 2-graph is not large, the constructed 4-graphs have large number of hyperedges which makes the calculation of gradient and objective function long. We used instances *johnson8-2-4* and *hamming6-2* to compare the classic FW with other variants with armijo line search as these instances have the smallest number of complement hyperedges and can run in decent time. The other instances were used to compare the performance of other variants with and without SSC procedure with backtracking line search.

For the instances *johnson8-2-4* and *hamming6-2*, the algorithms were run 50 times with the same random initialization. For instances *johnson8-4-4* and *hamming6-4* the algorithms were run only 10 times due to the time constraints. The aggregated results for all runs showing the number of iterations and CPU time required to converge, and maximum clique found are presented in tables 2 and 3 in the appendix. The plots showing o.f. value and duality gap versus CPU time and number of iterations for some seeds are presented in figures 1, 2, 3, 5, 4, 6.

As expected, classic FW is worse than other methods both in terms of number of iterations and CPU time required to converge. Away-step FW performed slightly worse than pairwise FW algorithms in most cases. The PFW performed exactly the same as BPFW most of the time which indicates the absence of swap steps. In addition, as we start with a full active set eliminating the bad atoms, local pairwise updates and global pairwise updates coincide. In our plots we show the cases where there is a difference between them. For a given problem, when swap steps occur BPFW gives a small

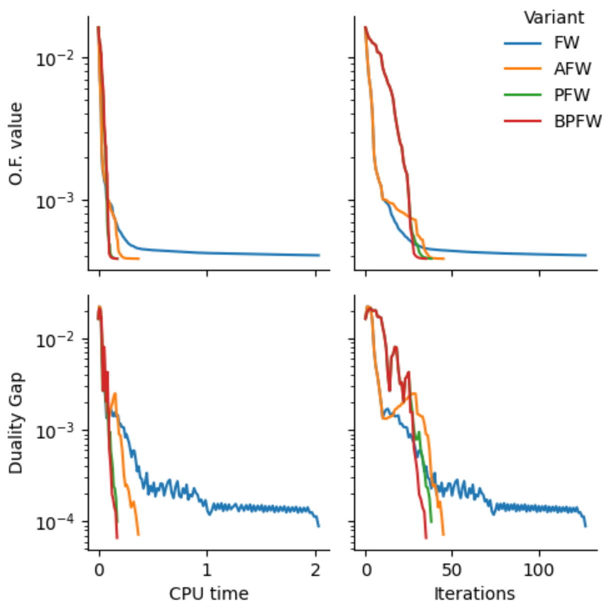

Fig. 1: FW variants with armijo line search. Instance johnson8-2-4.

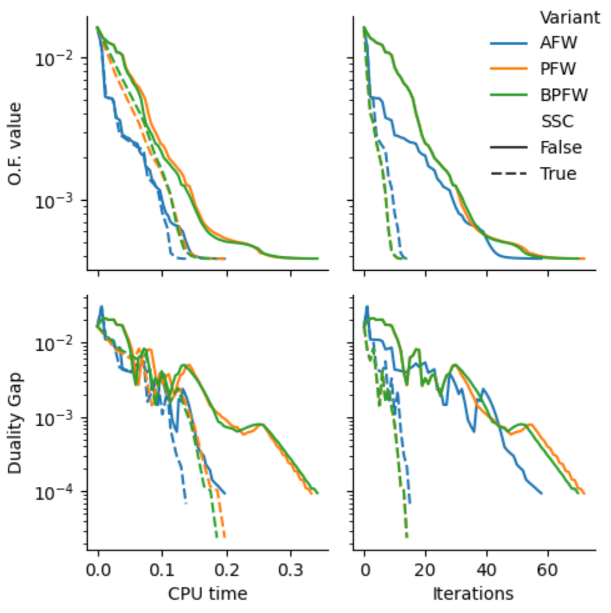

Fig. 3: FW variants with and without SSC procedure, backtracking line search. Instance johnson8-2-4.

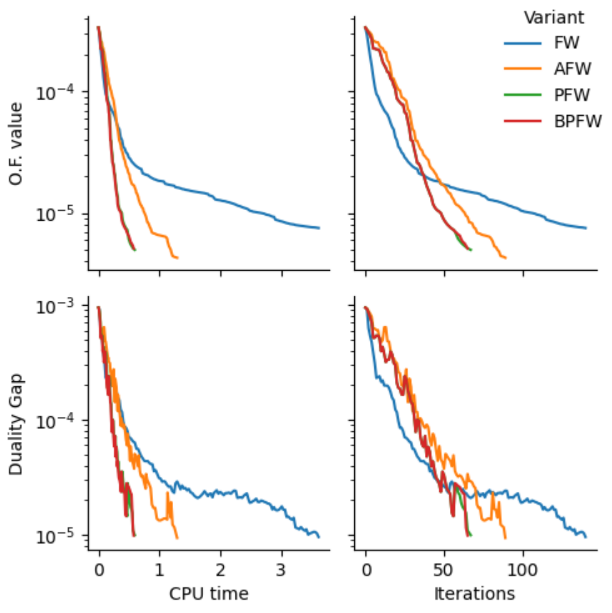

Fig. 2: FW variants with armijo line search. Instance hamming6-2.

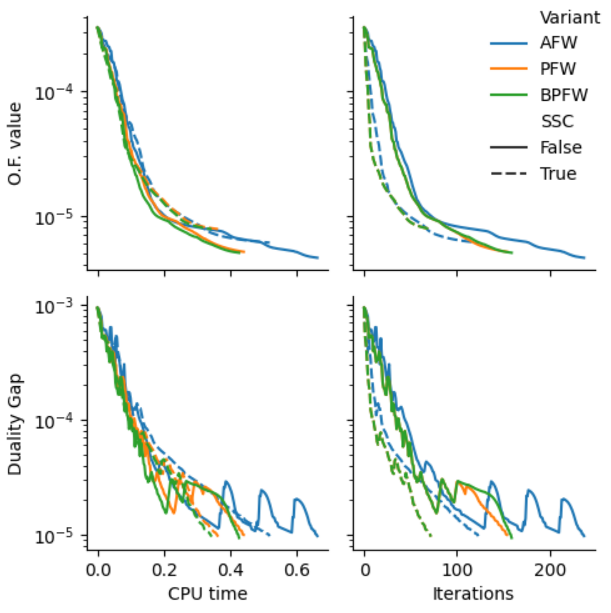

Fig. 4: FW variants with and without SSC procedure, backtracking line search. Instance hamming6-2.

advantage, however, given that swap steps are rare in our problem the averaged results are the same.

Regarding the SSC procedure, it gives a significant advantage in terms of the number or iterations required to converge. However, CPU time required to converge didn't improve for *hamming6-2* instance with the use of SSC procedure. This can be due to the fact that the as the 4-graph is dense, we have only 5360 complement hyperedges compared to 630016 existing ones, which makes the gradient calculation as fast as the calculation of the objective function value. Thus, mitigating the advantage of SSC procedure. For other instances, SSC procedure improves all FW variants that it can be appled to.

Regarding the quality of the solution, we can see that all variants perform similarly well in solving the given problem.

VI. Conclusions

We have analysed the performance of classic FW, AFW, PFW, and BPFW algorithms when solving maximum clique problem on k-uniform hypergraphs. BPFW and PFW performed better than other variants on all tested instances in terms of both convergence in time and number of iterations. BPFW and PFW show the same performance as starting

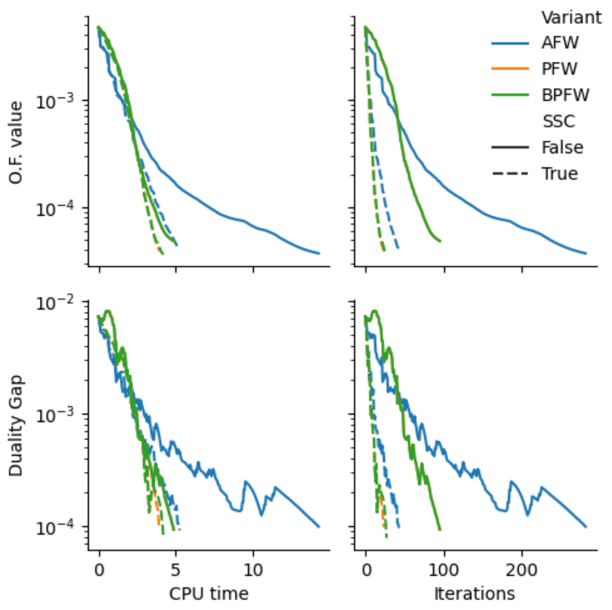

Fig. 5: FW variants with and without SSC procedure, backtracking line search. Instance johnson8-4-4.

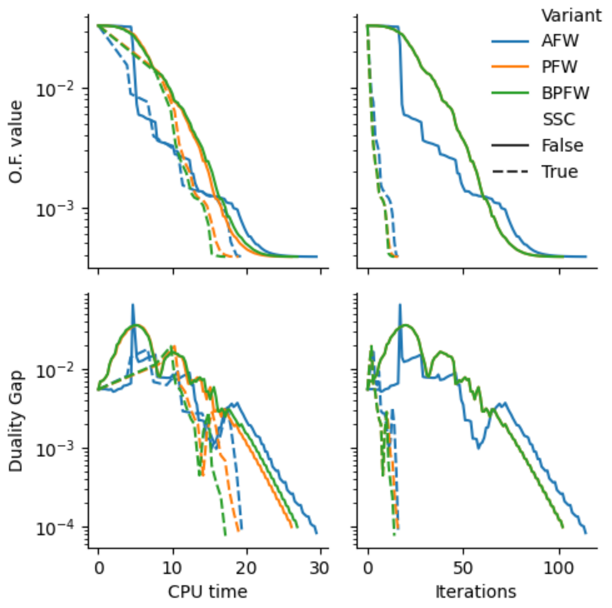

Fig. 6: FW variants with and without SSC procedure, backtracking line search. Instance hamming6-4.

with the full active set, local pairwise updates correspond to global ones. The use of SSC procedure showed a significant improvement in terms of the number of iteration required to converge, while the improvement in CPU time depended on the instance with larger improvement on more difficult instances.

References

[1] M. Jaggi, "Revisiting frank-wolfe: Projection-free sparse convex optimization," in *International conference on machine learning*, pp. 427–435, PMLR, 2013.

[2] S. Lacoste-Julien and M. Jaggi, "On the global linear convergence of frank-wolfe optimization variants," *Advances in neural information processing systems*, vol. 28, 2015.

[3] K. K. Tsuji, K. Tanaka, and S. Pokutta, "Pairwise conditional gradients without swap steps and sparser kernel herding," in *International Conference on Machine Learning*, pp. 21864–21883, PMLR, 2022.

[4] F. Rinaldi and D. Zeffiro, "Avoiding bad steps in frank-wolfe variants," *Computational Optimization and Applications*, vol. 84, no. 1, pp. 225–264, 2023.

[5] S. Rota Bulò and M. Pelillo, "A generalization of the motzkin–straus theorem to hypergraphs," *Optimization Letters*, vol. 3, pp. 287–295, 2009.

[6] F. Rinaldi, "Chapter 5: Constrained optimization." Lecture notes in Optimization for Data Science course.

[7] F. Rinaldi, "Chapter 4: Unconstrained optimization." Lecture notes in Optimization for Data Science course.

[8] F. Pedregosa, G. Negiar, A. Askari, and M. Jaggi, "Linearly convergent frank-wolfe with backtracking line-search," in *International conference on artificial intelligence and statistics*, pp. 1–10, PMLR, 2020.

[9] D. Zeffiro, I. M. Bomze, and F. Rinaldi, "Fast cluster detection in networks by first-order optimization," *PROCEEDINGS OF SIMAI 2020+21*, 2021.

Appendix

TABLE 2: Number of iterations, CPU time and Max Clique for 4 FW variants with armijo line search.

			Iterations			CPU time			Max clique	
Instance	Variant	max	mean	std	max	mean	std	max	mean	std
hamming6-2	AFW	101	77.320	14.174	0.701	0.407	0.117	32	26.840	2.359
	BPFW	75	61.120	5.524	0.303	0.208	0.048	32	26.640	1.626
	FW	166	131.640	16.964	2.223	1.459	0.297	32	25.900	1.799
	PFW	76	60.520	5.970	0.313	0.197	0.044	32	26.580	1.605
johnson8-2-4	AFW	48	43.160	2.526	0.519	0.331	0.070	6	6.000	0.000
	BPFW	45	37.620	2.791	0.369	0.241	0.053	6	5.980	0.141
	FW	171	116.900	20.528	3.360	2.169	0.522	6	6.000	0.000
	PFW	45	37.820	2.731	0.386	0.247	0.060	6	5.980	0.141

TABLE 3: Number of iterations, CPU time and Max Clique for AFW, PFW, BPFW with and without SSC procedure, backtracking line search.

				Iterations			CPU time			Max clique	
Instance	Variant	SSC	max	mean	std	max	mean	std	max	mean	std
hamming6-2	AFW	False	348	173.220	54.301	0.876	0.437	0.153	32	25.640	2.164
		True	214	122.900	30.824	0.859	0.485	0.130	32	26.100	2.315
	BPFW	False	228	123.800	29.703	0.521	0.316	0.087	32	26.560	1.786
		True	163	89.640	26.753	0.610	0.380	0.103	32	26.640	1.860
	PFW	False	182	113.180	21.546	0.470	0.290	0.073	32	26.620	1.817
		True	119	80.920	18.632	0.591	0.356	0.085	32	26.780	2.023
hamming6-4	AFW	False	144	97.900	18.321	36.522	25.285	4.570	6	6.000	0.000
		True	25	20.400	2.716	22.998	20.741	1.335	6	6.000	0.000
	BPFW	False	109	99.900	6.919	28.511	25.629	1.972	6	6.000	0.000
		True	17	15.000	1.155	19.921	18.516	0.886	6	6.000	0.000
	PFW	False	109	100.100	7.172	29.255	25.670	2.360	6	6.000	0.000
		True	18	15.400	1.578	20.340	18.833	1.343	6	6.000	0.000
johnson8-2-4	AFW	False	87	56.560	11.720	0.414	0.231	0.054	6	5.980	0.141
		True	20	16.400	1.906	0.281	0.194	0.041	7	6.020	0.141
	BPFW	False	79	51.160	10.961	0.367	0.200	0.060	6	5.980	0.141
		True	18	14.040	2.080	0.275	0.175	0.039	6	5.860	0.351
	PFW	False	79	51.180	10.982	0.355	0.201	0.058	6	5.980	0.141
		True	18	14.120	2.096	0.277	0.180	0.040	6	5.860	0.351
johnson8-4-4	AFW	False	649	220.000	157.579	36.124	11.464	8.994	16	15.000	1.155
		True	49	43.900	4.175	6.009	5.147	0.690	16	14.900	1.101
	BPFW	False	132	100.700	15.004	7.243	5.143	1.076	17	14.400	1.350
		True	33	28.200	3.011	5.337	4.188	0.628	15	14.000	0.667
	PFW	False	143	104.800	18.214	7.883	5.303	1.250	16	14.300	0.949
		True	32	27.600	3.098	5.172	4.121	0.590	15	14.200	0.632