Efficient Optimization of Dominant Set Clustering with Frank-Wolfe Algorithms

Efficient Optimization of Dominant Set Clustering with Frank-Wolfe Algorithms

Carl Johnell
Chalmers University of Technology
Department of Computer Science and Engineering
Gothenburg, Sweden
cjohnell@gmail.com

Morteza Haghir Chehreghani
Chalmers University of Technology
Department of Computer Science and Engineering
Gothenburg, Sweden
morteza.chehreghani@chalmers.se

ABSTRACT

We study Frank-Wolfe algorithms – standard, pairwise, and away-steps – for efficient optimization of Dominant Set Clustering. We present a unified and computationally efficient framework to employ the different variants of Frank-Wolfe methods, and we investigate its effectiveness via several experimental studies. In addition, we provide explicit convergence rates for the algorithms in terms of the so-called Frank-Wolfe gap. The theoretical analysis has been specialized to Dominant Set Clustering and covers consistently the different variants.

CCS CONCEPTS

• **Computing methodologies → Cluster analysis**.

KEYWORDS

Clustering, Dominant Set, Replicator Dynamics, Frank-Wolfe methods

ACM Reference Format:
Carl Johnell and Morteza Haghir Chehreghani. 2022. Efficient Optimization of Dominant Set Clustering with Frank-Wolfe Algorithms. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), Oct. 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557306

1 INTRODUCTION

Clustering plays an important role in unsupervised learning and exploratory data analytics [20]. It is used in applications from different domains such as network analysis, image segmentation, document and text processing, community detection and bioinformatics. Given a set of n objects with indices $V = \{1, ..., n\}$ and the nonnegative pairwise similarities $\mathbf{A} = (a_{ij})$, i.e., graph $\mathcal{G}(V, \mathbf{A})$ with vertices V and edge weights $\mathbf{A}$, the goal is to partition the data into coherent groups that look dissimilar from each other. We assume zero self-similarities, i.e., $a_{ii} = 0\ \forall i$. Several clustering methods compute the clusters via minimizing a cost function. Examples are Ratio Cut [7], Normalized Cut [36], Correlation Clustering [1], and shifted Min Cut [9, 15]. For some of them, for example Normalized Cut, approximate solutions have been developed in the context of spectral analysis [27, 36], Power Iteration Clustering (PIC) method [25] and P-Spectral Clustering [3, 18]. They are sometimes combined with greedy hierarchical clustering, e.g. K-means combined with agglomerative clustering [10].

Another prominent clustering approach has been developed in the context of Dominant Set Clustering (DSC) and its connection to discrete-time dynamical systems and *replicator dynamics* [4, 31]. Unlike the methods based on cost function minimization, DSC does not define a global cost function for the clusters. Instead, it applies the generic principles of clustering where each cluster should be coherent and well separated from the other clusters. These principles are formulated via the concepts of dominant sets [31]. Then, several variants of the method have been proposed. The method in [26] proposes an iterative clustering algorithm in two Shrink and Expand steps. These steps are suitable for sparse data and lead to reducing the runtime of performing replicator dynamics. [6] develops an enumeration technique for different clusters via unstabilizing the underlying equilibrium of replicator dynamics. [29] proposes a hierarchical variant of DSC via regularization and shifting the off-diagonal elements of the similarity matrix. [8] analyzes adaptively the trajectories of replicator dynamics in order to discover suitable phase transitions that correspond to evolving fast clusters. Several studies demonstrate the effectiveness of DSC variants compared to other clustering methods, such as spectral methods [4, 8, 26, 31].

In this paper, we investigate efficient optimization for DSC based on Frank-Wolfe algorithms [13, 24, 33] as an alternative to replicator dynamics. Frank-Wolfe optimization has been successfully applied to several constrained optimization problems. We develop a unified and computationally efficient framework to employ the different variants of Frank-Wolfe algorithms for DSC, and we investigate its effectiveness via several experimental studies. Our theoretical analysis is specialized to DSC, and we provide explicit convergence rates for the algorithms in terms of Frank-Wolfe gap – including pairwise Frank-Wolfe with nonconvex/nonconcave objective function for which we have not seen any theoretical analysis in prior work. In addition, we study multi-start Dominant Set Clustering that can be potentially useful for parallelizing the method.

We note that beyond DSC, replicator dynamics is used in several other domains such as evolutionary game theory [11], theoretical biology [28], dynamical systems [35], online learning [37], combinatorial optimization problems [32], and several other tasks beyond clustering [4]. Hence, our contribution opens novel possibilities to investigate more efficient alternatives in those paradigms.

2 DOMINANT SET CLUSTERING

DSC follows an iterative procedure to compute the clusters: i) computes a dominant set using the similarity matrix $\mathbf{A}$ of the

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10.
https://doi.org/10.1145/3511808.3557306

available data, ii) peels off (removes) the clustered objects from the data, and iii) repeats until a predefined number of clusters have been obtained.[1]

Dominant sets correspond to local optima of the following quadratic problem [31], called standard quadratic problem (StQP).

$$\text{maximize } f(\mathbf{x}) = \mathbf{x}^T \mathbf{A}\mathbf{x} \tag{1}$$
$$\text{subject to } \mathbf{x} \in \Delta = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq \mathbf{0}^n \text{ and } \sum_{i=1}^{n} x_i = 1 \right\}.$$

The constraint Δ is called the standard simplex. We note that $\mathbf{A}$ is generally not negative definite, and the objective function $f(\mathbf{x})$ is thus *not* concave.

Every unclustered object i corresponds to a component of the n-dimensional characteristic vector $\mathbf{x}$. The support of local optimum $\mathbf{x}^*$ specifies the objects that belong to the dominant set (cluster), i.e., i is in the cluster if component $x_i^* > 0$. In practice we use $x_i^* > \delta$, where δ is a small number called the cutoff parameter. Previous works employ replicator dynamics to solve StQP, where $\mathbf{x}$ is updated according to the following dynamics.

$$x_i^{(t+1)} = x_i^{(t)} \frac{(\mathbf{A}\mathbf{x}_t)_i}{\mathbf{x}_t^{\mathsf{T}} \mathbf{A}\mathbf{x}_t}, i = 1, .., n, \tag{2}$$

where $\mathbf{x}_t$ indicates the solution at iterate t, and $x_i^{(t)}$ is the i-th component of $\mathbf{x}_t$. We note the $O(n^2)$ per-iteration time complexity due to the matrix multiplication.

In this paper we investigate an alternative optimization framework based on Frank-Wolfe methods.

3 UNIFIED FRANK-WOLFE OPTIMIZATION METHODS

Let $\mathcal{P} \subset \mathbb{R}^n$ be a finite set of points and $\mathcal{D} = \text{convex}(\mathcal{P})$ its convex hull (convex polytope). The Frank-Wolfe algorithm, first introduced in [13], aims at solving the following constrained optimization.

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

where f is nonlinear and differentiable. The formulation in [23] has extended the concavity assumption to arbitrary functions with L-Lipschitz ('well-behaved') gradients. Algorithm 1 outlines the steps of a Frank-Wolfe method to solve the optimization in (3).

In this work, in addition to the standard FW (called FW), we also consider two other variants of FW: pairwise FW (PFW) and away-steps FW (AFW), adapted from [24]. They differ in the way the ascent direction $\mathbf{d}_t$ is computed.

From the definition of $\mathcal{D}$, any point $\mathbf{x}_t \in \mathcal{D}$ can be written as a convex combination of the points in $\mathcal{P}$, i.e.,

$$\mathbf{x}_t = \sum_{\mathbf{v} \in \mathcal{P}} \lambda_{\mathbf{v}}^{(t)} \mathbf{v}, \tag{4}$$

where the coefficients $\lambda_{\mathbf{v}}^{(t)} \in [0, 1]$ and $\sum_{\mathbf{v} \in \mathcal{P}} \lambda_{\mathbf{v}}^{(t)} = 1$. Define

$$S_t = \{\mathbf{v} \in \mathcal{P} : \lambda_{\mathbf{v}}^{(t)} > 0\} \tag{5}$$

[1] With some abuse of the notation, n, V, $\mathbf{A}$ and $\mathbf{x}$ sometimes refer to the available (i.e., still unclustered) objects and the similarities between them. This is obvious from the context. The reason is that DSC performs a sequential procedure where at each step separates a cluster from the available unclustered data.

Algorithm 1 Frank-Wolfe pseudocode

```
procedure PSEUDO-FW(f, D, T)        ▷ Function f, convex polytope D, and iterations T.
    Select x_0 ∈ D
    for t = 0, ..., T − 1 do
        if x_t is stationary then break
        Compute feasible ascent direction d_t at x_t
        Compute step size γ_t ∈ [0, 1] such that f(x_t + γ_t d_t) > f(x_t)
        x_{t+1} := x_t + γ_t d_t
    return x_t
```

as the set of points with nonzero coefficients at iterate t. Moreover, let

$$\mathbf{s}_t \in \arg\max_{\mathbf{s} \in \mathcal{D}} \nabla f(\mathbf{x}_t)^T \mathbf{s}, \tag{6}$$
$$\mathbf{v}_t \in \arg\min_{\mathbf{v} \in S_t} \nabla f(\mathbf{x}_t)^T \mathbf{v}. \tag{7}$$

Since $\mathcal{D}$ is a convex polytope, $\mathbf{s}_t$ is the point that maximizes the linearization and $\mathbf{v}_t$ is the point with nonzero coefficient that minimizes it over S_t. Let $\mathbf{x}_t$ be the estimated solution of (3) at iterate t and define

$$\begin{aligned}
\mathbf{d}_t^A &= \mathbf{x}_t - \mathbf{v}_t, \\
\mathbf{d}_t^{FW} &= \mathbf{s}_t - \mathbf{x}_t, \\
\mathbf{d}_t^{AFW} &= \begin{cases} \mathbf{d}_t^{FW}, & \text{if } \nabla f(\mathbf{x}_t)^T \mathbf{d}_t^{FW} \geq f(\mathbf{x}_t)^T \mathbf{d}_t^A \\ \frac{\lambda_{\mathbf{v}_t}^{(t)}}{1 - \lambda_{\mathbf{v}_t}^{(t)}} \mathbf{d}_t^A, & \text{otherwise} \end{cases} \\
\mathbf{d}_t^{PFW} &= \mathbf{s}_t - \mathbf{v}_t
\end{aligned} \tag{8}$$

respectively as the away, FW, pairwise, and away/FW directions (to be maximized). The FW direction moves towards a 'good' point, and the away direction moves away from a 'bad' point. The pairwise direction shifts from a 'bad' point to a 'good' point [24]. The coefficient with $\mathbf{d}_t^A$ in $\mathbf{d}_t^{AFW}$ ensures the next iterate remains feasible.

An issue with standard FW, which PFW and AFW aim to fix, is the zig-zagging phenomenon. This occurs when the optimal solution of (3) lies on the boundary of the domain. Then the iterates start to zig-zag between the points, which negatively affects the convergence. By adding the possibility of an away step in AFW, or alternatively using the pairwise direction, zig-zagging can be attenuated.

The step size γ_t can be computed by line-search, i.e.,

$$\gamma_t \in \arg\max_{\gamma \in [0,1]} f(\mathbf{x}_t + \gamma \mathbf{d}_t). \tag{9}$$

Finally, the Frank-Wolfe gap is used to check if an iterate is (close enough to) a stationary solution.

Definition 1. *The Frank-Wolfe gap g_t of $f : \mathcal{D} \to \mathbb{R}$ at iterate t is defined as*

$$g_t = \max_{\mathbf{s} \in \mathcal{D}} \nabla f(\mathbf{x}_t)^T (\mathbf{s} - \mathbf{x}_t) \iff g_t = \nabla f(\mathbf{x}_t)^T \mathbf{d}_t^{FW}. \tag{10}$$

A point $\mathbf{x}_t$ is stationary if and only if $g_t = 0$, meaning there are no ascent directions. The Frank-Wolfe gap is thus a reasonable measure of nonstationarity and is frequently used as a stopping

criterion [23]. Specifically, a threshold ϵ is defined, and if $g_t \leq \epsilon$, then we conclude the $\mathbf{x}_t$ at the current iterate t is sufficiently close to a stationary point and we stop the algorithm.

4 FRANK-WOLFE FOR DOMINANT SET CLUSTERING

Here we apply the Frank-Wolfe methods from the previous section to the optimization problem (1) defined by DSC.

Optimization in Simplex Domain. Because of the simplex form – the constraints in (1) – the convex combination in (4) for $\mathbf{x} \in \Delta$ can be written as $\mathbf{x} = \sum_{i=1}^{n} \lambda_{\mathbf{e}_i} \mathbf{e}_i$, where $\mathbf{e}_i$ are the standard basis vectors. That is, the i-th coefficient corresponds to the i-th component of $\mathbf{x}$, $\lambda_{\mathbf{e}_i} = x_i$. The set of components with nonzero coefficients of $\mathbf{x}_t$ gives the support, i.e., $\sigma_t = \{i \in V : x_i^{(t)} > 0\}$.

Due to the structure of the simplex Δ, the solution of the optimization (6) is

$$\begin{cases} \mathbf{s}_t & \in \Delta \\ s_i^{(t)} & = 1, \quad \text{where } i \in \arg\max_i \nabla_i f(\mathbf{x}_t) \\ s_j^{(t)} & = 0, \quad \text{for } j \neq i, \end{cases} \tag{11}$$

and the optimization (7) is obtained by

$$\begin{cases} \mathbf{v}_t & \in \Delta \\ v_i^{(t)} & = 1, \quad \text{where } i \in \arg\min_{i \in \sigma_t} \nabla_i f(\mathbf{x}_t) \\ v_j^{(t)} & = 0, \quad \text{for } j \neq i. \end{cases} \tag{12}$$

The maximum and minimum values of the linearization are the largest and smallest components of the gradient, respectively (subject to $i \in \sigma_t$ in the latter case). Note that the gradient is $\nabla f(\mathbf{x}_t) = 2\mathbf{A}\mathbf{x}_t$.

Step Sizes. We compute the optimal step sizes for FW, PFW, and AFW. Iterate subscripts t are omitted for clarity. We define the step size function as

$$\begin{aligned} \psi(\gamma) &= f(\mathbf{x} + \gamma\mathbf{d}) \\ &= (\mathbf{x} + \gamma\mathbf{d})^T \mathbf{A}(\mathbf{x} + \gamma\mathbf{d}) \\ &= \mathbf{x}^T\mathbf{A}\mathbf{x} + 2\gamma\mathbf{d}^T\mathbf{A}\mathbf{x} + \gamma^2\mathbf{d}^T\mathbf{A}\mathbf{d} \\ &= f(\mathbf{x}) + \gamma\nabla f(\mathbf{x}_t)^T\mathbf{d} + \gamma^2\mathbf{d}^T\mathbf{A}\mathbf{d}, \end{aligned} \tag{13}$$

for some ascent direction $\mathbf{d}$. This expression is a single variable second degree polynomial in γ. The function is concave if the coefficient $\mathbf{d}^T\mathbf{A}\mathbf{d} \leq 0$ – second derivative test – and admits a global maximum in that case.

In the following it is assumed that $\mathbf{s}$ and $\mathbf{v}$ satisfy (11) and (12), and their nonzero components are i and j, respectively.

FW direction: Substitute $\mathbf{d}^{FW} = \mathbf{s} - \mathbf{x}$ into $\mathbf{d}^T\mathbf{A}\mathbf{d}$.

$$\begin{aligned} \mathbf{d}^T\mathbf{A}\mathbf{d} &= (\mathbf{s} - \mathbf{x})^T\mathbf{A}(\mathbf{s} - \mathbf{x}) \\ &= \mathbf{s}^T\mathbf{A}\mathbf{s} - 2\mathbf{s}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{A}\mathbf{x} \\ &= -(2\mathbf{s}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{A}\mathbf{x}) \\ &= \mathbf{x}^T\mathbf{A}\mathbf{x} - 2\mathbf{a}_{i*}^T\mathbf{x}. \end{aligned} \tag{14}$$

The i-th row of $\mathbf{A}$ is $\mathbf{a}_{i*}$ and its j-th column is $\mathbf{a}_{*j}$.

Pairwise direction: Substitute $\mathbf{d}^{PFW} = \mathbf{s} - \mathbf{v}$ into $\mathbf{d}^T\mathbf{A}\mathbf{d}$.

$$\begin{aligned} \mathbf{d}^T\mathbf{A}\mathbf{d} &= (\mathbf{s} - \mathbf{v})^T\mathbf{A}(\mathbf{s} - \mathbf{v}) \\ &= \mathbf{s}^T\mathbf{A}\mathbf{s} - 2\mathbf{v}^T\mathbf{A}\mathbf{s} + \mathbf{v}^T\mathbf{A}\mathbf{v} = -2a_{ij}. \end{aligned} \tag{15}$$

Away direction: Substitute $\mathbf{d}^A = \mathbf{x} - \mathbf{v}$ into $\mathbf{d}^T\mathbf{A}\mathbf{d}$.

$$\begin{aligned} \mathbf{d}^T\mathbf{A}\mathbf{d} &= (\mathbf{x} - \mathbf{v})^T\mathbf{A}(\mathbf{x} - \mathbf{v}) \\ &= \mathbf{x}^T\mathbf{A}\mathbf{x} - 2\mathbf{v}^T\mathbf{A}\mathbf{x} + \mathbf{v}^T\mathbf{A}\mathbf{v} \\ &= \mathbf{x}^T\mathbf{A}\mathbf{x} - 2\mathbf{a}_{j*}^T\mathbf{x}. \end{aligned} \tag{16}$$

Recall $\mathbf{A}$ has nonnegative entries and zeros on the main diagonal. Therefore $\mathbf{s}^T\mathbf{A}\mathbf{s} = 0$ and $\mathbf{v}^T\mathbf{A}\mathbf{v} = 0$. It is immediate that (15) is nonpositive. From $\mathbf{x}^T\mathbf{A}\mathbf{x} \leq \mathbf{s}^T\mathbf{A}\mathbf{x}$ we conclude that (14) is also nonpositive. The corresponding step size functions are therefore always *concave*. We cannot make any conclusion for (16), and the sign of $\mathbf{d}^T\mathbf{A}\mathbf{d}$ depends on $\mathbf{x}$.

The derivative of $\psi(\gamma)$ is

$$\frac{d\psi}{d\gamma}(\gamma) = \nabla f(\mathbf{x})^T\mathbf{d} + 2\gamma\mathbf{d}^T\mathbf{A}\mathbf{d}. \tag{17}$$

By solving $\frac{d\psi}{d\gamma}(\gamma) = 0$ we obtain

$$\begin{aligned} &\nabla f(\mathbf{x})^T\mathbf{d} + 2\gamma\mathbf{d}^T\mathbf{A}\mathbf{d} = 0 \\ &\iff \\ &\gamma^* = -\frac{\nabla f(\mathbf{x})^T\mathbf{d}}{2\mathbf{d}^T\mathbf{A}\mathbf{d}} = -\frac{\mathbf{x}^T\mathbf{A}\mathbf{d}}{\mathbf{d}^T\mathbf{A}\mathbf{d}}. \end{aligned} \tag{18}$$

Since $\nabla f(\mathbf{x})^T\mathbf{d} \geq 0$, we also conclude here that $\mathbf{d}^T\mathbf{A}\mathbf{d} < 0$ has to hold in order for the step size to make sense.

By substituting the directions and corresponding $\mathbf{d}^T\mathbf{A}\mathbf{d}$ into (18) we obtain the optimal step sizes.

FW direction and (14):

$$\gamma^{FW} = -\frac{\mathbf{x}^T\mathbf{A}\mathbf{d}}{\mathbf{d}^T\mathbf{A}\mathbf{d}} = \frac{\mathbf{a}_{i*}^T\mathbf{x} - \mathbf{x}^T\mathbf{A}\mathbf{x}}{2\mathbf{a}_{i*}^T\mathbf{x} - \mathbf{x}^T\mathbf{A}\mathbf{x}}. \tag{19}$$

Pairwise direction and (15):

$$\gamma^{PFW} = -\frac{\mathbf{x}^T\mathbf{A}\mathbf{d}}{\mathbf{d}^T\mathbf{A}\mathbf{d}} = \frac{\mathbf{a}_{i*}^T\mathbf{x} - \mathbf{a}_{j*}^T\mathbf{x}}{2a_{ij}}. \tag{20}$$

Away direction and (16):

$$\gamma^{A} = -\frac{\mathbf{x}^T\mathbf{A}\mathbf{d}}{\mathbf{d}^T\mathbf{A}\mathbf{d}} = \frac{\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{a}_{j*}^T\mathbf{x}}{2\mathbf{a}_{j*}^T\mathbf{x} - \mathbf{x}^T\mathbf{A}\mathbf{x}}. \tag{21}$$

Algorithms. Here, we describe in detail standard FW (Algorithm 2), pairwise FW (Algorithm 3), and away-steps FW (Algorithm 4) for problem (1), following the high-level structure of Algorithm 1. All variants have $O(n)$ per-iteration time complexity, where the linear operations are arg max, arg min, and vector addition. The key for this complexity is that we can update the gradient $\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x}$ in linear time. Lemmas 1, 2 and 3 show why this is the case. Recall the updates in replicator dynamics are quadratic w.r.t. n. [2]

[2] Proofs are discarded due to space limit. They will be provided in a longer version of the paper.

Algorithm 2 FW for DSC

1: **procedure** FW($\mathbf{A}$, ϵ, T)
2: Select $\mathbf{x}_0 \in \Delta$
3: $\mathbf{r}_0 := \mathbf{A}\mathbf{x}_0$
4: $f_0 := \mathbf{r}_0^T \mathbf{x}_0$
5: **for** $t = 0, ..., T-1$ **do**
6: $\mathbf{s}_t := \mathbf{e}_i$, where $i \in \arg\max_\ell r_\ell^{(t)}$
7: $g_t := r_i^{(t)} - f_t$
8: **if** $g_t \leq \epsilon$ **then break**
9: $\gamma_t := \frac{r_i^{(t)} - f_t}{2r_i^{(t)} - f_t}$
10: $\mathbf{x}_{t+1} := (1-\gamma_t)\mathbf{x}_t + \gamma_t \mathbf{s}_t$
11: $\mathbf{r}_{t+1} := (1-\gamma_t)\mathbf{r}_t + \gamma_t \mathbf{a}_{*i}$
12: $f_{t+1} := (1-\gamma_t)^2 f_t + 2\gamma_t(1-\gamma_t) r_i^{(t)}$
13: **return** $\mathbf{x}_t$

Lemma 1. *For $\mathbf{x}_{t+1} = (1-\gamma_t)\mathbf{x}_t + \gamma_t \mathbf{s}_t$, lines 11 and 12 in Algorithm 2 satisfy*

$$\mathbf{r}_{t+1} = \mathbf{A}\mathbf{x}_{t+1}, f_{t+1} = \mathbf{x}_{t+1}^T \mathbf{A}\mathbf{x}_{t+1}.$$

Algorithm 3 Pairwise FW for DSC

1: **procedure** PFW($\mathbf{A}$, ϵ, T)
2: Select $\mathbf{x}_0 \in \Delta$
3: $\mathbf{r}_0 := \mathbf{A}\mathbf{x}_0$
4: $f_0 := \mathbf{r}_0^T \mathbf{x}_0$
5: **for** $t = 0, ..., T-1$ **do**
6: $\sigma_t := \{i \in V : x_i^{(t)} > 0\}$
7: $\mathbf{s}_t := \mathbf{e}_i$, where $i \in \arg\max_\ell r_\ell^{(t)}$
8: $\mathbf{v}_t := \mathbf{e}_j$, where $j \in \arg\min_{\ell \in \sigma_t} r_\ell^{(t)}$
9: $g_t := r_i^{(t)} - f_t$
10: **if** $g_t \leq \epsilon$ **then break**
11: $\gamma_t := \min\left(x_j^{(t)}, \frac{r_i^{(t)} - r_j^{(t)}}{2a_{ij}}\right)$
12: $\mathbf{x}_{t+1} := \mathbf{x}_t + \gamma_t(\mathbf{s}_t - \mathbf{v}_t)$
13: $\mathbf{r}_{t+1} := \mathbf{r}_t + \gamma_t(\mathbf{a}_{*i} - \mathbf{a}_{*j})$
14: $f_{t+1} := f_t + 2\gamma_t(r_i^{(t)} - r_j^{(t)}) - 2\gamma_t^2 a_{ij}$
15: **return** $\mathbf{x}_t$

Lemma 2. *For $\mathbf{x}_{t+1} = \mathbf{x}_t + \gamma_t(\mathbf{s}_t - \mathbf{v}_t)$, lines 13 and 14 in Algorithm 3 satisfy*

$$\mathbf{r}_{t+1} = \mathbf{A}\mathbf{x}_{t+1}, f_{t+1} = \mathbf{x}_{t+1}^T \mathbf{A}\mathbf{x}_{t+1}.$$

Lines 12-15 are identical to the updates in Algorithm 2 included in Lemma 1. We thus only show the away direction.

Lemma 3. *For $\mathbf{x}_{t+1} = (1+\gamma_t)\mathbf{x}_t - \gamma_t \mathbf{v}_t$, lines 22 and 23 in Algorithm 4 satisfy*

$$\mathbf{r}_{t+1} = \mathbf{A}\mathbf{x}_{t+1}, f_{t+1} = \mathbf{x}_{t+1}^T \mathbf{A}\mathbf{x}_{t+1}.$$

Algorithm 4 (AFW) is actually equivalent to the *infection and immunization dynamics* (InImDyn) with the pure strategy selection function, introduced in [5] as an alternative to replicator dynamics. However, InImDyn is derived from the perspective of evolutionary game theory as opposed to Frank-Wolfe. Thus, our framework provides a way to connect replicator dynamics and InImDyn in a principled way. Moreover, it allows us to further analyze this method and study its convergence rate.

Algorithm 4 Away-steps FW for DSC

1: **procedure** AFW($\mathbf{A}$, ϵ, T)
2: Select $\mathbf{x}_0 \in \Delta$
3: $\mathbf{r}_0 := \mathbf{A}\mathbf{x}_0$
4: $f_0 := \mathbf{r}_0^T \mathbf{x}_0$
5: **for** $t = 0, ..., T-1$ **do**
6: $\sigma_t := \{i \in V : x_i^{(t)} > 0\}$
7: $\mathbf{s}_t := \mathbf{e}_i$, where $i \in \arg\max_\ell r_\ell^{(t)}$
8: $\mathbf{v}_t := \mathbf{e}_j$, where $j \in \arg\min_{\ell \in \sigma_t} r_\ell^{(t)}$
9: $g_t := r_i^{(t)} - f_t$
10: **if** $g_t \leq \epsilon$ **then break**
11: **if** $(r_i^{(t)} - f_t) \geq (f_t - r_j^{(t)})$ **then** ▷ FW direction
12: $\gamma_t := \frac{r_i^{(t)} - f_t}{2r_i^{(t)} - f_t}$
13: $\mathbf{x}_{t+1} := (1-\gamma_t)\mathbf{x}_t + \gamma_t \mathbf{s}_t$
14: $\mathbf{r}_{t+1} := (1-\gamma_t)\mathbf{r}_t + \gamma_t \mathbf{a}_{*i}$
15: $f_{t+1} := (1-\gamma_t)^2 f_t + 2\gamma_t(1-\gamma_t) r_i^{(t)}$
16: **else** ▷ Away direction
17: $\gamma_t := x_j^{(t)} / (1 - x_j^{(t)})$
18: **if** $(2r_j^{(t)} - f_t) > 0$ **then**
19: $\gamma_t \leftarrow \min\left(\gamma_t, \frac{f_t - r_j^{(t)}}{2r_j^{(t)} - f_t}\right)$
20: $\mathbf{x}_{t+1} := (1+\gamma_t)\mathbf{x}_t - \gamma_t \mathbf{v}_t$
21: $\mathbf{r}_{t+1} := (1+\gamma_t)\mathbf{r}_t - \gamma_t \mathbf{a}_{*j}$
22: $f_{t+1} := (1+\gamma_t)^2 f_t - 2\gamma_t(1+\gamma_t) r_j^{(t)}$
23: **return** $\mathbf{x}_t$

Proposition 1. *Algorithm 4 (AFW) is equivalent to InImDyn, i.e., Algorithm 1 in [5].*

5 ANALYSIS OF CONVERGENCE RATES

[23] shows that the Frank-Wolfe gap for standard FW decreases at rate $O(1/\sqrt{t})$ for nonconvex/nonconcave objective functions, where t is the number of iterations. A similar convergence rate is shown in [2] for nonconvex AFW over the simplex. When the objective function is convex/concave, linear convergence rates for PFW and AFW are shown in [24]. The analysis in [38] shows linear convergence rate of standard FW for nonconvex but multi-linear functions. We are not aware of any work analyzing the convergence rate in terms of the Frank-Wolfe gap for nonconvex/nonvoncave PFW.

Following the terminology and techniques in [2, 23, 24], we present a unified and specialized framework to analyze convergence rates for Algorithms 2, 3, and 4. The analysis is split into a number of different cases, where each case handles a unique ascent direction

and step size combination. For the step sizes, we consider one case when the optimal step size is used ($\gamma_t < \gamma_{max}$), and a second case when it has been truncated ($\gamma_t = \gamma_{max}$). The former case is referred to as a good step, since in this case we can provide a lower bound on the progress $f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t)$ in terms of the Frank-Wolfe gap. The latter case is referred to as a drop step or a swap step. It is called a drop step when the cardinality of the support reduces by one, i.e., $|\sigma_{t+1}| = |\sigma_t| - 1$, and it is called a swap step when it remains unchanged, i.e., $|\sigma_{t+1}| = |\sigma_t|$. When $\gamma_t = \gamma_{max}$ we cannot provide a bound on the progress in terms of the Frank-Wolfe gap, and instead we bound the number of drop/swap steps. Furthermore, this case can only happen for PFW and AFW as the step size for FW always satisfies $\gamma_t < \gamma_{max}$. Swap steps can only happen for PFW.

Let $\tilde{g}_t = \min_{0 \le \ell \le t} g_\ell$, $\underline{M} = \min_{i,j:i \neq j} a_{ij}$, $\overline{M} = \max_{i,j:i \neq j} a_{ij}$ be the smallest Frank-Wolfe gap after t iterations, and the smallest and largest off-diagonal elements of $\mathbf{A}$. Let I be the indexes that take a good step. That is, for $t \in I$ we have $\gamma_t < \gamma_{max}$. Then, we show the following results.

Lemma 4. *The smallest Frank-Wolfe gap for Algorithms 2, 3, and 4 satisfy*

$$\tilde{g}_t \le 2\sqrt{\frac{\beta\,(f(\mathbf{x}_t) - f(\mathbf{x}_0))}{|I|}}, \tag{22}$$

where $\beta = 2\overline{M} - \underline{M}$ for FW and AFW, and $\beta = 2\overline{M}$ for PFW.

Theorem 1. *The smallest Frank-Wolfe gap for Algorithm 2 (FW) satisfies*

$$\tilde{g}_t^{FW} \le 2\sqrt{\frac{(2\overline{M} - \underline{M})\,(f(\mathbf{x}_t) - f(\mathbf{x}_0))}{t}}. \tag{23}$$

Theorem 2. *The smallest Frank-Wolfe gap for Algorithm 3 (PFW) satisfies*

$$\tilde{g}_t^{PFW} \le 2\sqrt{\frac{6n!\overline{M}\,(f(\mathbf{x}_t) - f(\mathbf{x}_0))}{t}}. \tag{24}$$

Theorem 3. *The smallest Frank-Wolfe gap for Algorithm 4 (AFW) satisfies*

$$\tilde{g}_t^{AFW} \le 2\sqrt{\frac{2(2\overline{M} - \underline{M})\,(f(\mathbf{x}_t) - f(\mathbf{x}_0))}{t + 1 - |\sigma_0|}}. \tag{25}$$

From Theorems 1, 2 and 3 we conclude Corollary 1.

Corollary 1. *The smallest Frank-Wolfe gap for Algorithms 2, 3, and 4 decrease at rate $O(1/\sqrt{t})$.*

Initialization. The way the algorithms are initialized – value of $\mathbf{x}_0$ – affects the local optima the algorithms converge to. Let $\bar{\mathbf{x}}^B = \frac{1}{n}\mathbf{e}$ be the barycenter of the simplex Δ, where $\mathbf{e}^T = (1, 1, ..., 1)$. We also define $\bar{\mathbf{x}}^V$ as

$$\begin{cases} \bar{\mathbf{x}}^V & \in \Delta \\ \bar{x}_i^V & = 1, \quad \text{where } i \in \arg\max_i \nabla_i f(\bar{\mathbf{x}}^B) \\ \bar{x}_j^V & = 0, \quad \text{for } j \neq i. \end{cases} \tag{26}$$

Initializing $\mathbf{x}_0$ with $\bar{\mathbf{x}}^B$ avoids initial bias to a particular solution as it considers a uniform distribution over the available objects. Since $\nabla f(\bar{\mathbf{x}}^B) = 2\mathbf{A}\bar{\mathbf{x}}^B$, the nonzero component of $\bar{\mathbf{x}}^V$ corresponds to the row of $\mathbf{A}$ with largest total sum. Therefore, it is biased to an object that is highly similar to many other objects.

The starting point for replicator dynamics is $\bar{\mathbf{x}}^{RD} = \bar{\mathbf{x}}^B$, as used for example in [30, 31]. Note that if a component of $\bar{\mathbf{x}}^{RD}$ starts at zero it will remain at zero for the entire duration of the dynamics according to (2). Furthermore, $(\bar{\mathbf{x}}^V)^T\mathbf{A}\bar{\mathbf{x}}^V = 0$ since $\mathbf{A}$ has zeros on the main diagonal, and the denominator in replicator dynamics is then zero for this point. Thus, $\bar{\mathbf{x}}^V$ is not a viable starting point for replicator dynamics.

The starting point for standard FW is $\bar{\mathbf{x}}^{FW} = \bar{\mathbf{x}}^V$, and is found experimentally to work well. As explained in convergence rate analysis, FW never performs any drop steps since the step size always satisfies $\gamma_t < \gamma_{max}$. Hence, using $\bar{\mathbf{x}}^B$ as starting point for FW will lead to a solution that has full support – this is found experimentally to hold true as well. Therefore, with FW, we use only initialization with $\bar{\mathbf{x}}^V$. With PFW and AFW, we can use both $\bar{\mathbf{x}}^B$ and $\bar{\mathbf{x}}^V$ as starting points. We denote the PFW and AFW variants by PFW-B, PFW-V, AFW-B, and AFW-V, respectively, to specify the starting point.

6 EXPERIMENTS

In this section, we describe the experimental results of the different optimization methods on synthetic, image segmentation and 20 newsgroup datasets. In addition, we study multi-start optimization for DSC which can be potentially useful for parallel computations.

6.1 Experimental Setup

The Frank-Wolfe gap (Definition 1) and the distance between two consecutive iterates are used as the stopping criterion for the FW variants and replicator dynamics. Specifically, let ϵ be the threshold, then an algorithm stops if $g_t \le \epsilon$ or if $||\mathbf{x}_{t+1} - \mathbf{x}_t|| \le \epsilon$. In the experiments we set ϵ to Python's epsilon, $\epsilon \approx 2.2 \cdot 10^{-16}$, and the cutoff parameter δ to $\delta = 2 \cdot 10^{-12}$. We denote the true number of clusters in the dataset by k and the maximum number of clusters to extract by K. For a dataset with n objects, the clustering solution is represented by a discrete n-dimensional vector $\mathbf{c}$, i.e., $c_i \in \{0, 1, ..., K-1, K\}$ for $i = 1, ..., n$. If $c_i = c_j$, then objects i and j are in the same cluster. The discrete values $0, 1, ..., K-1, K$ are called labels and represent the different clusters. Label 0 is designated to represent 'no cluster', i.e., if $c_i = 0$, then object i is unassigned. We may regularize the pairwise similarities by a shift parameter, as described in detail in [21].

To evaluate the clusterings, we compare the predicted solution and the ground truth w.r.t. Adjusted Rand Index (ARI) [19] and V-Measure [34]. The Rand index is the ratio of the object pairs that are either in the same or in different clusters, in both the predicted and ground truth solutions. V-measure is the harmonic mean of homogeneity and completeness. We may also report the Assignment Rate (AR), representing the rate of the objects assigned to a valid cluster. As we will discuss, it is common in DSC to apply a postprocessing in order to make AR equal to 1.

6.2 Experiments on Synthetic Data

For synthetic experiments, we fix $n = 200$ and $K = k = 5$, and assign the objects uniformly to one of the k clusters.

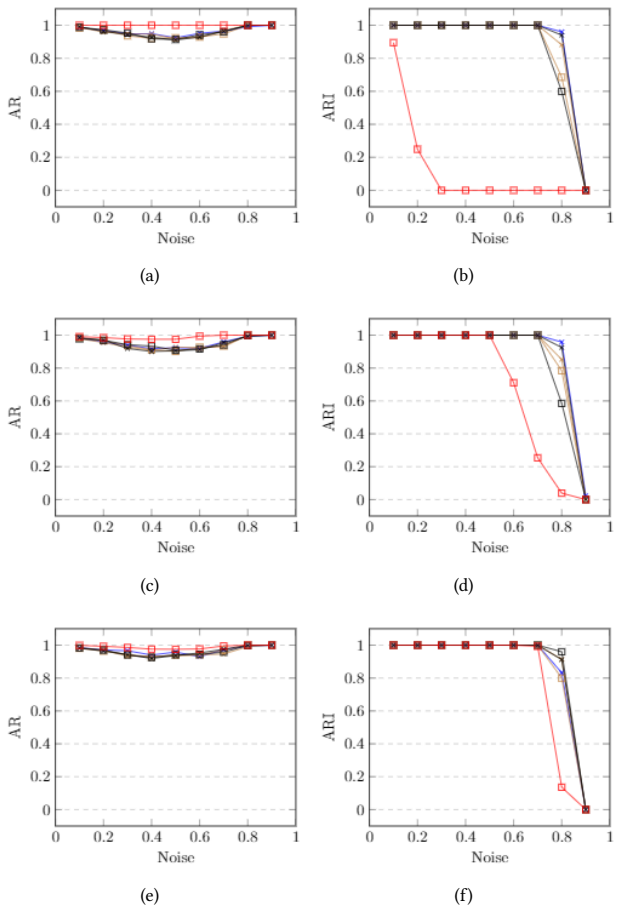


Figure 1: Results on the synthetic dataset for RD (red), FW (blue), PFW (brown), and AFW (black). PFW-B (brown) and AFW-B (black) have squares; PFW-V (brown) and AFW-V (black) have crosses. (a)-(f): $n = 200$. (a, b): $t = 400$, $\delta = 2 \cdot 10^{-12}$. (c, d): $t = 400$, $\delta = 2 \cdot 10^{-3}$. (e, f): $t = 4000$, $\delta = 2 \cdot 10^{-12}$. V-measure demonstrates very similar results to ARI.

Let $\mu \sim \mathcal{U}(0, 1)$ be uniformly distributed and

$$\begin{cases} z = 0, & \text{with probability } p \\ z = 1, & \text{with probability } 1 - p, \end{cases}$$

where p is the noise ratio. The similarity matrix $\mathbf{A} = (a_{ij})$ is then constructed as follows:

$$\begin{cases} a_{ij} = a_{ji} = z\mu, & \text{if } i \text{ and } j \text{ are in the same cluster} \\ a_{ij} = 0, & \text{otherwise.} \end{cases}$$

For each parameter configuration, we generate a similarity matrix, perform the clustering five times and then report the average results in Figure 1. We observe that the different FW methods are considerably more robust w.r.t. the noise in pairwise measurements and yield higher quality results. Also, the performance of FW variants is consistent with different parameter configurations, whereas RD is more sensitive to the number of iterations t and the cutoff parameter δ.

	t	time	AR	ARI	V-Meas.
FW	1000	0.36s	0.6325	0.4695	0.5388
	4000	1.35s	0.6885	0.4593	0.5224
	8000	2.41s	0.6969	0.4673	0.5325
PFW-B	1000	0.43s	0.7429	0.1944	0.4289
	4000	1.86s	0.6605	0.467	0.5327
	8000	2.62s	0.642	0.471	0.5335
PFW-V	1000	0.52s	0.6471	0.5178	0.5745
	4000	1.6s	0.6487	0.4565	0.5237
	8000	2.47s	0.642	0.471	0.5335
AFW-B	1000	0.35s	0.8527	0.076	0.2854
	4000	1.69s	0.6258	0.3887	0.5316
	8000	2.93s	0.6599	0.4676	0.5328
AFW-V	1000	0.46s	0.6415	0.5184	0.5736
	4000	1.38s	0.6482	0.518	0.5754
	8000	2.75s	0.6476	0.4618	0.5257
RD	1000	1.06s	1.0	0.0	0.0
	4000	4.56s	0.9081	0.1852	0.3003
	8000	11.4s	0.6997	0.4121	0.5384

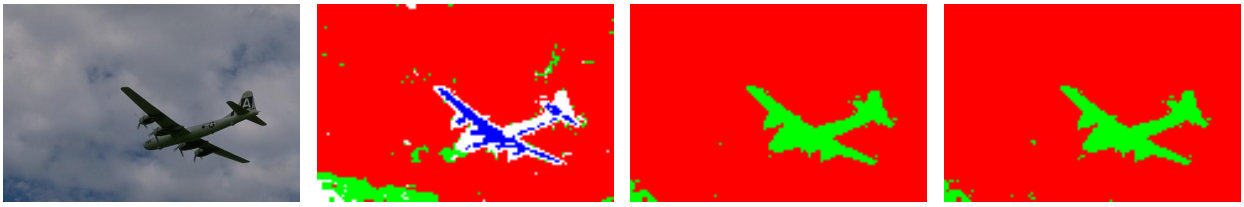
Table 1: The results on newsgroups1 dataset.

	t	time	AR	ARI	V-Meas.
FW	1000	0.37s	0.6587	0.5594	0.5929
	4000	1.38s	0.6674	0.5479	0.5866
	8000	2.6s	0.6679	0.5473	0.5864
PFW-B	1000	0.45s	0.7508	0.135	0.3555
	4000	1.57s	0.6172	0.6257	0.6364
	8000	2.06s	0.6172	0.6257	0.6364
PFW-V	1000	0.59s	0.6281	0.6095	0.6241
	4000	1.85s	0.6172	0.6257	0.6364
	8000	3.1s	0.6172	0.6257	0.6364
AFW-B	1000	0.41s	0.8653	0.0979	0.316
	4000	1.9s	0.6172	0.6257	0.6364
	8000	3.39s	0.6172	0.6257	0.6364
AFW-V	1000	0.48s	0.663	0.5548	0.5907
	4000	1.75s	0.6172	0.6257	0.6364
	8000	3.38s	0.6172	0.6257	0.6364
RD	1000	0.76s	1.0	0.0	0.0
	4000	4.67s	1.0	0.1795	0.333
	8000	13.52s	0.7585	0.4391	0.5161

Table 2: The results on newsgroups2 dataset.

6.3 Image Segmentation

Next, we study segmentation of colored images in HSV space. We define the feature vector $\mathbf{f}(i) = [v, vs\sin(h), vs\cos(h)]^T$ as in [31], where h, s, and v are the HSV values of pixel i. The similarity matrix $\mathbf{A}$ is then defined as follows. (i) Compute $||\mathbf{f}(i) - \mathbf{f}(j)||$, for every pair of pixels i and j to obtain $\mathbf{D}^{L2}$. (ii) Compute the minimax (path-based) distances [12, 14, 16] from $\mathbf{D}^{L2}$ to obtain $\mathbf{D}^{P}$. (iii) Finally, $\mathbf{A} = \max(\mathbf{D}^{P}) - \mathbf{D}^{P}$, where max is over the elements in $\mathbf{D}^{P}$ as used in [8, 17]. Figure 2 shows the segmentation results of the airplane image in Figure 2(a). The image has the dimensions 120×80, which leads to a clustering problem with $n = 120 \times 80 = 9600$ objects. We run the FW variants for $t = 10000$ and RD for $t = 250$ iterations. Due to the linear versus quadratic per-iteration time complexity



(a) Original image (b) RD (c) FW, PFW-B, AFW-V, and AFW-B (d) PFW-V

Figure 2: Original image and the segmentation results from different DSC optimization methods.

	t	time	AR	ARI	V-Meas.
FW	1000	0.41s	0.6756	0.5206	0.5879
	4000	1.35s	0.6468	0.5309	0.5975
	8000	2.63s	0.6473	0.5314	0.5978
PFW-B	1000	0.49s	0.758	0.217	0.4617
	4000	1.79s	0.6468	0.5317	0.6004
	8000	2.88s	0.6468	0.5317	0.6004
PFW-V	1000	0.56s	0.6468	0.5317	0.6004
	4000	1.96s	0.6468	0.5317	0.6004
	8000	3.71s	0.6468	0.5317	0.6004
AFW-B	1000	0.37s	0.8373	0.1381	0.3594
	4000	1.83s	0.6462	0.5316	0.6003
	8000	3.19s	0.6468	0.5317	0.6004
AFW-V	1000	0.49s	0.6468	0.5322	0.5993
	4000	1.63s	0.6468	0.5317	0.6004
	8000	2.99s	0.6468	0.5317	0.6004
RD	1000	0.86s	1.0	0.0	0.0
	4000	4.69s	0.9089	0.2212	0.3465
	8000	12.9s	0.8012	0.3526	0.4556

Table 3: The results on newsgroups3 dataset.

	t	time	AR	ARI	V-Meas.
FW	1000	0.42s	0.653	0.5097	0.5706
	4000	1.38s	0.6169	0.4672	0.5437
	8000	2.6s	0.7002	0.5014	0.5514
PFW-B	1000	0.43s	0.8092	0.2247	0.4483
	4000	1.82s	0.6697	0.6211	0.6484
	8000	3.04s	0.6697	0.6211	0.6484
PFW-V	1000	0.58s	0.6591	0.6446	0.6717
	4000	2.02s	0.6565	0.6462	0.675
	8000	2.74s	0.6565	0.6462	0.675
AFW-B	1000	0.35s	0.9041	0.1109	0.3361
	4000	1.87s	0.6687	0.6191	0.6463
	8000	3.55s	0.6697	0.6211	0.6484
AFW-V	1000	0.5s	0.6525	0.5071	0.5651
	4000	1.84s	0.6565	0.6462	0.675
	8000	3.6s	0.6565	0.6462	0.675
RD	1000	0.93s	1.0	0.0	0.0
	4000	5.46s	1.0	0.3197	0.4112
	8000	14.52s	0.8559	0.4528	0.5328

Table 4: The results on newsgroups4 dataset.

of the FW variants and RD, we are able to run FW for many more iterations. This allows us to have more flexibility in tuning the parameters and thus obtain more robust results. According to the results in Figure 2, the FW variants visually yield more meaningful and consistent results compared to RD, and separate better the airplane from the background.

6.4 Experiments on 20 Newsgroups Data

We study the clustering of different subsets of 20 newsgroups data collection. The collection consists of 18000 documents in 20 categories split into training and test subsets. We use four datasets with documents from randomly selected categories from the test subset. (i) *newsgroups1*: the set of documents in the categories soc.religion.christian, comp.os.ms-windows.misc, talk.politics.guns, alt.atheism, talk.politics.misc. (ii) *newsgroups2*: the set of documents in the categories comp.windows.x, sci.med, rec.autos, sci.crypt, talk.religion.misc. (iii) *newsgroups3*: the set of documents in the categories misc.forsale, comp.sys.mac.hardware, talk.politics.mideast, sci.electronics, rec.motorcycles. (iv) *newsgroups4*: the set of documents in the categories comp.graphics, rec.sport.hockey, sci.space, rec.sport.baseball, comp.sys.ibm.pc.hardware.

Each dataset has $k = 5$ true clusters and $1700 \leq n \leq 2000$ documents, where we use $K = 5$ for peeling off the computed clusters. We obtain the tf-idf (term-frequency times inverse document-frequency) vector for each document and then apply PCA to reduce the dimensionality to 20. We obtain the similarity matrix $\mathbf{A}$ using the cosine similarity between the PCA vectors and then shift the off-diagonal elements by 1 to ensure nonnegative entries. We set the regularization parameter to $\alpha = 15$, that seems a reasonable choice for different methods. Using smaller or larger α results in too small clusters or too slow convergence, in particular for replicator dynamics (RD).

Tables 1, 2, 3, and 4 show the results for the different datasets. We observe that different variants of FW yield significantly better results compared to replicator dynamics (RD), w.r.t. both ARI and V-Measure. In particular, PFW-V and AFW-V are computationally efficient and perform very well even with $t = 1000$. On the other hand, these methods are more robust w.r.t. different parameter settings. Since all the objects in the ground truth solutions are assigned to a cluster, the assignment rate (AR) indicates the ratio of the objects assigned (correctly or incorrectly) to a cluster during the clustering. A high AR does not necessarily indicate a good clustering solution, rather it may imply slower convergence (as happens for RD). High AR and low ARI/V-measure means assignment of many objects to wrong clusters. This is what happens for RD with $t = 1000$. As discussed in [31], it is common for DSC to perform a post processing to assign each unassigned object to the cluster which

		newsgroups1		newsgroups2		newsgroups3		newsgroups4	
Method	t	ARI	V-Meas.	ARI	V-Meas.	ARI	V-Meas.	ARI	V-Meas.
FW	1000	0.4068	0.4969	0.5158	0.5733	0.5751	0.595	0.4663	0.5252
	4000	0.4639	0.5225	0.5084	0.5699	0.572	0.5962	0.4409	0.5084
	8000	0.4766	0.5351	0.5084	0.5699	0.5729	0.5972	0.4973	0.5396
PFW-B	1000	0.2063	0.3919	0.178	0.3814	0.2764	0.4859	0.288	0.4878
	4000	0.4623	0.5324	0.5332	0.5834	0.5734	0.5992	0.587	0.6094
	8000	0.4356	0.5219	0.5332	0.5834	0.5734	0.5992	0.587	0.6094
PFW-V	1000	0.5091	0.5763	0.5331	0.5824	0.5734	0.5992	0.605	0.6226
	4000	0.4298	0.5149	0.5332	0.5834	0.5734	0.5992	0.6072	0.6268
	8000	0.4356	0.5219	0.5332	0.5834	0.5734	0.5992	0.6072	0.6268
AFW-B	1000	0.0966	0.2751	0.131	0.344	0.1782	0.3967	0.1313	0.3577
	4000	0.3162	0.4806	0.5332	0.5834	0.5734	0.5992	0.588	0.6097
	8000	0.4615	0.5319	0.5332	0.5834	0.5734	0.5992	0.587	0.6094
AFW-V	1000	0.5066	0.5744	0.5099	0.5699	0.5741	0.5983	0.4592	0.5183
	4000	0.5047	0.5719	0.5332	0.5834	0.5734	0.5992	0.6072	0.6268
	8000	0.4308	0.5148	0.5332	0.5834	0.5734	0.5992	0.6072	0.6268
RD	1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	4000	0.1892	0.3042	0.1795	0.333	0.2394	0.3575	0.3197	0.4112
	8000	0.3659	0.4937	0.4123	0.5065	0.4227	0.4973	0.4858	0.5556

Table 5: Result of different methods on 20 newsgroup datasets after post assignment of the unassigned documents. The FW variants especially PFW-V and AFW-V yield the best and computationally the most efficient results even with $t = 1000$.

it has the highest average similarity with. Specifically, let $C_0 \subseteq V$ contain the unassigned objects and $C_i \subseteq V$, $1 \leq i \leq K$, be the predicted clusters. Object $j \in C_0$ is then assigned to cluster C_i that satisfies

$$i \in \arg\max_{\ell \geq 1} \frac{1}{|C_\ell|} \sum_{p \in C_\ell} \mathbf{A}_{jp}. \tag{27}$$

Table 5 shows the performance of different methods after assigning all the documents to valid clusters, i.e., when AR is always 1. We observe that ARI and V-measure are usually consistent for pre and post assignment settings. In both cases the FW variants (especially PFW-V and AFW-V) yield the best and computationally the most efficient results. Consistent to the previous results, PFW-V and AFW-V yield high scores already with $t = 1000$. These results are consistent with the results on synthetic and image datasets.

6.5 Multi-Start Dominant Set Clustering

Finally, as a side study, we study a combination of multi-start Dominant Set Clustering with the peeling off strategy. For this, we perform the following procedure. 1. Sample a subset of objects, and use them to construct a number of starting points for the same similarity matrix. 2. Run an optimization method for each starting point. 3. Identify the nonoverlapping clusters from the solutions and remove (peel off) the corresponding objects from the similarity matrix. 4. Repeat until no objects are left or a sufficient number of clusters have been found.

This scenario can be potentially useful in particular when multiple processors can perform clustering in parallel. However, if all the different starting points converge to the same cluster, then there would be no computational benefit. Thus, here we investigate such a possibility for our optimization framework. For this, we consider the number of passes through the entire data, where a pass is defined as one complete run of the aforementioned steps. After the solutions from a pass are computed, they are sorted based on the function value $f(\mathbf{x})$. The sorted solutions are then permuted in a decreasing order, and if the support of the current solution overlaps more than 10% with the support of the other (previous) solutions, it is discarded. Each pass will therefore yield at least one new cluster. With K the maximum number of clusters to extract, there will be at most K passes. Thus in order for the method to be useful and effective, less than K passes should be performed.

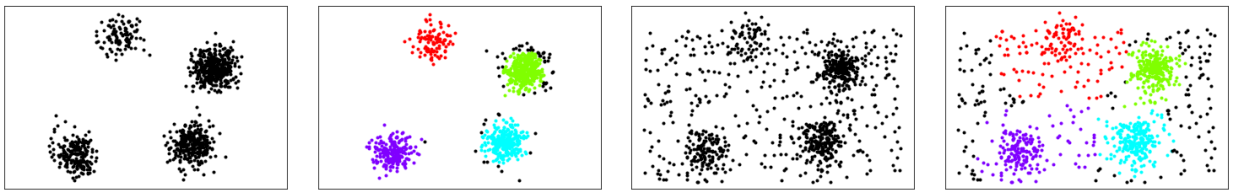
Figure 3 shows the form of the datasets used in this study. Each cluster corresponds to a two dimensional Gaussian distribution with a fixed mean and an identity co-variance matrix (see Figure 3(a)). We fix $n = 1000$ and $K = k = 4$, and use the parameter p to control the noise ratio. Set $n_1 = pn$ and $n_2 = n - n_1$. A dataset is then generated by sampling n_1 objects from a uniform distribution (background noise in Figure 3(c)), $0.1 \cdot n_2$, $0.2 \cdot n_2$, $0.3 \cdot n_2$, and $0.4 \cdot n_2$ objects from the respective Gaussians.

Let $\mathbf{D}$ be the matrix with pairwise Euclidean distances between all objects in the dataset. The similarity matrix is then defined as $\mathbf{A} = \max(\mathbf{D}) - \mathbf{D}$, similar to the image segmentation study but with a different base distance measure. The regularization parameter is set to $\alpha = 15$. To determine the starting points we sample 4 components from $\{1, ..., n\}$, denoted by i_1, i_2, i_3, i_4. The number 4 matches the number of CPUs in our machine. For a given component $i \in \{i_1, i_2, i_3, i_4\}$, we define the starting points as

$$\begin{cases} \bar{x}_i^V & = 1 \\ \bar{x}_j^V & = 0, \quad \text{for } j \neq i \end{cases}$$

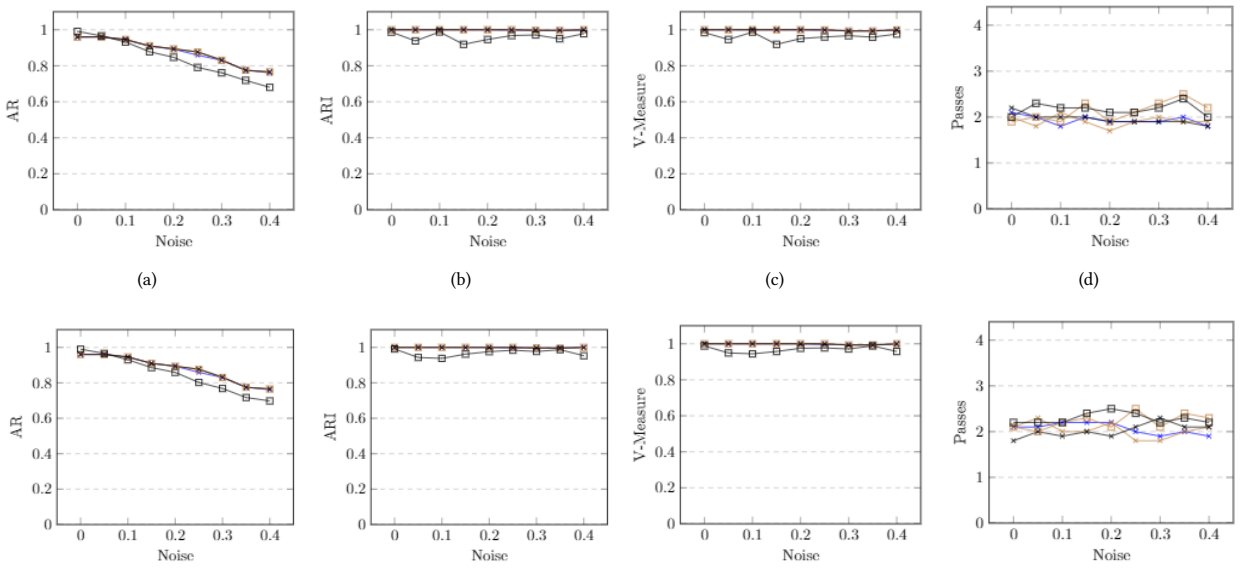
and

$$\begin{cases} \bar{x}_i^B & = 0.5 \\ \bar{x}_j^B & = 0.5/(n-1), \quad \text{for } j \neq i. \end{cases}$$



(a) No noise (b) AR 0.94, ARI 1.0, and V-measure 1.0 (c) Noise $p = 0.4$ (d) AR 0.74, ARI 1.0, and V-measure 1.0

Figure 3: Two example datasets used for multi-start study. (b) and (d) show the FW clustering results; PFW and AFW produce similar results.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4: Results of multi-start paradigm with FW (blue) , PFW (brown), and AFW (black) where PFW-B (brown) and AFW-B (black) are marked by squares; and PFW-V (brown) and AFW-V (black) are marked by crosses. The first row corresponds to UNI and the second row corresponds to DPP sampling. All optimization and sampling methods require only about two passes to compute the clusters.

FW uses only $\bar{\mathbf{x}}^V$ while PFW and AFW use both $\bar{\mathbf{x}}^V$ and $\bar{\mathbf{x}}^B$. To sample the components, we use uniform sampling and Determinantal Point Processes [22], denoted as UNI and DPP, respectively.

Figure 4 illustrates the results for the different sampling methods and starting objects. For a given dataset, sampling method, and optimization method, we generate starting objects and run the experiments 10 times and report the average results. Each method is run for $t = 1000$ iterations. For this type of dataset we do not observe any significant difference between FW, PFW, or AFW when using either DPP or UNI. It seems that AFW with $\bar{\mathbf{x}}^B$ as starting object performs slightly worse.

However, we observe that all the sampling and optimization methods require only two passes, whereas we have $K = 4$. This observation implies that the multi-start paradigm is potentially useful for computing the clusters in parallel with the different FW variants. We note that the peeling of strategy in Dominant Set Clustering is inherently sequential, thus such an achievement can be potentially very useful for parallelization.

7 CONCLUSION

We developed a unified and computationally efficient framework to employ the different variants of Frank-Wolfe methods for Dominant Set Clustering. In particular, replicator dynamics was replaced with standard, pairwise, and away-steps Frank-Wolfe algorithms when optimizing the quadratic problem defined by DSC. We provided a specialized analysis of the algorithms' convergence rates, and demonstrated the effectiveness of the framework via experimental studies on synthetic, image and text datasets. We additionally studied aspects such as multi-start Dominant Set Clustering. Our framework is generic enough to be investigated as an alternative for replicator dynamics in many other problems.

ACKNOWLEDGMENTS

The work of Morteza Haghir Chehreghani was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

[1] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation Clustering. *Mach. Learn.* 56, 1-3, 89–113.

[2] Immanuel M Bomze, Francesco Rinaldi, and Damiano Zeffiro. 2019. Active set complexity of the Away-step Frank-Wolfe Algorithm. *arXiv preprint arXiv:1912.11492* (2019).

[3] Thomas Bühler and Matthias Hein. 2009. Spectral Clustering Based on the Graph p-Laplacian. In *26th Annual International Conference on Machine Learning (ICML)*. 81–88.

[4] Samuel Rota Bulò and Marcello Pelillo. 2017. Dominant-set clustering: A review. *European Journal of Operational Research* 262, 1 (2017), 1–13.

[5] Samuel Rota Bulò, Marcello Pelillo, and Immanuel M Bomze. 2011. Graph-based quadratic optimization: A fast evolutionary approach. *Computer Vision and Image Understanding* 115, 7 (2011), 984–995.

[6] Samuel Rota Bulò, Andrea Torsello, and Marcello Pelillo. 2009. A game-theoretic approach to partial clique enumeration. *Image Vision Comput.* 27, 7 (2009), 911–922.

[7] Pak K. Chan, Martine D. F. Schlag, and Jason Y. Zien. 1994. Spectral K-way ratio-cut partitioning and clustering. *IEEE Trans. on CAD of Integrated Circuits and Systems* 13, 9 (1994), 1088–1096.

[8] Morteza Haghir Chehreghani. 2016. Adaptive trajectory analysis of replicator dynamics for data clustering. *Machine Learning* 104, 2-3 (2016), 271–289.

[9] Morteza Haghir Chehreghani. 2022. Shift of Pairwise Similarities for Data Clustering. *Machine Learning* (2022). https://doi.org/10.1007/s10994-022-06189-6

[10] Morteza Haghir Chehreghani, Hassan Abolhassani, and Mostafa Haghir Chehreghani. 2008. Improving density-based methods for hierarchical clustering of web pages. *Data Knowl. Eng.* 67, 1 (2008), 30–50.

[11] Ross Cressman and Yi Tao. 2014. The replicator equation and other game dynamics. *Proceedings of the National Academy of Sciences* 111 (2014), 10810–10817.

[12] Bernd Fischer and Joachim M. Buhmann. 2003. Path-based clustering for grouping of smooth curves and texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, 4 (2003), 513–518.

[13] Marguerite Frank and Philip Wolfe. 1956. An algorithm for quadratic programming. *Naval research logistics quarterly* 3, 1-2 (1956), 95–110.

[14] Morteza Haghir Chehreghani. 2017. Classification with Minimax Distance Measures. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI Press, 1784–1790.

[15] Morteza Haghir Chehreghani. 2017. Clustering by Shift. In *IEEE International Conference on Data Mining, ICDM*. 793–798.

[16] Morteza Haghir Chehreghani. 2020. Unsupervised representation learning with Minimax distance measures. *Machine Learning* 109, 11 (2020), 2063–2097.

[17] Morteza Haghir Chehreghani and Mostafa Haghir Chehreghani. 2020. Learning representations from dendrograms. *Machine Learning* 109, 9-10 (2020), 1779–1802.

[18] Matthias Hein and Thomas Bühler. 2010. An Inverse Power Method for Nonlinear Eigenproblems with Applications in 1-Spectral Clustering and Sparse PCA. In *Advances in Neural Information Processing Systems (NIPS)*. 847–855.

[19] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2, 1 (1985), 193–218.

[20] Anil K Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern recognition letters* 31, 8 (2010), 651–666.

[21] Carl Johnell. 2020. *Frank-Wolfe Optimization for Dominant Set Clustering*. Master's thesis. Chalmers University of Technology, Department of Computer Science and Engineering.

[22] Alex Kulesza et al. 2012. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning* 5, 2–3 (2012), 123–286.

[23] Simon Lacoste-Julien. 2016. Convergence rate of Frank-Wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345* (2016).

[24] Simon Lacoste-Julien and Martin Jaggi. 2015. On the global linear convergence of Frank-Wolfe optimization variants. In *Advances in Neural Information Processing Systems*. 496–504.

[25] Frank Lin and William W. Cohen. 2010. Power Iteration Clustering. In *27th International Conference on Machine Learning (ICML)*. 655–662.

[26] Hairong Liu, Longin Jan Latecki, and Shuicheng Yan. 2013. Fast Detection of Dense Subgraphs with Iterative Shrinking and Expansion. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 9 (2013), 2131–2142.

[27] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*. MIT Press, 849–856.

[28] Karen M. Page and Martin A. Nowak. 2002. Unifying Evolutionary Dynamics. *Journal of Theoretical Biology* 219 (2002), 93 – 98.

[29] Massimiliano Pavan and Marcello Pelillo. 2003. Dominant sets and hierarchical clustering. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, Vol. 2. IEEE, 362–369.

[30] Massimiliano Pavan and Marcello Pelillo. 2003. A new graph-theoretic approach to clustering and segmentation. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, Vol. 1. IEEE, I–I.

[31] Massimiliano Pavan and Marcello Pelillo. 2007. Dominant sets and pairwise clustering. *IEEE transactions on pattern analysis and machine intelligence* 29, 1 (2007), 167–172.

[32] Marcello Pelillo. 2001. *Replicator dynamics in combinatorial optimization*. 2197–2209.

[33] Sashank J Reddi, Suvrit Sra, Barnabás Póczos, and Alex Smola. 2016. Stochastic frank-wolfe methods for nonconvex optimization. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 1244–1251.

[34] Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 410–420.

[35] Josep Sardanyés i Cayuela. 2009. *Dynamics, evolution and information in nonlinear dynamical systems of replicators*. Ph. D. Dissertation. Universitat Pompeu Fabra, ISBN: 9788469274538.

[36] Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 8 (2000), 888–905.

[37] Sylvain Sorin. 2020. Replicator dynamics: Old and new. *Journal of Dynamics and Games* 7 (2020), 365.

[38] Erik Thiel, Morteza Haghir Chehreghani, and Devdatt P. Dubhashi. 2019. A Non-Convex Optimization Approach to Correlation Clustering. In *The Thirty-Third AAAI Conference on Artificial Intelligence*. 5159–5166.