

Simplex Frank-Wolfe: Linear Convergence and Its Numerical Efficiency for Convex Optimization over Polytopes

Haoning Wang¹, Houduo Qi ^{*2}, and Liping Zhang¹

¹Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China

²Department of Data Science and Artificial Intelligence, and Department of Applied Mathematics, The Hong Kong Polytechnic University, Hong Kong

Abstract

We investigate variants of the Frank-Wolfe (FW) algorithm for smoothing and strongly convex optimization over polyhedral sets, with the goal of designing algorithms that achieve linear convergence while minimizing per-iteration complexity as much as possible. Starting from the simple yet fundamental unit simplex, and based on geometrically intuitive motivations, we introduce a novel oracle called Simplex Linear Minimization Oracle (SLMO), which can be implemented with the same complexity as the standard FW oracle. We then present two FW variants based on SLMO: Simplex Frank-Wolfe and the refined Simplex Frank-Wolfe (rSFW). Both variants achieve a linear convergence rate for all three common step-size rules. Finally, we generalize the entire framework from the unit simplex to arbitrary polytopes. Furthermore, the refinement step in rSFW can accommodate any existing FW strategies such as the well-known away-step and pairwise-step, leading to outstanding numerical performance. We emphasize that the oracle used in our rSFW method requires only one more vector addition compared to the standard LMO, resulting in the lowest per-iteration computational overhead among all known Frank-Wolfe variants with linear convergence.

keywords: Frank-Wolfe algorithm, conditional gradient methods, linear convergence, convex optimization, first-order methods, linear programming

*Corresponding author: houduo.qi@polyu.edu.hk

1 Introduction

Over the past decades, Frank-Wolfe (FW) algorithms [10] (a.k.a. conditional gradients [25]) have been extensively investigated due to its lower per-iteration complexity compared to projected or proximal gradient-based methods, in particular for large-scale machine learning applications and sparse optimization. This topic has been comprehensively covered in several recent publications including [3, 4, 27] and [24, Chapter 7],[2, Chapter 10], to just name a few. The key step in FW algorithms is Linear Minimization Oracle (LMO). We refer to [23] for (worst-case) complexity analysis for general LMOs. One of the most often cited examples is LMO over the unit Simplex $S_n := \{\mathbf{x} \in \mathbb{R}^n \mid \sum x_i = 1, \mathbf{x} \geq 0\}$. Projection onto S_n is much expensive than LMO over S_n . Research effort has been on developing LMOs that may lead to linear convergence while keeping the computation of each LMO as low as possible. Therefore, the total computational complexity of a typical FW-type algorithm can be calculated as follows.

Total Computation = (#Iterations) $\times$ (Computation of LMOs per iteration).

Note that some existing algorithms may require more than one LMO each iteration. The purpose of this paper is to propose a new LMO, whose computational complexity is probably the cheapest among all existing algorithms. Furthermore, it also guarantees a linear convergence rate comparable to the known ones for the convex optimization over a polytope:

$$\min f(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{P} = \text{Conv}(\mathcal{V}), \quad (1.1)$$

where $\mathcal{V} \subseteq \mathbb{R}^n$ is a *finite* set of vectors that we call *atoms*. For the moment, we only assume $f : \mathcal{C} \mapsto \mathbb{R}$ is convex and differentiable for the convenience of discussion below. Here, $\mathcal{C}$ is an open set containing $\mathcal{P}$. Later, we will enforce strong convexity as well as other properties for our analysis.

1.1 Related Work

There are a large number of publications that directly or remotely motivated this work. We are only able to list a few of them below with some critical analysis. Given an LMO, the original FW algorithm [10] states as:

$$\begin{cases} \mathbf{y}_k = \text{LMO}(\nabla f(\mathbf{x}_{k-1}), \mathcal{P}) \in \arg \min \{ \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{y} \rangle \mid \mathbf{y} \in \mathcal{P} \}, \\ \mathbf{x}_k = (1 - \delta_k)\mathbf{x}_{k-1} + \delta_k\mathbf{y}_k, \quad \delta_k \in (0, 1], \end{cases}$$

where δ_k is a steplength often satisfying certain conditions [6, 18, 19]. One of the key advantages of the FW method over the well-known projected gradient

method is its lower cost per iteration in many common scenarios, such as the simplex [6], flow polytope [7, 21], spectrahedron [18, 12], and nuclear norm ball [20]. This efficiency makes the FW method particularly advantageous for large-scale problems. Numerous studies [19, 23, 11] have demonstrated that the convergence rate of the FW method is $\mathcal{O}(\frac{1}{k})$ and that this rate is generally not improvable, except for some special cases, e.g., when the optimal solution lies in the interior of the constraint set [16]. In fact, there exist examples for which the convergence rate of the FW method does not improve even when the objective function is strongly convex, see [19, 23].

Therefore, modifications on the original FW method must be made in order to achieve linear convergence rate. Significant advances have been made along this line of research and there exist a large number of variants of FW methods that enjoy linear convergence rate, see [4, Chapter 3]. The well-known ones include FW-method with away-step (AFW) and the pairwise FW (PFW) [22, 9, 13]. Most of those modified methods can be cast in the following framework:

$$\begin{cases} \mathbf{y}_k = \text{LMO}(\nabla f(\mathbf{x}_{k-1}), \mathcal{P}_k) \in \arg \min \{ \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{y} \rangle \mid \mathbf{y} \in \mathcal{P}_k \}, \\ \mathbf{g}_k = \text{direction-correction satisfying certain conditions}, \\ \mathbf{x}_k = (1 - \delta_k)\mathbf{x}_{k-1} + \delta_k(\mathbf{y}_k + \mathbf{g}_k), \quad \delta_k \in (0, 1], \end{cases} \quad (1.2)$$

where $\mathcal{P}_k \subseteq \mathcal{P}$ is a well-constructed convex subset of $\mathcal{P}$ at the current iterate $\mathbf{x}_{k-1}$. This framework has a flexibility for more technical strategies to be added. For example, one may mix $\mathbf{y}_k$ and $\mathbf{g}_k$ through certain combinations with some linesearch strategies. Both AFW and PFW make use of such flexibility. One major concern is that the computation of LMO over $\mathcal{P}_k$ may be significantly higher than that over $\mathcal{P}$. This is the case when $\mathcal{P}$ is Simplex-like polytopes including S_n .

In a significant development aiming to address this issue, Garber and Hazan [14] proposed the methodology of LLOO (Local Linear Optimization Oracle), where $\mathcal{P}_k$ is the intersection of S_n and a ℓ_1 -ball:

$$\mathcal{P}_k = S_n \cap B_1(\mathbf{x}_{k-1}, d_k), \quad \text{with} \quad B_1(\mathbf{x}, d) = \{ \mathbf{y} \mid \|\mathbf{x} - \mathbf{y}\|_1 \leq d \}.$$

A key result is that LMO over this $\mathcal{P}_k$ is LLOO, which is referred to as ℓ_1 -LMO. Hence, linear convergence follows when the radius is exponentially reduced at each iteration under the strongly convex setting. We note that the framework of LLOO can be cast as a special case of Shrinking Conditional Gradient Methods (sCndG) by Lan [23, Eq. 3.34] and [24, Alg. 7.2], where an arbitrary norm is used. The LLOO framework does not require the step of $\mathbf{g}_k$ in (1.2). To understand its actual performance, Fig. 2a in Sect. 5 illustrates

its computational time in comparison to the LMO over the unit simplex as well a projection algorithm.

It can be clearly observed that the time taken by ℓ_1 -LMO is roughly same as the projection method, but is significantly slower (e.g., $100\times$ slower when n gets big) than $\text{LMO}(\mathbf{c}, S_n)$. There is a deep reason behind this performance and it can be best appreciated from the perspective of geometric intuition by considering the situation of $n = 3$. Fig. 1a illustrates the intersection of ℓ_1 -ball with the unit simplex. Note that for any point $\mathbf{x} \in S_3$, the intersection of ℓ_1 -ball with the hyperplane containing S_3 forms a regular hexagon. As the center $\mathbf{x}$ and radius d vary, the shape corresponding to the intersection of this regular hexagon and unit simplex becomes more complex, as shown by the blue region in Fig. 1a. This increased complexity of the constraint set makes solving ℓ_1 -LMO more challenging. From a computational point of view, ℓ_1 -LMO requires a sorting procedure [14] to handle the complexity and hence takes up too much time.

We also observe that LLOO/sCndG framework was largely omitted from the recent surveys [3, 4, 27] probably due to the following two reasons. One is on the concern of computational cost per iteration discussed above. The other is that there lacks flexibility of incorporating existing accelerating strategies such as Away-steps. In this paper, we propose a new framework of constructing the subset $\mathcal{P}_k$ that is not based on any norms. In the meantime, the computational cost per iteration is reduced probably to minimum and there is flexibility to include various acceleration strategies. We explain our framework below.

1.2 Simplex LMO and Simplex FW: a New Proposal

Ideally, we would like our subset $\mathcal{P}_k$ to be like the simplex S_n so that linear optimization over it can be fast executed. We here introduce the *simplex ball* $S(\mathbf{x}, d)$ with centroid $\mathbf{x}$ and radius $d > 0$ (detailed definition later). It coincides with the *atom norm* of the unit simplex as introduced by [5]. Moreover, the unit simplex S_n is a simplex ball. A very useful property is that the intersection of two simplex balls is again a simplex ball:

$$S(\mathbf{x}_1, d_1) \cap S(\mathbf{x}_2, d_2) = S(\mathbf{x}_3, d_3),$$

where $\mathbf{x}_3$ and d_3 can be cheaply computed from $(\mathbf{x}_i, d_i)$, $i = 1, 2$. This property is illustrated in Fig. 1b. Consequently, given $\mathbf{x} \in S_n$, a radius $d > 0$ and $\mathbf{c} \in \mathbb{R}^n$, we define the Simplex Linear Minimization Oracle $\text{SLMO}(\mathbf{x}, d, \mathbf{c})$ by

$$\text{SLMO}(\mathbf{x}, d, \mathbf{c}) = \arg \min_{\mathbf{y}} \left\{ \langle \mathbf{c}, \mathbf{y} \rangle \mid \mathbf{y} \in S_n \cap S(\mathbf{x}, d) \right\}.$$

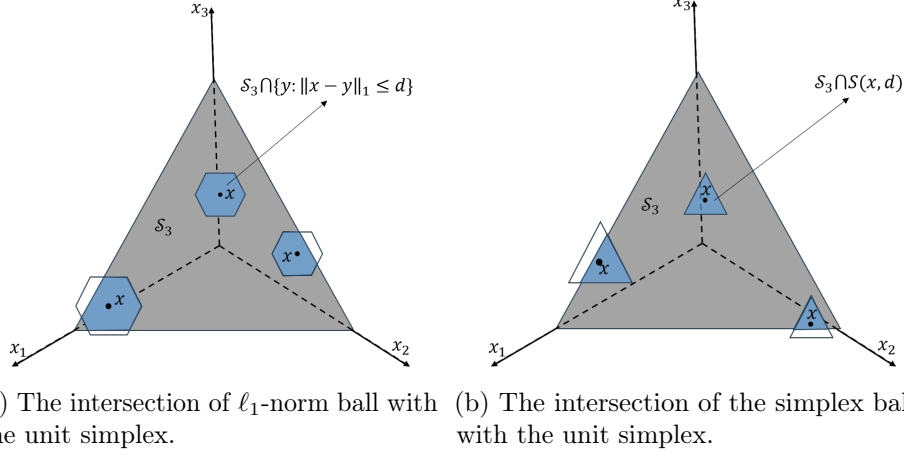


Figure 1: Schematic diagram of the feasible sets for solving ℓ_1 -LMO and SLMO when $n = 3$.

Since the constraint is again a simplex ball, SLMO has a closed-form formula (see Alg. 1) and its complexity is roughly same as $\text{LMO}(\mathbf{c}, S_n)$. Furthermore, we prove that SLMO is LLOO in Lemma 3.2. The consequence is that linearly convergent algorithm can be developed by following the template in [14]. This part forms the first contribution of the paper.

Casting SLMO as an instance of LLOO does not benefit too much in terms of computational efficiency because, as a common practice in FW methods, direction-correction step in (1.2) is essential in improving numerical performance. To accommodate this need, we make two additions. The first one is on a flexible rule to update the radius of the simplex ball. Any lower-bound for the objective function f is permitted and the lower-bound by SLMO is a choice. We opt for the use of the best lower-bound available at the current iterate. This leads to the Simplex Frank-Wolfe (SFW) method in Alg. 2, which is proved to be linearly convergent in Thm. 3.4.

This second addition makes use of an important observation that SLMO can be split into two parts. The first part is the construction of the simplex ball and the second part is linear optimization over the simplex ball. Linear optimization is much cheaper than construction of the simplex ball. It would be much economical if we perform linear optimization a few more times for each simplex ball:

$$\mathbf{p}_k \approx \arg \min_{\mathbf{y}} \left\{ f(\mathbf{y}) \mid \mathbf{y} \in S_n \cap S(\mathbf{x}_{k-1}, d_{k-1}) \right\}.$$

This $\mathbf{p}_k$ functions like the direction-correction used in the general framework (1.2). This allows us to take advantage of existing FW algorithms. For example, AFW and PFW can be used for this part. This leads to our refined SFW method (rSFW) (see Alg. 3 and Alg. 6). We emphasize that the oracle used in our rSFW method requires only one additional vector addition compared to the standard LMO, whose computational complexity is probably the cheapest among all existing FW-type algorithms. Our numerical experiments show that rSFW with Away step and Pairwise steps improves its performance significantly. The resulting algorithmic scheme is hence different from LLOO scheme and we provide complete convergence analysis. This part may be treated as our second contribution.

Our third contribution is on extending the simplex case to general polytope case. We will make use of some fundamental connections between them established in [14]. Since the simplex ball is not defined from any norm, some part of the extension is highly non-trivial. In particular, the iteration complexity of the extended SFW depends only on the problem dimension n instead of the number of extreme points N of $\mathcal{P}$, see Thm. 4.4. Computationally, this can all be achieved for three popular polytopes: Hypercube, Flow polytope and ℓ_1 -ball.

Our final part is to address the implementation issues including adaptive backtracking techniques on choosing the problem parameters L and μ , and incorporating Away-FW and Pairwise-FW steps to SFW methods. Numerical experiments demonstrate that our SFW methods are highly competitive.

1.3 Organization

In the preceding discussion, we only focus on the framework (1.2) that may lead to linearly convergent FW methods. We avoid specifying the actual conditions on f because various conditions can ensure such linear rate. In Section 2, we describe such conditions as well as some background on polytopes. We will explain the key concept of LLOO proposed in [14]. Section 3 contains the detailed development of SLMO and the resulting Simplex FW methods (SFW and rSFW) for the case $\mathcal{P} = S_n$. The extension to the general polytope case is conducted in Section 4. Lengthy proofs are moved to the Supplement for the benefit of readability of the paper. Section 5 reports some illustrative examples to demonstrate the advantage of SFW methods over some existing algorithms. We conclude the paper in Section 6.

2 Notation and Background

2.1 Notation

We employ lower-case letters, bold lower-case letters, and capital letters to denote scalars, vectors, and matrices, respectively (e.g., $x, \mathbf{x}$ and X). For two column vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\max\{\mathbf{x}, \mathbf{y}\}$ is a new column vector that takes the component-wise maximum of $\mathbf{x}$ and $\mathbf{y}$. The vector $\min\{\mathbf{x}, \mathbf{y}\}$ is similarly defined. For vectors, we denote the standard Euclidean norm by $\|\cdot\|$ and the standard inner-product by $\langle \cdot, \cdot \rangle$. For a vector $\mathbf{x} \in \mathbb{R}^n$, a subset $C \subseteq \mathbb{R}^n$, and $\tau > 0$, we define

$$\mathbf{x} + C := \{\mathbf{x} + \mathbf{y} \mid \mathbf{y} \in C\} \quad \text{and} \quad \tau C := \{\tau \mathbf{y} \mid \mathbf{y} \in C\},$$

where “ $:=$ ” means “define”.

We let $\mathbb{B}(\mathbf{x}, r)$ to denote the Euclidean ball of radius r centered at $\mathbf{x}$. For matrices, we let $\|\cdot\|$ denote the spectral norm. For a vector $\mathbf{x} \in \mathbb{R}^n$, we use x_i or $x(i)$ to denote the i -th component. For a matrix A , we use $A(i)$ to denote the i -th row of A . The vector $\mathbf{1}_n$ represents a vector with all entries equal to 1, and $\mathbf{e}_i$ is the standard i th unit vector in $\mathbb{R}^n$ which takes value 1 at its i th position and 0 elsewhere. Given a set $\mathcal{V}$, we denote its convex hull as $\text{Conv}\{\mathcal{V}\}$. For any positive integer n , we use the notation $[n]$ to represent the set $\{1, \dots, n\}$. We use $S_n := \{\mathbf{x} \in \mathbb{R}^n \mid \sum_{i=1}^n x_i = 1, \mathbf{x} \geq 0\}$ to denote the unit simplex.

2.2 Smoothness, Strong Convexity and Stepsizes

Throughout the paper, we will assume L -smoothness and μ -strong convexity of f .

Definition 1 (Smooth function). *We say that a function $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ is L -smooth over a convex set $\mathcal{P} \subset \mathbb{R}^n$, if for every $\mathbf{x}, \mathbf{y} \in \mathcal{P}$ there holds*

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \mathbf{y} - \mathbf{x}, \nabla f(\mathbf{x}) \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{y}\|^2.$$

Definition 2 (Strongly convex function). *We say that a function $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ is μ -strongly convex over a convex set $\mathcal{P} \subset \mathbb{R}^n$, if for every $\mathbf{x}, \mathbf{y} \in \mathcal{P}$ there holds*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \mathbf{y} - \mathbf{x}, \nabla f(\mathbf{x}) \rangle + \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}\|^2.$$

The above definition combined with first order optimality conditions imply that for a μ -strongly convex function f , if $\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{P}} f(\mathbf{x})$, then for any $\mathbf{x} \in \mathcal{P}$ we have

$$f(\mathbf{x}) - f^* \geq \frac{\mu}{2} \|\mathbf{x} - \mathbf{x}^*\|^2. \quad (2.1)$$

This property, while weaker than strong convexity, is essential for demonstrating linear convergence rather than relying solely on strong convexity. A natural generalization of this property is known as the quadratic growth property.

There are three popular step size strategies:

1. Simple step size:

$$\delta_k = 2/(k+1), \quad k = 1, \dots \quad (2.2)$$

2. Line-search step size:

$$\delta_k = \arg \min_{\delta \in [0,1]} f((1-\delta)\mathbf{x}_{k-1} + \delta\mathbf{y}_k), \quad k = 1, \dots \quad (2.3)$$

3. Short step size:

$$\delta_k = \min \left\{ 1, \frac{\langle \nabla f(\mathbf{x}_{k-1}), \mathbf{x}_{k-1} - \mathbf{y}_k \rangle}{L \|\mathbf{x}_{k-1} - \mathbf{y}_k\|^2} \right\}, \quad k = 1, \dots \quad (2.4)$$

For the three step size strategies described above, it can be shown that the standard Frank-Wolfe method exhibits the following convergence rates. For a detailed proof, refer to the modern surveys by [19], [23] or [11].

Theorem 2.1. *Let $\{\mathbf{x}_k\}$ be the sequences generated by standard FW method with step size policy for $\{\delta_k\}$ in (2.2), (2.3), or (2.4). Then, for $k \geq 1$, we have*

$$f(\mathbf{x}_k) - f^* \leq \langle \nabla f(\mathbf{x}_k), \mathbf{x}_k - \mathbf{y}_{k+1} \rangle \leq 2LD^2/(k+1), \quad (2.5)$$

where $\mathbf{y}_{k+1} = \text{LMO}(\nabla f(\mathbf{x}_k), \mathcal{P})$ and D is the diameter of $\mathcal{P}$.

2.3 Quantities of Polytope

The quantities reviewed in this part are well defined and investigated in [14] and they are mainly used in the extension of SFW to polytopes.

Let $\mathcal{P}$ be a polytope described by linear equations and inequalities, specifically $\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n | A_1 \mathbf{x} = \mathbf{b}_1, A_2 \mathbf{x} \leq \mathbf{b}_2\}$, where $A_2 \in \mathbb{R}^{m \times n}$. Without loss

of generality, we assume that all rows of A_2 have been scaled to possess a unit l_2 norm. We denote the set of vertices of $\mathcal{P}$ as $\mathcal{V}(\mathcal{P})$ and let $N = |\mathcal{V}(\mathcal{P})|$ represent the number of vertices.

Next, we introduce several geometric parameters related to $\mathcal{P}$ that will naturally arise in our analysis. The Euclidean diameter of $\mathcal{P}$ is defined as $D(\mathcal{P}) = \max_{\mathbf{x}, \mathbf{y} \in \mathcal{P}} \|\mathbf{x} - \mathbf{y}\|$. We define

$$\xi(\mathcal{P}) = \min_{\mathbf{v} \in \mathcal{V}(\mathcal{P})} (\min \{b_2(j) - A_2(j)\mathbf{v} \mid j \in [m], A_2(j)\mathbf{v} < b_2(j)\}).$$

This means that for any inequality constraint defining the polytope and for a given vertex, that vertex either satisfies the constraint with equality or is at least $\xi(\mathcal{P})$ units away from satisfying it with equality. Let $r(A_2)$ denote the row rank of A_2 , and let $\mathbb{A}(\mathcal{P})$ represent the set of all $r(A_2) \times n$ matrices with linearly independent rows selected from the rows of A_2 . We then define $\psi(\mathcal{P}) = \max_{M \in \mathbb{A}(\mathcal{P})} \|M\|$. Finally, we introduce condition number of $\mathcal{P}$ as

$$\eta(\mathcal{P}) = \psi(\mathcal{P})D(\mathcal{P})/\xi(\mathcal{P}). \quad (2.6)$$

It is important to note that the translation, rotation and scaling of the polytope $\mathcal{P}$ are invariant to $\eta(\mathcal{P})$. For convenience we use $\mathcal{V}, D, \xi, \psi, \eta$ without explicitly mentioning the polytope when $\mathcal{P}$ is clear from context. It is worth noting that in many relevant scenarios—particularly in cases where efficient algorithms exist for linear optimization over the given polytope—estimating the parameters D, ξ, ψ is often straightforward. This is particularly true in convex domains encountered in combinatorial optimization, such as flow polytopes, matching polytopes, and matroid polytopes, among others. Furthermore, our algorithm relies primarily on the parameter η and D .

2.4 LLOO

A major concept proposed by Garber and Hazan [14, Def. 2.5] is LLOO. Consider the problem (1.1). We say a procedure $\mathcal{A}(\mathbf{x}, d, \mathbf{c})$, where $\mathbf{x} \in \mathcal{P}$, $d > 0$, $\mathbf{c} \in \mathbb{R}^n$, is an LLOO with parameter $\rho \geq 1$ for polytope $\mathcal{P}$ if $\mathcal{A}(\mathbf{x}, d, \mathbf{c})$ returns a feasible point $\mathbf{p} \in \mathcal{P}$ such that

- (i) $\langle \mathbf{y}, \mathbf{c} \rangle \geq \langle \mathbf{p}, \mathbf{c} \rangle$ for all $\mathbf{y} \in \mathbb{B}(\mathbf{x}, d) \cap \mathcal{P}$, and
- (ii) $\|\mathbf{x} - \mathbf{p}\| \leq \rho d$.

Suppose the optimal solution $\mathbf{x}^*$ of (1.1) is contained in $\mathbb{B}(\mathbf{x}, d)$ and LLOO $\mathcal{A}(\mathbf{x}, d, \nabla f(\mathbf{x}))$ return a feasible point $\mathbf{p} \in \mathcal{P}$. The convexity of f

implies the following.

$$\begin{aligned} f(\mathbf{x}^*) &\geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{x}^* - \mathbf{x} \rangle \\ &\geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{p} - \mathbf{x} \rangle \quad (\text{by } \mathbf{x}^* \in \mathbb{B}(\mathbf{x}, d) \text{ and Property LLOO(i)}) \end{aligned}$$

That is, LLOO naturally provides a lower bound for the optimal objective. Such lower bounds will be used in our updating scheme of the radius d . We also note that LLOO $\mathcal{A}(\mathbf{x}, d, \nabla f(\mathbf{x}))$ often return an optimal solution over a subset $\mathcal{P}_k$, which should be constructed in (1.2) bearing in mind of its solution efficiency.

Given an LLOO procedure available, a general FW framework can be developed for it to enjoy a linear convergence rate over general polytope $\mathcal{P}$ provided f being L -smooth and μ -strongly convex, see [14, Thm. 4.1]. It is proved that ℓ_1 -LMO is an LLOO over the simplex polytope S_n . The framework is then extended to general polytope. As we discussed in Introduction, ℓ_1 -LMO is much less efficient than the original LMO over the simplex polytope S_n . This is the one of the motivations for us to develop the simplex LMO below.

3 Simplex FW Method

This section is solely devoted to the case of simplex polytope: Problem (1.1) with $\mathcal{P} = S_n$. We will then extend the obtained results to general polytopes in the next section. We start with the introduction of simplex ball.

3.1 Simplex Ball and Simplex-based Linear Minimization Oracle

In this subsection, we will formally define the concept of the simplex ball and present some of its useful properties. Following this, we will introduce the Simplex-based Linear Minimization Oracle (SLMO) and provide an efficient algorithm for solving it.

Definition 3 (Simplex ball). *Let $S_0 := S_n - \frac{1}{n}\mathbf{1}_n$. For any $\mathbf{x} \in \mathbb{R}^n$ and $d > 0$, we define $S(\mathbf{x}, d)$ as the simplex ball of radius d centered at $\mathbf{x}$ by*

$$S(\mathbf{x}, d) := \mathbf{x} + (nd)S_0 = \left\{ (\mathbf{x} - d\mathbf{1}_n) + nd\boldsymbol{\lambda} \mid \boldsymbol{\lambda} \in S_n \right\}. \quad (3.1)$$

The following properties of the simplex ball are crucial to our development. The proof is moved to the Supplement A.

Lemma 3.1. *Given $\mathbf{x} \in S_n$ and $d > 0$, we have*

- (1) *The unit simplex is a simplex ball, i.e., $S_n = S(\frac{1}{n}\mathbf{1}_n, \frac{1}{n})$. Moreover, we have*

$$S(\mathbf{x}, d) = \left\{ \mathbf{x} + d\mathbf{r} \mid \mathbf{r} \in \text{Conv}\{n\mathbf{e}_i - \mathbf{1}_n : i \in [n]\} \right\}. \quad (3.2)$$

- (2) *The intersection of two simplex balls, if nonempty, is again a simplex ball. In particular,*

$$S_n \cap S(\mathbf{x}, d) = S(\hat{\mathbf{x}}, \hat{d}) \quad \text{where} \quad \begin{cases} \hat{d} = \frac{\sum_{i=1}^n \min\{d, x_i\}}{n} \\ \hat{x}_i = \max\{x_i, d\} + \hat{d} - d, \quad i \in [n]. \end{cases} \quad (3.3)$$

Moreover, for $\mathbf{x}_1, \mathbf{x}_2 \in S_n$ and radius $d_1, d_2 > 0$ such that $S(\mathbf{x}_1, d_1) \cap S(\mathbf{x}_2, d_2) \neq \emptyset$, it holds

$$S(\mathbf{x}_1, d_1) \cap S(\mathbf{x}_2, d_2) = S(\mathbf{x}_3, d_3),$$

where

$$d_3 = \frac{1 + \sum_{i=1}^n \min\{d_1 - x_1(i), d_2 - x_2(i)\}}{n}, \quad (3.4)$$

$$x_3(i) = \max\{x_1(i) - d_1, x_2(i) - d_2\} + d_3, \quad i \in [n]$$

Consequently, we have $d_3 \leq \min\{d_1, d_2\}$.

- (3) *The linear optimization over a simplex ball has the following closed-form solution:*

$$\mathbf{y}^* := \mathbf{x} + (nd) \left(\mathbf{e}_{i^*} - \frac{\mathbf{1}_n}{n} \right) \in \arg \min_{\mathbf{y} \in S(\mathbf{x}, d)} \langle \mathbf{c}, \mathbf{y} \rangle \quad \text{with} \quad i^* = \arg \min_{i \in [n]} c_i.$$

- (4) *The diameter of the simplex ball $S(\mathbf{x}, d)$ is $\sqrt{2}nd$, i.e., $\max_{\mathbf{y}_1, \mathbf{y}_2 \in S(\mathbf{x}, d)} \|\mathbf{y}_1 - \mathbf{y}_2\| = \sqrt{2}nd$.*

- (5) *For any point $\mathbf{y} \in S_n$, if $\|\mathbf{x} - \mathbf{y}\| \leq d$, then $\mathbf{y} \in S(\mathbf{x}, d)$. Moreover, for any point $\mathbf{y} \in S(\mathbf{x}, d)$, we have $\|\mathbf{y} - \mathbf{x}\| \leq nd$.*

We now give a formal definition of our LMO based on simplex ball.

Definition 4 (SLMO). *Given a linear objective $\mathbf{c} \in \mathbb{R}^n$, radius $d > 0$ and a point $\mathbf{x} \in S_n$, a solution $\mathbf{y}^* \in \text{SLMO}(\mathbf{x}, d, \mathbf{c})$ is called simplex-based linear minimization oracle if it solves the following optimization problem*

$$\min \langle \mathbf{y}, \mathbf{c} \rangle \quad \text{s.t.} \quad \mathbf{y} \in S(\mathbf{x}, d) \cap S_n. \quad (3.5)$$

We have proved in Lemma 3.1(5) that $\mathbb{B}(\mathbf{x}, d) \subseteq S(\mathbf{x}, d) \subseteq \mathbb{B}(\mathbf{x}, nd)$. Therefore, for any $\mathbf{y} \in \mathbb{B}(\mathbf{x}, d) \cap S_n$, we must have $\langle \mathbf{y}, \mathbf{c} \rangle \geq \langle \mathbf{y}^*, \mathbf{c} \rangle$. This is the first property of LLOO. Moreover, since both $\mathbf{x}, \mathbf{y}^* \in S(\mathbf{x}, d)$, we must have $\|\mathbf{x} - \mathbf{y}^*\| \leq \rho d$ with $\rho = n$. This leads to the following key result.

Lemma 3.2. *Given $\mathbf{x} \in \mathcal{P}$, $d > 0$ and $\mathbf{c} \in \mathbb{R}^n$ such that $S(\mathbf{x}, d) \cap S_n \neq \emptyset$, then $\text{SLMO}(\mathbf{x}, d, \mathbf{c})$ is an LLOO $\mathcal{A}(\mathbf{x}, d, \mathbf{c})$ with $\rho = n$.*

The implication of this result is far-reaching because the framework developed in [14] can be followed to get a linearly convergent algorithm with SLMO. An even more important result is that SLMO problem (3.5) can be solved by the following simple algorithm.

Algorithm 1 $\text{SLMO}(\mathbf{x}, d, \mathbf{c})$

Input: point $\mathbf{x} \in S_n$, linear objective $\mathbf{c} \in \mathbb{R}^n$, radius $d > 0$.

- 1: $\hat{d} \leftarrow \frac{\sum_{i=1}^n \min\{d, x_i\}}{n}$
- 2: $\hat{\mathbf{x}} \leftarrow \mathbf{x} - \min\{\mathbf{x}, d\mathbf{1}_n\} + \hat{d}\mathbf{1}_n$
- 3: $\mathbf{y}_+ \leftarrow \hat{\mathbf{x}} - \hat{d}\mathbf{1}_n$
- 4: $i^* \leftarrow \arg \min_{i \in [n]} c_i$
- 5: $\mathbf{y}^* \leftarrow \mathbf{y}_+ + n\hat{d} \mathbf{e}_{i^*}$

Output: $\mathbf{y}^*$.

The algorithm follows these basic steps. Firstly, it represents the constraint set as a single simplex ball: $S(\hat{\mathbf{x}}, \hat{d})$. Secondly, it uses the existing theoretical results of linear programming over the simplex ball to find the optimal solution.

Lemma 3.3. *Alg. 1 finds an optimal solution to Problem (3.5).*

Proof. First, by Lemma 3.1(2), we have $S(\mathbf{x}, d) \cap S_n = S(\hat{\mathbf{x}}, \hat{d})$, where the definitions of $\hat{\mathbf{x}}$ and $\hat{d}$ are given in (3.3). Consequently, Problem (3.5) is equivalent to $\min_{\mathbf{y} \in S(\hat{\mathbf{x}}, \hat{d})} \langle \mathbf{y}, \mathbf{c} \rangle$. Note that this is the same form as the problem in Lemma 3.1(3). Thus, we have

$$\mathbf{y}^* = \hat{\mathbf{x}} + \hat{d}(n\mathbf{e}_{i^*} - \mathbf{1}_n) = \max\{\mathbf{x}, d\mathbf{1}_n\} - d\mathbf{1}_n + n\hat{d} \mathbf{e}_{i^*}.$$

Consequently, Alg. 1 solves Problem (3.5). $\square$

Remark 1. (Comparison with $\text{LMO}(\mathbf{c}, S_n)$ and $\ell_1\text{-LMO}(\mathbf{x}, d, \mathbf{c})$) If we treat the element-wise minimum between two vectors as a basic operation, then SLMO requires only one extra basic operation, one more vector summation, and one more vector addition compared to the the original LMO over the

simplex S_n . Therefore, its total exact complexity is $4n \text{ flops}$, making it nearly as efficient as $\text{LMO}(\mathbf{c}, S_n)$. However, $\ell_1\text{-LMO}(\mathbf{x}, d, \mathbf{c})$ involves a sorting operation, whose overall complexity is usually $O(n \log(n))$. It also involves a few more vector additions. As will be illustrated in Fig. 2c, it is far less efficient than the original $\text{LMO}(\mathbf{c}, S_n)$ and SLMO. Therefore, we expect that a linearly convergent algorithm with SLMO should be efficient as well. We develop it below.

3.2 SFW: Simplex Frank-Wolfe Method

In this subsection, we present a new variant of Frank-Wolfe method called Simplex Frank-Wolfe (abbreviated as SFW), obtained by replacing the LMO with SLMO. The algorithm is formally described as follows.

Algorithm 2 Simplex Frank-Wolfe Method: SFW

Input: $\mathbf{x}_0 \in S_n$, initial lower bound B_0 .

- 1: Set: $d_0 \leftarrow \sqrt{\frac{2(f(\mathbf{x}_0) - B_0)}{\mu}}$.
 - 2: **for** $k = 1, \dots$ **do**
 - 3: Compute $\mathbf{y}_k \in \text{SLMO}(\mathbf{x}_{k-1}, d_{k-1}, \nabla f(\mathbf{x}_{k-1}))$.
 - 4: Compute the working lower bound: $B_k^w \leftarrow f(\mathbf{x}_{k-1}) + \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{y}_k - \mathbf{x}_{k-1} \rangle$.
 - 5: Update best bound $B_k \leftarrow \max\{B_{k-1}, B_k^w\}$.
 - 6: Set $\mathbf{x}_k \leftarrow (1 - \delta_k)\mathbf{x}_{k-1} + \delta_k\mathbf{y}_k$ for some $\delta_k \in [0, 1]$.
 - 7: Set: $d_k \leftarrow \sqrt{\frac{2(f(\mathbf{x}_k) - B_k)}{\mu}}$.
 - 8: **end for**
-

Before stating its convergence rate result, we make the following remarks regarding Alg. 2.

Remark 2. (Choice of d_k update strategy) We could follow the linear shrinking rule of d_k in [14, 24]: $d_k = \gamma d_{k-1}$ with properly chosen $\gamma < 1$. The linear convergent rate would be guaranteed by invoking the LLOO property of SLMO (Lemma 3.2). We do not take this route for convergence analysis because of the following two reasons. One is that the key property $\mathbb{B}(\mathbf{x}, d) \subseteq S(\mathbf{x}, d) \subseteq \mathbb{B}(\mathbf{x}, nd)$ ensuring LLOO will have to be modified when it comes to the general polytope as our simplex ball is not based on any norm. The corresponding relationship becomes $\mathbb{B}(\mathbf{x}, dD/\eta) \subseteq S_{\mathcal{P}}(\mathbf{x}, d) \subseteq \mathbb{B}(\mathbf{x}, (n+1)dD)$, see Lemma 4.3, where $S_{\mathcal{P}}$ is defined. At least at a technical level, the original LLOO will have to be generalized to suit this extension and the corresponding proofs have also to be reproduced. The proof we provided below is more direct. The other reason is that we use the best lower bound B_k provided by the algorithm to define d_k . This choice is important because we are going to incorporate other accelerating strategies to SFW resulting in

rSFW. The stopping criterion used there will be also based on the best lower bounds obtained. Therefore, the convergence analysis for SFW will naturally be adapted to rSFW.

At the k -th iteration, Alg. 2 first invokes SLMO to find the minimum point $\mathbf{y}_k$ of the first-order approximate expansion of the objective function within the region $S(\mathbf{x}_{k-1}, d_{k-1}) \cap S_n$. Subsequently, using a suitable step size δ_k , a convex combination of $\mathbf{y}_k$ and $\mathbf{x}_{k-1}$ is computed to update the iteration point to $\mathbf{x}_{k+1}$. Finally, the algorithm updates the radius d , ensuring that the optimal solution $\mathbf{x}^*$ progressively falls within a smaller neighborhood $S(\mathbf{x}_k, d_k)$. For Alg. 2, we propose the following simple step size, as an alternative to the simple step size selection in the original Frank-Wolfe algorithm:

$$\delta_k = \frac{\mu}{2Ln^2}. \quad (3.6)$$

We have the following linear convergence rate result. The induction technique in the proof below was taken from [14, Lemma 4.3].

Theorem 3.4. *Let $\{\mathbf{x}_k\}$ be the sequences generated by Alg. 2 with step size policy for $\{\delta_k\}$ in (2.3), (2.4), or (3.6). Then, for $k \geq 0$, we have*

$$f(\mathbf{x}_k) - f^* \leq f(\mathbf{x}_k) - B_k \leq \frac{\mu d_0^2}{2} e^{-\frac{\mu}{4Ln^2}k}. \quad (3.7)$$

Proof. We first claim that $\mathbf{x}^* \in S(\mathbf{x}_k, d_k)$ and that $f(\mathbf{x}_k) - B_k \leq \frac{\mu d_k^2}{2}$. We prove this by induction. First, we have

$$\frac{\mu d_0^2}{2} = f(\mathbf{x}_0) - B_0 \geq f(\mathbf{x}_0) - f^* \stackrel{(a)}{\geq} \frac{\mu}{2} \|\mathbf{x}_0 - \mathbf{x}^*\|^2,$$

where (a) comes from (2.1). This implies that $\|\mathbf{x}_0 - \mathbf{x}^*\| \leq d_0$, and by Lemma 3.1(5), we have $\mathbf{x}^* \in S(\mathbf{x}_0, d_0)$. Therefore, the claim holds for $k = 0$.

Now suppose that $\mathbf{x}^* \in S(\mathbf{x}_t, d_t)$ and $f(\mathbf{x}_t) - B_t \leq \frac{\mu d_t^2}{2}$ for all $t \leq k-1$. Let $\gamma := \frac{\mu}{2Ln^2} \leq 1$. For step size policy δ_k in (2.3) (exact line search stepsize) or (3.6), we both have

$$\begin{aligned} f(\mathbf{x}_k) &= f(\mathbf{x}_{k-1} + \delta_k(\mathbf{y}_k - \mathbf{x}_{k-1})) \leq f(\mathbf{x}_{k-1} + \gamma(\mathbf{y}_k - \mathbf{x}_{k-1})) \\ &\leq f(\mathbf{x}_{k-1}) + \gamma \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{y}_k - \mathbf{x}_{k-1} \rangle + \frac{L\gamma^2}{2} \|\mathbf{y}_k - \mathbf{x}_{k-1}\|^2 \end{aligned} \quad (3.8)$$

Similarly, for the step size policy (2.4) (short stepsize), we have

$$\begin{aligned} f(\mathbf{x}_k) &\leq f(\mathbf{x}_{k-1}) + \delta_k \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{y}_k - \mathbf{x}_{k-1} \rangle + \frac{L\delta_k^2}{2} \|\mathbf{y}_k - \mathbf{x}_{k-1}\|^2 \\ &\leq f(\mathbf{x}_{k-1}) + \gamma \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{y}_k - \mathbf{x}_{k-1} \rangle + \frac{L\gamma^2}{2} \|\mathbf{y}_k - \mathbf{x}_{k-1}\|^2 \end{aligned} \quad (3.9)$$

Combining (3.8) and (3.9), we have

$$\begin{aligned}
f(\mathbf{x}_k) &\leq f(\mathbf{x}_{k-1}) + \gamma \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{y}_k - \mathbf{x}_{k-1} \rangle + \frac{L\gamma^2}{2} \|\mathbf{y}_k - \mathbf{x}_{k-1}\|^2 \\
&\stackrel{(b)}{\leq} (1 - \gamma)f(\mathbf{x}_{k-1}) + \gamma B_k^w + \frac{L\gamma^2}{2} \|\mathbf{y}_k - \mathbf{x}_{k-1}\|^2 \\
&\stackrel{(c)}{\leq} (1 - \gamma)(f(\mathbf{x}_{k-1}) - B_{k-1}) + B_k + \frac{L\gamma^2}{2} \|\mathbf{y}_k - \mathbf{x}_{k-1}\|^2
\end{aligned}$$

holds for step size policy (2.3), (2.4), or (3.6), where (b) comes from the definition of B_k^w , and (c) is due to $B_k \geq \max\{B_{k-1}, B_k^w\}$. Subtracting B_k from the both sides of the above inequality, we obtain

$$\begin{aligned}
f(\mathbf{x}_k) - B_k &\leq (1 - \gamma)(f(\mathbf{x}_{k-1}) - B_{k-1}) + \frac{L\gamma^2}{2} \|\mathbf{y}_k - \mathbf{x}_{k-1}\|^2 \\
&\stackrel{(d)}{\leq} (1 - \gamma) \frac{\mu}{2} d_{k-1}^2 + \frac{L\gamma^2}{2} n^2 d_{k-1}^2 = \left[(1 - \gamma) \frac{\mu}{2} + \frac{L\gamma^2 n^2}{2} \right] d_{k-1}^2,
\end{aligned}$$

where (d) is due to our inductive hypothesis and Lemma 3.1(5). By plugging in the value of γ , and using $1 - x \leq e^{-x}$, we have that

$$f(\mathbf{x}_k) - B_k \leq \frac{\mu}{2} \left(1 - \frac{\mu}{4Ln^2} \right) d_{k-1}^2 \leq \frac{\mu}{2} e^{-\frac{\mu}{4Ln^2}} d_{k-1}^2. \quad (3.10)$$

With the definition of d_k , we have $f(\mathbf{x}_k) - B_k = \frac{\mu}{2} d_k^2$. The bound in (3.10) implies

$$d_k \leq e^{-\frac{\mu}{8Ln^2}} d_{k-1}. \quad (3.11)$$

By the inductive hypothesis, we know that $\mathbf{x}^* \in S(\mathbf{x}_t, d_t)$ holds for all $t \leq k - 1$. Thus B_{t+1}^w is a valid lower bound of f^* , and consequently, B_k is also a lower bound of f^* . Now by (2.1), we have

$$\|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq \frac{2}{\mu} (f(\mathbf{x}_k) - f^*) \leq \frac{2}{\mu} (f(\mathbf{x}_k) - B_k) = d_k^2.$$

This implies that $\mathbf{x}^* \in S(\mathbf{x}_k, d_k)$ by Lemma 3.1(5). Therefore, we have completed the proof of the claim.

We now start to prove the conclusion in Theorem 3.4. From the earlier proof, we know that $B_k \leq f^*$, thus confirming the first part of the inequality. By the definition of d_k and the established reduction inequality (3.11), we have

$$f(\mathbf{x}_k) - B_k = \frac{\mu d_k^2}{2} \leq \frac{\mu d_0^2}{2} e^{-\frac{\mu}{4Ln^2} k}.$$

The proof is thus completed. $\square$

Remark 3. (Iteration complexity of SFW) If we skip Lines 4–5 and replace Line 7 in Alg. 2 with $\sqrt{\frac{2\langle \nabla f(\mathbf{x}_{k-1}), \mathbf{x}_{k-1} - \mathbf{y}_k \rangle}{\mu}}$, the algorithm remains correct and preserves its convergence guarantee. This modification avoids computing the objective value $f(\mathbf{x}_k)$, making the algorithm more practical and simple when the objective is expensive or difficult to evaluate. In this case, assuming that there is an oracle to obtain the gradient information $\nabla f(\mathbf{x})$ at each iteration, we are able to give the exact number of *flops* operations to compute the next iterate. The SLMO part to get $\mathbf{y}_k$ is $4n$ *flops*, and computing the direction d_k requires an additional $3n$ *flops*. For the short step size strategy, evaluating the step size δ_k incurs $2n$ *flops*, and updating the iterate $\mathbf{x}_k$ takes another $3n$ *flops*. Thus, SFW needs $10n$ *flops* with a simple step size, or $12n$ *flops* with the short step size—yet still achieves linear convergence for the simplex. To our knowledge, this is the lowest per-iteration cost among FW variants with linear convergence.

3.3 Refining SFW

In this subsection, we aim to further reduce the computational overhead of the proposed oracle as much as possible, while retaining the linear convergence rate. The motivation stems from an important observation. $\text{SLMO}(\mathbf{x}, d, \mathbf{c})$ can be split into two parts. The first part is to construct a new Simplex-ball $S(\hat{\mathbf{x}}, \hat{d}) = S(\mathbf{x}, d) \cap S_n$. For easy reference, we call it SLMO-1, which corresponds to Lines 1 in Alg. 1. The second part, which finds an optimal solution over $S(\hat{\mathbf{x}}, \hat{d})$, is referred to as SLMO-2 and corresponds to Lines 4–5 in Alg. 1. It is easy to see that the computation of SLMO-2 requires only one more vector addition compared to the standard LMO over S_n and hence its computation is already kept minimum. The extra computation of SLMO is from SLMO-1. Since the new Simplex ball is already constructed, we like to carry out a few more times of the SLMO-2 part. This is roughly to find an approximate solution $\mathbf{p}$ to the problem

$$\min f(\mathbf{y}) \quad \text{s.t.} \quad \mathbf{y} \in S(\hat{\mathbf{x}}, \hat{d})$$

with starting point $\mathbf{x}$. Our control of this refinement step is for the overall computation to remain $O(n)$ and the overall convergence rate to remain linear. The overall algorithm is given in Alg. 3 and it is called Refined SFW.

Alg. 3 keeps three sequences $\{(\mathbf{x}_k, d_k)\}$, $\{(\bar{\mathbf{x}}_k, \bar{d}_k)\}$, and $\{(\hat{\mathbf{x}}_k, \hat{d}_k)\}$, each associated with a simplex ball, namely $S(\mathbf{x}_k, d_k)$, $S(\bar{\mathbf{x}}_k, \bar{d}_k)$, and $S(\hat{\mathbf{x}}_k, \hat{d}_k)$. Starting with $(\mathbf{x}_0, d_0) = (\bar{\mathbf{x}}_0, \bar{d}_0)$, we define $(\hat{\mathbf{x}}_0, \hat{d}_0)$ such that $S(\hat{\mathbf{x}}_0, \hat{d}_0) =$

Algorithm 3 rSFW: Refined Simplex Frank-Wolfe Method

Input: Radius contraction ratio $\rho > 1$, initial lower bound B_0 .

```

1: Set:  $d_0 \leftarrow \frac{1}{n}$ ,  $\mathbf{x}_0 \leftarrow \frac{\mathbf{1}_n}{n}$ ,  $J \leftarrow \frac{8\rho^2 n^2 L}{\mu}$ ,  $\bar{\mathbf{x}}_0 \leftarrow \mathbf{x}_0$ ,  $\bar{d}_0 \leftarrow d_0$ .
2: for  $k = 1, \dots$  do
3:   Set:  $\mathbf{p}_0 \leftarrow \mathbf{x}_{k-1}$ ,  $C_0 \leftarrow B_{k-1}$ .
4:   (SLMO-1) construct the new Simplex ball:  $S(\hat{\mathbf{x}}_{k-1}, \hat{d}_{k-1}) = S(\bar{\mathbf{x}}_{k-1}, \bar{d}_{k-1}) \cap S_n$ .
5:   for  $j = 1, \dots, J$  do
6:     (SLMO-2): Compute  $\mathbf{y}_j = \arg \min_{\mathbf{y} \in S(\hat{\mathbf{x}}_{k-1}, \hat{d}_{k-1})} \langle \nabla f(\mathbf{p}_{j-1}), \mathbf{y} \rangle$ .
7:     Compute the current lower bound:  $C_j^w \leftarrow f(\mathbf{p}_{j-1}) + \langle \nabla f(\mathbf{p}_{j-1}), \mathbf{y}_j - \mathbf{p}_{j-1} \rangle$ .
8:     Update the best lower bound  $C_j \leftarrow \max\{C_{j-1}, C_j^w\}$ .
9:     if  $f(\mathbf{p}_j) - C_j \leq \frac{\mu}{2\rho^2} \hat{d}_{k-1}^2$  then
10:       Break the inner loop.
11:     end if
12:     Set  $\mathbf{p}_j \leftarrow (1 - \delta_j)\mathbf{p}_{j-1} + \delta_j \mathbf{y}_j$  for some  $\delta_j \in [0, 1]$ .
13:   end for
14:   Set:  $\mathbf{x}_k \leftarrow \mathbf{p}_j$ ,  $d_k \leftarrow \frac{\hat{d}_{k-1}}{\rho}$ ,  $B_k \leftarrow C_j$ .
15:   Find  $(\bar{\mathbf{x}}_k, \bar{d}_k)$  such that  $S(\bar{\mathbf{x}}_k, \bar{d}_k) = S(\mathbf{x}_k, d_k) \cap S(\hat{\mathbf{x}}_{k-1}, \hat{d}_{k-1})$  by using (3.4).
16: end for
```

$S(\bar{\mathbf{x}}_0, \bar{d}_0) \cap S_n$. At the k th iteration, we first compute

$$\mathbf{x}_k \approx \arg \min \left\{ f(\mathbf{y}) \mid \mathbf{y} \in S(\hat{\mathbf{x}}_{k-1}, \hat{d}_{k-1}) \right\} \quad \text{and} \quad d_k = \hat{d}_{k-1}/\rho.$$

We then define $(\bar{\mathbf{x}}_k, \bar{d}_k)$ by its simplex ball, which satisfies $S(\bar{\mathbf{x}}_k, \bar{d}_k) = S(\mathbf{x}_k, d_k) \cap S(\hat{\mathbf{x}}_{k-1}, \hat{d}_{k-1})$. We further define the iterate $(\hat{\mathbf{x}}_k, \hat{d}_k)$ by its simplex ball satisfying $S(\hat{\mathbf{x}}_k, \hat{d}_k) = S(\bar{\mathbf{x}}_k, \bar{d}_k) \cap S_n$. They can all be efficiently computed via the formula (3.4).

A great advantage of Alg. 3 is its computation of $\mathbf{x}_k$. The oracle we call is SLMO-2, which ensures that the iteration complexity remains the same as the original FW algorithm. It is also important to highlight that the inner loop of our algorithm (Lines 5-13) follows the standard FW algorithm. Its primary goal is to find a solution $\mathbf{p}_j$ that satisfies $f(\mathbf{p}_j) - C_j \leq \frac{\mu}{2\rho^2} \hat{d}_{k-1}^2$. Therefore, various speedup techniques for the classical FW algorithm can be directly applied to this inner loop without interfering with the outer loop of the algorithm. These include approaches such as the ‘away-step’ and ‘pairwise’ variants of FW proposed by [22], ‘fully-corrective’ variant of FW proposed by [19], as well as the warm start technique suggested by [11].

The following theorem summarizes the convergence result for this algorithm.

Theorem 3.5. *Let $\{\mathbf{x}_k\}$ be the sequences generated by Alg. 3 with step size*

policy for $\{\delta_j\}$ in (2.2)-(2.4). Then, for $k \geq 1$, we have

$$f(\mathbf{x}_k) - f^* \leq f(\mathbf{x}_k) - B_k \leq \frac{\mu}{2n^2\rho^{2k}}.$$

Proof. We first claim that $\mathbf{x}^* \in S(\widehat{\mathbf{x}}_k, \widehat{d}_k)$ for any $k \geq 0$ and we prove this by induction. For $k = 0$, we have $(\bar{\mathbf{x}}_0, \bar{d}_0) = (\mathbf{x}_0, d_0) = (\mathbf{1}_n/n, 1/n)$. By the definition of $S(\widehat{\mathbf{x}}_0, \widehat{d}_0)$, we have

$$S(\widehat{\mathbf{x}}_0, \widehat{d}_0) = S(\bar{\mathbf{x}}_0, \bar{d}_0) \cap S_n = S(\mathbf{1}_n/n, 1/n) \cap S_n = S_n.$$

Hence, $\mathbf{x}^* \in S(\widehat{\mathbf{x}}_0, \widehat{d}_0)$. Now suppose that $\mathbf{x}^* \in S(\widehat{\mathbf{x}}_{k-1}, \widehat{d}_{k-1})$ for some $k \geq 1$. Note that the inner loop of Alg. 3 corresponds to the standard Frank-Wolfe algorithm. By Theorem 2.1 and Lemma 3.1(4), which implies that the diameter of $S(\widehat{\mathbf{x}}_{k-1}, \widehat{d}_{k-1})$ is $\sqrt{2n\widehat{d}_{k-1}}$, we have

$$f(\mathbf{p}_j) - f^* \leq \frac{2L}{j+1} \left(\sqrt{2n\widehat{d}_{k-1}} \right)^2 = \frac{4Ln^2\widehat{d}_{k-1}^2}{j+1}$$

hold for all $j \in [J]$. In the case where the inner loop terminates at $j = J$, we obtain

$$f(\mathbf{x}_k) - f^* = f(\mathbf{p}_J) - f^* \leq f(\mathbf{p}_J) - C_J \leq \frac{2L}{\frac{8\rho^2 n^2 L}{\mu}} 2n^2 \widehat{d}_{k-1}^2 = \frac{\mu}{2\rho^2} \widehat{d}_{k-1}^2 = \frac{\mu}{2} d_k^2.$$

Similarly, if the inner loop is interrupted due to lines 9-11 of the algorithm, we still have

$$f(\mathbf{x}_k) - f^* \leq f(\mathbf{p}_j) - C_j \leq \frac{\mu}{2\rho^2} \widehat{d}_{k-1}^2 = \frac{\mu}{2} d_k^2.$$

Using the fact that $f(\mathbf{x}_k) - f^* \geq \frac{\mu}{2} \|\mathbf{x}_k - \mathbf{x}^*\|^2$, we have $\|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq d_k^2$, which implies via Lemma 3.1(5) that $\mathbf{x}^* \in S(\mathbf{x}_k, d_k)$. This implies

$$\begin{aligned} \mathbf{x}^* &\in S(\mathbf{x}_k, d_k) \cap S(\widehat{\mathbf{x}}_{k-1}, \widehat{d}_{k-1}) \quad (\text{as } \mathbf{x}^* \in S(\widehat{\mathbf{x}}_{k-1}, \widehat{d}_{k-1}) \text{ by induction}) \\ &= S(\mathbf{x}_k, d_k) \cap S(\widehat{\mathbf{x}}_{k-1}, \widehat{d}_{k-1}) \cap S_n \quad (\text{as } \mathbf{x}^* \in S_n) \\ &= S(\bar{\mathbf{x}}_k, \bar{d}_k) \cap S_n = S(\widehat{\mathbf{x}}_k, \widehat{d}_k). \end{aligned}$$

We now start to prove the conclusion in Theorem 3.5. Since $d_0 = \frac{1}{n}$ and $\widehat{d}_k \leq d_k = \frac{\widehat{d}_{k-1}}{\rho} \leq \frac{d_{k-1}}{\rho}$, we have $\widehat{d}_k \leq \frac{1}{n\rho^k}$ and thus

$$f(\mathbf{x}_k) - f^* \leq f(\mathbf{x}_k) - B_k \leq \frac{\mu}{2} d_k^2 \leq \frac{\mu}{2n^2\rho^{2k}}.$$

We complete the proof. $\square$

Remark 4. (Warm-start Strategy) For rSFW and rSFW $_{\mathcal{P}}$ in the upcoming Section 4, when using the simple step size, the initial steps of each inner loop may perform poorly, causing the iteration point $\mathbf{p}_j$ far away from the optimal solution. We found that the following heuristic warm-start strategy is effective in practice. Let $J_k \ll J$ denote the actual number of inner loop iterations during the k -th outer loop iteration. When initiating the $(k+1)$ -th outer loop, instead of starting the inner loop from $j = 1$, begin from either $j = \frac{J_k}{\rho'}$ or $j = \sqrt{\frac{dk}{dk-1}} J_k$. Here $\rho' > 1$ is a hyperparameter, with $\rho' = 2$ serving as a reasonable default value.

Remark 5. (Extension to Quadratic Growth Condition) Although the two main Thms. 3.4 and 3.5 are established under the strong convexity of $f(\cdot)$, we would like to point out that the assumption can be weakened to quadratic growth condition:

$$(f(\mathbf{x}) - f(\mathbf{x}^*))^{1/2} \geq c \text{dist}(\mathbf{x}, X^*), \quad \forall \mathbf{x} \in \mathcal{P},$$

where $c > 0$ and X^* is the solution set of Problem (1.1). This includes the well-known case $f(\mathbf{x}) = g(A\mathbf{x}) + \langle \mathbf{b}, \mathbf{x} \rangle$ with g strongly convex, $A \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^n$, for a detailed investigation of this class of functions with FW methods, see [1]. In this paper, we did not make effort for such extension as our main purpose is to introduce SLMO under the strong convexity setting for the sake of simplicity.

4 Generalization to Arbitrary Polytopes

This section extends the previous results for the unit simplex to arbitrary polytopes. This generalization allows for a broader application of our findings, facilitating their relevance to a wider range of optimization problems. Important properties between the standard simplex S_n and general polytopes have been established in [14]. Our extension heavily relies on some of those results. This section is patterned after Section 3 with some details omitted to avoid repeating. We first define the simplex ball for general polytope and the corresponding SLMO. We then describe the Simplex Frank-Wolfe for general polytope, followed by its refined version.

4.1 Simplex Ball and SLMO for Arbitrary Polytopes

Consider Problem (1.1) with $\mathcal{P} = \text{Conv}(\mathcal{V})$ and $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_N\}$. Therefore, any given $\mathbf{x} \in \mathcal{P}$ can be represented as a convex combination of those atoms

$\mathbf{v}_i$. However, the convex combination may not be unique. We define a set-valued mapping $\mathcal{M}$ from $\mathcal{P}$ to the following set:

$$\mathcal{M}(\mathbf{x}) := \left\{ \boldsymbol{\lambda} \in S_N \mid \mathbf{x} = \sum_{i=1}^N \lambda_i \mathbf{v}_i \right\}.$$

Recall that for $\boldsymbol{\lambda} \in \mathbb{R}^N$ and $d > 0$, $S(\boldsymbol{\lambda}, d)$ is the simplex ball defined in (3.2). The idea of defining a similar Simplex ball over the polytope can be summarized as follows:

$$\mathbf{x} \in \mathcal{P} \implies \boldsymbol{\lambda}_x \in \mathcal{M}(\mathbf{x}) \implies S(\boldsymbol{\lambda}_x, d) \implies S_{\mathcal{P}}(\mathbf{x}, d) := \{V\boldsymbol{\lambda} \mid \boldsymbol{\lambda} \in S(\boldsymbol{\lambda}_x, d)\}, \quad (4.1)$$

where V consists of the columns $\mathbf{v}_i$, $i \in [N]$. It seems that the Simplex ball $S_{\mathcal{P}}(\mathbf{x}, d)$ depends on a particular choice of $\boldsymbol{\lambda}_x \in \mathcal{M}(\mathbf{x})$. The following result dismisses this dependence.

Lemma 4.1. *Given $\mathbf{x} \in \mathcal{P}$ and $d > 0$, let $\boldsymbol{\lambda}_x, \boldsymbol{\lambda}'_x \in \mathcal{M}(\mathbf{x})$. Then for any $\boldsymbol{\lambda} \in S(\boldsymbol{\lambda}_x, d)$, there exist $\boldsymbol{\lambda}' \in S(\boldsymbol{\lambda}'_x, d)$ such that*

$$\sum_{i=1}^N \lambda(i) \mathbf{v}_i = \sum_{i=1}^N \lambda'(i) \mathbf{v}_i.$$

Proof. By definition of $\mathcal{M}(\mathbf{x})$, we know

$$\sum_{i=1}^N \lambda_x(i) \mathbf{v}_i = \sum_{i=1}^N \lambda'_x(i) \mathbf{v}_i. \quad (4.2)$$

It follows from $\boldsymbol{\lambda}'_x = \boldsymbol{\lambda}_x - \boldsymbol{\lambda}_x + \boldsymbol{\lambda}'_x$ and the definition of Simplex ball (3.1) that

$$S(\boldsymbol{\lambda}'_x, d) = S(\boldsymbol{\lambda}_x, d) - \boldsymbol{\lambda}_x + \boldsymbol{\lambda}'_x. \quad (4.3)$$

Let $\boldsymbol{\lambda}' := \boldsymbol{\lambda} - \boldsymbol{\lambda}_x + \boldsymbol{\lambda}'_x$. Since $\boldsymbol{\lambda} \in S(\boldsymbol{\lambda}_x, d)$, the identity (4.3) implies $\boldsymbol{\lambda}' \in S(\boldsymbol{\lambda}'_x, d)$. Moreover, we have

$$\sum_{i=1}^N \lambda'(i) \mathbf{v}_i = \sum_{i=1}^N (\lambda(i) - \lambda_x(i) + \lambda'_x(i)) \mathbf{v}_i = \sum_{i=1}^N \lambda(i) \mathbf{v}_i,$$

where the last equation used (4.2). This completes the proof. $\square$

Lemma 4.1 ensures that the definition is independent of choice of $\boldsymbol{\lambda}_x \in \mathcal{M}(\mathbf{x})$. Hence, the definition is well defined. Given a linear objective $\mathbf{c} \in$

$\mathbb{R}^n$, we extend it to $\mathbf{c}_{ext} \in \mathbb{R}^N$ such that $c_{ext}(i) = \langle \mathbf{v}_i, \mathbf{c} \rangle$ for all $i \in [N]$. Consequently, the following equivalence holds:

$$\min_{\mathbf{y} \in \mathcal{P}} \langle \mathbf{y}, \mathbf{c} \rangle = \min_{\boldsymbol{\lambda} \in S_N} \langle \boldsymbol{\lambda}, \mathbf{c}_{ext} \rangle.$$

Leveraging this equivalence, we define the generalized SLMO for $\mathcal{P}$ as follows.

Definition 5 (SLMO $_{\mathcal{P}}$: SLMO over $\mathcal{P}$). *Given a linear objective $\mathbf{c} \in \mathbb{R}^n$, radius $d > 0$, a point $\mathbf{x} \in \mathcal{P}$, and its corresponding $\boldsymbol{\lambda}_x \in \mathcal{M}(\mathbf{x})$, a solution $\mathbf{y}^* \in \text{SLMO}_{\mathcal{P}}(\mathbf{x}, d, \mathbf{c}, \boldsymbol{\lambda}_x)$ is referred to as a generalized simplex-based linear minimization oracle if*

$$\mathbf{y}^* = \sum_{i=1}^N \lambda_i^* \mathbf{v}_i,$$

where $\boldsymbol{\lambda}^*$ is an optimal solution to the following optimization problem

$$\begin{aligned} \min \quad & \langle \boldsymbol{\lambda}, \mathbf{c}_{ext} \rangle \\ \text{s.t.} \quad & \boldsymbol{\lambda} \in S(\boldsymbol{\lambda}_x, d) \cap S_N. \end{aligned} \tag{4.4}$$

We note that (4.4) can be efficiently solved by Alg. 1. Consequently, SLMO $_{\mathcal{P}}$ can also be efficiently solved provided an element in $\mathcal{M}(\mathbf{x})$ can be cheaply obtained. The detailed steps are outlined in Alg. 4.

Algorithm 4 SLMO $_{\mathcal{P}}(\mathbf{x}, d, \mathbf{c}, \boldsymbol{\lambda}_x)$

Input: point $\mathbf{x} \in \mathcal{P}$ with $\boldsymbol{\lambda}_x \in \mathcal{M}(\mathbf{x})$, linear objective $\mathbf{c} \in \mathbb{R}^n$, radius $d > 0$.

- 1: $\hat{d} \leftarrow \frac{\sum_{i=1}^N \min\{\lambda_x(i), d\}}{n+1}$
- 2: $\hat{\boldsymbol{\lambda}} \leftarrow \boldsymbol{\lambda}_x - \min\{\boldsymbol{\lambda}_x, d\mathbf{1}_N\} + \hat{d}\mathbf{1}_N$
- 3: $\mathbf{y}_+ \leftarrow \sum_{i=1}^N (\hat{\lambda}_i - \hat{d}) \mathbf{v}_i$
- 4: $\mathbf{v}_{i^*} \leftarrow \arg \min_{\mathbf{v} \in \mathcal{P}} \langle \mathbf{v}, \mathbf{c} \rangle$
- 5: $\mathbf{y}^* \leftarrow \mathbf{y}_+ + (n+1)\hat{d}\mathbf{v}_{i^*}$

Output: $\mathbf{y}^*$.

One can observe that SLMO $_{\mathcal{P}}$ -2—corresponding to Lines 4-5 in Alg. 4 and serving as the oracle in our subsequent Alg. 6—requires only one more vector addition and one extra scalar-vector multiplication compared to the standard LMO.

We summarize the optimality of Alg. 4 in the following result, which is direct consequence of Lemma 3.3

Lemma 4.2. *Algorithm SLMO $_{\mathcal{P}}(\mathbf{x}, d, \mathbf{c}, \boldsymbol{\lambda}_x)$ returns an optimal solution $\mathbf{y}^*$ for the problem:*

$$\mathbf{y}^* \in \arg \min \{ \langle \mathbf{c}, \mathbf{y} \rangle \mid \mathbf{y} \in S_{\mathcal{P}}(\mathbf{x}, d) \cap \mathcal{P} \}.$$

Remark 6. (Carathéodory's Representation Assumption) By Carathéodory's Representation Theorem [28, Thm. 17.1], for any point $\mathbf{x} \in \mathcal{P}$, there exists $\boldsymbol{\lambda}_x \in \mathcal{M}(\mathbf{x})$ such that $|\mathcal{I}_+(\boldsymbol{\lambda}_x)| \leq n + 1$ where $\mathcal{I}_+(\boldsymbol{\lambda}_x) := \{i \in [N] \mid \lambda_x(i) > 0\}$. As demonstrated in the illustrative examples in Supplement C.1, this representation can be easily implemented for common types of $\mathcal{P}$. In this representation, the running time of $\text{SLMO}_{\mathcal{P}}$ does not explicitly depend on the number of vertices N , but rather on the natural dimension of $\mathcal{P}$, that is, n . For the analysis in the subsequent sections, we assume without loss of generality that the selected $\boldsymbol{\lambda}_x$ always satisfies $|\mathcal{I}_+(\boldsymbol{\lambda}_x)| \leq n + 1$.

The following lemma demonstrates some useful properties of $S_{\mathcal{P}}$ and $\text{SLMO}_{\mathcal{P}}$, which can be regarded as a generalization of Lemma 3.1(5) and is crucial for proving the convergence of our algorithm in the next subsection. The detailed proof can be found in Supplement B.

Lemma 4.3. *Given $\mathbf{x} \in \mathcal{P}$, $d > 0$ and $\mathbf{y}^* \in \text{SLMO}_{\mathcal{P}}(\mathbf{x}, d, \mathbf{c}, \boldsymbol{\lambda}_x)$, for any point $\mathbf{y} \in \mathcal{P}$ satisfying $\|\mathbf{x} - \mathbf{y}\| \leq \frac{dD}{\eta}$, it follows that $\mathbf{y} \in S_{\mathcal{P}}(\mathbf{x}, d)$ and $\langle \mathbf{c}, \mathbf{y}^* \rangle \leq \langle \mathbf{c}, \mathbf{y} \rangle$. Furthermore, we have $\|\mathbf{x} - \mathbf{y}^*\| \leq (n + 1)dD$.*

We also like to note that, though similar to LLOO, $\text{SLMO}_{\mathcal{P}}(\mathbf{x}, d, \mathbf{c}, \boldsymbol{\lambda}_x)$ is not exactly an LLOO because Lemma 4.3 only proves that $\mathbb{B}(\mathbf{x}, (D/\eta)d) \subseteq S_{\mathcal{P}}(\mathbf{x}, d)$, not $\mathbb{B}(\mathbf{x}, d) \subseteq S_{\mathcal{P}}(\mathbf{x}, d)$ which would be sufficient for the first property of LLOO. We note that $D/\eta = \xi/\psi$. Therefore, the condition $\xi/\psi \geq 1$ would be enough for $\text{SLMO}_{\mathcal{P}}(\mathbf{x}, d, \mathbf{c}, \boldsymbol{\lambda}_x)$ to be LLOO.

4.2 SFW _{$\mathcal{P}$} : Simplex Frank-Wolfe for Arbitrary Polytopes

In this subsection, we extend the SFW to the polytope case. The generalized SFW algorithm is presented as follows.

Algorithm 5 SFW _{$\mathcal{P}$} : Simplex Frank-Wolfe Method for Polytope $\mathcal{P}$

Input: $\mathbf{x}_0 \in S_n$, initial lower bound B_0 , condition number η and diameter D of $\mathcal{P}$.

- 1: Set: $d_0 \leftarrow \sqrt{\frac{2(f(\mathbf{x}_0) - B_0)}{\mu}}$, $\boldsymbol{\lambda}_0 \in \mathcal{M}(\mathbf{x}_0)$.
 - 2: **for** $k = 1, \dots$ **do**
 - 3: Compute $\mathbf{y}_k \in \text{SLMO}_{\mathcal{P}}(\mathbf{x}_{k-1}, \frac{\eta}{D}d_{k-1}, \nabla f(\mathbf{x}_{k-1}), \boldsymbol{\lambda}_{k-1})$.
 - 4: Compute the working lower bound: $B_k^w \leftarrow f(\mathbf{x}_{k-1}) + \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{y}_k - \mathbf{x}_{k-1} \rangle$.
 - 5: Update best bound $B_k \leftarrow \max\{B_{k-1}, B_k^w\}$.
 - 6: Set $\mathbf{x}_k \leftarrow (1 - \delta_k)\mathbf{x}_{k-1} + \delta_k\mathbf{y}_k$ for some $\delta_k \in [0, 1]$.
 - 7: Set: $d_k \leftarrow \sqrt{\frac{2(f(\mathbf{x}_k) - B_k)}{\mu}}$, $\boldsymbol{\lambda}_k \in \mathcal{M}(\mathbf{x}_k)$.
 - 8: **end for**
-

Notice that when $\mathcal{P}$ degenerates to S_n , the algorithm differs from Alg. 5 only slightly at line 7 since $\eta = D = \sqrt{2}$ for S_n . The convergence for Alg. 5

stated below is proved in Supplement B.

Theorem 4.4. *Let $\{\mathbf{x}_k\}$ be the sequences generated by Alg. 5 with step size policy for $\{\delta_k\}$ in (2.3), (2.4), or simple step size*

$$\delta_k = \frac{\mu}{2L(n+1)^2\eta^2}. \quad (4.5)$$

Then, for $k \geq 0$, we have

$$f(\mathbf{x}_k) - f^* \leq f(\mathbf{x}_k) - B_k \leq \frac{\mu d_0^2}{2} e^{-\frac{\mu}{4L\eta^2(n+1)^2}k}. \quad (4.6)$$

4.3 rSFW _{$\mathcal{P}$} : Refining SFW _{$\mathcal{P}$}

As with the motivation for rSFW for the Simplex case, once we constructed the Simplex ball for $\mathcal{P}$, we may compute an approximate solution:

$$\mathbf{p}_k \approx \arg \min f(\mathbf{p}) \quad \text{s.t.} \quad \mathbf{p} \in S(\mathbf{x}_{k-1}, d_{k-1}) \cap S_N,$$

with the initial point $\mathbf{p}_0 = \mathbf{x}_{k-1}$. The motivation is based on a similar observation that SFW _{$\mathcal{P}$} can be split into two independent parts with the first part of constructing the Simplex ball being the major computation. Hence, once such a ball is constructed we run a few more cheap SLMO steps over this ball. Once again, other methods such as Away-step FW and pairwise FW can be used for computing $\mathbf{p}_k$. The generalized rSFW algorithm is presented as follows.

Similar to Theorem 3.5, we can provide the following convergence analysis for Alg. 6, which is proven in Supplement B.

Theorem 4.5. *Let $\{\mathbf{x}_k\}$ be the sequences generated by Alg. 6 with step size policy for $\{\delta_j\}$ in (2.2)-(2.4). Then, for $k \geq 1$, we have*

$$f(\mathbf{x}_k) - f^* \leq f(\mathbf{x}_k) - B_k \leq (f(\mathbf{x}_0) - B_0)\rho^{-2k}.$$

Remark 7. (Adaptive Lower Bound Update) We estimate the lower bound of f^* by $f(\mathbf{x}_{k-1}) + \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{y}_k - \mathbf{x}_{k-1} \rangle$ for the Simplex Frank-Wolfe method and its refined version. In fact, when the objective function exhibits specific structural properties, we can derive an additional lower B_k^o and update the best bound B_k as $B_k \leftarrow \max\{B_{k-1}, B_k^w, B_k^o\}$. For instance, when the objective function has a minmax structure, we can construct a minmax lower bound B_k^o for f^* , see [11] for detailed analysis. Moreover, in certain application scenarios, there may be exact information about the optimal value f^* , such as in linear regression or machine learning tasks, where it is known a priori that the optimal value of the loss function is 0. In such cases, it is straightforward to set $B_k^o \leftarrow f^*$.

Algorithm 6 rSFW _{$\mathcal{P}$} : Refined Simplex Frank-Wolfe Method for Polytope $\mathcal{P}$

Input: $\mathbf{x}_0 \in \mathcal{P}$, $\boldsymbol{\lambda}_0 \in \mathcal{M}(\mathbf{x}_0)$, radius contraction ratio $\rho > 1$, initial lower bound B_0 , condition number η and diameter D of $\mathcal{P}$.

- 1: Set: $d_0 \leftarrow \frac{\eta}{D} \sqrt{\frac{2(f(\mathbf{x}_0) - B_0)}{\mu}}$, $J \leftarrow \frac{4\rho^2(n+1)^2\eta^2L}{\mu}$.
 - 2: **for** $k = 1, \dots$ **do**
 - 3: Set: $\mathbf{p}_0 \leftarrow \mathbf{x}_{k-1}$, $C_0 \leftarrow B_{k-1}$.
 - 4: (SLMO _{$\mathcal{P}$} -1) Compute $\hat{\boldsymbol{\lambda}}_{k-1}$ and $\hat{d}_{k-1}$ such that $S(\hat{\boldsymbol{\lambda}}_{k-1}, \hat{d}_{k-1}) = S(\boldsymbol{\lambda}_{k-1}, d_{k-1}) \cap S_N$.
 - 5: **for** $j = 1, \dots, J$ **do**
 - 6: (SLMO _{$\mathcal{P}$} -2) Compute $\mathbf{y}_j \in \text{SLMO}_{\mathcal{P}}(\mathbf{x}_{k-1}, d_{k-1}, \nabla f(\mathbf{p}_{j-1}), \boldsymbol{\lambda}_{k-1})$.
 - 7: Set: $C_j^w \leftarrow f(\mathbf{p}_{j-1}) + \langle \nabla f(\mathbf{p}_{j-1}), \mathbf{y}_j - \mathbf{p}_{j-1} \rangle$.
 - 8: Update best bound $C_j \leftarrow \max\{C_{j-1}, C_j^w\}$.
 - 9: **if** $f(\mathbf{p}_j) - C_j \leq \frac{\mu}{2\rho^2\eta^2} d_{k-1}^2 D^2$ **then**
 - 10: Break out of the inner loop.
 - 11: **end if**
 - 12: Set $\mathbf{p}_j \leftarrow (1 - \delta_j)\mathbf{p}_{j-1} + \delta_j\mathbf{y}_j$ for some $\delta_j \in [0, 1]$.
 - 13: **end for**
 - 14: Set: $\mathbf{x}_k \leftarrow \mathbf{p}_j$, $d_k \leftarrow \frac{d_{k-1}}{\rho}$, $B_k \leftarrow C_j$ and $\boldsymbol{\lambda}_k \in \mathcal{M}(\mathbf{x}_k)$.
 - 15: **end for**
-

Remark 8. (Robustness to Parameter Estimation) Our algorithms rely on parameters L, μ, η, D . In practice, using overestimates L', η', D' and an underestimate μ' such that $\frac{L'\eta'D'\mu}{L\eta D\mu'} = O(1)$ only increases the bounds by a constant factor. Moreover, as shown in Supplement C.2, both η and D can be efficiently estimated for common $\mathcal{P}$. For L and μ , one can use the backtracking strategy from [26] to estimate their local values and compute adaptive short step sizes; see Subsection 5.4 for details.

5 Numerical Experiments

In this section, we present numerical experiments to evaluate the efficiency, convergence, and adaptability of the proposed methods. All tests were performed using MATLAB R2022b on a Windows laptop equipped with a 14-core Intel(R) Core(TM) 2.30GHz CPU and 16GB of RAM.

We try to furnish four tasks. (T1) We first assess the computational efficiency of SLMO and SLMO-2 across four representative polytopes, consolidating their role of the workhorse in our SFW methods. (T2) We illustrate the linear convergence behavior of SFW and rSFW using two numerical experiments. (T3) We show that our methods can be enhanced with a backtracking strategy to eliminate the need for predefined values of the parameters L and μ . (T4) We demonstrate how integrating the away-step variants of

the Frank-Wolfe method (AFW and PFW) into the rSFW framework significantly enhances its performance, outperforming the original AFW and PFW methods. Those four tasks are addressed in four subsections.

5.1 Efficiency of SLMO and SLMO-2

In this subsection, we evaluate the performance of the proposed SLMO and SLMO-2 through comparative experiments on four common polytopes $\mathcal{P}$: (a) Unit simplex; (b) Hypercube; (c) ℓ_1 -ball; and (d) Flow polytope, derived from the video co-localization problem in [21].

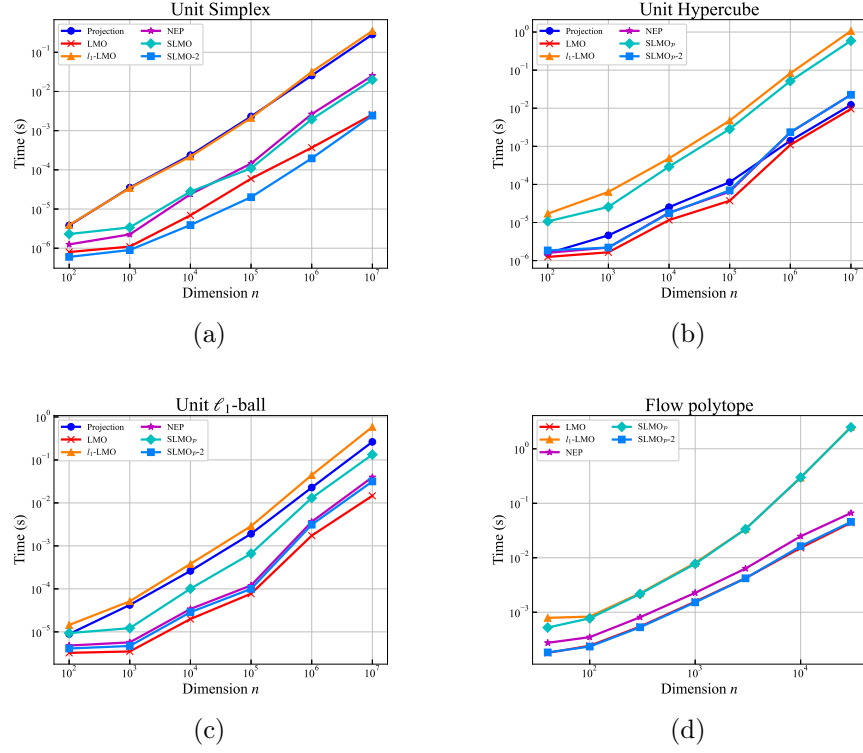


Figure 2: Comparison of solving Projection, LMO, ℓ_1 -LMO, NEP, SLMO, and SLMO-2 over the following polytopes: (a) Unit Simplex; (b) Unit Hypercube; (c) Unit ℓ_1 -ball; and (d) Flow polytope. All the results are averaged over 20 i.i.d runs. We omit the projection onto the flow polytope due to its prohibitively high computational cost.

We consider the six different methods, including projection (a key sub-

Table 1: Description of projection and five LMO variants used in the numerical comparison. These six methods shares the same randomly generated parameters: $\mathbf{c} \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{x} \in \mathcal{P}$ and $d \in \mathcal{U}_{[0,1]}$.

Algorithm	Formulation	Description
Projection	$\arg \min_{\mathbf{y} \in \mathcal{P}} \ \mathbf{y} - \mathbf{z}\ ^2$	The projection onto the polytope $\mathcal{P}$, and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, I_n)$ is a randomly generated point. We implement projections onto the Simplex and ℓ_1 -ball using the method from [8, Fig. 2], while the projection onto the hypercube is straightforward. Although a closed-form solution exists for projection onto the flow polytope [29, Thm. 20], its computational complexity of $O(m^3n + n^2)$ makes it significantly more expensive than other LMO variants.
LMO	$\arg \min_{\mathbf{y} \in \mathcal{P}} \langle \mathbf{y}, \mathbf{c} \rangle$	The standard linear minimization oracle.
ℓ_1 -LMO	$\arg \min_{\mathbf{y} \in \{V\boldsymbol{\lambda} \boldsymbol{\lambda} \in B_1(\boldsymbol{\lambda}_x, d) \cap S_N\}} \langle \mathbf{y}, \mathbf{c} \rangle$	The ℓ_1 -norm constrained LMO, Alg. 3 and Alg. 4 in [14]. Here, V consists of columns of $\mathbf{v} \in \mathcal{V}(\mathcal{P})$, and the computation of $\boldsymbol{\lambda}_x \in \mathcal{M}(\mathbf{x})$ is included in the timing.
NEP	$\arg \min_{\mathbf{y} \in \mathcal{V}(\mathcal{P})} \langle \mathbf{y}, \mathbf{c} \rangle + \lambda \ \mathbf{y} - \mathbf{x}\ ^2$	Nearest extreme point oracle in [15]. Here, $\lambda \sim \mathcal{U}_{[0,10000]}$ is a randomly generated positive number.
SLMO $_{\mathcal{P}}$	$\arg \min_{\mathbf{y} \in \{V\boldsymbol{\lambda} \boldsymbol{\lambda} \in S(\boldsymbol{\lambda}_x, d) \cap S_N\}} \langle \mathbf{y}, \mathbf{c} \rangle$	Our proposed Simplex Linear Minimization Oracle (Alg. 1 and Alg. 4). Here, the computation of $\boldsymbol{\lambda}_x \in \mathcal{M}(\mathbf{x})$ is included in the timing.
SLMO $_{\mathcal{P}-2}$	$\arg \min_{\mathbf{y} \in \{V\boldsymbol{\lambda} \boldsymbol{\lambda} \in S(\widehat{\boldsymbol{\lambda}}_x, \widehat{d})\}} \langle \mathbf{y}, \mathbf{c} \rangle$	The latter phase of SLMO, consisting of Lines 4-5 of Alg. 1 and Alg. 4. Here, $S(\widehat{\boldsymbol{\lambda}}_x, \widehat{d}) = S(\boldsymbol{\lambda}_x, d) \cap S_N$ is precomputed and not included in the timing.

problem in projection/proximal based methods) and five variants of LMO, as detailed in Table 1. These six methods shares the same randomly generated $\mathbf{c} \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{x} \in \mathcal{P}$ and $d \in \mathcal{U}_{[0,1]}$. Figure 2 illustrates the relationship between running time and dimensionality for the six methods. Additionally, Table 2 reports the proportion of time spent on LMO calls within the SLMO-2 algorithm. We draw the following observations from these results:

- **ℓ_1 -LMO:** Across all four polytopes, the ℓ_1 -LMO method incurs the highest computational overhead surpassing even that of projection-based methods.
- **SLMO $_{\mathcal{P}-2}$ vs. SLMO $_{\mathcal{P}}$:** The overhead of SLMO-2 is significantly lower than that of SLMO. In most cases, as shown in Table 2, its overhead closely matches that of the LMO itself. This indicates that our proposed rSFW and rSFW $_{\mathcal{P}}$ achieve iterative complexity comparable to that of the standard Frank-Wolfe algorithm.
- **NEP:** While NEP demonstrates very low runtime overhead, it is important to note that the corresponding Frank-Wolfe variant, NEP-FW, converges only sublinearly as shown in Subsection 5.2.
- **ℓ_1 -LMO and SLMO $_{\mathcal{P}}$ on the Flow Polytope:** Both methods exhibit rising overhead with increasing dimension, mainly due to the cost of computing the Carathéodory representation $\lambda_{\mathbf{x}} \in \mathcal{M}(\mathbf{x})$, which dominates the runtime.

Table 2: Time overhead of LMO calls as a percentage of total computation time when using the SLMO-2 algorithm across four different polytopes.

	Simplex ($n = 10^7$)	Hypercube ($n = 10^7$)	ℓ_1 -ball ($n = 10^7$)	Flow polytope ($n = 3 \times 10^4$)
$\frac{\text{Time(LMO)}}{\text{Time(SLMO-2)}}$	98.8%	46.0%	52.6%	99.7%

5.2 Linear Convergence of SFW and rSFW

We demonstrate the linear convergence of our proposed methods—SFW and rSFW through two numerical experiments. These methods are compared

against the standard Frank-Wolfe (FW) algorithm, its variant NEP-FW¹, and two well-known variants: Away-step FW² (AFW) and Pairwise FW (PFW), all summarized in Table 3.

To evaluate algorithmic performance, we adopt the Frank-Wolfe gap defined by $\langle \nabla f(\mathbf{x}_k), \mathbf{x}_k - \mathbf{y}_{k+1} \rangle$, where $\mathbf{x}_k$ is the k -th iterate and $\mathbf{y}_k$ denotes the solution returned by the respective LMO variant at that iteration. This FW gap provides a valid upper bound on the primal gap, i.e., $f(\mathbf{x}_k) - f^* \leq \langle \nabla f(\mathbf{x}_k), \mathbf{x}_k - \mathbf{y}_{k+1} \rangle$, and can thus be used as a practical stopping criterion³.

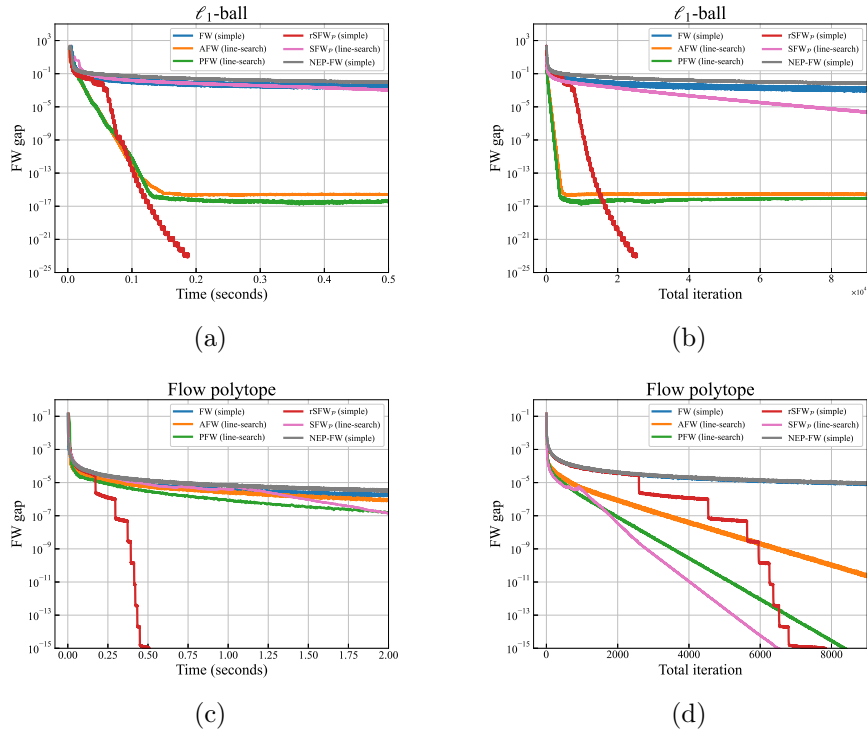


Figure 3: FW gap vs time/iterations.

The first experiment involves an ℓ_1 -regularized least squares regression,

¹As the code for NEP-FW is not publicly available, we implemented it ourselves.

²The implementations of AFW and PFW are available at <https://github.com/Simon-Lacoste-Julien/linearFW>.

³For NEP-FW, however, this inequality does not hold in general for $\mathbf{y}_{k+1} = \text{NEP}(\mathbf{x}_k) = \text{LMO}(\nabla f(\mathbf{x}_k) - \lambda_k \mathbf{x}_k, \mathcal{P})$. Therefore, to ensure a consistent and fair comparison, we use the standard FW gap to evaluate NEP-FW as well.

that is $\min_{\|\mathbf{x}\|_1 \leq 1} \|A\mathbf{x} - \mathbf{b}\|_2^2$, where $A \in \mathbb{R}^{m \times n}$ with $m = 400, n = 100$, and the entries of A are drawn from a standard Gaussian distribution. We set $\mathbf{b} = A\mathbf{x}^*$, where $\mathbf{x}^*$ is constructed by first generating a random vector with sparsity parameter $s = 0.7$, followed by normalization to lie on the boundary of the ℓ_1 -ball. Thus, the optimal value of this problem is 0. We use the same initial point $\mathbf{x}_0 = \mathbf{0}_n$ for all methods.

The second experiment involves a convex quadratic problem over the flow polytope, derived from the *video co-localization* task introduced by [21]. The problem is formulated as $\min_{\mathbf{x} \in \mathcal{F}_{s,t}} \frac{1}{2} \mathbf{x}' A \mathbf{x} + \mathbf{b}' \mathbf{x}$, where $A \in \mathbb{R}^{n \times n}$ is a positive definite matrix, $\mathbf{b} \in \mathbb{R}^n$, and $\mathcal{F}_{s,t}$ represents the s-t flow polytope. We used the same dataset and initial point as in [22, 15]. The problem has a dimension of $n = 660$.

The results are presented in Figure 3. We make some comments below.

- **Linear convergence of SFW $\mathcal{P}$ and rSFW $\mathcal{P}$:** Both SFW $\mathcal{P}$ and rSFW $\mathcal{P}$ show linear convergence, confirming our theoretical guarantees.
- **Superior efficiency of rSFW $\mathcal{P}$:** In both experiments, our proposed rSFW method significantly outperforms all other algorithms—including the well-established AFW and PFW—in terms of running time.
- **Limitations of NEP-FW:** Although NEP performs well in iteration complexity, its NEP-FW variant converges sublinearly and is slightly slower than standard FW.
- **Time inefficiency in video co-localization:** In the video co-localization task, while SFW $\mathcal{P}$, AFW, and PFW converge quickly by iteration count, their runtime is slower due to overhead—Carathéodory computation for SFW $\mathcal{P}$, and growing active sets for AFW and PFW.

5.3 SFW/rSFW with Backtracking

We further show that our methods can be enhanced with the backtracking technique proposed by [26], thereby eliminating the need to manually specify the parameters L and μ . While [26, Alg. 2] does not specify how to estimate the strong convexity constant μ , we outline our approach to estimating both L and μ as well as determining an adaptive step size; see D for the detailed algorithm. As an example, consider the k -th iteration of SFW $\mathcal{P}$. Before updating $\mathbf{x}_k$, we perform the following backtracking step:

$$\delta_k, L_k, \mu_k \leftarrow \text{Backtracking-Routine}(\mathbf{x}_{k-1}, \mathbf{y}_k - \mathbf{x}_{k-1}, L_{k-1}, \mu_{k-1}, 1). \quad (5.1)$$

We focus on the ℓ_1 -constrained logistic regression problem with the form:

$$\min_{\|\mathbf{x}\|_1 \leq \beta} \frac{1}{m} \sum_{i=1}^m \ln(1 + \exp(-b_i \langle \mathbf{a}_i, \mathbf{x} \rangle)) + \frac{\lambda}{2} \|\mathbf{x}\|^2,$$

where $A = [\mathbf{a}_1, \dots, \mathbf{a}_m] \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. We use the dataset Madelon [17], which has $m = 4400, n = 500$, and fully-density (i.e. density = 1). We set $\beta = 1$ and $\lambda = 1/n$. We compare our methods against AFW, PFW, and the standard FW, all of which are equipped with the backtracking technique. The results are presented in Figure 4. It can be observed that our two methods achieve the best performance in terms of running time. Although AFW and PFW perform well in terms of iteration count, their overall efficiency is hindered by the increasing cost of maintaining a growing active set and computing the away direction.

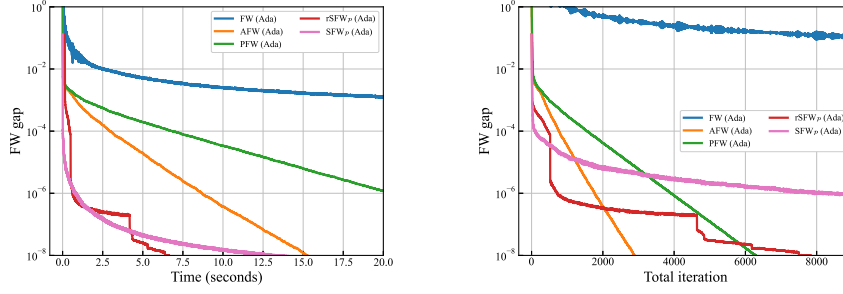


Figure 4: FW gap vs time/iterations on the ℓ_1 -constrained logistic regression problem.

5.4 rSFW Framework Combined with the Away Step Technique

In this subsection, we demonstrate that the well-known linearly converging variants of the standard Frank-Wolfe method AFW and PFW [22] can be seamlessly integrated into the inner loop of the rSFW framework. This straightforward combination leads to a significant performance improvement over the original AFW and PFW methods.

We focus on the simplex-regularized problem $\min_{\mathbf{x} \in S_n} \|A\mathbf{x} - \mathbf{b}\|_2^2$, where $A \in \mathbb{R}^{m \times n}, m = 800, n = 200$, with standard Gaussian entries. We set $\mathbf{b} = A\mathbf{x}^*$, where $\mathbf{x}^*$ is constructed by first generating a random nonnegative vector with sparsity parameter $d = 0.6$ and then normalized it so that its

components sum to 1. Thus 0 is the optimal value of this problem. We use the same initial point $\mathbf{x}_0 = \mathbf{1}_n/n$ for all methods.

In comparison to the experiments in the previous subsection, we introduce four additional algorithms: AFW and PFW, along with their respective versions integrated into the rSFW framework, denoted as rSFW-A and rSFW-P. Details of these methods are provided in Table 3.

We briefly explain here how the direction-correction $\mathbf{g}$ is computed within the general framework (1.2) for both the rSFW-A and rSFW-P algorithms. Based on Alg. 3, during the k -th outer loop and the j -th inner loop, let $S^{(k,j)} \subset \mathcal{V}(S(\hat{\mathbf{x}}_{k-1}, \hat{d}_{k-1}))$ denote the active set corresponding to the point $\mathbf{p}_{j-1}$. Thus, $\mathbf{p}_{j-1}$ can be represented as $\mathbf{p}_{j-1} = \sum_{\mathbf{v} \in S^{(k,j)}} \alpha_{\mathbf{v}} \mathbf{v}$ where $\alpha_{\mathbf{v}} > 0$. Let $\mathbf{v}_j = \arg \max_{\mathbf{v} \in S^{(k,j)}} \langle \nabla f(\mathbf{p}_{j-1}), \mathbf{v} \rangle$. For the rSFW-A method, the direction-correction is computed as:

$$\mathbf{g}_j = \begin{cases} \frac{1}{1-\alpha_{\mathbf{v}_j}} \mathbf{p}_{j-1} - \mathbf{y} - \frac{\alpha_{\mathbf{v}_j}}{1-\alpha_{\mathbf{v}_j}} \mathbf{v}_j & \text{if } \Delta_j < 0, \\ \mathbf{0} & \text{if } \Delta_j \geq 0, \end{cases} \quad (5.2)$$

where $\Delta_j := \langle -\nabla f(\mathbf{g}_{j-1}), \mathbf{y}_j - \mathbf{p}_{j-1} \rangle - \langle -\nabla f(\mathbf{g}_{j-1}), \mathbf{p}_{j-1} - \mathbf{v}_j \rangle$.

For the rSFW-P method, the direction-correction is computed as:

$$\mathbf{g}_j = \mathbf{p}_{j-1} - (1 - \alpha_{\mathbf{v}_j}) \mathbf{y}_j - \alpha_{\mathbf{v}_j} \mathbf{v}_j. \quad (5.3)$$

Finally, we update the point $\mathbf{p}_j$ using the iteration:

$$\mathbf{p}_j \leftarrow (1 - \delta_j) \mathbf{p}_{j-1} + \delta_j (\mathbf{y}_j + \mathbf{g}_j), \quad (5.4)$$

which replaces the original iteration.

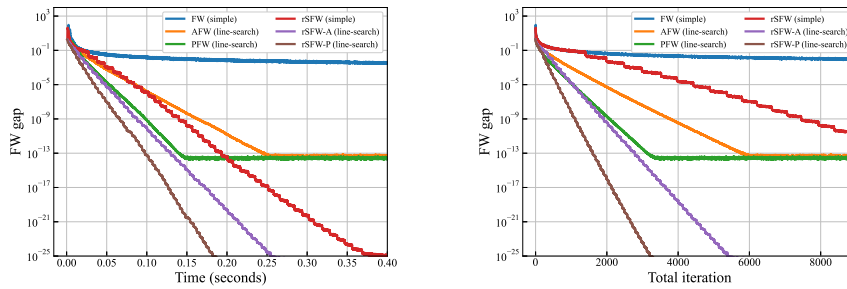


Figure 5: FW gap vs time/iterations on the Simplex-constrained least squared problem with $(m, n) = (800, 200)$.

The results are given in Figure 5. rSFW-P demonstrates superior performance compared to all other algorithms, excelling in both the number of iterations and running time, achieving nearly twice the efficiency of PFW. Additionally, both framework-based acceleration algorithms exhibit substantial performance improvements over the standalone rSFW framework.

6 Conclusion

In this paper, we introduced a novel oracle: SLMO, which leverages the advantageous geometric properties of the unit simplex. This design enables SLMO to be implemented with the same computational complexity as the standard linear optimization oracle, preserving the efficiency of the Frank-Wolfe framework. Building on this oracle, we proposed two new variants of the classical Frank-Wolfe algorithm: the Simplex Frank-Wolfe (SFW) and refined Simplex Frank-Wolfe (rSFW) algorithms. Both methods achieve linear convergence for smooth and strongly convex optimization problems over polytopes. The linear convergence rates of these methods depend only on the condition number of the objective function, the polytope's quantity, and the problem's dimension, demonstrating their scalability and robustness in various settings.

The purpose of this paper is to develop the basic framework for the new SFW methods and demonstrate that they are highly competitive. We do so with the simplest setting of f being strongly convex and smooth. We made no attempt to weaken such assumption except pointing out that the obtained results should also hold under the quadratic growth condition. An immediate question would be to extend the methods to convex or even nonconvex setting. Furthermore, LLOO proposed in [14] was an elegant framework and it was largely omitted from recent surveys on FW methods. To our best knowledge, SFW was the first LLOO instance that was extensively tested and compared with other popular FW methods. An intriguing question is whether there exist alternative LLOO approaches that simultaneously satisfy the following criteria: (i) adhering to the LLOO framework, (ii) enabling fast and accurate computation, and (iii) incurring significantly lower overhead compared to projection-based methods? We leave those topics to our future research.

acknowledgement

This work was supported by the National Natural Science Foundation of China (Grant No. 12171271) and by Hong Kong RGC General Research

Appendix

A Proof of Lemma 3.1

Proof. (1) By the definition of the simplex ball $S(\mathbf{x}, d)$, we have

$$S(\mathbf{1}_n/n, 1/n) = \frac{1}{n}\mathbf{1}_n + n \times \frac{1}{n}S_0 = \frac{1}{n}\mathbf{1}_n + S_0 = S_n.$$

Furthermore, we have

$$\text{Conv}\{n\mathbf{e}_i - \mathbf{1}_n : i \in [n]\} = n\text{Conv}\{\mathbf{e}_i : i \in [n]\} - \mathbf{1}_n = nS_n - \mathbf{1}_n = nS_0.$$

Consequently, the characterization (3.2) holds.

(2) On one hand, for every $\mathbf{y} \in S(\mathbf{x}, d) \cap S_n$, there exists $\boldsymbol{\lambda} \in S_n$ such that $y_i = x_i - d + nd\lambda_i, \forall i \in [n]$. Define for each $i \in [n]$,

$$\hat{\lambda}_i = \begin{cases} \frac{nd\lambda_i}{\sum_{j=1}^n \min\{d, x_j\}} & \text{if } x_i \geq d, \\ \frac{x_i - d + nd\lambda_i}{\sum_{j=1}^n \min\{d, x_j\}} & \text{if } x_i < d. \end{cases}$$

Since $\mathbf{y} \in S_n$ and $\boldsymbol{\lambda} \in S_n$, we have $nd\lambda_i \geq 0$ and $x_i - d + nd\lambda_i = y_i \geq 0$, which shows that $\hat{\lambda}_i \geq 0$ for each $i \in [n]$. Moreover, let $\mathcal{I}_- := \{i \in [n] \mid x_i < d\}$ be an index set. Then we have

$$\sum_{i=1}^n \hat{\lambda}_i = \frac{\sum_{i \in \mathcal{I}_-} (x_i - d) + nd}{\sum_{i=1}^n \min\{d, x_i\}} = \frac{\sum_{i \in \mathcal{I}_-} x_i + (n - |\mathcal{I}_-|)d}{\sum_{i \in \mathcal{I}_-} x_i + (n - |\mathcal{I}_-|)d} = 1.$$

As above, we have verified that $\hat{\boldsymbol{\lambda}} \in S_n$. We now show that $\mathbf{y} = (\hat{\mathbf{x}} - \hat{d}\mathbf{1}_n) + nd\hat{\boldsymbol{\lambda}}$, where $\hat{\mathbf{x}}$ and $\hat{d}$ are defined in (3.3). This follows from the fact that, for each $i \in [n]$,

$$\begin{aligned} & \hat{x}_i - \hat{d} + nd\hat{\lambda}_i \\ &= \max\{x_i, d\} + \hat{d} - d - \hat{d} + n \frac{\sum_{j=1}^n \min\{d, x_j\}}{n} \times \frac{\min\{x_i - d, 0\} + nd\lambda_i}{\sum_{j=1}^n \min\{d, x_j\}} \\ &= \max\{x_i, d\} - d + \min\{x_i - d, 0\} + nd\lambda_i = x_i - d + nd\lambda_i = y_i. \end{aligned}$$

Thus, we have $\mathbf{y} \in S(\hat{\mathbf{x}}, \hat{d})$, leading to $S_n \cap S(\mathbf{x}, d) \subset S(\hat{\mathbf{x}}, \hat{d})$.

On the other hand, for every $\mathbf{y} \in S(\widehat{\mathbf{x}}, \widehat{d})$, there exists $\widehat{\boldsymbol{\lambda}} \in S_n$ such that $y_i = \widehat{x}_i - \widehat{d} + n\widehat{d}\widehat{\lambda}_i$. Due to the fact that for $i \in [n]$, $y_i \geq \widehat{x}_i - \widehat{d} = \max\{x_i, d\} - d \geq 0$ and

$$\sum_{i=1}^n y_i = \sum_{i=1}^n \max\{x_i, d\} + n\widehat{d} - nd = \sum_{i=1}^n \max\{x_i, d\} + \sum_{i=1}^n \min\{x_i, d\} - nd = \sum_{i=1}^n x_i = 1,$$

we have $\mathbf{y} \in S_n$. We now turn to show that $\mathbf{y} \in S(\mathbf{x}, d)$.

Let $\lambda_i := \frac{\max\{x_i, d\} - x_i + \sum_{j=1}^n \min\{x_j, d\} \widehat{\lambda}_i}{n\widehat{d}}$. It is not difficult to verify that $\lambda_i \geq 0$ and $\sum_{i=1}^n \lambda_i = 1$. Moreover, we have

$$\begin{aligned} x_i - d + nd\lambda_i &= x_i - d + \max\{x_i, d\} - x_i + \sum_{j=1}^n \min\{x_j, d\} \widehat{\lambda}_i \\ &= (\max\{x_i, d\} + \widehat{d} - d) - \widehat{d} + n\widehat{d}\widehat{\lambda}_i = \widehat{x}_i - \widehat{d} + n\widehat{d}\widehat{\lambda}_i = y_i, \end{aligned}$$

implying $\mathbf{y} \in S(\mathbf{x}, d)$. Thus $S(\widehat{\mathbf{x}}, \widehat{d}) \subset S_n \cap S(\mathbf{x}, d)$. This finishes the proof for (3.3).

We now proceed to prove (3.4). Following the definition of $S(\mathbf{x}, d)$, we have

$$S(\mathbf{x}, d) = (nd)S_n + (\mathbf{x} - d\mathbf{1}_n).$$

Translating to $(\mathbf{x}_1, d_1)$ and $(\mathbf{x}_2, d_2)$, we have

$$\begin{aligned} S(\mathbf{x}_1, d_1) &= (nd_1)S_n + (\mathbf{x}_1 - d_1\mathbf{1}_n), \\ S(\mathbf{x}_2, d_2) &= (nd_2)S_n + (\mathbf{x}_2 - d_2\mathbf{1}_n) \\ &= (nd_1) \left[n \times \frac{d_2}{nd_1} S_n + \left(\frac{\mathbf{x}_2 - (\mathbf{x}_1 - d_1\mathbf{1}_n)}{nd_1} - \frac{d_2}{nd_1} \mathbf{1}_n \right) \right] + (\mathbf{x}_1 - d_1\mathbf{1}_n) \\ &= nd_1 S \left(\frac{\mathbf{x}_2 - (\mathbf{x}_1 - d_1\mathbf{1}_n)}{nd_1}, \frac{d_2}{nd_1} \right) + (\mathbf{x}_1 - d_1\mathbf{1}_n). \end{aligned}$$

Therefore,

$$S(\mathbf{x}_1, d_1) \cap S(\mathbf{x}_2, d_2) = (\mathbf{x}_1 - d_1\mathbf{1}_n) + (nd_1) \left[S_n \cap \left(\frac{\mathbf{x}_2 - (\mathbf{x}_1 - d_1\mathbf{1}_n)}{nd_1}, \frac{d_2}{nd_1} \right) \right]$$

Using (3.3), we have

$$S_n \cap S \left(\frac{\mathbf{x}_2 - (\mathbf{x}_1 - d_1\mathbf{1}_n)}{nd_1}, \frac{d_2}{nd_1} \right) = S(\widehat{\mathbf{x}}, \widehat{d}),$$

where

$$\begin{aligned}\widehat{d} &= \frac{1}{n} \sum_{i=1}^n \min \left\{ \frac{d_2}{nd_1}, \frac{x_2(i) - x_1(i) + d_1}{nd_1} \right\} = \frac{\sum_{i=1}^n \min\{d_2, x_2(i) - x_1(i) + d_1\}}{n^2 d_1} \\ &\stackrel{(a)}{=} \frac{1 + \sum_{i=1}^n \min\{d_1 - x_1(i), d_2 - x_2(i)\}}{n^2 d_1} \\ \widehat{x}_i &= \max \left\{ \frac{d_2}{nd_1}, \frac{x_2(i) - x_1(i) + d_1}{nd_1} \right\} + \left(\widehat{d} - \frac{d_2}{nd_1} \right), \quad \forall i \in [n].\end{aligned}$$

Here, (a) follows from the fact $\mathbf{x}_2 \in S_n$. We then have

$$\begin{aligned}S(\mathbf{x}_1, d_1) \cap S(\mathbf{x}_2, d_2) &= (nd_1)S(\widehat{\mathbf{x}}, \widehat{d}) + (\mathbf{x}_1 - d_1 \mathbf{1}_n) \\ &= (nd_1) \left[(n\widehat{d})S_n + (\widehat{\mathbf{x}} - \widehat{d}\mathbf{1}_n) \right] + (\mathbf{x}_1 - d_1 \mathbf{1}_n) \\ &= n(nd_1 \widehat{d})S_n + \left[(\mathbf{x}_1 + nd_1 \widehat{\mathbf{x}} - d_1 \mathbf{1}_n) - (nd_1 \widehat{d})\mathbf{1}_n \right] \\ &= S(\mathbf{x}_1 + nd_1 \widehat{\mathbf{x}} - d_1 \mathbf{1}_n, nd_1 \widehat{d}) = S(\mathbf{x}_3, d_3),\end{aligned}$$

where

$$\begin{aligned}d_3 &= nd_1 \widehat{d} = \frac{1 + \sum_{i=1}^n \min\{d_1 - x_1(i), d_2 - x_2(i)\}}{n}, \\ x_3(i) &= nd_1 \widehat{x}(i) + (x_1(i) - d_1) \\ &= \max\{d_2, x_2(i) - x_1(i) + d_1\} + (nd_1 \widehat{d} - d_2) + (x_1(i) - d_1) \\ &= \max\{d_2, x_2(i) - x_1(i) + d_1\} + (x_1(i) - d_1 - d_2) + d_3 \\ &= \max\{x_1(i) - d_1, x_2(i) - d_2\} + d_3.\end{aligned}$$

Thus, we have proven (3.4).

(3) Since the optimal solution $\mathbf{y}^*$ lies on the extreme point of $S(\mathbf{x}, d)$, we have

$$\begin{aligned}\mathbf{y}^* &= \arg \min_{\mathbf{y}=\mathbf{x}+d(n\mathbf{e}_i-\mathbf{1}_n), i \in [n]} \langle \mathbf{c}, \mathbf{x} + d(n\mathbf{e}_i - \mathbf{1}_n) \rangle \\ &= \arg \min_{\mathbf{y}=\mathbf{x}+d(n\mathbf{e}_i-\mathbf{1}_n), i \in [n]} \langle \mathbf{c}, \mathbf{e}_i \rangle = \mathbf{x} + d(n\mathbf{e}_{i^*} - \mathbf{1}_n),\end{aligned}$$

where $i^* = \arg \min_{i \in [n]} c_i$.

(4) By (3.1), for any points $\mathbf{y}_1, \mathbf{y}_2 \in S(\mathbf{x}, d)$, there exist $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \in S_n$ such that $\mathbf{y}_i = (\mathbf{x} - d\mathbf{1}_n) + nd\boldsymbol{\lambda}_i$ for $i \in [2]$. Thus, we have

$$\max_{\mathbf{y}_1, \mathbf{y}_2 \in S(\mathbf{x}, d)} \|\mathbf{y}_1 - \mathbf{y}_2\| = nd \cdot \max_{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \in S_n} \|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\| = \sqrt{2}nd.$$

(5) Let $\boldsymbol{\lambda} = \frac{1}{n}(\mathbf{1}_n + \frac{\mathbf{y}-\mathbf{x}}{d})$. Since $\|\mathbf{y} - \mathbf{x}\| \leq d$, we have $\lambda_i \geq 0$. Moreover, since $\mathbf{x}, \mathbf{y} \in S_n$, we have $\sum_{i=1}^n \lambda_i = 1$, which implies $\boldsymbol{\lambda} \in S_n$. Thus by (3.1),

$\mathbf{y} = (\mathbf{x} - d\mathbf{1}_n) + nd\boldsymbol{\lambda} \in S(\mathbf{x}, d)$. We now turn to the last part of Lemma 3.1(5). This simply follows from

$$\max_{\mathbf{y} \in S(\mathbf{x}, d)} \|\mathbf{y} - \mathbf{x}\| = nd \cdot \max_{\boldsymbol{\lambda} \in S_n} \left\| \boldsymbol{\lambda} - \frac{\mathbf{1}_n}{n} \right\| = \sqrt{n(n-1)}d \leq nd.$$

The proof is completed. $\square$

B Proofs for Section 4

B.1 A Useful Bound

The proof of Lemma 4.3 relies on the following lemma, whose proof used some key technical results established in [14]. In particular, for given $\mathbf{x}, \mathbf{y} \in \mathcal{P}$ and $\boldsymbol{\lambda} \in \mathcal{M}(\mathbf{x})$, there must exist $\mathbf{z} \in \mathcal{P}$ and $\gamma \in [0, 1]$ such that

$$\mathbf{y} = \gamma\mathbf{x} + (1-\gamma)\mathbf{z} = \gamma \sum_{j=1}^N \lambda_j \mathbf{v}_j + (1-\gamma)\mathbf{z} = \sum_{i=1}^N \left(\lambda_j - \lambda_j(1-\gamma) \right) \mathbf{v}_j + (1-\gamma)\mathbf{z}.$$

Let $\Delta_j := \lambda_j(1-\gamma) \in [0, \lambda_j]$. We then have $1-\gamma = \sum_{j=1}^N \Delta_j =: \Delta$. To put another way, the point $\mathbf{y}$ can always be represented by

$$\mathbf{y} = \sum_{j=1}^N (\lambda_j - \Delta_j) \mathbf{v}_j + \Delta \mathbf{z}, \quad (\text{B.1})$$

for some $\mathbf{z} \in \mathcal{P}$, $\Delta_j \in [0, \lambda_j]$. Since $\mathcal{P}$ is compact. There must exist a representation of (B.1) with the smallest Δ among all such representations. An important fact established in [14, lemma 5.3] is that the minimal value Δ can be bounded. We refine this bound below for the largest Δ_j in Δ .

Lemma B.1. *Let $\mathbf{x}, \mathbf{y} \in \mathcal{P}$ with $\boldsymbol{\lambda} \in \mathcal{M}(\mathbf{x})$. Let $\mathbf{y}$ be represented as in (B.1) with Δ having been minimized. Then it holds that*

$$\max_{i \in [N]} \{\Delta_i\} \leq \frac{\psi}{\xi} \|\mathbf{x} - \mathbf{y}\|.$$

Proof. The claim is trivial for the case $\sum_{i=1}^N \Delta_i = 0$. Now we suppose that $\sum_{i=1}^N \Delta_i > 0$ (i.e., at least one $\Delta_i > 0$). The following index sets $C(\mathbf{z})$ and $C_0(\mathbf{z})$ are defined in [14]. We simply describe them and use some established results relating to them. Denote the index set $C(\mathbf{z}) := \{j \in [m] \mid A_2(j)\mathbf{z} = b_2(j)\}$. By [14, Lemma 5.3] we have $C(\mathbf{z}) \neq \emptyset$ since one

$\Delta_i > 0$. Let $C_0(\mathbf{z}) \subseteq C(\mathbf{z})$ be such that the set $\{A_2(j)\}_{j \in C_0(\mathbf{z})}$ forms a basis for the set $\{A_2(j)\}_{j \in C(\mathbf{z})}$. Denote by $A_{2,z} \in \mathbb{R}^{|C_0(\mathbf{z})| \times n}$ consisting of the set $\{A_2(j)\}_{j \in C_0(\mathbf{z})}$. By definition we have $\|A_{2,z}\| \leq \psi$. Then we obtain

$$\begin{aligned} \|\mathbf{x} - \mathbf{y}\|^2 &= \left\| \sum_{i \in [N]: \Delta_i > 0} \Delta_i (\mathbf{v}_i - \mathbf{z}) \right\|^2 \\ &\geq \frac{1}{\psi^2} \sum_{j \in C_0(\mathbf{z})} \left(\sum_{i \in [N]: \Delta_i > 0} \Delta_i (b_2(j) - A_2(j) \mathbf{v}_i) \right)^2 \\ &\stackrel{(a)}{\geq} \frac{1}{\psi^2} \sum_{j \in C_0(\mathbf{z})} \sum_{i \in [N]: \Delta_i > 0} \Delta_i^2 (b_2(j) - A_2(j) \mathbf{v}_i)^2 \\ &= \frac{1}{\psi^2} \sum_{i \in [N]: \Delta_i > 0} \sum_{j \in C_0(\mathbf{z})} \Delta_i^2 (b_2(j) - A_2(j) \mathbf{v}_i)^2, \end{aligned}$$

where the first inequality is established in the proof of [14, Lemma 5.5], (a) follows from the fact that for any $i \in [N]$, and any $j \in C_0(\mathbf{z})$ we have $b_2(j) - A_2(j) \mathbf{v}_i \geq 0$. Combining [14, Lemma 5.3] and [14, Lemma 5.4], we obtain that for all $i \in [N]$ such that $\Delta_i > 0$ there exists $j \in C_0(\mathbf{z})$ such that $b_2(j) - A_2(j) \mathbf{v}_i \geq \xi$. Hence,

$$\|\mathbf{x} - \mathbf{y}\|^2 \geq \frac{\xi^2}{\psi^2} \sum_{i \in [N]: \Delta_i > 0} \Delta_i^2 \geq \frac{\xi^2}{\psi^2} \max_{i \in [N]} \{\Delta_i^2\}.$$

Thus we conclude that $\max_{i \in [N]} \{\Delta_i\} \leq \frac{\psi}{\xi} \|\mathbf{x} - \mathbf{y}\|$. $\square$

B.2 Proof of Lemma 4.3

Proof. We begin by proving the first part. Write $\mathbf{x} = \sum_{i=1}^N \lambda_i \mathbf{v}_i$ for $\boldsymbol{\lambda} \in S_N$ and express $\mathbf{y} = \sum_{i=1}^N (\lambda_i - \Delta_i) \mathbf{v}_i + (\sum_{i=1}^N \Delta_i) \mathbf{z}$, where $\Delta_i \in [0, \lambda_i]$, $\forall i \in [N]$ and $\mathbf{z} \in \mathcal{P}$. Here, the sum $\Delta = \sum_{i=1}^N \Delta_i$ is minimized (as in Lemma B.1). We then have

$$\max_{i \in [N]} \{\Delta_i\} \leq \frac{\psi}{\xi} \|\mathbf{x} - \mathbf{y}\| \leq \frac{\psi}{\xi} \times \frac{dD}{\eta} = d,$$

where the first inequality used Lemma B.1, the second inequality used the assumption $\|\mathbf{x} - \mathbf{y}\| \leq (dD)/\eta$, and the last equation is by the definition of η in (2.6). Express $\mathbf{z}$ as $\mathbf{z} = \sum_{i=1}^N \lambda'_i \mathbf{v}_i$, where $\boldsymbol{\lambda}' \in S_N$. We can then rewrite

$\mathbf{y}$ as follows:

$$\mathbf{y} = \sum_{i=1}^N (\lambda_i - \Delta_i + \Delta \lambda'_i) \mathbf{v}_i = \sum_{i=1}^N ((\lambda_i - d) + d - \Delta_i + \Delta \lambda'_i) \mathbf{v}_i.$$

Since $\max_{i \in [N]} \{\Delta_i\} \leq d$, we have $d - \Delta_i + \Delta \lambda'_i \geq 0$ for all $i \in [N]$. Moreover, the sum $\sum_{i=1}^N (d - \Delta_i + \Delta \lambda'_i) = Nd$, which implies that $\frac{(d - \Delta_i + \Delta \lambda'_i)_i}{Nd} \in S_N$. By the definition in (3.1), we have $\boldsymbol{\lambda}_y := (\lambda_i - \Delta_i + \Delta \lambda'_i) \in S(\boldsymbol{\lambda}, d)$, thus $\mathbf{y} \in S_{\mathcal{P}}(\mathbf{x}, d)$. Moreover, we have

$$\langle \mathbf{y}, \mathbf{c} \rangle = \langle \boldsymbol{\lambda}_y, \mathbf{c}_{ext} \rangle \geq \langle \boldsymbol{\lambda}^*, \mathbf{c}_{ext} \rangle = \langle \mathbf{y}^*, \mathbf{c} \rangle.$$

We now turn to prove the second part. Referring to Algorithm 4, we note that

$$\boldsymbol{\lambda}_+ - d\mathbf{1}_N = \max\{\boldsymbol{\lambda}_x, d\mathbf{1}_N\} - d\mathbf{1}_N = \boldsymbol{\lambda}_x - \min\{\boldsymbol{\lambda}_x, d\mathbf{1}_N\}$$

and

$$N\hat{d} = \sum_{i=1}^N \min\{\lambda_x(i), d\}.$$

Denote $\mathcal{I}_+(\boldsymbol{\lambda}_x) := \{i \in [N] \mid \lambda_x(i) > 0\}$, $\delta_i := \min\{\lambda_x(i), d\}$, and $\delta := \sum_{i \in \mathcal{I}_+(\boldsymbol{\lambda}_x)} \delta_i$, the optimal solution $\mathbf{y}^*$ produced by Algorithm 4 has

$$\mathbf{y}^* = \boldsymbol{\lambda}_+ + d\mathbf{1}_N + N\hat{d}\mathbf{e}_{i^*} = \sum_{i \in \mathcal{I}_+(\boldsymbol{\lambda}_x)} (\lambda_x(i) - \delta_i) \mathbf{v}_i + \delta \mathbf{v}_{i^*},$$

Thus we have that

$$\begin{aligned} \|\mathbf{x} - \mathbf{y}^*\| &= \left\| \sum_{i \in \mathcal{I}_+(\boldsymbol{\lambda}_x)} \min\{\lambda_x(i), d\} (\mathbf{v}_i - \mathbf{v}_{i^*}) \right\| \\ &\leq \sum_{i \in \mathcal{I}_+(\boldsymbol{\lambda}_x)} \min\{\lambda_x(i), d\} \|\mathbf{v}_i - \mathbf{v}_{i^*}\| \leq |\mathcal{I}_+(\boldsymbol{\lambda}_x)| dD \leq (n+1)dD. \end{aligned}$$

□

B.3 Proof of Theorem 4.4

Proof. The proof follows the framework of the proof of Theorem 3.4 and Lemma 4.3. We first claim that $\mathbf{x}^* \in S_{\mathcal{P}}(\mathbf{x}_k, \frac{\eta}{D}d_k)$ and that $f(\mathbf{x}_k) - B_k \leq \frac{\mu d_k^2}{2}$. We prove this by induction. First, we have

$$\frac{\mu d_0^2}{2} = f(\mathbf{x}_0) - B_0 \geq f(\mathbf{x}_0) - f^* \stackrel{(a)}{\geq} \frac{\mu}{2} \|\mathbf{x}_0 - \mathbf{x}^*\|^2,$$

where (a) comes from (2.1). This implies that $\|\mathbf{x}_0 - \mathbf{x}^*\| \leq d_0$, and by Lemma 4.3, we have $\mathbf{x}^* \in S_{\mathcal{P}}(\mathbf{x}_0, \frac{\eta}{D}d_0)$. Therefore, the claim holds for $k = 0$.

Now suppose that $\mathbf{x}^* \in S_{\mathcal{P}}(\mathbf{x}_t, \frac{\eta}{D}d_t)$ and $f(\mathbf{x}_t) - B_t \leq \frac{\mu d_t^2}{2}$ for all $t \leq k-1$. Let $\gamma := \frac{\mu}{2L(n+1)^2\eta^2}$. In the same manner as the proof in Theorem 3.4, for step size policy (2.3), (2.4) or (4.5), we all have

$$\begin{aligned} f(\mathbf{x}_k) - B_k &\leq (1 - \gamma)(f(\mathbf{x}_{k-1}) - B_{k-1}) + \frac{L\gamma^2}{2}\|\mathbf{y}_k - \mathbf{x}_{k-1}\|^2 \\ &\stackrel{(d)}{\leq} (1 - \gamma)\frac{\mu}{2}d_{k-1}^2 + \frac{L\gamma^2}{2}(n+1)^2\eta^2d_{k-1}^2 \\ &= \left[(1 - \gamma)\frac{\mu}{2} + \frac{L\gamma^2(n+1)^2\eta^2}{2} \right] d_{k-1}^2, \end{aligned}$$

where (d) is due to our inductive hypothesis and Lemma 4.3. By plugging in the value of γ , and using $1 - x \leq e^{-x}$, we have that

$$f(\mathbf{x}_k) - B_k \leq \frac{\mu}{2} \left(1 - \frac{\mu}{4L(n+1)^2\eta^2}\right) d_{k-1}^2 \leq \frac{\mu}{2} e^{-\frac{\mu}{4L(n+1)^2\eta^2}} d_{k-1}^2.$$

Combining the above inequality with the fact that $f(\mathbf{x}_k) - B_k = \frac{\mu}{2} \left(\sqrt{\frac{2(f(\mathbf{x}_k) - B_k)}{\mu}} \right)^2$, and by the definition of d_k , we conclude that $f(\mathbf{x}_k) - B_k \leq \frac{\mu d_k^2}{2}$. By the inductive hypothesis, we know that $\mathbf{x}^* \in S_{\mathcal{P}}(\mathbf{x}_t, d_t)$ holds for all $t \leq k-1$. Thus B_{t+1}^w is a valid lower bound of f^* , and consequently, B_k is also a lower bound of f^* . Now by (2.1), we have

$$\|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq (2/\mu)(f(\mathbf{x}_k) - f^*) \leq (2/\mu)(f(\mathbf{x}_k) - B_k) \leq d_k^2. \quad (\text{B.2})$$

This implies that $\mathbf{x}^* \in S_{\mathcal{P}}(\mathbf{x}_k, \frac{\eta}{D}d_k)$ by Lemma 4.3. Therefore, we have completed the proof of the claim.

We now start to prove the conclusion in Theorem 4.4. From the earlier proof, we know that $B_k \leq f^*$, thus confirming the first part of the inequality. By the definition of d_k and the established claim, we have

$$f(\mathbf{x}_k) - B_k \leq \frac{1}{2}\mu d_k^2 \leq \frac{\mu d_0^2}{2} e^{-\frac{\mu}{4L(n+1)^2\eta^2}k}.$$

The proof is thus completed. $\square$

B.4 Proof of Theorem 4.5

The proof of Theorem 4.5 relies on the following lemma.

Lemma B.2. For any $\lambda_1, \lambda_2 \in S(\lambda_x, d)$, define the two corresponding points:

$$\mathbf{y}_j = \sum_{i=1}^N \lambda_j(i) \mathbf{v}_i \in S_{\mathcal{P}}(\mathbf{x}, d), \quad j = 1, 2.$$

We must have $\|\mathbf{y}_1 - \mathbf{y}_2\| \leq (n+1)dD$.

Proof. We note that $S_{\mathcal{P}}(\mathbf{x}, d)$ is compact. Let $\mathcal{V}(S)$ denote the set of its vertices. Obviously, we have

$$\|\mathbf{y}_1 - \mathbf{y}_2\| \leq \max_{\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{V}(S)} \|\mathbf{u}_1 - \mathbf{u}_2\|.$$

For $\mathbf{u}_1 \in \mathcal{V}(S)$ being a vertex of $S_{\mathcal{P}}(\mathbf{x}, d)$, according to the separation theorem [28] that there exists a vector $\mathbf{c}_1 \in \mathbb{R}^N$ such that

$$\langle \mathbf{c}_1, \mathbf{u}_1 \rangle < \langle \mathbf{c}_1, \mathbf{u} \rangle \quad \text{for all } \mathbf{u} \in \mathcal{V}(S) \setminus \{\mathbf{u}_1\}.$$

In other words, $\mathbf{u}_1$ is the unique solution of the following problem:

$$\min \langle \mathbf{c}_1, \mathbf{u} \rangle \quad \text{s.t.} \quad \mathbf{u} \in S_{\mathcal{P}}(\mathbf{x}, d) \cap \mathcal{P}.$$

It follows Lemma 4.2 that $\mathbf{u}_1$ can be represented as

$$\mathbf{u}_1 = \sum_{j=1}^N (\lambda_x(j) - \delta_j) \mathbf{v}_j + \delta \mathbf{z}_1 \quad \text{for some } \mathbf{z}_1 \in \mathcal{P},$$

where $\delta_j := \min\{\lambda_x(j), d\}$ and $\delta := \sum_{j=1}^N \delta_j$ independent of $\mathbf{z}_1$. Similarly, $\mathbf{u}_2$ has a representation:

$$\mathbf{u}_2 = \sum_{j=1}^N (\lambda_x(j) - \delta_j) \mathbf{v}_j + \delta \mathbf{z}_2 \quad \text{for some } \mathbf{z}_2 \in \mathcal{P}.$$

Therefore, we have

$$\begin{aligned} \|\mathbf{y}_1 - \mathbf{y}_2\| &\leq \max_{\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{V}(S)} \|\mathbf{u}_1 - \mathbf{u}_2\| \\ &\leq \delta \max_{\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{P}} \|\mathbf{z}_1 - \mathbf{z}_2\| \leq \delta D \\ &= D \sum_{i \in \mathcal{I}_+(\lambda_x)} \delta_i \leq D |\mathcal{I}_+(\lambda_x)| d \leq (n+1)dD, \end{aligned}$$

where $\mathcal{I}_+(\lambda_x) := \{i \mid \lambda_i(x) > 0\}$ and we have used $|\mathcal{I}_+(\lambda_x)| \leq (n+1)$. $\square$

Proof of Theorem 4.5. We first claim that $\mathbf{x}^* \in S_{\mathcal{P}}(\mathbf{x}_k, d_k)$ for any $k \geq 0$ and prove this by induction. From the proof of Theorem 4.4, this is true for $k = 0$. Now suppose that $\mathbf{x}^* \in S_{\mathcal{P}}(\mathbf{x}_{k-1}, d_{k-1})$ for some $k \geq 1$. Note that the inner loop of Alg. 6 corresponds to the standard Frank-Wolfe algorithm. By Theorem 2.1 and Lemma B.2, we have

$$f(\mathbf{p}_j) - f^* \leq \frac{2L}{j+1} ((n+1)^2 d_{k-1} D)^2 = \frac{2L(n+1)^2 d_{k-1}^2 D^2}{j+1}$$

hold for all $j \in [J]$. In the case where the inner loop terminates at $j = J$, we obtain

$$f(\mathbf{x}_k) - f^* = f(\mathbf{p}_J) - f^* \leq f(\mathbf{p}_J) - C_J \leq \frac{\mu d_k^2 D^2}{2\eta^2}.$$

Similarly, if the inner loop is interrupted due to lines 9-11 of the algorithm, we still have $f(\mathbf{x}_k) - f^* \leq f(\mathbf{p}_j) - C_j \leq \frac{\mu}{2\rho^2\eta^2} d_{k-1}^2 D^2 = \frac{\mu d_k^2 D^2}{2\eta^2}$. Using the fact that $f(\mathbf{x}_k) - f^* \geq \frac{\mu}{2} \|\mathbf{x}_k - \mathbf{x}^*\|^2$, we have $\|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq \frac{d_k^2 D^2}{\eta^2}$, which implies via Lemma 4.3 that $\mathbf{x}^* \in S_{\mathcal{P}}(\mathbf{x}_k, d_k)$.

We now start to prove the conclusion in Theorem 4.5. Since $d_0 = \frac{\eta}{D} \sqrt{\frac{2(f(\mathbf{x}_0) - B_0)}{\mu}}$ and $d_k = \frac{d_{k-1}}{\rho}$, we have

$$f(\mathbf{x}_k) - f^* \leq f(\mathbf{x}_k) - B_k \leq \frac{\mu d_k^2 D^2}{2\eta^2} \leq (f(\mathbf{x}_0) - B_0) \rho^{-2k}.$$

□

C Properties of Some Common Polytopes

C.1 Carath odory Representation Examples

Hypercube: When $\mathcal{P}$ is a hypercube $B_n := \{\mathbf{x} \in \mathbb{R}^n \mid x_i \in [0, 1], \forall i \in [n]\}$, any point $\mathbf{x} \in B_n$ can be naturally represented as

$$\mathbf{x} = \sum_{i=1}^{n-1} (x_{j_i} - x_{j_{i+1}}) \mathbf{v}_i + x_{j_n} \mathbf{1}_n + (1 - x_{j_1}) \mathbf{0}_n,$$

where $j_1, \dots, j_n$ is a permutation over $[n]$ such that $x_{j_1} \geq \dots \geq x_{j_n}$ and $\mathbf{v}_i$ is a vector with components from j_1 to j_i equal to 1 and the rest equal to 0.

ℓ_1 -ball: When $\mathcal{P}$ is a ℓ_1 -ball $L_n := \{\mathbf{x} \in \mathbb{R}^n \mid \sum_{i=1}^n |x_i| \leq 1\}$, any point $\mathbf{x} \in L_n$ can be naturally represented as

$$\mathbf{x} = \sum_{i=1}^{n-1} |x_i|(\text{sgn}(x_i)\mathbf{e}_i) + (|x_n| + s_x)(\text{sgn}(x_n)\mathbf{e}_n) + s_x(-\text{sgn}(x_n)\mathbf{e}_n),$$

where $s_x = 1 - \sum_{i=1}^n |x_i|/2$ and $\text{sgn}(x) = 1$ if $x \geq 0$ and -1 otherwise.

Flow polytope: Let G be a *directed acyclic graph* (DAG) with a set of vertices V and edges E such that $|E| = n$. Let s, t be two vertices in V , referred to as the *source* and *target*, respectively. The s - t flow polytope, here denoted by $\mathcal{F}_{s,t}$, is the set of all unit s - t flows in G . For any point $\mathbf{x} \in \mathcal{F}_{s,t}$ and $i \in [n]$, the entry x_i represents the amount of flow through edge $i \in [n]$, where the flow vector $\mathbf{x}$ satisfies the flow conservation constraints at each vertex, ensuring that the flow entering any vertex (except s and t) equals the flow leaving it. The extreme points of $\mathcal{F}_{s,t}$ are the extreme unit flows. To find the Carathéodory representation of a given flow $\mathbf{x} \in \mathcal{F}_{s,t}$, we can proceed recursively as follows.

Starting with the flow $\mathbf{x}$, we repeatedly perform the following steps until $\mathbf{x} = \mathbf{0}_n$:

1. Remove all edges with zero flow from the graph.
2. Identify the edge i corresponding to the smallest non-zero flow in $\mathbf{x}$, i.e., $i \leftarrow \arg \min_{x_i > 0} x_i$.
3. Find the extreme unit flow $\mathbf{v}$ in the reduced graph that includes edge i .
4. Subtract $x_i \mathbf{v}$ from the current flow, i.e., $\mathbf{x} \leftarrow \mathbf{x} - x_i \mathbf{v}$.

Since each operation eliminates at least one non-zero entry in the current flow, the loop will terminate within at most m steps. As a result, we obtain a Carathéodory representation of $\mathbf{x}$. This algorithm can be implemented in $O(n^2)$ time when the graph is represented using sparsely structured adjacency matrices.

C.2 Quantities of Some Common Polytopes

Hypercube: The diameter of B_n is given by

$$D(B_n) = \max_{\mathbf{x}, \mathbf{y} \in B_n} \|\mathbf{x} - \mathbf{y}\| = \|\mathbf{1}_n - \mathbf{0}_n\| = \sqrt{n}.$$

Since B_n can be represented as

$$B_n = \left\{ \mathbf{x} \in \mathbb{R}^n \mid \begin{pmatrix} I_n \\ -I_n \end{pmatrix} \mathbf{x} \leq \begin{pmatrix} \mathbf{1}_n \\ \mathbf{0}_n \end{pmatrix} \right\},$$

it follows from the definition in Subsection 2.3 that $\xi(B_n) = 1$ and $\psi(B_n) = 1$. Thus, the quantity of B_n is

$$\eta(B_n) = \psi(B_n)D(B_n)/\xi(B_n) = \sqrt{n}.$$

ℓ_1 -ball: The diameter of L_n is given by

$$D(L_n) = \max_{\mathbf{x}, \mathbf{y} \in L_n} \|\mathbf{x} - \mathbf{y}\| = \|\mathbf{e}_1 - (-\mathbf{e}_1)\| = 2.$$

Note that L_n can be described by the linear inequalities system $L_n = \{\mathbf{x} \in \mathbb{R}^n \mid A_2 \mathbf{x} \leq \frac{1}{\sqrt{n}} \mathbf{1}_{2^n}\}$, where $A_2 \in \mathbb{R}^{2^n \times n}$ is a matrix whose entries are either $\pm \frac{1}{\sqrt{n}}$ and whose rows all have unit ℓ_2 norm. Following the definition in Subsection 2.3, we have $\xi(L_n) = \frac{2}{\sqrt{n}}$ and

$$\frac{1}{\sqrt{n}} = \max_{M \in \mathbb{A}(L_n)} \frac{1}{\sqrt{n}} \|M\|_F \leq \psi(L_n) = \max_{M \in \mathbb{A}(L_n)} \|M\| \leq \max_{M \in \mathbb{A}(L_n)} \|M\|_F = \sqrt{n},$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Thus, the quantity of L_n can be estimated as

$$\eta(L_n) = \psi(L_n)D(L_n)/\xi(L_n) \in [1, n].$$

Flow Polytope: For every two extreme unit flows $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{V}(\mathcal{F}_{s,t})$, since $\mathcal{V}(\mathcal{F}_{s,t}) \subseteq \{0, 1\}^n$, we have $\|\mathbf{x}_1 - \mathbf{x}_2\| \leq \sqrt{n}$. Thus, the diameter of $\mathcal{F}_{s,t}$ can be estimated as

$$D(\mathcal{F}_{s,t}) = \max_{\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{V}(\mathcal{F}_{s,t})} \|\mathbf{x}_1 - \mathbf{x}_2\| \leq \sqrt{n}.$$

When representing $\mathcal{F}_{s,t}$ using a system of linear equations and inequalities, the inequality constraints are given by $-I_n \mathbf{x} \leq \mathbf{0}_n$. Thus, by definition, we have $\xi(\mathcal{F}_{s,t}) = \psi(\mathcal{F}_{s,t}) = 1$, leading to

$$\eta(\mathcal{F}_{s,t}) = D(\mathcal{F}_{s,t}) \leq \sqrt{n}.$$

It is worth noting that in the numerical experiment in Subsection 5.2, we set $\eta = D = \sqrt{66}$ while the dimension is $n = 660$. This choice stems from the specific characteristics of the dataset used in [22, 15]. Specifically, we observe that each extreme unit flow contains exactly 33 entries of 1, with the remaining entries being 0. Consequently, we can estimate $\eta = D \leq \sqrt{66}$.

D Backtracking Details

The routine of estimating local parameters L, μ and step-size δ is shown as follows. For rSFW, if we use the simple step-size, we can employ only the estimates of L and μ without applying the corresponding step size.

Algorithm 7 Backtracking-Routine($\mathbf{x}, \mathbf{d}, L, \mu, \delta_{\max}$)

Input: Iterate $\mathbf{x} \in \mathcal{P}$, update direction $\mathbf{d}$, previous estimation L and μ .

```

1: Choose  $\tau_1 > 1, \tau_2 \leq 1$ .
2:  $L \leftarrow \tau_2 L, \mu \leftarrow \mu / \tau_2$ 
3:  $\delta \leftarrow \min \left\{ \frac{\langle \nabla f(\mathbf{x}), \mathbf{d} \rangle}{L \|\mathbf{d}\|^2}, \delta_{\max} \right\}$ 
4: while  $f(\mathbf{x} + \delta \mathbf{d}) > f(\mathbf{x}) + \delta \langle \nabla f(\mathbf{x}), \mathbf{d} \rangle + \frac{\delta^2 L}{2} \|\mathbf{d}\|^2$  do
5:    $L \leftarrow \tau_1 L$ 
6:    $\mu \leftarrow \min \left\{ \frac{2(f(\mathbf{x} + \delta \mathbf{d}) - f(\mathbf{x}) - \delta \langle \nabla f(\mathbf{x}), \mathbf{d} \rangle)}{\delta^2 \|\mathbf{d}\|^2}, \mu \right\}$ 
7:    $\delta \leftarrow \min \left\{ \frac{\langle \nabla f(\mathbf{x}), \mathbf{d} \rangle}{L \|\mathbf{d}\|^2}, \delta_{\max} \right\}$ 
8: end while

```

Output: δ, L, μ .

References

- [1] Beck, A., Teboulle, M.: A conditional gradient method with linear rate of convergence for solving convex linear systems. *Mathematical Methods of Operations Research* **59**, 235–247 (2004)
- [2] Bernd, G., Jaggi, M.: Optimization for machine learning. *Lecture Notes CS-439, ETH, Spring 2023* (2023)
- [3] Bomze, I.M., Rinaldi, F., Zeffiro, D.: Frank–wolfe and friends: a journey into projection-free first-order optimization methods. *4OR* **19**(3), 313–345 (2021)
- [4] Braun, G., Carderera, A., Combettes, C.W., Hassani, H., Karbasi, A., Mokhtari, A., Pokutta, S.: Conditional gradient methods. *arXiv preprint arXiv:2211.14103* (2022)
- [5] Chandrasekaran, V., Recht, B., Parrilo, P.A., Willsky, A.S.: The convex geometry of linear inverse problems. *Foundations of Computational mathematics* **12**(6), 805–849 (2012)

- [6] Clarkson, K.L.: Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)* **6**(4), 1–30 (2010)
- [7] Combettes, C.W., Pokutta, S.: Complexity of linear minimization and projection on some sets. *Operations Research Letters* **49**(4), 565–571 (2021)
- [8] Condat, L.: Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming* **158**(1), 575–585 (2016)
- [9] Damla Ahipasaoglu, S., Sun, P., Todd, M.J.: Linear convergence of a modified frank-wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimisation Methods and Software* **23**(1), 5–19 (2008)
- [10] Frank, M., Wolfe, P., et al.: An algorithm for quadratic programming. *Naval research logistics quarterly* **3**(1-2), 95–110 (1956)
- [11] Freund, R.M., Grigas, P.: New analysis and results for the frank-wolfe method. *Mathematical Programming* **155**(1), 199–230 (2016)
- [12] Garber, D.: Faster projection-free convex optimization over the spectrahedron. *Advances in Neural Information Processing Systems* **29** (2016)
- [13] Garber, D., Hazan, E.: Playing non-linear games with linear oracles. In: 2013 IEEE 54th annual symposium on foundations of computer science, pp. 420–428. IEEE (2013)
- [14] Garber, D., Hazan, E.: A linearly convergent variant of the conditional gradient algorithm under strong convexity, with applications to online and stochastic optimization. *SIAM Journal on Optimization* **26**(3), 1493–1528 (2016)
- [15] Garber, D., Wolf, N.: Frank-wolfe with a nearest extreme point oracle. In: *Conference on Learning Theory*, pp. 2103–2132. PMLR (2021)
- [16] Guélat, J., Marcotte, P.: Some comments on wolfe’s ‘away step’. *Mathematical Programming* **35**(1), 110–119 (1986)
- [17] Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A.: *Feature extraction: foundations and applications*, vol. 207. Springer (2008)

- [18] Hazan, E.: Sparse approximate solutions to semidefinite programs. In: Latin American symposium on theoretical informatics, pp. 306–316. Springer (2008)
- [19] Jaggi, M.: Revisiting frank-wolfe: Projection-free sparse convex optimization. In: International conference on machine learning, pp. 427–435. PMLR (2013)
- [20] Jaggi, M., Sulovsk, M., et al.: A simple algorithm for nuclear norm regularized problems. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp. 471–478 (2010)
- [21] Joulin, A., Tang, K., Fei-Fei, L.: Efficient image and video co-localization with frank-wolfe algorithm. In: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (eds.) Computer Vision – ECCV 2014, pp. 253–268. Springer International Publishing, Cham (2014)
- [22] Lacoste-Julien, S., Jaggi, M.: On the global linear convergence of frank-wolfe optimization variants. *Advances in neural information processing systems* **28** (2015)
- [23] Lan, G.: The complexity of large-scale convex programming under a linear optimization oracle. *arXiv preprint arXiv:1309.5550* (2013)
- [24] Lan, G.: First-order and stochastic optimization methods for machine learning, vol. 1. Springer (2020)
- [25] Levin, E.S., Polyak, B.T.: Constrained minimization methods. *USSR Computational mathematics and mathematical physics* **6**(5), 1–50 (1966)
- [26] Pedregosa, F., Negiar, G., Askari, A., Jaggi, M.: Linearly convergent frank-wolfe with backtracking line-search. In: International conference on artificial intelligence and statistics, pp. 1–10. PMLR (2020)
- [27] Pokutta, S.: The frank-wolfe algorithm: a short introduction. *Jahresbericht der Deutschen Mathematiker-Vereinigung* **126**(1), 3–35 (2024)
- [28] Rockafellar, R.T.: *Convex analysis*, vol. 11. Princeton university press (1997)
- [29] Végh, L.A.: Strongly polynomial algorithm for a class of minimum-cost flow problems with separable convex objectives. In: Proceedings of the forty-fourth annual ACM symposium on Theory of computing, pp. 27–40 (2012)

Table 3: Description of Frank-Wolfe variants used in the numerical comparison.

Algorithm	Description
FW (simple/Ada)	Frank-Wolfe with simple step size $\delta_k = 2/(k+1)$. The ‘Ada’ variant employs a backtracking step (5.1) prior to updating the iterate, in order to estimate the local parameters L and μ , thereby enabling an adaptive short step size. We set $\tau_1 = 2$ and $\tau_2 = 0.9$ in Alg. 7, and apply the same configuration to the subsequent ‘Ada’ variants.
SFW/SFW _{$\mathcal{P}$} (line-search)	Simplex Frank-Wolfe with exact line-search (Alg. 2 and Alg. 5). For the ℓ_1 -constrained least squares problem, we set $\mu = 2\lambda_{\min}(A'A)$, $D = 2$ and $\eta = \sqrt{n}$. Although Supplement C.2 estimates $\eta \leq n$ for ℓ_1 -ball, this setting does not hinder the algorithm’s linear convergence and demonstrates strong practical performance. For the video co-localization task, we set $\mu = \lambda_{\min}(A)$ and $\eta = D = \sqrt{66}$; see Supplement C.2 for details. For the Simplex-constrained least squared problem, we set $\mu = 2\lambda_{\min}(A'A)$.
NEP-FW (simple)	Frank-Wolfe with Nearest Extreme Point Oracle, with theoretical step size $2/(k+1)$ [15, Alg. 1]. We omitted Line 5, as it showed no noticeable effect on performance.
rSFW/rSFW _{$\mathcal{P}$} (simple)	Refined Simplex Frank-Wolfe with simple step size $\delta_j = 2/(j+1)$ (Alg. 3 and Alg. 6). We utilize the warm-start strategy with $\rho' = 2$ as mentioned in Remark. 4. The parameters μ, D and η are set the same as in SFW/SFW _{$\mathcal{P}$} . Additionally, we set $\rho = 1.01, L = 2\lambda_{\max}(A'A)$ for ℓ_1 /Simplex-constrained least squares problems, and $\rho = 1.01, L = \lambda_{\max}(A)$ for video co-localization problem.
PFW (line-search)	Pairwise Frank-Wolfe with exact line-search [22, Alg. 2].
AFW (line-search)	Away-steps Frank-Wolfe with exact line-search [22, Alg. 1].
rSFW-P (line-search)	The Refined Simplex Frank-Wolfe framework enhanced with Pairwise technique. Specifically, we incorporate (5.3) after Line 6 in Alg. 3 and replace Line 12 with (5.4).
rSFW-A (line-search)	The Refined Simplex Frank-Wolfe framework enhanced with Away-steps technique. Specifically, we incorporate (5.2) after Line 6 in Alg. 3 and replace Line 12 with (5.4).