Parameter-free Locally Accelerated Conditional Gradients

Alejandro Carderera [*1] **Jelena Diakonikolas** [*2] **Cheuk Yin (Eric) Lin** [*2] **Sebastian Pokutta** [*3]

Abstract

Projection-free conditional gradient (CG) methods are the algorithms of choice for constrained optimization setups in which projections are often computationally prohibitive but linear optimization over the constraint set remains computationally feasible. Unlike in projection-based methods, globally accelerated convergence rates are in general unattainable for CG. However, a very recent work on *Locally accelerated CG* (LaCG) has demonstrated that local acceleration for CG is possible for many settings of interest. The main downside of LaCG is that it requires knowledge of the smoothness and strong convexity parameters of the objective function. We remove this limitation by introducing a novel, Parameter-Free Locally accelerated CG (PF-LaCG) algorithm, for which we provide rigorous convergence guarantees. Our theoretical results are complemented by numerical experiments, which demonstrate local acceleration and showcase the practical improvements of PF-LaCG over non-accelerated algorithms, both in terms of iteration count and wall-clock time.

1. Introduction

Conditional gradient (CG) (or Frank-Wolfe (FW)) methods (Frank & Wolfe, 1956; Levitin & Polyak, 1966) are a fundamental class of projection-free optimization methods, most frequently used to minimize smooth convex objective functions over constrained sets onto which projections are computationally prohibitive; see Combettes & Pokutta (2021) for an overview. These methods have received significant recent attention in the machine learning and optimization communities, due to the fact that they eschew projections and produce solutions with sparse representations (Jaggi, 2013; Garber, 2016; Hazan & Luo, 2016; Braun et al., 2017; 2019; Lei et al., 2019; Tsiligkaridis & Roberts, 2020; Combettes et al., 2020).

While CG methods have been applied to many different problem settings (see, e.g., Hazan & Luo (2016); Zhou et al. (2018); Pedregosa et al. (2020); Lei et al. (2019); Tsiligkaridis & Roberts (2020); Dvurechensky et al. (2020); Zhang et al. (2020); Négiar et al. (2020); Carderera & Pokutta (2020); Kerdreux et al. (2021)), in this paper, we are interested in using CG-type methods to solve problems of the form:

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \tag{P}$$

where f is an L-smooth (gradient-Lipschitz) and m-strongly convex function and $\mathcal{X} \subseteq \mathbb{R}^n$ is a polytope.

We assume that we are given access to the objective function f and to the feasible set $\mathcal{X}$ via the following two oracles:

Oracle 1.1 (First Order Oracle (FOO))**.** Given $\mathbf{x} \in \mathcal{X}$, the FOO returns $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$.

Oracle 1.2 (Linear Minimization Oracle (LMO))**.** Given $\mathbf{c} \in \mathbb{R}^n$, the LMO returns $\operatorname{argmin}_{\mathbf{x}\in\mathcal{X}} \langle \mathbf{c}, \mathbf{x}\rangle$.

While being of extreme practical importance, these LMO-based methods in general do not achieve the *globally* optimal rates for smooth (and possibly strongly convex) minimization that are attained by projection-based methods. In particular, LMO-based algorithms cannot converge globally faster than $O(1/k)$ for the class of smooth strongly convex functions where k is the number of iterations (up to the dimension threshold n) (Lan, 2013; Jaggi, 2013). Moreover, the dependence of the convergence rate on the dimension is unavoidable in general.

At the same time, it was shown by Diakonikolas et al. (2020) that optimal rates can be obtained asymptotically, referred to as *locally optimal rates*. That is, after a finite number of iterations (independent of the target accuracy ϵ) with potentially sub-optimal convergence, optimal convergence rates can be achieved. While Diakonikolas et al. (2020) resolve the question of acceleration for CG-type methods, it unfortunately depends on the knowledge of m and L. Although the latter can be removed with the common line search-based arguments, the knowledge of a good estimate of the former is crucial in achieving acceleration. This makes the *Locally-*

[*]Equal contribution [1]Georgia Institute of Technology, Atlanta, GA, USA [2]Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, USA [3]Zuse Institute Berlin and Technische Universität Berlin, Berlin, Germany. Correspondence to: Jelena Diakonikolas <jelena@cs.wisc.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

accelerated Conditional Gradient (LaCG) algorithm from Diakonikolas et al. (2020), despite being of theoretical interest, of limited use in practice. Not only is it usually hard to come by a good estimate of the parameter m, but working with an estimated lower bound does not take advantage of the potentially better local strong convexity behavior in the vicinity of an iterate.

To remedy these shortcomings, we devise a new *Parameter-Free Locally-accelerated Conditional Gradient algorithm (PF-LaCG)* that is based on a similar coupling between a variant of the *Away-step Frank-Wolfe (AFW)* method (Guélat & Marcotte, 1986; Lacoste-Julien & Jaggi, 2015) and an accelerated method as used in LaCG. However, beyond this basic inspiration, not many things can be reused from Diakonikolas et al. (2020), as in order to achieve parameter-freeness, we need to devise a completely new algorithm employing a gradient-mapping technique that out-of-the-box is incompatible with the approach used in LaCG.

1.1. Contributions and Further Related Work

Our main contributions can be summarized as follows (see Section 3.1 for a detailed overview of the main technical ideas).

Near-optimal and parameter-free acceleration with inexact projections. To devise PF-LaCG, we introduce a parameter-free accelerated algorithm for smooth strongly convex optimization that utilizes *inexact* projections onto low-dimensional simplices. While near-optimal (i.e., optimal up to poly-log factors) parameter-free projection-based algorithms were known in the literature prior to our work (Nesterov, 2013; Ito & Fukuda, 2019), their reliance on exact projections which are computationally infeasible makes them unsuitable for our setting.

Parameter-free Locally-accelerated Conditional Gradient (PF-LaCG) algorithm. We propose a novel, parameter-free and locally accelerated CG-type method. Up to poly-logarithmic factors, our algorithm PF-LaCG attains an optimal accelerated local rate of convergence. PF-LaCG leverages efficiently computable projections onto low dimensional simplices, but is otherwise projection-free (i.e., it does not assume access to a projection oracle for $\mathcal{X}$). Local acceleration is achieved by coupling the parameter-free accelerated method with inexact projections described in the previous paragraph and AFW with a fractional exit condition (Kerdreux et al., 2019). This coupling idea is inspired by the coupling between μAGD+ (Cohen et al., 2018) (where AGD stands for Accelerated Gradient Descent) and AFW (Guélat & Marcotte, 1986) used in Diakonikolas et al. (2020); however, most of the similarities between our work and Diakonikolas et al. (2020) stop there, as there are major technical challenges that have to be overcome to attain the results in the parameter-free setting.

Computational experiments. We demonstrate the efficacy of PF-LaCG using numerical experiments, comparing the performance of the proposed algorithms to several relevant CG-type algorithms. The use of PF-LaCG brings demonstrably faster local convergence in primal gap with respect to both iteration count and wall-clock time.

1.2. Outline

In Section 2 we introduce the notation and preliminaries that are required for stating our main results. We then present our approach to parameter-free local acceleration in Section 3 and derive our main results. Finally, we demonstrate the practicality of our approach with computational experiments in Section 4, and conclude with a discussion in Section 5.

2. Notation and Preliminaries

We denote the unique minimizer of Problem (P) by $\mathbf{x}^*$. Let $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ denote the Euclidean norm and the standard inner product, respectively. We denote the *diameter* of the polytope $\mathcal{X}$ by $D = \max_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \|\mathbf{x}-\mathbf{y}\|$, and its *vertices* by $\operatorname{vert}(\mathcal{X}) \subseteq \mathcal{X}$. Given a non-empty set $\mathcal{S} \subset \mathbb{R}^n$, we denote its *convex hull* by $\operatorname{co}(\mathcal{S})$. For any $\mathbf{x} \in \mathcal{X}$ we denote by $\mathcal{F}(\mathbf{x})$ the *minimal face* of $\mathcal{X}$ that contains $\mathbf{x}$. We call a subset of vertices $\mathcal{S} \subseteq \operatorname{vert}(\mathcal{X})$ a *support* of $\mathbf{x} \in \mathcal{X}$ if $\mathbf{x}$ can be expressed as a convex combination of the elements of $\mathcal{S}$. A support $\mathcal{S}$ of $\mathbf{x}$ is a *proper support* of $\mathbf{x}$ if the weights associated with the convex decomposition are positive. Let $\mathcal{B}(\mathbf{x}, r)$ denote the ball around $\mathbf{x}$ with radius r with respect to $\|\cdot\|$. We say that $\mathbf{x}$ is r-*deep* in a convex set $\mathcal{C} \subseteq \mathbb{R}^n$ if $\mathcal{B}(\mathbf{x}, r) \cap \operatorname{aff}(\mathcal{C}) \subseteq \mathcal{C}$, where $\operatorname{aff}(\mathcal{C})$ denotes the smallest affine space that contains $\mathcal{C}$. The point $\mathbf{x}$ is contained in the *relative interior of* $\mathcal{C}$, $\mathbf{x} \in \operatorname{rel.int}(\mathcal{C})$, if there exists an $r > 0$ such that $\mathbf{x}$ is r-deep in $\mathcal{C}$.

Measures of optimality. The two key measures of optimality that we will use in this paper are the *Strong Wolfe Gap* and the *Gradient Mapping*. We define the former as:

Definition 2.1 (Strong Wolfe Gap)**.** Given $\mathbf{x} \in \mathcal{X}$, the *strong Wolfe gap* $w(\mathbf{x})$ of f over $\mathcal{X}$ is defined as

$$w(\mathbf{x}) := \min_{\mathcal{S}\in\mathcal{S}_\mathbf{x}} w(\mathbf{x}, \mathcal{S}),$$

where $\mathcal{S}_\mathbf{x}$ denotes the set of all proper supports of $\mathbf{x}$ and $w(\mathbf{x}, \mathcal{S}) := \max_{\mathbf{y}\in\mathcal{S},\mathbf{z}\in\mathcal{X}} \langle \nabla f(\mathbf{x}), \mathbf{y}-\mathbf{z}\rangle$.

Note that any polytope $\mathcal{X}$ satisfies what is known as a $\delta(\mathcal{X})$-scaling inequality, where $\delta(\mathcal{X})$ is the pyramidal width of the polytope (see, e.g., (Lacoste-Julien & Jaggi, 2015; Beck & Shtern, 2017; Peña & Rodríguez, 2019; Gutman & Peña, 2018)) and where the inequality is defined as:

Definition 2.2 (δ-scaling inequality)**.** There exists $\delta > 0$ such that for all $\mathbf{x} \in \mathcal{X} \setminus \mathbf{x}^*$ we have that Problem (P) satisfies $w(\mathbf{x}) \geq \delta(\mathcal{X}) \langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{x}^* \rangle / \|\mathbf{x} - \mathbf{x}^*\|$.

To implement a parameter-free variant of a projection-based accelerated algorithm, we will rely on a second measure of optimality, the *gradient mapping*. Recall that we do *not* assume the availability of projections onto the polytope $\mathcal{X}$; instead, our algorithm will only rely on low-complexity projections onto simplices spanned by a small number of vertices of $\mathcal{X}$ (see Diakonikolas et al. (2020) for a more detailed discussion). In the following, given a convex set $\mathcal{C}$, we denote the projection of $\mathbf{x} \in \mathbb{R}^n$ onto $\mathcal{C}$ by $P_{\mathcal{C}}(\mathbf{x})$.

Definition 2.3 (Gradient mapping)**.** Given a convex set $\mathcal{C} \subseteq \mathbb{R}^n$, a differentiable function $f : \mathcal{C} \to \mathbb{R}$, and a scalar $\eta > 0$, the gradient mapping of f w.r.t. $\eta, \mathcal{C}$ is defined by:

$$G_\eta(\mathbf{x}) = \eta\Big(\mathbf{x} - P_{\mathcal{C}}\Big(\mathbf{x} - \frac{1}{\eta}\nabla f(\mathbf{x})\Big)\Big).$$

The gradient mapping is a generalization of the gradient to constrained sets: when $\mathcal{C} \equiv \mathbb{R}^n$, $G_\eta(\mathbf{x}) = \nabla f(\mathbf{x})$. The norm of the gradient mapping can also be used as a measure of optimality: the gradient mapping at a point $\mathbf{x}$ is zero if and only if $\mathbf{x}$ minimizes f over $\mathcal{C}$; more generally, a small gradient mapping norm for a smooth function implies a small optimality gap. See Beck (2017, Chapter 10) and Appendix B for more useful properties.

2.1. Assumptions

We make two key assumptions. The first one, the *strict complementarity assumption* (Assumption 2.4) is key to proving the convergence of the iterates to $\mathcal{F}(\mathbf{x}^*)$, and is a common assumption in the Frank-Wolfe literature (Guélat & Marcotte, 1986; Garber, 2020), which is related to the stability of the solution with respect to noise, and rules out degenerate instances.

Asssumption 2.4 (Strict complementarity)**.** We have that $\langle \nabla f(\mathbf{x}^*), \mathbf{x} - \mathbf{x}^* \rangle = 0$ if and only if $\mathbf{x} \in \mathcal{F}(\mathbf{x}^*)$. Or stated equivalently, there exists a $\tau > 0$ such that $\langle \nabla f(\mathbf{x}^*), \mathbf{x} - \mathbf{x}^* \rangle \geq \tau$ for all $\mathbf{x} \in \mathrm{vert}(\mathcal{X}) \setminus \mathcal{F}(\mathbf{x}^*)$.

Lastly, to achieve local acceleration, similar to Diakonikolas et al. (2020), we require that the optimal solution $\mathbf{x}^*$ is *sufficiently deep* in the relative interior of a face of $\mathcal{X}$. We make the following assumption about the problem, which covers all cases of interest.

Asssumption 2.5 (Location of $\mathbf{x}^*$)**.** The optimum satisfies $\mathbf{x}^* \notin \mathrm{vert}(\mathcal{X})$, or conversely, there exists an $r > 0$ such that $\mathbf{x}^*$ is r-deep in a face $\mathcal{F}$ of $\mathcal{X}$.

Note that the entire polytope $\mathcal{X}$ is an n-dimensional face of itself, and thus Assumption 2.5 allows $\mathbf{x}^* \in \mathrm{rel.int}(\mathcal{X})$. Note that if $\mathbf{x}^* \in \mathrm{vert}(\mathcal{X})$ and the strict complementarity assumption is satisfied (Assumption 2.4), then the projection-free algorithm that will be presented in later sections will reach $\mathbf{x}^*$ in a finite number of iterations (see Garber (2020)), and so there is no need for acceleration. For computational feasibility, we assume that r is bounded away from zero and much larger than the desired accuracy $\epsilon > 0$.

3. Parameter-Free Local Acceleration

This section provides our main result: a parameter-free locally-accelerated CG method (PF-LaCG). Before delving into the technical details, we first describe the core ideas driving our algorithm and its analysis. Due to space constraints, most of the proofs are omitted and are instead provided in the supplementary material.

3.1. Overview of Main Technical Ideas

A standard idea for achieving acceleration in smooth and strongly convex setups where m is unknown is to use a restart-based strategy, which can be described as follows: run an accelerated method for smooth (non-strongly) convex minimization and restart it every time some measure of optimality is reduced by a constant factor. The measures of optimality used in these strategies are either $f(\mathbf{x}) - f(\mathbf{x}^*)$ (Roulet & d'Aspremont, 2020) or $\|G_\eta(\mathbf{x})\|$ (Ito & Fukuda, 2019; Nesterov, 2013).

Neither of these two optimality measures can be applied directly to our setting, as neither can be evaluated: (i) it is rarely the case that $f(\mathbf{x}^*)$ is known, which is needed for evaluating $f(\mathbf{x}) - f(\mathbf{x}^*)$, and we make no such assumption here; and (ii) the gradient mapping norm $\|G_\eta(\mathbf{x})\|$ is a valid optimality measure only for the entire feasible set and requires computing projections onto it; and we do not assume availability of a projection operator onto $\mathcal{X}$.

Even though our algorithm utilizes a restarting-based strategy, the restarts are not used for local acceleration, but for the coupling of a CG method and a projection-based accelerated method. This idea is similar to Diakonikolas et al. (2020); however, there are important technical differences. First, the restarts in Diakonikolas et al. (2020) are scheduled and parameter-based; as such, they cannot be utilized in our parameter-free setting. Our idea is to instead use $w(\mathbf{x}, \mathcal{S})$ as a measure of optimality over the polytope, which is observable naturally in many CG algorithms (see Definition 2.1), where $\mathcal{S}$, dubbed the *active set* of the CG algorithm, is a proper support of $\mathbf{x}$. We perform restarts each time $w(\mathbf{x}, \mathcal{S})$ is halved. The idea of using $w(\mathbf{x}, \mathcal{S})$ as a measure of optimality comes from Kerdreux et al. (2019); however in the aforementioned paper $w(\mathbf{x}, \mathcal{S})$ was not used to obtain a locally accelerated algorithm.

The aforementioned idea of coupling an active-set-based

CG method and an accelerated projection-based method can be summarized as follows. Because the objective function is strongly convex, any convergent algorithm (under any global optimality measure) will be reducing the distance between its iterates and the optimal solution $\mathbf{x}^*$. The role of the CG method is to ensure such convergence without requiring projections onto the feasible set $\mathcal{X}$. When $\mathbf{x}^*$ is contained in the interior of a face of $\mathcal{X}$, it can be argued that after a finite burn-in phase whose length is independent of the target accuracy ϵ, every active set of the utilized CG method will contain $\mathbf{x}^*$ in its convex hull. As it is possible to keep the active sets reasonably small (and their size can never be larger than the current iteration count), projections onto the convex hull of an active set can be performed efficiently, using low-complexity projections onto a probability simplex. Thus, after the burn-in phase, we could switch to a projection-based accelerated algorithm that uses the convex hull of the active set as its feasible set and attains an accelerated convergence rate from then onwards.

There are, of course, several technical challenges related to implementing such a coupling strategy. To begin with, there is no computationally feasible approach we are aware of that could allow detecting the end of the burn-in phase. Thus any reasonable locally accelerated algorithm needs to work without this information, i.e., we do not know when $\mathbf{x}^*$ is contained in the convex hull of the active set. In Diakonikolas et al. (2020), this is achieved by using a parameter-based accelerated algorithm that monotonically decreases the objective value, running this accelerated algorithm and a CG method (AFW or the *Pairwise-step Frank-Wolfe (PFW)*) in parallel, and, on restarts, updating the iterate of the accelerated method and the active set to the ones from the coupled CG method whenever the iterate of CG provides a lower function value. Thus, after the burn-in phase, if the feasible set of the accelerated method does not contain $\mathbf{x}^*$ (or its close approximation), CG eventually constructs a point with the lower function value, after which the accelerated algorithm takes over, leading to local acceleration.

We can neither rely on the scheduled restarts nor the accelerated algorithm used in Diakonikolas et al. (2020), as both are parameter-based. Instead, our monotonic progress is w.r.t. $w(\mathbf{x}, \mathcal{S})$ (i.e., upon restarts we pick a point and the active set with the lower value of $w(\mathbf{x}, \mathcal{S})$) and we rely on a parameter-free accelerated method. As mentioned before, even using a parameter-free projection-based acceleration requires new results, as we need to rely on inexact projections (and inexact gradient mappings). Further, for our argument to work, it is required that after a burn-in phase the accelerated method contracts $w(\mathbf{x}, \mathcal{S})$ at an accelerated rate. Although this may seem like a minor point, we note that it is not true in general, as $w(\mathbf{x}, \mathcal{S})$ can be related to other notions of optimality only when the algorithm iterates are contained in $\mathcal{F}(\mathbf{x}^*)$ and the primal gap is sufficiently small. This is where the strict complementarity assumption (Assumption 2.4) comes into play, as it allows us to show that after a burn-in phase (independent of ϵ), we can upper bound $w(\mathbf{x}, \mathcal{S})$ using $f(\mathbf{x}) - f(\mathbf{x}^*)$ (Theorem 3.4).

3.2. Burn-in Phase

The variant of CG used in our work is the *Away Frank-Wolfe* (AFW) method (Guélat & Marcotte, 1986; Lacoste-Julien & Jaggi, 2015) with a stopping criterion based on halving the Frank-Wolfe gap (Kerdreux et al., 2019), shown in Algorithm 1. For completeness, the useful technical results from Kerdreux et al. (2019) utilized in our analysis are provided in Appendix A.

Algorithm 1 Away-Step Frank-Wolfe Algorithm: $\mathrm{AFW}(\mathbf{x}_0, \mathcal{S}_0)$

$k := 0$
while $w(\mathbf{x}_k, \mathcal{S}_k) > w(\mathbf{x}_0, \mathcal{S}_0)/2$ **do**
  $\mathbf{v}_k := \operatorname{argmin}_{\mathbf{u} \in \mathcal{X}} \langle \nabla f(\mathbf{x}_k), \mathbf{u} \rangle$, $\mathbf{d}_k^{\mathrm{FW}} := \mathbf{v}_k - \mathbf{x}_k$
  $\mathbf{s}_k := \operatorname{argmax}_{\mathbf{u} \in \mathcal{S}_k} \langle \nabla f(\mathbf{x}_k), \mathbf{u} \rangle$, $\mathbf{d}_k^{\mathrm{Away}} := \mathbf{x}_k - \mathbf{s}_k$
  if $-\left\langle \nabla f(\mathbf{x}_k), \mathbf{d}_k^{\mathrm{FW}} \right\rangle \geq -\left\langle \nabla f(\mathbf{x}_k), \mathbf{d}_k^{\mathrm{Away}} \right\rangle$ **then**
    $\mathbf{d}_k := \mathbf{d}_k^{\mathrm{FW}}$ with $\lambda_{\max} := 1$
  else
    $\mathbf{d}_k := \mathbf{d}_k^{\mathrm{Away}}$ with $\lambda_{\max} := \frac{\alpha_k^{\mathbf{s}_k}}{1 - \alpha_k^{\mathbf{s}_k}}$
  end if
  $\mathbf{x}_{k+1} := \mathbf{x}_k + \lambda_k \mathbf{d}_k$ with $\lambda_k \in [0, \lambda_{\max}]$ via line-search
  Update active set $\mathcal{S}_{k+1}$ and $\{\alpha_{k+1}^{\mathbf{v}}\}_{\mathbf{v} \in \mathcal{S}_{k+1}}$
  $k := k + 1$
end while
return $(\mathbf{x}_k, \mathcal{S}_k, w(\mathbf{x}_k, \mathcal{S}_k))$

We now briefly outline how after a finite number of iterations T we can guarantee that $\mathbf{x}_k \in \mathcal{F}(\mathbf{x}^*)$ and $\mathbf{x}^* \in \operatorname{co}(\mathcal{S}_k)$ for $k \geq K_0$. In particular, using Assumption 2.4, if the primal gap is made sufficiently small and the iterate $\mathbf{x}_k$ is not contained in $\mathcal{F}(\mathbf{x}^*)$, then the AFW algorithm will continuously drop those vertices in $\mathcal{S}_k$ that are not in $\mathcal{F}(\mathbf{x}^*)$, until the iterates reach $\mathcal{F}(\mathbf{x}^*)$ (see Garber (2020, Theorem 5), or Theorem A.5 in Appendix A.1, included for completeness).

Theorem 3.1. *If the strict complementarity assumption is satisfied (Assumption 2.4) and the primal gap satisfies* $f(\mathbf{x}_k) - f(\mathbf{x}^*) < 1/2 \min\left\{(\tau/(2D(\sqrt{L/m}+1)))^2/L, \tau, LD^2\right\}$ *then the following holds for the AFW algorithm (Algorithm 5):*

1. *If* $\mathbf{x}_k \notin \mathcal{F}(\mathbf{x}^*)$, *AFW will perform an away step that drops a vertex* $\mathbf{s}_k \in \operatorname{vert}(\mathcal{X}) \setminus \mathcal{F}(\mathbf{x}^*)$.
2. *If* $\mathbf{x}_k \in \mathcal{F}(\mathbf{x}^*)$, *AFW will either perform a Frank-Wolfe step with a vertex* $\mathbf{v}_k \in \operatorname{vert}(\mathcal{F}(\mathbf{x}^*))$ *or an*

away-step with a vertex $\mathbf{s}_k \in \operatorname{vert}(\mathcal{F}(\mathbf{x}^*))$. *Regardless of which step is chosen, the iterate will satisfy:*

$$w(\mathbf{x}_k, \mathcal{S}_k) \leq LD\sqrt{2(f(\mathbf{x}_k) - f(\mathbf{x}^*))/m}.$$

Assuming that $\mathbf{x}_0 \in \operatorname{vert}(\mathcal{X})$ in the AFW algorithm, and using the primal gap convergence gap guarantee in Lacoste-Julien & Jaggi (2015, Theorem 1), we can bound the number of iterations until $f(\mathbf{x}_k) - f(\mathbf{x}^*)$ satisfies the requirement in Theorem 3.1. Using this bound, and the fact that the AFW algorithm can pick up at most one vertex per iteration, we can bound the number of iterations until $\mathbf{x}_k \in \mathcal{F}(\mathbf{x}^*)$. Note that by the second claim in Theorem 3.1, this means that when $\mathbf{x}_k \in \mathcal{F}(\mathbf{x}^*)$, then the iterates will not leave $\mathcal{F}(\mathbf{x}^*)$. Furthermore, once the iterates are inside the optimal face, there are two options: if $\mathbf{x}^* = \mathcal{F}(\mathbf{x}^*)$, then the AFW algorithm will exit once $\mathbf{x}_k \in \mathcal{F}(\mathbf{x}^*)$, as $w(\mathbf{x}_k, \mathcal{S}_k) = 0$, otherwise if $\mathbf{x}^* \notin \operatorname{vert}(\mathcal{X})$ (the case of interest in our setting, by Assumption 2.5), then we need to prove that after a given number of iterations the active set will satisfy $\mathbf{x}^* \in \operatorname{co}(\mathcal{S}_k)$. We prove the former using Fact 3.2 (a variation of Diakonikolas et al. (2020, Fact B.3)).

Fact 3.2 (Critical strong Wolfe gap)**.** There exists a $w_c > 0$ such that for any subset $\mathcal{S} \subseteq \operatorname{vert}(\mathcal{F}(\mathbf{x}^*))$ and point $\mathbf{x} \in \mathcal{F}(\mathbf{x}^*)$ with $\mathbf{x} \in \operatorname{co}(\mathcal{S})$ and $w(\mathbf{x}, \mathcal{S}) \leq w_c$ it follows that $\mathbf{x}^* \in \operatorname{co}(\mathcal{S})$.

Remark 3.3. The critical strong Wolfe gap in Fact 3.2, is a crucial parameter in the coming proofs. However, like the strict complementarity parameter τ (Guélat & Marcotte, 1986; Garber, 2020) and the critical radius r_c defined in Diakonikolas et al. (2020), the critical strong Wolfe gap can be arbitrarily small for some problems. Fortunately, as we will show in the proofs to come, it only affects the length of the burn-in phase of the accelerated algorithm, and moreover this dependence is logarithmic. In Remark A.8 in the Appendix we show a simple example for which one can compute w_c exactly, and we give a lower bound and an upper bound for w_c for Problem (P).

With these tools at hand, we have the bound shown in Theorem 3.4 (see Appendix A for the proof).

Theorem 3.4. *Assume that the AFW algorithm (Algorithm 5) is run starting with* $\mathbf{x}_0 \in \operatorname{vert}(\mathcal{X})$. *If the strict complementarity assumption (Assumption 2.4) is satisfied and* $\mathbf{x}^* \notin \operatorname{vert}(\mathcal{X})$, *then for* $k \geq K_0$ *with*

$$K_0 = \frac{32L}{m \ln 2}\left(\frac{D}{\delta(\mathcal{X})}\right)^2 \cdot \log\left(\frac{2w(\mathbf{x}_0, \mathcal{S}_0)}{\min\{\frac{1}{L}\left(\frac{\tau}{(2D(\sqrt{L/m}+1))}\right)^2, \tau, LD^2, 2w_c\}}\right),$$

where $\delta(\mathcal{X})$ *is the pyramidal width from Definition 2.2 and* $w_c > 0$ *is the critical strong Wolfe gap from Fact 3.2, we have that* $\mathbf{x}_k \in \mathcal{F}(\mathbf{x}^*)$, $\mathbf{x}^* \in \operatorname{co}(\mathcal{S}_k)$. *Moreover:*

$$w(\mathbf{x}_k, \mathcal{S}_k) \leq LD\sqrt{2(f(\mathbf{x}_k) - f(\mathbf{x}^*))/m}.$$

3.3. Parameter-free Projection-based Acceleration

The main idea for obtaining parameter-free projection-based acceleration is to use the regularization trick of Nesterov (Nesterov, 2012) to obtain a near-optimal method for minimizing $\|G_\eta(\mathbf{x})\|$ for a smooth convex function. Then a near-optimal method for smooth strongly convex minimization is obtained by restarting this method every time $\|G_\eta(\mathbf{x})\|$ is halved.

The restarting approach is important here, as it removes the requirement of knowing the parameter m. However, there are a few technical challenges in implementing the near-optimal method for minimizing $\|G_\eta(\mathbf{x})\|$ without the knowledge of the parameter L or the distance to $\mathbf{x}^*$, which is needed for setting the value of the regularization parameter. Some of these challenges were addressed in Ito & Fukuda (2019); Nesterov (2013). However, as discussed before, the approaches from Ito & Fukuda (2019) and Nesterov (2013) are insufficient for our purposes, as they assume access to exact projections onto the feasible set (and thus, exact evaluations of the gradient mapping).

In the following, we first present a near-optimal method for minimizing $\|G_\eta(\mathbf{x})\|$ for a smooth convex function that does not require knowledge of L and works with inexact projections. We then show how to couple this method with adaptive tuning of the regularization parameter and restarts to obtain an overall near-optimal and parameter-free method for smooth strongly convex minimization. This subsection can be read independently from the rest of the paper.

3.3.1. Small Gradient Mapping of Smooth Functions

Let $\mathcal{C} \subseteq \mathbb{R}^n$ be a closed, convex, nonempty set and assume that $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth on $\mathcal{C}$. The rough idea of the regularization trick is the following: instead of working directly with f, use a method for smooth strongly convex minimization to minimize $f_\sigma(\mathbf{x}) = f(\mathbf{x}) + \frac{\sigma}{2}\|\mathbf{x} - \mathbf{x}_0\|^2$ for some sufficiently small $\sigma > 0$ (for accuracy $\epsilon > 0$, $\sigma = \Theta\left(\frac{\epsilon}{\|\mathbf{x}^*_{\mathcal{C}} - \mathbf{x}_0\|}\right)$ suffices, where $\mathbf{x}^*_{\mathcal{C}} \in \operatorname{argmin}_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x})$). As we select σ ourselves, the method can assume knowledge of the strong convexity parameter σ of $f_\sigma(\mathbf{x})$.

The method that we employ here is a variant of μAGD+ from Cohen et al. (2018), which is similar to Diakonikolas et al. (2020). However, unlike Cohen et al. (2018); Diakonikolas et al. (2020), this method is adapted to work with an unknown smoothness parameter and to provide convergence guarantees on $\|G_\eta(\mathbf{x})\|$. One iteration of the algorithm is provided in AGD-Iter (Algorithm 2). In the

algorithm statement, we use $\overset{\epsilon}{\sim}\operatorname{argmin}$ to indicate that the function that follows the argmin is minimized to additive error ϵ. The main result is summarized in the following lemma, while the complete analysis is deferred to Appendix B.

Algorithm 2 AGD-Iter($\mathbf{y}_{k-1}, \mathbf{v}_{k-1}, \mathbf{z}_{k-1}, A_{k-1}, \eta_k, \sigma, \epsilon_0, \eta_0$)

$\eta_k = \eta_k/2$
repeat
  $\eta_k = 2\eta_k$
  $\theta_k = \sqrt{\frac{\sigma}{2(\eta_k+\sigma)}}$, $a_k = \frac{\theta_k}{1-\theta_k} A_{k-1}$
  $\epsilon_k^\ell = \theta_k \epsilon_0/4$, $\epsilon_k^M = a_k\epsilon_0/4$
  $\mathbf{x}_k = \frac{1}{1+\theta_k}\mathbf{y}_{k-1} + \frac{\theta_k}{1+\theta_k}\mathbf{v}_{k-1}$
  $\mathbf{z}_k = \mathbf{z}_{k-1} - a_k \nabla f_\sigma(\mathbf{x}_k) + \sigma a_k \mathbf{x}_k$
  $\mathbf{v}_k \overset{\epsilon_k^M}{\sim} \operatorname{argmin}_{\mathbf{u}\in\mathcal{C}} M_k(\mathbf{u})$, where $M_k(\mathbf{u}) = -\langle \mathbf{z}_k, \mathbf{u}\rangle + \frac{\sigma A_k + \eta_0}{2}\|\mathbf{u}\|^2$
  $\hat{\mathbf{y}}_k = (1-\theta_k)\mathbf{y}_{k-1} + \theta_k \mathbf{v}_k$
  $\mathbf{y}_k \overset{\epsilon_k^\ell}{\sim} \operatorname{argmin}_{\mathbf{u}\in\mathcal{C}} \ell_k(\mathbf{u})$, where $\ell_k(\mathbf{u}) = \langle \nabla f_\sigma(\hat{\mathbf{y}}_k), \mathbf{u} - \hat{\mathbf{y}}_k\rangle + \frac{\eta_k+\sigma}{2}\|\mathbf{u} - \hat{\mathbf{y}}_k\|^2$
until $f(\hat{\mathbf{y}}_k) \le f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \hat{\mathbf{y}}_k - \mathbf{x}_k\rangle + \frac{\eta_k}{2}\|\hat{\mathbf{y}}_k - \mathbf{x}_k\|^2$ and $f(\mathbf{y}_k) \le f(\hat{\mathbf{y}}_k) + \langle \nabla f(\hat{\mathbf{y}}_k), \mathbf{y}_k - \hat{\mathbf{y}}_k\rangle + \frac{\eta_k}{2}\|\mathbf{y}_k - \hat{\mathbf{y}}_k\|^2$
$A_k = A_{k-1} + a_k$, $\tilde{G}^\sigma_{\eta_k+\sigma}(\hat{\mathbf{y}}_k) = (\eta_k+\sigma)(\hat{\mathbf{y}}_k - \mathbf{y}_k)$
return $\eta_k, A_k, \mathbf{z}_k, \mathbf{v}_k, \hat{\mathbf{y}}_k, \mathbf{y}_k, \tilde{G}^\sigma_{\eta_k+\sigma}(\hat{\mathbf{y}}_k)$

Lemma 3.5. *Let $\mathcal{C} \subseteq \mathbb{R}^n$ be a closed convex set and let $f : \mathbb{R}^n \to \mathbb{R}$ be an L-smooth function on $\mathcal{C}$. Let $\mathbf{x}_0 \in \mathcal{C}$ be an arbitrary initial point, and, given $\sigma > 0$, define $f_\sigma(\mathbf{x}) = f(\mathbf{x}) + \frac{\sigma}{2}\|\mathbf{x} - \mathbf{x}_0\|^2$, $\mathbf{x}^*_\sigma = \operatorname{argmin}_{\mathbf{x}\in\mathcal{C}} f_\sigma(\mathbf{x})$. Let $\mathbf{z}_0 = (\eta_0+\sigma)\mathbf{x}_0 - \nabla f(\mathbf{x}_0)$, $\mathbf{y}_0 = \mathbf{v}_0 = \hat{\mathbf{y}}_0 \overset{\epsilon_0^M}{\sim} \operatorname{argmin}_{\mathbf{u}\in\mathcal{C}} M_0(\mathbf{u})$, where $\epsilon_0^M > 0$, $M_0(\mathbf{u})$ is defined as in Algorithm 2, and the estimate η_0 is doubled until $f(\mathbf{y}_0) \le f(\mathbf{x}_0) + \langle \nabla f(\mathbf{x}_0), \mathbf{y}_0 - \mathbf{x}_0\rangle + \frac{\eta_0}{2}\|\mathbf{y}_0 - \mathbf{x}_0\|^2$, same as in Algorithm 2. Given $\eta_0 > 0$, sequence $\{a_k\}_{k\ge 0}$, $A_k = \sum_{i=0}^k a_i$, $\theta_k = \frac{a_k}{A_k}$, and the sequences of errors $\{\epsilon_k^M\}_{k\ge0}$, $\{\epsilon_k^\ell\}_{k\ge0}$, let the sequences of points $\{\hat{\mathbf{y}}_k, \mathbf{y}_k\}_{k\ge0}$ evolve according to Algorithm 2 for $k \ge 1$.*

If $a_0 = A_0 = 1$ and $\theta_k = \frac{a_k}{A_k} \le \sqrt{\frac{\sigma}{2(\eta_k+\sigma)}}$ for $k \ge 1$, then for all $k \ge 1$

$$\frac{\|G^\sigma_{\eta_k+\sigma}(\hat{\mathbf{y}}_k)\|^2}{\eta_k+\sigma} \le \frac{\eta_0}{A_k}\|\mathbf{x}^*_\sigma - \mathbf{x}_0\|^2 + \frac{2}{A_k}\sum_{i=0}^k (2\epsilon_i^M + \epsilon_i^\ell),$$

where $G^\sigma_{\eta_k+\sigma}(\hat{\mathbf{y}}_k)$ is the gradient mapping w.r.t. f_σ, at $\hat{\mathbf{y}}_k$.

In particular, if $a_0 = A_0 = 1$, $\theta_k = \frac{a_k}{A_k} = \sqrt{\frac{\sigma}{2(\eta_k+\sigma)}}$ for $k \ge 1$, $\epsilon_k^M = \frac{a_k\epsilon^2}{8}$, and $\epsilon_k^\ell \le \frac{a_k}{A_k}\frac{\epsilon^2}{8}$, then $\frac{1}{\eta_k+\sigma}\|G^\sigma_{\eta_k}(\hat{\mathbf{y}}_k)\|^2 \le \epsilon^2$ after at most

$$k = O\left(\sqrt{\frac{L}{\sigma}}\log\left(\frac{L\|\mathbf{x}^*_\sigma - \mathbf{x}_0\|}{\epsilon}\right)\right)$$

iterations. Further, the total number of first-order queries to f and oracle queries to inexact projections is at most

$$k' = k + 2\log\left(\frac{2L}{\eta_0}\right).$$

3.3.2. Adaptive Regularization and Restarts

We now provide the full details for the updates of the accelerated sequence. There are two main questions that need to be addressed: (1) how to adaptively adjust the regularization parameter σ and (2) how to perform adaptive restarts based on inexact evaluations of the gradient mapping. For the former, note that, to obtain a near-optimal algorithm, σ should not be allowed to decrease too much (and in fact, needs to satisfy $\sigma = \Omega(m)$; see the proof of Theorem 3.6). For the latter, we can rely on the strong convexity of the function $\ell_k(\mathbf{u}) = \langle \nabla f_\sigma(\hat{\mathbf{y}}_k), \mathbf{u} - \mathbf{y}_k\rangle + \frac{\eta_k+\sigma}{2}\|\mathbf{u} - \hat{\mathbf{y}}_k\|^2$ to relate the exact and the inexact gradient mapping at $\hat{\mathbf{y}}_k$.

We state the main convergence bound of the ACC algorithm (Algorithm 3) here, while the detailed description of the algorithm and its analysis are deferred to Appendix B, for space considerations. Note that the algorithm name stems from *ACCeleration*.

Algorithm 3 ACC($\mathbf{x}_0, \eta_0, \sigma$)

$\sigma = 2\sigma$
repeat
  $\sigma = \sigma/2$
  Run a minimization procedure for $\ell_0(\mathbf{u}) = \langle \nabla f(\mathbf{x}_0), \mathbf{u} - \mathbf{x}_0\rangle + \frac{\eta_0+\sigma}{2}\|\mathbf{u} - \mathbf{x}_0\|^2$. Halt when the current iterate $\mathbf{y}$ of the procedure satisfies $\ell_0(\mathbf{y}) - \min_{\mathbf{u}\in\mathcal{C}} \ell_0(\mathbf{u}) \le \epsilon_0$, where $\epsilon_0 = \frac{\eta_0+\sigma}{32}\|\mathbf{y}_0 - \mathbf{x}_0\|^2 = \frac{1}{32(\eta_0+\sigma)}\|\tilde{G}_{\eta_0+\sigma}(\mathbf{x}_0)\|^2$.
  Set $\hat{\mathbf{y}}_0 = \mathbf{v}_0 = \mathbf{y}_0$; $\mathbf{z}_0 = (\eta_0+\sigma)\mathbf{x}_0 - \nabla f(\mathbf{x}_0)$
  $a_0 = A_0 = 1$
  repeat
    $k = k + 1$
    $\eta_k, A_k, \mathbf{z}_k, \mathbf{v}_k, \hat{\mathbf{y}}_k, \mathbf{y}_k, \tilde{G}^\sigma_{\eta_k+\sigma}(\hat{\mathbf{y}}_k) =$ AGD-Iter($\mathbf{y}_{k-1}, \mathbf{v}_{k-1}, \mathbf{z}_{k-1}, A_{k-1}, \eta_{k-1}, \sigma, \epsilon_0, \eta_0$)
  until $\frac{1}{\eta_k+\sigma}\|\tilde{G}^\sigma_{\eta_k+\sigma}(\hat{\mathbf{y}}_k)\|^2 \le \frac{9\epsilon_0}{4}$
until $\frac{\sigma}{\sqrt{\eta_k+\sigma}}\|\hat{\mathbf{y}}_k - \mathbf{x}_0\| \le \sqrt{\epsilon_0}$
return $\hat{\mathbf{y}}_k, \eta_k\sigma$

Theorem 3.6. *Let $\mathcal{C} \subseteq \mathbb{R}^n$ be a closed convex set, and let $f : \mathbb{R}^n \to \mathbb{R}$ be a function that is L-smooth and m-strongly convex on $\mathcal{C}$. Let $\mathbf{x}_0 \in \mathcal{C}$ be an arbitrary initial point, and let σ_0, η_0 be such that $m \le \sigma_0 \le \eta_0 \le L$. Let $\mathbf{x}_0^{\text{out}} = \mathbf{x}_0$, and for $k \ge 0$, consider the following updates:*

$$\mathbf{x}^{\text{out}}_{k+1}, \eta_{k+1}, \sigma_{k+1} = \text{ACC}(\mathbf{x}^{\text{out}}_k, \eta_k, \sigma_k),$$

where ACC is provided in Algorithm 3. Let $G_{\eta+\sigma}$ denote the gradient mapping of f on $\mathcal{C}$ with parameter

$\eta + \sigma$. Then, for any $\epsilon > 0$, $\|G_{\eta_k+\sigma_k}(\mathbf{x}_k^{\text{out}})\| \leq \epsilon$ for $k = O\left(\log\left(\frac{L\|G_{\eta_0+\sigma_0}(\mathbf{x}_0)\|}{m\epsilon}\right)\right)$. The algorithm for computing $\mathbf{x}_k^{\text{out}}$ utilizes a total number of

$$K = \left(\sqrt{\frac{L}{m}}\log\left(\frac{L}{m}\right)\log\left(\frac{L}{m}\frac{\|G_{\eta_0+\sigma_0}(\mathbf{x}_0)\|}{\epsilon}\right)\right)$$

queries to the FOO for f and an inexact and efficiently computable projection oracle for $\mathcal{C}$, without knowledge of any of the problem parameters.

Observe that, up to the $\log(L/m)$ factor, the bound in Theorem 3.6 is optimal for the class of smooth strongly convex functions, under the local oracle model (which includes the FOO model as a special case).

3.4. Coupling of Methods and Local Acceleration

We now show how to couple the AFW algorithm (Algorithm 5 in Appendix A) with the algorithm described in the previous subsection to achieve local acceleration. The key observation is that after a burn-in phase whose length is independent of ϵ, every active set that AFW constructs contains $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x})$. Thus, minimizing $f(\mathbf{x})$ over $\mathcal{X}$ becomes equivalent to minimizing $f(\mathbf{x})$ over the convex hull of the active set of AFW. Of course, the algorithm needs to adapt to this setting without knowledge of when the burn-in phase has ended. The pseudocode for the resulting algorithm Parameter-Free Locally Accelerated Conditional Gradients (PF-LaCG) is provided in Algorithm 4. The coupling between AFW and ACC in PF-LaCG is illustrated on a simple example in Fig. 1.

For simplicity, Algorithm 4 is stated with the accelerated sequence started with $\mathbf{x}_0 \in \operatorname{vert}(\mathcal{X})$ and with the active set $\mathcal{S}_0 = \{\mathbf{x}_0\}$. As $\mathcal{S}_0$ contains only one vertex, there is no need to run the accelerated algorithm until a later restart (triggered by the condition in Line 6) where the active set contains more than one vertex occurs. Additionally, for the bound on the number of oracle queries in Theorem 3.7 to hold, we need the iterations of ACC (Algorithm 6) and AFW (Algorithm 5) to be aligned, in the sense that one iteration of ACC (Algorithm 2) occurs in parallel to one iteration of AFW. This is only needed for the theoretical bound on oracle complexity. In practice, the two algorithms can be run in parallel without aligning the iterations, so that the overall execution time (modulo coordination at restarts) is never higher than for running AFW alone.

We are now ready to state our main result. The complete proof is deferred to Appendix C.

Theorem 3.7. *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a closed convex polytope of diameter D, and let $f : \mathbb{R}^n \to \mathbb{R}$ be a function that is L-smooth and m-strongly convex on $\mathcal{X}$. Let Assumptions 2.2 and 2.4 be satisfied for $f, \mathcal{X}$. Denote $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$. Given $\epsilon > 0$, let $\mathbf{x}^{\text{out}}$ be the output point of PF-LaCG (Algorithm 4), initialized at an arbitrary vertex $\mathbf{x}_0$ of $\mathcal{X}$. Then $w(\mathbf{x}^{\text{out}}, \mathcal{S}^{\text{out}}) \leq \epsilon$ and PF-LaCG uses a total of at most*

$$K = O\Bigg(\min\Bigg\{\log\left(\frac{w(\mathbf{x}_0,\mathcal{S}_0)}{LD^2}\right) + \frac{LD^2}{m\delta^2}\log\left(\frac{w(\mathbf{x}_0,\mathcal{S}_0)}{\epsilon}\right),$$
$$K_0 + K_1 + \sqrt{\frac{L}{m}}\log\left(\frac{L}{m}\right)\log\left(\frac{LD}{m\delta}\right)\log\left(\frac{LD}{\epsilon}\right)\Bigg\}\Bigg)$$

queries to the FOO for f and the LMO for $\mathcal{X}$, where

$$K_0 = \frac{32L}{m\ln 2}\left(\frac{D}{\delta(\mathcal{X})}\right)^2 \cdot \log\left(\frac{2w(\mathbf{x}_0,\mathcal{S}_0)}{\min\{\frac{1}{L}\left(\frac{\tau}{(2D(\sqrt{L/m}+1))}\right)^2, \tau, LD^2, 2w_c\}}\right),$$

and $K_1 = \frac{128LD^2}{m\delta^2}$.

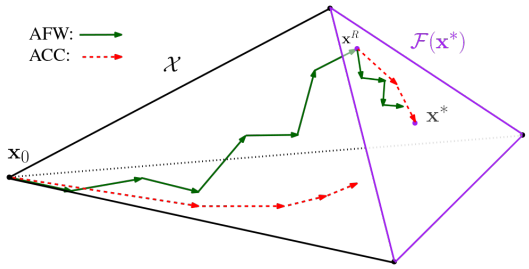

Figure 1. An example of coupling between AFW and ACC in PF-LaCG on a tetrahedron as the feasible set, starting from initial point $\mathbf{x}_0$ with the base of the tetrahedron as its support $\mathcal{S}_0$. The two algorithms are run in parallel from $\mathbf{x}_0$: AFW optimizes over the entire tetrahedron, allowing it to add and remove vertices, while ACC optimizes over the base of the tetrahedron only and it cannot converge to the optimal point $\mathbf{x}^*$, as $\mathbf{x}^* \notin \operatorname{co}(\mathcal{S}_0)$. After several iterations, once the restart criterion for AFW is triggered, PF-LaCG chooses the output point of AFW over that of ACC, as $w^{\text{AFW}} \leq \min\{w^{\text{ACC}}, w_{\text{prev}}^{\text{ACC}}/2\}$, hence a PF-LaCG restart occurs at $\mathbf{x}^R$. For ease of exposition we assume that the point outputted by AFW is contained in $\mathcal{F}(\mathbf{x}^*)$ after a single halving of $w(\mathbf{x}, \mathcal{S})$, although in practice several restarts may be needed for AFW to reach $\mathcal{F}(\mathbf{x}^*)$. Since $\mathbf{x}^R$ is on the optimal face $\mathcal{F}(\mathbf{x}^*)$, PF-LaCG has completed the burn-in phrase. The two algorithms again run in parallel from $\mathbf{x}^R$ after the restart. However, ACC converges to the optimal $\mathbf{x}^*$ at an accelerated rate, much faster than AFW. Hence, local acceleration is achieved by PF-LaCG while being at least as fast as vanilla AFW.

Strong Wolfe Gap as an Upper Bound For Primal Gap. Note that while in Theorem 3.7 we show a convergence rate for $w(\mathbf{x}, \mathcal{S})$, this translates to the same convergence rate in primal gap, as the aforementioned quantity is an upper bound on the strong Wolfe gap and hence an upper bound on the primal gap (Kerdreux et al., 2019). Furthermore, we

Algorithm 4 PF-LaCG($\mathbf{x}_0 \in \text{vert}(\mathcal{X}), \epsilon > 0$)

$\mathcal{S}^{\text{out}} = \mathcal{S}^{\text{AFW}} = \mathcal{S}^{\text{ACC}} = \{\mathbf{x}_0\}$
$\mathbf{x}^{\text{out}} = \mathbf{x}^{\text{AFW}} = \mathbf{x}^{\text{ACC}} = \mathbf{x}_0$
$w^{\text{out}} = w^{\text{AFW}}_{\text{prev}} = w(\mathbf{x}_0, \mathcal{S}_0)$
while $w^{\text{out}} > \epsilon$ **do**
  Run AFW and restarted ACC (Theorem 3.6) in parallel, and let $(\mathbf{x}^{\text{AFW}}, \mathcal{S}^{\text{AFW}})$ and $(\mathbf{x}^{\text{ACC}}, \mathcal{S}^{\text{ACC}})$ denote their respective most recent output points and active sets; note that ACC is run on $\mathcal{C} = \text{co}(\mathcal{S}^{\text{ACC}})$
  if $w^{\text{AFW}} = w(\mathbf{x}^{\text{AFW}}, \mathcal{S}^{\text{AFW}}) \leq \frac{1}{2} w^{\text{AFW}}_{\text{prev}}$ **then**
    $w^{\text{AFW}}_{\text{prev}} = w^{\text{AFW}}$, $w^{\text{ACC}}_{\text{prev}} = w^{\text{ACC}}$,
    Compute $w^{\text{ACC}} = w(\mathbf{x}^{\text{ACC}}, \mathcal{S}^{\text{ACC}})$
    if $w^{\text{AFW}} \leq \min\{w^{\text{ACC}}, w^{\text{ACC}}_{\text{prev}}/2\}$ **then**
      $\mathbf{x}^{\text{ACC}} = \mathbf{x}^{\text{AFW}}$, $\mathcal{S}^{\text{ACC}} = \mathcal{S}^{\text{AFW}}$
      $\mathbf{x}^{\text{out}} = \mathbf{x}^{\text{AFW}}$, $\mathcal{S}^{\text{out}} = \mathcal{S}^{\text{AFW}}$
      $w^{\text{out}} = w^{\text{AFW}}$
    else
      if $|\mathcal{S}^{\text{ACC}}| \leq |\mathcal{S}^{\text{AFW}}|$ **then**
        $\mathbf{x}^{\text{AFW}} = \mathbf{x}^{\text{ACC}}$, $\mathcal{S}^{\text{AFW}} = \mathcal{S}^{\text{ACC}}$
        $w^{\text{AFW}} = w^{\text{ACC}}$
      end if
      $\mathbf{x}^{\text{out}} = \mathbf{x}^{\text{ACC}}$, $\mathcal{S}^{\text{out}} = \mathcal{S}^{\text{ACC}}$, $w^{\text{out}} = w^{\text{ACC}}$
    end if
  end if
end while
return $\mathbf{x}^{\text{out}}$

remark that $w(\mathbf{x}, \mathcal{S}) = 0$ if and only if $\mathbf{x} = \mathbf{x}^*$ when $\mathcal{S} \subseteq \text{vert}(\mathcal{X})$ is a proper support of $\mathbf{x}$ with respect to the polytope $\mathcal{X}$. In the next section we illustrate our computational results with respect to the primal gap $f(\mathbf{x}) - f(\mathbf{x}^*)$, as this quantity is more common in the optimization literature.

4. Computational Experiments

We numerically demonstrate that PF-LaCG (Algorithm 4) when implemented in Python 3 outperforms other parameter-free CG methods in both iteration count and in wall-clock time. Most notably, we compare against the AFW, PFW, *Decomposition-Invariant CG* (DICG) (in the case of the probability simplex, as this algorithm only applies to $0-1$ polytopes (Garber & Meshi, 2016)) and the *Lazy AFW* (AFW (Lazy)) (Braun et al., 2017) algorithms. As predicted by our theoretical results, the improvement is observed locally, once the iterates of the algorithm reach the optimal face. Further, we empirically observe on the considered examples that the active sets do not become too large, which is important for the accelerated algorithm to have an edge over standard CG updates in the overall wall-clock time. Our code can be found at `https://github.com/ericlincc/Parameter-free-LaCG`.

Similar to Diakonikolas et al. (2020), we solve the minimization subproblems (projections onto the convex hull of the active sets) within ACC (Algorithm 6) using Nesterov's accelerated gradient descent (Nesterov, 2018) with $\mathcal{O}(n \log n)$ projections onto the simplex described in Duchi et al. (2008, Algorithm 1). Even though in our analysis we consider coupling accelerated sequence with AFW, we note that PF-LaCG can be coupled with any CG variant that maintains an active set, such as standard AFW or PFW.

Since the execution of local acceleration within PF-LaCG is completely independent of the execution of the coupled CG variant, it is possible to run the locally accelerated algorithm in parallel on a separate core within one machine or even on a secondary machine, allowing us to utilize more computational power with the goal of solving large-scale problems to high accuracy in less computing time. In our experiments, we implemented the former approach where we run each process using one CPU core, and we run the accelerated algorithm of PF-LaCG on a separate process. This approach enables us to guarantee that PF-LaCG is not slower than the CG variant such as AFW and PFW barring negligible process creation overhead and inter-process communication while achieving much faster convergence once we have reached the optimal face.

Probability Simplex. The unit probability simplex, although a toy example, can give us insight into the behaviour of PF-LaCG. We know that after a restart in which the AFW active set satisfies $\mathbf{x}^* \in \text{co}(\mathcal{S}_k)$, we should expect to see accelerated convergence from the iterates computed by the ACC algorithm. Given the structure of the probability simplex, this is easy to check in the experiments once we have a high-accuracy solution to the minimization problem. The function being minimized in this example is $f(\mathbf{x}) = \mathbf{x}^T \left(M^T M + \alpha \mathbf{1}_n\right) \mathbf{x}/2 + \mathbf{b}^T \mathbf{x}$, where $M \in \mathbb{R}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}^n$ have entries sampled uniformly at random between 0 and 1 and $n = 10000$. The parameter $\alpha = 500$ is set so that the objective function satisfies $m \approx 500$. The resulting condition number is $L/m = 50000$, and the number of nonzero elements in $\mathbf{x}^*$ is around 320. The AFW algorithm in PF-LaCG (AFW) satisfies that $\mathbf{x}^* \in \text{co}(\mathcal{S}_k)$ around iteration 400, consequently we achieve the accelerated convergence rate from then onwards. The same can be said regarding PF-LaCG (PFW) around iteration 350.

Structured LASSO Regression. The LASSO is an extremely popular sparse regression analysis method in statistics, and has become an important tool in ML. In many applications, we can impose additional structure on the solution being learnt through the addition of linear constraints; this is useful, for example, when learning sparse physics dynamics from data (see Carderera et al. (2021)), in which we can impose additional linear equality constraints on the

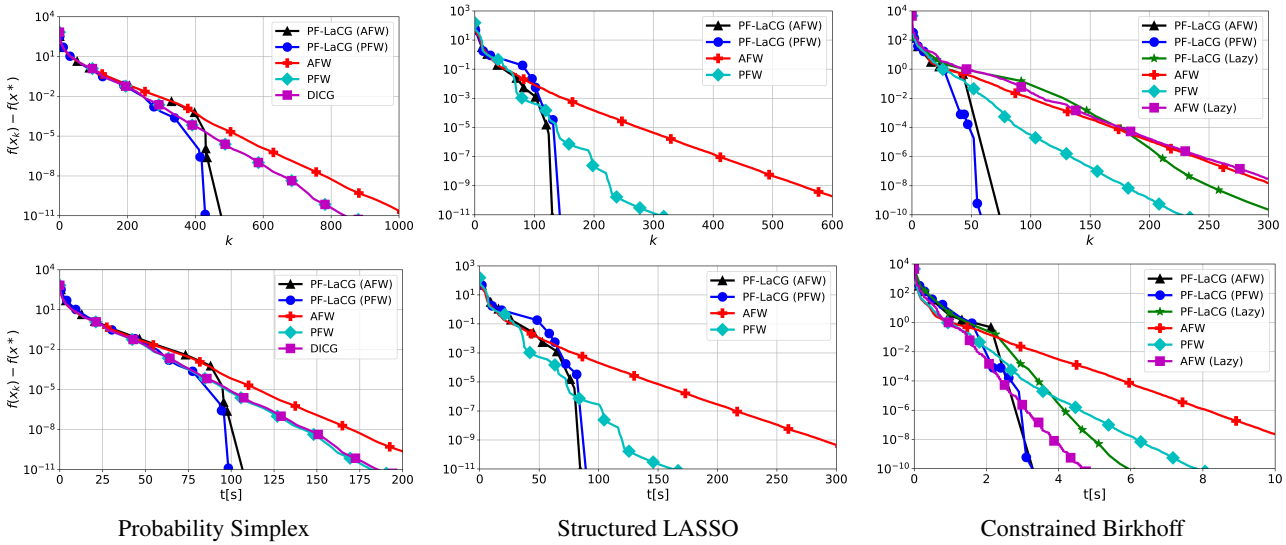

Figure 2. **Numerical performance of PF-LaCG**: Top row depicts primal gap convergence in terms of iteration count, while bottom row depicts primal gap convergence in terms of time. Left-most column shows the results over the probability simplex, center column shows the results for the structured Lasso problem, and right-most column shows the results for the constrained Birkhoff polytope.

problem to reflect the symmetries present in the physical problem. This allows us to learn dynamics that are consistent with the underlying physics, and which potentially generalize better when predicting on unseen data. The objective function is the same as in the last section, except we now have $n = 1000$, $\alpha = 100$, and we choose the elements of $\mathbf{b}$ uniformly from 0 to 100. This results in a condition number of $L/m = 250000$. The feasible region is the intersection of the ℓ_1 unit ball and a series of equality constraints. To generate the additional equality constraints, we sample 125 pairs of distinct integers (i, j) from $1 \leq i, j \leq n$ without replacement, and we set $x_i = x_j$ for each pair, adding 125 linear constraints.

Constrained Birkhoff Polytope. We also solve a matching with the same quadratic function as in the previous section with $\alpha = 1$, and where we have scaled the matrix $M^T M$ to have a maximum eigenvalue of 100000. This results in an objective function that has a condition number of $L/m = 100000$. The matching problem is solved over the Birkhoff polytope with $n = 400$ where we have imposed additional linear constraints. We sample 80 integers i from $1 \leq i \leq n$ without replacement, and we set $x_i = 0$ for the first 40 integers (to represent that certain matchings are not possible), and $x_i \leq 0.5$ for the remaining 40 integers to represent a maximum fractional matching. As in the LaCG algorithm (Diakonikolas et al., 2020), our approach is compatible with the lazification technique for AFW (Braun et al., 2017). In this last example we couple our algorithm with the AFW (Lazy) algorithm, resulting in the PF-LaCG (Lazy) algorithm. Moreover, we also benchmark against the AFW (Lazy) algorithm for reference.

5. Discussion

We have introduced a novel projection-free PF-LaCG algorithm for minimizing smooth and strongly convex functions over polytopes. This algorithm is parameter-free and locally accelerated. In particular, we have shown that after a finite burn-in phase (independent of the target error ϵ) PF-LaCG achieves a near-optimal accelerated convergence rate without knowledge of any of the problem parameters (such as the smoothness or strong convexity of the function). As mentioned in the introduction, global acceleration is generally not possible for CG-type methods. We have also demonstrated the improved locally accelerated convergence rate of PF-LaCG using numerical experiments. Some interesting questions for future research remain. For example, it is an interesting and practically relevant question whether local acceleration is possible for CG methods that are not active set-based, such as, e.g., DICG (Garber & Meshi, 2016), which would possibly lead to even faster algorithms.

Acknowledgments

This research was partially funded by NSF grant CCF-2007757, by the Office of the Vice Chancellor for Research and Graduate Education at the University of Wisconsin–Madison with funding from the Wisconsin Alumni Research Foundation, and by the Deutsche Forschungsgemeinschaft (DFG) through the DFG Cluster of Excellence MATH+ and the Research Campus Modal funded by the German Federal Ministry of Education and Research (fund numbers 05M14ZAM, 05M20ZBM).

References

Bashiri, M. A. and Zhang, X. Decomposition-invariant conditional gradient for general polytopes with line search. In *NIPS*, pp. 2690–2700, 2017.

Beck, A. *First-order methods in optimization*. MOS-SIAM, 2017.

Beck, A. and Shtern, S. Linearly convergent away-step conditional gradient for non-strongly convex functions. *Mathematical Programming*, 164(1-2):1–27, 2017.

Braun, G., Pokutta, S., and Zink, D. Lazifying conditional gradient algorithms. In *Proc. ICML'2017*, 2017.

Braun, G., Pokutta, S., Tu, D., and Wright, S. Blended conditional gradients: The unconditioning of conditional gradients. In *Proc. ICML'19*, 2019.

Carderera, A. and Pokutta, S. Second-order conditional gradient sliding. *arXiv preprint arXiv:2002.08907*, 2020.

Carderera, A., Pokutta, S., Schütte, C., and Weiser, M. CINDy: Conditional gradient-based identification of non-linear dynamics–noise-robust recovery. *arXiv preprint arXiv:2101.02630*, 2021.

Chen, S. S., Donoho, D. L., and Saunders, M. A. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

Cohen, M. B., Diakonikolas, J., and Orecchia, L. On acceleration with noise-corrupted gradients. In *Proc. ICML'18*, 2018.

Combettes, C. W. and Pokutta, S. Complexity of linear minimization and projection on some sets. *arXiv preprint arXiv:2101.10040*, 2021.

Combettes, C. W., Spiegel, C., and Pokutta, S. Projection-free adaptive gradients for large-scale optimization. *arXiv preprint arXiv:2009.14114*, 2020.

Condat, L. Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming*, 158(1):575–585, 2016.

Diakonikolas, J. and Orecchia, L. The approximate duality gap technique: A unified theory of first-order methods. *SIAM Journal on Optimization*, 29(1):660–689, 2019.

Diakonikolas, J., Carderera, A., and Pokutta, S. Locally accelerated conditional gradients. In *Proc. AISTATS'20*, 2020.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proc. NIPS'08*, 2008.

Dvurechensky, P., Ostroukhov, P., Safin, K., Shtern, S., and Staudigl, M. Self-concordant analysis of Frank-Wolfe algorithms. In *Proc. ICML'20*, 2020.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2): 95–110, 1956.

Garber, D. Faster projection-free convex optimization over the spectrahedron. In *Proc. NIPS'16*, 2016.

Garber, D. Revisiting Frank-Wolfe for polytopes: Strict complementarity and sparsity. In *Proc. NeurIPS'20*, 2020.

Garber, D. and Meshi, O. Linear-memory and decomposition-invariant linearly convergent conditional gradient algorithm for structured polytopes. In *Proc. NIPS'16*, 2016.

Guélat, J. and Marcotte, P. Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35(1):110–119, 1986.

Gutman, D. H. and Peña, J. F. The condition of a function relative to a polytope. *arXiv preprint arXiv:1802.00271*, 2018.

Hazan, E. and Luo, H. Variance-reduced and projection-free stochastic optimization. In *Proc. ICML'16*, 2016.

Held, M., Wolfe, P., and Crowder, H. P. Validation of subgradient optimization. *Mathematical programming*, 6 (1):62–88, 1974.

Ito, M. and Fukuda, M. Nearly optimal first-order methods for convex optimization under gradient norm measure: An adaptive regularization approach. *arXiv preprint arXiv:1912.12004*, 2019.

Jaggi, M. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proc. ICML'13*, 2013.

Kerdreux, T., d'Aspremont, A., and Pokutta, S. Restarting Frank-Wolfe: Faster rates under Hölderian error bounds. In *Proc. AISTATS'19*, 2019.

Kerdreux, T., d'Aspremont, A., and Pokutta, S. Projection-free optimization on uniformly convex sets. In *Proc. AISTATS'21*, 2021.

Lacoste-Julien, S. and Jaggi, M. On the global linear convergence of Frank-Wolfe optimization variants. In *Proc. NIPS'15*, 2015.

Lam, S. K., Pitrou, A., and Seibert, S. Numba: A LLVM-based Python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pp. 1–6, 2015.

Lan, G. The complexity of large-scale convex programming under a linear optimization oracle. *arXiv preprint arXiv:1309.5550*, 2013.

Lei, Q., Zhuo, J., Caramanis, C., Dhillon, I. S., and Dimakis, A. G. Primal-dual block generalized Frank-Wolfe. *Proc. NeurIPS'19*, 2019.

Levitin, E. S. and Polyak, B. T. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6(5):1–50, 1966.

Négiar, G., Dresdner, G., Tsai, A., El Ghaoui, L., Locatello, F., Freund, R., and Pedregosa, F. Stochastic Frank-Wolfe for constrained finite-sum minimization. In *Proc. ICML'20*, 2020.

Nesterov, Y. How to make the gradients small. *Optima. Mathematical Optimization Society Newsletter*, (88):10–11, 2012.

Nesterov, Y. Gradient methods for minimizing composite functions. *Mathematical Programming*, (140(1)):125–161, 2013.

Nesterov, Y. *Lectures on Convex Optimization*. Springer, 2018.

Peña, J. and Rodríguez, D. Polytope conditioning and linear convergence of the Frank–Wolfe algorithm. *Mathematics of Operations Research*, 44(1):1–18, 2019.

Pedregosa, F., Negiar, G., Askari, A., and Jaggi, M. Linearly convergent Frank–Wolfe with backtracking line-search. In *Proc. AISTATS'20*, 2020.

Roulet, V. and d'Aspremont, A. Sharpness, restart, and acceleration. *SIAM Journal on Optimization*, 30(1):262–289, 2020.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Tsiligkaridis, T. and Roberts, J. On Frank-Wolfe optimization for adversarial robustness and interpretability. *arXiv preprint arXiv:2012.12368*, 2020.

Zhang, M., Shen, Z., Mokhtari, A., Hassani, H., and Karbasi, A. One sample stochastic Frank-Wolfe. In *Proc. AISTATS'20*, 2020.

Zhou, S., Gupta, S., and Udell, M. Limited memory Kelley's method converges for composite convex and submodular objectives. *arXiv preprint arXiv:1807.07531*, 2018.