Locally Accelerated Conditional Gradients

Jelena Diakonikolas
University of Wisconsin-Madison

Alejandro Carderera
Georgia Institute of Technology

Sebastian Pokutta
Zuse Institute Berlin
Technische Universität Berlin

Abstract

Conditional gradients constitute a class of *projection-free* first-order algorithms for smooth convex optimization. As such, they are frequently used in solving smooth convex optimization problems over polytopes, for which the computational cost of projections is prohibitive. However, they do not enjoy the optimal convergence rates achieved by projection-based accelerated methods; moreover, *achieving such globally-accelerated rates is information-theoretically impossible*. To address this issue, we present *Locally Accelerated Conditional Gradients* – an algorithmic framework that couples accelerated steps with conditional gradient steps to achieve *local* acceleration on smooth strongly convex problems. Our approach does not require projections onto the feasible set, but only on (typically low-dimensional) simplices, thus keeping the computational cost of projections at bay. Further, it achieves *optimal accelerated local convergence*. Our theoretical results are supported by numerical experiments, which demonstrate significant speedups over state of the art methods in both per-iteration progress and wall-clock time.

1 Introduction

Smooth convex optimization problems over polytopes arise in a variety of settings, such as video co-localization (Joulin et al., 2014), structured energy minimization (Swoboda and Kolmogorov, 2019), greedy particle optimization in Bayesian inference (Futami et al., 2019), and structural SVMs (Lacoste-Julien et al., 2013). The methods of choice in such settings are variants of the conditional gradient (CG) or Frank-Wolfe method (Frank and Wolfe, 1956), which eschew the computationally-expensive projections of standard first-order methods while enjoying favorable characteristics of the produced solutions such as sparse representation.

Despite their simplicity and broad applicability, CG methods do not attain the accelerated convergence of projection-based methods such as Nesterov accelerated gradient descent (Nesterov, 1983) and variants thereof (Diakonikolas and Orecchia, 2018; Cohen et al., 2018; Tseng, 2008; Beck and Teboulle, 2009). This limitation is a consequence of the access to the feasible polytope $\mathcal{X}$ being restricted to a linear minimization oracle: in such a setting, global acceleration is information-theoretically impossible (Jaggi, 2013; Lan, 2013) and improving the (global) rate of convergence of CG in various structured settings has been an active area of research (Jaggi, 2013; Garber and Hazan, 2016; Lacoste-Julien and Jaggi, 2015; Garber and Meshi, 2016; Lan et al., 2017; Braun et al., 2017; Kerdreux et al., 2018b,a; Garber et al., 2018; Braun et al., 2019; Guélat and Marcotte, 1986). While exciting progress has been made, the global convergence rate of CG cannot be dimension-independent in the worst case.

In this work, we depart from the path of improving global convergence guarantees of CG and instead ask:

Is local dimension-independent acceleration possible?

To address this question, we focus on smooth strongly convex objective functions and introduce the *Locally Accelerated Conditional Gradients* algorithmic framework that achieves the optimal dimension-independent locally accelerated convergence rate.

1.1 Limits to Global Acceleration

Acceleration for conditional gradient methods has been an important topic of interest. Apart from several technical challenges, there is a strong lower bound (Jaggi, 2013; Lan, 2013) that significantly limits what type of acceleration is achievable. This lower bound applies to

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

arbitrary methods whose access to the feasible region is limited to a linear optimization oracle. As an illustration, let $\mathcal{X} \stackrel{\text{def}}{=} \{\mathbf{x} \in \mathbb{R}^n \mid \sum_{i=1}^n x_i = 1, \mathbf{x} \geq 0\}$ be the probability simplex on n coordinates and consider:

$$\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \equiv \|\mathbf{x}\|_2^2 . \tag{LB}$$

If a first-order method has access to $\mathcal{X}$ only by means of a linear optimization oracle, then each query to the oracle reveals at most a single vertex of $\mathcal{X}$, and it is guaranteed that after $k < n$ iterations the primal gap satisfies $f(\mathbf{x}_k) - f(\mathbf{x}^*) \geq \frac{1}{k} - \frac{1}{n}$. In particular, for $k = n/2$ (assuming w.l.o.g. that n is even), the primal gap is bounded below by $f(\mathbf{x}_{n/2}) - f(\mathbf{x}^*) \geq \frac{1}{n}$, and even the $O(1/k^2)$ rate cannot be achieved in full generality.

Note that the objective in (LB) is also strongly convex. Thus, if we have a generic algorithm that is linearly convergent, contracting the primal gap as $f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq e^{-\rho k}(f(\mathbf{x}_0) - f(\mathbf{x}^*))$ with a *global rate* ρ, then it follows that $\rho \leq 2\frac{\log n}{n}$. The *Fully-Corrective Frank-Wolfe* algorithm converges with (roughly) $\rho = \frac{1}{2n}$ (Lacoste-Julien and Jaggi, 2015), and similar rates apply to other variants. Given the lower bound, it follows that up to logarithmic factors these *global* rates cannot be improved and acceleration with the rate parameter $\rho = \sqrt{1/(2n)}$ is not possible.

At the same time, it is known that, e.g., AFW can be globally accelerated using the *Catalyst* framework (Lin et al., 2015) (see also a discussion in Lacoste-Julien and Jaggi (2015)). However, the obtained accelerated rate involves large dimension-dependent constants to be compatible with the lower bound, making the algorithm impractical. Another form of acceleration in the case of linear optimization based methods is achieved by *Conditional Gradient Sliding* (CGS) (Lan and Zhou, 2016), where the complexity is separated into calls to the first-order and linear optimization oracle. While the optimal first-order oracle complexity of $O(1/\sqrt{\epsilon})$ is achieved in the case of smooth (non-strongly) convex minimization, the linear optimization oracle complexity is $O(1/\epsilon)$, compatible with the lower bound.

In addition to the work on accelerating it, recent work has also generalized CG to stochastic settings (Fang et al., 2018; Hassani et al., 2019; Shen et al., 2019). While considering the stochastic settings is of high practical relevance, it is beyond the scope of this paper.

1.2 Contributions and Related Work

We show that *dimension-independent* acceleration for CG methods *is possible* for smooth strongly convex minimization after a burn-in phase whose length does not depend on the target accuracy ϵ (but could potentially depend on the dimension). This allows for local acceleration, achieving the *asymptotically optimal rate*. Our contributions are summarized as follows.

LaCG. We introduce a new algorithmic framework for the class of conditional gradient methods, which we dub *Locally Accelerated Conditional Gradients (LaCG)*. The framework couples active-set-based linearly convergent CG methods for smooth strongly convex minimization (such as, e.g., AFW or PFW) with the accelerated algorithm μAGD+ (Cohen et al., 2018). The coupling ensures that the updates make at least as much progress as the employed CG-type method, while being able to seamlessly transition to faster locally-accelerated convergence once suitable conditions are met and without ever having to explicitly test such conditions. LaCG achieves an asymptotically optimal iteration complexity of $K + O\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\epsilon}\right)$ to solve $\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$ up to accuracy ϵ, where f is L-smooth and μ-strongly convex and K is a constant that only depends on $\mathcal{X}$ and f.

Slightly simplifying, we achieve acceleration once we identify the optimal face of $\mathcal{X}$ and are reasonably close to the optimal solution. Our reported complexity depends on the smoothness L and strong convexity μ parameters of f *independent of the dimension of* $\mathcal{X}$. This stands in contrast to the previously reported linearly convergent methods, whose problem parameters had to be adjusted to account for the geometry of $\mathcal{X}$, e.g., via the *pyramidal width* in Lacoste-Julien and Jaggi (2015) or *smoothness and strong-convexity relative to the polytope* in Pena and Rodriguez (2018); Gutman and Pena (2019), which both bring in a dimension dependence. Note that such dimension-dependent terms are *unavoidable* if global linear rates of convergence are sought, due to the aforementioned lower bound.

We bypass the limitations of global linear convergence by accelerating the methods *locally*: the faster convergence applies after a constant number iterations with possibly weaker rates. Following the burn-in phase, the method requires no more than $O\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\epsilon}\right)$ queries to either the first-order oracle or the linear minimization oracle. This dimension-independent acceleration following a slower phase is clearly observed in our numerical experiments, particularly in comparison with Catalyst-augmented AFW. As discussed before, Catalyst acceleration is necessarily weakened by a geometric correction leading to the (either first-order or linear optimization) oracle complexity of $O\left(\sqrt{\frac{L}{\mu}} \frac{D^2}{\delta^2} \log \frac{1}{\epsilon}\right)$, where D is the diameter of $\mathcal{X}$ and δ is the pyramidal width of $\mathcal{X}$; see Lacoste-Julien and Jaggi (2015).

To achieve local acceleration, we assume that we can project relatively efficiently onto simplices spanned by sets of vertices of small size. However, while we employ projections onto the convex hulls of maintained active

sets, we stress that the feasible region is only accessed via a linear optimization oracle, i.e., our method is an LO-based method and the active sets are typically small, so that projections onto those sets are cheap. In this sense of employing projections *internally* but not over the entire feasible set, our algorithm is similar to *Conditional Gradient Sliding* (Lan et al., 2017).

Generalized Accelerated Method. While there is an extensive literature on accelerated methods in optimization (Betancourt et al., 2018; Cohen et al., 2018; Diakonikolas and Orecchia, 2018; Tseng, 2008; Beck and Teboulle, 2009; Bubeck et al., 2015; Nesterov, 2018; Drusvyatskiy et al., 2018; Polyak, 1964), none of these approaches directly applies to local acceleration of CG. Most relevant to our work is Cohen et al. (2018), and, in the process of constructing LaCG, we generalize the μAGD+ (Cohen et al., 2018) algorithm in a few important ways (note that μAGD+ is also a generalization of Nesterov's method).

First, we show that μAGD+ retains its convergence guarantees when coupled with an arbitrary alternative algorithm, where the coupling selects the point with the lower function value between the two algorithms, in each iteration. This is crucial for turning μAGD+ into a descent method and ensures that it makes at least as much progress per iteration as the CG-type method with which it is coupled. This coupling also allows us to achieve acceleration without any explicit knowledge of the parameters of the polytope $\mathcal{X}$ or the position of the function minimizer $\mathbf{x}^*$.

Second, we show that μAGD+ allows inexact projections. While this is not surprising, as similar results have been shown for proximal methods (Schmidt et al., 2011), this generalization of μAGD+ is a necessary ingredient to ensure that the per-iteration complexity of LaCG does not become too high.

Finally, we prove that μAGD+ converges to the optimal solution at no computational loss even if the convex set on which the projections are performed changes between iterations, as long as the convex set in iteration k is contained in the convex set from iteration $k-1$ and it contains the minimizer $\mathbf{x}^*$. We are not aware of any other results of this type. Note that this result allows us to update the projection simplex with each update of the active set, and, as vertices are dropped from the active set by away steps in AFW or PFW, the iterations become less expensive.

Computational Experiments. We compare our methods to other CG variants and provide computational evidence that our algorithms achieve a practical speed-up, both in per-iteration progress and in wall-clock time, outperforming state of the art methods.

2 Preliminaries

We consider problems of the form $\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$, where f is a smooth (gradient Lipschitz) strongly convex function and $\mathcal{X}\subseteq\mathbb{R}^n$ is a convex polytope. We assume (i) first-order access to f: given $\mathbf{x}\in\mathcal{X}$, we can compute $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$, and (ii) linear optimization oracle access to $\mathcal{X}$: given a vector $\mathbf{c}\in\mathbb{R}^n$, we can compute $\mathbf{v}=\operatorname{argmin}_{\mathbf{u}\in\mathcal{X}}\langle\mathbf{c},\mathbf{u}\rangle$.

Let $\|\cdot\|$ be the Euclidean norm and 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 $\mathcal{X}\subseteq\mathbb{R}^n$ if $\mathcal{B}(\mathbf{x},2r)\cap\operatorname{aff}(\mathcal{X})\subseteq\mathcal{X}$. The point $\mathbf{x}$ is contained in the *relative interior of* $\mathcal{X}$, written as $\mathbf{x}\in\operatorname{rel.int}(\mathcal{X})$, if there exists an $r>0$ such that $\mathbf{x}$ is r-deep in $\mathcal{X}$; if $\operatorname{aff}(\mathcal{X})=\mathbb{R}^n$, then $\mathbf{x}$ is contained in the *interior of* $\mathcal{X}$, written as $\mathbf{x}\in\operatorname{int}(\mathcal{X})$. Further, given a polytope $\mathcal{X}$, let $\operatorname{vert}(\mathcal{X})\subseteq\mathcal{X}$ denote the (finite) set of vertices of $\mathcal{X}$ and let $\Delta_n\stackrel{\text{def}}{=}\{\mathbf{x}\in\mathbb{R}^n\mid\sum_{i=1}^n x_i=1, x\geq 0\}\subseteq\mathbb{R}^n$ denote the *probability simplex of dimension n*. We denote the convex hull of a set $\mathcal{X}$ as $\operatorname{co}(\mathcal{X})$.

2.1 Conditional Gradient Descent

We provide a very brief introduction to the *Conditional Gradient Descent algorithm* (Levitin and Polyak, 1966), which is also known as the *Frank-Wolfe algorithm*; see Frank and Wolfe (1956). Assume that f is L-smooth with $L<\infty$. The Frank-Wolfe algorithm with step sizes $\eta_k\in[0,1]$ is defined via the following updates:

$$\mathbf{x}_{k+1}=(1-\eta_k)\mathbf{x}_k+\eta_k\mathbf{v}_k=\mathbf{x}_k+\eta_k(\mathbf{v}_k-\mathbf{x}_k), \quad (2.1)$$

where $\mathbf{x}_0\in\mathcal{X}$ is an arbitrary initial point from the feasible set $\mathcal{X}$ and $\mathbf{v}_k$ is computed using the linear optimization oracle as $\mathbf{v}_k=\operatorname{argmin}_{\mathbf{u}\in\mathcal{X}}\langle\nabla f(\mathbf{x}_k),\mathbf{u}\rangle$.

2.2 Approximate Duality Gap Technique

To analyze the convergence of LaCG, we employ the *Approximate Duality Gap Technique (ADGT)* of Diakonikolas and Orecchia (2019). The core idea behind ADGT is to ensure that A_kG_k is non-increasing with iteration count k, where G_k is an upper approximation of the optimality gap (namely, $f(\mathbf{x}_k)-f(\mathbf{x}^*)\leq G_k$, where $\mathbf{x}_k$ is the point output by the algorithm at iteration k), and A_k is a positive strictly increasing function of k. If such a condition is met, we immediately have $A_kG_k\leq A_0G_0$, which implies $f(\mathbf{x}_k)-f(\mathbf{x}^*)\leq\frac{A_0G_0}{A_k}$. Thus, as long as A_0G_0 is bounded (it typically corresponds to some initial distance to the minimizer $\mathbf{x}^*$), we have that the algorithm converges at rate $1/A_k$. This also means that, to obtain the highest rate of convergence, one should always aim for the fastest-growing A_k for which it holds that $A_kG_k\leq A_{k-1}G_{k-1}$, $\forall k$.

The approximate gap G_k is defined as the difference of an upper bound U_k on the function value at the output point $\mathbf{x}_k$, $U_k \geq f(\mathbf{x}_k)$, and a lower bound L_k on the minimum function value, $L_k \leq f(\mathbf{x}^*)$. Clearly, this choice ensures that $f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq G_k$. In all the algorithms analyzed in this paper, we will use $U_k = f(\mathbf{x}_k)$. The lower bound requires more effort; however, it is similar to those used in previous work (Cohen et al., 2018; Diakonikolas and Orecchia, 2019), and its detailed construction is provided in Appendix A.

Because of its generality, ADGT is well-suited to our setting, as it allows coupling different types of steps and performing a more fine-grained and local analysis than typical approaches. Further, it allows accounting for inexact minimization oracles invoked as part of the algorithm subroutines in a generic way.

3 LaCG Framework

In this section, we establish our main result. We first consider a simple case in which $\mathbf{x}^* \in \operatorname{int}(\mathcal{X})$ as a warm-up to explain our approach and derive LaCG specifically for this case (Section 3.1). Then, in Section 3.2, we consider the more general case where $\mathbf{x}^* \in \operatorname{rel.int}(\mathcal{F})$ with $\mathcal{F}$ being a face of $\mathcal{X}$. Together, Section 3.1 and 3.2 cover all cases of interest (except some degenerate cases). Further, the general LaCG framework presented in Section 3.2 applies to either of the two cases.

3.1 Warm-up: Optimum in the Interior of $\mathcal{X}$

In the case where the optimum is contained in the interior of $\mathcal{X}$, we have $\nabla f(\mathbf{x}^*) = \mathbf{0}$. As such, the unconstrained optimum and the constrained optimum coincide. Thus, one might be tempted to assert that there is no need for an accelerated CG algorithm in this case. However, whether the optimum is contained in the interior is not known *a priori*. The presented algorithm is adaptive, as it accelerates if $\mathbf{x}^* \in \operatorname{int}(\mathcal{X})$, and otherwise it converges with the standard $1/k$ rate.

The main idea can be summarized as follows. Suppose that $\mathbf{x}^*$ is contained $2r$-deep in the *interior* of $\mathcal{X}$. Due to the function's strong convexity, any method that contracts the optimality gap $f(\mathbf{x}_k) - f(\mathbf{x}^*)$ over k also contracts the distance $\|\mathbf{x}_k - \mathbf{x}^*\|$. In particular, this follows by: $\frac{\mu}{2}\|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq f(\mathbf{x}_k) - f(\mathbf{x}^*)$. Hence, roughly, after an iterate $\mathbf{x}_k$ is guaranteed to be inside the $2r$-ball, we can switch to a faster (accelerated) method for *unconstrained* minimization. This idea, however, requires a careful formalization, for the following reasons:

1. In general, we cannot assume that the algorithm has knowledge of r and D, or access to information on $\|\mathbf{x}_k - \mathbf{x}^*\|$, as $\mathbf{x}^*$ is unknown.
2. The algorithm should converge even if $r = 0$; we cannot in general assume that it is a priori known that $\mathbf{x}^* \in \operatorname{int}(\mathcal{X})$. Because, if that were the case and we knew that $r \geq \epsilon$, we would be able to run an accelerated algorithm for $O(\sqrt{L/\mu}\log(1/\epsilon))$ iterations, without any need to worry about whether the outputted solution belongs to $\mathcal{X}$.

We show that both issues can be resolved by implementing a monotonic version of a hybrid algorithm that chooses at each iteration whether to perform an accelerated step or a CG step from Eq. (2.1). Monotonicity is crucial to ensure contraction of the distance to $\mathbf{x}^*$. In *this subsection only*, we assume access to a membership oracle for $\mathcal{X}$. This is generally a mild assumption, especially when $\mathcal{X}$ is a polytope, which is a standard setting for CG.[1] The convergence of the resulting algorithm is shown in the following theorem. Full technical details are deferred to Appendix B.

Theorem 3.1. *Let $\mathbf{x}_k$ be the solution output by Algorithm 2 (Appendix B.1) for $k \geq 1$. If:*

$$k \geq \min\left\{\frac{2LD^2}{\epsilon},\ \frac{LD^2}{\mu r^2} + \sqrt{\frac{L}{\mu}}\log\left(\frac{2(L+\mu)r^2}{\mu\epsilon}\right)\right\},$$

then $f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \epsilon$.

3.2 Optimum in the Relative Interior of a Face of $\mathcal{X}$

We now formulate the general case that subsumes the case from the previous subsection. Note that when $\mathbf{x}^*$ is in the relative interior, it may not be the case that $\nabla f(\mathbf{x}^*) = \mathbf{0}$ anymore. Due to space constraints, we only state the main ideas here, while full technical details are deferred to Appendix B.2.

We assume that, given points $\mathbf{x}_1, ..., \mathbf{x}_m$ and a point $\mathbf{y}$, the following problem is easily solvable:

$$\min_{\substack{\mathbf{u}=\sum_{i=1}^m \lambda_i \mathbf{x}_i,\\ \boldsymbol{\lambda}\in\Delta_m}} \frac{1}{2}\|\mathbf{u} - \mathbf{y}\|^2. \tag{3.1}$$

In other words, we assume that the projection onto the convex hull of a given set of vertices can be implemented efficiently; however, we do not require access to a membership oracle anymore. Solving this problem amounts to minimizing a quadratic function over the

[1] For polynomially-sized LPs, checking membership amounts to evaluating the (in)equalities describing $\mathcal{X}$, which is typically much cheaper than linear optimization. For structured LPs that are solvable in polynomial time but have exponential representation (e.g., matching over non-bipartite graphs (Rothvoß, 2017)), there typically exist membership oracles with running times comparable to those of the LP oracle used in CG steps (Fleischer et al., 2006).

probability simplex. The size of the program m from Eq. (3.1) corresponds to the size of the active set of the CG-type method employed within LaCG. Note that m is never larger than the iteration count k, and is often much lower than the dimension of the original problem. Further, there exist multiple heuristics for keeping the size of the active set small in practice; see, e.g., Braun et al. (2017). The projection from Eq. (3.1) does not require access to either the first-order or the linear optimization oracle. Finally, due to Lemma 3.2 stated below, we only need to solve this problem to accuracy of the order $\frac{\epsilon}{\sqrt{\mu L}}$, where ϵ is the target accuracy.

For simplicity, we illustrate the framework using AFW as the coupled CG method. However, the same ideas can be applied to other active-set-based methods such as PFW in a straightforward manner. Unlike in the previous subsection, the assumption that $\mathcal{X}$ is a polytope is crucial here, as the linear convergence for the AFW algorithm established in Lacoste-Julien and Jaggi (2015) relies on a constant, the *pyramidal width*, that is only known to be bounded away from 0 for polytopes.

To achieve local acceleration, we couple the AFW steps with a modification of the μAGD+ algorithm (Cohen et al., 2018) that we introduce here. This modified μAGD+ algorithm (Lemma 3.2) allows the coupling of the method with an arbitrary sequence of points from the feasible set and supports inexact minimization oracles and changes in the convex set (which correspond to active sets from AFW) on which projections are performed. These modifications allow local acceleration without any additional knowledge about the polytope or the position of the minimizer $\mathbf{x}^*$. Further, we are not aware of any other methods that allow changes to the feasible set as described here, and, thus, the result from Lemma 3.2 may be of independent interest.

Lemma 3.2. *(Convergence of the modified μAGD+) Let $f : \mathcal{X} \to \mathbb{R}$ be L-smooth and μ-strongly convex, and let $\mathcal{X}$ be a closed convex set. Let $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{u}\in\mathcal{X}} f(\mathbf{x})$, and let $\{\mathcal{C}_i\}_{i=0}^k$ be a sequence of convex subsets of $\mathcal{X}$ such that $\mathcal{C}_i \subseteq \mathcal{C}_{i-1}$ for all i and $\mathbf{x}^* \in \bigcap_{i=0}^k \mathcal{C}_i$. Let $\{\tilde{\mathbf{x}}_i\}_{i=0}^k$ be any (fixed) sequence of points from $\mathcal{X}$. Let $a_0 = 1$, $\frac{a_k}{A_k} = \theta$ for $k \geq 1$, where $A_k = \sum_{i=0}^k a_i$ and $\theta = \sqrt{\frac{\mu}{2L}}$. Let $\mathbf{y}_0 \in \mathcal{X}$, $\mathbf{x}_0 = \mathbf{w}_0$, and $\mathbf{z}_0 = L\mathbf{y}_0 - \nabla f(\mathbf{y}_0)$. For $k \geq 1$, define iterates $\mathbf{x}_k$ by:*

$$\begin{aligned} \mathbf{y}_k &= \frac{1}{1+\theta}\mathbf{x}_{k-1} + \frac{\theta}{1+\theta}\mathbf{w}_{k-1}, \\ \mathbf{z}_k &= \mathbf{z}_{k-1} - a_k\nabla f(\mathbf{y}_k) + \mu a_k \mathbf{y}_k, \\ \hat{\mathbf{x}}_k &= (1-\theta)\mathbf{x}_{k-1} + \theta\mathbf{w}_k, \\ \mathbf{x}_k &= \operatorname{argmin}\{f(\hat{\mathbf{x}}_k),\, f(\tilde{\mathbf{x}}_k)\} \end{aligned} \tag{3.2}$$

where, for all $k \geq 0$, $\mathbf{w}_k$ is defined as an ϵ_k^m-approximate solution of:

$$\min_{\mathbf{u}\in\mathcal{C}_k} \Big\{ -\langle \mathbf{z}_k, \mathbf{u}\rangle + \frac{\mu A_k + \mu_0}{2}\|\mathbf{u}\|^2 \Big\}, \tag{3.3}$$

with $\mu_0 \stackrel{\text{def}}{=} L - \mu$. Then, for all $k \geq 0$, $\mathbf{x}_k \in \mathcal{X}$ and:

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq (1-\theta)^k \frac{(L-\mu)\|\mathbf{x}^* - \mathbf{y}_0\|^2}{2} + \frac{2\sum_{i=0}^{k-1}\epsilon_i^m + \epsilon_k^m}{A_k}.$$

A simple observation, which turns out to be key for the coupling to work is that when running AFW, there exists an iteration K_0 such that for all $k \geq K_0$ it holds that $\mathbf{x}^* \in \operatorname{co}(\mathcal{S}_k^{\text{AFW}})$, where $\mathcal{S}_k^{\text{AFW}}$ denotes the active set maintained by AFW in iteration k. This iteration K_0 only depends on the feasible region $\mathcal{X}$ and $\mathbf{x}^*$ and, as such, it is a burn-in period of constant length. In particular, this fact is a consequence of strong convexity of f that ensures that after a certain number of steps, all iterates are localized in a ball of radius r_0 centered at $\mathbf{x}^*$, where r_0 is independent of target error ϵ.

To obtain locally accelerated convergence, we show that from iteration K_0 onwards, we can apply the accelerated method from Lemma 3.2 with $\mathcal{C}_k$ being the convex hull of the vertices from the active set and the sequence $\tilde{\mathbf{x}}_k$ being the sequence of the AFW steps. The pseudocode for the LaCG-AFW algorithm is provided in Algorithm 1. For completeness, pseudocode for one iteration of the accelerated method (ACC), based on Eq. (3.2) is provided in Algorithm 5 (Appendix B.2).

Our main theorem is stated below, with a proof sketch. The full proof is deferred to Appendix B.2.

Main Theorem 3.3. (Convergence analysis of *Locally Accelerated Conditional Gradients*) Let $\mathbf{x}_k$ be the solution output by Algorithm 1 and r_0 be the critical radius (see Fact B.3 in Appendix B.2). If:

$$k \geq \min\left\{ \frac{8L}{\mu}\left(\frac{D}{\delta}\right)^2 \log\left(\frac{f(\mathbf{x}_0) - f(\mathbf{x}^*)}{\epsilon}\right), \; K_0 + H + 2\sqrt{\frac{2L}{\mu}}\log\left(\frac{(L-\mu){r_0}^2}{2\epsilon}\right)\right\},$$

where $H = 2\sqrt{2L/\mu}\log(L/\mu - 1)$ and $K_0 = \frac{8L}{\mu}\left(\frac{D}{\delta}\right)^2 \log\left(\frac{2(f(\mathbf{x}_0)-f(\mathbf{x}^*))}{\mu {r_0}^2}\right)$, then:

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \epsilon.$$

Proof Sketch. The statement of the theorem is a direct consequence of the following observations about Algorithm 1. First, observe that the AFW algorithm is run independently of the accelerated sequence, and, in particular, the accelerated sequence has no effect on the

Algorithm 1 Locally Accelerated Conditional Gradients with Away-Step Frank-Wolfe (LaCG-AFW)

Let $\mathbf{x}_0 \in \mathcal{X}$ be an arbitrary point, $\mathcal{S}_0^{\text{AFW}} = \{\mathbf{x}_0\}$, $\boldsymbol{\lambda}_0^{\text{AFW}} = [1]$
Let $\mathbf{y}_0 = \hat{\mathbf{x}}_0 = \mathbf{w}_0 = \mathbf{x}_0$, $\mathbf{z}_0 = -\nabla f(\mathbf{y}_0) + L\mathbf{y}_0$, $\mathcal{C}_1 = \text{co}(\mathcal{S}_0^{\text{AFW}})$, $a_0 = A_0 = 1$, $\theta = \sqrt{\frac{\mu}{2L}}$, $\mu_0 = L - \mu$
$H = \frac{2}{\theta}\log(1/(2\theta^2) - 1)$ ▷ Minimum restart period
$r_f = \text{false}$, $r_c = 0$ ▷ Restart flag and restart counter initialization
for $k = 1$ to K **do**
  $\mathbf{x}_k^{\text{AFW}}, \mathcal{S}_k^{\text{AFW}}, \boldsymbol{\lambda}_k^{\text{AFW}} = \text{AFW}(\mathbf{x}_{k-1}^{\text{AFW}}, \mathcal{S}_{k-1}^{\text{AFW}}, \boldsymbol{\lambda}_{k-1}^{\text{AFW}})$ ▷ Independent AFW update
  $A_k = A_{k-1}/(1-\theta)$, $a_k = \theta A_k$
  if r_f and $r_c \geq H$ **then** ▷ Restart criterion is met
    $\mathbf{y}_k = \text{argmin}\{f(\mathbf{x}_k^{\text{AFW}}), f(\hat{\mathbf{x}}_k)\}$
    $\mathcal{C}_{k+1} = \text{co}(\mathcal{S}_k^{\text{AFW}})$ ▷ Updating feasible set for the accelerated sequence
    $a_k = A_k = 1$, $\mathbf{z}_k = -\nabla f(\mathbf{y}_k) + L\mathbf{y}_k$ ▷ Restarting accelerated sequence
    $\hat{\mathbf{x}}_k = \mathbf{w}_k = \text{argmin}_{\mathbf{u}\in\mathcal{C}_{k+1}}\{-\langle \mathbf{z}_k, \mathbf{u}\rangle + \frac{L}{2}\|\mathbf{u}\|^2\}$
    $r_c = 0$, $r_f = \text{false}$ ▷ Resetting the restart indicators
  else
    if $\mathcal{S}_k^{\text{AFW}} \setminus \mathcal{S}_{k-1}^{\text{AFW}} \neq \emptyset$ **then** ▷ If a vertex was added to the active set
      $r_f = \text{true}$ ▷ Raise freeze flag
    if $r_f = \text{true}$ **then**
      $\mathcal{C}_k = \mathcal{C}_{k-1}$ ▷ Freeze the feasible set
      $\hat{\mathbf{x}}_k, \mathbf{z}_k, \mathbf{w}_k = \text{ACC}(\hat{\mathbf{x}}_{k-1}, \mathbf{z}_{k-1}, \mathbf{w}_{k-1}, a_k, A_k, \mathcal{C}_k)$ ▷ Run AGD+ uncoupled from CG.
    else
      $\mathcal{C}_k = \text{co}(\mathcal{S}_k^{\text{AFW}})$ ▷ Update the feasible set
      $\hat{\mathbf{x}}_k, \mathbf{z}_k, \mathbf{w}_k = \text{ACC}(\mathbf{x}_{k-1}, \mathbf{z}_{k-1}, \mathbf{w}_{k-1}, a_k, A_k, \mathcal{C}_k)$ ▷ Run AGD+ coupled to CG.
  $\mathbf{x}_k = \text{argmin}\{f(\mathbf{x}_k^{\text{AFW}}), f(\hat{\mathbf{x}}_k), f(\mathbf{x}_{k-1})\}$ ▷ Choose the better step + monotonicity
  $r_c = r_c + 1$ ▷ Increment the restart counter

AFW-sequence whatsoever. Further, in any iteration, the set $\mathcal{C}_k$ that we project onto is the convex hull of some active set $\mathcal{S}_i^{\text{AFW}} \subseteq \mathcal{X}$ for some $0 \leq i \leq k-1$ implying $\hat{\mathbf{x}}_k \in \mathcal{X}$ – each $\hat{\mathbf{x}}_k$ is hence feasible.

Now, as in any iteration k the solution outputted by the algorithm is $\mathbf{x}_k = \text{argmin}\{f(\mathbf{x}_k^{\text{AFW}}), f(\hat{\mathbf{x}}_k)\}$, the algorithm never makes less progress than AFW. This immediately implies (by a standard AFW guarantee; see Lacoste-Julien and Jaggi (2015) and Proposition B.4) that for $k \geq \frac{8L}{\mu}\left(\frac{D}{\delta}\right)^2 \log\left(\frac{f(\mathbf{x}_0)-f(\mathbf{x}^*)}{\epsilon}\right)$, it must be that $f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \epsilon$, which establishes the unaccelerated part of the minimum in the asserted rate.

Further, there exists an iteration $K \leq K_0$ such that for all $k \geq K$ it holds $\mathbf{x}^* \in \text{co}(\mathcal{S}_k^{\text{AFW}})$ (see Proposition B.4). Let K be the first such iteration. Then, the AFW algorithm must have added a vertex in iteration K as otherwise $\mathbf{x}^* \in \text{co}(\mathcal{S}_{k-1}^{\text{AFW}})$, contradicting the minimality of K. Due to the restarting criterion from Algorithm 1, a restart must happen by iteration $K_0 + H$. Thus, for $k \geq K_0 + H$, it must be $\mathbf{x}^* \in \mathcal{C}_k$. The rest of the proof invokes Lemma 3.2 and its corollary (Corollary B.5), which ensures accelerated convergence following iteration $K_0 + H$. □

Remark 3.4 (Inexact projection oracles)**.** We stated Theorem 3.3 assuming exact minimization oracle ($\epsilon_i^m = 0$ in Lemma 3.2). Clearly, it suffices to have $\epsilon_i^m = \frac{a_i}{8}\epsilon$ and invoke Theorem 3.3 for target accuracy $\epsilon/2$.

Remark 3.5 (Early stopping)**.** If in any iteration the Wolfe gap of the accelerated sequence on $\mathcal{C}_k$ is smaller than the target accuracy of the projection subproblem (order-$\frac{\epsilon}{\sqrt{\mu L}}$), then f cannot be reduced by more than order-$\frac{\epsilon}{\sqrt{\mu L}}$ on $\mathcal{C}_k$, and one can restart without affecting the theoretical convergence guarantee.

Remark 3.6 (Variant relying exclusively on a linear optimization oracle)**.** Similar as in the Conditional Gradient Sliding (CGS) algorithm (Lan and Zhou, 2016) we can solve the projection problems using CG. In fact, a variant of CGS is recovered if we ignore the AFW steps and only run the accelerated sequence with such projections realized by CG.

4 Computational Results

The theoretical results from the previous section showed that LaCG leads to asymptotically optimal local convergence rates for smooth strongly convex optimization over polytopes. However, this acceleration comes at the cost of more computationally-intensive iterations. In this section, we show that the improvement in the convergence rate is sufficiently high to offset the increased cost of iterations, allowing LaCG to outperform other CG variants in terms of wall-clock time in most settings.

We implemented Algorithm 1 in Python 3, employing the $\mathcal{O}(n \log n)$ projections onto the simplex described in Duchi et al. (2008, Algorithm 1) and Nesterov's accelerated gradient descent (Nesterov, 2018) to solve the projection subproblems. In all experiments, we used $\epsilon = 10^{-5} \cdot (f(x_0) - f(x^*))$ as the target error. As mentioned before, LaCG can be used with any CG variant that maintains an active set; this excludes e.g., *Decomposition invariant CG (DiCG)* variants; see Garber and Meshi (2016); Bashiri and Zhang (2017).

Results for each example are shown in a figure with four plots. The first two plots compare standard LaCG variants to AFW, PFW, and DiCG (if applicable) in iterations (log-log scale) and wall-clock time (log scale). The second two plots compare the standard and lazified variants of LaCG (denoted by (L)) to CGS and Catalyst in iterations (log-log scale) and wall-clock time (log scale). As shown in Figs. 1-4, LaCG can make up for more expensive iterations thanks to its higher convergence rate following the burn-in phase, outperforming other methods in many important examples.

There are also examples in which LaCG is outperformed by other methods; in particular, by DiCG (see Appendix C). This happens when the linear optimization oracle is very efficient and the active sets are large. In this case, DiCG outperforms all active set based methods. Note that, however, DiCG is not broadly applicable; see, e.g., the discussion for the structured regression example below. In all other experiments (see Figs. 1-4) LaCG outperforms DiCG.

Birkhoff polytope. The first example, shown in Fig. 1(a)-1(d), corresponds to minimization over the Birkhoff polytope in $\mathbb{R}^{n \times n}$ where $n^2 = 1600$ with $f(\mathbf{x}) = \mathbf{x}^T \frac{M^T M + I}{2} \mathbf{x}$, $M \in \mathbb{R}^{n \times n}$ is a sparse matrix whose 1% of the elements are drawn from a standard Gaussian distribution, and I is the identity matrix (the matrix $M^T M$ has 15% non-zero elements). The resulting condition number is $L/\mu \approx 100$. The linear optimization oracle in this instance uses the Hungarian algorithm, which has complexity $\mathcal{O}(n^3)$.

Structured Regression. The second example, shown in Fig. 2(a)-2(d), corresponds to a structured regression problem over the convex hull of the feasible set defined by integer and linear constraints, where the linear optimization oracle corresponds to solving a mixed integer program (MIP) with a linear objective function. The specific instance used is `ran14x18-disj-8`, of dimension 504, from the MIPLIB library. The objective function $f(\mathbf{x}) = \mathbf{x}^T \frac{M}{2} \mathbf{x} + \mathbf{b}$ was obtained by first generating an orthonormal basis $\mathcal{B} = \{\mathbf{u}_1, \cdots, \mathbf{u}_n\}$ in $\mathbb{R}^n$ and a set of n uniformly distributed values $\{\lambda_1, \cdots, \lambda_n\}$ between μ and L and setting $M = \sum_{i=1}^n \lambda_i \mathbf{u}_i \mathbf{u}_i^T$. The vector $\mathbf{b}$ was set to be outside of the feasible region, and has random entries uniformly distributed in $[0, 1]$. Note that DiCG (Garber and Meshi, 2016; Bashiri and Zhang, 2017) is not applicable here, as the representation of the MIPLIP polytope mixes continuous and integer variables. In fact, for polytopes over which linear optimization is NP-hard (e.g., TSP polytope), efficiently computing away steps with an away step oracle as in Bashiri and Zhang (2017) is not possible unless NP = co-NP.

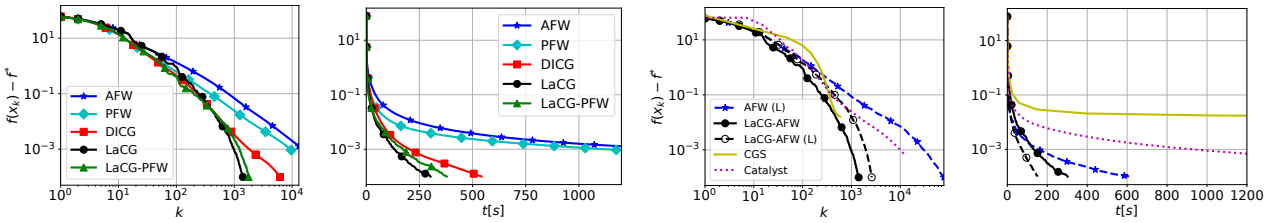
Congestion Balancing in Traffic Networks. We consider the problem of congestion balancing in traffic networks. This problem can be posed as a feasible flow with multiple source-sink pairs and with convex costs on flows over the edges (see, e.g., Ahuja et al. (1993, Chapter 14)). We choose the costs to be weighted quadratic, as such costs have the role of balancing the congestion over the network edges (see, e.g., Diakonikolas et al. (2018)). The weights are chosen randomly between μ and L, with $L/\mu = 100$, leading to a separable quadratic objective. The problem instance is the `road_paths_01_DC_a` flow polytope (same as in Lan et al. (2017) and Braun et al. (2019), with dimension $n = 29682$). As the flow polytope in this case is not a 0/1 polytope, DiCG (Garber and Meshi, 2016) does not apply to this problem instance.

Probability Simplex. The last example, shown in Fig. 4(a)-4(d), is the probability simplex, which, while a toy example, lends itself nicely for comparisons between methods. Due to the structure of this polytope, there is no need to explicitly maintain an active set in the AFW, PFW or the LaCG algorithm. This greatly speeds up all the algorithms, and eliminates the main advantage of the DiCG algorithm. There are also further enhancements that can be made to the LaCG algorithm in this case (see Appendix C.3). We generate the objective function in the same way as in the Structured Regression, with $n = 1500$, $L/\mu = 1000$ and with $\mathbf{b}$ having random integer values of $-1, 0$, or 1.

Lazification. Our proposed approach is also compatible with the lazification technique from Braun et al. (2017). To demonstrate the advantage of lazification in terms of wall-clock time, we run it as an alternative version of LaCG when comparing against CGS and Catalyst for each example. As a reference and for comparison, we also include the lazified AFW algorithm.

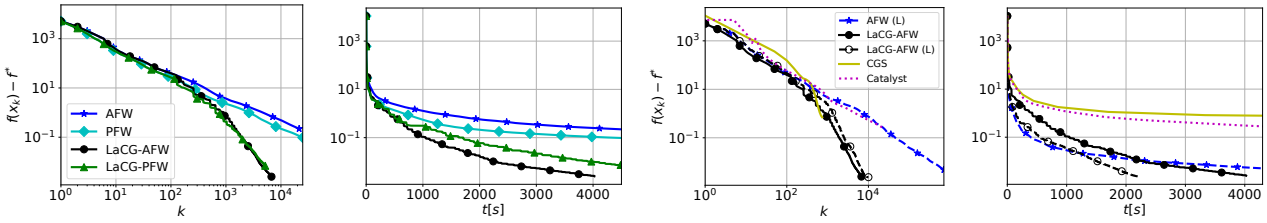
5 Discussion

We presented the Locally-Accelerated Conditional Gradients framework that achieves asymptotically optimal rate in a local region around the minimum and improves



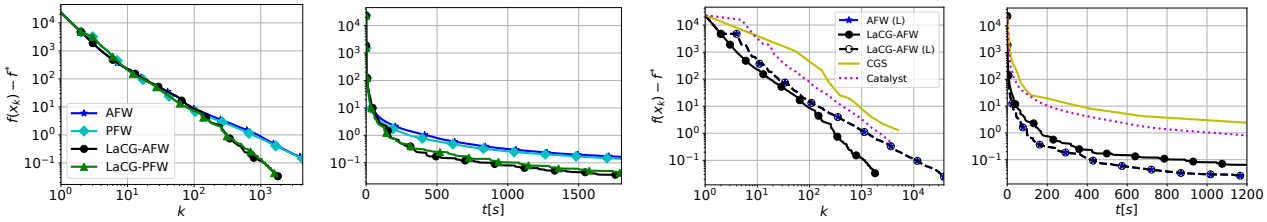
(a) Iteration (b) Wall-clock time (c) Iteration (d) Wall-clock time

Figure 1: Birkhoff polytope: Algorithm comparison in terms of (a),(c) iteration count and (b),(d) wall-clock time.



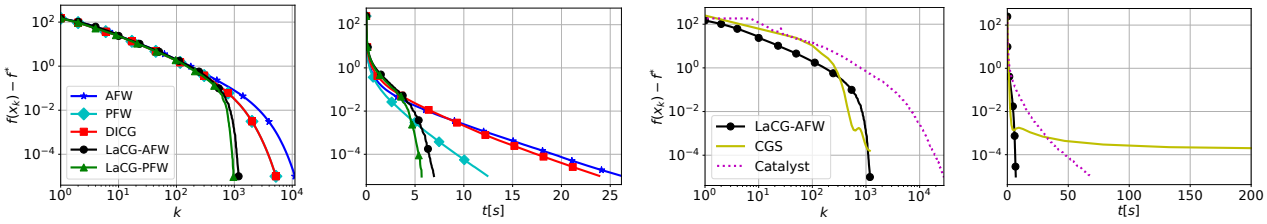
(a) Iteration (b) Wall-clock time (c) Iteration (d) Wall-clock time

Figure 2: MIPLIB polytope: Algorithm comparison in terms of (a),(c) iteration count and (b),(d) wall-clock time for the `ran14x18-disj-8` polytope from the MIPLIB library.



(a) Iteration (b) Wall-clock time (c) Iteration (d) Wall-clock time

Figure 3: Traffic network: Algorithm comparison in terms of (a),(c) iteration count and (b),(d) wall-clock time.



(a) Iteration (b) Wall-clock time (c) Iteration (d) Wall-clock time

Figure 4: Simplex: Algorithm comparison in terms of (a),(c) iteration count and (b),(d) wall-clock time.

upon the existing conditional gradients methods, both in theory and in experiments. As discussed before, such an accelerated rate cannot be achieved globally.

Some interesting questions for future research remain. For example, can we obtain dimension-independent local acceleration (i.e., $1/k^2$ local convergence rate) for smooth (non-strongly) convex minimization? The current impediment to extending our techniques to this case is that the burn-in time becomes dependent on ϵ, scaling with $1/\epsilon$, which effectively cancels out any benefits from local acceleration. Further, it would be interesting to understand whether the version of μAGD+ from Lemma 3.2, which allows changing the projection set $\mathcal{C}_k$, can speed up the practical performance of accelerated methods in other settings.

Acknowledgements

Research reported in this paper was partially supported by the NSF grant CCF-1740855, NSF CAREER Award CMMI-1452463, and Award W911NF-18-1-0223 from the Army Research Office. Part of it was done while JD

and SP were visiting Simons Institute for the Theory of Computing.

References

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.

Mohammad Ali Bashiri and Xinhua Zhang. Decomposition-invariant conditional gradient for general polytopes with line search. In *Proc. NIPS'17*, 2017.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009.

Michael Betancourt, Michael I Jordan, and Ashia C Wilson. On symplectic optimization. *arXiv preprint arXiv:1802.03653*, 2018.

G. Braun, S. Pokutta, and D. Zink. Lazifying Conditional Gradient Algorithms. In *Proc. ICML'17*, 2017.

Gábor Braun, Sebastian Pokutta, Dan Tu, and Stephen Wright. Blended conditional gradients: the unconditioning of conditional gradients. In *Proc. ICML'19*, 2019.

Sébastien Bubeck, Yin Tat Lee, and Mohit Singh. A geometric alternative to Nesterov's accelerated gradient descent. *arXiv preprint, arXiv:1506.08187*, 2015.

Michael B Cohen, Jelena Diakonikolas, and Lorenzo Orecchia. On acceleration with noise-corrupted gradients. In *Proc. ICML'18*, 2018.

Jelena Diakonikolas and Lorenzo Orecchia. Accelerated extra-gradient descent: A novel, accelerated first-order method. In *Proc. ITCS'18*, 2018.

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

Jelena Diakonikolas, Maryam Fazel, and Lorenzo Orecchia. Width-independence beyond linear objectives: Distributed fair packing and covering algorithms. *arXiv preprint arXiv:1808.02517*, 2018.

Dmitriy Drusvyatskiy, Maryam Fazel, and Scott Roy. An optimal first order method based on optimal quadratic averaging. *SIAM J. Optimiz.*, 28(1):251–271, 2018.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proc. NIPS'08*, 2008.

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Proc. NIPS'18*, 2018.

Lisa K Fleischer, Adam N Letchford, and Andrea Lodi. Polynomial-time separation of a superclass of simple comb inequalities. *Math. of Oper. Res.*, 31(4): 696–713, 2006.

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

Futoshi Futami, Zhenghang Cui, Issei Sato, and Masashi Sugiyama. Bayesian posterior approximation via greedy particle optimization. In *Proc. AAAI'19*, 2019.

D. Garber and E. Hazan. A linearly convergent variant of the conditional gradient algorithm under strong convexity, with applications to online and stochastic optimization. *SIAM J. Optimiz.*, 26(3):1493–1528, 2016.

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

Dan Garber, Shoham Sabach, and Atara Kaplan. Fast generalized conditional gradient method with applications to matrix recovery problems. *arXiv preprint arXiv:1802.05581*, 2018.

Jacques Guélat and Patrice Marcotte. Some comments on Wolfe's 'away step'. *Math. Program.*, 35(1):110–119, 1986.

David H Gutman and Javier F Pena. The condition number of a function relative to a set. *arXiv preprint arXiv:1901.08359*, 2019.

Hamed Hassani, Amin Karbasi, Aryan Mokhtari, and Zebang Shen. Stochastic conditional gradient++. *arXiv preprint arXiv:1902.06992*, 2019.

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

Armand Joulin, Kevin Tang, and Li Fei-Fei. Efficient image and video co-localization with Frank-Wolfe algorithm. In *Proc. ECCV'14*, 2014.

Thomas Kerdreux, Alexandre d'Aspremont, and Sebastian Pokutta. Restarting Frank–Wolfe. In *Proc. AISTATS'18*, 2018a.

Thomas Kerdreux, Fabian Pedregosa, and Alexandre D'Aspremont. Frank-Wolfe with subsampling oracle. In *Proc. ICML'18*, 2018b.

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

Simon Lacoste-Julien, Martin Jaggi, Mark W Schmidt, and Patrick Pletscher. Block-coordinate Frank-Wolfe optimization for structural svms. In *Proc. ICML'13*, 2013.

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

Guanghui Lan and Yi Zhou. Conditional gradient sliding for convex optimization. *SIAM J. Optimiz.*, 26(2):1379–1409, 2016.

Guanghui Lan, Sebastian Pokutta, Yi Zhou, and Daniel Zink. Conditional accelerated lazy stochastic gradient descent. In *Proc. ICML'17*, 2017.

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

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Proc. NIPS'15*, 2015.

Yu E Nesterov. An $O(1/k)$-rate of convergence method for smooth convex functions minimization. In *Dokl. Acad. Nauk SSSR*, volume 269, pages 543–547, 1983.

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

Javier Pena and Daniel Rodriguez. Polytope conditioning and linear convergence of the Frank-Wolfe algorithm. *Math. of Oper. Res.*, 44(1):1–18, 2018.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. & Math. Phys.*, 4(5):1–17, 1964.

Thomas Rothvoß. The matching polytope has exponential extension complexity. *Journal of the ACM (JACM)*, 64(6):41, 2017.

Mark Schmidt, Nicolas L Roux, and Francis R Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Proc. NIPS'11*, 2011.

Zebang Shen, Cong Fang, Peilin Zhao, Junzhou Huang, and Hui Qian. Complexities in projection-free stochastic non-convex minimization. In *Proc. AISTATS'19*, 2019.

Paul Swoboda and Vladimir Kolmogorov. MAP inference via block-coordinate Frank-Wolfe algorithm. In *Proc. IEEE CVPR'19*, 2019.

Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization, 2008.