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., 2018a; 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
arbitrary methods whose access to the feasible region is limited to a linear optimization oracle. As an illustration, let $\mathcal{X} \triangleq \{x \in \mathbb{R}^n \mid \sum_{i=1}^n x_i = 1, x_i \geq 0\}$ be the probability simplex on $n$ coordinates and consider:

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

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 $f(x_k) - f(x^*) \geq \frac{1}{k} - \frac{1}{n}$. In particular, for $k = n/2$ (assuming w.l.o.g. that $n$ is even), the primal gap is bounded below by $f(x_{n/2}) - f(x^*) \geq \frac{1}{2}$, 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 $f(x_k) - f(x^*) \leq e^{-\rho k} (f(x_0) - f(x^*))$ with a global rate $\rho$, then it follows that $\rho \leq 2 \frac{\log n}{n}$. The Fully-Corrective Frank-Wolfe algorithm converges with (roughly) $\rho = \frac{1}{2\sqrt{n}}$ (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 $\rho = \sqrt{\log n}$ 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 $\epsilon$ (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 $\mu$AGD+ (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 $K + O\left(\sqrt{\frac{n}{\mu}} \log \frac{1}{\epsilon}\right)$ to solve $\min_{x \in \mathcal{X}} f(x)$ up to accuracy $\epsilon$, where $f$ is $L$-smooth and $\mu$-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 $\mu$ 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 $O\left(\sqrt{\frac{n}{\mu}} \log \frac{1}{\epsilon}\right)$ 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 $O\left(\sqrt{\frac{n}{\mu}} \frac{D^2}{\delta^2} \log \frac{1}{\epsilon}\right)$, where $D$ is the diameter of $\mathcal{X}$ and $\delta$ 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 optimization 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 \( \mu \text{AGD+} \) (Cohen et al., 2018) algorithm in a few important ways (note that \( \mu \text{AGD+} \) is also a generalization of Nesterov’s method).

First, we show that \( \mu \text{AGD+} \) 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 \( \mu \text{AGD+} \) 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 \( X \) or the position of the function minimizer \( x^* \).

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

Finally, we prove that \( \mu \text{AGD+} \) 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 \( x^* \). 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 \( \min_{x \in X} f(x) \), where \( f \) is a smooth (gradient Lipschitz) strongly convex function and \( X \subseteq \mathbb{R}^n \) is a convex polytope. We assume (i) first-order access to \( f \): given \( x \in X \), we can compute \( f(x) \) and \( \nabla f(x) \), and (ii) linear optimization oracle access to \( A \): given a vector \( c \in \mathbb{R}^n \), we can compute \( v = \text{argmin}_{u \in A} \langle c, u \rangle \).

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

### 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:

\[
x_{k+1} = (1 - \eta_k)x_k + \eta_k v_k = x_k + \eta_k (v_k - x_k),
\]

where \( x_0 \in X \) is an arbitrary initial point from the feasible set \( X \) and \( v_k \) is computed using the linear optimization oracle as \( v_k = \text{argmin}_{u \in X} \langle \nabla f(x_k), u \rangle \).

### 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 \( A_k G_k \) is non-increasing with iteration count \( k \), where \( G_k \) is an upper approximation of the optimality gap (namely, \( f(x_k) - f(x^*) \leq G_k \), where \( x_k \) 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 \( A_k G_k \leq A_0 G_0 \), which implies \( f(x_k) - f(x^*) \leq A_0 G_0 / A_k \). Thus, as long as \( A_0 G_0 \) is bounded (it typically corresponds to some initial distance to the minimizer \( x^* \)), 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 \( A_k G_k \leq A_{k-1} G_{k-1}, \forall k \).
The approximate gap $G_k$ is defined as the difference of an upper bound $U_k$ on the function value at the output point $x_k$, $U_k \geq f(x_k)$, and a lower bound $L_k$ on the minimum function value, $L_k \leq f(x^*)$. Clearly, this choice ensures that $f(x_k) - f(x^*) \leq G_k$. In all the algorithms analyzed in this paper, we will use $U_k = f(x_k)$. 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 $x^* \in \text{int}(X)$ 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 $x^* \in \text{rel. int}(F)$ with $F$ being a face of $X$. 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 $X$

In the case where the optimum is contained in the interior of $X$, we have $\nabla f(x^*) = 0$. 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 $x^* \in \text{int}(X)$, and otherwise it converges with the standard $1/k$ rate.

The main idea can be summarized as follows. Suppose that $x^*$ is contained $2r$-deep in the interior of $X$. Due to the function’s strong convexity, any method that contracts the optimality gap $f(x_k) - f(x^*)$ over $k$ also contracts the distance $\|x_k - x^*\|$. In particular, this follows by: $\frac{\mu}{2} \|x_k - x^*\|^2 \leq f(x_k) - f(x^*)$. Hence, roughly, after an iterate $x_k$ 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 $\|x_k - x^*\|$, as $x^*$ is unknown.

2. The algorithm should converge even if $r = 0$; we cannot in general assume that it is a priori known that $x^* \in \text{int}(X)$. 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 $X$.

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 $x^*$. In this subsection only, we assume access to a membership oracle for $X$. This is generally a mild assumption, especially when $X$ is a polytope, which is a standard setting for CG. The convergence of the resulting algorithm is shown in the following theorem. Full technical details are deferred to Appendix B.

Theorem 3.1. Let $x_k$ 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 $f(x_k) - f(x^*) \leq \epsilon$.

3.2 Optimum in the Relative Interior of a Face of $X$

We now formulate the general case that subsumes the case from the previous subsection. Note that when $x^*$ is in the relative interior, it may not be the case that $\nabla f(x^*) = 0$ 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 $x_1, \ldots, x_m$ and a point $y$, the following problem is easily solvable:

$$\min_{\lambda \in \Delta_m} \frac{1}{2} \|u - y\|^2.$$

(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

For polynomially-sized LPs, checking membership amounts to evaluating the (in)equalities describing $X$, 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 \( \frac{\epsilon}{\sqrt{\mu_k^2}} \), where \( \epsilon \) 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 \( X \) 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 \( \mu \)AGD+ algorithm (Cohen et al., 2018) that we introduce here. This modified \( \mu \)AGD+ 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 \( x^* \). 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 \( \mu \)AGD+) Let \( f : X \to \mathbb{R} \) be \( L \)-smooth and \( \mu \)-strongly convex, and let \( X \) be a closed convex set. Let \( x^* = \arg\min_{x \in X} f(x) \), and let \( \{C_i\}_{i=0}^k \) be a sequence of convex subsets of \( X \) such that \( C_i \subseteq C_{i-1} \) for all \( i \) and \( x^* \in \bigcap_{i=0}^k C_i \). Let \( \{\tilde{x}_i\}_{i=0}^k \) be any (fixed) sequence of points from \( X \). Let \( a_0 = 1 \), \( \frac{1}{\sqrt{4L}} \) = \( \theta \) for \( k \geq 1 \), where \( A_k = \sum_{i=0}^k a_i \) and \( \theta = \sqrt{\frac{1}{4L}} \). Let \( y_0 \in X \), \( x_0 = w_0 \), and \( z_0 = L y_0 - \nabla f(y_0) \).

For \( k \geq 1 \), define iterates \( x_k \) by:

\[
\begin{align*}
  y_k &= \frac{1}{1+\theta} x_{k-1} + \frac{\theta}{1+\theta} w_{k-1}, \\
  z_k &= z_{k-1} - a_k \nabla f(y_k) + \mu a_k y_k, \\
  \hat{x}_k &= (1-\theta) x_{k-1} + \theta w_k, \\
  x_k &= \arg\min \{ f(\hat{x}_k), f(x_{k-1}) \}.
\end{align*}
\]

where, for all \( k \geq 0 \), \( w_k \) is defined as an \( \epsilon^m \)-approximate solution of:

\[
\min_{u \in C_k} \left\{ -\langle z_k, u \rangle + \frac{\mu A_k + \mu_0}{2} \|u\|^2 \right\},
\]

with \( \mu_0 \equiv L - \mu \). Then, for all \( k \geq 0 \), \( x_k \in X \) and:

\[
f(x_k) - f(x^*) \leq \left( 1 - \theta \right)^k \frac{(L - \mu) \|x^* - y_0\|^2}{2} + \frac{2 \sum_{i=0}^{k-1} \epsilon_i^m + \epsilon_k^m}{A_k}.
\]

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 \( x^* \in \text{co}(S_{k}^{\text{AFW}}) \), where \( S_{k}^{\text{AFW}} \) denotes the active set maintained by AFW in iteration \( k \). This iteration \( K_0 \) only depends on the feasible region \( X \) and \( x^* \) 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 \( x^* \), where \( r_0 \) is independent of target error \( \epsilon \).

To obtain locally accelerated convergence, we show that from iteration \( K_0 \) onwards, we can apply the accelerated method from Lemma 3.2 with \( C_k \) being the convex hull of the vertices from the active set and the sequence \( \tilde{x}_k \) 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 \( x_k \) be the solution output by Algorithm 1 and \( r_0 \) be the critical radius (see Fact B.3 in Appendix B.2). If:

\[
k \geq \min \left\{ \frac{8L}{\mu} \left( \frac{D}{\theta} \right)^2 \log \left( \frac{f(x_0) - f(x^*)}{\epsilon} \right), \right. \\
K_0 + H + 2 \sqrt{\frac{2L}{\mu}} \log \left( \frac{L - \mu) r_0^2}{2\epsilon} \right) \}
\]

where \( H = 2 \sqrt{\frac{2L}{\mu}} \log((L/\mu) - 1) \) and \( K_0 = \frac{8L}{\mu} \left( \frac{D}{\theta} \right)^2 \log \left( \frac{2(f(x_0) - f(x^*))}{\mu^2 \epsilon^2} \right) \), then:

\[
f(x_k) - f(x^*) \leq \epsilon.
\]

**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)

1: Let \( x_0 \in \mathcal{X} \) be an arbitrary point, \( S_0^{\text{AFW}} = \{ x_0 \} \), \( \lambda_0^{\text{AFW}} = [1] \)
2: Let \( y_0 = x_0 = w_0 = x_0 \), \( z_0 = -\nabla f(y_0) + Ly_0 \), \( C_1 = \text{co}(S_0^{\text{AFW}}, a_0 = A_0 = 1, \theta = \sqrt{\frac{r}{\mu L}}, \mu_0 = L - \mu \)
3: \( H = \frac{2}{\theta} \log(1/(2\theta^2 - 1)) \) \( \triangleright \) Minimum restart period
4: \( r_f = \text{false}, r_c = 0 \) \( \triangleright \) Restart flag and restart counter initialization
5: for \( k = 1 \) to \( K \) do
6: \( x_k^{\text{AFW}}, S_k^{\text{AFW}}, \lambda_k^{\text{AFW}} = \text{AFW}(x_{k-1}^{\text{AFW}}, S_{k-1}^{\text{AFW}}, \lambda_{k-1}^{\text{AFW}}) \) \( \triangleright \) Independent AFW update
7: \( A_k = A_{k-1}/(1 - \theta), a_k = \theta A_k \)
8: if \( r_f \) and \( r_c \geq H \) then 
\hspace{1em} \( y_k = \text{argmin}\{f(x_k^{\text{AFW}}), f(\tilde{x}_k)\} \) \( \triangleright \) Updaring feasible set for the accelerated sequence
\hspace{1em} \( C_k = \text{co}(S_k^{\text{AFW}}) \)
\hspace{1em} \( a_k = A_k = 1, z_k = -\nabla f(y_k) + Ly_k \)
\hspace{1em} \( \tilde{x}_k = w_k = \text{argmin}_{i \in C_k} \{ -\langle z_k, u \rangle + \frac{1}{2} \|u\|^2 \} \)
\hspace{1em} \( r_c = 0, r_f = \text{false} \) \( \triangleright \) Restarting accelerated sequence
\hspace{1em} \( r_c = 0, r_f = \text{true} \) \( \triangleright \) Resetting the restart indicators

9: else if \( S_k^{\text{AFW}} \setminus S_{k-1}^{\text{AFW}} \neq \emptyset \) then
\hspace{1em} \( r_f = \text{true} \) \( \triangleright \) If a vertex was added to the active set
\hspace{1em} \( r_f = \text{false} \) \( \triangleright \) Raise freeze flag

10: if \( r_f = \text{true} \) then
11: \( C_k = C_{k-1} \)
12: \( \tilde{x}_k, z_k, w_k = \text{ACC}(\tilde{x}_{k-1}, z_{k-1}, w_{k-1}, a_k, A_k, C_k) \) \( \triangleright \) Run AGD+ uncoupled from CG.
13: else
14: \( C_k = \text{co}(S_k^{\text{AFW}}) \)
15: \( \tilde{x}_k, z_k, w_k = \text{ACC}(x_{k-1}, z_{k-1}, w_{k-1}, a_k, A_k, C_k) \)
16: end if

17: \( r_c = r_c + 1 \) \( \triangleright \) Choose the better step + monotonicity
\( \triangleright \) Increment the restart counter

AFW-sequence whatsoever. Further, in any iteration, the set \( C_k \) that we project onto is the convex hull of some active set \( S_i^{\text{AFW}} \subseteq \mathcal{X} \) for some \( 0 \leq i \leq k - 1 \) implying \( \tilde{x}_k \in \mathcal{X} \) - each \( \tilde{x}_k \) is hence feasible.

Now, as in any iteration \( k \) the solution outputted by the algorithm is \( x_k = \text{argmin}\{f(x_k^{\text{AFW}}), f(\tilde{x}_k)\} \), 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 \( k \geq \frac{\sqrt{r}}{\mu L} (\frac{D}{2})^2 \log \left( \frac{f(x_0) - f(x^*)}{\epsilon} \right) \), it must be that \( f(x_k) - f(x^*) \leq \epsilon \), 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 \( x^* \in \text{co}(S_k^{\text{AFW}}) \) (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 \( x^* \in \text{co}(S_{k-1}^{\text{AFW}}) \), 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 \( x^* \in C_k \). The rest of the proof invokes Lemma 3.2 and its corollary (Corollary B.5), which ensures accelerated convergence following iteration \( K_0 + H \). \( \square \)

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 \( \epsilon_i^m = \frac{\alpha_i}{\pi} \epsilon \) 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 \( C_k \) is smaller than the target accuracy of the projection subproblem (order \( \frac{\sqrt{\mu L}}{\sqrt{\mu L}} \), then \( f \) cannot be reduced by more than order- \( \frac{\sqrt{\mu L}}{\sqrt{\mu L}} \) on \( C_k \), 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 $O(n \log n)$ 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 $\epsilon = 10^{-5} \cdot \left( f(x_0) - f(x^*) \right)$ 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 $f(x) = x^T M^T M x / 2$, $M \in \mathbb{R}^{n \times n}$ 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 $O(n^3)$.

**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 $f(x) = x^T \frac{1}{2} x + b$ was obtained by first generating an orthonormal basis $B = \{ u_1, \cdots, u_n \}$ in $\mathbb{R}^n$ and a set of $n$ uniformly distributed values $\{ \lambda_1, \cdots, \lambda_n \}$ between $\mu$ and $L$ and setting $M = \sum_{i=1}^n \lambda_i u_i u_i^T$. The vector $b$ 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 MIPLIB 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 $\mu$ 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 explicitely 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 $b$ having random integer values of $-1, 0, 1$.

**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
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 $\epsilon$, scaling with $1/\epsilon$, which effectively cancels out any benefits from local acceleration. Further, it would be interesting to understand whether the version of $\mu\text{AGD}^+$ from Lemma 3.2, which allows changing the projection set $C_k$, 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


John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the $\ell_1$-ball for learning in high dimensions. In Proc. NIPS’08, 2008.


