Fast Stochastic Composite Minimization and an Accelerated Frank-Wolfe Algorithm under Parallelization

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider the problem of minimizing the sum of two convex functions. One of those functions has Lipschitz-continuous gradients, and can be accessed via stochastic oracles, whereas the other is "simple". We provide a Bregman-type algorithm with accelerated convergence in function values to a ball containing the minimum. The radius of this ball depends on problem-dependent constants, including the variance of the stochastic oracle. We further show that this algorithmic setup naturally leads to a variant of Frank-Wolfe achieving acceleration under parallelization. More precisely, when minimizing a smooth convex function on a bounded domain, we show that one can achieve an ϵ primal-dual gap (in expectation) in $\tilde{O}(1/\sqrt{\epsilon})$ iterations, by only accessing gradients of the original function and a linear maximization oracle with $O(1/\sqrt{\epsilon})$ computing units in parallel. We illustrate this fast convergence on synthetic numerical experiments.

1 Introduction

We consider the composite minimization problem

$$\min_{y \in \mathbf{V}} \left\{ F(y) := G(y) + H(y) \right\}, \tag{1}$$

where $\mathbf{V}$ is some real vector space equipped with a norm $\|\cdot\|$ and an inner product $\langle \cdot, \cdot \rangle$, see Section 2 for details. For instance, one can consider $\mathbf{V} = \mathbb{R}^d$ or $\mathbf{V} = \mathbb{R}^{p \times q}$ equipped with the standard Euclidean inner product and norm. The function G is convex and β-smooth and the function H is μ-strongly convex, both with respect to (w.r.t.) some norm $\|.\|$ on $\mathbf{V}$ (see Section 2 for precise definitions). We assume that H is "simple", meaning that one can efficiently solve

$$\arg\min_{y \in \mathbf{V}} \left\{ \langle z, y \rangle + H(y) + \alpha D_w(y, z_0) \right\}, \tag{2}$$

where D_w is the Bregman divergence induced by a strongly convex function w. For instance, when w is the standard squared Euclidean norm, this amounts to computing the proximal operator of H. When deterministic gradients of G are available, an accelerated method relying on (2) and achieving rates of the form $O\left(\exp\left(-k\sqrt{\frac{\mu}{\beta}}\right)\right)$ was proposed by Diakonikolas and Guzmán [11, AGD+].

In this work, we are interested in the case where G is only available through a stochastic oracle. In particular, we provide an accelerated algorithm converging in function values to a neighborhood of the minimum with the same rate as above. The size of the neighborhood is of order $O\left(\frac{\sigma^2}{\sqrt{\mu\beta}}\right)$ where

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

σ^2 is the variance of the stochastic gradients. The dependence of the noise term on $1/\sqrt{\mu\beta}$ is similar to that of previous stochastic accelerated methods in simpler settings, see, e.g., [3]. Although our rate is an extension of the result in [11], the parameters are different and tailored for the stochastic setup.

In Section 4, we focus on minimization problems over a compact convex set K for which we have access to a linear optimization oracle, just as in the Frank-Wolfe algorithm. Formally, we consider solving $\min_{x\in K} f(x)$ where f is convex with Lipschitz-continuous gradients. In that case, the dual problem can be seen as a particular instance of (1) where $H := f^*$ is the Fenchel conjugate of f, and G is a smoothed version of the support function of K. Such smoothed functions can only be accessed through a stochastic oracle whose computation boils down to solving a linear optimization problem on K. An appropriate choice of w allows us to use the algorithmic framework by only computing gradients of f and solving linear optimization subproblems over K. In short, minimizing f over the set K with an ϵ primal-dual gap requires $\tilde{O}\left(\max\left\{\frac{1}{\sqrt{\epsilon}}, \frac{1}{\epsilon m}\right\}\right)$ iterations when m computing units can be used in parallel (each of them only solving linear optimization subproblems on K). In particular, when $m = 1/\sqrt{\epsilon}$, this effectively yields an accelerated rate of $\tilde{O}\left(1/\sqrt{\epsilon}\right)$ iterations with a single gradient evaluation per iteration to achieve the required accuracy.

1.1 Related Work

As previously emphasized, the main algorithmic ingredients which inspired this work were developped by [11] and [17]. For further references on acceleration and gradient-based composite convex minimization, we refer to the original works [33, 35] or to a recent survey [13]. For the stochastic setup, we refer to [28]. We refer to [34, 10] for further references and pointers on those topics.

Let us briefly describe the main differences between this work and [11, 17]. First, Gasnikov and Nesterov [17] obtain accelerated rates similar to ours when the strong convexity assumption is placed on G instead of H here (which has practical consequences, including well-posedness of (2) and the size of the constants β and μ, see discussion in [11, Introduction]). When the gradients of G are stochastic, [17] provides convergence rates, but only when the underlying norm is the Euclidean norm. Finally, whereas Diakonikolas and Guzmán [11] consider more general assumptions in the deterministic non-Euclidean case, our work also covers stochastic approximations. We emphasize that each assumption we make (namely accelerated, stochastic, proximal Bregman methods) is necessary to yield acceleration of Frank-Wolfe under parallelization in Section 4.

Two key observations underlying this work are on the one hand the duality link between Frank-Wolfe methods and Bregman methods [4], and on the other hand randomized smoothing techniques [36, 12, 1, 2, 7, 8] which can be naturally computed using linear optimization steps. The main question of interest was whether accelerated Bregman methods should naturally give rise to accelerated Frank-Wolfe methods on the dual.

The Frank-Wolfe (FW) method (a.k.a. conditional gradient method) and its variants were first introduced by Frank and Wolfe [14] (see also [24, 23] for more modern presentations). When considering optimization over a convex set K, classical first-order methods are often naturally embedded with a projection operator (onto K). Depending on K, those projections are potentially costly. An alternate approach for taking constraints into account within first-order method consists in using linear optimization oracles (a.k.a. Frank-Wolfe techniques). In many applications, such linear minimizations are much cheaper than projecting onto the feasible set, see for example [15, 20]. Despite its wide use in practical applications, the main drawback of the Frank-Wolfe method lies in its slow convergence rate of $O(1/\epsilon)$, standing in sharp contrast with the $O(1/\sqrt{\epsilon})$ convergence of Nesterov's accelerated gradient descent [33] relying on projections.

As "purely accelerated" rates of convergence are out of reach for vanilla Frank-Wolfe methods (see lower complexity bound in [29]), most works on accelerating Frank-Wolfe have focused on exploiting specific additional assumptions on the problems at hand. Common such assumptions include strong convexity of the feasible set [9, 30, 16], and strong convexity of the objective function along with the assumption that the minimizer lies in the interior of the feasible set [18, 27]. In both cases,

Frank-Wolfe is known to improve on the $O(1/\epsilon)$ rate. Some efforts have also gone into finding rates matching performances of Nesterov's accelerated gradient descent [33] without strong convexity. In particular, when K is a polytope and when a certain type of constraint qualification is satisfied, Frank-Wolfe converges asymptotically at rate $O(1/k^2)$ [5]. Adding momentum to Frank-Wolfe has also been studied, and accelerated rates are attained on some ℓ_p-norm balls when the minimizer is on the boundary of the feasible set [31]. Our approach in this work is orthogonal to the aforementioned results, in that we do not make additional assumptions on the objective function or the feasible set, but instead show that parallelization can help reaching accelerated rates for a variant of Frank-Wolfe.

The rest of the paper is organized as follows. In the next section we define notations and review some classical definitions from convex analysis. In Section 3 we provide worst-case rates for a stochastic Bregman dual-averaging-type algorithm, along with some intuitions and proof sketches. In Section 4 we show how a Frank-Wolfe method directly fits within this framework, thereby obtaining accelerated worst-case guarantees on the primal-dual gap. In Section 5, we illustrate our theoretical findings on a set of simple numerical examples.

2 Notation and definitions

Formally, we consider a real finite-dimensional vector space $\mathbf{V}$ and its dual space $\mathbf{V}^*$ consisting of all linear functions on $\mathbf{V}$, as well as a dual pairing denoted by $\langle \cdot, \cdot \rangle : \mathbf{V}^* \times \mathbf{V} \to \mathbb{R}$, and a norm $\|\cdot\| : \mathbf{V} \to \mathbb{R}$. We also consider the corresponding dual norm $\|\cdot\|_* : \mathbf{V}^* \to \mathbb{R}$ induced by the choice of $\|\cdot\|$ and $\langle \cdot, \cdot \rangle$ using $\|z\|_* = \sup_{\|x\| \leq 1} \langle z, x \rangle$. For instance, one can use $\mathbf{V} = \mathbf{V}^* = \mathbb{R}^d$ or $\mathbf{V} = \mathbf{V}^* = \mathbb{R}^{p \times q}$ equipped with the standard Euclidean inner product and norm. We insist on the fact that the formal notation $\mathbf{V}$ and $\mathbf{V}^*$ is used mostly for emphasizing differences between primal and dual spaces/problems below. For a closed, proper, convex function $\Psi : \mathbf{V} \to \mathbb{R}$, we denote $\partial \Psi(x)$ the set of all subgradients of Ψ at x. When $\partial \Psi(x)$ is a singleton, we denote its single element by $\nabla \Psi(x)$.

Definition 1. *$\Psi : \mathbf{V} \to \mathbb{R}$ is L-smooth w.r.t. $\|\cdot\|$ if for all $x, y \in \mathbf{V}$,*

$$\|\nabla \Psi(x) - \nabla \Psi(y)\|_* \leq L \|x - y\| . \tag{3}$$

The following proposition will be helpful throughout the paper (see [34] for a proof).

Proposition 1. *$\Psi : \mathbf{V} \to \mathbb{R}$ is L-smooth w.r.t. $\|\cdot\|$ if and only if for all $x, y \in \mathbf{V}$,*

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

Definition 2. *$\Psi : \mathbf{V} \to \mathbb{R}$ is μ-strongly convex w.r.t. $\|\cdot\|$ if for all $x, y \in \mathbf{V}$ and any $g_\Psi(y) \in \partial \Psi(y)$,*

$$\Psi(x) \geq \Psi(y) + \langle g_\Psi(y), x - y \rangle + \frac{\mu}{2} \|x - y\|^2 . \tag{4}$$

Definition 3. *For a closed, proper, convex function $\Psi : \mathbf{V} \to \mathbb{R}$, its conjugate function is defined as*

$$\Psi^*(z) = \sup_{x \in \mathbf{V}} \langle z, x \rangle - \Psi(x). \tag{5}$$

For later references, we need a few results related to smoothing the support function of a set $K \subset \mathbf{V}^*$ (K will appear as a convex set in the dual problem to (1)).

Definition 4. *For a set $K \subset \mathbf{V}^*$, we define I_K as the indicator function of K, i.e., $I_K(x) = 0$ for $x \in K$ and $I_K(x) = \infty$ for $x \in \mathbf{V}^* \setminus K$.*

The support function of K is defined as

$$s(y) = \sup_{x \in K} \langle x, y \rangle. \tag{6}$$

If K is non-empty, convex and closed, the indicator and support functions of K are conjugates of each other, i.e., $(I_K)^* = s$ and $s^* = (I_K)^{**} = I_K$ [37]. The support function is convex yet not differentiable in general applications. For the purpose of our work, we consider smoothing the support function via randomization. Such stochastic smoothing has recently gained popularity in the optimization literature, see for example [36, 12, 1, 2, 7]. We define the main tools and state the relevant properties behind this idea.

Definition 5. *For a set $K \subset \mathbf{V}^*$, a scalar $\alpha > 0$ and a random variable Δ, we define the smoothed support function of K as*

$$s_\alpha(y) = \mathbb{E}_\Delta\left[s(y + \alpha\Delta)\right] = \mathbb{E}_\Delta\left[\sup_{x\in K} \langle x,\, y + \alpha\Delta\rangle\right]. \tag{7}$$

We will use the following proposition on the smoothed support function, proved in [7].

Proposition 2. *Suppose $K \subset \mathbf{V}^*$ is convex compact and let $R_K = \max_{x\in K} \|x\|_*$. Suppose the random variable Δ has positive differentiable density $d\pi(z) \propto \exp(-\eta(z))dz$ for some function $\eta(\cdot)$, and let $M^2 = \mathbb{E}\left[\|\nabla_z \eta(Z)\|_*^2\right]$. Then s_α is convex, $\frac{R_K M}{\alpha}$-smooth w.r.t. $\|\cdot\|$ and for any $y \in \mathbf{V}$,*

$$\nabla s_\alpha(y) = \mathbb{E}_\Delta\left[\arg\max_{x\in K} \langle x,\, y + \alpha\Delta\rangle\right] \quad \textit{and} \quad s(y) \le s_\alpha(y) \le s(y) + \alpha s_1(0).$$

3 Fast Stochastic Composite Minimization

In this section we focus on solving the composite problem

$$\min_{y\in\mathbf{V}} \left\{F(y) := G(y) + H(y)\right\}, \tag{8}$$

where G is convex and β-smooth, and H is μ-strongly convex w.r.t. $\|\cdot\|$. Algorithm 1 summarizes our proposed algorithm, which resembles the general AGD+ algorithm from [11]. Notable differences include access to gradients of G through a stochastic oracle, and explicit dependencies on the smoothness and strong convexity constants for the different updates. Moreover, the presented analysis is different and specifically tailored to handle stochasticity in the gradients of G.

An actual implementation of Algorithm 1 requires the ability to efficiently solve the minimization step (13), which should be well-defined. This intermediate minimization subproblem is often referred to as a Bregman proximal problem, and a sufficient condition for this operation to be well-defined is to require w to be strongly convex. In the case where w is the Euclidean norm, (13) amounts to computing the proximal operator of H. Considering a general regularizer w has several benefits, in that the Euclidean norm might not well capture the geometry of the problem, and because particular choices of w might make (13) easily solvable. The latter will become particularly clear in Section 4.

Before we move on to the analysis, let us emphasize that the first step of Algorithm 1 consisting in picking $z_0 \in \arg\min_y w(y)$ is not restrictive. Indeed one could instead pick any $z_0 \in \mathbf{V}$ and set $\tilde{w}(y) = w(y) - \langle g_w(z_0),\, y\rangle$ where $g_w(z_0) \in \partial w(z_0)$. One could then run Algorithm 1 with the "shifted" $\tilde{w}$ instead of w. Doing so does not change the complexity of the minimization step (13), and the following analysis remains unchanged.

We are now ready to analyse Algorithm 1. For this purpose let us define

$$c_k := \sum_{i=0}^{k-1} (A_{i+1} - A_i)(G(y_i) - \langle \nabla G(y_i),\, y_i\rangle).$$

where the sequences $\{A_k\}_{k\in\mathbb{N}}$ and $\{y_k\}_{k\in\mathbb{N}}$ are defined in Algorithm 1. We also use the notation $\mathbb{E}_k$ for denoting the expectation at iteration k conditioned on the previous iterations (that is, $\mathbb{E}_k$ shortens $\mathbb{E}_k[\,\cdot\,] = \mathbb{E}[\,\cdot \mid y_k,\, z_k,\, d_k,\, c_k]$), while $\mathbb{E}$ denotes the total expectation. Before we can state our results, we need one more assumption on the stochastic gradients.

Assumption 1. *For any $k \in \mathbb{N}$, the stochastic gradients satisfy $\mathbb{E}_k[g_k] = \nabla G(v_k)$ and $\mathbb{E}_k \|g_k - \nabla G(v_k)\|_*^2 \le \sigma^2$.*

The assumption on the variance of the stochastic gradients is common when studying stochastic first-order methods, and allows proving the next proposition, which relates consecutive iterations.

Proposition 3. *Suppose Assumption 1 holds and let $m_k(y) := \langle d_k,\, y\rangle + c_k + A_k H(y) + \beta w(y)$. At iteration k, the iterates of Algorithm 1 satisfy*

$$\mathbb{E}_k\left[A_{k+1}F(y_{k+1}) - m_{k+1}(z_{k+1})\right] \le A_k F(y_k) - m_k(z_k) + (A_{k+1} - A_k)\frac{\sigma^2}{2\sqrt{\mu\beta}}. \tag{15}$$

Algorithm 1 Stochastic Composite Minimization

Input: (β, μ, ν). β-smooth function G, μ-strongly convex function H, ν-strongly convex function w, all w.r.t. the same norm $\|\cdot\|$.
Pick $z_0 \in \arg\min_y w(y)$ and set $y_0 = z_0$.
Set $A_0 = 0$ and $d_0 = 0$.
for $k = 0, 1, \ldots$ **do**

$$A_{k+1} = \frac{A_k(\mu+2\beta+\sqrt{\mu\beta})+\beta\nu+\sqrt{(\beta\nu+\mu A_k)^2+4A_k\beta^2\nu+5A_k^2\mu\beta+2A_k\sqrt{\mu\beta}(\beta\nu+A_k\mu)}}{2(\beta+\sqrt{\mu\beta})} \tag{9}$$

$$\tau_k = 1 - \frac{A_k}{A_{k+1}} \tag{10}$$

$$v_k = (1-\tau_k)y_k + \tau_k z_k \tag{11}$$

Compute a stochastic gradient g_k of function G at iterate v_k,

$$d_{k+1} = d_k + (A_{k+1} - A_k)g_k \tag{12}$$

$$z_{k+1} \in \arg\min_{y\in\mathbf{V}} \{\langle d_{k+1},\, y\rangle + A_{k+1}H(y) + \beta w(y)\} \tag{13}$$

$$y_{k+1} = (1-\tau_k)y_k + \tau_k z_{k+1} \tag{14}$$

end

We defer the full proof to Appendix A, but highlight the main steps here. This way we also hope to shed light on the update of A_{k+1}. The first step is to compute an upper bound on $A_{k+1}G(y_{k+1})$ depending on $A_kG(y_k)$. The second step is similar, and computes an upper bound on $A_{k+1}H(y_{k+1})$ depending on $A_kH(y_k)$. Summing up the two inequalities and taking expectations yields inequality (15) with an additional term in $\|z_{k+1} - z_k\|^2$. To exactly obtain inequality (15) we set A_{k+1} so as to cancel out the coefficient multiplying $\|z_{k+1} - z_k\|^2$. This turns out to be equivalent to setting A_{k+1} as the root of a quadratic polynomial, explaining the form of update (9) in Algorithm 1.

Proposition 3 allows us to get the final rate of convergence of Algorithm 1. We again defer the proof to Appendix A. The idea here is first to unroll the recursion in (15) so as to get a constant term on the right-hand side of the inequality. We then relate $m_N(z_N)$ to $m_N(y_\star)$ and to the minimum of F and finally, we show that $A_{k+1} \geq A_k\left(1 + \frac{\sqrt{\mu}}{2(\sqrt{\beta}+\sqrt{\mu})}\right)$, which gives the exponential decay term.

Theorem 1. *Suppose Assumption 1 holds, and let $y_\star \in \arg\min F$. The convergence rate of Algorithm 1 after k iterations is*

$$\mathbb{E}[F(y_k) - F(y_\star)] \leq \exp\left(-k\frac{\sqrt{\mu}}{2\left(\sqrt{\beta}+\sqrt{\mu}\right)}\right)\beta D_w(y_\star, y_0) + \frac{\sigma^2}{2\sqrt{\mu\beta}}, \tag{16}$$

where $D_w(y_\star, y_0) = w(y_\star) - w(y_0) \geq 0$.

This rate is a typical accelerated rate. In the (usual) case where $\mu \leq \beta$, the exponential decay is bounded above by $\exp\left(-\frac{k}{4}\sqrt{\frac{\mu}{\beta}}\right)$, which shows the natural dependence on $\sqrt{\frac{\mu}{\beta}}$. In addition, the neighborhood term is of the form $O\left(\frac{\sigma^2}{\sqrt{\mu\beta}}\right)$, which is again typical of accelerated stochastic methods [3]. Note that $w(y_\star) - w(y_0)$ is equal to the Bregman divergence $D_w(y_\star, y_0) := w(y_\star) - w(y_0) - \langle g_w(y_0); y_\star - y_0\rangle$ (with $g_w(y_0) \in \partial w(y_0)$) through the choice $g_w(y_0) = 0 \in \partial w(y_0)$, which is valid as y_0 minimizes $w(\cdot)$. We emphasize that we do not require differentiability of the function w anywhere.

In the next section, we show how the above algorithm can be directly applied to a smoothed dual of a minimization problem over a compact convex set, yielding a variant of the Frank-Wolfe algorithm which can achieve accelerated rates under parallelization.

4 Accelerating Frank-Wolfe with parallelization

We consider the following minimization problem over a compact convex set $K \subset \mathbf{V}^*$

$$\min_{x \in \mathbf{V}^*} \left\{ f(x) + I_K(x) \right\}, \tag{17}$$

where f is convex and L-smooth w.r.t. $\|\cdot\|_*$. Its Fenchel-Rockafellar dual [37] reads

$$\max_{y \in \mathbf{V}} \left\{ d(y) := -s(-y) - f^*(y) \right\}. \tag{18}$$

The smoothness of f implies that f^* is $1/L$-strongly convex w.r.t. $\|\cdot\|$ [38, Proposition 12.60]. The term in s is however not smooth w.r.t. $\|\cdot\|$, which prevents us from directly applying results from the previous section. Instead, we choose some smoothing parameter $\alpha > 0$ and, following Definition 5 and Proposition 2, we consider the smoothed minimization problem

$$\min_{y \in \mathbf{V}} \left\{ s_\alpha(-y) + f^*(y) \right\}. \tag{19}$$

Problem (19) fits within the framework of (8) with $G(y) = s_\alpha(-y)$, $H(y) = f^*(y)$, $\mu = 1/L$ and $\beta = \frac{R_K M}{\alpha}$. Using Proposition 2, we see that an unbiased stochastic gradient g of G at some point y can be obtained by sampling Δ and computing $g = -\arg\max_{u \in K} \langle u, -y + \alpha\Delta \rangle$. This boils down to a linear optimization oracle over K, the same oracle as that of the Frank-Wolfe algorithm.

We now show how to pick the distance generating function w such that the minimization step of Algorithm 1 has a closed form. Recall also that z_0 must minimize w. We therefore set $w(y) = f^*(y) - \langle g_{f^*}(z_0), y \rangle$ where $g_{f^*}(z_0) \in \partial f^*(z_0)$. Clearly, w is $1/L$-strongly convex w.r.t. $\|\cdot\|$ and is minimized at z_0. Moreover, first-order optimality conditions for the minimization step (13) of Algorithm 1 read

$$0 \in d_{k+1} + A_{k+1} \partial f^*(z_{k+1}) + \frac{R_K M}{\alpha} \left(\partial f^*(z_{k+1}) - g_{f^*}(z_0) \right) \tag{20}$$

$$\iff \partial f^*(z_{k+1}) \ni \frac{\frac{R_K M}{\alpha}}{A_{k+1} + \frac{R_K M}{\alpha}} g_{f^*}(z_0) - \frac{d_{k+1}}{A_{k+1} + \frac{R_K M}{\alpha}}. \tag{21}$$

Choosing some $x_0 \in K$ and setting $z_k = \nabla f(x_k)$ for all k, one can replace (21) by

$$x_{k+1} = \frac{\frac{R_K M}{\alpha}}{A_{k+1} + \frac{R_K M}{\alpha}} x_0 - \frac{d_{k+1}}{A_{k+1} + \frac{R_K M}{\alpha}}. \tag{22}$$

Doing this allows to avoid computing subgradients of f^*. Instead, whenever the value of z_k is needed, we compute $\nabla f(x_k)$. We summarize this fully primal algorithm in Algorithm 2. Note that similar tricks to obtain "dual-free" methods have already been used, see for example [28, 39, 22].

Observe that each iteration of Algorithm 2 requires the computation of one gradient of f, and m calls to a linear maximization oracle. When $m = 1$, an iteration of Algorithm 2 is as costly as one iteration of Frank-Wolfe. We now show how to further exploit parallelization. To do so, we need to ensure that using multiple samples appropriately improves the quality of the approximation of the true gradient. This the point of the following assumption.

Assumption 2. *There exists a norm-dependent constant $\rho_{\|\cdot\|_*}$ such that the variance verifies*

$$\sigma^2 = \mathbb{E}_k \left\| \frac{1}{m} \sum_{i=1}^{m} g_{k,i} - \nabla G(v_k) \right\|_*^2 \leq \frac{4 R_K^2 \rho_{\|\cdot\|_*}}{m}. \tag{23}$$

Remark 1. *A few examples are in order. For the standard Euclidean norm $\|\cdot\| = \|\cdot\|_* = \|\cdot\|_2$ it holds that $\rho_{\|\cdot\|_2} = 1$. For ℓ_p-norms on a space of dimension d, we have*

$$\rho_{\|\cdot\|_p} = \begin{cases} d^{2/p-1} & \text{for } 1 \leq p < 2 \\ p - 1 & \text{for } 2 \leq p < \infty \\ e^2(\log d + 1) & \text{for } p = \infty. \end{cases} \tag{24}$$

In the general case, as all norms are equivalent in finite dimensional spaces, there exist constants $c, C > 0$ such that $c \|\cdot\|_2 \leq \|\cdot\|_* \leq C \|\cdot\|_2$. Then $\rho_{\|\cdot\|_*} \leq \frac{C^2}{c^2}$. We refer to Appendix B.3 for more details and references.

Algorithm 2 Parallel Frank-Wolfe (PFW)

Input: $(L, x_0, R_K, M, m, \alpha, T)$. L-smooth convex function f w.r.t. $\|\cdot\|_*$, $x_0 \in K$, $R_K = \max_{x\in K} \|x\|_*$, distribution with density $d\pi(z) \propto \exp(-\eta(z))dz$ such that $M^2 = \mathbb{E}_\Delta \|\nabla\eta(\Delta)\|_*^2$, m computing units in parallel, smoothing parameter α, number of iterations T.
Set $A_0 = 0$, $d_0 = 0$, $\beta = \frac{R_K M}{\alpha}$, $\mu = \nu = \frac{1}{L}$, $y_0 = \nabla f(x_0)$.
for $k = 0, 1, \ldots, T-1$ **do**

$$A_{k+1} = \frac{A_k(\mu+2\beta+\sqrt{\mu\beta})+\beta\mu+\sqrt{\mu^2(\beta+A_k)^2+4A_k\beta^2\mu+5A_k^2\mu\beta+2A_k\mu\sqrt{\mu\beta}(\beta+A_k)}}{2(\beta+\sqrt{\mu\beta})}$$
$$\tau_k = 1 - \frac{A_k}{A_{k+1}}$$
$$v_k = (1-\tau_k)y_k + \tau_k \nabla f(x_k)$$

For all $i \in [m]$ in parallel, sample $\Delta_i \sim du$, compute $g_{k,i} = -\arg\max_{u\in K}\langle u, -v_k + \alpha\Delta_i\rangle$

$$g_k = \frac{1}{m}\sum_{i=1}^{m} g_{k,i}$$
$$d_{k+1} = d_k + (A_{k+1} - A_k)g_k$$
$$x_{k+1} = \frac{\beta}{A_{k+1}+\beta}x_0 - \frac{1}{A_{k+1}+\beta}d_{k+1}$$
$$y_{k+1} = (1-\tau_k)y_k + \tau_k\nabla f(x_{k+1})$$

end

As mentioned earlier, one can directly obtain convergence of the dual problem (19) by applying Theorem 1. The next theorem states that not only are the iterates $\{x_k\}_{k\in\mathbb{N}}$ always feasible, but also that the previous rate of convergence also holds for the primal-dual gap between the iterates $\{x_k\}_{k\in\mathbb{N}}$ and $\{y_k\}_{k\in\mathbb{N}}$. For simplicity of the exposition we assume that $\frac{R_K M}{\alpha} \geq \frac{1}{L}$. In practice this is not an issue since we will choose small values of α.

Theorem 2. *Suppose f is convex and L-smooth w.r.t. $\|\cdot\|_*$ and let $x_0 \in K$. Under Algorithm 2, the primal iterates are always feasible, i.e., $x_k \in K$ for all $k \in \mathbb{N}$. Moreover, after k iterations, we have*

$$\mathbb{E}\left[f(x_k) - d(y_k)\right] \leq \exp\left(-k\frac{\sqrt{\alpha}}{4\sqrt{LR_K M}}\right)\frac{R_K M}{\alpha}\left(f(x_0) - f(x_\star)\right) + \frac{2R_K^2\rho_{\|\cdot\|_*}}{m}\sqrt{\frac{\alpha L}{R_K M}} + \alpha s_1(0).$$

The above convergence result is obtained by first plugging the smoothness and strong convexity constants of this section within the result of Proposition 3. Then, one has to relate $m_k(z_k)$ to the primal objective $f(x_k)$. The first term in the upper bound then follows by recursion. The second term comes from the variance, which can be bounded by $\frac{4R_K^2\rho_{\|\cdot\|}}{m}$ when m stochastic gradients are computed in parallel. The last term $\alpha s_1(0)$ is due to the error induced by the smoothing of the dual. The full proof can be found in Appendix B. From Theorem 2 we are able to show that Algorithm 2 achieves acceleration under parallelization.

Theorem 3. *Suppose f is convex and L-smooth w.r.t. $\|\cdot\|_*$ and let $x_0 \in K$. Under Algorithm 2, the number of iterations required to achieve an ϵ primal-dual gap, i.e., $\mathbb{E}[f(x_k) - d(y_k)] \leq \epsilon$, is*

$$k \geq \frac{4\sqrt{LR_K M}}{\sqrt{\alpha}}\log\left(\frac{3R_K M(f(x_0) - f(x_\star))}{\epsilon\alpha}\right), \tag{25}$$

by choosing $\alpha = \min\left\{\frac{\epsilon}{3s_1(0)}, \frac{M\epsilon^2 m^2}{36LR_K^3\rho_{\|\cdot\|_*}^2}\right\}$. The complexity is therefore

$$k \geq \tilde{O}\left(\max\left(\frac{1}{\sqrt{\epsilon}}, \frac{1}{\epsilon m}\right)\right). \tag{26}$$

The complexity result (26) shows that (i) when $m = 1/\sqrt{\epsilon}$, we get an accelerated rate $\tilde{O}(1/\sqrt{\epsilon})$, (ii) when $m = 1$, we recover the classical $\tilde{O}(1/\epsilon)$ complexity of Frank-Wolfe (up to logarithmic terms), (iii) there is no theoretical gain in going beyond $m = 1/\sqrt{\epsilon}$ computing units in parallel.

Moreover, while the total number of calls to the linear optimization oracle to reach an ϵ primal-dual gap is the same as in Frank-Wolfe for any value of m, in the case where $m > 1$, the number of required gradients of f is strictly less than in Frank-Wolfe. This is because in each iteration of Algorithm 2 we compute one gradient of f no matter what the value of m is. In particular, when $m = 1/\sqrt{\epsilon}$, we need only $\tilde{O}(1/\sqrt{\epsilon})$ gradients of f in total, compared to $O(1/\epsilon)$ in Frank-Wolfe.

Finally, note that it might seem like a small modification to Algorithm 2 could give a variant of Frank-Wolfe capable of handling stochastic gradients of f (for previous works on stochastic Frank-Wolfe, see [21, 32] and references therein). Indeed, one could see the noise in $\nabla f(x_k)$ in the update of v_k as the random variable Δ itself and study the resulting algorithm. However, the current analysis would only work if the noise on the stochastic gradients does not depend on the current iterate. Moreover, even under this simplifying assumption, the result does not follow immediately as the stochasticity would appear in both the updates of v_k and y_{k+1}, in contrast with the current analysis.

5 Experiments

We first consider minimizing a quadratic function in $\mathbb{R}^d$ over the simplex

$$\min_{x \in K_1} \{f_1(x) := \frac{1}{2}\|Ax - b\|_2^2\} \quad \text{where} \quad K_1 = \left\{x \in \mathbb{R}^d \mid x \geq 0, \sum_{i=1}^{d} x_i = 1\right\}, \tag{27}$$

where $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$. We choose Δ to have the Gumbel distribution [19] with location and scale parameters equal to 0 and 1 respectively. In this case $R_{K_1} = 1$ and $M = \sqrt{d}$ (see Appendix C for details). We set $n = 200$, $d = 50$ and compare the bound from Theorem 2 with the practical performance of Algorithm 2 for both $m = 1$ and $m = 1/\sqrt{\alpha}$ parallel computer(s). Fig. 1 shows that our upper bound captures well the speed of convergence to a neighborhood.

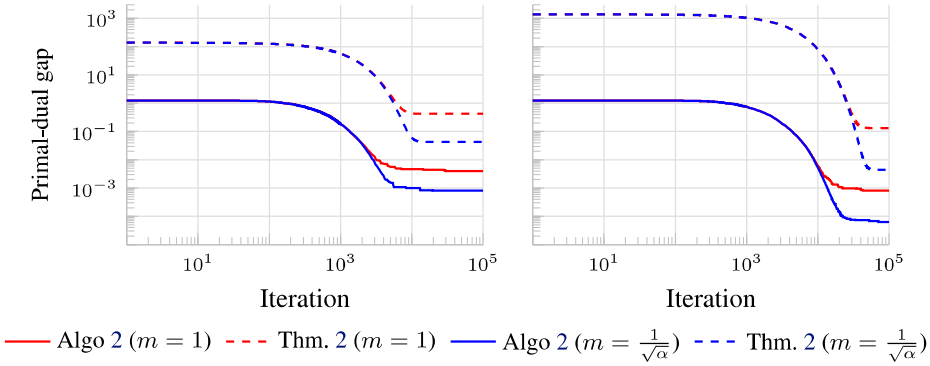

Figure 1: Comparisons between the behavior of Algorithm 2 and that of its theoretical upper bound (see Theorem 2) on a least-squares problem on the simplex for $\alpha = 10^{-2}$ (left) and $\alpha = 10^{-3}$ (right). The plots report the value of the best primal-dual gap incurred at the current iteration.

Next, in order to circumvent the inevitable stalling due to the fixed value of α observed above, we suggest a restarted algorithm which decreases the value of α during training. Starting from $\alpha = 1$ and some $x_0 \in K$, we run Algorithm 2 for T_α iterations with m_α computers in parallel to obtain some approximate solution $x_\alpha \in K$. We then decrease α by a constant factor $c < 1$ and run Algorithm 2 again with the new value of α starting from x_α. We repeat this process until a satisfying solution is obtained. We formalize this procedure in Algorithm 3. In practice we choose $c = 0.5$ and set $T_\alpha = \sqrt{\frac{L}{\alpha}} \log \frac{1}{\alpha}$ as a simplified version of the bound (25), and run the above procedure for both $m_\alpha = 1$ and $m_\alpha = 1/\sqrt{\alpha}$ computers in parallel.

Algorithm 3 Restarted Parallel Frank-Wolfe (R-PFW)

Input: (L, x_0, R_K, M, c). L-smooth convex function f, $x_0 \in K$, $R_K = \max_{x \in K} \|x\|_*$, distribution with density $d\pi(z) \propto \exp(-\eta(z))dz$ such that $M^2 = \mathbb{E}_\Delta \|\nabla \eta(\Delta)\|_*^2$, decreasing factor c.
Set $\alpha = 1$.
for $i = 0, 1, \ldots$ **do**
  Set $T_\alpha = \sqrt{\frac{L}{\alpha}} \log \frac{1}{\alpha}$ and either set $m_\alpha = \frac{1}{\sqrt{\alpha}}$ or $m_\alpha = 1$.
  $x_\alpha =$ Algorithm 2$(L, x_0, R_K, M, m_\alpha, \alpha, T_\alpha)$
  Set $x_0 = x_\alpha$ and $\alpha = c\alpha$.
end
Return x_α.

We test the restarted scheme on problem (27) as well as on a generalized matrix completion problem over the trace norm ball,

$$\min_{X \in K_2} \{f_2(x) := \frac{1}{2} \|CX - D\|_F^2\} \quad \text{where} \quad K_2 = \left\{X \in \mathbb{R}^{p \times q} \mid \|X\|_{tr} \leq 1\right\}, \tag{28}$$

where $C \in \mathbb{R}^{p \times p}$ and $D \in \mathbb{R}^{p \times q}$. Here $\|\cdot\|_F$ stands for the Frobenius norm and $\|\cdot\|_{tr}$ is the trace, or nuclear, norm. We choose Δ to have entries distributed according to a standard normal distribution. In this case $R_{K_2} = 1$ and $M = \sqrt{pq}$. We compare the restarted scheme with the Frank-Wolfe algorithm (with step-size $\eta_k = \frac{2}{k+1}$). We set $p = 10$, $q = 8$ and plot the results in Fig. 2. For both problems, we observed that using $M = 1$ instead of the theoretical value ($M = \sqrt{d}$ for (27) and $M = \sqrt{pq}$ for (28)) led to significant speedups. We consequently plot both strategies. We observe that for $m_\alpha = 1/\sqrt{\alpha}$, Algorithm 3 significantly outperforms Frank-Wolfe, and for $m_\alpha = 1$ the performance is similar to Frank-Wolfe, which was expected from the discussion at the end of Section 4.

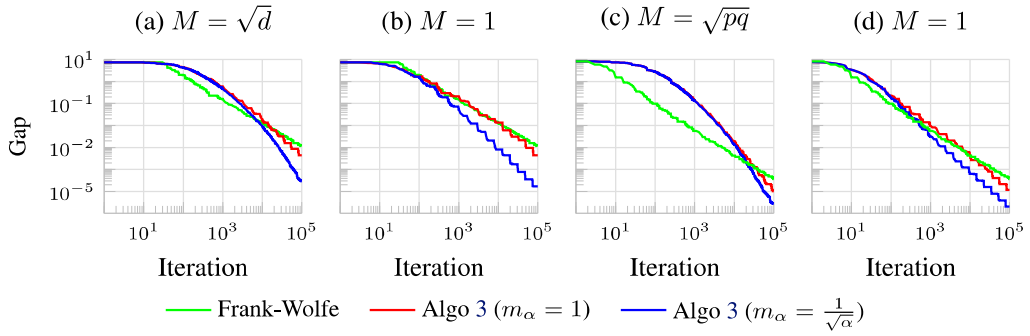

Figure 2: Comparisons between Frank-Wolfe and the restarting scheme (Algorithm 3): a least-squares problem on the simplex ((a) and (b)), and a generalized matrix completion problem on the trace ball ((c) and (d)). The plots report the value of the best primal-dual gap incurred at the current iteration.

6 Conclusion

In this work we introduced a stochastic proximal Bregman algorithm able to converge to a neighborhood of the solution at the same accelerated rate as its deterministic counterpart. We then used it to design a variant of the Frank-Wolfe algorithm able to achieve accelerated rates under parallelization. One drawback is that the resulting algorithm is not "any time", in the sense that the desired precision ϵ must be given as an input to the algorithm. We circumvent this by designing a simple heuristic which slowly decreases the value of ϵ. We plan to investigate the theory for this restarted algorithm in the future. We also hope to use randomized smoothing techniques similar to the ones used in the work to obtain stochastic Frank-Wolfe methods.

References

[1] Jacob Abernethy, Chansoo Lee, Abhinav Sinha, and Ambuj Tewari. Online linear optimization via smoothing. In *Conference on Learning Theory (COLT)*, 2014.

[2] Jacob Abernethy, Chansoo Lee, and Ambuj Tewari. Perturbation techniques in online learning and optimization. *Perturbations, Optimization, and Statistics*, page 223, 2016.

[3] Necdet S. Aybat, Alireza Fallah, Mert Gurbuzbalaban, and Asuman Ozdaglar. A universally optimal multistage accelerated stochastic gradient method. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[4] Francis Bach. Duality between subgradient and conditional gradient methods. *SIAM Journal on Optimization*, 25(1):115–129, 2015.

[5] Francis Bach. On the effectiveness of Richardson extrapolation in data science. *SIAM Journal on Mathematics of Data Science*, 3(4):1251–1277, 2021.

[6] Keith Ball, Eric A Carlen, and Elliott H Lieb. Sharp uniform convexity and smoothness inequalities for trace norms. *Inventiones mathematicae*, 115(1):463–482, 1994.

[7] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable pertubed optimizers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[8] Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and modular implicit differentiation. *arXiv preprint arXiv:2105.15183*, 2021.

[9] Vladimir F. Demianov and Aleksandr M. Rubinov. *Approximate Methods in Optimization Problems*. 1970.

[10] Olivier Devolder. *Exactness, Inexactness and Stochasticity in First-Order Methods for Large-Scale Convex Optimization*. PhD thesis, 2013.

[11] Jelena Diakonikolas and Cristóbal Guzmán. Complementary composite minimization, small gradients in general norms, and applications to regression problems. *arXiv preprint 2101.11041*, 2021.

[12] John C. Duchi, Peter L. Bartlett, and Martin J. Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

[13] Alexandre d'Aspremont, Damien Scieur, and Adrien Taylor. Acceleration methods. *Foundations and Trends® in Optimization*, 5(1-2):1–245, 2021.

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

[15] Robert M. Freund, Paul Grigas, and Rahul Mazumder. An extended Frank–Wolfe method with "in-face" directions, and its application to low-rank matrix completion. *SIAM Journal on optimization*, 27(1):319–346, 2017.

[16] Dan Garber and Elad Hazan. Faster rates for the frank-wolfe method over strongly-convex sets. In *International Conference on Machine Learning*, pages 541–549. PMLR, 2015.

[17] Alexander V. Gasnikov and Yurii Nesterov. Universal method for stochastic composite optimization problems. *Computational Mathematics and Mathematical Physics*, 58(1):48–64, 2018.

[18] Jacques Guélat and Patrice Marcotte. Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35(1):110–119, 1986.

[19] Emil J. Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954.

[20] Zaid Harchaoui, Anatoli Juditsky, and Arkadi Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152(1):75–112, 2015.

[21] Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pages 1263–1271. PMLR, 2016.

[22] Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. Dual-free stochastic decentralized optimization with variance reduction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[23] Martin Jaggi. *Sparse Convex Optimization Methods for Machine Learning*. PhD thesis, ETH Zurich, 2011.

[24] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning (ICML)*, 2013.

[25] Sham Kakade, Shai Shalev-Shwartz, Ambuj Tewari, et al. On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. *Unpublished Manuscript, http://ttic. uchicago. edu/shai/papers/KakadeShalevTewari09. pdf*, 2(1), 2009.

[26] Sham M Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. *Advances in neural information processing systems*, 21, 2008.

[27] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[28] Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1):365–397, 2012.

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

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

[31] Bingcong Li, Mario Coutino, Georgios B. Giannakis, and Geert Leus. How does momentum help Frank Wolfe? *arXiv preprint 2006.11116*, 2020.

[32] Geoffrey Négiar, Gideon Dresdner, Alicia Tsai, Laurent El Ghaoui, Francesco Locatello, Robert Freund, and Fabian Pedregosa. Stochastic frank-wolfe for constrained finite-sum minimization. In *International Conference on Machine Learning*, pages 7253–7262. PMLR, 2020.

[33] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. In *Doklady an USSR*, volume 269, pages 543–547, 1983.

[34] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Science & Business Media, 2003.

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

[36] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[37] Ralph T. Rockafellar. *Convex Analysis*. Princeton University Press, 2015.

[38] Ralph T. Rockafellar and Roger J.-B. Wets. *Variational Analysis*. Springer Science & Business Media, 2009.

[39] Jialei Wang and Lin Xiao. Exploiting strong convexity from data with primal-dual first-order algorithms. In *International Conference on Machine Learning (ICML)*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See conclusion.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See proposition and theorem statements.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code is available in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]