Frank-Wolfe with Subsampling Oracle

Thomas Kerdreux [*1] **Fabian Pedregosa** [*2 3] **Alexandre d'Aspremont** [4 1]

Abstract

We analyze two novel randomized variants of the Frank-Wolfe (FW) or conditional gradient algorithm. While classical FW algorithms require solving a linear minimization problem over the domain at each iteration, the proposed method only requires to solve a linear minimization problem over a small *subset* of the original domain. The first algorithm that we propose is a randomized variant of the original FW algorithm and achieves a $\mathcal{O}(1/t)$ sublinear convergence rate as in the deterministic counterpart. The second algorithm is a randomized variant of the Away-step FW algorithm, and again as its deterministic counterpart, reaches linear (i.e., exponential) convergence rate making it the first provably convergent randomized variant of Away-step FW. In both cases, while subsampling reduces the convergence rate by a constant factor, the cost of the linear minimization step can be a fraction of the deterministic versions, especially when the data is streamed. We illustrate computational gains on regression problems, involving both ℓ_1 and latent group lasso penalties.

1. Introduction

The Frank-Wolfe (FW) or conditional gradient algorithm (Frank & Wolfe, 1956; Jaggi, 2013) is designed to solve optimization problems of the form

$$\underset{\boldsymbol{x}\in\mathcal{M}}{\text{minimize}}\ f(\boldsymbol{x})\ , \text{ with } \mathcal{M} = \text{conv}(\mathcal{A})\ , \qquad \text{(OPT)}$$

where $\mathcal{A}$ is a (possibly infinite) set of vectors which we call *atoms*, and $\text{conv}(\mathcal{A})$ is its convex hull. The FW algorithm and variants have seen an impressive revival in recent years, due to their low memory requirements and projection-free iterations, which make them particularly appropriate to solve large scale convex problems, for instance convex relaxations of problems written over combinatorial polytopes (Zaslavskiy et al., 2009; Joulin et al., 2014; Vogelstein et al., 2015).

The Frank-Wolfe algorithm is projection-free, i.e. unlike most methods to solve (OPT), it does not require to compute a projection onto the feasible set $\mathcal{M}$. Instead, it relies on a linear minimization oracle over a set $\mathcal{A}$, denoted $\text{LMO}(\cdot, \mathcal{A})$, which solves the following linear problem

$$\text{LMO}(\boldsymbol{r}, \mathcal{A}) \in \underset{\boldsymbol{v}\in\mathcal{A}}{\arg\min} \langle \boldsymbol{v}, \boldsymbol{r}\rangle\ . \qquad (1)$$

For some feasible sets such as the nuclear or latent group norm ball (Jaggi et al., 2010; Vinyes & Obozinski, 2017), computing the LMO can be orders of magnitude faster than projecting. Another feature of FW that has greatly contributed to its practical success is its low memory requirements. The algorithm maintains its iterates as a convex combination of a few atoms, enabling the resulting sparse and low rank iterates to be stored efficiently. This feature allows the FW algorithm to be used in situations with a huge or even infinite number of features, such as architecture optimization in neural networks (Ping et al., 2016) or estimation of an infinite-dimensional sparse matrix arising in multi-output polynomial networks (Blondel et al., 2017).

Despite these attractive properties, for problems with a large number of variables or with a very large atomic set (or both), computing the full gradient and LMO at each iteration can become prohibitive. Designing variants of the FW algorithm which alleviate this computational burden would have a significant practical impact on performance.

One recent direction to achieve this is to replace the LMO with a *randomized* linear oracle in which the linear minimization is performed only over a random sample of the original atomic domain. This approach has proven to be highly successful on specific problems such as structured SVMs (Lacoste-Julien et al., 2013) and ℓ_1-constrained regression (Frandi et al., 2016), however little is known in the general case. Is it possible to design a FW variant with a randomized oracle that achieves the same convergence rate (up to a constant factor) as the non-randomized vari-

[*]Equal contribution [1]D.I., UMR 8548, École Normale Supérieure, Paris, France. [2]UC Berkeley, USA. [3]ETH Zurich, Switzerland. [4]CNRS, France.. Correspondence to: Thomas Kerdreux <thomas.kerdreux@inria.fr>, Fabian Pedregosa <f@bianp.net>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

ant? Can this be extended to linearly-convergent FW algorithms (Lacoste-Julien & Jaggi, 2013; 2015; Garber & Hazan, 2015; Pena & Rodriguez, 2015)? In this paper we give a positive answer to both questions and explore the trade-offs between subsampling and convergence rate.

Outline and main contribution. The main contribution of this paper is to develop and analyze two algorithms that share the low memory requirements and projection-free iterations of FW, but in which the LMO is computed only over a random subset of the original domain. In many cases, this results in large gains in computing the LMO which can also speed up the overall FW algorithm. In practice, the algorithm will run a larger number of cheaper iterations, which is typically more efficient for very large data sets (e.g. in a streaming model where the data does not fit in core memory and can only be accessed by chunks). The paper is structured as follows

- §2 describes the "Randomized FW" algorithm, proving a sublinear convergence rate.
- §3 describes "Randomized Away FW" algorithm, a variant which enjoys a linear convergence rate on polytopes. To the best of our knowledge this is the first provably convergent randomized version of the Away-steps FW algorithm.
- Finally, in §4 we discuss implementation aspects of the proposed algorithms and study their performance on lasso and latent group lasso problems.

Note that with the proven sub-linear rate of convergence for Randomized FW, the cost of the LMO is reduced by the subsampling rate, but this is compensated by the fact that the number of iterations required by RFW to reach same convergence guarantee as FW is itself multiplied by the sampling rate. Similarly, the linear convergence rate in Randomized AFW does not theoretically show a computational advantage since the number of iterations is multiplied by the squared sampling rate, in our highly conservative bounds at least. Nevertheless, our numerical experiments show that randomized versions are often numerically superior to their deterministic counterparts.

1.1. Related work

Several references have focused on reducing the cost of computing the linear minimization oracle. The analysis of (Jaggi, 2013) allows for an error term in the LMO, and so a randomized linear oracle could in principle be analyzed under this framework. However, this is not fully satisfactory as it requires the approximation error to decrease towards zero as the algorithm progresses. In our algorithm, the subsampling approximation error doesn't need to decrease.

Lacoste-Julien et al. (2013) studied a randomized FW variant named block-coordinate FW in which at each step the LMO is computed only over a subset (block) of variables. In this case, the approximation error need not decrease to zero, but the method can only be applied to a restricted class of problems: those with block-separable domain, leaving out important cases such as ℓ_1–constrained minimization. Because of the block separability, a more aggressive step-size strategy can be used in this case, resulting overall in a different algorithm.

Finally, Frandi et al. (2014) proposed a FW variant which can be seen as a special case of our Algorithm 1 for the Lasso problem, analyzed in (Frandi et al., 2016). Our analysis here brings three key improvements on this last result. First, it is provably convergent for arbitrary atomic domains, not just the ℓ_1 ball (furthermore the proof in (Frandi et al., 2016) has technical issues discussed in Appendix C). Second, it allows a choice of step size that does not require exact line-search (Variant 2), which is typically only feasible for quadratic loss functions. Third, we extend our analysis to linearly-convergent FW variants such as the Away-step FW.

A different technique to alleviate the cost of the linear oracle was recently proposed by Braun et al. (2017). In that work, the authors propose a FW variant that replaces the LMO by a "weak" separation oracle and showed significant speedups in wall-clock performance on problems such as the video co-localization. This approach was combined with gradient sliding in (Lan et al., 2017), a technique (Lan & Zhou, 2016) that allows skipping the computation of gradients from time to time. However, for problems such as Lasso or latent group lasso, a randomized LMO avoids all full gradient computations, while the lazy weak separation oracle still requires it. Combining these various techniques is an interesting open question.

Proximal coordinate-descent methods (Richtárik & Takáč, 2014) (not based on FW) have also been used to solve problems with a huge number of variables. They are particularly effective when combined with variable screening rules such as (Tibshirani et al., 2012; Fercoq et al., 2015). However, for constrained problems they require evaluating a projection operator, which on some sets such as the latent group lasso ball can be much more expensive than the LMO. Furthermore, these methods require that the projection operator is block-separable, while our method does not.

Notation. We denote vectors with boldface lower case letters (i.e., $\boldsymbol{x}$), and sets in calligraphic letter (i.e., $\mathcal{A}$). We denote $\text{clip}_{[0,1]}(s) = \max\{0, \min\{1, s\}\}$. Probability is denoted $\mathcal{P}$. The cardinality of a set $\mathcal{A}$ is denoted $|\mathcal{A}|$. For $\boldsymbol{x}^*$ a solution of (OPT), we denote $h(\boldsymbol{x}) = f(\boldsymbol{x}) - f(\boldsymbol{x}^*)$.

Randomized vs stochastic. We denote FW variants with

randomness in the LMO *randomized* and reserve the name *stochastic* for FW variants that replace the gradient with a stochastic approximation, as in (Hazan & Luo, 2016).

2. Randomized Frank-Wolfe

In this section we present our first contribution, the Randomized Frank-Wolfe (RFW) algorithm. The method is detailed in Algorithm 1. Compared to the standard FW algorithm, it has the following two distinct features.

First, the LMO is computed over a random subset $\mathcal{A}_t \subseteq \mathcal{A}$ of the original atomic set in which each atom is equally likely to appear, i.e., in which $\mathcal{P}(\boldsymbol{v} \in \mathcal{A}_t) = \eta$ for all $\boldsymbol{v} \in \mathcal{A}$ (Line 3). For discrete sets this can be implemented simply by drawing uniformly at random a fixed number of elements at each iteration. The sampling parameter η controls the fraction of the domain that is considered by the LMO at each iteration. If $\eta = 1$, the LMO considers the full domain at each iteration and the algorithm defaults to the classical FW algorithm. However, for $\eta < 1$, the LMO only needs to consider a fraction of the atoms in the original dataset and can be faster than the FW LMO.

Second, unlike in the FW algorithm, the atom chosen by the LMO is not necessarily a descent direction and so it is no longer possible to use the "oblivious" (i.e., independent on the result of the LMO) $2/(2+t)$ step-size commonly used in the FW algorithm. We provide two possible choices for this step-size: the first variant (Line 6) chooses the step-size by exact line search and requires to solve a 1-dimensional convex optimization problem. This approach is efficient when this sub-problem has a closed form solution, as it happens for example in the case of quadratic loss functions. The second variant does not need to solve this sub-problem, but in exchange requires to have an estimate of the curvature constant C_f (defined in next subsection). Note that in absence of an estimate of this quantity, one can use the bound $C_f \leq \operatorname{diam}(\mathcal{M})^2 L$, where L is the Lipschitz constant of ∇f and $\operatorname{diam}(\mathcal{M})$ is the diameter of the domain in euclidean norm.

Gradient coordinate subsampling. We note that the gradient of f only enters Algorithm 1 through the computation of the randomized LMO, and so only the dot product between the gradient and the subsampled atomic set are truly necessary. In some cases the elements of the atomic set have a specific structure that makes computing dot products particularly effective. For example, when the atomic elements are sparse, only the coordinates of the gradient that are in the support of the atomic set need to be evaluated. As a result, for sparse atomic sets such as the ℓ_1 ball, the group lasso ball (also known as ℓ_1/ℓ_2 ball), or even the latent group lasso (Obozinski et al., 2011) ball, only a few coordinates of the gradient need to be evaluated at each iteration. The number of exact gradients that need to be evaluated will depend on both the sparsity of this atomic set and the subsampling rate. For example, in the case of the ℓ_1 ball, the extreme atoms have a single nonzero coefficient, and so RFW only needs to compute on average $d\eta$ gradient coefficients at each iteration, where d denotes the ambient dimension.

Algorithm 1: Randomized Frank-Wolfe algorithm

1 **Input:** $\boldsymbol{x}_0 \in \mathcal{M}$, sampling ratio $0 < \eta \leq 1$.
2 **for** $t = 0, 1 \ldots, T$ **do**
3 Choose $\mathcal{A}_t$ such that $\mathcal{P}(\boldsymbol{v} \in \mathcal{A}_t) = \eta$ for all $\boldsymbol{v} \in \mathcal{A}$
4 $\boldsymbol{s}_t = \mathrm{LMO}(\nabla f(\boldsymbol{x}_t), \mathcal{A}_t)$
5 **Variant 1** ▷ set γ_t by line-search
6 $\gamma_t = \arg\max_{\gamma \in [0,1]} f((1-\gamma)\boldsymbol{x}_t + \gamma \boldsymbol{s}_t)$
7 **Variant 2**
8 $\gamma_t = \mathrm{clip}_{[0,1]}(\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{x}_t \rangle / C_f)$
9 $\boldsymbol{x}_{t+1} = (1-\gamma_t)\boldsymbol{x}_t + \gamma_t \boldsymbol{s}_t$

Stopping criterion. A side-effect of subsampling the linear oracle is that $\langle -\nabla f(\boldsymbol{x}_t); \boldsymbol{s}_t - \boldsymbol{x}_t \rangle$, where $\boldsymbol{s}_t$ is the atom selected by the randomized linear oracle is no longer an upper bound on $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)$. This property is a feature of FW algorithms that cannot be retrieved in our variant. As a replacement, the stopping criteria that we propose is to compute a full LMO every $k\lfloor \frac{1}{\eta} \rfloor$ iterations, with $k \in \mathbb{N}^*$ ($k = 2$ is a good default value).

2.1. Analysis

In this subsection we prove an $\mathcal{O}(1/t)$ convergence rate for the RFW algorithm. As is often the case for FW-related algorithms, our convergence result will be stated in terms of the *curvature constant* C_f, which is defined as follows for a convex and differentiable function f and a convex and compact domain $\mathcal{M}$:

$$C_f \stackrel{\text{def}}{=} \sup_{\substack{\boldsymbol{x}, \boldsymbol{s} \in \mathcal{M}, \gamma \in [0,1] \\ \boldsymbol{y} = \boldsymbol{x} + \gamma(\boldsymbol{s} - \boldsymbol{x})}} \frac{2}{\gamma^2} \big(f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle \big).$$

It is worth mentioning that a bounded curvature constant C_f corresponds to a Lipschitz assumption on the gradient of f (Jaggi, 2013).

Theorem 2.1. *Let f be a function with bounded smoothness constant C_f and subsampling parameter $\eta \in (0, 1]$. Then Algorithm 1 (in both variants) converges towards a solution of* (OPT). *Furthermore, the following inequality is satisfied:*

$$\mathbb{E}[h(\boldsymbol{x}_T)] \leq \frac{2(C_f + f(\boldsymbol{x}_0) - f(\boldsymbol{x}^*))}{\eta T + 2}. \qquad (2)$$

Proof. See Appendix A. ■

The rate obtained in the previous theorem is similar to known bounds for FW. For example, (Jaggi, 2013, Theorem 1) established for FW a bound of the form

$$h(\boldsymbol{x}_T) \leq \frac{2C_f}{T+2}\,. \tag{3}$$

This is similar to the rate of Theorem 2.1, except for the factor η in the denominator. Hence, if our updates are η times as costly as the full FW update (as is the case e.g. for the ℓ_1 ball), then the theoretical convergence rate is the same. This bound is likely tight, as in the worst case one will need to sample the whole atomic set to decrease the objective if there is only one descent direction. This is however a very pessimistic scenario, and in practice good descent directions can often be found without sampling the whole atomic set. As we will see in the experimental section, despite these conservative bounds, the algorithm often exhibits large computational gains with respect to the deterministic algorithm.

3. Randomized Away-steps Frank-Wolfe

A popular variant of the FW algorithm is the Away-steps FW variant of Guélat & Marcotte (1986). This algorithm adds the option to move away from an atom in the current representation of the iterate. In the case of a polytope domain, it was recently shown to have much better convergence properties, such as linear (i.e. exponential) convergence rates for generally-strongly convex objectives (Garber & Hazan, 2013; Beck & Tetruashvili, 2013; Lacoste-Julien & Jaggi, 2015).

In this section we describe the first provably convergent randomized version of the Away-steps FW, which we name *Randomized Away-steps FW* (RAFW). We will assume throughout this section that the domain is a polytope, i.e. that $\mathcal{M} = \operatorname{conv}(\mathcal{A})$, where $\mathcal{A}$ is a finite set of atoms. We will make use of the following notation.

- *Active set.* We denote by $\mathcal{S}_t$ the active set of the current iterate, i.e. $\boldsymbol{x}_t$ decomposes as $\boldsymbol{x}_t = \sum_{\boldsymbol{v}\in\mathcal{S}_t} \alpha_{\boldsymbol{v}}^{(t)} \boldsymbol{v}$, where $\alpha_{\boldsymbol{v}}^{(t)} > 0$ are positive weights that are iteratively updated.
- *Subsampling parameter.* The method depends on a subsampling parameter p. It controls the amount of computation per iteration of the LMO. In this case, the atomic set is finite and p denotes an integer $1 \leq p \leq |\mathcal{A}|$. This sampling rate is approximately $\lfloor \eta |\mathcal{A}| \rfloor$ in the RFW formulation of §2.

The method is described in Algorithm 2 and, as in the Away-steps FW, requires computing two linear minimization oracles at each iteration. Unlike the deterministic version, the first oracle is computed on the subsampled set $\mathcal{S}_t \cup \mathcal{A}_t$ (Line 3), where $\mathcal{A}_t$ is a subset of size $\min\{p, |\mathcal{A}\backslash\mathcal{S}_t|\}$, sampled uniformly at random from $\mathcal{A}\backslash\mathcal{S}_t$. The second LMO (Line 5) is computed on the active set, which is also typically much smaller than the atomic domain.

Algorithm 2: Randomized Away-steps FW (RAFW)

Input: $\boldsymbol{x}_0 \in \mathcal{M}$, $\boldsymbol{x}_0 = \sum_{\boldsymbol{v}\in\mathcal{A}} \alpha_{\boldsymbol{v}}^{(0)} \boldsymbol{v}$ with $|\mathcal{S}_0| = s$, a subsampling parameter $1 \leq p \leq |\mathcal{A}|$.

1 **for** $t = 0, 1 \ldots, T$ **do**
2 Get $\mathcal{A}_t$ by sampling $\min\{p, |\mathcal{A}\backslash\mathcal{S}_t|\}$ elements uniformly from $\mathcal{A}\backslash\mathcal{S}_t$.
3 Compute $\boldsymbol{s}_t = \mathrm{LMO}(\nabla f(\boldsymbol{x}_t), \mathcal{S}_t \cup \mathcal{A}_t)$
4 Let $\boldsymbol{d}_t^{\mathrm{FW}} = \boldsymbol{s}_t - \boldsymbol{x}_t$ ▷ RFW direction
5 Compute $\boldsymbol{v}_t = \mathrm{LMO}(-\nabla f(\boldsymbol{x}_t), \mathcal{S}_t)$
6 Let $\boldsymbol{d}_t^{A} = \boldsymbol{x}_t - \boldsymbol{v}_t$. ▷ Away direction
7 **if** $\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t^{FW} \rangle \geq \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t^{A} \rangle$ **then**
8 $\boldsymbol{d}_t = \boldsymbol{d}_t^{\mathrm{FW}}$ and $\gamma_{\max} = 1$ ▷ FW step
9 **else**
10 $\boldsymbol{d}_t = \boldsymbol{d}_t^{A}$ and $\gamma_{\max} = \alpha_{\boldsymbol{v}_t}^{(t)} / (1 - \alpha_{\boldsymbol{v}_t}^{(t)})$ ▷ Away step
11 Set γ_t by line-search, with $\gamma_t = \arg\max_{\gamma\in[0,\gamma_{\max}]} f(\boldsymbol{x}_t + \gamma \boldsymbol{d}_t)$
12 Let $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t$ ▷ update $\alpha^{(t+1)}$ (see text)
13 Let $\mathcal{S}_{t+1} = \{\boldsymbol{v} \in \mathcal{A} \text{ s.t. } \alpha_{\boldsymbol{v}}^{(t+1)} > 0\}$

As a result of both oracle calls, we obtain two potential descent directions, the RFW direction $\boldsymbol{d}_t^{\mathrm{FW}}$ and the Away direction $\boldsymbol{d}_t^{\mathrm{A}}$. The chosen direction is the one that correlates the most with the negative gradient, and a maximum step size is chosen to guarantee that the iterates remain feasible (Lines 7–10).

Updating the support. Line 12 requires updating the support and the associated α coefficients. For a FW step we have $\mathcal{S}_{t+1} = \{\boldsymbol{s}_t\}$ if $\gamma_t = 1$ and otherwise $\mathcal{S}_{t+1} = \mathcal{S}_t \cup \{\boldsymbol{s}_t\}$. The corresponding update of the weights is $\alpha_{\boldsymbol{v}}^{(t+1)} = (1-\gamma_t)\alpha_{\boldsymbol{v}}^{(t)}$ when $\boldsymbol{v} \in \mathcal{S}_t \setminus \{\boldsymbol{s}_t\}$ and $\alpha_{\boldsymbol{s}_t}^{(t+1)} = (1-\gamma_t)\alpha_{\boldsymbol{s}_t}^{(t)} + \gamma_t$ otherwise.

For an away step we instead have the following update rule. When $\gamma_t = \gamma_{\max}$ (which is called a *drop step*), then $\mathcal{S}_{t+1} = \mathcal{S}_t \setminus \{\boldsymbol{v}_t\}$. Combined with $\gamma_{\max} < 1$ (or equivalently $\alpha_{\boldsymbol{v}_t} \leq \frac{1}{2}$) we call them *bad drop step*, as it corresponds to a situation in which we are not able to guarantee a geometrical decrease of the dual gap.

For away steps in which $\gamma_t < \gamma_{\max}$, the away atom is not removed from the current representation of the iterate. Hence $\mathcal{S}_{t+1} = \mathcal{S}_t$, $\alpha_{\boldsymbol{v}}^{(t+1)} = (1+\gamma_t)\alpha_{\boldsymbol{v}}^{(t)}$ for $\boldsymbol{v} \in \mathcal{S}_t \setminus \{\boldsymbol{v}_t\}$ and $\alpha_{\boldsymbol{v}_t}^{(t+1)} = (1+\gamma_t)\alpha_{\boldsymbol{v}_t}^{(t)} - \gamma_t$ otherwise.

Note that when choosing Away step in Line 10, it cannot happen that $\alpha_{\boldsymbol{v}_t} = 1$. Indeed this would imply $\boldsymbol{x}_t = \boldsymbol{v}_t$,

and so $\boldsymbol{d}_t^A = 0$. Since we would have $\mathcal{S}_t = \{\boldsymbol{v}_t\}$ and the LMO of Line 3 is performed over $\mathcal{S}_t \cup \mathcal{A}_t$, we necessarily have $\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t^{\text{FW}} \rangle \geq 0$. It thus leads to a choice of FW step, contradiction.

Per iteration cost. Establishing the per iteration cost of this algorithm is not as straightforward as for RFW, as the cost of some operations depends on the size of the active set, which varies throughout the iterations. However, for problems with sparse solutions, we have observed empirically that the size of the active set remains small, making the cost of the second LMO and the comparison of Line 7 negligible compared to the cost of an LMO over the full atomic domain. In this regime, and assuming that the atomic domain has a sparse structure that allows gradient coordinate subsampling, RAFW can achieve a per iteration cost that is, like RFW, roughly $|\mathcal{A}|/p$ times lower than that of its deterministic counterpart.

3.1. Analysis

We now provide a convergence analysis of the Randomized Away-steps FW algorithm. These convergence results are stated in terms of the away curvature constant C_f^A and the geometric strong convexity μ_f^A, which are described in Appendix B and in (Lacoste-Julien & Jaggi, 2015). Throughout this section we assume that f has bounded C_f^A, which is implied by the usual assumption of Lipschitz continuity of the gradient, and strictly positive geometric strong convexity constant μ_f^A, which is verified whenever f is strongly convex and the domain is a polytope.

Theorem 3.1. *Let $\mathcal{M} = \operatorname{conv}(\mathcal{A})$, with $\mathcal{A}$ a finite set of extreme atoms. Then after T iterations of Algorithm 2 (RAFW) we have the following linear convergence rate*

$$\mathbb{E}\big[h(\boldsymbol{x}_{T+1})\big] \leq \big(1 - \eta^2 \rho_f\big)^{\max\{0, \lfloor (T-s)/2 \rfloor\}} h(\boldsymbol{x}_0) \;, \quad (4)$$

with $\rho_f = \frac{\mu_f^A}{4 C_f^A}$, $\eta = \frac{p}{|\mathcal{A}|}$ and $s = |\mathcal{S}_0|$.

Proof. See Appendix B. ■

Proof sketch. Our proof structure roughly follows that of the deterministic case in (Lacoste-Julien & Jaggi, 2015; Beck & Tetruashvili, 2013) with some key differences due to the LMO randomness, and can be decomposed into three parts.

The *first part* consists in upper bounding h_t and is no different from the proof of its deterministic counterpart (Lacoste-Julien & Jaggi, 2015; Beck & Tetruashvili, 2013).

The *second part* consists in lower bounding the progress $h_t - h_{t+1}$. For this algorithm we can guarantee a decrease of the form

$$h_{t+1} \leq h_t \big(1 - \rho_f \big(\frac{g_t}{\tilde{g}_t}\big)^2\big)^{z_t} \;, \quad (5)$$

where $g_t = \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{v}_t \rangle$ is the *partial pair-wise dual gap* while $\tilde{g}_t$ is the *pair-wise dual gap*, in which $\boldsymbol{s}_t$ is replaced by the result of a full (and not subsampled) LMO.

We can guarantee a geometric decrease in expectation on h_t at each iteration, except for bad drop steps, where we can only secure $h_{t+1} \leq h_t$. We mark these by setting $z_t = 0$.

One crucial issue is then to quantify $g_t/\tilde{g}_t$. This can be seen as a measure of the quality of the subsampled oracle: if it selects the same atom as the non-subsampled oracle the quotient will be 1, in all other cases it will be ≤ 1.

To ensure a geometrical decrease we further study the probability of events $z_t = 1$ and $\widetilde{g}_t = g_t$: first, we produce a simple bound on the number of bad drop steps (where $z_t = 0$). Second, when $z_t = 1$ holds, Lemma 3 provides a lower bound on the probability of $g_t = \widetilde{g}_t$.

The *third and last part* of the proof analyzes the expectation of the decrease rate $\prod_{t=0}^{T} (1 - \rho_f \big(\frac{g_t}{\tilde{g}_t}\big)^2)^{\boldsymbol{z}_t}$ given the above discussion. We produce a conservative bound assuming the maximum possible number of bad drop steps. The key element in this part is to make this maximum a function of the size of the support of the initial iterate and of the number of iteration. The convergence bound is then proven by induction. ■

Comparison with deterministic convergence rates. The rate for away Frank-Wolfe in (Lacoste-Julien & Jaggi, 2015, Theorem 8), after T iteration is

$$h(\boldsymbol{x}_{T+1}) \leq \big(1 - \rho_f\big)^{\lfloor T/2 \rfloor} h(\boldsymbol{x}_0) \;. \quad (6)$$

Due to the dependency on η^2 of the convergence rate in Theorem 3.1, our bound does not show that RAFW is computationally more efficient than AFW. Indeed we use a very conservative proof technique in which we measure progress only when the sub-sampling oracle equals the full one. Also, the cost of both LMOs depends on the support of the iterates which is unknown a priori except for a coarse upper bound (e.g. the support cannot be more than the number of iterations). Nevertheless, the numerical results do show speed ups compared to the deterministic method.

Beyond strong convexity. The strongly convex objective assumption may not hold for many problem instances. However, the linear rate easily holds for f of the form $g(\boldsymbol{A}\boldsymbol{x})$ where g is strongly convex and $\boldsymbol{A}$ a linear operator. This type of function is commonly know as a $\tilde{\mu}$-generally strongly convex function (Beck & Tetruashvili, 2013; Wang & Lin, 2014) or (Lacoste-Julien & Jaggi, 2015) (see "Away curvature and geometric strong convexity" in Appendix B for definition). The proof simply adapts that of (Lacoste-Julien & Jaggi, 2015, Th. 11) to our setting.

Theorem 3.2. *Suppose f has bounded smoothness constant C_f^A and is $\tilde{\mu}$-generally-strongly convex. Consider the set $\mathcal{M} = \operatorname{conv}(\mathcal{A})$, with $\mathcal{A}$ a finite set of extreme atoms. Then after T iterations of Algorithm 2, with $s = |\mathcal{S}_0|$ and a p parameter of sub-sampling, we have*

$$\mathbb{E}\big[h(\boldsymbol{x}_{T+1})\big] \leq \big(1 - \eta^2 \tilde{\rho}_f\big)^{\max\{0, \lfloor \frac{T-s}{2} \rfloor\}} h(\boldsymbol{x}_0)\,, \qquad (7)$$

with $\tilde{\rho}_f = \frac{\tilde{\mu}}{4C_f^A}$ and $\eta = \frac{p}{|\mathcal{A}|}$.

Proof. See end of Appendix B. ■

4. Applications

In this section we compare the proposed methods with their deterministic versions. We consider two regularized least squares problems: one with ℓ_1 regularization and another one with latent group lasso (LGL) regularization. In the first case, the domain is a polytope and the analysis of AFW and RAFW holds.

Our results show the FW gap versus number of iterations, and also cumulative number of computed gradient coefficients, which we will label *"nbr coefficients of grad"*. This allows to better reflect the true complexity of our experiments since sub-sampling the LMO in the problems we consider amounts to computing the gradient on a subset of coordinates.

In the case of latent group lasso, we also compared the performance of RFW against FW in terms of wall-clock time on a large dataset stored in disk and accessed sequentially in chunks (i.e., in streaming model).

4.1. Lasso problem

Synthetic dataset. We generate a synthetic dataset following the setting of (Lacoste-Julien & Jaggi, 2015), with a Gaussian design matrix A of size $(200, 500)$ and noisy measurements $\boldsymbol{b} = A\boldsymbol{x}^* + \boldsymbol{\varepsilon}$, with $\boldsymbol{\varepsilon}$ a random Gaussian vector and $\boldsymbol{x}^*$ a vector with 10% of nonzero coefficients and values in $\{-1, +1\}$.

In Figures 1 and 2, we consider a problem of the form (OPT), where the domain is an ℓ_1 ball, a problem often referred to as Lasso. We compare FW against RFW, and AFW against RAFW. The ℓ_1 ball radius set to 40, so that the unconstrained optimum lies outside the domain.

RFW experiments. Figure 1 shows a comparison between FW and RFW. Each call to the randomized LMO outputs a direction, likely less aligned with the opposite of the gradient than the direction proposed by FW, which explains why RFW requires more iterations to converge on the upper left graph of Figure 1. Each call of the randomized LMO is cheaper than the LMO in terms of number of computed coefficients of the gradient, and the trade-off is beneficial as can be seen on the bottom left graph, where RFW outperforms its deterministic variant in terms of *nbr coefficients of grad*.

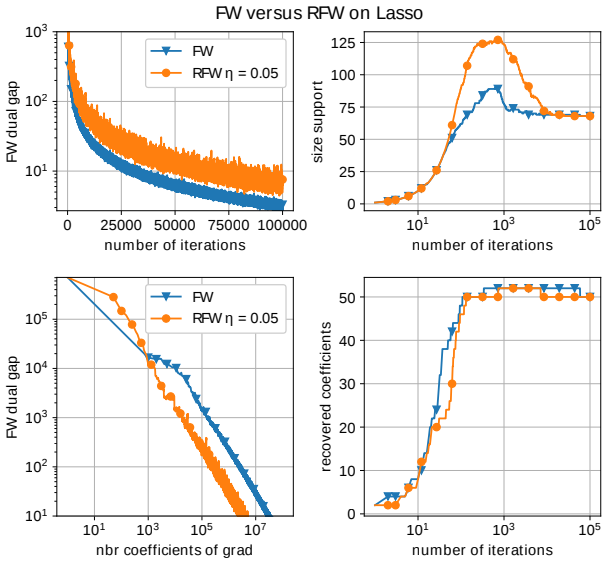

Figure 1. Comparison between FW and RFW with subsampling parameter $\eta = \frac{p}{|\mathcal{A}|} = 0.05$ (chosen arbitrarily) on the lasso problem. *Upper left:* progress in FW dual gap versus number of iterations. *Lower left:* progress of the FW dual gap versus cumulative number of computed coefficients of gradient per call to LMO, called *nbr coefficients of grad* here. *Lower right:* recovered coefficients in support of the ground truth versus number of iterations. *Upper right*: size of support of iterate versus number of iterations.

Finally, the right panels of Figure 1 provide an insight on the evolution of the sparsity of the iterate, depending on the algorithm. FW and RFW perform similarly in terms of the fraction of recovered support (bottom right graph). In terms of the sparsity of the iterate, RFW under-performs FW (upper right graph). This can be explained as follows: because of the sub-sampling, each atom of the randomized LMO provides a direction less aligned with the opposite of the gradient than the one provided by the LMO. Each update in such a direction may result in putting weight on an atom that would better be off the representation of the iterate. It impacts the iterate all along the algorithm as vanilla FW removes past atoms from the representation only by multiplicatively shrinking their weight.

RAFW experiments. Unlike RFW, the RAFW method outperforms AFW in terms on number of iterations in the upper left graph in Figure 2. These graphs also illustrate the linear rate of convergence of both algorithms. The bottom left graph shows that the gap between RAFW and AFW is even larger when comparing the cumulative number of computed coefficients of the gradient required to reach a certain target precision.

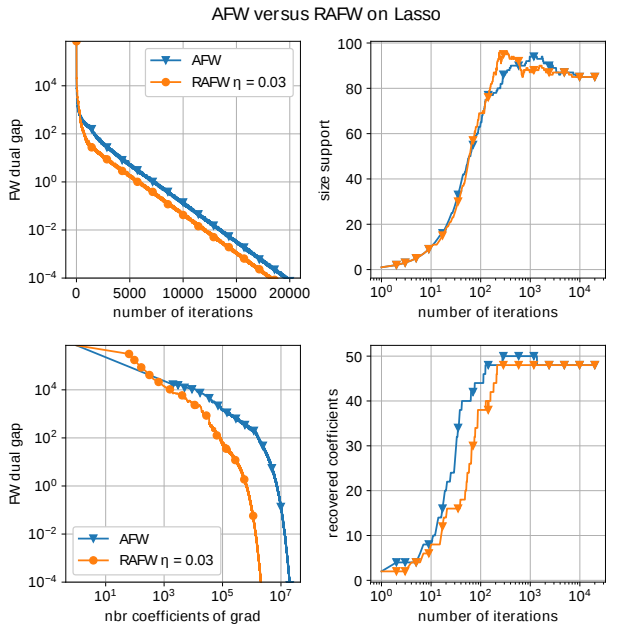

Figure 2. Same parameters and setting as in Figure 1 but to compare RAFW and AFW. AFW performed 880 away steps among which 14 were a drop steps while RAFW performed 1242 away steps and 37 drop steps.

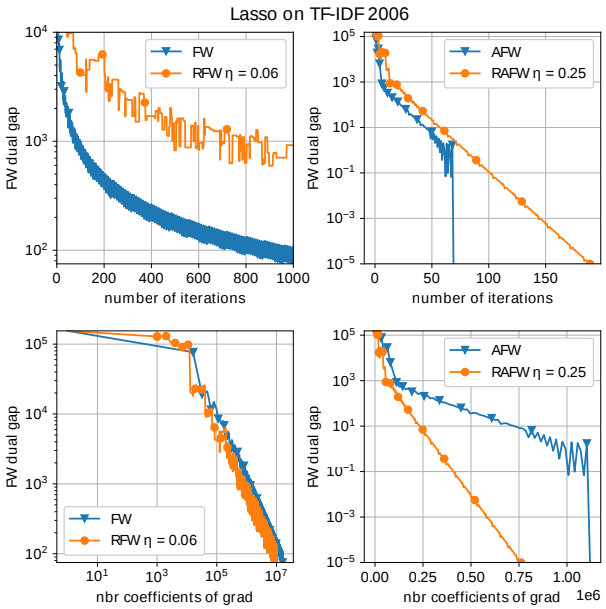

Figure 3. Performance of FW and AFW against RFW and RAFW respectively on the lasso problem with TF-IDF 2006 dataset. The subsampling parameter is $\eta = \frac{p}{|\mathcal{A}|} = 0.06$ (again chosen arbitrarily) for RFW and $\eta = 0.25$ for RAFW. *Right:* Comparison of RAFW against RFW. *Left:* Comparison of RFW against FW. *Upper:* progress in FW dual gap versus number of iterations. *Lower:* progress of the FW dual gap versus cumulative number of computed coefficients in gradient per call to LMO.

This out-performance of RAFW over AFW in term of number of iteration to converge is not predicted by our convergence analysis. We conjecture that the away mechanism improves the trade-off between the cost of the LMO and the alignment of the descent direction with the opposite of the gradient. Indeed, because of the oracle subsampling, the partial FW gap (e.g. the scalar product of the Randomized FW direction with the opposite of the gradient) in RAFW is smaller than in the non randomized variant, and so there is a higher likelihood of performing an away step.

Finally, the away mechanism enables the support of the RAFW to stay close to that of AFW, which was not the case in the comparison of RFW versus FW. This is illustrated in the right panels of Figure 2.

Real dataset. On figure 3, we test the Lasso problem on the E2006-tf-idf data set (Kogan et al., 2009), which gathers volatility of stock returns from companies with financial reports. Each financial reports is then represented through its TF-IDF embedding ($n = 16087$ and $d = 8000$ after an initial round of feature selection). The regularizing parameter is chosen to obtain solution with a fraction of 0.01 nonzero coefficients.

4.2. Latent Group-Lasso

Notation. We denote by $[d]$ the set of indices from 1 to d. For $g \subseteq [d]$ and $\boldsymbol{x} \in \mathbb{R}^d$, we denote by $\boldsymbol{x}_{(g)}$ the projection of $\boldsymbol{x}$ onto the coordinates in g. We use the notation $\nabla_{(g)} f(\boldsymbol{x}_t)$ to denote the gradient with respect to the variables in group g. Similarly $\boldsymbol{x}_{[g]} \in \mathbb{R}^d$ is the vector that equals $\boldsymbol{x}$ in the coordinates of g and 0 elsewhere.

Model. As outlined by Jaggi (2013), FW algorithms are particularly useful when the domain is a ball of the latent group norm (Obozinski et al., 2011). Consider a collection $\mathcal{G}$ of subsets of $[d]$ such that $\bigcup_{g\in\mathcal{G}} g = [d]$ and denote by $||\cdot||_g$ any norm on $\mathbb{R}^{|g|}$. Frank-Wolfe can be used to solve (OPT) with $\mathcal{M}$ being the ball corresponding to the latent group norm

$$||\boldsymbol{x}||_{\mathcal{G}} \overset{\text{def}}{=} \min_{v_{(g)}\in\mathbb{R}^{|g|}} \sum_{g\in\mathcal{G}} ||\boldsymbol{v}_{(g)}||_g \quad s.t.\ \boldsymbol{x} = \textstyle\sum_{v\in\mathcal{G}} \boldsymbol{v}_{[g]}\ . \tag{8}$$

This formulation matches a constrained version of the regularized (Obozinski et al., 2011, equation (5)) when each $||\cdot||_g$ is proportional to the Euclidean norm. For simplicity we will consider $||\cdot||_g$ to be the euclidean norm.

When $\mathcal{G}$ forms a partition of $[d]$ (i.e., there is no overlap between groups), this norm coincides with the group lasso norm.

Sub-sampling. Given $g \in \mathcal{G}$, consider the hyper-disk

$$\mathbb{D}_g(\beta) = \left\{\boldsymbol{v} \in \mathbb{R}^d \ \middle|\ \boldsymbol{v} = \boldsymbol{v}_{[g]}, ||\boldsymbol{v}_{(g)}|| \leq \beta\right\}\ .$$

Obozinski et al. (2011, Lemma 8) shows that such constrain set $\mathcal{M}$ is the convex hull of $\mathcal{A} \overset{\text{def}}{=} \bigcup_{g \in \mathcal{G}} \mathbb{D}_g$.

The RFW can be used to solve this problem, with $\mathcal{A}_t \overset{\text{def}}{=} \bigcup_{g \in \mathcal{G}_p} \mathbb{D}_g$ and where the random oracle is performed over a random subset $\mathcal{G}_p \subseteq \mathcal{G}$ of size p. Denoting by $g_p = \bigcup_{g \in \mathcal{G}_p} g$ the LMO in RFW becomes

$$\text{LMO}(\boldsymbol{x}_t, \mathcal{A}_t) \in \arg\max_{\boldsymbol{v} \in \mathcal{A}_t} \langle \boldsymbol{v}_{(g_p)}, -\nabla_{(g_p)} f(\boldsymbol{x}_t) \rangle \ .$$

With this formulation we only need to compute the gradient on the g_p index. Depending on $\mathcal{G}$ and on the sub-sampling rate, this can be a significant computational benefit.

Experiments. We illustrate the convergence speed-up of using RFW over FW for latent group lasso regularized least square regression.

For $d = 10000$ we consider a collection $\mathcal{G}$ of groups of size 10 with an overlap of 3 and the associated atomic set $\mathcal{A}$. We chose the ground truth parameter vector $\boldsymbol{w}_0 \in \text{conv}(\mathcal{A})$ with a fraction of 0.01 of nonzero coefficients, where on each active group, the coefficients are generated from a Gaussian distribution. The data is a set of n pairs $(y_i, \boldsymbol{w}_i) \in \mathbb{R} \times \mathbb{R}^d$, where $\boldsymbol{w}_i$ is generated from a Gaussian distribution and $y_i = \boldsymbol{w}_i^T \boldsymbol{w}_0 + \varepsilon_i$, where ε_i is again a Guassian random variable. The regularizing parameter is $\beta = 14$, set so that the unconstrained optimum lies outside of the constrain set.

Large dataset and Streaming Model. The design matrix is stored in disk. We allow both RFW and FW to access it only through chunks of size $n \times 500$. This streaming model allows a wall clock comparison of the two methods on very large scale problems.

Computing the gradient when the objective is the least squares loss consists in a matrix vector product. Computing it on a batch of coordinates then requires same operation with a smaller matrix. When computing the gradient at each randomized LMO call, the cost of slicing the design matrix can then compensate the gain in doing a smaller matrix vector product.

With data loaded in memory, which is typically the case for large datasets, both the LMO and the randomized LMO have this access data cost. Consider also that RFW allows any scheme of sampling, including one that minimizes the cost of data retrieval.

5. Conclusion and future work

We give theoretical guarantees of convergence of randomized versions of FW that exhibit same order of convergence as their deterministic counter-parts. As far as we know, for the case of RAFW, this is the first contribution of the kind. While the theoretical complexity bounds don't necessarily imply this, our numerical experiments show that randomized versions often outperform their deterministic ones on ℓ_1-regularized and latent group lasso regularized least squares. In both cases, randomizing the LMO allows us to compute the gradient only on a subset of its coordinates. We use it to speed up the method in a streaming model where the data is accessed by chunks, but there might be other situations where the structure of the polytope can be leveraged to make subsampling computationally beneficial.

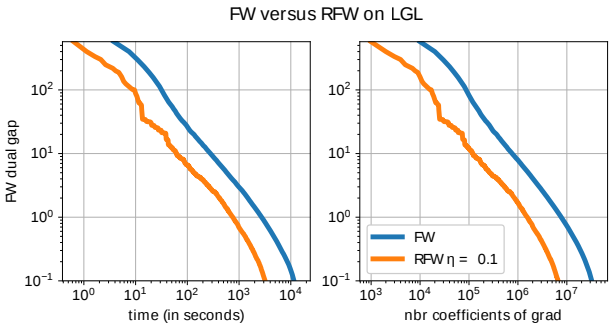

Figure 4. Both panels are in log log scale and show convergence speed up for FW and RFW on latent group lasso regularized least square regression. The parameter of subsampling $\eta = 0.1$, is chosen arbitrarily. *Left:* evolution of the precision in FW dual gap versus the wall clock time. *Right:* evolution of the precision in FW dual gap versus the cumulative number of computed coefficients of the gradient.

There are other linearly-convergent variants of FW besides AFW, such as the Pairwise FW algorithm (Lacoste-Julien & Jaggi, 2015), for which it might be possible to derive randomized variants.

Finally, recent results such as (Goldfarb et al., 2016; 2017; Hazan & Luo, 2016) combine various improvements of FW (away mechanism, sliding, lazy oracles, stochastic FW, etc.). Randomized oracles add to this toolbox and could further improve its benefits.

Acknowledgements

A.A. is at the département d'informatique de l'ENS, École normale supérieure, UMR CNRS 8548, PSL Research University, 75005 Paris, France, and INRIA Sierra project-team. T.K. is a PhD student under the supervision of A.A. and acknowledges funding from the CFM-ENS chaire *les modèles et sciences des données*. FP is funded through the European Union's Horizon 2020 research and innovation programme under the Marie Sklodorowska-Curie grant agreement 748900. The authors would like to thanks Robert Gower, Vincent Roulet and Federico Vaggi for help-

ful discussions.

References

Beck, A. and Tetruashvili, L. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization*, 2013.

Blondel, M., Niculae, V., Otsuka, T., and Ueda, N. Multi-output Polynomial Networks and Factorization Machines. In *Advances in Neural Information Processing Systems 30*. 2017.

Braun, G., Pokutta, S., and Zink, D. Lazifying Conditional Gradient Algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Fercoq, O., Gramfort, A., and Salmon, J. Mind the duality gap: safer rules for the Lasso. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Frandi, E., Ñanculef, R., and Suykens, J. Complexity issues and randomization strategies in Frank-Wolfe algorithms for machine learning. *arXiv preprint arXiv:1410.4062*, 2014.

Frandi, E., Ñanculef, R., Lodi, S., Sartori, C., and Suykens, J. A. K. Fast and scalable Lasso via stochastic Frank–Wolfe methods with a convergence guarantee. *Machine Learning*, 2016.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 1956.

Garber, D. and Hazan, E. A linearly convergent conditional gradient algorithm with applications to online and stochastic optimization. *arXiv preprint arXiv:1301.4666*, 2013.

Garber, D. and Hazan, E. Faster rates for the frank-wolfe method over strongly-convex sets. In *International Conference on Machine Learning*, 2015.

Goldfarb, D., Iyengar, G., and Zhou, C. Semi-Stochastic Frank-Wolfe Algorithms with Away-Steps for Block-Coordinate Structure Problems, 2016.

Goldfarb, D., Iyengar, G., and Zhou, C. Linear Convergence of Stochastic Frank Wolfe Variants. *arXiv preprint arXiv:1703.07269*, 2017.

Guélat, J. and Marcotte, P. Some comments on Wolfe's away step. *Mathematical Programming*, 1986.

Hazan, E. and Luo, H. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pp. 1263–1271, 2016.

Jaggi, M. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. In *International Conference on Machine Learning*, 2013.

Jaggi, M., Sulovsk, M., et al. A simple algorithm for nuclear norm regularized problems. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

Joulin, A., Tang, K., and Fei-Fei, L. Efficient image and video co-localization with frank-wolfe algorithm. In *European Conference on Computer Vision*. Springer, 2014.

Kogan, S., Levin, D., Routledge, B. R., Sagi, J. S., and Smith, N. A. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2009.

Lacoste-Julien, S. and Jaggi, M. An affine invariant linear convergence analysis for Frank-Wolfe algorithms. *arXiv preprint arXiv:1312.7864*, 2013.

Lacoste-Julien, S. and Jaggi, M. On the global linear convergence of Frank-Wolfe optimization variants. In *Advances in Neural Information Processing Systems*, 2015.

Lacoste-Julien, S., Jaggi, M., Schmidt, M., and Pletscher, P. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, 2013.

Lan, G. and Zhou, Y. Conditional gradient sliding for convex optimization. *SIAM Journal on Optimization*, 2016.

Lan, G., Pokutta, S., Zhou, Y., and Zink, D. Conditional Accelerated Lazy Stochastic Gradient Descent. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Locatello, F., Khanna, R., Tschannen, M., and Jaggi, M. A Unified Optimization View on Generalized Matching Pursuit and Frank-Wolfe. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Obozinski, G., Jacob, L., and Vert, J.-P. Group lasso with overlaps: the latent group lasso approach. *arXiv preprint arXiv:1110.0413*, 2011.

Pena, J. and Rodriguez, D. Polytope conditioning and linear convergence of the frank-wolfe algorithm. *arXiv preprint arXiv:1512.06142*, 2015.

Ping, W., Liu, Q., and Ihler, A. T. Learning Infinite RBMs with Frank-Wolfe. In *Advances in Neural Information Processing Systems*, 2016.

Richtárik, P. and Takáč, M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 2014.

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J., and Tibshirani, R. J. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 2012.

Vinyes, M. and Obozinski, G. Fast column generation for atomic norm regularization. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Vogelstein, J. T., Conroy, J. M., Lyzinski, V., Podrazik, L. J., Kratzer, S. G., Harley, E. T., Fishkind, D. E., Vogelstein, R. J., and Priebe, C. E. Fast approximate quadratic programming for graph matching. *PLOS one*, 2015.

Wang, P.-W. and Lin, C.-J. Iteration complexity of feasible descent methods for convex optimization. *Journal of Machine Learning Research*, 2014.

Zaslavskiy, M., Bach, F., and Vert, J.-P. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.