A version of this paper has been published by Society for Industrial and Applied Mathematics.
SIAM Journal on Optimization Vol. 27, No. 1 (2017): 319-346. DOI: 10.1137/15M104726X

An Extended Frank-Wolfe Method with "In-Face" Directions, and its Application to Low-Rank Matrix Completion

Robert M. Freund* Paul Grigas† Rahul Mazumder‡

July 27, 2016

Abstract

Motivated principally by the low-rank matrix completion problem, we present an extension of the Frank-Wolfe Method that is designed to induce near-optimal solutions on low-dimensional faces of the feasible region. This is accomplished by a new approach to generating "in-face" directions at each iteration, as well as through new choice rules for selecting between in-face and "regular" Frank-Wolfe steps. Our framework for generating in-face directions generalizes the notion of away-steps introduced by Wolfe. In particular, the in-face directions always keep the next iterate within the minimal face containing the current iterate. We present computational guarantees for the new method that trade off efficiency in computing near-optimal solutions with upper bounds on the dimension of minimal faces of iterates. We apply the new method to the matrix completion problem, where low-dimensional faces correspond to low-rank matrices. We present computational results that demonstrate the effectiveness of our methodological approach at producing nearly-optimal solutions of very low rank. On both artificial and real datasets, we demonstrate significant speed-ups in computing very low-rank nearly-optimal solutions as compared to the Frank-Wolfe Method (as well as several of its significant variants).

1 Introduction

In the last ten years the problem of *matrix completion* (see, for example, [7, 8, 32]) has emerged as an important and ubiquitous problem in statistics and machine learning, with applications in diverse areas [6, 36], with perhaps the most notable being recommender systems [2, 3, 20]. In matrix completion one is given a partially observed data matrix $X \in \mathbb{R}^{m \times n}$, i.e., there is only knowledge of the entries X_{ij} for $(i, j) \in \Omega$ where $\Omega \subseteq \{1, \ldots, m\} \times \{1, \ldots, n\}$ (often, $|\Omega| \ll m \times n$), and the task is to predict (fill in) the unobserved entries of X. The observed entries are possibly contaminated

*MIT Sloan School of Management, 77 Massachusetts Avenue, Cambridge, MA 02139 (mailto: rfreund@mit.edu). This author's research is supported by AFOSR Grant No. FA9550-15-1-0276, the MIT-Chile-Pontificia Universidad Católica de Chile Seed Fund, and the MIT-Belgium Université Catholique de Louvain Fund.

†MIT Operations Research Center, 77 Massachusetts Avenue, Cambridge, MA 02139 (mailto: pgrigas@mit.edu). This author's research has been partially supported through an NSF Graduate Research Fellowship, the MIT-Chile-Pontificia Universidad Católica de Chile Seed Fund, and the MIT-Belgium Université Catholique de Louvain Fund.

‡MIT Sloan School of Management, 77 Massachusetts Avenue, Cambridge, MA 02139 (mailto: rahulmaz@mit.edu). This author's research was partially supported by ONR Grant N000141512342 and a grant from the Moore Sloan Foundation.

with noise, i.e., $X = Z^* + \mathcal{E}$ where $Z^* \in \mathbb{R}^{m\times n}$ represents the "true data matrix" and $\mathcal{E}$ is the noise term, and the goal is to accurately estimate the entire matrix Z^*, which most importantly includes estimating the entries Z^*_{ij} for $(i,j) \notin \Omega$. Clearly, this problem is in general ill-posed – without any restrictions, the unobserved entries can take on any real values. The ill-posed nature of the problem necessitates that any successful approach must, either explicitly or implicitly, make some type of assumption(s) about underlying *structure* of the matrix Z^*. The most common approach, especially without *a priori* knowledge about the data-generating mechanism, is to assume that the matrix Z^* is *low-rank*. This situation is similar to the "bet on sparsity" principle in linear regression [16]: if Z^* does not have low-rank structure, then we cannot expect any method to successfully fill in the missing entries; on the other hand, if Z^* does have low-rank, then a method that makes such a structural assumption should have a better chance at success.

The low-rank structural assumption naturally leads to the following optimization problem:

$$\begin{aligned} P_r: \qquad \min_{Z\in\mathbb{R}^{m\times n}} \quad & \frac{1}{2}\sum_{(i,j)\in\Omega}(Z_{ij}-X_{ij})^2 \\ \text{s.t.} \quad & \operatorname{rank}(Z) \le r\ , \end{aligned} \tag{1}$$

where r is a parameter representing the assumed belief about the rank of Z^*. Notice that (1) is a combinatorially hard problem due to the rank constraint [9].

Pioneered by [10], a promising strategy for attacking (1) is to use the *nuclear norm* as a proxy for the rank. Recall that, for a given $Z \in \mathbb{R}^{m\times n}$, the sum of the singular values of Z is a norm often referred to as the nuclear norm. Directly replacing the combinatorially hard rank constraint in (1) with a constraint on the nuclear norm of Z leads to the following convex optimization problem:

$$\begin{aligned} NN_\delta: \quad f^* \quad := \qquad \min_{Z\in\mathbb{R}^{m\times n}} \quad & \frac{1}{2}\sum_{(i,j)\in\Omega}(Z_{ij}-X_{ij})^2 \\ \text{s.t.} \quad & \|Z\|_{N1} \le \delta\ . \end{aligned} \tag{2}$$

Let $\mathcal{B}_{N1}(Z,\delta) := \{Y \in \mathbb{R}^{m\times n} : \|Y - Z\|_{N1} \le \delta\}$ denote the nuclear norm ball of radius δ centered at the point Z, so that the feasible region of (2) is $\mathcal{B}_{N1}(0,\delta)$. Despite its apparent absence from the problem formulation, it is nevertheless imperative that computed solutions of (2) have low rank. Such low-rank computed solutions are coerced by the nuclear norm constraint, and there has been substantial and influential work showing that, for many types of data generating mechanisms, an optimal solution of (2) will have appropriately low rank (see, for instance, [7, 8, 10, 31]). This line of work typically focuses on studying the properties of optimal solutions of (2), and thus abstracts away the choice of algorithm to solve (2). Although this abstraction may be reasonable in some situations, and is certainly a reasonable way to study the benefits of nuclear norm regularization, it may also be limiting. Indeed, in recent years, the notion that "convex optimization is a black box" has become increasingly unreasonable. Concurrently with the explosion of "big data" applications, there has been a substantial amount of recent work on the development and analysis of algorithms for huge-scale convex optimization problems where interior point methods and other polynomial-time algorithms are ineffective. Moreover, there has been an increasing interest in algorithms that directly promote desirable structural properties of their iterates. One such algorithm that satisfies

both of these properties – scalability to huge-size problems and structurally favorable iterates – is the Frank-Wolfe Method and its extensions, which is the starting point of the work herein. Indeed, much of the recent computational work for matrix completion is based on directly applying first-order methods and related methods that have structurally favorable iterates [5, 19, 24, 35]. Mazumder et al. [25] develop a related algorithm based on SVD soft thresholding that efficiently utilizes the special structure of matrix completion problems. In one of the earlier works applying the Frank-Wolfe Method to nuclear norm regularized problems, Jaggi and Sulovský [18] consider first lifting the nuclear norm regularized problem (2) to a problem over the semidefinite cone and then apply the Frank-Wolfe Method. Tewari et al. [34] as well as Harchaoui et al. [14] pointed out that the Frank-Wolfe Method can be applied directly to the nuclear norm regularized problem (2), and [14] also developed a variant of the method that applies to penalized nuclear norm problems, which was also studied in [35]. Mishra et al. [26] develop a second-order trust region method that shares a few curious similarities with the extended Frank-Wolfe Method developed herein. Mu et al. [27] consider a hybrid proximal gradient/Frank-Wolfe method for low-rank matrix and tensor recovery. Rao et al. [30] consider a variant of Frank-Wolfe with "backward steps" (which differ from the classical "away steps" of Wolfe [38] and Guélat and Marcotte [13]) in the general context of atomic norm regularization. Backward steps comprise a flexible methodology aimed at producing sparse representations of solutions. In this regard, backward steps are unrelated to away steps except to the extent that both may result in sparse solutions.

The Frank-Wolfe Method, in-face directions, and structural implications. Due to its low iteration cost and convenient structural properties (as we shall soon discuss), the Frank-Wolfe Method (also called the conditional gradient method) is especially applicable in several areas of machine learning and has thus received much renewed interest in recent years, see [12, 15, 17, 23, 34] and the references therein. The Frank-Wolfe Method, originally developed by [11] in the context of quadratic programming, was later generalized to convex optimization problems with smooth (differentiable) convex objective functions and bounded convex feasible regions, of which (2) is a particular instance. Indeed, letting $f(Z) := \frac{1}{2}\sum_{(i,j)\in\Omega}(Z_{ij} - X_{ij})^2$ denote the least squares objective in (2), it is easy to see that $f(\cdot)$ is a smooth convex function, and the feasible region of (2) is $\mathcal{B}_{N1}(0,\delta)$, which is a bounded convex set.

As applied to problem (2), the Frank-Wolfe Method proceeds at the current iterate Z^k by solving a linear optimization subproblem to compute $\tilde{Z}^k \leftarrow \arg\min_{Z\in\mathcal{B}_{N1}(0,\delta)} \left\{\nabla f(Z^k) \bullet Z\right\}$ (here "$\bullet$" denotes the usual trace inner product) and updates the next iterate as

$$Z^{k+1} \leftarrow Z^k + \bar{\alpha}_k(\tilde{Z}^k - Z^k) \ , \tag{3}$$

for some $\bar{\alpha}_k \in [0,1]$. It can be shown, see for instance [12, 15, 17] and as we expand upon in Section 2, that for appropriate choices of the step-size sequence $\{\bar{\alpha}_k\}$ it holds that

$$f(Z^k) - f^* \leq \frac{8\delta^2}{k+3} \quad \text{and} \quad \text{rank}(Z^k) \leq k+1 \ . \tag{4}$$

The bound on the objective function gap in (4) is well understood and follows from a standard analysis of the Frank-Wolfe Method. The bound on the rank of Z^k in (4), while also well understood, follows from the special structure of the nuclear norm ball. Specifically, and as we further expand upon in Sections 2 and 3, for the nuclear norm regularized matrix completion problem (2), the solutions to the linear optimization subproblem solved at each iteration are specially structured –

they are rank one matrices arising from the leading left and right singular vectors of the matrix $\nabla f(Z^k)$. Thus, assuming that Z^0 is a rank one matrix, the simple additive form of the updates (3) leads to the bound on the rank in (4). The above bound on the rank of Z^k is precisely the "favorable structural property" of the iterates of the Frank-Wolfe Method that was mentioned earlier, and when combined with the bound on the objective function gap in (4) yields a nice tradeoff between data fidelity and low-rank structure. However, note that when k is large – as might be necessary if the desired objective function value gap needs to be very small – then the bound on the rank of Z^k might not be as favorable as one might wish. Indeed, one of the primary motivations underlying the research herein is to develop theoretical and practical methods for solving (2) that simultaneously achieve both good data fidelity (i.e., a small optimality gap in (2)) and low rank of the iterates Z^k.

Here we see that in the case of the Frank-Wolfe Method, the *properties of the algorithm* provide additional insight into how problem (2) induces low-rank structure. A natural question is: can the tradeoff given by (4) be improved, either theoretically or practically or both? That is, can we modify the Frank-Wolfe Method in a way that maintains the bound on the objective function gap in (4) while strictly improving the bound on the rank? This is the motivation for the development of what we call "in-face" directions and their subsequent analysis herein. We define an in-face direction to be any descent direction that keeps the next iterate within the minimal face of $\mathcal{B}_{N1}(0, \delta)$ containing the current iterate (where the minimal face of a point $x \in S$ is the smallest face of the convex set S that contains the point x). It turns out that the faces of the nuclear norm ball are characterized by the (thin) SVDs of the matrices contained within them [33]. Therefore an in-face direction will move to a new point Z^{k+1} with a similar SVD structure as Z^k, and moreover will keep the rank of Z^{k+1} the same (or will lower it, which is even better), i.e., $\text{rank}(Z^{k+1}) \leq \text{rank}(Z^k)$. Clearly if we can find good in-face directions, then the bound on the rank in (4) will be improved. At the same time, if there are no in-face directions that are "good enough" with respect to improvements in objective function values, then a "regular" Frank-Wolfe direction may be chosen, which will usually increase the rank of the next iterate by one. In this paper, we develop an extension of the Frank-Wolfe Method that incorporates in-face directions and we provide both a precise theoretical analysis of the resulting tradeoff akin to (4), as well as computational results that demonstrate significant improvements over existing methods both in terms of ranks and run times.

1.1 Organization/Results

The paper is organized as follows. In Section 2, after reviewing the basic Frank-Wolfe Method and the away-step modification of Wolfe and Guélat and Marcotte, we present our extended Frank-Wolfe Method based on "in-face" directions (in addition to regular Frank-Wolfe directions), this being the main methodological contribution of the paper. This In-Face Extended Frank-Wolfe Method is specifically designed to induce iterates that lie on low-dimensional faces of the feasible set S, since low-dimensional faces of the feasible region contain desirable "well-structured" points (sparse solutions when S is the ℓ_1 ball, low-rank matrices when S is the nuclear norm ball). The in-face directions are any directions that keep the current iterate in its current minimal face of S. We present two main strategies for computing in-face directions: *(i)* away-steps as introduced by Wolfe [38] and Guélat and Marcotte [13], and *(ii)* approximate full optimization of the objective $f(\cdot)$ over the current minimal face. The In-Face Extended Frank-Wolfe Method uses a simple decision criterion for selecting between in-face and regular Frank-Wolfe directions. In Theorem 2 we

present computational guarantees for the In-Face Extended Frank-Wolfe Method. These guarantees essentially show that the In-Face Extended Frank-Wolfe Method maintains $O(c/k)$ convergence after k iterations (which is optimal for Frank-Wolfe type methods in the absence of polyhedral structure or strong convexity [23]), all the while promoting low-rank iterates via the parameters of the method which affect the constant c above, see Theorem 2 for specific details.

In Section 3 we discuss in detail how to apply the In-Face Extended Frank-Wolfe Method to solve the matrix completion problem (2). We resolve issues such as characterizing and working with the minimal faces of the nuclear norm ball, solving linear optimization subproblems on the nuclear norm ball and its faces, computing steps to the boundary of the nuclear norm ball, and updating the SVD of the iterates. In Proposition 2 we present a bound on the ranks of the matrix iterates of the In-Face Extended Frank-Wolfe Method that specifies how the in-face directions reduce the rank of the iterates over the course of the algorithm. Furthermore, as a consequence of our developments we also demonstrate, for the first time, how to effectively apply the away-step method of [13] to problem (2) in a manner that works with the natural parameterization of variables $Z \in \mathbb{R}^{m\times n}$ (as opposed to an "atomic" form of [13], as we expand upon at the end of Section 2.1).

Section 4 contains a detailed computational evaluation of the In-Face Extended Frank-Wolfe Method and discusses several versions of the method based on different strategies for computing in-face directions and different algorithmic parameter settings. We compare these versions to the regular Frank-Wolfe Method, the away-step method of [13], an atomic version of [13] (as studied in [1, 21, 22, 28]), as well as the "fully corrective" variant of Frank-Wolfe [15, 17, 22] and the CoGEnT "forward-backward" method of [30]. We present several experiments on simulated problem instances as well as on the MovieLens10M dataset. Our results demonstrate that the In-Face Extended Frank-Wolfe Method (in different versions) shows significant computational advantages in terms of delivering low rank and low run time to compute a target optimality gap. Especially for larger instances, one version of our method delivers very low rank solutions with reasonable run times, while another version delivers the best run times, beating existing methods by a factor of 10 or more.

1.2 Notation

Let E be a finite-dimensional linear space. For a norm $\|\cdot\|$ on E, let $\|\cdot\|^*$ be the associated dual norm, namely $\|c\|^* := \max\{c^T z : \|z\| \le 1\}$ and $c^T z$ denotes the value of the linear operator c acting on z. The ball of radius δ centered at $\bar{z}$ is denoted $\mathcal{B}(\bar{z}, \delta) := \{z : \|z - \bar{z}\| \le \delta\}$. We use I to denote the identity matrix whose dimension is dictated by the context. For $X, Y \in \mathbb{S}^{k\times k}$ (the set of $k \times k$ symmetric matrices), we write "$X \succeq 0$" to denote that X is symmetric and positive semidefinite, "$X \succeq Y$" to denote that $X - Y \succeq 0$, and "$X \succ 0$" to denote that X is positive definite, etc. For a given $Z \in \mathbb{R}^{m\times n}$ with $r := \text{rank}(Z)$, the (thin) singular value decomposition (SVD) of Z is $Z = UDV^T$ where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ are each orthonormal ($U^TU = I$ and $V^TV = I$), and $D = \text{Diag}(\sigma_1, \ldots, \sigma_r)$ comprises the non-zero (hence positive) singular values of Z. The nuclear norm of Z is then defined to be $\|Z\|_{N1} := \sum_{j=1}^r \sigma_j$. (In much of the literature, this norm is denoted $\|\cdot\|_*$; we prefer to limit the use of "$*$" to dual norms, and hence we use the notation $\|\cdot\|_{N1}$ instead.) Let $\mathcal{B}_{N1}(Z, \delta) := \{Y \in \mathbb{R}^{m\times n} : \|Y - Z\|_{N1} \le \delta\}$ denote the nuclear norm ball of radius δ centered at the point Z. Let $\|Z\|_F$ denote the Frobenius norm of Z, namely $\|Z\|_F = \sqrt{\sum_{j=1}^r \sigma_j^2} = \sqrt{\text{Tr}(Z^TZ)}$. The dual norm of the nuclear norm is

the largest singular value of a matrix and is denoted by $\|\cdot\|_{N1}^* = \|\cdot\|_{N\infty}$; given $S \in \mathbb{R}^{m\times n}$ with SVD $S = UDV^T$, then $\|S\|_{N\infty} = \max\{\sigma_1, \ldots, \sigma_r\}$. A spectrahedron is a set of the form $\mathcal{S}_t^k := \{X \in \mathbb{S}^{k\times k} : X \succeq 0,\ I \bullet X \leq t\}$ or $\bar{\mathcal{S}}_t^k := \{X \in \mathbb{S}^{k\times k} : X \succeq 0,\ I \bullet X = t\}$, where "$\bullet$" denotes the usual trace inner product.

2 Frank-Wolfe Method, Away Steps, and In-Face Steps

Problem (2) is an instance of the more general problem:

$$f^* \ := \ \min_{x\in S} \ f(x) \tag{5}$$

where $S \subset E$ is a closed and bounded convex set, and $f(\cdot)$ is a differentiable convex function on S. We first review solving instances of (5) using the Frank-Wolfe Method, whose basic description is given in Algorithm 1.

Algorithm 1 Frank-Wolfe Method for optimization problem (5)

Initialize at $x_0 \in S$, (optional) initial lower bound B_{-1}, $k \leftarrow 0$.

At iteration k:

1. Compute $\nabla f(x_k)$.
2. Compute $\tilde{x}_k \leftarrow \arg\min_{x\in S}\{f(x_k) + \nabla f(x_k)^T(x - x_k)\}$.
   $B_k^w \leftarrow f(x_k) + \nabla f(x_k)^T(\tilde{x}_k - x_k)$.
   Update best bound: $B_k \leftarrow \max\{B_{k-1}, B_k^w\}$.
3. Set $x_{k+1} \leftarrow x_k + \bar{\alpha}_k(\tilde{x}_k - x_k)$, where $\bar{\alpha}_k \in [0, 1]$.

Typically the main computational burden at each iteration of the Frank-Wolfe Method is solving the linear optimization subproblem in Step (2.) of Algorithm 1. The quantities B_k^w are lower bounds on the optimal objective function value f^* of (5), a fact which follows easily from the gradient inequality, see Jaggi [17] or [12], and hence $B_k = \max\{B_{-1}, B_0^w, \ldots, B_k^w\}$ is also a lower bound on f^*. The lower bound sequence $\{B_k\}$ can be used in a variety of step-size strategies [12] in addition to being useful in termination criteria.

When the step-size sequence $\{\bar{\alpha}_k\}$ is chosen using the simple rule $\bar{\alpha}_k := \frac{2}{k+2}$, then the Frank-Wolfe Method has the following computational guarantee at the k^{th} iteration, for $k \geq 0$:

$$f(x_k) - f^* \leq f(x_k) - B_k \leq \frac{2LD^2}{k+3} \ , \tag{6}$$

where $D := \max_{x,y\in S} \|x - y\|$ is the diameter of S, and L is a Lipschitz constant of the gradient of $f(\cdot)$ on S, namely:

$$\|\nabla f(x) - \nabla f(y)\|_* \leq L\|x - y\| \quad \text{for all } x, y \in S \ . \tag{7}$$

If $\bar{\alpha}_k$ is instead chosen by exact line-search, namely $\bar{\alpha}_k \leftarrow \arg\min_{\alpha\in[0,1]} f(x_k + \alpha(\tilde{x}_k - x_k))$, then the guarantee (6) still holds, see Section 3.4 of [12], this being particularly relevant when $f(\cdot)$ is a convex quadratic as in (2) in which case the exact line-search reduces to a simple formula. Alternatively,

one can consider a step-size rule based on minimizing an upper-approximation of $f(\cdot)$ inherent from the smoothness of the gradient, namely:

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \tfrac{L}{2}\|y-x\|^2 \quad \text{for all } x, y \in S \ , \tag{8}$$

which follows from (7) (see [12], for example, for a concise proof). The following is a modest extension of the original analysis of Frank and Wolfe in [11].

Theorem 1. (extension of [11]) *Let $\bar{L} \geq L$ be given, and consider using either an exact line-search or the following step-size rule for the Frank-Wolfe Method:*

$$\bar{\alpha}_k \leftarrow \min\left\{ \frac{\nabla f(x_k)^T(x_k - \tilde{x}_k)}{\bar{L}\|x_k - \tilde{x}_k\|^2}, 1 \right\} \ \textit{for all } k \geq 0 \ . \tag{9}$$

Then $f(x_k)$ is monotone decreasing in k, and it holds that:

$$f(x_k) - f^* \leq f(x_k) - B_k \leq \frac{1}{\frac{1}{f(x_0)-B_0} + \frac{k}{2\bar{L}D^2}} \quad < \quad \frac{2\bar{L}D^2}{k} \ . \tag{10}$$

Proof: The first inequality of (10) follows from the fact that $B_k \leq f^*$, and the third inequality follows from the fact that $f(x^0) \geq f^* \geq B_0$. The second inequality can be rewritten as:

$$\frac{1}{f(x_k) - B_k} \geq \frac{1}{f(x_0) - B_0} + \frac{k}{2\bar{L}D^2} \ ,$$

which states that the reciprocal of the optimality bound gap grows at least according to the indicated linear function in k. The above inequality holds trivially for $k = 0$, and hence to prove the second inequality of (10) it suffices to show that:

$$\frac{1}{f(x_{k+1}) - B_{k+1}} \geq \frac{1}{f(x_k) - B_k} + \frac{1}{2\bar{L}D^2} \quad \text{for all } k \geq 0 \ , \tag{11}$$

whose proof is given in Appendix A, and wherein the monotonicity of $f(x_k)$ is also proved. □

In addition to being the crux of the proof of (10), we will also use inequality (11) and related inequalities as the basis for choosing among candidate directions in the in-face extension of Frank-Wolfe that we will develop in Section 2.2.

2.1 Away Steps

In [38] Wolfe introduced the concept of an "away step" in a modified version of the Frank-Wolfe method, and Guélat and Marcotte [13] provided a modification thereof and an extensive treatment of the convergence properties of the away-step-modified Frank-Wolfe method, including eventual linear convergence of the method when the objective function is strongly convex, the feasible region is polyhedral, and a form of strict complementarity holds. Quite recently there has been much renewed interest in the Frank-Wolfe method with away steps, with most of the focus being on demonstrating global linear convergence with computational guarantees for a particular "atomic" version of [13], see Lacoste-Julien and Jaggi [21, 22], Beck and Shtern [1], and Peña et al. [28].

Algorithm 2 presents the modified Frank-Wolfe Method with Away Steps as developed in [13]. The algorithm needs to work with the minimal face of a point $x \in S$, which is the smallest face of S that contains the point x; here we use the notation $\mathcal{F}_S(x)$ to denote the minimal face of S which contains x. Step (2.) of the modified Frank-Wolfe method is the "away step" computation, where $\check{x}_k$ is the point on the current minimal face $\mathcal{F}_S(x_k)$ that is farthest along the ray from the "bad" solution $\hat{x}_k$ through the current point x_k. Step (3.) of the modified method is the regular Frank-Wolfe step computation, which is called a "toward step" in [13]. (Please see [13] as well as [38] for an expanded exposition of away-steps, including illustrative figures.) Notice that implementation of the away-step modified Frank-Wolfe method depends on the ability to characterize and work with the minimal face $\mathcal{F}_S(x_k)$ of the iterate x_k. When S is not a polytope this minimal face capability is very much dependent on problem-specific knowledge of the structure of the set S.

Algorithm 2 Modified Frank-Wolfe Method with Away Steps, for optimization problem (5)

Initialize at $x_0 \in S$, (optional) initial lower bound B_{-1}, $k \leftarrow 0$.

At iteration k:

1. Compute $\nabla f(x_k)$.
2. Compute $\hat{x}_k \leftarrow \arg\max_x \{\nabla f(x_k)^T x : x \in \mathcal{F}_S(x_k)\}$.
   $\alpha_k^{\text{stop}} \leftarrow \arg\max_\alpha \{\alpha : x_k + \alpha(x_k - \hat{x}_k) \in \mathcal{F}_S(x_k)\}$.
   $\check{x}_k \leftarrow x_k + \alpha_k^{\text{stop}}(x_k - \hat{x}_k)$.
3. Compute $\tilde{x}_k \leftarrow \arg\min_x \{\nabla f(x_k)^T x : x \in S\}$.
   $B_k^w \leftarrow f(x_k) + \nabla f(x_k)^T(\tilde{x}_k - x_k)$.
   Update best bound: $B_k \leftarrow \max\{B_{k-1}, B_k^w\}$.
4. Choose descent direction:
   If $\nabla f(x_k)^T(\tilde{x}_k - x_k) \leq \nabla f(x_k)^T(x_k - \hat{x}_k)$, then $d_k \leftarrow \tilde{x}_k - x_k$ and $\bar{\beta}_k \leftarrow 1$;
   Else $d_k \leftarrow x_k - \hat{x}_k$ and $\bar{\beta}_k \leftarrow \alpha_k^{\text{stop}}$.
5. Set $x_{k+1} \leftarrow x_k + \bar{\alpha}_k d_k$, where $\bar{\alpha}_k \in [0, \bar{\beta}_k]$.

The convergence of the modified Frank-Wolfe method is proved in Theorem 4 of [13] under the assumption that $\bar{\alpha}_k$ in Step (5.) is chosen by exact line-search; however a careful review of the proof therein shows that convergence is still valid if one uses a step-size rule in the spirit of (9) that uses the quadratic upper-approximation of $f(\cdot)$ using L or $\bar{L} \geq L$. The criterion in Step (4.) of Algorithm 2 for choosing between the regular Frank-Wolfe step and the away step seems to be tailor-made for the convergence proof in [13]. In examining the proof of convergence in [13], one finds the fact that $\hat{x}_k$ is an extreme point is not relevant for the proof, nor even is the property that $\hat{x}_k$ is a solution of a linear optimization problem. Indeed, this begs for a different way to think about both generating and analyzing away steps, which we will do shortly in Subsection 2.2.

Away-steps are *not* affine-invariant. The feasible region S of (5) can always be (implicitly) expressed as $S = \text{conv}(\mathcal{A})$ where $\mathcal{A} = \{\tilde{x}^j : j \in \mathcal{J}\}$ is a (possibly infinite) collection of points in S that includes all of the extreme points of S. In fact, in many current applications of Frank-Wolfe and its relatives, S is explicitly constructed as $S := \text{conv}(\mathcal{A})$ for a given collection $\mathcal{A}$ whose members are referred to as "atoms"; and each atom $\tilde{x}^j \in \mathcal{A}$ is a particularly "simple" point (such as a unit coordinate vector $\pm e^i$, a rank-1 matrix, etc.). Let us consider the (possibly infinite-dimensional)

vector space $V := \{\alpha \in \mathbb{R}^{|\mathcal{J}|} : \text{support}(\alpha) \text{ is finite}\}$, and define the simplicial set $\Delta_{\mathcal{J}}$ by:

$$\Delta_{\mathcal{J}} := \left\{ \alpha \in V : \alpha \geq 0,\ \sum_{j \in \mathcal{J}} \alpha_j = 1 \right\} ,$$

and consider the linear map $M(\cdot) : \Delta_{\mathcal{J}} \to S$ such that $M(\alpha) := \sum_{j \in \mathcal{J}} \alpha_j \tilde{x}^j$. Then it is obvious that the following two optimization problems are equivalent:

$$\min_{x \in S} f(x) \quad \equiv \quad \min_{\alpha \in \Delta_{\mathcal{J}}} f(M(\alpha)) , \tag{12}$$

where the left-side is our original given problem of interest (5) and the right-side is its re-expression using the convex weights $\alpha \in \Delta_{\mathcal{J}}$ as the variables. Furthermore, it follows from the fundamental affine-invariance of the regular Frank-Wolfe Method (Algorithm 1) as articulated by Jaggi [17] that the Frank-Wolfe Method applied to the left-side problem above is equivalent (via the linear mapping $M(\cdot)$) to the Frank-Wolfe Method applied to the right-side problem above. However, this affine invariance property does *not* extend to the away-step modification of the method, due to the fact that the facial structure of a convex set is not affine invariant – not even so in the case when S is a polytope. This is illustrated in Figure 1. The left panel shows a polytopal feasible region $S \subset \mathbb{R}^3$ with $\mathcal{F}_S(x_k)$ highlighted. The polytope S has 10 extreme points. The right panel shows $\mathcal{F}_S(x_k)$ by itself in detail, wherein we see that $x_k = .25\tilde{x}_1 + .25\tilde{x}_2 + .50\tilde{x}_3$ (among several other combinations of other extreme points of $\mathcal{F}_S(x_k)$ as well). Let us now consider the atomic expression of the set S using the 10 extreme points S and instead expressing our problem in the format of the right-side of (12), wherein the feasible region is the unit simplex in $\mathbb{R}^{10}$, namely $\Delta_{10} := \{\alpha \in \mathbb{R}^{10} : \alpha \geq 0,\ e^T\alpha = 1\}$ where $e = (1, \ldots, 1)$ is the vector of ones. If the current iterate x_k is given the atomic expression $\alpha_k = (.25, .25, .50, 0, 0, 0, 0, 0, 0, 0)$, then the minimal face $\mathcal{F}_{\Delta_{10}}(\alpha_k)$ of α_k in Δ_{10} is the sub-simplex $\{\alpha \in \mathbb{R}^{10} : \alpha \geq 0,\ e^T\alpha = 1,\ \alpha_4 = \cdots \alpha_{10} = 0\}$, which corresponds back in $S \subset \mathbb{R}^3$ to the narrow triangle in the right panel of Figure 1, and which is a small subset of the pentagon corresponding to the minimal face $\mathcal{F}_S(x_k)$ of x_k in S. Indeed, this example illustrates the general fact that the faces of the atomic expression of S will always correspond to subsets of the faces of the facial structure of S. Therefore, away-step sub-problem optimization computations using the original representation of S will optimize over larger subsets of S than will the corresponding computations using the atomic re-expression of the problem. Indeed, we will show in Section 4 in the context of matrix completion that by working with the original representation of the set S in the setting of using away-steps, one can obtain significant computational savings over working with the atomic representation of the problem.

Last of all, we point out that the away-step modified Frank-Wolfe methods studied by Lacoste-Julien and Jaggi [21, 22], Beck and Shtern [1], and Peña et al. [28] can all be viewed as applying the away-step method (Algorithm 2) to the "atomic" representation of the optimization problem as in the right-side of (12).

2.2 An "In-Face" Extended Frank-Wolfe Method

Here we present an "in-face" extension of the Frank-Wolfe method, that is significantly more general than the away-step method of Wolfe [38] and Guélat and Marcotte [13] (Algorithm 2), and its atomic

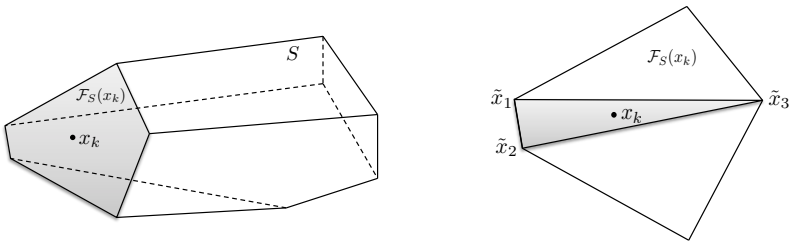

Figure 1: Illustration that facial structure of a polytope is not affine invariant.

version studied by by Lacoste-Julien and Jaggi [21, 22], Beck and Shtern [1], and Peña et al. [28]. The method is motivated by the desire to compute and work with points x that have specific structure, usually sparsity (in the case when x is a vector or matrix) or low-rank (in the case when x is a matrix). More generally, we will think of the structure as being related to the dimension of the minimal face $\mathcal{F}_S(x)$ of S containing x. The algorithm is designed to balance progress towards two different goals, namely (i) progress towards optimizing the objective function, and (ii) the aim of having the iterates lie in low-dimensional faces of S. In the case of the matrix completion problem (2) in particular, if an iterate lies in a low-dimensional face of S then the iterate will have low rank (see Theorem 3). Such low rank is advantageous not only because we want the output solution to have low rank, but also because a low-rank iterate yields a substantial reduction in the computation costs at that iteration. This last point will be further developed and exploited in Sections 3 and 4.

We present our "In-Face Extended Frank-Wolfe Method" in Algorithm 3. At Step (2.) of each iteration the algorithm works with an "in-face" direction d_k which will keep the next candidate point in the current minimal face $\mathcal{F}_S(x_k)$. This is equivalent to requiring that $x_k + d_k$ lies in the affine hull of $\mathcal{F}_S(x_k)$, which is denoted by $\mathrm{Aff}(\mathcal{F}_S(x_k))$. Other than the affine hull condition, the direction d_k can be *any* descent direction of $f(\cdot)$ at x_k if such a direction exists. The candidate iterate x_k^B is generated by stepping in the direction d_k all the way to the relative boundary of the minimal face of the current point x_k. The point x_k^A is the candidate iterate generated using the in-face direction and a suitable step-size $\bar{\beta}_k$, perhaps chosen by exact line-search or by a quadratic approximation rule. In Steps (3a.) and (3b.) the algorithm applies criteria for choosing which, if any, of x_k^B or x_k^A to accept as the next iterate of the method. If the criteria are not met for either x_k^B or x_k^A, then the method computes a regular Frank-Wolfe step in Step (3c.) and updates the lower bound B_k.

Let us now discuss a few strategies for computing in-face directions. One recovers the away-step direction of the method of Guélat and Marcotte [13] by choosing:

$$d_k \leftarrow x_k - \hat{x}_k \ , \quad \text{where} \quad \hat{x}_k \leftarrow \arg\max_x \{\nabla f(x_k)^T x : x \in \mathcal{F}_S(x_k)\} \ . \tag{13}$$

Another natural way to compute a suitable d_k, that is computationally facile for relatively low-dimensional faces and for certain problem instances (including matrix completion), is to directly solve for an (approximately) optimal objective function solution over the low-dimensional face $\mathcal{F}_S(x_k)$ and thereby set:

$$d_k \leftarrow x_k^M - x_k \ , \quad \text{where} \quad x_k^M \leftarrow \arg\min_x \{f(x) : x \in \mathcal{F}_S(x_k)\} \ . \tag{14}$$

Algorithm 3 In-Face Extended Frank-Wolfe Method for optimization problem (5)

Initialize at $x_0 \in S$, (optional) initial lower bound B_{-1}, $k \leftarrow 0$.
Choose $\bar{L} \geq L$, $\bar{D} \geq D$, and constants γ_1, γ_2 satisfying $0 \leq \gamma_1 \leq \gamma_2 \leq 1$.

At iteration k:
1. Compute $\nabla f(x_k)$. $B_k \leftarrow B_{k-1}$.
2. Compute direction d_k for which $x_k + d_k \in \text{Aff}(\mathcal{F}_S(x_k))$ and $\nabla f(x_k)^T d_k < 0$. (If no d_k exists, go to Step (3c.).)
   $\alpha_k^{\text{stop}} \leftarrow \arg\max_{\alpha}\{\alpha : x_k + \alpha d_k \in \mathcal{F}_S(x_k)\}$.
   $x_k^B := x_k + \alpha_k^{\text{stop}} d_k$.
   $x_k^A := x_k + \bar{\beta}_k d_k$ where $\bar{\beta}_k \in [0, \alpha_k^{\text{stop}}]$.
3. Choose next iterate:
   (a.) (Go to a lower-dimensional face.)
   If $\frac{1}{f(x_k^B)-B_k} \geq \frac{1}{f(x_k)-B_k} + \frac{\gamma_1}{2L\bar{D}^2}$, set $x_{k+1} \leftarrow x_k^B$.
   (b.) (Stay in current face.)
   Else if $\frac{1}{f(x_k^A)-B_k} \geq \frac{1}{f(x_k)-B_k} + \frac{\gamma_2}{2L\bar{D}^2}$, set $x_{k+1} \leftarrow x_k^A$.
   (c.) (Do regular FW step and update lower bound.) Else, compute:
   $\tilde{x}_k \leftarrow \arg\min_{x}\{\nabla f(x_k)^T x : x \in S\}$.
   $x_{k+1} \leftarrow x_k + \bar{\alpha}_k(\tilde{x}_k - x_k)$ where $\bar{\alpha}_k \in [0, 1]$.
   $B_k^w \leftarrow f(x_k) + \nabla f(x_k)^T(\tilde{x}_k - x_k)$, $B_k \leftarrow \max\{B_{k-1}, B_k^w\}$.

Note that in this case, we may naturally set $\bar{\beta}_k := 1$. Another related type of in-face direction that may be of interest is to consider a regular Frank-Wolfe step within $\mathcal{F}_S(x_k)$, whereby we select:

$$d_k \leftarrow \tilde{x}_k^F - x_k \ , \quad \text{where} \quad \tilde{x}_k^F \leftarrow \arg\min_{x}\{\nabla f(x_k)^T x : x \in \mathcal{F}_S(x_k)\} \ . \tag{15}$$

One may interpret this "in-face Frank-Wolfe step" as a single iteration of the Frank-Wolfe Method applied to the subproblem in (14). As we elaborate in Section 4 when discussing the practical merits of these approaches, our main interests are in the away-step strategy (13) and the full optimization strategy (14). Both of these in-face Frank-Wolfe step strategies lead to significant computational advantages over the regular Frank-Wolfe Method, as will be shown in Section 4.

One immediate advantage of the In-Face Extended Frank-Wolfe Method (Algorithm 3) compared to the away-step modified Frank-Wolfe method of Guélat and Marcotte [13] (Algorithm 2) has to do with the number and sizes of linear optimization sub-problems that are solved. Algorithm 2 needs to solve two linear optimization subproblems at each iteration – a "small" subproblem on the minimal face $\mathcal{F}_S(x_k)$ and a "large" subproblem on the entire set S. In contrast, even when computing directions using away-step computations, Algorithm 3 must solve the "small" linear optimization problem on the minimal face $\mathcal{F}_S(x_k)$, but the method will only need to solve the "large" subproblem on the entire set S if it needs to process Step (3c.). The computational advantage from not having to solve the "large" subproblem at every iteration will be shown in Section 4.

We now discuss the criteria that are used in Step (3.) to choose between the next step x_k^B that lies in the relative boundary of the current minimal face $\mathcal{F}_S(x_k)$, the step x_k^A that does not necessarily lie in the relative boundary of the current minimal face $\mathcal{F}_S(x_k)$, and a regular Frank-

Wolfe step. We see from Step (3.) of Algorithm 3 that a regular Frank-Wolfe step will be chosen as the next iterate unless the criterion of either Step (3a.) or (3b.) are met. The criteria in Step (3a.) is met if x_k^B (which lies on the relative boundary of $\mathcal{F}_S(x_k)$ by virtue of the definition of α_k^{stop}) provides sufficient decrease in the optimality gap as measured with the criterion:

$$\frac{1}{f(x_k^B) - B_k} \geq \frac{1}{f(x_k) - B_k} + \frac{\gamma_1}{2\bar{L}\bar{D}^2} \ .$$

The criteria in Step (3b.) is met if x_k^A provides sufficient decrease in the optimality gap as measured similar to above but using γ_2 rather than γ_1. Since $\gamma_1 \leq \gamma_2$, Step (3a.) requires a lesser decrease in the optimality bound gap than does Step (3b.).

In settings when we strongly desire to compute iterates that lie on low-dimensional faces (as in the low-rank matrix completion problem (2)), we would like the criteria in Steps (3a.) and (3b.) to be relatively easily satisfied (perhaps with it being even easier to satisfy the criteria in Step (3a.) as this will reduce the dimension of the minimal face). This can be accomplished by setting the values of γ_1 and γ_2 to be lower rather than higher. Indeed, setting $\gamma_1 = 0$ ensures in Step (3a.) that the next iterate lies in a lower-dimensional face whenever x_k^B (which by definition lies in a lower dimensional face than x_k) does not have a worse objective function value than $f(x_k)$. Also, if one sets γ_2 to be smaller, then the criteria in Step (3b.) is more easily satisfied, which ensures that the new iterate will remain in the current face $\mathcal{F}_S(x_k)$ as desired when the criterion of Step (3b.) is satisfied.

As we have discussed, the ability to induce solutions on low-dimensional faces by setting γ_1 and γ_2 to have low values can be extremely beneficial. However, this all comes at a price in terms of computational guarantees, as we now develop. Before presenting the computational guarantee for Algorithm 3 we first briefly discuss step-sizes; the step-size $\bar{\beta}_k$ for steps to the in-face point x_k^A are determined in Step (2.), and the step-size $\bar{\alpha}_k$ for regular Frank-Wolfe steps is chosen in Step (3c.). One strategy is to choose these step-sizes using an exact line-search if the line-search computation is not particularly burdensome (such as when $f(\cdot)$ is a quadratic function). Another strategy is to determine the step-sizes according to the quadratic upper approximation of $f(\cdot)$ much as in Theorem 1, which in this context means choosing the step-sizes as follows:

$$\bar{\beta}_k := \min\left\{\frac{-\nabla f(x_k)^T d_k}{\bar{L}\|d_k\|^2},\ \alpha_k^{\text{stop}}\right\} \quad , \quad \bar{\alpha}_k := \min\left\{\frac{\nabla f(x_k)^T (x_k - \tilde{x}_k)}{\bar{L}\|x_k - \tilde{x}_k\|^2},\ 1\right\} \ . \tag{16}$$

Let N_k^a, N_k^b, and N_k^c denote the number of times within the first k iterations that the iterates are chosen according to the criteria in Steps (3a.), (3b.), and (3c.), respectively. Then $k = N_k^a + N_k^b + N_k^c$, and we have the following computational guarantee.

Theorem 2. *Suppose that the step-sizes used in Algorithm 3 are determined either by exact line-search or by* (16)*. After k iterations of Algorithm 3 it holds that:*

$$f(x_k) - f^* \leq f(x_k) - B_k \leq \frac{1}{\frac{1}{f(x_0) - B_0} + \frac{\gamma_1 N_k^a}{2\bar{L}\bar{D}^2} + \frac{\gamma_2 N_k^b}{2\bar{L}\bar{D}^2} + \frac{N_k^c}{2\bar{L}\bar{D}^2}} < \frac{2\bar{L}\bar{D}^2}{\gamma_1 N_k^a + \gamma_2 N_k^b + N_k^c} \ .$$

Proof: The first inequality is true since $B_k \leq f^*$, and the third inequality is true since $f(x_0) \geq B_0$,

so we need only prove the second inequality, which can be equivalently written as:

$$\frac{1}{f(x_k) - B_k} \geq \frac{1}{f(x_0) - B_0} + \frac{\gamma_1 N_k^a}{2\bar{L}\bar{D}^2} + \frac{\gamma_2 N_k^b}{2\bar{L}\bar{D}^2} + \frac{N_k^c}{2\bar{L}\bar{D}^2} \ . \tag{17}$$

Notice that (17) is trivially true for $k = 0$ since $N_k^a = N_k^b = N_k^c = 0$ for $k = 0$. Let $\Delta^k := (f(x_k) - B_k)^{-1}$ denote the inverse objective function bound gap at iteration k. Then if the next iterate is chosen by satisfying the criteria in Step (3a.), it holds that $\Delta^{k+1} \geq (f(x_{k+1}) - B_k)^{-1} \geq \Delta^k + \frac{\gamma_1}{2\bar{L}\bar{D}^2}$ where the first inequality derives from $B_{k+1} \geq B_k$ and the second inequality is from the criterion of Step (3a.). Similarly, if the next iterate is chosen by satisfying the criteria in Step (3b.), it holds using similar logic that $\Delta^{k+1} \geq \Delta^k + \frac{\gamma_2}{2\bar{L}\bar{D}^2}$. And if the next iterate is chosen in Step (3c.), namely we take a regular Frank-Wolfe step, then inequality (11) holds, which is $\Delta^{k+1} \geq \Delta^k + \frac{1}{2\bar{L}\bar{D}^2}$. Applying induction then establishes (17), which completes the proof. □

Here we see that choosing smaller values of γ_1 and γ_2 can have a detrimental effect on the progress of the algorithm in terms of the objective function optimality gap, while larger values ensure better convergence guarantees. At the same time, smaller values of γ_1 and γ_2 are more effective at promoting iterates to lie on low-dimensional faces. Thus there is a clear tradeoff between objective function optimality gap accuracy and low-dimensional structure, dictated by the values of γ_1 and γ_2. One strategy that is worth studying is setting $\gamma_1 = 0$ and γ_2 to be relatively large, say $\gamma_2 = 1$ for example. With these values of the parameters we take an in-face step in Step (3a.) (which lowers the dimension of the face of the iterate) whenever doing so will not adversely affect the objective function value. This and other strategies for setting γ_1 and γ_2 will be examined in Section 4.

A simplified algorithm in the case of full optimization over the current minimal face. Let us further examine the dynamics of Algorithm 3 in the case of (14), where we select the in-face direction by fully optimizing the objective function $f(\cdot)$ over the low-dimensional face $\mathcal{F}_S(x_k)$. Consider performing an in-face step in this case, i.e., suppose that the next iterate is chosen according to the criteria in Steps (3a.)/(3b.) (recall that we set $\bar{\beta}_k := 1$ in this case). Then, at the next iteration, Algorithm 3 is guaranteed to select a regular Frank-Wolfe step via Step (3c.). Indeed, since the next iterate x_{k+1} is chosen as the optimal solution over $\mathcal{F}_S(x_k)$, by definition there are no descent directions at x_{k+1} that remain within $\mathcal{F}_S(x_{k+1}) \subseteq \mathcal{F}_S(x_k)$ and thus no valid in-face directions to be selected. Here we see that the parameters γ_1 and γ_2 are superfluous – a much more natural procedure is to simply alternate between regular Frank-Wolfe steps and fully optimizing over $\mathcal{F}_S(x_k)$. This bears some similarity to, but is distinct from, the "fully corrective" variant of Frank-Wolfe, see, e.g., [15, 17, 22]. (Indeed, these two algorithms coincide if we consider this alternating procedure applied to the lifted problem (12).) In this case, the following computational guarantee follows simply from Theorem 1.

Proposition 1. *Consider a slight variation of Algorithm 3 that alternates between full optimizations* (14) *over the current face* $\mathcal{F}_S(x_k)$ *and regular Frank-Wolfe steps, with step-size* $\bar{\alpha}_k$ *chosen either by exact line-search or by a quadratic approximation rule* (16)*. For simplicity, consider one iteration to consist of both of these operations in sequence. Then, for all* $k \geq 0$*, it holds that:*

$$f(x_k) - f^* \leq f(x_k) - B_k \leq \frac{1}{\frac{1}{f(x_0) - B_0} + \frac{k}{2\bar{L}\bar{D}^2}} \quad < \quad \frac{2\bar{L}\bar{D}^2}{k} \ .$$

3 Solving Matrix Completion Problems using the In-Face Extended Frank-Wolfe Method

We now turn our attention to solving instances of (2) using the the In-Face Extended Frank-Wolfe Method (Algorithm 3). We work directly with the natural parameterization of variables as $m \times n$ matrices $Z \in \mathbb{R}^{m \times n}$ (although, as we discuss in Section 3.6, we utilize low-rank SVD updating to maintain the variables in an extremely memory efficient manner). Recall that the objective function of (2) is $f(Z) := \frac{1}{2}\sum_{(i,j)\in\Omega}(Z_{ij} - X_{ij})^2$, whose gradient is $\nabla f(Z) = (Z - X)_\Omega$. The feasible region of (2) is $S = \mathcal{B}_{N1}(0, \delta)$, which notation we shorten to $\mathcal{B} := \mathcal{B}_{N1}(0, \delta)$. We first discuss the specification and implementation issues in using Algorithm 3 to solve (2).

We will fix the norm on Z to be the nuclear norm $\|\cdot\|_{N1}$, whose dual norm is easily seen to be $\|\cdot\|_{N1}^* = \|\cdot\|_{N\infty}$. Then it is plain to see that under the nuclear norm it holds that the Lipschitz constant of the objective function of (2) is $L = 1$. This follows since for any $Z, Y \in \mathbb{R}^{m\times n}$ we have:

$$\begin{aligned}\|\nabla f(Z) - \nabla f(Y)\|_{N\infty} \;&\leq\; \|\nabla f(Z) - \nabla f(Y)\|_{N2} \;=\; \|(Z - X)_\Omega - (Y - X)_\Omega\|_F \\ &\leq\; \|(Z - Y)\|_F \;=\; \|(Z - Y)\|_{N2} \;\leq\; \|(Z - Y)\|_{N1} \;.\end{aligned}$$

Since the feasible region of (2) is $S = \mathcal{B} := \mathcal{B}_{N1}(0, \delta)$ it follows that the diameter of S is $D = 2\delta$. Let us use the superscript Z^k to denote the k^{th} iterate of the algorithm, to avoid confusion with the subscript notation Z_{ij} for indices of the $(i, j)^{\text{th}}$ component of Z.

3.1 Characterization of faces of the nuclear norm ball

In order to implement Algorithm 3 we need to characterize and work with the minimal face of $\mathcal{B} = \mathcal{B}_{N1}(0, \delta)$ containing a given point. Let $\bar{Z} \in \mathcal{B}$ be given. The minimal face of $\mathcal{B}$ containing $\bar{Z}$ is formally notated as $\mathcal{F}_{\mathcal{B}}(\bar{Z})$. We have the following characterization of $\mathcal{F}_{\mathcal{B}}(\bar{Z})$ due to So [33]:

Theorem 3. (So [33]) *Let $\bar{Z} \in \mathcal{B}$ have thin SVD $\bar{Z} = UDV^T$ and let $r = \text{rank}(\bar{Z})$. Let $\mathcal{F}_{\mathcal{B}}(\bar{Z})$ denote the minimal face of $\mathcal{B}$ containing $\bar{Z}$. If $\sum_{j=1}^r \sigma_j = \delta$, then $\bar{Z} \in \partial\mathcal{B}$ and it holds that:*

$$\mathcal{F}_{\mathcal{B}}(\bar{Z}) = \{Z \in \mathbb{R}^{m\times n} : Z = UMV^T \text{ for some } M \in \mathbb{S}^{r\times r},\; M \succeq 0,\; I \bullet M = \delta\}\;,$$

and $\dim(\mathcal{F}_{\mathcal{B}}(\bar{Z})) = r(r+1)/2 - 1$. Otherwise $\sum_{j=1}^r \sigma_j < \delta$ and it holds that $\mathcal{F}_{\mathcal{B}}(\bar{Z}) = \mathcal{B}$ and $\dim(\mathcal{F}_{\mathcal{B}}(\bar{Z})) = \dim(\mathcal{B}) = m \times n$. □

Theorem 3 above is a reformulation of Theorem 3 of So [33], as the latter pertains to square matrices ($m = n$) and also does not explicitly treat the minimal faces containing a given point, but is a trivial extension of So's theorem.

Theorem 3 explicitly characterizes the correspondence between the faces of the nuclear norm ball and low-rank matrices on its boundary. Note from Theorem 3 that if $\bar{Z} \in \partial\mathcal{B}$ and $r = \text{rank}(\bar{Z})$, then $\mathcal{F}_{\mathcal{B}}(\bar{Z})$ is a linear transformation of the $r \times r$ spectrahedron $\bar{\mathcal{S}}_\delta^r := \{M \in \mathbb{S}^{r\times r} : M \succeq 0,\; I \bullet M = \delta\}$. This property will be most useful as it will make it very easy to compute in-face directions, especially when r is relatively small, as we will see in Section 3.3 and Section 3.4.

3.2 Linear optimization subproblem solution for regular Frank-Wolfe step

In Step (3c.) of Algorithm 3 we need solve a linear optimization problem. Here we show how this can be done efficiently. We need to compute:

$$\tilde{Z}^k \leftarrow \arg \min_{Z \in \mathcal{B}_{N1}(0,\delta)} \nabla f(Z^k) \bullet Z \ . \tag{18}$$

Then an optimal solution $\tilde{Z}^k$ is readily seen to be:

$$\tilde{Z}^k \leftarrow -\delta \mathbf{u}_k \mathbf{v}_k^T \tag{19}$$

where $\mathbf{u}_k$ and $\mathbf{v}_k$ denote the left and right singular vectors, respectively, of the matrix $\nabla f(Z^k)$ corresponding to the largest singular value of $\nabla f(Z^k)$. Therefore computing $\tilde{Z}^k$ in Step (3c.) is relatively easy so long as the computation of the largest singular value of $\nabla f(Z^k)$ and associated left and right eigenvalues thereof are easy to accurately compute. If $|\Omega|$ is relatively small, then there are practically efficient methods (such as power iterations) that can effectively leverage the sparsity of $\nabla f(Z^k)$.

3.3 Strategies and computation of the in-face direction D^k

Let D^k denote the in-face direction computed in Step (2.) of Algorithm 3. As suggested in Section 2.2, we present and discuss two different strategies for generating a suitable D^k, namely (i) using an away-step approach (13), and (ii) directly solving for an optimal objective function solution over the low-dimensional face $\mathcal{F}_{\mathcal{B}}(Z^k)$ (14). In either case, computing D^k requires working with the thin SVD of Z^k, which characterizes $\mathcal{F}_{\mathcal{B}}(Z^k)$ as stated in Theorem 3. Of course, the thin SVD of Z^k can be recomputed at every iteration, but this is generally very inefficient. As we expand upon in Section 3.6, the thin SVD of Z^{k+1} can be efficiently *updated* from the thin SVD of Z^k by utilizing the structure of the regular Frank-Wolfe and in-face directions. For now, we simply assume that we have access to the thin SVD of Z^k at the start of iteration k.

Away-step Strategy. Here we choose D^k by setting $D^k \leftarrow Z^k - \hat{Z}^k$ where $\hat{Z}^k$ is the solution of the linear optimization maximization problem over the current minimal face, as in Step (2.) of the away-step algorithm (Algorithm 2). We compute the "away-step point" $\hat{Z}^k$ by computing:

$$\hat{Z}^k \leftarrow \arg \max_{Z \in \mathcal{F}_{\mathcal{B}}(Z^k)} \nabla f(Z^k) \bullet Z \ , \tag{20}$$

and set $D^k \leftarrow Z^k - \hat{Z}^k$. To see how to solve (20) efficiently, we consider two cases, namely when $Z^k \in \text{int}(\mathcal{B})$ and when $Z^k \in \partial(\mathcal{B})$. In the case when $Z^k \in \text{int}(\mathcal{B})$, then $\mathcal{F}_{\mathcal{B}}(Z^k) = \mathcal{B}$ and the optimal solution in (20) is just the negative of the solution of (19), namely $\hat{Z}^k = \delta \mathbf{u}_k \mathbf{v}_k^T$.

In the case when $Z^k \in \partial(\mathcal{B})$, $\text{rank}(Z^k) = r$, and Z^k has thin SVD $Z^k = UDV^T$, we use the characterization of $\mathcal{F}_{\mathcal{B}}(Z^k)$ in Theorem 3 to reformulate (20) as:

$$\hat{Z}^k \leftarrow U\hat{M}^k V^T \quad \text{where} \quad \hat{M}^k \leftarrow \arg \max_{M \in \bar{\mathcal{S}}_\delta^r} G^k \bullet M \ , \tag{21}$$

and where $G^k := \frac{1}{2}(V^T \nabla f(Z^k)^T U + U^T \nabla f(Z^k) V)$ so that $\nabla f(Z^k) \bullet UMV^T = G^k \bullet M$ for all $M \in \bar{\mathcal{S}}_\delta^r$. An optimal solution to the subproblem in (21) is readily seen to be

$$\hat{M}^k \leftarrow \delta \mathbf{u_k} \mathbf{u_k}^T \tag{22}$$

where $\mathbf{u}_k$ is the normalized eigenvector corresponding to the largest eigenvector of the $r \times r$ symmetric matrix G^k. Therefore computing $\hat{Z}^k$ in (20) is relatively easy so long as the computation of the largest eigenvalue of G^k and associated eigenvector thereof are easy to accurately compute. Furthermore, note that $\hat{Z}^k = U\hat{M}^k V^T = \delta U \mathbf{u_k}\mathbf{u_k}^T V^T$ is a rank-one matrix.

The above computational steps require the thin SVD of Z^k as well as being able to efficiently compute the largest eigenvalue/eigenvector pair of G^k. Efficient computational strategies for managing the thin SVD of Z^k are described in Section 3.6. We compute the largest eigenvalue/eigenvector pair of G^k by either direct factorization of the $r \times r$ matrix G^k, or by power-method approximation, depending on the value of r.

The development of the in-face Frank-Wolfe step strategy (15) in this case is quite similar. Indeed, we simply replace the maximization in (21) with a minimization, which corresponds to a smallest eigenvalue computation, and set D^k accordingly.

Direct Solution on the Minimal Face. In this strategy we use the alternating version of Algorithm 3 described at the end of Section 2.2 and we choose D^k by setting $D^k \leftarrow \bar{Z}^k - Z^k$ where $\bar{Z}^k$ optimizes (exactly or perhaps only approximately) the original objective function $f(Z)$ over the current minimal face, under the assumption that such optimization can be done efficiently and accurately. Indeed, when $Z^k \in \mathrm{int}(\mathcal{B})$, then we default to the previous away-step strategy since optimizing over the minimal face is identical to the original problem (2). Otherwise, when $Z^k = UDV^T \in \partial(\mathcal{B})$ we again use the characterization of $\mathcal{F}_{\mathcal{B}}(Z^k)$ in Theorem 3 to compute $\bar{Z}^k$ as:

$$\bar{Z}^k \;\leftarrow\; U\bar{M}^k V^T \quad \text{where} \quad \bar{M}^k \;\leftarrow\; \arg\min_{M \in \bar{\mathcal{S}}^r_\delta} \; f(UMV^T) \; . \tag{23}$$

Of course, it is only sensible to consider this strategy when Z^k has low rank, for otherwise (23) is nearly as difficult to solve as the original problem (2) whose solution we seek to approximate using the In-face Extended Frank-Wolfe Method. Since $f(\cdot)$ is a convex quadratic function, it follows that the subproblem in (23) is solvable as a semidefinite/second-order conic optimization problem and thus conic interior-point methods may be practical. Alternatively, one can approximately solve (23) by taking a number of steps of any suitably effective method, such as a proximal/accelerated first-order method [37] (or even the Frank-Wolfe Method itself).

3.4 Computing the maximal step-size α_k^{stop} in Step (2.)

Here we describe how to efficiently compute the maximal step-size α_k^{stop} in Step (2.) of Algorithm 3, which is determined as:

$$\alpha_k^{\mathrm{stop}} \leftarrow \arg\max_{\alpha} \{\alpha : Z^k + \alpha D^k \in \mathcal{F}_{\mathcal{B}}(Z^k)\} \; . \tag{24}$$

Let us first assume that $Z^k \in \partial(\mathcal{B})$. We will utilize the SVD of the current iterate $Z^k = UDV^T$. Using either the away-step strategy or the direct solution strategy for determining the in-face direction D^k in Section 3.3, it is simple to write $D^k = U\Delta V^T$ for an easily given matrix $\Delta \in \mathbb{S}^{r\times r}$. Since $Z^k \in \partial(\mathcal{B})$ and $Z^k + D^k \in \mathcal{F}_{\mathcal{B}}(Z^k)$ it holds that $I \bullet D = \delta$ and hence $I \bullet \Delta = 0$. Using the characterization of $\mathcal{F}_{\mathcal{B}}(Z^k)$ in Theorem 3 it follows that (24) can be reformulated as:

$$\alpha_k^{\mathrm{stop}} \;\leftarrow\; \arg\max_{\alpha, M} \{\alpha : UDV^T + \alpha U\Delta V^T = UMV^T, \; M \in \bar{\mathcal{S}}^r_\delta\} \;=\; \arg\max_{\alpha} \{\alpha : D + \alpha\Delta \succeq 0\} \; . \tag{25}$$

In the case when D^k is chosen using the away-step approach, we have from (21) and (22) that $\Delta := D - \delta \mathbf{u}_k \mathbf{u}_k{}^T$ satisfies $D^k = Z^k - \hat{Z}^k = U \Delta V^T$. In this case the maximum α satisfying (25) is easily seen to be $\alpha_k^{\text{stop}} := \left(\delta \mathbf{u}_k{}^T D^{-1} \mathbf{u}_k - 1\right)^{-1}$. When D^k is chosen by some other method, such as the direct solution method on the minimal face, the optimal solution of (25) is seen to be $\alpha_k^{\text{stop}} := -\left[\lambda_{\min}\left(D^{-\frac{1}{2}} \Delta D^{-\frac{1}{2}}\right)\right]^{-1}$.

In the case when $Z^k \in \text{int}(\mathcal{B})$, then (24) can be written as $\alpha_k^{\text{stop}} \leftarrow \arg\max\{\alpha : \|Z^k + \alpha D^k\|_{N1} \leq \delta\}$, and we use binary search to approximately determine α_k^{stop}.

3.5 Initial values, step-sizes, and computational guarantees

We initialize Algorithm 3 by setting

$$Z^0 \leftarrow -\delta \mathbf{u}_0 \mathbf{v}_0^T \tag{26}$$

where $\mathbf{u}_0$ and $\mathbf{v}_0$ denote the left and right singular vectors, respectively, of the matrix $\nabla f(0)$ corresponding to the largest singular value of $\nabla f(0)$. This initialization corresponds to a "full step" iteration of the Frank-Wolfe Method initialized at 0 and conveniently satisfies $\text{rank}(Z^0) = 1$ and $Z^0 \in \partial\mathcal{B}$. We initialize the lower bound as $B_{-1} \leftarrow \max\left\{f(0) + \nabla f(0) \bullet Z^0, 0\right\}$, where the first term inside the max corresponds to the lower bound generated when computing Z^0 and the second term is a valid lower bound since $f^* \geq 0$. Moreover, this initialization has a provably good optimality gap, namely $f(Z^0) \leq B_{-1} + 2\delta^2 \leq f^* + 2\delta^2$, which follows from Proposition 3.1 of [12].

Because $f(\cdot)$ is a convex quadratic function, we use an exact line-search to determine $\bar{\beta}_k$ and $\bar{\alpha}_k$ in Steps (2.) and (3c.), respectively, since the line-search reduces to a simple formula in this case.

Utilizing the bound on the optimality gap for Z^0, and recalling that $L = 1$ and $D = 2\delta$, we have from Theorem 2 that the computational guarantee for Algorithm 3 is:

$$f(Z^k) - B_k \leq f(Z^k) - f^* \leq \frac{1}{\frac{1}{f(Z^0) - B_0} + \frac{\gamma_1 N_k^a}{8\delta^2} + \frac{\gamma_2 N_k^b}{8\delta^2} + \frac{N_k^c}{8\delta^2}} \leq \frac{8\delta^2}{4 + \gamma_1 N_k^a + \gamma_2 N_k^b + N_k^c} .$$

3.6 Efficiently Updating the Thin SVD of Z^k

At each iteration of Algorithm 3 we need to access two objects related to the current iterate Z^k: *(i)* the current gradient $\nabla f(Z^k) = (Z^k - X)_\Omega$ (for solving the regular Frank-Wolfe linear optimization subproblem and for computing in-face directions), and *(ii)* the thin SVD $Z^k = UDV^T$ (for computing in-face directions). For large-scale matrix completion problems, it can be very burdensome to store and access all mn entries of the (typically dense) matrix Z^k. On the other hand, if $r := \text{rank}(Z^k)$ is relatively small, then storing the thin SVD of Z^k requires only keeping track of $mr + r + nr$ entries. Thus, when implementing Algorithm 3 as discussed above, instead of storing the entire matrix Z^k, we store in memory the thin SVD of Z^k (i.e., the matrices U, V, and D), which we initialize from (26) and efficiently update as follows. Let D^k denote the direction chosen by Algorithm 3 at iteration $k \geq 0$, which is appropriately scaled so that $Z^{k+1} = Z^k + D^k$. To compute the thin SVD of Z^{k+1}, given the thin SVD of Z^k, we consider the cases of regular Frank-Wolfe directions and in-face directions separately. In the case of a regular Frank-Wolfe direction,

we have that $D^k = \bar{\alpha}_k(-\delta \mathbf{u}_k \mathbf{v}_k^T - Z^k)$ and therefore:

$$Z^{k+1} = Z^k + \bar{\alpha}_k(-\delta \mathbf{u}_k \mathbf{v}_k^T - Z^k) = (1-\bar{\alpha}_k)Z^k - \bar{\alpha}_k \delta \mathbf{u}_k \mathbf{v}_k^T = (1-\bar{\alpha}_k)UDV^T - \bar{\alpha}_k \delta \mathbf{u}_k \mathbf{v}_k^T \ .$$

Thus, given the thin SVD of Z^k, computing the thin SVD of Z^{k+1} is a scaling plus a rank-1 update of the thin SVD, which can be performed very efficiently in terms of both computation time and memory requirements, see [4]. An analogous argument applies to the away-step strategy when $Z^k \in \text{int}(\mathcal{B})$. Otherwise, when $Z^k \in \partial(\mathcal{B})$, recall that we can write any in-face direction as $D^k = U\Delta V^T$ for an easily given matrix $\Delta \in \mathbb{S}^{r\times r}$. Thus we have:

$$Z^{k+1} = Z^k + D^k = UDV^T + U\Delta V^T = U(D+\Delta)V^T \ .$$

Recall from (25) that we have $D + \Delta \succeq 0$. Therefore, to compute the thin SVD of Z^{k+1}, we first compute an eigendecomposition of the $r \times r$ symmetric positive semidefinite matrix $D+\Delta$, so that $D + \Delta = RSR^T$ where R is orthonormal and S is diagonal with nonnegative entries, and then update the thin SVD of Z^{k+1} as $Z^{k+1} = (UR)S(VR)^T$.

To compute the current gradient from the thin SVD of Z^k, note that $\nabla f(Z^k) = (Z^k - X)_\Omega$ is a sparse matrix that is 0 everywhere except on the Ω entires; thus computing $\nabla f(Z^k)$ from the thin SVD of Z^k requires performing $|\Omega|$ length r inner product calculations. As compared to storing the entire matrix Z^k, our implementation requires a modest amount of extra work to compute $\nabla f(Z^k)$, but the cost of this extra work is far outweighed by the benefits of not storing the entire matrix Z^k. Alternatively, it is slightly more efficient to update only the Ω entries of Z^k at each iteration (separately from the thin SVD of Z^k) and to use these entries to compute $\nabla f(Z^k)$.

3.7 Rank accounting

As developed throughout this Section, the computational effort required at iteration k of Algorithm 3 depends very much on $\text{rank}(Z^k)$ for tasks such as computing the in-face direction D^k (using either the away-step approach or direct solution on the minimal face), computing the maximal step-size α_k^{stop} in Step (3.), and updating the thin SVD of Z^k. Herein we examine how $\text{rank}(Z^k)$ can change over the course of the algorithm. At any given iteration i, there are four relevant possibilities for how the next iterate is chosen:

(a) The current iterate Z^i lies on the boundary of $\mathcal{B}$, and the next iterate Z^{i+1} is chosen according to the criteria in Step (3a.).
(b) The current iterate Z^i lies on the boundary of $\mathcal{B}$, and the next iterate Z^{i+1} is chosen according to the criteria in Step (3b.).
(c) The next iterate Z^{i+1} is chosen according to the criteria in Step (3c.).
(d) The current iterate Z^i lies in the interior of $\mathcal{B}$, and the next iterate is chosen according to either the criteria in Step (3a.) or Step (3b.).

The following proposition presents bounds on rank of Z^k.

Proposition 2. *Let N_k^a, N_k^b, N_k^c, and N_k^d denote the number of times within the first k iterations that the above conditions (a), (b), (c), and (d) hold, respectively. Then*

$$\text{rank}(Z^k) \ \leq \ k + 1 - 2N_k^a - N_k^b \ . \tag{27}$$

Proof. Using the choice of the initial point Z^0 developed in Section 3.5, it holds that $\text{rank}(Z^0) = 1$. Now consider the i^{th} iterate value Z^i for $i = 1, \ldots, k$. If condition (a) holds, then Z^{i+1} lies on a lower-dimensional face of $\mathcal{F}_{\mathcal{B}}(Z^i) \subset \mathcal{B}$, whence from Theorem 3 it follows that $\text{rank}(Z^{i+1}) \leq \text{rank}(Z^i) - 1$. If instead condition (b) holds, then $\text{rank}(Z^{i+1}) = \text{rank}(Z^i)$ since Z^{i+1} lies in the relative interior of $\mathcal{F}_{\mathcal{B}}(Z^i) \subset \mathcal{B}$. Finally, in either case that condition (c) or condition (d) holds, it follows from (19) that $\tilde{Z}^i$ is a rank-one matrix and thus it holds that $\text{rank}(Z^{i+1}) \leq \text{rank}(Z^i) + 1$. Since the four cases above are exhaustive, we have $k = N_k^a + N_k^b + N_k^c + N_k^d$ and we obtain $\text{rank}(Z^k) \leq 1 + N_k^c + N_k^d - N_k^a = k + 1 - 2N_k^a - N_k^b$. □

4 Computational Experiments and Results

In this section we present computational results of experiments wherein we apply different versions of the In-Face Extended Frank-Wolfe Method to the nuclear norm regularized matrix completion problem (2).[1] Our main focus is on simulated problem instances, but we also present results for the MovieLens10M dataset. The simulated instances were generated according to the model $X := w_1 UV^T + w_2 \mathcal{E}$, where the entries of $U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{n\times r}$ and $\mathcal{E} \in \mathbb{R}^{m\times n}$ are all i.i.d. standard normal random variables, and the scalar parameters w_1, w_2 control the signal to noise ratio (SNR), namely $w_1 := 1/\|UV^T\|_F$ and $w_2 := 1/(\text{SNR}\|\mathcal{E}\|_F)$. The set of observed entries Ω was determined using uniform random sampling of entries with probability ρ, where ρ is the target fraction of observed entries. The objective function $f(\cdot)$ values were normalized so that $f(0) = .5$ and we chose the regularization parameter δ using a cross-validation-like procedure based on an efficient path algorithm variant of Algorithm 1.[2]

We study several versions of the In-Face Extended Frank-Wolfe Method (Algorithm 3) based on different strategies for setting the parameters γ_1,γ_2, which we compare to the regular Frank-Wolfe Method (Algorithm 1) and the away-step method (Algorithm 2). We also study the atomic version of the away-step method and the "fully corrective" variant of Frank-Wolfe [15, 17, 22] – both of which reformulate (2) in the atomic format of the right-side of (12). Finally, we also include comparisons with CoGEnT: the "forward-backward" variant of the Frank-Wolfe Method studied in [30]. All methods are implemented according to the details presented in Section 3, except for CoGEnT.[3] We focus on the following ten versions of methods with names given below and where "IF- ·" stands for In-Face:

- Frank-Wolfe – Algorithm 1
- IF-(1,1) – Algorithm 3 using an away-step strategy, with $\gamma_1 = 1$, $\gamma_2 = 1$
- IF-(0,1) – Algorithm 3 using an away-step strategy, with $\gamma_1 = 0$, $\gamma_2 = 1$
- IF-(0,∞) – Algorithm 3 using an away-step strategy, with $\gamma_1 = 0$, $\gamma_2 = \infty$. This corresponds to always moving to the relative boundary of the minimal face containing Z_k (thereby reducing the rank of Z^{k+1}) as long as the objective function value does not increase, while never moving partially within the current face.

[1] All computations were performed using MATLAB R2015b on a 3 GHz Intel Core i7 MacBook Pro laptop.

[2] Specifically, we apply a version of Algorithm 1 that periodically increases the value of δ, utilizing the previously found solution as a warm-start at the new value of δ. We maintain a holdout set Ω' and ultimately select the value of δ that minimizes the least-squares error on this set.

[3] The MATLAB code for CoGEnT was obtained from [29].

- IF-Optimization – The simplified version of Algorithm 3 with full in-face optimization as described at the end of Section 2.2. The in-face optimization subproblem is (approximately) solved using the proximal gradient method with matrix entropy prox function.
- IF-Rank-Strategy – Algorithm 3 with the away-step strategy and with γ_1 and γ_2 set dynamically as follows: we initially set $\gamma_1 = \gamma_2 = \infty$, and then reset $\gamma_1 = \gamma_2 = 1$ after we observe five consecutive iterations where $\text{rank}(Z^k)$ does not increase. This version can be interpreted as a two-phase method where we run Algorithm 1 until we observe that $\text{rank}(Z^k)$ begins to "stall," at which point we switch to Algorithm 3 with $\gamma_1 = \gamma_2 = 1$.
- FW-Away-Natural – Algorithm 2
- FW-Away-Atomic – Algorithm 2 applied to the atomic reformulation of (2) using the right-side of (12) [1, 21, 22, 28].
- FW-Fully-Corrective – The "fully corrective" variant of Frank-Wolfe [15, 17, 22], which works with the atomic reformulation of (2) and, at each iteration, fully optimizes the objective function of (2) over the convex hull of the current set of active atoms. The "correction" optimization subproblem is (approximately) solved using the proximal gradient method with entropy prox function over the standard unit simplex.
- CoGEnT – The matrix completion variant of the CoGEnT Method studied in [30]. This variant uses singular value thresholding for the truncation/backward step – at each iteration, the algorithm computes the SVD of the current iterate and truncates small singular values to zero. This step is followed by an enhancement step that optimizes the objective function over the weights in the SVD. The singular value thresholding parameter is set to $.05 \cdot \delta$ and the algorithmic parameter η is set to 0.5.

Tables 1, 2, and 3 present our aggregate computational results. Before discussing these in detail, it is useful to first study Figure 2 which shows the behavior of each method in terms of ranks of iterates[4] (left panel) and relative optimality gap (right panel) as a function of run time, for a particular (and very typical) simulated instance. Examining the rank plots in the left panel, we see that the evolution of $\text{rank}(Z^k)$ is as follows: the four methods IF-(1,1), IF-(0,1), IF-(0,∞), and FW-Away-Natural all quickly attain $\text{rank}(Z^k) \approx 37$ (the apparent rank of the optimum) and then stay at or near this rank from then on. In contrast, the four methods Frank-Wolfe, IF-Rank-Strategy, IF-Optimization and FW-Away-Atomic all grow $\text{rank}(Z^k)$ approximately linearly during the early stages (due to a larger percentage of regular Frank-Wolfe steps), and then reach a maximum value that can be an order of magnitude larger than the optimal rank before the rank starts to decrease. Once the rank starts to decrease, IF-Rank-Strategy and IF-Optimization decrease $\text{rank}(Z^k)$ rather rapidly, whereas Frank-Wolfe and FW-Away-Atomic decrease $\text{rank}(Z^k)$ painfully slowly.

The right panel of Figure 2 shows the relative optimality gaps of the methods. It is noteworthy that two methods – IF-Optimization and IF-Rank-Strategy – achieve very rapid progress during their early stages, a point that we will soon revisit. However, all methods exhibit eventual slow convergence rates which is in line with the $O(1/k)$ theoretical convergence bound.

Let us now synthesize the two panels of Figure 2. The four methods Frank-Wolfe, IF-Rank-Strategy, IF-Optimization and FW-Away-Atomic each go through two phases: in the

[4]The rank of a matrix is computed as the number of singular values larger than 10^{-6}. The rank-1 SVD computation for equation (18) is performed using the Matlab function `eigs`.

first phase each constructs a "high information" (high rank) solution (by taking mostly regular Frank-Wolfe steps), followed by a second phase where the solution is "refined" by lowering the rank while further optimizing the objective function (by taking proportionally more away-steps). FRANK-WOLFE and FW-AWAY-ATOMIC build up to very high information but their build-down is sorely ineffective both in terms of ranks and objective function values. IF-RANK-STRATEGY is extremely effective at the refinement phase, and IF-OPTIMIZATION is less effective in terms of rank reduction but still more so than the other methods except of course for IF-RANK-STRATEGY. The other four methods, namely IF-(1,1), IF-(0,1), IF-(0,∞), and FW-AWAY-NATURAL, all rarely exceed rank 37, as they spend a very high proportion of their effort on away-steps. Of these four methods, IF-(0,∞) tends to perform best in terms of objective function values, as will be seen shortly in Tables 1 and 2. Last of all, we point out that for very large-scale problems storing the SVD of a high-rank matrix may become burdensome (over and above the computational cost for computing in-face directions on high-dimensional faces); thus it is important that the maximum rank of the iterates be kept small. In this regard Figure 2 indicates that excessive memory may arise for FRANK-WOLFE, IF-RANK-STRATEGY, FW-AWAY-ATOMIC, and possibly IF-OPTIMIZATION.

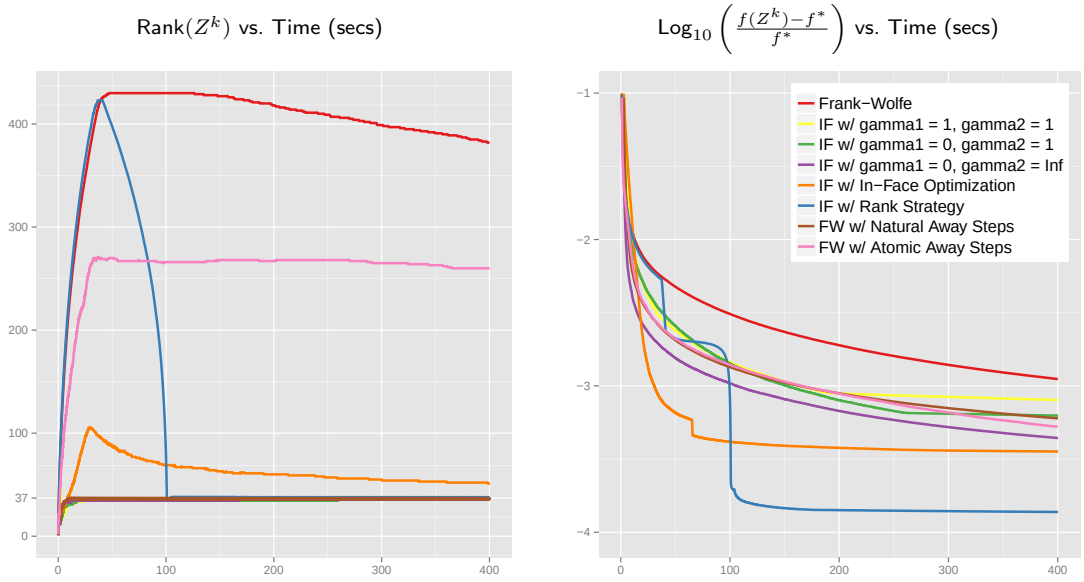

Figure 2: Figure showing plots of rank and relative optimality gap (log-scale) versus run time for different methods/strategies, for a single randomly generated problem instance with $m = 2000$, $n = 2500$, $\rho = 0.01$, $r = 10$, SNR $= 4$, and $\delta = 8.01$. This problem has a (very nearly) optimal solution with rank 37.

Table 1 presents computational results for three different types of small-scale examples, averaged over 25 sample instances generated and run for each type. Note that the run time, final rank, and maximum ranks reported in Table 1 are in sync with the patterns observed in Figure 2. IF-RANK-STRATEGY exhibits the best run times, followed by IF-OPTIMIZATION and then by IF-(0,∞), all of which significantly outperform FRANK-WOLFE, FW-AWAY-NATURAL, and FW-AWAY-ATOMIC. Furthermore, IF-OPTIMIZATION and IF-(0,∞) have relatively low values of the maximum rank (unlike IF-RANK-STRATEGY), while not giving up too much in terms of run time relative to IF-RANK-STRATEGY. Note that FW-AWAY-ATOMIC and FW-FULLY-CORRECTIVE are dramatically

ineffective at delivering low rank solutions, which is undoubtedly related to the fact that the faces of the atomic representation are simply too small to be effective – see Figure 1 and the discussion at the end of Section 2. Note that our best In-Face methods – IF-(0,∞), IF-Optimization, and IF-Rank-Strategy – significantly beat both FW-Fully-Corrective and CoGEnT in both run time and final rank; this fact may be attributed to several factors including the considerable time required to solve the correction/enhancement subproblems when the number of atoms is large.

Table 2 presents computational results for eight individual medium-large scale examples. Here we see mostly similar performance of the different methods as was seen for the small-scale examples in Table 1. IF-Rank-Strategy, IF-(0,∞), and IF-Optimization deliver the best balance between final rank, maximum rank, and run time, with perhaps IF-(0,∞) delivering consistently lower rank solutions albeit with higher run times. We note that for these instances IF-Rank-Strategy does not consistently deliver low rank solutions, which is due to the extra time it takes before the second phase ("refinement") of the method commences. We did not include results for CoGEnT as there was insufficient memory to run CoGEnT on any of these instances.[5] Similar to observations from Table 1, our best In-Face methods – IF-(0,∞), IF-Optimization, and IF-Rank-Strategy – significantly beat FW-Fully-Corrective in both run time and final rank.

Small-Scale Examples (25 samples per example)											
			In-Face Extended FW (IF-…)					Away Steps			
			γ_1, γ_2								
Data	Metric	Regular FW	1,1	0,1	0,∞	In-Face Opt.	Rank Strategy	Natural	Atomic	Fully Corrective FW	CoGEnT
$m = 200, n = 400, \rho = 0.10$	Time (secs)	29.51	22.86	23.07	7.89	**2.34**	**2.30**	14.71	6.21	8.76	20.85
$r = 10, \text{SNR} = 5, \delta_{\text{avg}} = 3.75$	Final Rank	118.68	16.36	16.36	**16.44**	29.32	28.20	16.72	119.00	92.84	79.96
	Maximum Rank	146.48	19.04	17.28	**17.56**	32.08	145.20	18.04	121.96	991.60*	**
$m = 200, n = 400, \rho = 0.20$	Time (secs)	115.75	153.42	150.89	27.60	20.62	**3.48**	50.52	24.52	196.29	65.88
$r = 15, \text{SNR} = 4, \delta_{\text{avg}} = 3.82$	Final Rank	96.44	16.16	16.12	**16.52**	19.88	21.24	16.68	106.60	107.04	93.40
	Maximum Rank	156.52	26.72	17.96	**17.80**	31.48	160.36	18.84	106.80	1812.92*	**
$m = 200, n = 400, \rho = 0.30$	Time (secs)	171.23	198.96	202.01	35.93	31.67	**5.04**	66.22	67.72	>381.91	93.93
$r = 20, \text{SNR} = 3, \delta_{\text{avg}} = 3.63$	Final Rank	91.80	20.08	20.08	**20.60**	**21.72**	25.56	20.44	94.64	113.84	104.60
	Maximum Rank	162.24	25.80	22.04	**21.96**	33.36	168.72	22.16	95.08	1609.40*	**

Table 1: Table reports the time required for each method to reach a relative optimality gap of $10^{-2.5}$, the rank of the corresponding final solution, and the maximum rank of the iterates therein (as an indicator of additional memory/computational requirements). Numbers highlighted in boldface indicate the methods that perform well with regard to each criteria, while not performing poorly on run time. All results are averaged over 25 samples for each problem type.
*For this algorithm, we counted the maximum number of atoms instead of the maximum rank.
**For CoGEnT, neither maximum rank nor maximum atoms is a relevant metric of memory requirements.

Table 3 shows computational tests on a large-scale real dataset, namely the MovieLens10M dataset, with $m = 69878$, $n = 10677$, $|\Omega| = 10^7$ (with sparsity approximately 1.3%), and $\delta = 2.59$. We only tested IF-(0,∞) (and benchmarked against Frank-Wolfe and FW-Away-Natural) since IF-(0,∞) appears to be very promising for large-scale instances due to its ability to maintain relatively low-rank iterates throughout, while also performing well in terms of run time. The results in Table 3 further reinforce the findings from Table 2 concerning the advantages of IF-(0,∞) both in terms of rank of the final iterate as well as run time to achieve the target optimality gap.

[5]The CoGEnT code directly works with the variables Z_{ij}, and thus has large memory requirements. A more efficient implementation of CoGEnT may be able to run on the instances in Table 2 and may also have better performance on the instances in Table 1.

Medium-Large Scale Examples										
			In-Face Extended FW (IF-...)					Away Steps		
			γ_1, γ_2							
Data	**Metric**	**Regular FW**	**1,1**	**0,1**	**0,∞**	**In-Face Opt.**	**Rank Strategy**	**Natural**	**Atomic**	**Fully Corrective FW**
$m = 500, n = 1000, \rho = 0.25$	Time (secs)	137.62	51.95	53.21	18.20	**4.41**	**6.37**	31.55	157.31	39.81
$r = 15, \text{SNR} = 2, \delta = 3.57$	Final Rank (Max Rank)	53 (126)	16 (17)	15 (17)	16 (17)	**17 (19)**	121 (136)	15 (17)	50 (52)	78 (984*)
$m = 500, n = 1000, \rho = 0.25$	Time (secs)	256.08	110.37	110.77	46.07	**6.76**	**7.91**	73.95	322.24	227.50
$r = 15, \text{SNR} = 10, \delta = 4.11$	Final Rank (Max Rank)	41 (128)	15 (17)	15 (17)	16 (17)	**15 (18)**	**18** (140)	16 (17)	48 (48)	81 (971*)
$m = 1500, n = 2000, \rho = 0.05$	Time (secs)	124.76	108.97	113.58	24.75	**11.09**	**12.71**	40.23	60.83	48.76
$r = 15, \text{SNR} = 2, \delta = 6.01$	Final Rank (Max Rank)	169 (210)	15 (18)	16 (17)	**16 (16)**	31 (44)	206 (206)	16 (16)	128 (138)	106 (736*)
$m = 1500, n = 2000, \rho = 0.05$	Time (secs)	>800.01	518.72	496.08	166.01	**21.90**	**31.41**	309.58	407.22	>801.89
$r = 15, \text{SNR} = 10, \delta = 8.94$	Final Rank (Max Rank)	119 (266)	15 (17)	15 (17)	15 (17)	**15 (23)**	**15** (256)	15 (18)	172 (185)	125 (790*)
$m = 2000, n = 2500, \rho = 0.01$	Time (secs)	105.44	45.39	36.47	**23.15**	**20.07**	47.83	30.07	26.92	39.65
$r = 10, \text{SNR} = 4, \delta = 7.92$	Final Rank (Max Rank)	436 (436)	37 (38)	35 (38)	**37 (38)**	67 (107)	430 (430)	37 (39)	245 (276)	238 (502*)
$m = 2000, n = 2500, \rho = 0.05$	Time (secs)	99.84	51.90	48.26	18.79	**6.92**	**6.70**	30.37	89.09	55.11
$r = 10, \text{SNR} = 2, \delta = 5.82$	Final Rank (Max Rank)	68 (98)	10 (11)	10 (11)	**11 (11)**	13 (15)	94 (94)	10 (11)	52 (52)	62 (370*)
$m = 5000, n = 5000, \rho = 0.01$	Time (secs)	251.33	168.66	172.21	64.56	**26.25**	**17.70**	96.79	90.41	350.88
$r = 10, \text{SNR} = 4, \delta = 12.19$	Final Rank (Max Rank)	161 (162)	10 (24)	11 (18)	**11 (20)**	22 (34)	20 (112)	10 (16)	181 (182)	92 (616*)
$m = 5000, n = 7500, \rho = 0.01$	Time (secs)	272.19	107.19	116.58	**52.65**	**54.02**	145.13	107.60	94.96	209.86
$r = 10, \text{SNR} = 4, \delta = 12.19$	Final Rank (Max Rank)	483 (483)	33 (43)	34 (36)	**32 (37)**	63 (123)	476 (476)	36 (42)	229 (298)	204 (331*)

Table 2: Table reports the time required for each method to reach a relative optimality gap of $10^{-2.5}$, the rank of the corresponding final solution, and the maximum rank of the iterates therein (as an indicator of additional memory/computational requirements), for eight single problem instances. Numbers highlighted in boldface indicate the methods that perform well with regard to each criteria, while not performing poorly on run time.
*For this algorithm, we counted the maximum number of atoms instead of the maximum rank.

Note that IF-(0,∞) dominates both Frank-Wolfe and FW-Away-Natural in terms of run time, and dominates Frank-Wolfe in terms of final rank, while it is essentially the same as FW-Away-Natural on the final rank. Also note that FW-Away-Natural generally dominates Frank-Wolfe in terms of both run time and final rank.

MovieLens10M Dataset						
	Frank-Wolfe		**FW-Away-Natural**		**IF-(0,∞)**	
Relative Optimality Gap	Time (mins)	Rank	Time (mins)	Rank	Time (mins)	Rank
$10^{-1.5}$	7.38	103	10.86	52	7.01	44
10^{-2}	28.69	315	23.08	87	14.73	79
$10^{-2.25}$	69.53	461	34.78	113	22.80	107
$10^{-2.5}$	178.54	454	76.64	141	42.24	138

Table 3: CPU time and rank of final solutions for Frank-Wolfe, FW-Away-Natural, and IF-(0,∞) for different relative optimality gaps for the MovieLens10M dataset.

We conclude our computational research with a diagnostic evaluation of the different types of iterations and associated CPU times of different methods. Table 4 presents a detailed breakdown of the types of iterations and other algorithmic diagnostics for different methods applied to the middle grouping of 25 small-scale examples of Table 1. Recall that there are four types of iterations that can arise in the In-Face Extended Frank-Wolfe method, namely types (a), (b), (c), and (d) as exposited in Section 3.7. These types naturally extend to FW-Away-Natural, FW-Away-Atomic and FW-Fully-Corrective, but not to CoGEnT; hence CoGEnT is not included in

our evaluation. Rows 2-5 of Table 4 break down the iterations into the four types, and rows 7-9 report information on the CPU time spent on in-face directions and regular Frank-Wolfe directions. For methods that use standard away steps (IF-(1,1), IF-(0,1), IF-(0,∞), IF-Rank-Strategy, and FW-Away-Natural), most of the time is spent computing regular Frank-Wolfe directions. Indeed, as the bottom row of the table indicates, for four of these methods the average CPU time spent per in-face direction is a mere $2-6\%$ of that spent computing regular Frank-Wolfe directions. IF-Rank-Strategy spends comparatively more time computing the in-face direction because the computational burden of the in-face direction scales with the ranks of the iterates. Also, IF-Optimization spends more time computing in-face directions because solving the proximal gradient algorithm is more expensive than elementary linear optimization. Furthermore, the atom-based methods (FW-Away-Atomic and FW-Fully-Corrective) spend more time computing in face directions because the computational burden scales with the number of atoms and the number of atoms becomes extremely large.

Row 6 of Table 4 reports the final rank and the bound on the final rank from Proposition 2. Very curiously, the bound from Proposition 2 is nearly tight for both IF-(0,∞) and FW-Away-Natural, whereas it is generally very loose otherwise. The tightness of the bounds for these two methods is due to the fact that the different steps taken are almost evenly split between regular Frank-Wolfe steps (type (c)), and steps of type (a) – iterations that go to the boundary of the current minimal face. The former almost always increases the rank by one, whereas the latter always decreases the rank by at least one.

Details of Algorithm Steps Averaged over the 25 Small-Scale Examples with $m = 200$, $n = 400$, $\rho = 0.20$, $r = 15$, SNR $= 4$, and $\delta_{\text{avg}} = 3.82$

		In-Face Extended FW (IF-...)					Away Steps		
		γ_1, γ_2			In-Face	Rank			Fully
Metric	Regular FW	1,1	0,1	0,∞	Opt.	Strategy	Natural	Atomic	Corrective FW
Total Number of Iterations	5368	6691	6605	2744	790	330	2804	1163	2797
Number of Regular FW Steps (Type (c))	5368	2338	2304	1374	396	189	1400	684	1399
Number of Away Steps from the Interior of $\mathcal{B}_{N1}(0,\delta)$ (Type (d))	0	12	13	8	7	1	8	0	0
Number of Interior IF Steps (Type (b))	0	2280	2224	0	325	0	9	434	1355
Number of Boundary IF Steps (Type (a))	0	2062	2063	1362	63	139	1387	45	43
Final Rank (Upper Bound from Proposition 2)	96 (5368)	16 (288)	16 (254)	17 (19)	20 (340)	21 (51)	17 (22)	107 (639)	108 (1356)
Percentage of CPU Time Spent Computing IF Directions	0.00%	4.55%	4.44%	9.93%	16.87%	21.58%	5.63%	39.00%	79.33%
Percentage of CPU Time Spent Computing Regular FW Directions	79.58%	91.94%	92.13%	82.33%	80.88%	37.09%	90.90%	50.99%	12.90%
Avg. IF Comp. Time/Avg. Regular FW Comp. Time	-	0.02	0.02	0.06	0.22	0.68	0.06	0.77	6.31

Table 4: Table reporting the breakdown of types of iterations and other algorithmic diagnostics for different methods applied to middle grouping of 25 small-scale examples of Table 1. Iteration counts might not add up to totals due to independent rounding of the averages.

Summary Conclusions. In addition to its theoretical computational guarantees (Theorem 2, Proposition 1), the In-Face Extended Frank-Wolfe Method (in different versions) shows significant computational advantages in terms of delivering low rank and low run time to compute a target optimality gap. Especially for larger instances, IF-(0,∞) delivers very low rank solutions with reasonable run times. IF-Rank-Strategy delivers the best run times, beating existing methods by a factor of 10 or more. And in the large-scale regime, IF-Optimization generally delivers both low rank and low run times simultaneously, and is usually competitive with the best methods on one or both of rank and run time.

References

[1] A. Beck and S. Shtern. Linearly convergent away-step conditional gradient for non-strongly convex functions. *arXiv preprint arXiv:1504.05002*, 2015.

[2] R. M. Bell and Y. Koren. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[3] J. Bennett and S. Lanning. The Netflix prize. In *Proceedings of KDD Cup and Workshop*, volume 2007, page 35, 2007.

[4] M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30, 2006.

[5] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[6] E. J. Candès and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[7] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[8] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[9] A. L. Chistov and D. Y. Grigor'ev. Complexity of quantifier elimination in the theory of algebraically closed fields. In *Mathematical Foundations of Computer Science 1984*, pages 17–31. Springer, 1984.

[10] M. Fazel. *Matrix rank minimization with applications*. PhD thesis, Stanford University, 2002.

[11] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[12] R. M. Freund and P. Grigas. New analysis and results for the Frank-Wolfe method. *to appear in Mathematical Programming*, 2014.

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

[14] Z. Harchaoui, A. Juditsky, and A. Nemirovski. Conditional gradient algorithms for machine learning. In *NIPS Workshop*, 2012.

[15] Z. Harchaoui, A. Juditsky, and A. Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152(1-2):75–112, 2015.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Verlag, New York, 2009.

[17] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 427–435, 2013.

[18] M. Jaggi and M. Sulovský. A simple algorithm for nuclear norm regularized problems. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 471–478, 2010.

[19] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th annual international conference on machine learning*, pages 457–464. ACM, 2009.

[20] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[21] S. Lacoste-Julien and M. Jaggi. An affine invariant linear convergence analysis for Frank-Wolfe algorithms. Technical report, 2014.

[22] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 496–504. Curran Associates, Inc., 2015.

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

[24] S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

[25] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322, 2010.

[26] B. Mishra, G. Meyer, F. Bach, and R. Sepulchre. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization*, 23(4):2124–2149, 2013.

[27] C. Mu, Y. Zhang, J. Wright, and D. Goldfarb. Scalable robust matrix recovery: Frank-Wolfe meets proximal methods. *arXiv preprint arXiv:1403.7588*, 2014.

[28] J. Pena, D. Rodrguez, and N. Soheili. On the von Neumann and Frank–Wolfe algorithms with away steps. *SIAM Journal on Optimization*, 26(1):499–512, 2016.

[29] N. Rao, P. Shah, and S. Wright. MATLAB code for COnditional Gradient with ENhancement and Truncation (CoGEnT). `http://www.cs.utexas.edu/~nikhilr/`.

[30] N. Rao, P. Shah, and S. Wright. Forward-backward greedy algorithms for atomic norm regularization. *IEEE Transactions on Signal Processing*, 63(21):5798–5811, 2015.

[31] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[32] J. D. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 713–719. ACM, 2005.

[33] W. So. Facial structures of Schatten p-norms. *Linear and Multilinear Algebra*, pages 207–212, 1990.

[34] A. Tewari, P. Ravikumar, and I. S. Dhillon. Greedy algorithms for structurally constrained high dimensional problems. In *Advances in Neural Information Processing Systems 24*, pages 882–890, 2011.

[35] K. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific J. Optimization*, 6:615–640, 2010.

[36] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[37] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. Technical report, May 21, 2008.

[38] P. Wolfe. Convergence theory in nonlinear programming. In J. Abadie, editor, *Integer and Nonlinear Programming*. North-Holland, Amsterdam, 1970.

A Appendix - Remainder of the Proof of Theorem 1

Recall that it remains to prove the following inequality:

$$\frac{1}{f(x_{k+1}) - B_{k+1}} \geq \frac{1}{f(x_k) - B_k} + \frac{1}{2\bar{L}D^2} \quad \text{for all } k \geq 0 \ . \tag{28}$$

Let us fix some simplifying notation. Let $r_k := f(x_k) - B_k \geq 0$ and $G_k := \nabla f(x_k)^T(x_k - \tilde{x}_k) \geq 0$, and note that $B_k \geq B_k^w = f(x_k) - G_k$, so that $G_k \geq r_k \geq 0$ for $k \geq 0$. Also define $C_k := \bar{L}\|\tilde{x}_k - x_k\|^2$, whereby $C_k \leq \bar{L}D^2$ and $\bar{\alpha}_k = \min\left\{\frac{G_k}{C_k}, 1\right\}$ for $k \geq 0$. With this notation (28) can be written as $1/r_{k+1} \geq 1/r_k + 1/(2\bar{L}D^2)$. Substituting $x = x_k$ and $y = x_{k+1} = x_k + \bar{\alpha}_k(\tilde{x}_k - x_k)$ in (8) and using $\bar{L} \geq L$ yields:

$$f(x_{k+1}) \leq f(x_k) + \bar{\alpha}_k \nabla f(x_k)^T(\tilde{x}_k - x_k) + \tfrac{\bar{L}}{2}\bar{\alpha}_k^2\|\tilde{x}_k - x_k\|^2 = f(x_k) - \bar{\alpha}_k G_k + \tfrac{1}{2}\bar{\alpha}_k^2 C_k \ . \qquad (29)$$

Note that if we instead used an exact line-search to determine x_{k+1}, then (29) also holds since in that case we have $f(x_{k+1}) \leq f(x_k + \bar{\alpha}_k(\tilde{x}_k - x_k))$. We now examine two cases depending on the relative magnitudes of G_k and C_k. **Case 1:** $G_k \leq C_k$. In this case $\bar{\alpha}_k = G_k/C_k$, and substituting this value in the right side of (29) yields $f(x_{k+1}) \leq f(x_k) - \frac{(G_k)^2}{2C_k}$, which shows that $f(x_{k+1}) \leq f(x_k)$ as well as $r_{k+1} \leq r_k$, and also yields:

$$r_{k+1} \ = \ f(x_{k+1}) - B_{k+1} \leq f(x_{k+1}) - B_k \leq f(x_k) - \frac{(G_k)^2}{2C_k} - B_k = r_k - \frac{(G_k)^2}{2C_k} \leq r_k - \frac{r_k r_{k+1}}{2C_k} \ ,$$

where the last inequality uses $r_{k+1} \leq r_k \leq G_k$. Dividing the above inequality by $r_{k+1}r_k$ and rearranging yields

$$\frac{1}{r_{k+1}} \geq \frac{1}{r_k} + \frac{1}{2C_k} \geq \frac{1}{r_k} + \frac{1}{2\bar{L}D^2} \ ,$$

where the second inequality above uses $C_k \leq \bar{L}D^2$. This shows that (28) holds in this case.

Case 2: $G_k > C_k$. In this case $\bar{\alpha}_k = 1$. Substituting $x = x_k$ and $y = x_{k+1} = x_k + \bar{\alpha}_k(\tilde{x}_k - x_k) = \tilde{x}_k$ in (29) yields $f(x_{k+1}) \leq f(x_k) - G_k + \frac{1}{2}C_k < f(x_k) - C_k + \frac{1}{2}C_k = f(x_k) - \frac{1}{2}C_k$, which shows that $f(x_{k+1}) < f(x_k)$ as well as $r_{k+1} < r_k$, and also yields:

$$r_{k+1} \ = \ f(x_{k+1}) - B_{k+1} \leq f(x_{k+1}) - B_k \leq f(x_k) - G_k + \tfrac{1}{2}C_k - B_k = r_k - G_k + \tfrac{1}{2}C_k \ , \qquad (30)$$

from which we derive:

$$0 \leq r_{k+1} \leq r_k - G_k + \tfrac{1}{2}C_k < r_k - G_k + \tfrac{1}{2}G_k = r_k - \tfrac{1}{2}G_k \ , \qquad (31)$$

where the last inequality above uses $G_k > C_k$. We now consider two sub-cases, one for $k = 0$ and another sub-case for $k \geq 1$. Let us first consider when $k = 0$. Then

$$G_0 r_0 + G_0 C_0 = G_0 r_0 + \tfrac{1}{2}G_0 C_0 + \tfrac{1}{2}G_0 C_0 \geq (r_0)^2 + \tfrac{1}{2}(C_0)^2 + \tfrac{1}{2}r_0 C_0 \ ,$$

since $G_0 \geq C_0$ and $G_0 \geq r_0$, and now add $r_0 C_0$ to both sides and rearrange to yield:

$$r_0 C_0 \geq r_0 C_0 + (r_0)^2 - G_0 r_0 - G_0 C_0 + \tfrac{1}{2}(C_0)^2 + \tfrac{1}{2}r_0 C_0 = (r_0 - G_0 + \tfrac{1}{2}C_0)(r_0 + C_0) \geq r_1(r_0 + C_0) \ ,$$

where the second inequality uses (30) with $k = 0$. Therefore:

$$\frac{1}{r_1} \geq \frac{r_0 + C_0}{r_0 C_0} = \frac{1}{r_0} + \frac{1}{C_0} \geq \frac{1}{r_0} + \frac{1}{\bar{L}D^2} \ ,$$

which proves (28) for this case for $k = 0$. Last of all, we consider when $k \geq 1$. Taking (31) and dividing by $r_k r_{k+1}$ and rearranging yields:

$$\frac{1}{r_{k+1}} > \frac{1}{r_k} + \frac{G_k}{2r_k r_{k+1}} \geq \frac{1}{r_k} + \frac{1}{2r_{k+1}} \ ,$$

where the second inequality follows since $G_k \geq r_k$. Now notice from (31) that $r_{k+1} \leq r_k - G_k + C_k/2 \leq C_k/2$ since $G_k \geq r_k$. Substituting this last inequality into the right-most term above yields:

$$\frac{1}{r_{k+1}} \geq \frac{1}{r_k} + \frac{1}{C_k} \geq \frac{1}{r_k} + \frac{1}{2C_k} \geq \frac{1}{r_k} + \frac{1}{2\bar{L}D^2} \ ,$$

which shows (28) for this case for $k \geq 1$, and completes the proof. □