Predictive Low Rank Matrix Learning under Partial Observations: Mixed-Projection ADMM

Dimitris Bertsimas[1†] and Nicholas Johnson[2*†]

[1]Sloan School of Management, Massachusetts Institute of Technology, 100 Main Street, Cambridge, 02142, MA, USA.
[2]Operations Research Center, Massachusetts Institute of Technology, 1 Amherst Street, Cambridge, 02142, MA, USA.

*Corresponding author(s). E-mail(s): nagj@mit.edu;
Contributing authors: dbertsim@mit.edu;
†These authors contributed equally to this work.

Abstract

We study the problem of learning a partially observed matrix under the low rank assumption in the presence of fully observed side information that depends linearly on the true underlying matrix. This problem consists of an important generalization of the Matrix Completion problem, a central problem in Statistics, Operations Research and Machine Learning, that arises in applications such as recommendation systems, signal processing, system identification and image denoising. We formalize this problem as an optimization problem with an objective that balances the strength of the fit of the reconstruction to the observed entries with the ability of the reconstruction to be predictive of the side information. We derive a mixed-projection reformulation of the resulting optimization problem and present a strong semidefinite cone relaxation. We design an efficient, scalable alternating direction method of multipliers algorithm that produces high quality feasible solutions to the problem of interest. Our numerical results demonstrate that in the small rank regime ($\boldsymbol{k \leq 10}$), our algorithm outputs solutions that achieve on average $\mathbf{2.3\%}$ lower objective value and $\mathbf{41\%}$ lower $\boldsymbol{\ell_2}$ reconstruction error than the solutions returned by the best performing benchmark method on synthetic data. The runtime of our algorithm is competitive with and often superior to that of the benchmark methods. Our algorithm is able to solve problems with $\boldsymbol{n = 10000}$ rows and $\boldsymbol{m = 10000}$ columns in less than a minute. On large scale real world data, our algorithm produces solutions that achieve $\mathbf{67\%}$ lower out of sample error than benchmark methods in $\mathbf{97\%}$ less execution time.

Keywords: Matrix Completion; Rank; Mixed-Projection; ADMM;

1 Introduction

In many real world applications, we are faced with the problem of recovering a (often large) matrix from a (often small) subset of its entries. This problem, known as Matrix Completion (MC), has gained significant attention due to its broad range of applications in areas such as signal processing [1], system identification [2] and image denoising [3]. The fundamental task in MC is to accurately reconstruct the missing entries of a matrix given a limited number of observed entries. The challenge is particularly pronounced when the number of observed entries is small relatively to the dimension of the matrix, yet this is the common scenario in practice.

One of the most prominent uses of MC is in recommendation systems, where the goal is to predict user preferences for items (e.g., movies, products) based on a partially observed user-item rating matrix [4]. The Netflix Prize competition highlighted the potential of MC techniques, where the objective was to predict missing ratings in a user-movie matrix to improve recommendation accuracy [5]. The success of such systems hinges on the assumption that the underlying rating matrix is low rank, meaning that the preferences of users can be well-approximated by a small number of factors. Indeed, it has been well studied that many real world datasets are low rank [6].

In many practical applications, in addition to a collection of observed matrix entries we additionally have access to auxiliary side information that can be leveraged when performing the reconstruction. For example, in a recommendation system, side information might consist of social network data or item attributes. The vast majority of existing approaches to MC in the presence of side information incorporate the side information by making additional structural restrictions on the reconstructed matrix beyond the usual low rank assumption (see, for example, [7–9]). In this work, we take an alternate approach by assuming that the side information can be well modeled as a linear function of the underlying full matrix. In this setting, the side information can be thought of as labels for a regression problem where the unobserved matrix consists of the regression features. This assumption is in keeping with ideas from the predictive low rank kernel learning literature [10] (note however that low rank kernel learning assumes a fully observed input matrix). One motivation for this approach is that it incorporates the side information in a more flexible manner in contrast to existing approaches that make fixed structural assumptions through an explicit parametrization of the partially observed matrix as a function of the side information. Indeed, the model we present is most natural for the setting of MC with noisy side information in contrast to the setting of perfectly noiseless side information [11, 12] or graph based side information [13, 14].

Formally, let $\Omega \subseteq [n] \times [m]$ denote a collection of revealed entries of a partially observed matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, let $\boldsymbol{Y} \in \mathbb{R}^{n \times d}$ denote a matrix of side information and let k denote a specified target rank. We consider the problem given by

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times m}, \boldsymbol{\alpha} \in \mathbb{R}^{m \times d}} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2 + \lambda \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\alpha}\|_F^2 + \gamma \|\boldsymbol{X}\|_\star \quad \text{s.t.} \quad \operatorname{rank}(\boldsymbol{X}) \leq k, \tag{1}$$

where $\lambda, \gamma > 0$ are hyperparameters that in practice can either take a default value or can be cross-validated by minimizing a validation metric [15] to obtain strong out-of-sample performance [16]. We assume that the ground truth matrix $\boldsymbol{A}$ has low rank and that the side information can be well approximated as $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\alpha}+\boldsymbol{N}$ for some weighting matrix $\boldsymbol{\alpha}$ and noise matrix $\boldsymbol{N}$. The first term in the objective function of (1) measures how well the observed entries of the unknown matrix are fit by the estimated matrix $\boldsymbol{X}$, the second term of the objective function measures how well the side information $\boldsymbol{Y}$ can be represented as a linear function of the estimated matrix $\boldsymbol{X}$ and the final term of the objective is a regularization term. To the best of our knowledge, Problem (1) has not previously been directly studied despite its very natural motivation.

1.1 Contribution and Structure

In this paper, we tackle (1) by developing novel mixed-projection optimization techniques [17]. We show that solving (1) is equivalent to solving an appropriately defined robust optimization problem. We develop an exact reformulation of (1) by combining a parametrization of the $\boldsymbol{X}$ decision variable as the product of two low rank factors with the introduction of a projection matrix to model the column space of $\boldsymbol{X}$. We derive a semidefinite cone convex relaxation for our mixed-projection reformulation and we present an efficient, scalable alternating direction method of multipliers (ADMM) algorithm that produces high quality feasible solutions to (1). Our numerical results show that across all synthetic data experiments in the small rank regime ($k \leq 10$), our algorithm outputs solutions that achieve on average 2.3% lower objective value in (1) and 41% lower ℓ_2 reconstruction error than the solutions returned by the best performing benchmark method on a per experiment basis. For the 5 synthetic data experiments with $k > 15$, the only benchmark that returns a solution with superior quality than that returned by our algorithm takes on average 3 times as long to execute. The runtime of our algorithm is competitive with and often superior to that of the benchmark methods. Our algorithm is able to solve problems with $n = 10000$ rows and $m = 10000$ columns in less than a minute. On large scale real world data, our algorithm produces solutions that achieve 67% lower out of sample error than benchmark methods in 97% less execution time.

The rest of the paper is laid out as follows. In Section 2, we review previous work that is closely related to (1). In Section 3, we study (1) under a robust optimization lens and investigate formulating (1) as a two stage optimization problem where the inner stage is a regression problem that can be solved in closed form. We formulate (1) as a mixed-projection optimization problem in Section 4 and present a natural convex relaxation. In Section 5, we present and rigorously study our ADMM algorithm. Finally, in Section 6 we investigate the performance of our algorithm against benchmark methods on synthetic and real world data.

Notation:

We let nonbold face characters such as b denote scalars, lowercase bold faced characters such as $\boldsymbol{x}$ denote vectors, uppercase bold faced characters such as $\boldsymbol{X}$ denote matrices, and calligraphic uppercase characters such as $\mathcal{Z}$ denote sets. We let $[n]$ denote the set of running indices $\{1, ..., n\}$. We let $\mathbf{0}_n$ denote an n-dimensional vector of all 0's,

$\mathbf{0}_{n\times m}$ denote an $n \times m$-dimensional matrix of all 0's, and $\boldsymbol{I}_n$ denote the $n \times n$ identity matrix. We let $\mathcal{S}^n$ denote the cone of $n \times n$ symmetric matrices and $\mathcal{S}^n_+$ denote the cone of $n \times n$ positive semidefinite matrices.

2 Literature Review

In this section, we review a handful of notable approaches from the literature that have been employed to solve MC and to solve general low rank optimization problems. As an exhaustive literature review of MC methods is outside of the scope of this paper, we focus our review on a handful of well studied approaches which we will employ as benchmark methods in this work. We additionally give an overview of the ADMM algorithmic framework which is of central relevance to this work. For a more detailed review of the MC literature, we refer the reader to [4] and [18].

2.1 Matrix Completion Methods

2.1.1 Iterative-SVD

Iterative-SVD is an expectation maximization style algorithm [19] that generates a solution to the MC problem by iteratively computing a singular value decomposition (SVD) of the current iterate and estimating the missing values by performing a regression against the low rank factors returned by SVD [20]. This is one of a handful of methods in the literature that leverage the SVD as their primary algorithmic workhorse [21, 22]. Concretely, given a partially observed matrix $\{X_{ij}\}_{(i,j)\in\Omega}$ where $\Omega \subseteq [n] \times [m]$ and a target rank $k \in \mathbf{N}_+$, Iterative-SVD proceeds as follows:

1. Initialize the iteration count $t \leftarrow 0$ and initialize missing entries of $X_{ij}, (i,j) \notin \Omega$ with the row average $X_{ij} = \frac{\sum_{l:(i,l)\in\Omega} X_{il}}{|\{l:(i,l)\in\Omega\}|}$.
2. Compute a rank k SVD $\boldsymbol{X}_t = \boldsymbol{U}_t\boldsymbol{\Sigma}_t\boldsymbol{V}_t^T$ of the current iterate where $\boldsymbol{U}_t \in \mathbb{R}^{n\times k}, \boldsymbol{\Sigma}_t \in \mathbb{R}^{k\times k}, \boldsymbol{V}_t \in \mathbb{R}^{m\times k}$.
3. For each $(i,j) \notin \Omega$, estimate the missing value $(\boldsymbol{X}_{t+1})_{ij}$ by regressing all other entries in row i against all except the j^{th} row of $\boldsymbol{V}_t$. Concretely, letting $\tilde{\boldsymbol{x}} = (\boldsymbol{X}_t)_{i,\star\setminus j} \in \mathbb{R}^{m-1}$ denote the column vector consisting of the i^{th} row of $\boldsymbol{X}_t$ excluding the j^{th} entry, letting $\tilde{\boldsymbol{V}} = (\boldsymbol{V}_t)_{\star\setminus j,\star} \in \mathbb{R}^{(m-1)\times k}$ denote the matrix formed by eliminating the j^{th} row from $\boldsymbol{V}_t$ and letting $\hat{\boldsymbol{v}} = (\boldsymbol{V}_t)_{j,\star} \in \mathbb{R}^k$ denote the column vector consisting of the j^{th} row of $\boldsymbol{V}_t$, we set $(\boldsymbol{X}_{t+1})_{ij} = \hat{\boldsymbol{v}}^T(\tilde{\boldsymbol{V}}^T\tilde{\boldsymbol{V}})^{-1}\tilde{\boldsymbol{V}}^T\tilde{\boldsymbol{x}}$.
4. Terminate if the total change between $\boldsymbol{X}_t$ and $\boldsymbol{X}_{t+1}$ is less than 0.01. Otherwise, increment t and return to Step 2.

2.1.2 Soft-Impute

Soft-Impute is a convex relaxation inspired algorithm that leverages the nuclear norm as a low rank inducing regularizer [23]. This approach is one of a broad class of methods that tackle MC from a nuclear norm minimization lens [24–26]. Seeking a reconstruction with minimum nuclear norm is typically motivated by the observation that the nuclear norm ball given by $\mathcal{B} = \{\boldsymbol{X} \in \mathbb{R}^{n\times n} : \|\boldsymbol{X}\|_\star \leq k\}$ is the convex hull of

the nonconvex set $\mathcal{X} = \{\boldsymbol{X} \in \mathbb{R}^{n\times n} : \text{rank}(\boldsymbol{X}) \leq k, \|\boldsymbol{X}\|_\sigma \leq 1\}$, where $\|\cdot\|_\sigma$ denotes the spectral norm. Moreover, several conditions have been established under which nuclear norm minimization methods are guaranteed to return the ground truth matrix [25, 26] though such conditions tend to be strong and hard to verify in practice. Soft-Impute proceeds by iteratively replacing the missing elements of the matrix with those obtained from a soft thresholded low rank singular value decomposition. Accordingly, similarly to Iterative-SVD, Soft-Impute relies on the computation of a low rank SVD as the primary algorithmic workhorse. The approach relies on the result that for an arbitrary matrix $\boldsymbol{X}$, the solution of the problem $\min_{\boldsymbol{Z}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Z}\|_F^2 + \lambda\|\boldsymbol{Z}\|_\star$ is given by $\hat{\boldsymbol{Z}} = S_\lambda(\boldsymbol{X})$ where $S_\lambda(\cdot)$ denotes the soft-thresholding operation [27]. Explicitly, Soft-Impute proceeds as follows for a given regularization parameter $\lambda > 0$ and termination threshold $\epsilon > 0$:

1. Initialize the iteration count $t \leftarrow 0$ and initialize $\boldsymbol{Z}_t = \boldsymbol{0}_{n\times m}$.
2. Compute $\boldsymbol{Z}_{t+1} = S_\lambda(P_\Omega(\boldsymbol{X}) + P_\Omega^\perp(\boldsymbol{Z}_t))$ where $P_\Omega(\cdot)$ denotes the operation that projects onto the revealed entries of $\boldsymbol{X}$ while $P_\Omega^\perp(\cdot)$ denotes the operation that projects onto the missing entries of $\boldsymbol{X}$.
3. Terminate if $\frac{\|\boldsymbol{Z}_t - \boldsymbol{Z}_{t+1}\|_F^2}{\|\boldsymbol{Z}_t\|_F^2}$. Otherwise, increment t and return to Step 2.

2.1.3 Fast-Impute

Fast-Impute is a projected gradient descent approach to MC that has desirable global convergence properties [9]. Fast-Impute belongs to the broad class of methods that solve MC by factorizing the target matrix as $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$ where $\boldsymbol{U} \in \mathbb{R}^{n\times k}, \boldsymbol{V} \in \mathbb{R}^{m\times k}$ and performing some variant of gradient descent (or alternating minimization) on the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ [28–31]. We note that we leverage this common factorization in the approach to (1) presented in this work. Gradient descent based methods have shown great success. Despite the non-convexity of the factorization, it has been shown that in many cases gradient descent and its variants will nevertheless converge to a globally optimal solution [9, 32–35]. Fast-Impute takes the approach of expressing $\boldsymbol{U}$ as a closed form function of $\boldsymbol{V}$ after performing the facorization and directly performs projected gradient descent updates on $\boldsymbol{V}$ with classic Nesterov acceleration [36]. Moreover, to enhance scalability of their method, [9] design a stochastic gradient extension of Fast-Impute that estimates the gradient at each update step by only considering a sub sample of the rows and columns of the target matrix. Finally, given side information $\boldsymbol{Y} \in \mathbb{R}^{m\times p}$,[9] present a version of their algorithm that incorporates the side information by factorizing the target matrix as $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}^T\boldsymbol{Y}^T$ for $\boldsymbol{U} \in \mathbb{R}^{n\times k}, \boldsymbol{S} \in \mathbb{R}^{p\times k}$.

2.2 Low Rank Optimization Methods

2.2.1 ScaledGD

ScaledGD is a highly performant method to obtain strong solutions to low rank matrix estimation problems that take the following form:

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times m}} f(\boldsymbol{X}) = \frac{1}{2}\|\mathcal{A}(\boldsymbol{X}) - \boldsymbol{y}\|_2^2 \quad \text{s.t. } \text{rank}(\boldsymbol{X}) \leq k,$$

where $\mathcal{A}(\cdot) : \mathbb{R}^{n\times m} \to \mathbb{R}^l$ models some measurement process and we have $\boldsymbol{y} \in \mathbb{R}^l$ [37]. ScaledGD proceeds by factorizing the target matrix as $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$ and iteratively performing gradient updates on the low rank factors $\boldsymbol{U}, \boldsymbol{V}$ after preconditioning the gradients with an adaptive matrix that is efficient to compute. Doing so yields a linear convergence rate that is notably independent of the condition number of the low rank matrix. In so doing, ScaledGD combines the desirable convergence rate of alternating minimization with the desirable low per-iteration cost of gradient descent. Explicitly, letting $\mathcal{L}(\boldsymbol{U}, \boldsymbol{V}) = f(\boldsymbol{U}\boldsymbol{V}^T)$, ScaledGD updates the low rank factors as:

$$\boldsymbol{U}_{t+1} \leftarrow \boldsymbol{U}_t - \eta\nabla_{\boldsymbol{U}}\mathcal{L}(\boldsymbol{U}_t, \boldsymbol{V}_t)(\boldsymbol{V}^T\boldsymbol{V})^{-1},$$

$$\boldsymbol{V}_{t+1} \leftarrow \boldsymbol{V}_t - \eta\nabla_{\boldsymbol{V}}\mathcal{L}(\boldsymbol{U}_t, \boldsymbol{V}_t)(\boldsymbol{U}^T\boldsymbol{U})^{-1},$$

where $\eta > 0$ denotes the step size.

2.2.2 Mixed-Projection Conic Optimization

Mixed-projection conic optimization is a recently proposed modelling and algorithmic framework designed to tackle a broad class of matrix optimization problems [17, 38]. Specifically, this approach considers problems that have the following form:

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times m}} \langle \boldsymbol{C}, \boldsymbol{X}\rangle + \lambda \cdot \text{rank}(\boldsymbol{X}) + \Omega(\boldsymbol{X}) \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{X} = \boldsymbol{B},\ \text{rank}(\boldsymbol{X}) \leq k,\ \boldsymbol{X} \in \mathcal{K}, \tag{2}$$

where $\boldsymbol{C} \in \mathbb{R}^{n\times m}$ is a cost matrix, $\lambda > 0$, $k \in \mathbb{N}_+$, $\boldsymbol{A} \in \mathbb{R}^{l\times n}, \boldsymbol{B} \in \mathbb{R}^{l\times m}, \mathcal{K}$ denotes a proper cone in the sense of [39] and $\Omega(\cdot)$ is a Frobenius norm regularization function or a spectral norm regularization function of the input matrix. The main workhorse of mixed-projection conic optimization is the use of a projection matrix to cleverly model the rank terms in (2). This can be viewed as the matrix generalization of using binary variables to model the sparsity of a vector in mixed-integer optimization. [17] show that for an arbitrary matrix $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, we have

$$\text{rank}(\boldsymbol{X}) \leq k \iff \exists \boldsymbol{P} \in \mathbb{S}^n : \boldsymbol{P}^2 = \boldsymbol{P}, \text{Tr}(\boldsymbol{P}) \leq k, \boldsymbol{X} = \boldsymbol{P}\boldsymbol{X}.$$

Introducing projection matrices allows the rank functions to be eliminated from (2) at the expense of introducing non convex quadratic equality constraints. From here, most existing works that leverage mixed-projection conic optimization have either focused on obtaining strong semidefinite based convex relaxations [17, 38] or have focused

on obtaining certifiably optimal solutions for small and moderately sized problem instances [40, 41]. In this work, we leverage the mixed-projection framework to scalably obtain high quality solutions to large problem instances.

2.3 Alternating Direction Method of Multipliers

Alternating direction method of multipliers (ADMM) is an algorithm that was originally designed to solve linearly constrained convex optimization problems of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^n,\boldsymbol{z}\in\mathbb{R}^m} \quad f(\boldsymbol{x})+g(\boldsymbol{z}) \quad \text{s.t. } \boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}=\boldsymbol{c}, \tag{3}$$

where we have $\boldsymbol{A}\in\mathbb{R}^{l\times n}, \boldsymbol{B}\in\mathbb{R}^{l\times m}, \boldsymbol{c}\in\mathbb{R}^l$ and the functions f and g are assumed to be convex [42]. The main benefit of ADMM is that it can combine the decomposition benefits of dual ascent with the desirable convergence properties of the method of multipliers. Letting $\boldsymbol{y}\in\mathbb{R}^l$ denote the dual variable, the augmented Lagrangian of (3) is given by

$$\mathcal{L}^A(\boldsymbol{x},\boldsymbol{z},\boldsymbol{y}) = f(\boldsymbol{x})+g(\boldsymbol{z})+\boldsymbol{y}^T(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}-\boldsymbol{c})+\frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}-\boldsymbol{c}\|_2^2,$$

where $\rho>0$ is the augmented Lagrangian parameter. ADMM then proceeds by iteratively updating the primal variable $\boldsymbol{x}$, updating the primal variable $\boldsymbol{z}$ and taking a gradient ascent step on the dual variable $\boldsymbol{y}$. Explicitly, ADMM consists of the following updates:

1. $\boldsymbol{x}_{t+1}\leftarrow \arg\min_{\boldsymbol{x}} \mathcal{L}^A(\boldsymbol{x},\boldsymbol{z}_t,\boldsymbol{y}_t)$,
2. $\boldsymbol{z}_{t+1}\leftarrow \arg\min_{\boldsymbol{z}} \mathcal{L}^A(\boldsymbol{x}_{t+1},\boldsymbol{z},\boldsymbol{y}_t)$,
3. $\boldsymbol{y}_{t+1}\leftarrow \boldsymbol{y}_t+\rho(\boldsymbol{A}\boldsymbol{x}_{t+1}+\boldsymbol{B}\boldsymbol{z}_{t+1}-\boldsymbol{c})$.

Under very mild regularity conditions on f, g and $\mathcal{L}^A$, it is well known that ADMM is guaranteed to produce a sequence of primal iterates that converges to the optimal value of (3) and a sequence of dual iterates that converge to the optimal dual variable (note that there is no guarantee of primal variable convergence) [42]. Importantly, although ADMM was originally designed for linearly constrained convex optimization, it has often been applied to non convex optimization problems and yielded empirically strong results [43]. This observation has motivated work to explore the theoretical convergence behavior of ADMM and its variants on specific classes of non convex optimization problems [44–46].

3 Formulation Properties

In this section, we rigorously investigate certain key features of (1). Specifically, we establish an equivalence between (1) and an appropriately defined robust optimization problem. Moreover, we illustrate that (1) can be reduced to an optimization problem over only $\boldsymbol{X}$ and establish that the resulting objective function is not convex, not concave and non-smooth. Finally, we study how efficient evaluations of the reduced problem objective function can be performed.

3.1 Equivalence Between Regularization and Robustness

Real-world datasets frequently contain inaccuracies and missing values, which hinder the ability of machine learning models to generalize effectively to new data when these inconsistencies are not appropriately modeled. Consequently, robustness is a crucial quality for machine learning models, both in theory and application [47, 48]. In this section, we show that our regularized problem (1) can be viewed as a robust optimization (RO) problem. This finding justifies the inclusion of the nuclear norm regularization term in (1) and is in a similar flavor as known results from the robust optimization literature in the case of vector [49] and matrix [40] problems. Note that the equivalence presented in Proposition 1 follows from the well studied duality of the nuclear norm and the spectral norm.

Proposition 1. *Problem* (1) *is equivalent to the following robust optimization problem:*

$$\min_{\substack{\boldsymbol{X}\in\mathbb{R}^{n\times m}\\ \boldsymbol{\alpha}\in\mathbb{R}^{m\times d}}} \max_{\boldsymbol{\Delta}\in\mathcal{U}} \sum_{(i,j)\in\Omega} (X_{ij}-A_{ij})^2 + \lambda\|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\alpha}\|_F^2 + \langle \boldsymbol{X},\boldsymbol{\Delta}\rangle \quad \textit{s.t. rank}(\boldsymbol{X})\le k_0, \tag{4}$$

where $\mathcal{U}=\{\boldsymbol{\Delta}\in\mathbb{R}^{n\times m} : \|\boldsymbol{\Delta}\|_\sigma \le \gamma\}$.

Proof. To establish this result, it suffices to argue that $\max_{\boldsymbol{\Delta}\in\mathcal{U}}\langle \boldsymbol{X},\boldsymbol{\Delta}\rangle = \gamma\|\boldsymbol{X}\|_\star$. This equivalence follows immediately from the fact that the nuclear norm is dual to the spectral norm. So as to keep this manuscript self contained, we present a proof of this equivalence below.

Consider any matrix $\bar{\boldsymbol{\Delta}}\in\mathbb{R}^{n\times m}$ such that $\|\bar{\boldsymbol{\Delta}}\|_\sigma \le \gamma$. Let $\boldsymbol{X}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be a singular value decomposition of $\boldsymbol{X}$ where we let $r=\mathrm{rank}(\boldsymbol{X})$ and we have $\boldsymbol{U}\in\mathbb{R}^{n\times r}, \boldsymbol{\Sigma}\in\mathbb{R}^{r\times r}, \boldsymbol{V}\in\mathbb{R}^{m\times r}$. We have

$$\begin{aligned}\langle \boldsymbol{X},\bar{\boldsymbol{\Delta}}\rangle &= \mathrm{Tr}(\bar{\boldsymbol{\Delta}}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T) = \mathrm{Tr}(\boldsymbol{V}^T\bar{\boldsymbol{\Delta}}^T\boldsymbol{U}\boldsymbol{\Sigma}) = \langle \boldsymbol{U}^T\bar{\boldsymbol{\Delta}}\boldsymbol{V},\boldsymbol{\Sigma}\rangle = \sum_{i=1}^r \Sigma_{ii}(\boldsymbol{U}^T\bar{\boldsymbol{\Delta}}\boldsymbol{V})_{ii}\\ &= \sum_{i=1}^r \Sigma_{ii}U_i^T\bar{\boldsymbol{\Delta}}V_i \le \sum_{i=1}^r \Sigma_{ii}\sigma_1(\bar{\boldsymbol{\Delta}}) \le \gamma\sum_{i=1}^r\Sigma_{ii} = \gamma\|\boldsymbol{X}\|_\star,\end{aligned}$$

where we have used the fact that $\boldsymbol{\Sigma}$ is a diagonal matrix and the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ have unit length. Thus, we have shown that $\gamma\|\boldsymbol{X}\|_\star$ is an upper bound for $\max_{\boldsymbol{\Delta}\in\mathcal{U}}\langle \boldsymbol{X},\boldsymbol{\Delta}\rangle$. To show that the upper bound is always achieved, consider the matrix $\tilde{\boldsymbol{\Delta}}=\gamma\boldsymbol{U}\boldsymbol{V}^T\in\mathbb{R}^{n\times m}$ where $\boldsymbol{U}$ and $\boldsymbol{V}$ are taken from a singular value decomposition of $\boldsymbol{X}$. Observe that

$$\|\tilde{\boldsymbol{\Delta}}\|_\sigma = \gamma\|\boldsymbol{U}\boldsymbol{V}^T\|_\sigma \le \gamma \implies \tilde{\boldsymbol{\Delta}}\in\mathcal{U}.$$

We conclude by noting that

$$\langle \boldsymbol{X},\tilde{\boldsymbol{\Delta}}\rangle = Tr(\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\gamma\boldsymbol{U}\boldsymbol{V}^T) = \gamma Tr(\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{U}) = \gamma\mathrm{Tr}(\boldsymbol{I}\boldsymbol{\Sigma}\boldsymbol{I}) = \gamma\|\boldsymbol{X}\|_\star.$$

□

Proposition 1 implies that solving the nuclear norm regularized (1) is equivalent to solving an unregularized robust optimization problem that protects against adversarial perturbations that are bounded in spectral norm. This result is not surprising given the duality of norms, yet is nevertheless insightful.

3.2 A Partial Minimization

Let $g(\boldsymbol{X}, \boldsymbol{\alpha})$ denote the objective function of (1). Note that $g(\boldsymbol{X}, \boldsymbol{\alpha})$ is bi-convex in $(\boldsymbol{X}, \boldsymbol{\alpha})$ but is not jointly convex due to the product $\boldsymbol{X}\boldsymbol{\alpha}$. Observe that we can simplify (1) by performing a partial minimization in $\boldsymbol{\alpha}$. For any $\boldsymbol{X}$, the problem in $\boldsymbol{\alpha}$ requires finding the unconstrained minimum of a convex quadratic function. The gradient of g with respect to $\boldsymbol{\alpha}$ is given by $\nabla_{\boldsymbol{\alpha}} g(\boldsymbol{X}, \boldsymbol{\alpha}) = 2\lambda \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{Y})$. Setting $\nabla_{\boldsymbol{\alpha}} g(\boldsymbol{X}, \boldsymbol{\alpha})$ to 0 yields $\boldsymbol{\alpha}^\star = (\boldsymbol{X}^T\boldsymbol{X})^\dagger \boldsymbol{X}^T\boldsymbol{Y}$ as a minimizer of g over $\boldsymbol{\alpha}$. Note that $\boldsymbol{M}^\dagger$ denotes the pseudo-inverse of a (possibly rank deficient) square matrix $\boldsymbol{M} \in \mathbb{R}^{l \times l}$. Specifically, letting $r = \operatorname{rank}(\boldsymbol{M})$ and $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be a singular value decomposition of $\boldsymbol{M}$ with $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{l \times r}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{l \times l}$, we have $\boldsymbol{M}^\dagger = \boldsymbol{U}\boldsymbol{\Sigma}^{-1}\boldsymbol{V}^T$. Letting $f(\boldsymbol{X})$ correspond to the partially minimized objective function of (1), we have

$$
\begin{aligned}
f(\boldsymbol{X}) = \min_{\boldsymbol{\alpha}} g(\boldsymbol{X}, \boldsymbol{\alpha}) &= \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 + \lambda \|(\boldsymbol{I}_n - \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^\dagger \boldsymbol{X}^T)\boldsymbol{Y}\|_F^2 + \gamma \|\boldsymbol{X}\|_\star \\
&= \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 + \lambda \operatorname{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^\dagger \boldsymbol{X}^T)\boldsymbol{Y}\big) + \gamma \|\boldsymbol{X}\|_\star.
\end{aligned}
$$

We note that $\boldsymbol{\alpha}^\star$ corresponds to the well studied ordinary least squares solution. When $\boldsymbol{X}^T\boldsymbol{X}$ has full rank, $\boldsymbol{\alpha}$ is the unique minimizer of g. If $\boldsymbol{X}^T\boldsymbol{X}$ is rank deficient, $\boldsymbol{\alpha}^\star$ corresponds to the minimizer with minimum norm.

Though we have simplified the objective function of (1), $f(\boldsymbol{X})$ is not a particularly well behaved function. We formalize this statement in Proposition (2).

Proposition 2. *The function $f(\boldsymbol{X})$ is in general neither convex nor concave and is non-smooth.*

Proof. To illustrate that $f(\boldsymbol{X})$ is in general neither convex nor concave, suppose that $\Omega = \emptyset$, $n = 2$ and $m = d = \lambda = \gamma = 1$. In this setting, we have $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{2\times 1}$. Assuming that $\boldsymbol{x} \neq \boldsymbol{0}_2$, we can write the objective function as

$$
\begin{aligned}
f(\boldsymbol{x}) &= \operatorname{Tr}(\boldsymbol{y}^T(\boldsymbol{I}_2 - \boldsymbol{x}(\boldsymbol{x}^T\boldsymbol{x})^{-1}\boldsymbol{x}^T)\boldsymbol{y}) + \|\boldsymbol{x}\|_\star \\
&= \boldsymbol{y}^T\boldsymbol{y} - \frac{\boldsymbol{y}^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{y}}{\boldsymbol{x}^T\boldsymbol{x}} + \|\boldsymbol{x}\|_2 \\
&= \boldsymbol{y}^T\boldsymbol{y} - \frac{(\boldsymbol{y}^T\boldsymbol{x})^2}{\boldsymbol{x}^T\boldsymbol{x}} + \sqrt{\boldsymbol{x}^T\boldsymbol{x}}.
\end{aligned}
$$

For $\boldsymbol{x} = \boldsymbol{0}_2$, the objective value $f(\boldsymbol{0}_2)$ is equal to $\boldsymbol{y}^T\boldsymbol{y}$. Let $\boldsymbol{y} = \boldsymbol{1}_2$ and consider the line in $\boldsymbol{R}^2$ defined by $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^2 : x_2 = x_1 + 1\}$. The restriction of $f(\boldsymbol{x})$ to the line

defined by $\mathcal{X}$ is a univariate function given by

$$f_{\mathcal{X}}(t) = 2 - \frac{(2t+1)^2}{2t^2+2t+1} + \sqrt{2t^2+2t+1},$$

where $t \in \mathbb{R}$ is a dummy variable. Observe that we have $f_{\mathcal{X}}(-1) = f_{\mathcal{X}}(0) = 2, f_{\mathcal{X}}(-0.5) = 2+\frac{\sqrt{2}}{2}$ and $f_{\mathcal{X}}(-4) = f_{\mathcal{X}}(3) = 5.04$. Thus, the point $(-0.5, f_{\mathcal{X}}(0.5))$ lies above the chord connecting $(-1, f_{\mathcal{X}}(-1))$ and $(0, f_{\mathcal{X}}(0))$, so $f_{\mathcal{X}}(t)$ is not a convex function. Moreover, the point $(-1, f_{\mathcal{X}}(-1))$ lies below the chord connecting $(-4, f_{\mathcal{X}}(-4))$ and $(-0.5, f_{\mathcal{X}}(-0.5))$, so $f_{\mathcal{X}}(t)$ is not a concave function. Since a function is convex (respectively concave) if and only if its restriction to every line is convex (respectively concave), we have established that $f(\boldsymbol{X})$ is neither convex nor concave since $f_{\mathcal{X}}(t)$ is neither convex nor concave. To conclude the proof of Proposition 2, note that the non-smooth property of $f(\boldsymbol{X})$ follows immediately from the non-smooth property of the nuclear norm function. □

Although the above closed form partial minimization in $\boldsymbol{\alpha}$ eliminates $m \times d$ variables form (1), this comes at the expense of introducing a $m \times m$ matrix pseudo-inverse term into the objective function which can be computationally expensive to evaluate. Efficient evaluation of an objective function is crucial in many optimization problems to quickly measure solution quality. A plethora of modern optimization techniques require iterative objective function evaluations. As a result, the computational cost of evaluating an objective function can quickly become the bottleneck of an algorithm's complexity. Directly evaluating $f(\boldsymbol{X})$ naively requires $O(|\Omega|)$ operations for the first term, $O(m^2n + m^3 + n^2d)$ for the second term (forming the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is $O(m^2n)$, taking the pseudo-inverse is $O(m^3)$, computing the products involving $\boldsymbol{Y}$ is $O(n^2d)$) and requires $O(mn\min(m,n))$ for the third term (the nuclear norm can be evaluated by computing a singular value decomposition of $\boldsymbol{X}$). We observe that computing the second term of $f(\boldsymbol{X})$ involving the pseudo-inverse dominates the complexity calculation. Indeed, the overall complexity of evaluating $f(\boldsymbol{X})$ naively is $O(m^2n + m^3 + n^2d)$.

Fortunately, it is possible to make evaluations of $f(\boldsymbol{X})$ without explicitly forming the product $\boldsymbol{X}^T\boldsymbol{X}$ or taking a pseudo-inverse. Proposition 3 illustrates that it suffices (in terms of computational complexity) to take a singular value decomposition of $\boldsymbol{X}$. Moreover, a large class of optimization algorithms require only function evaluations for feasible solutions. If we consider only those values of $\boldsymbol{X}$ that are feasible to (1), it is sufficient (in terms of computational complexity) to take a rank k truncated singular value decomposition of $\boldsymbol{X}$ to make functions evaluations of $f(\boldsymbol{X})$.

Proposition 3. *The function $f(\boldsymbol{X})$ can equivalently be written as*

$$f(\boldsymbol{X}) = \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 + \lambda Tr\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{U}\boldsymbol{U}^T)\boldsymbol{Y}\big) + \gamma\|\boldsymbol{X}\|_\star,$$

where $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is a singular value decomposition of $\boldsymbol{X}$ where we let $r = rank(\boldsymbol{X})$ and we have $\boldsymbol{U} \in \mathbb{R}^{n\times r}, \boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}, \boldsymbol{V} \in \mathbb{R}^{m\times r}$.

Proof. To establish the result, it suffices to show that

$$\mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^\dagger\boldsymbol{X}^T\boldsymbol{Y}\big) = \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{Y}\big).$$

Let $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be a singular value decomposition of $\boldsymbol{X}$ where $r = \mathrm{rank}(\boldsymbol{X})$ and $\boldsymbol{U} \in \mathbb{R}^{n\times r}, \boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}, \boldsymbol{V} \in \mathbb{R}^{m\times r}$. Observe that

$$\begin{aligned}
\mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^\dagger\boldsymbol{X}^T\boldsymbol{Y}\big) &= \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T(\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^\dagger\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{Y}\big)\\
&= \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T(\boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T)^\dagger\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{Y}\big)\\
&= \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}^{-2}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{Y}\big)\\
&= \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-2}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{Y}\big)\\
&= \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{Y}\big),
\end{aligned}$$

where we have repeatedly invoked the property that $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_r$. □

In light of Proposition 3, evaluating $f(\boldsymbol{X})$ for feasible solutions still requires $O(|\Omega|)$ operations for the first term, but the second term can be evaluated using $O(kn(m+d))$ operations (performing a truncated singular value decomposition is $O(knm)$ and computing the products involving $\boldsymbol{Y}$ is $O(knd)$) and the third term can be evaluated using $O(knm)$ operations (by performing a truncated singular value decomposition) for an overall complexity of $O(kn(m+d))$. This is significantly less expensive than the $O(m^2n + m^3 + n^2d)$ complexity of naive direct evaluation of $f(\boldsymbol{X})$ introduced previously.

4 An Exact Mixed-Projection Formulation

In this section, we reformulate (1) as a mixed-projection optimization problem and further reduce the dimension of the resulting problem in a commonly studied manner by parameterizing $\boldsymbol{X}$ as the matrix product of two low dimensional matrices. Thereafter, we illustrate how to employ the matrix generalization of the perspective relaxation [17, 38, 40, 50] to construct a convex relaxation of (1).

We first note that given the result of Section 3.2, we can rewrite (1) as an optimization problem only over $\boldsymbol{X}$ as follows:

$$\begin{aligned}
\min_{\boldsymbol{X}\in\mathbb{R}^{n\times m}} \quad & \sum_{(i,j)\in\Omega}(X_{ij} - A_{ij})^2 + \lambda\mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^\dagger\boldsymbol{X}^T)\boldsymbol{Y}\big) + \gamma\|\boldsymbol{X}\|_\star \\
\text{s.t.} \quad & \mathrm{rank}(\boldsymbol{X}) \leq k.
\end{aligned} \tag{5}$$

Observe that the matrix $\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^\dagger\boldsymbol{X}^T$ is the linear transformation that projects vectors onto the subspace spanned by the columns of the matrix $\boldsymbol{X}$. Drawing on ideas presented in [17, 38, 40], we introduce an orthogonal projection matrix $\boldsymbol{P} \in \mathcal{P}_k$ to model the column space of $\boldsymbol{X}$ where $\mathcal{P}_\eta = \{\boldsymbol{P} \in \mathcal{S}^n : \boldsymbol{P}^2 = \boldsymbol{P}, \mathrm{tr}(\boldsymbol{P}) \leq \eta\}$ for $\eta \geq 0$. We can express the desired relationship between $\boldsymbol{P}$ and $\boldsymbol{X}$ as $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{X}$ since

projecting a matrix onto its own column space leaves the matrix unchanged. This gives the following reformulation of (1):

$$\begin{aligned} \min_{\boldsymbol{X}\in\mathbb{R}^{n\times m},\boldsymbol{P}\in\mathbb{R}^{n\times n}} \quad & \sum_{(i,j)\in\Omega}(X_{ij}-A_{ij})^2+\lambda\mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{Y}\big)+\gamma\|\boldsymbol{X}\|_\star \\ \text{s.t.} \quad & (\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{X}=\boldsymbol{0}_{n\times m},\ \boldsymbol{P}\in\mathcal{P}_{\min(k,\mathrm{rank}(\boldsymbol{X}))}. \end{aligned} \tag{6}$$

Observe that the matrix pseudo-inverse term has been eliminated from the objective function, however we have introduced the bilinear constraint $\boldsymbol{X}=\boldsymbol{P}\boldsymbol{X}$ which is non convex in the optimization variables as well as the non convex constraint $\boldsymbol{P}\in\mathcal{P}_{\min(k,\mathrm{rank}(\boldsymbol{X}))}$. We now have the following result:

Proposition 4. *Problem (6) is a valid reformulation of (5).*

Proof. We show that given a feasible solution to (6), we can construct a feasible solution to (5) that achieves the same objective value and vice versa.

Consider an arbitrary feasible solution $(\bar{\boldsymbol{X}},\bar{\boldsymbol{P}})$ to (6). Since $\bar{\boldsymbol{P}}\bar{\boldsymbol{X}}=\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{P}}\in\mathcal{P}_{\min(k,\mathrm{rank}(\bar{\boldsymbol{X}}))}$, we have $\mathrm{rank}(\bar{\boldsymbol{X}})\le k$. We claim that $\bar{\boldsymbol{X}}$ achieves the same objective value in (5) as $(\bar{\boldsymbol{X}},\bar{\boldsymbol{P}})$ achieves in (6). To show this, it suffices to illustrate that for all $(\bar{\boldsymbol{X}},\bar{\boldsymbol{P}})$ feasible to (6) we have $H(\bar{\boldsymbol{X}}):=\bar{\boldsymbol{X}}(\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}})^\dagger\bar{\boldsymbol{X}}^T=\bar{\boldsymbol{P}}$. The matrix $\bar{\boldsymbol{P}}$ is an orthogonal projection matrix since it is symmetric and satisfies $\bar{\boldsymbol{P}}^2=\bar{\boldsymbol{P}}$. Moreover, since $\mathrm{rank}(\bar{\boldsymbol{P}})=\mathrm{rank}(\bar{\boldsymbol{X}})$ and $\bar{\boldsymbol{P}}\bar{\boldsymbol{X}}=\bar{\boldsymbol{X}}$ we know that $\bar{\boldsymbol{P}}$ is an orthogonal projection onto the subspace spanned by the columns of $\bar{\boldsymbol{X}}$. Similarly, it can easily be verified that $H(\bar{\boldsymbol{X}})$ is symmetric and satisfies $H(\bar{\boldsymbol{X}})^2=H(\bar{\boldsymbol{X}})$, $\mathrm{rank}(H(\bar{\boldsymbol{X}}))=\mathrm{rank}(\bar{\boldsymbol{X}})$ and $H(\bar{\boldsymbol{X}})\bar{\boldsymbol{X}}=\bar{\boldsymbol{X}}$. Thus, $H(\bar{\boldsymbol{X}})$ is also an orthogonal projection matrix onto the subspace spanned by the columns of $\bar{\boldsymbol{X}}$. To conclude, we invoke the property that given a subspace $\mathcal{V}\subset\mathbb{R}^n$ the orthogonal projection onto $\mathcal{V}$ is uniquely defined. To see this, suppose $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are two orthogonal projections onto $\mathcal{V}$. Let $l=\dim(\mathcal{V})$. Let $\{\boldsymbol{e}_i\}_{i=1}^l$ be an orthogonal basis for $\mathcal{V}$ and let $\{\boldsymbol{e}_i\}_{l+1}^n$ be an orthogonal basis for $\mathcal{V}^\perp$. Since $\boldsymbol{P}_1$ is an orthogonal projection onto $\mathcal{V}$, we have $\boldsymbol{P}_1\boldsymbol{e}_i=\boldsymbol{e}_i$ for all $1\le i\le l$ and $\boldsymbol{P}_1\boldsymbol{e}_i=\boldsymbol{0}_n$ for all $l+1\le i\le n$. However, the same must hold for $\boldsymbol{P}_2$ which implies that $\boldsymbol{P}_1=\boldsymbol{P}_2$.

Consider an arbitrary feasible solution $\bar{\boldsymbol{X}}$ to (5). Let $r=\mathrm{rank}(\bar{\boldsymbol{X}})$ and $\bar{\boldsymbol{X}}=\bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{V}}^T$ be a singular value decomposition of $\bar{\boldsymbol{X}}$ where we have $\bar{\boldsymbol{U}}\in\mathbb{R}^{n\times r},\bar{\boldsymbol{\Sigma}}\in\mathbb{R}^{r\times r},\bar{\boldsymbol{V}}\in\mathbb{R}^{m\times r}$. Define $\bar{\boldsymbol{P}}=\bar{\boldsymbol{U}}\bar{\boldsymbol{U}}^T$. By construction, we have $\bar{\boldsymbol{P}}\in\mathcal{P}_{\min(k,\mathrm{rank}(\bar{\boldsymbol{X}}))}$ since $r\le k$. Moreover, it is easy to verify that

$$\bar{\boldsymbol{P}}\bar{\boldsymbol{X}}=\bar{\boldsymbol{U}}\bar{\boldsymbol{U}}^T\bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{V}}^T=\bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{V}}^T=\bar{\boldsymbol{X}},$$

where we have used the property $\bar{\boldsymbol{U}}^T\bar{\boldsymbol{U}}=\boldsymbol{I}_r$. Finally, Proposition 3 immediately implies that $(\bar{\boldsymbol{X}},\bar{\boldsymbol{P}})$ achieves the same objective in (6) as $\bar{\boldsymbol{X}}$ achieves in (5). This completes the proof. □

Optimizing explicitly over the space of $n\times m$ matrices can rapidly become prohibitively costly in terms of runtime and memory requirements. Accordingly, we adopt the common approach of factorizing $\boldsymbol{X}\in\mathbb{R}^{n\times m}$ as $\boldsymbol{U}\boldsymbol{V}^T$ for $\boldsymbol{U}\in\mathbb{R}^{n\times k},\boldsymbol{V}\in\mathbb{R}^{m\times k}$. This leads to the following formulation:

$$\begin{aligned} \min_{\substack{\boldsymbol{U}\in\mathbb{R}^{n\times k}\\ \boldsymbol{V}\in\mathbb{R}^{m\times k},\\ \boldsymbol{P}\in\mathbb{R}^{n\times n}}} \quad & \sum_{(i,j)\in\Omega} ((\boldsymbol{U}\boldsymbol{V}^T)_{ij} - A_{ij})^2 + \lambda \mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Y}\big) + \frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) \\ \text{s.t.} \quad & (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{U} = \boldsymbol{0}_{n\times k},\ \boldsymbol{P} \in \mathcal{P}_{\min(k,\mathrm{rank}(\boldsymbol{U}\boldsymbol{V}^T))}. \end{aligned} \tag{7}$$

Notice that we have replaced $n\times m$ optimization variables with $k\times(n+m)$ optimization variables, an often significant dimension reduction in practice. Attentive readers may object that though this is true, we have introduced n^2 decision variables through the introduction of the projection matrix variable $\boldsymbol{P}$ which nullifies any savings introduced through the factorization of $\boldsymbol{X}$. Note, however, that it is possible to factor any feasible projection matrix as $\boldsymbol{P} = \boldsymbol{M}\boldsymbol{M}^T$ for some $\boldsymbol{M} \in \mathbb{R}^{n\times k}$. In Section 5, we leverage this fact so that the presence of the projection matrix incurs a cost of $n \times k$ additional variables rather than n^2 variables. We have the following result:

Proposition 5. *Problem (7) is a valid reformulation of (6).*

Proof. We show that given a feasible solution to (7), we can construct a feasible solution to (6) that achieves the same or lesser objective value and vice versa.

Consider an arbitrary feasible solution $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ to (7). Let $\bar{\boldsymbol{X}} = \bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^T$. We will show that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ is feasible to (6) and achieves the same or lesser objective as $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ does in (7). Feasibility of (7) implies that $\bar{\boldsymbol{P}} \in \mathcal{P}_{\min(k,\mathrm{rank}(\bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^T))} = \mathcal{P}_{\min(k,\mathrm{rank}(\bar{\boldsymbol{X}}))}$ and also that

$$(\boldsymbol{I}_n - \bar{\boldsymbol{P}})\bar{\boldsymbol{X}} = (\boldsymbol{I}_n - \bar{\boldsymbol{P}})\bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^T = \boldsymbol{0}_{n\times k}\bar{\boldsymbol{V}}^T = \boldsymbol{0}_{n\times m},$$

thus the solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ is certainly feasible for (6). To see that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ achieves the same or lesser objective value, it suffices to argue that $\|\bar{\boldsymbol{X}}\|_\star \le \frac{1}{2}(\|\bar{\boldsymbol{U}}\|_F^2 + \|\bar{\boldsymbol{V}}\|_F^2)$. This follows immediately from the following well-known proposition established by [23] (see Appendix A.5 in their paper for a proof):

Proposition 6. *For any matrix $\boldsymbol{Z}$, the following holds:*

$$\|\boldsymbol{Z}\|_\star = \min_{\boldsymbol{U},\boldsymbol{V}:\boldsymbol{Z}=\boldsymbol{U}\boldsymbol{V}^T} \frac{1}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2).$$

If $\mathrm{rank}(\boldsymbol{Z}) = k \le \min(m,n)$, then the minimum above is attained at a factor decomposition $\boldsymbol{U}_{n\times k}\boldsymbol{V}_{m\times k}^T$. Letting $\boldsymbol{Z}_{n\times m} = \boldsymbol{L}_{n\times k}\boldsymbol{\Sigma}_{k\times k}\boldsymbol{R}_{m\times k}^T$ denote a singular value decomposition of $\boldsymbol{Z}$, the minimum above is attained at $\boldsymbol{U}_{n\times k} = \boldsymbol{L}_{n\times k}\boldsymbol{\Sigma}_{k\times k}^{\frac{1}{2}}, \boldsymbol{V}_{m\times k} = \boldsymbol{R}_{m\times k}\boldsymbol{\Sigma}_{k\times k}^{\frac{1}{2}}$.

Consider now an arbitrary feasible solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ to (6). Let $\bar{\boldsymbol{X}} = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}^T$ be a singular value decomposition of $\bar{\boldsymbol{X}}$ where $\boldsymbol{L} \in \mathbb{R}^{n\times k}, \boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}, \boldsymbol{R} \in \mathbb{R}^{m\times k}$ and define $\bar{\boldsymbol{U}} = \boldsymbol{L}\boldsymbol{\Sigma}^{\frac{1}{2}}, \bar{\boldsymbol{V}} = \boldsymbol{R}\boldsymbol{\Sigma}^{\frac{1}{2}}$. Feasibility of $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ in (6) implies that $\bar{\boldsymbol{P}} \in \mathcal{P}_{\min(k,\mathrm{rank}(\bar{\boldsymbol{X}}))} =$

$\mathcal{P}_{\min(k,\text{rank}(\bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^T))}$. Moreover, since the columns of $\boldsymbol{L}$ form an orthogonal basis for the columns space of $\bar{\boldsymbol{X}}$, the condition $(\boldsymbol{I}_n - \bar{\boldsymbol{P}})\bar{\boldsymbol{X}} = \boldsymbol{0}_{n\times m}$ implies that

$$(\boldsymbol{I}_n - \bar{\boldsymbol{P}})\bar{\boldsymbol{U}} = (\boldsymbol{I}_n - \bar{\boldsymbol{P}})\boldsymbol{L}\boldsymbol{\Sigma}^{\frac{1}{2}} = \boldsymbol{0}_{n\times k}\boldsymbol{\Sigma}^{\frac{1}{2}} = \boldsymbol{0}_{n\times k}.$$

Thus, the solution $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ is feasible to (7). Moreover, by Proposition 6 we have $\frac{1}{2}(\|\bar{\boldsymbol{U}}\|_F^2 + \|\bar{\boldsymbol{V}}\|_F^2) = \|\bar{\boldsymbol{X}}\|_\star$ so $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ achieves the same objective in (7) as $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ does in (6). This completes the proof.

□

In the remainder of the paper, we will relax the constraint $\boldsymbol{P} \in \mathcal{P}_{\min(k,\text{rank}(\boldsymbol{U}\boldsymbol{V}^T))}$ to $\boldsymbol{P} \in \mathcal{P}_k$ and develop a scalable algorithm to obtain high quality feasible solutions. Explicitly, we consider the problem given by:

$$\begin{aligned} \min_{\substack{\boldsymbol{U}\in\mathbb{R}^{n\times k},\\ \boldsymbol{V}\in\mathbb{R}^{m\times k},\\ \boldsymbol{P}\in\mathbb{R}^{n\times n}}} \quad & \sum_{(i,j)\in\Omega} ((\boldsymbol{U}\boldsymbol{V}^T)_{ij} - A_{ij})^2 + \lambda \text{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Y}\big) + \frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) \\ \text{s.t.} \quad & (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{U} = \boldsymbol{0}_{n\times k},\ \boldsymbol{P} \in \mathcal{P}_k. \end{aligned} \tag{8}$$

It is straightforward to see that the optimal value of (8) is no greater than the optimal value of (7). Unfortunately, the converse does not necessarily hold. To see why the optimal value of (8) can be strictly less than that of (7) in certain pathological cases, suppose we had $k = n = m$, $\Omega = \emptyset$. In this setting, letting $\bar{\boldsymbol{P}} = \boldsymbol{I}_n$, $\boldsymbol{U} = \boldsymbol{0}_{n\times k}$ and $\bar{\boldsymbol{V}} = \boldsymbol{0}_{m\times k}$, the solution $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ would be feasible to (8) and achieve an objective value of 0. However the optimal value of (7) would be strictly greater than 0 in this setting as long as $\boldsymbol{Y} \neq 0$. Although (8) is a relaxation of (1), we will see in Section 6 that the solutions we will obtain to (8) will be high quality solutions for (1), the main problem of interest.

4.1 A Positive Semidefinite Cone Relaxation

Convex relaxations are useful in non convex optimization primarily for two reasons. Firstly, given the objective value achieved by an arbitrary feasible solution, strong convex relaxations can be used to upperbound the worst case suboptimality of said solution. Secondly, convex relaxations can often be used as building blocks for global optimization procedures. In this section, we present a natural convex relaxation of (8) that leverages the matrix generalization of the perspective relaxation [17, 38, 40, 50].

Rather than working directly with (8), consider the equivalent formulation (6) with $\mathcal{P}_{\min(k,\text{rank}(\boldsymbol{X}))}$ replaced by $\mathcal{P}_k$. Before proceeding, we will assume knowledge of an upper bound $M \in \mathbb{R}_+$ on the spectral norm of an optimal $\boldsymbol{X}$ to (6). Tighter bounds M are desirable as they will lead to stronger convex relaxations of (6). We note that it is always possible to specify such an upper bound M without prior knowledge of an optimal solution to (6). To see this, note that setting $\boldsymbol{X} = \boldsymbol{0}_{n\times m}$ in (6) produces an objective value of $\sum_{(i,j)\in\Omega} A_{ij}^2 + \lambda\|\boldsymbol{Y}\|_F^2$. Thus, any $\boldsymbol{X}$ such that

$\gamma\|\boldsymbol{X}\|_\star > \sum_{(i,j)\in\Omega} A_{ij}^2 + \lambda\|\boldsymbol{Y}\|_F^2$ cannot possibly be optimal to (6). Finally, since the nuclear norm is an upper bound on the spectral norm of a matrix, we must have

$$\|\boldsymbol{X}\|_\sigma \leq \frac{\sum_{(i,j)\in\Omega} A_{ij}^2 + \lambda\|\boldsymbol{Y}\|_F^2}{\gamma},$$

for any matrix $\boldsymbol{X}$ that is optimal to (6). We can therefore take $M = \frac{\sum_{(i,j)\in\Omega} A_{ij}^2 + \lambda\|\boldsymbol{Y}\|_F^2}{\gamma}$.

Notice that the non convexity in (6) is captured entirely by the bilinear constraint $(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{X} = \boldsymbol{0}_{n\times m}$ and the quadratic constraint $\boldsymbol{P}^2 = \boldsymbol{P}$. In keeping with the approach presented in [38, 40], we leverage the matrix perspective to convexify the bilinear term and solve over the convex hull of the set $\mathcal{P}_k$. Recalling that the nuclear norm is semidefinite representable, we have the following formulation:

$$\begin{aligned}
\min_{\substack{\boldsymbol{P},\boldsymbol{W}_1\in\mathbb{R}^{n\times n},\\ \boldsymbol{X}\in\mathbb{R}^{n\times m}\\ \boldsymbol{W}_2\in\mathbb{R}^{m\times m}}} \quad & \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 + \lambda\mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Y}\big) + \frac{\gamma}{2}(\mathrm{Tr}(\boldsymbol{W}_1) + \mathrm{Tr}(\boldsymbol{W}_2)) \\
\text{s.t.} \quad & \boldsymbol{I}_n \succeq \boldsymbol{P} \succeq 0,\ \mathrm{Tr}(\boldsymbol{P}) \leq k, \\
& \begin{pmatrix} M\boldsymbol{P} & \boldsymbol{X} \\ \boldsymbol{X}^T & M\boldsymbol{I}_m \end{pmatrix} \succeq 0,\ \begin{pmatrix} \boldsymbol{W}_1 & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{W}_2 \end{pmatrix} \succeq 0.
\end{aligned} \tag{9}$$

We now have the following result:

Proposition 7. *Problem (9) is a valid convex relaxation of (8).*

Proof. Problem (9) is clearly a convex optimization problem. We will show that the optimal value of (9) is a lower bound on the optimal value of (8) by showing that given any optimal solution to (8), we can construct a feasible solution to (9) that achieves the same objective value.

Consider any optimal solution $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ to (8). From the proof of Proposition 5, we know that the solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ where $\bar{\boldsymbol{X}} = \bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^T$ is feasible to (6) (where we replace the constraint $\boldsymbol{P} \in \mathcal{P}_{\min(k,\mathrm{rank}(\boldsymbol{U}\boldsymbol{V}^T))}$ with $\boldsymbol{P} \in \mathcal{P}_k$) and must also be optimal. Let $\bar{\boldsymbol{X}} = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}^T$ be a singular value decomposition of $\bar{\boldsymbol{X}}$ with $\boldsymbol{L} \in \mathbb{R}^{n\times k}, \boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}$ and $\boldsymbol{R} \in \mathbb{R}^{m\times k}$. Let $\bar{\boldsymbol{W}}_1 = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T$ and $\bar{\boldsymbol{W}}_2 = \boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^T$. We claim that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{W}}_1, \bar{\boldsymbol{W}}_2)$ is feasible to (9) and achieves the same objective value as $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ does in (6).

From the feasibility of $\bar{\boldsymbol{P}}$ in (8), we know that $\bar{\boldsymbol{P}} \in \mathcal{P}_k$ which implies $\boldsymbol{I}_n \succeq \bar{\boldsymbol{P}} \succeq 0$ and $\mathrm{Tr}(\bar{\boldsymbol{P}}) \leq k$. By the generalized Schur complement lemma (see [51], Equation 2.41), we know that

$$\begin{pmatrix} M\bar{\boldsymbol{P}} & \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^T & M\boldsymbol{I}_m \end{pmatrix} \succeq 0 \iff M\boldsymbol{I}_m \succeq 0, \text{ and } M\boldsymbol{I}_m - \bar{\boldsymbol{X}}^T(M\bar{\boldsymbol{P}})^\dagger\bar{\boldsymbol{X}} \succeq 0.$$

We trivially have $M\boldsymbol{I}_m \succeq 0$. To see that the second condition holds, note that since $\bar{\boldsymbol{P}}$ is a projection matrix and $\bar{\boldsymbol{P}}\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}$, we have $\bar{\boldsymbol{X}}^T(M\bar{\boldsymbol{P}})^\dagger\bar{\boldsymbol{X}} = \frac{1}{M}\bar{\boldsymbol{X}}^T\bar{\boldsymbol{P}}\bar{\boldsymbol{X}} = \frac{1}{M}\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}}$. Furthermore, since $\bar{\boldsymbol{X}}$ is optimal to (6), we have $\|\bar{\boldsymbol{X}}\|_\sigma \leq M$ by assumption. Thus, we

have

$$\|\bar{\boldsymbol{X}}\|_\sigma \leq M \implies \|\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}}\|_\sigma \leq M^2 \implies M^2\boldsymbol{I}_m \succeq \bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}} \implies M\boldsymbol{I}_m \succeq \frac{1}{M}\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}}.$$

Finally, observe that

$$\begin{pmatrix} \bar{\boldsymbol{W}}_1 & \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^T & \bar{\boldsymbol{W}}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T & \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}^T \\ \boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{L}^T & \boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^T \end{pmatrix} = \begin{pmatrix} \boldsymbol{L} \\ \boldsymbol{R} \end{pmatrix} \boldsymbol{\Sigma} \begin{pmatrix} \boldsymbol{L} \\ \boldsymbol{R} \end{pmatrix}^T.$$

Since $\boldsymbol{\Sigma}$ is a diagonal matrix with non negative entries, the matrix $\begin{pmatrix} \bar{\boldsymbol{W}}_1 & \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^T & \bar{\boldsymbol{W}}_2 \end{pmatrix}$ is certainly positive semidefinite. Thus we have shown that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{W}}_1, \bar{\boldsymbol{W}}_2)$ is indeed feasible to (9). To conclude the proof, we note that

$$\begin{aligned} \frac{\gamma}{2}(\mathrm{Tr}(\bar{\boldsymbol{W}}_1) + \mathrm{Tr}(\bar{\boldsymbol{W}}_2)) &= \frac{\gamma}{2}(\mathrm{Tr}(\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T) + \mathrm{Tr}(\boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^T)) \\ &= \frac{\gamma}{2}(\mathrm{Tr}(\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{\Sigma}) + \mathrm{Tr}(\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{\Sigma})) \\ &= \frac{\gamma}{2}(\mathrm{Tr}(\boldsymbol{\Sigma}) + \mathrm{Tr}(\boldsymbol{\Sigma})) = \gamma\|\bar{\boldsymbol{X}}\|_\star, \end{aligned}$$

thus $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{W}}_1, \bar{\boldsymbol{W}}_2)$ achieves the same objective value in (9) as $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ achieves in (6). □

In general, an optimal solution to (9) will have $\boldsymbol{P} \notin \mathcal{P}_k$. We briefly note that to obtain a stronger convex relaxation, one could leverage eigenvector disjunctions [41, 52] to iteratively cut off solutions to (9) with $\boldsymbol{P} \notin \mathcal{P}_k$ and form increasingly tighter disjunctive approximations to the set $\mathcal{P}_k$.

5 Mixed-Projection ADMM

In this section, we present an alternating direction method of multipliers (ADMM) algorithm that is scalable and obtains high quality solutions for (8) and we investigate its convergence properties. Rather than forming the augmented Lagrangian directly for (8), we first modify our problem formulation by introducing a dummy variable $\boldsymbol{Z} \in \mathbb{R}^{n\times k}$ that is an identical copy of $\boldsymbol{U}$. Additionally, rather than directly enforcing the constraint $\boldsymbol{P} \in \mathcal{P}_k$, we introduce an indicator function penalty $\mathbb{I}_{\mathcal{P}_k}(\boldsymbol{P})$ into the objective function where $\mathbb{I}_{\mathcal{X}}(\boldsymbol{x}) = 0$ if $\boldsymbol{x} \in \mathcal{X}$, otherwise $\mathbb{I}_{\mathcal{X}}(\boldsymbol{x}) = \infty$. Explicitly, we consider the following problem:

$$\begin{aligned} \min_{\boldsymbol{U},\boldsymbol{Z}\in\mathbb{R}^{n\times k},\boldsymbol{V}\in\mathbb{R}^{m\times k},\boldsymbol{P}\in\mathbb{R}^{n\times n}} \quad & \sum_{(i,j)\in\Omega} ((\boldsymbol{U}\boldsymbol{V}^T)_{ij} - A_{ij})^2 + \lambda\mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Y}\big) \\ & + \frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) + \mathbb{I}_{\mathcal{P}_k}(\boldsymbol{P}) \\ \text{s.t.} \quad & (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z} = \boldsymbol{0}_{n\times k},\ \boldsymbol{U} - \boldsymbol{Z} = \boldsymbol{0}_{n\times k}. \end{aligned} \tag{10}$$

It is trivial to see that (10) is equivalent to (8). We will see in this section that working with formulation (10) leads to an ADMM algorithm with favorable decomposition properties. Introducing dual variables $\boldsymbol{\Phi}, \boldsymbol{\Psi} \in \mathbb{R}^{n\times k}$ for the constraints $(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{U} = \boldsymbol{0}_{n\times k}$ and $\boldsymbol{U} - \boldsymbol{Z} = \boldsymbol{0}_{n\times k}$ respectively, the augmented Lagrangian $\mathcal{L}^A$ for (10) is given by:

$$\begin{aligned}\mathcal{L}^A(\boldsymbol{U},\boldsymbol{V},\boldsymbol{P},\boldsymbol{Z},\boldsymbol{\Phi},\boldsymbol{\Psi}) = &\sum_{(i,j)\in\Omega}((\boldsymbol{U}\boldsymbol{V}^T)_{ij} - A_{ij})^2 + \lambda \mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Y}\big) \\ &+ \frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) + \mathbb{I}_{\mathcal{P}_k}(\boldsymbol{P}) + \mathrm{Tr}(\boldsymbol{\Phi}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}) \\ &+ \mathrm{Tr}(\boldsymbol{\Psi}^T(\boldsymbol{Z} - \boldsymbol{U})) + \frac{\rho_1}{2}\|(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}\|_F^2 + \frac{\rho_2}{2}\|\boldsymbol{Z} - \boldsymbol{U}\|_F^2,\end{aligned} \tag{11}$$

where $\rho_1, \rho_2 > 0$ are non-negative penalty parameters. In what follows, we show that performing a partial minimization of the augmented Lagrangian (11) over each of the primal variables $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}$ yields a subproblem that can be solved efficiently. We present each subproblem and investigate the complexity of computing the corresponding subproblem solutions.

5.1 Subproblem in $\boldsymbol{U}$

First, suppose we fix variables $\boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ and seek to minimize $\mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ over $\boldsymbol{U}$. Eliminating terms that do not depend on $\boldsymbol{U}$, the resulting subproblem is given by

$$\min_{\boldsymbol{U}\in\mathbb{R}^{n\times k}} \sum_{(i,j)\in\Omega}((\boldsymbol{U}\boldsymbol{V}^T)_{ij} - A_{ij})^2 + \frac{\gamma}{2}\|\boldsymbol{U}\|_F^2 - \mathrm{Tr}(\boldsymbol{\Psi}^T\boldsymbol{U}) + \frac{\rho_2}{2}\|\boldsymbol{Z} - \boldsymbol{U}\|_F^2. \tag{12}$$

We now have the following result:

Proposition 8. *The optimal solution $\bar{\boldsymbol{U}}$ for (12) is given by*

$$\bar{U}_{i,\star} = [2\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V} + (\gamma + \rho_2)\boldsymbol{I}_k]^{-1}[2\boldsymbol{V}^T\boldsymbol{W}_i A_{i,\star} + \Psi_{i,\star} + \rho_2 Z_{i,\star}], \tag{13}$$

for each $i \in \{1, \ldots, n\}$ where each $\boldsymbol{W}_i \in \mathbb{R}^{m\times m}$ is a diagonal matrix satisfying $(\boldsymbol{W}_i)_{jj} = 1$ if $(i,j) \in \Omega$, otherwise $(\boldsymbol{W}_i)_{jj} = 0$. Here, the column vectors $\bar{U}_{i,\star} \in \mathbb{R}^k, A_{i,\star} \in \mathbb{R}^m, \Psi_{i,\star} \in \mathbb{R}^k, Z_{i,\star} \in \mathbb{R}^k$ denote the i^{th} row of the matrices $\bar{\boldsymbol{U}}, \boldsymbol{A}, \boldsymbol{\Psi}, \boldsymbol{Z}$ respectively, where the unknown entries of $\boldsymbol{A}$ are taken to be 0.

Proof. Let $f(\boldsymbol{U})$ denote the objective function of (12). With $\{\boldsymbol{W}_i\}_{i=1}^n$ defined as in Proposition 8, observe that we can write $f(\boldsymbol{U})$ as

$$\begin{aligned} f(\boldsymbol{U}) &= \sum_{i=1}^{n} \left[\|\boldsymbol{W}_i(\boldsymbol{V}U_{i,\star} - A_{i,\star})\|_2^2 + \frac{\gamma}{2}\|U_{i,\star}\|_2^2 - \Psi_{i,\star}^T U_{i,\star} + \frac{\rho_2}{2}\|Z_{i,\star} - U_{i,\star}\|_2^2 \right] \\ &= \sum_{i=1}^{n} g_i(\boldsymbol{U}), \end{aligned}$$

where we define $g_i(\boldsymbol{U}) = \|\boldsymbol{W}_i(\boldsymbol{V}U_{i,\star} - A_{i,\star})\|_2^2 + \frac{\gamma}{2}\|U_{i,\star}\|_2^2 - \Psi_{i,\star}^T U_{i,\star} + \frac{\rho_2}{2}\|Z_{i,\star} - U_{i,\star}\|_2^2$. Thus, we have shown that $f(\boldsymbol{U})$ is separable over the rows of the matrix $\boldsymbol{U}$. Each function $g_i(\boldsymbol{U})$ is a (strongly) convex quadratic. Thus, we can minimize $g_i(\boldsymbol{U})$ by setting its gradient to 0. For any fixed row $i \in \{1, \ldots, n\}$, we can differentiate and obtain

$$\nabla_{\boldsymbol{U}_{i,\star}} g_i(\boldsymbol{U}) = 2\boldsymbol{V}^T\boldsymbol{W}_i(\boldsymbol{V}U_{i,\star} - A_{i\star}) + \gamma U_{i,\star} - \Psi_{i,\star} - \rho_2(Z_{i,\star} - U_{i,\star}).$$

By equating the gradient $\nabla_{\boldsymbol{U}_{i,\star}} g_i(\boldsymbol{U})$ to 0 and rearranging, we obtain that the optimal vector $\bar{U}_{i,\star}$ is given by (13). This completes the proof. □

Observe that since the matrix $\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}$ is positive semidefinite and $\gamma + \rho_2 > 0$, the matrix inverse $[2\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V} + (\gamma + \rho_2)\boldsymbol{I}_k]^{-1}$ is well defined for all $i \in \{1, \ldots, n\}$. Computing the optimal solution to (12) requires computing n different $k \times k$ matrix inverses (where in general $k \ll \min\{m, n\}$). Computing a single $k \times k$ matrix inverse requires $O(k^3)$ time and forming the matrix product $\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}$ requires $O(k^2 m)$ time for a given i. Thus, the complexity of computing the optimal solution for a single column is $O(k^3 + k^2 m)$. Notice that each column of $\bar{\boldsymbol{U}}$ can be computed independently of the other columns. We leverage this observation by developing a multi-threaded implementation of the algorithm presented in this section. Letting w denote the number of compute threads available, computing the optimal solution $\bar{\boldsymbol{U}}$ of (12) requires $O\left(\frac{k^3 n + k^2 mn}{\min\{w,n\}}\right)$ time (the term $\min\{w, n\}$ in the denominator reflects that fact that increasing the number of available compute threads beyond the number of columns of $\bar{\boldsymbol{U}}$ does not yield additional reduction in compute complexity).

5.2 Subproblem in $\boldsymbol{V}$

Now, suppose we fix variables $\boldsymbol{U}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ and seek to minimize $\mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ over $\boldsymbol{V}$. Eliminating terms that do not depend on $\boldsymbol{V}$, the resulting subproblem is given by

$$\min_{\boldsymbol{V} \in \mathbb{R}^{m \times k}} \sum_{(i,j) \in \Omega} ((\boldsymbol{U}\boldsymbol{V}^T)_{ij} - A_{ij})^2 + \frac{\gamma}{2}\|\boldsymbol{V}\|_F^2. \tag{14}$$

We now have the following result:

Proposition 9. *The optimal solution $\bar{\boldsymbol{V}}$ for (14) is given by*

$$\bar{V}_{j,\star} = [2\boldsymbol{U}^T\boldsymbol{W}_j\boldsymbol{U} + \gamma\boldsymbol{I}_k]^{-1}[2\boldsymbol{U}^T\boldsymbol{W}_j A_{\star,j}], \tag{15}$$

for each $j \in \{1, \ldots, m\}$ *where each* $\boldsymbol{W}_j \in \mathbb{R}^{n \times n}$ *is a diagonal matrix satisfying* $(\boldsymbol{W}_j)_{ii} = 1$ *if* $(i, j) \in \Omega$*, otherwise* $(\boldsymbol{W}_j)_{ii} = 0$*. Here, the column vector* $\bar{V}_{j,\star} \in \mathbb{R}^k$ *denotes the* j^{th} *row of* $\bar{\boldsymbol{V}}$ *while the column vector* $A_{\star,j} \in \mathbb{R}^n$ *denotes the* j^{th} *column of* $\boldsymbol{A}$ *where the unknown entries of* $\boldsymbol{A}$ *are taken to be* 0*.*

Proof. This proof follows the proof of Proposition 8. Let $f(\boldsymbol{V})$ denote the objective function of (14). With $\{\boldsymbol{W}_j\}_{j=1}^m$ defined as in Proposition 9, observe that we can write $f(\boldsymbol{V})$ as

$$f(\boldsymbol{V}) = \sum_{j=1}^{m} \left[\|\boldsymbol{W}_j(\boldsymbol{U}V_{j,\star} - A_{\star,j})\|_2^2 + \frac{\gamma}{2}\|V_{j,\star}\|_2^2 \right] = \sum_{j=1}^{m} g_j(\boldsymbol{V}),$$

where we define $g_j(\boldsymbol{V}) = \|\boldsymbol{W}_j(\boldsymbol{U}V_{j,\star} - A_{\star,j})\|_2^2 + \frac{\gamma}{2}\|V_{j,\star}\|_2^2$. Thus, we have shown that $f(\boldsymbol{V})$ is separable over the rows of the matrix $\boldsymbol{V}$. Each function $g_j(\boldsymbol{V})$ is a (strongly) convex quadratic. Thus, we can minimize $g_j(\boldsymbol{V})$ by setting its gradient to 0. For any fixed row $j \in \{1, \ldots, m\}$, we can differentiate and obtain

$$\nabla_{\boldsymbol{V}_{j,\star}} g_j(\boldsymbol{V}) = 2\boldsymbol{U}^T\boldsymbol{W}_j(\boldsymbol{U}V_{j,\star} - A_{\star,j}) + \gamma V_{j,\star}.$$

By equating the gradient $\nabla_{\boldsymbol{V}_{j,\star}} g_j(\boldsymbol{V})$ to 0 and rearranging, we obtain that the optimal vector $\bar{U}_{i,\star}$ is given by (15). This completes the proof. □

Observe that since the matrix $\boldsymbol{U}^T\boldsymbol{W}_j\boldsymbol{U}$ is positive semidefinite and $\gamma > 0$, the matrix inverse $[2\boldsymbol{U}^T\boldsymbol{W}_j\boldsymbol{U} + \gamma\boldsymbol{I}_k]^{-1}$ is well defined for all $j \in \{1, \ldots, m\}$. Computing the optimal solution to (14) requires computing m different $k \times k$ matrix inverses. Forming the matrix product $\boldsymbol{U}^T\boldsymbol{W}_j\boldsymbol{U}$ requires $O(k^2 n)$ time for a given j. Thus, the complexity of computing the optimal solution for a single column is $O(k^3 + k^2 n)$. Notice that, similarly to the solution of (12), each column of $\bar{\boldsymbol{V}}$ can be computed independently of the other columns. As before, we leverage this observation in our multi-threaded implementation of the algorithm presented in this section. Letting w denote the number of compute threads available, computing the optimal solution $\bar{\boldsymbol{V}}$ of (14) requires $O\left(\frac{k^3 m + k^2 mn}{\min\{w,m\}}\right)$ time.

The optimal solution $\bar{\boldsymbol{V}}$ to (14) reveals that the Frobenius norm regularization term on $\boldsymbol{V}$ in (8) (which emerges from the nuclear norm regularization term on $\boldsymbol{X}$ in (1)) has computational benefits. Indeed, if we had $\gamma = 0$, it is possible that the matrix $\boldsymbol{U}^T\boldsymbol{W}_j\boldsymbol{U}$ be singular at certain iterates of our ADMM algorithm, in which case the corresponding matrix inverse would be undefined. This observation is in keeping with several recent works in the statistics, machine learning and operations research literatures where the presence of a regularization penalty in the objective function yields improved out of sample performance as well as benefits in computational tractability (see for example [38, 40, 53–55]).

5.3 Subproblem in $\boldsymbol{P}$

Now, suppose we fix variables $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ and seek to minimize $\mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ over $\boldsymbol{P}$. Eliminating terms that do not depend on $\boldsymbol{P}$, the resulting subproblem is given by

$$\min_{\boldsymbol{P} \in \mathbb{S}^n_+} -\lambda \mathrm{Tr}(\boldsymbol{Y}^T \boldsymbol{P} \boldsymbol{Y}) - \mathrm{Tr}(\boldsymbol{\Phi}^T \boldsymbol{P} \boldsymbol{Z}) + \frac{\rho_1}{2} \|(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{P}^2 = \boldsymbol{P},\ \mathrm{Tr}(\boldsymbol{P}) \leq k. \tag{16}$$

We now have the following result:

Proposition 10. *Let $\boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T$ be a rank k truncated singular value decomposition for the matrix given by:*

$$\Big(\lambda \boldsymbol{Y}\boldsymbol{Y}^T + \frac{\rho_1}{2}\boldsymbol{Z}\boldsymbol{Z}^T + \frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{Z}^T + \boldsymbol{Z}\boldsymbol{\Phi}^T)\Big),$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{k \times k}, \boldsymbol{M} \in \mathbb{R}^{n \times k}, \boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{I}_k$. The optimal solution $\bar{\boldsymbol{P}}$ for (16) is given by $\bar{\boldsymbol{P}} = \boldsymbol{M}\boldsymbol{M}^T$.

Proof. Let $f(\boldsymbol{P})$ denote the objective function of (16). Observe that for any $\boldsymbol{P}$ that is feasible to (16), we can write $f(\boldsymbol{P})$ as:

$$\begin{aligned} f(\boldsymbol{P}) &= -\lambda \mathrm{Tr}(\boldsymbol{Y}^T \boldsymbol{P} \boldsymbol{Y}) - \mathrm{Tr}(\boldsymbol{\Phi}^T \boldsymbol{P} \boldsymbol{Z}) + \frac{\rho_1}{2} \|(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}\|_F^2 \\ &= -\lambda \mathrm{Tr}(\boldsymbol{Y}\boldsymbol{Y}^T \boldsymbol{P}) - \mathrm{Tr}(\boldsymbol{Z}\boldsymbol{\Phi}^T \boldsymbol{P}) + \frac{\rho_1}{2} \mathrm{Tr}(\boldsymbol{Z}\boldsymbol{Z}^T(\boldsymbol{I}_n - \boldsymbol{P})) \\ &= \frac{\rho_1}{2}\mathrm{Tr}(\boldsymbol{Z}\boldsymbol{Z}^T) - \langle \lambda \boldsymbol{Y}\boldsymbol{Y}^T + \boldsymbol{Z}\boldsymbol{\Phi}^T + \frac{\rho_1}{2}\boldsymbol{Z}\boldsymbol{Z}^T, \boldsymbol{P} \rangle. \end{aligned}$$

Thus, it is immediately clear that a solution will be optimal to (16) if and only if it is optimal to the problem given by:

$$\max_{\boldsymbol{P} \in \mathbb{S}^n_+} \langle \boldsymbol{C}, \boldsymbol{P} \rangle \quad \text{s.t.} \quad \boldsymbol{P}^2 = \boldsymbol{P},\ \mathrm{Tr}(\boldsymbol{P}) \leq k, \tag{17}$$

where we define the matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ as $\boldsymbol{C} = \lambda \boldsymbol{Y}\boldsymbol{Y}^T + \boldsymbol{Z}\boldsymbol{\Phi}^T + \frac{\rho_1}{2}\boldsymbol{Z}\boldsymbol{Z}^T$. Let $\bar{\boldsymbol{C}} = \frac{1}{2}(\boldsymbol{C} + \boldsymbol{C}^T)$ denote the symmetric part of $\boldsymbol{C}$. Observe that for any symmetric matrix $\boldsymbol{P}$, we can consider $\langle \bar{\boldsymbol{C}}, \boldsymbol{P} \rangle$ in place of $\langle \boldsymbol{C}, \boldsymbol{P} \rangle$ since we have

$$\begin{aligned} \langle \boldsymbol{C}, \boldsymbol{P} \rangle &= \sum_{i=1}^n \sum_{j=1}^n P_{ij} C_{ij} = \sum_{i=1}^n P_{ii} C_{ii} + \sum_{i=1}^{n-1} \sum_{j=i+1}^n P_{ij}(C_{ij} + C_{ji}) \\ &= \sum_{i=1}^n P_{ii} \bar{C}_{ii} + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n P_{ij} \bar{C}_{ij} = \sum_{i=1}^n \sum_{j=1}^n P_{ij} \bar{C}_{ij} = \langle \bar{\boldsymbol{C}}, \boldsymbol{P} \rangle. \end{aligned}$$

Let $\bar{\boldsymbol{C}} = \boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T$ be a full singular value decomposition of $\bar{\boldsymbol{C}}$ with $\boldsymbol{M}, \boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}, \boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{M}\boldsymbol{M}^T = \boldsymbol{I}_n$. The matrix $\boldsymbol{\Sigma}$ is the diagonal matrix of (ordered) singular values of $\bar{\boldsymbol{C}}$ and we let σ_i denote the i^{th} singular value. Any feasible matrix $\boldsymbol{P}$ to (17) can be written as $\boldsymbol{P} = \boldsymbol{L}\boldsymbol{L}^T$ where $\boldsymbol{L} \in \mathbb{R}^{n\times k}, \boldsymbol{L}^T\boldsymbol{L} = \boldsymbol{I}_k$. Thus, for any $\boldsymbol{P}$ feasible to (17) we express the objective value as:

$$\langle \bar{\boldsymbol{C}}, \boldsymbol{P}\rangle = \mathrm{Tr}(\boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T\boldsymbol{L}\boldsymbol{L}^T) = \mathrm{Tr}(\boldsymbol{\Sigma}(\boldsymbol{M}^T\boldsymbol{L}\boldsymbol{L}^T\boldsymbol{M})) = \sum_{i=1}^{n}\sigma_i\|(\boldsymbol{M}^T\boldsymbol{L})_{i,\star}\|_2^2.$$

Let $\boldsymbol{N} = \boldsymbol{M}^T\boldsymbol{L} \in \mathbb{R}^{n\times k}$. Note that we have $\boldsymbol{N}^T\boldsymbol{N} = \boldsymbol{L}^T\boldsymbol{M}\boldsymbol{M}^T\boldsymbol{L} = \boldsymbol{L}^T\boldsymbol{L} = \boldsymbol{I}_k$, which implies that the columns of $\boldsymbol{N}$ are orthonormal. This immediately implies that we have $N_{i,\star}^T N_{i,\star} = \|(\boldsymbol{M}^T\boldsymbol{L})_{i,\star}\|_2^2 \leq 1$. Moreover, we have

$$\sum_{i=1}^{n}\|(\boldsymbol{M}^T\boldsymbol{L})_{i,\star}\|_2^2 = \sum_{i=1}^{n}N_{i,\star}^T N_{i,\star} = \sum_{i=1}^{n}\sum_{j=1}^{k}N_{ij}^2 = \sum_{j=1}^{k}\sum_{i=1}^{n}N_{ij}^2 = \sum_{j=1}^{k}1 = k.$$

We can therefore upper bound the optimal objective value of (17) as

$$\langle \bar{\boldsymbol{C}}, \boldsymbol{P}\rangle = \sum_{i=1}^{n}\sigma_i\|(\boldsymbol{M}^T\boldsymbol{L})_{i,\star}\|_2^2 \leq \sum_{i=1}^{k}\sigma_i.$$

To conclude the proof, notice that by taking $\bar{\boldsymbol{P}} = \bar{\boldsymbol{M}}\bar{\boldsymbol{M}}^T$ where $\bar{\boldsymbol{M}} \in \mathbb{R}^{n\times k}$ is the matrix that consists of the first k columns of $\boldsymbol{M}$ we can achieve the upper bound on (17):

$$\langle \bar{\boldsymbol{C}}, \bar{\boldsymbol{P}}\rangle = \mathrm{Tr}(\boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T\bar{\boldsymbol{M}}\bar{\boldsymbol{M}}^T) = \mathrm{Tr}(\bar{\boldsymbol{M}}^T\boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T\bar{\boldsymbol{M}}) = \sum_{i=1}^{k}\sigma_i.$$

□

To compute the optimal solution of (16), we need to compute a rank k singular value decomposition of the matrix $\bar{\boldsymbol{C}} = \left(\lambda\boldsymbol{Y}\boldsymbol{Y}^T + \frac{\rho_1}{2}\boldsymbol{Z}\boldsymbol{Z}^T + \frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{Z}^T + \boldsymbol{Z}\boldsymbol{\Phi}^T)\right)$ which requires $O(kn^2)$ time since $\bar{\boldsymbol{C}} \in \mathbb{R}^{n\times n}$. Moreover, explicitly forming the matrix $\bar{\boldsymbol{C}}$ in memory from its constituent matrices $\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{\Phi}$ requires $O(n^2(d+k))$ operations. Thus, naively computing the optimal solution to (16) has complexity $O(n^2(d+k))$ where the bottleneck operation from a complexity standpoint is explicitly forming the matrix $\bar{\boldsymbol{C}}$.

Fortunately, it is possible to compute the optimal solution to (16) more efficiently. Observe that we can equivalently express the matrix $\bar{\boldsymbol{C}}$ as $\bar{\boldsymbol{C}} = \boldsymbol{F}_1\boldsymbol{F}_2^T$ where $\boldsymbol{F}_1, \boldsymbol{F}_2 \in \mathbb{R}^{n\times(d+3k)}$ are defined as

$$\boldsymbol{F}_1 = \left(\sqrt{\lambda}\boldsymbol{Y} \;\middle|\; \sqrt{\tfrac{\rho_1}{2}}\boldsymbol{Z} \;\middle|\; \sqrt{\tfrac{1}{2}}\boldsymbol{\Phi} \;\middle|\; \sqrt{\tfrac{1}{2}}\boldsymbol{Z}\right),$$
$$\boldsymbol{F}_2 = \left(\sqrt{\lambda}\boldsymbol{Y} \;\middle|\; \sqrt{\tfrac{\rho_1}{2}}\boldsymbol{Z} \;\middle|\; \sqrt{\tfrac{1}{2}}\boldsymbol{Z} \;\middle|\; \sqrt{\tfrac{1}{2}}\boldsymbol{\Phi}\right).$$

Computing a truncated singular value decomposition requires only computing repeated matrix vector products. Therefore, rather than explicitly forming the matrix

$\bar{\boldsymbol{C}}$ in memory at a cost of $O(n^2(d+k))$ operations, in our implementation we design a custom matrix class where matrix vector products between $\bar{\boldsymbol{C}}$ and arbitrary vectors $\boldsymbol{x} \in \mathbb{R}^n$ are computed by first evaluating the matrix vector product $\boldsymbol{\nu} = \boldsymbol{F}_2^T \boldsymbol{x}$ and subsequently evaluating the matrix vector product $\bar{\boldsymbol{C}}\boldsymbol{x} = \boldsymbol{F}_1 \boldsymbol{\nu}$. In so doing, we can evaluate matrix vector products $\bar{\boldsymbol{C}}\boldsymbol{x}$ in $O(n(d+k))$ time rather than $O(n^2)$ time (in general, we will have $d + k \ll n$). Computing a truncated singular value decomposition of $\bar{\boldsymbol{C}}$ with this methodology of evaluating matrix vector products requires only $O(k^2 n + knd)$ operations. Thus, our custom matrix class implementation avoids needing to explicitly for $\bar{\boldsymbol{C}}$ in memory and allows the optimal solution to (16) to be computed in $O(k^2 n + knd)$ time.

5.4 Subproblem in $\boldsymbol{Z}$

Now, suppose we fix variables $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ and seek to minimize $\mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ over $\boldsymbol{Z}$. Eliminating terms that do not depend on $\boldsymbol{Z}$, the resulting subproblem is given by

$$\min_{\boldsymbol{Z} \in \mathbb{R}^{n \times k}} \operatorname{Tr}(\boldsymbol{\Phi}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}) + \operatorname{Tr}(\boldsymbol{\Psi}^T \boldsymbol{Z}) + \frac{\rho_1}{2}\|(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}\|_F^2 + \frac{\rho_2}{2}\|\boldsymbol{Z} - \boldsymbol{U}\|_F^2. \tag{18}$$

We now have the following result:

Proposition 11. *The optimal solution $\bar{\boldsymbol{Z}}$ for (18) is given by*

$$\begin{aligned} \bar{\boldsymbol{Z}} &= \frac{1}{\rho_1 + \rho_2}\Big(\boldsymbol{I}_n + \frac{\rho_1}{\rho_2}\boldsymbol{P}\Big)\Big(\rho_2 \boldsymbol{U} - (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{\Phi} - \boldsymbol{\Psi}\Big) \\ &= \frac{1}{\rho_1 + \rho_2}\Big(\rho_2 \boldsymbol{U} - \boldsymbol{\Phi} + \boldsymbol{P}\boldsymbol{\Phi} - \boldsymbol{\Psi} + \rho_1 \boldsymbol{P}\boldsymbol{U} - \frac{\rho_1}{\rho_2}\boldsymbol{P}\boldsymbol{\Psi}\Big). \end{aligned} \tag{19}$$

Proof. Let $f(\boldsymbol{Z})$ denote the objective function of (18). The function $f(\boldsymbol{Z})$ is a convex quadratic, thus it can be minimized by setting its gradient to 0. Differentiating $f(\boldsymbol{Z})$, we obtain:

$$\nabla_{\boldsymbol{Z}} f(\boldsymbol{Z}) = (\boldsymbol{I}_n - \boldsymbol{P})^T \boldsymbol{\Phi} + \boldsymbol{\Psi} + \rho_1 (\boldsymbol{I}_n - \boldsymbol{P})^T (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z} + \rho_2 (\boldsymbol{Z} - \boldsymbol{U}).$$

Moreover, for any matrix $\boldsymbol{P}$ for which the augmented Lagrangian (11) takes finite value, we will have $\boldsymbol{P} \in \mathcal{P}_k$ which implies that $\boldsymbol{P}^T = \boldsymbol{P}$ and $\boldsymbol{P}^2 = \boldsymbol{P}$. We can therefore simplify $\nabla_{\boldsymbol{Z}} f(\boldsymbol{Z})$ as:

$$\nabla_{\boldsymbol{Z}} f(\boldsymbol{Z}) = (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{\Phi} + \boldsymbol{\Psi} + \rho_1 (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z} + \rho_2 (\boldsymbol{Z} - \boldsymbol{U}).$$

By equating the gradient $\nabla_{\boldsymbol{Z}} f(\boldsymbol{Z})$ to 0 and rearranging, we obtain that the optimal matrix $\bar{\boldsymbol{Z}}$ is given by:

$$\bar{\boldsymbol{Z}} = \Big(\rho_1(\boldsymbol{I}_n - \boldsymbol{P}) + \rho_2 \boldsymbol{I}_n\Big)^{-1}\Big(\rho_2 \boldsymbol{U} - (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{\Phi} - \boldsymbol{\Psi}\Big)$$

To conclude the proof, it remains to show that $\big(\rho_1(\boldsymbol{I}_n - \boldsymbol{P}) + \rho_2 \boldsymbol{I}_n\big)^{-1} = \frac{1}{\rho_1+\rho_2}\big(\boldsymbol{I}_n + \frac{\rho_1}{\rho_2}\boldsymbol{P}\big)$. Let $\boldsymbol{P} = \boldsymbol{M}\boldsymbol{M}^T$ where $\boldsymbol{M} \in \mathbb{R}^{n\times k}, \boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{I}_k$. Such a matrix $\boldsymbol{M}$ is guaranteed to exist for any $\boldsymbol{P} \in \mathcal{P}_k$. We have

$$\begin{aligned}
\rho_1(\boldsymbol{I}_n - \boldsymbol{P}) + \rho_2\boldsymbol{I}_n &= \rho_1(\boldsymbol{I}_n - \boldsymbol{M}\boldsymbol{M}^T) + \rho_2\boldsymbol{I}_n \\
&= (\rho_1+\rho_2)\boldsymbol{I}_n + \boldsymbol{M}(-\rho_1\boldsymbol{I}_n)\boldsymbol{M}^T \\
&= \frac{1}{\rho_1+\rho_2}\boldsymbol{I}_n - \frac{1}{(\rho_1+\rho_2)^2}\boldsymbol{M}\bigg(\frac{1}{\rho_1+\rho_2}\boldsymbol{M}^T\boldsymbol{M} - \frac{1}{\rho_1}\boldsymbol{I}_k\bigg)^{-1}\boldsymbol{M}^T \\
&= \frac{1}{\rho_1+\rho_2}\boldsymbol{I}_n - \frac{1}{(\rho_1+\rho_2)^2}\boldsymbol{M}\bigg(\frac{-\rho_1(\rho_1+\rho_2)}{\rho_2}\boldsymbol{I}_k\bigg)\boldsymbol{M}^T \\
&= \frac{1}{\rho_1+\rho_2}\Big(\boldsymbol{I}_n + \frac{\rho_1}{\rho_2}\boldsymbol{P}\Big),
\end{aligned}$$

where the third equality follows from the Woodbury matrix inversion lemma (see [56], Section 3.2.2). As a sanity check, one can verify that the product of $\big(\rho_1(\boldsymbol{I}_n-\boldsymbol{P})+\rho_2\boldsymbol{I}_n\big)$ and $\frac{1}{\rho_1+\rho_2}\big(\boldsymbol{I}_n + \frac{\rho_1}{\rho_2}\boldsymbol{P}\big)$ is indeed the n dimensional identity matrix. □

Evaluating the optimal solution to (18) requires only matrix-matrix multiplications. Computing the products of $\boldsymbol{P}\boldsymbol{\Phi}, \boldsymbol{P}\boldsymbol{U}, \boldsymbol{P}\boldsymbol{\Psi}$ in the definition of $\bar{\boldsymbol{Z}}$ from (19) requires $O(kn^2)$ operations. Thus, the naive cost of forming $\bar{\boldsymbol{Z}}$ is $O(kn^2)$. However, notice that if we had a factored representation of the matrix $\boldsymbol{P}$ as $\boldsymbol{P} = \boldsymbol{M}\boldsymbol{M}^T$ with $\boldsymbol{M} \in \mathbb{R}^{n\times k}$, for any matrix $\boldsymbol{R} \in \mathbb{R}^{n\times k}$ we could compute matrix-matrix products $\boldsymbol{P}\boldsymbol{R}$ by first computing $\boldsymbol{S} = \boldsymbol{M}^T\boldsymbol{R}$ and thereafter computing $\boldsymbol{P}\boldsymbol{R} = \boldsymbol{M}\boldsymbol{S}$ for a total complexity of $O(k^2n)$. One might object that this ignores the time required to compute such a matrix $\boldsymbol{M}$. However, observe that in computing a matrix $\boldsymbol{P}$ that is optimal to (16), we in fact must already generate such a matrix $\boldsymbol{M}$ (see proposition 10). In fact, in our implementation we never explicitly form a $n \times n$ matrix $\boldsymbol{P}$ as it suffices to only store a copy of its low rank factorization matrix $\boldsymbol{M}$. Thus, the optimal solution to (18) can be evaluated in $O(k^2n)$ time.

5.5 An ADMM Algorithm

Having illustrated that the partial minimization of the Lagrangian (11) across each of the primal variables (Problems (12), (14), (16), (18)) can be solved efficiently, we can now present the overall approach Algorithm 1.

We initialize primal iterates $\boldsymbol{U}_0 = \boldsymbol{Z}_0 = \boldsymbol{L}\boldsymbol{\Sigma}^{\frac{1}{2}}, \boldsymbol{P}_0 = \boldsymbol{L}\boldsymbol{L}^T, \boldsymbol{V}_0 = \boldsymbol{R}\boldsymbol{\Sigma}^{\frac{1}{2}}$ where $\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}$ denotes a rank k truncated singular value decomposition of $\boldsymbol{A}$ (the missing entries of $\boldsymbol{A}$ are filled in with 0s) and we initialize dual iterates $\boldsymbol{\Phi}_0 = \boldsymbol{\Psi}_0 = \boldsymbol{1}_{n\times k}$. Observe that the subproblems (12) and (16) can be solved simultaneously. Similarly, the subprobems (14) and (18) can be solved simultaneously. At each iteration of Algorithm 1, we first update the iterates $\boldsymbol{U}_{t+1}, \boldsymbol{P}_{t+1}$ by solving problems (12) and (16) with $(\boldsymbol{V}_t, \boldsymbol{Z}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)$ fixed. Next, we update the iterates $\boldsymbol{V}_{t+1}, \boldsymbol{Z}_{t+1}$ by solving problems (14) and (18) with $(\boldsymbol{U}_{t+1}, \boldsymbol{P}_{t+1}, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)$ fixed. Finally, we update the dual iterates $\boldsymbol{\Phi}, \boldsymbol{\Psi}$ by taking a gradient ascent step. The gradients of the augmented Lagrangian (11) with respect to $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are given by the primal residuals $(\boldsymbol{I}_n - \boldsymbol{P}_{t+1})\boldsymbol{Z}_{t+1}$ and $\boldsymbol{Z}_{t+1} - \boldsymbol{U}_{t+1}$

Algorithm 1 Mixed-Projection ADMM

Require: $n, m, k \in \mathbb{Z}_+, \Omega \subset [n] \times [m], \{A_{ij}\}_{(i,j)\in\Omega}, \lambda, \gamma \in \mathbb{R}_+$. Tolerance parameter $\epsilon > 0$. Maximum iteration parameter $T \in \mathbb{Z}_+$.

Ensure: $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ that is feasible to (8).

$(\boldsymbol{U}_0, \boldsymbol{P}_0, \boldsymbol{V}_0, \boldsymbol{Z}_0) \leftarrow (\boldsymbol{L}\boldsymbol{\Sigma}^{\frac{1}{2}}, \boldsymbol{L}\boldsymbol{L}^T, \boldsymbol{R}\boldsymbol{\Sigma}^{\frac{1}{2}}, \boldsymbol{L}\boldsymbol{\Sigma}^{\frac{1}{2}})$ where $\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}$ is a rank k truncated SVD of $\boldsymbol{A}$ and missing entries are filled in with 0;
$(\boldsymbol{\Phi}_0, \boldsymbol{\Psi}_0) \leftarrow (\boldsymbol{1}_{n\times k}, \boldsymbol{1}_{n\times k})$;
$t \leftarrow 0$;
while $t < T$ and $\max\{\|(\boldsymbol{I}_n - \boldsymbol{P}_t)\boldsymbol{Z}_t\|_F^2, \|\boldsymbol{Z}_t - \boldsymbol{U}_t\|_F^2\} > \epsilon$ **do**
  $(\boldsymbol{U}_{t+1}, \boldsymbol{P}_{t+1}) \leftarrow \arg\min_{\boldsymbol{U},\boldsymbol{P}} \mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}_t, \boldsymbol{P}, \boldsymbol{Z}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)$;
  $(\boldsymbol{V}_{t+1}, \boldsymbol{Z}_{t+1}) \leftarrow \arg\min_{\boldsymbol{V},\boldsymbol{Z}} \mathcal{L}^A(\boldsymbol{U}_{t+1}, \boldsymbol{V}, \boldsymbol{P}_{t+1}, \boldsymbol{Z}, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)$;
  $\boldsymbol{\Phi}_{t+1} \leftarrow \boldsymbol{\Phi}_t + \rho_1(\boldsymbol{I} - \boldsymbol{P}_{t+1})\boldsymbol{Z}_{t+1}$;
  $\boldsymbol{\Psi}_{t+1} \leftarrow \boldsymbol{\Psi}_t + \rho_2(\boldsymbol{Z}_{t+1} - \boldsymbol{U}_{t+1})$;
  $t \leftarrow t + 1$;
end while
Return $(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t)$.

respectively. We use ρ_1 and ρ_2 respectively as the step size. We proceed until the squared norm of each primal residual is below a numerical tolerance parameter ϵ or until we reach an input maximum number of iterations T. We know have the following result:

Proposition 12. *Assume that the number of compute threads w is less than $\min\{n, m\}$. The per iteration complexity of Algorithm 1 is $O\big(k^2 n + knd + \frac{k^3(n+m)+k^2nm}{w}\big)$.*

Proof. The result follows from the complexity analysis of problems (12), (14), (16) and (18). □

Having presented Algorithm 1 in extensive detail, it is natural to consider what types of guarantees can be made on the final output solution $(\boldsymbol{U}_T, \boldsymbol{V}_T, \boldsymbol{P}_T)$. We explore this in the following theorem:

Theorem 13. *Let $\{(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t, \boldsymbol{Z}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)\}$ denote a sequence generated by Algorithm 1 (assuming we allow Algorithm 1 to iterate indefinitely). Suppose the dual variable sequence $\{(\boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)\}$ is bounded and satisfies*

$$\sum_{t=0}^{\infty}(\|\boldsymbol{\Phi}_{t+1} - \boldsymbol{\Phi}_t\|_F^2 + \|\boldsymbol{\Psi}_{t+1} - \boldsymbol{\Psi}_t\|_F^2) < \infty. \tag{20}$$

Let $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ denote any accumulation point of $\{(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t, \boldsymbol{Z}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)\}$. If the set of k leading eigenvectors of the matrix $[\lambda\boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$ is the same as the set of k leading eigenvectors of the matrix $\bar{\boldsymbol{Z}}\bar{\boldsymbol{Z}}^T$, then $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ satisfies the first order optimality conditions for (10).

Proof. We leverage a proof technique similar to the technique used to establish Theorem 2.1 from [57]. Note that from (20), we immediately have $\boldsymbol{\Phi}_{t+1} - \boldsymbol{\Phi}_t \to 0$ and

$\boldsymbol{\Psi}_{t+1} - \boldsymbol{\Psi}_t \to 0$. We will first show that we also have $\boldsymbol{U}_{t+1} - \boldsymbol{U}_t \to 0, \boldsymbol{V}_{t+1} - \boldsymbol{V}_t \to 0$ and $\boldsymbol{Z}_{t+1} - \boldsymbol{Z}_t \to 0$.

Let $\mathcal{L}^A(\boldsymbol{U}; \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ denote the augmented Lagrangian (11) viewed as a function of $\boldsymbol{U}$. Notice that $\mathcal{L}^A(\boldsymbol{U}; \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}) - \frac{\gamma+\rho_2}{2}\|\boldsymbol{U}\|_F^2$ is a convex function, which implies that $\mathcal{L}^A(\boldsymbol{U}; \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ is a strongly convex function of $\boldsymbol{U}$ with parameter $\gamma+\rho_2$. Similarly, $\mathcal{L}^A(\boldsymbol{V}; \boldsymbol{U}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ is a strongly convex function of $\boldsymbol{V}$ with parameter γ and $\mathcal{L}^A(\boldsymbol{Z}; \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ is a strongly convex function of $\boldsymbol{Z}$ with parameter ρ_2. By strong convexity, we know that for any matrices $\boldsymbol{U}, \boldsymbol{\Delta U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ we have

$$\mathcal{L}^A(\boldsymbol{U} + \boldsymbol{\Delta U}) - \mathcal{L}^A(\boldsymbol{U}) \geq \langle \nabla_{\boldsymbol{U}} \mathcal{L}^A(\boldsymbol{U}), \boldsymbol{\Delta U} \rangle + (\gamma + \rho_2)\|\boldsymbol{\Delta U}\|_F^2, \tag{21}$$

where the shorthand notation $\mathcal{L}^A(\boldsymbol{U})$ is understood to denote the augmented Lagrangian (11) viewed as a function only of $\boldsymbol{U}$ with the other variables held fixed. Letting $\boldsymbol{U} = \boldsymbol{U}_{t+1}, \boldsymbol{\Delta U} = \boldsymbol{U}_t - \boldsymbol{U}_{t+1}$, and noting that $\langle \nabla_{\boldsymbol{U}} \mathcal{L}^A(\boldsymbol{U}_{t+1}), \boldsymbol{\Delta U} \rangle \geq 0$ since $\boldsymbol{U}_{t+1}$ minimizes $\mathcal{L}^A(\boldsymbol{U})$ at iteration t, from (21) we have $\mathcal{L}^A(\boldsymbol{U}_t) - \mathcal{L}^A(\boldsymbol{U}_{t+1}) \geq (\gamma + \rho_2)\|\boldsymbol{U}_t - \boldsymbol{U}_{t+1}\|_F^2$. Similarly, we have $\mathcal{L}^A(\boldsymbol{V}_t) - \mathcal{L}^A(\boldsymbol{V}_{t+1}) \geq \gamma\|\boldsymbol{V}_t - \boldsymbol{V}_{t+1}\|_F^2$ and $\mathcal{L}^A(\boldsymbol{Z}_t) - \mathcal{L}^A(\boldsymbol{Z}_{t+1}) \geq \rho_2\|\boldsymbol{Z}_t - \boldsymbol{Z}_{t+1}\|_F^2$. Moreover, since $\boldsymbol{P}_{t+1}$ minimizes $\mathcal{L}^A(\boldsymbol{P})$ at iteration t, we have $\mathcal{L}^A(\boldsymbol{P}_t) - \mathcal{L}^A(\boldsymbol{P}_{t+1}) \geq 0$. Observe that we can express the difference in the value of the augmented Lagrangian between iteration t and iteration $t+1$ as

$$\begin{aligned}\mathcal{L}^A(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t, \boldsymbol{Z}_t) - \mathcal{L}^A(\boldsymbol{U}_{t+1}, \boldsymbol{V}_{t+1}, \boldsymbol{P}_{t+1}, \boldsymbol{Z}_{t+1}) = \\ \mathcal{L}^A(\boldsymbol{U}_t) - \mathcal{L}^A(\boldsymbol{U}_{t+1}) + \mathcal{L}^A(\boldsymbol{V}_t) - \mathcal{L}^A(\boldsymbol{V}_{t+1}) \\ + \mathcal{L}^A(\boldsymbol{P}_t) - \mathcal{L}^A(\boldsymbol{P}_{t+1}) + \mathcal{L}^A(\boldsymbol{Z}_t) - \mathcal{L}^A(\boldsymbol{Z}_{t+1}) \\ + \mathcal{L}^A(\boldsymbol{\Phi}_t) - \mathcal{L}^A(\boldsymbol{\Phi}_{t+1}) + \mathcal{L}^A(\boldsymbol{\Psi}_t) - \mathcal{L}^A(\boldsymbol{\Psi}_{t+1}).\end{aligned} \tag{22}$$

Recognizing that we have $\mathcal{L}^A(\boldsymbol{\Phi}_t) - \mathcal{L}^A(\boldsymbol{\Phi}_{t+1}) = -\rho_1\|\boldsymbol{\Phi}_t - \boldsymbol{\Phi}_{t+1}\|_F^2, \mathcal{L}^A(\boldsymbol{\Psi}_t) - \mathcal{L}^A(\boldsymbol{\Psi}_{t+1}) = -\rho_2\|\boldsymbol{\Psi}_t - \boldsymbol{\Psi}_{t+1}\|_F^2$, (22) implies that

$$\begin{aligned}\mathcal{L}^A(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t, \boldsymbol{Z}_t) - \mathcal{L}^A(\boldsymbol{U}_{t+1}, \boldsymbol{V}_{t+1}, \boldsymbol{P}_{t+1}, \boldsymbol{Z}_{t+1}) \geq \\ (\gamma + \rho_2)\|\boldsymbol{U}_t - \boldsymbol{U}_{t+1}\|_F^2 + \gamma\|\boldsymbol{V}_t - \boldsymbol{V}_{t+1}\|_F^2 + \rho_2\|\boldsymbol{Z}_t - \boldsymbol{Z}_{t+1}\|_F^2 \\ - \rho_1\|\boldsymbol{\Phi}_t - \boldsymbol{\Phi}_{t+1}\|_F^2 - \rho_2\|\boldsymbol{\Psi}_t - \boldsymbol{\Psi}_{t+1}\|_F^2.\end{aligned} \tag{23}$$

We claim that the augmented Lagrangian is bounded from below. To see this, note that $\mathcal{L}^A$ can equivalently be written as

$$\begin{aligned}\mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}) = \sum_{(i,j)\in\Omega} ((\boldsymbol{U}\boldsymbol{V}^T)_{ij} - A_{ij})^2 + \lambda\|(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Y}\|_F^2 \\ + \frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) + \mathbb{I}_{\mathcal{P}_k}(\boldsymbol{P}) + \frac{\rho_1}{2}\|(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z} + \frac{\boldsymbol{\Phi}}{\rho_1}\|_F^2 \\ + \frac{\rho_2}{2}\|\boldsymbol{Z} - \boldsymbol{U} + \frac{\boldsymbol{\Psi}}{\rho_2}\|_F^2 - \frac{1}{2\rho_1}\|\boldsymbol{\Phi}\|_F^2 + \frac{1}{2\rho_2}\|\boldsymbol{\Psi}\|_F^2,\end{aligned} \tag{24}$$

and recall that by assumption the dual variables $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are bounded. Thus, the bounded-ness of $\mathcal{L}^A$ coupled with summing (23) over t implies that

$$\begin{aligned}&\sum_{t=0}^{\infty} c_1(\|\boldsymbol{U}_t - \boldsymbol{U}_{t+1}\|_F^2 + \|\boldsymbol{V}_t - \boldsymbol{V}_{t+1}\|_F^2 + \|\boldsymbol{Z}_t - \boldsymbol{Z}_{t+1}\|_F^2) \\ &- \sum_{t=0}^{\infty} c_2(\|\boldsymbol{\Phi}_t - \boldsymbol{\Phi}_{t+1}\|_F^2 + \|\boldsymbol{\Psi}_t - \boldsymbol{\Psi}_{t+1}\|_F^2) < \infty,\end{aligned} \tag{25}$$

where $c_1 = \min\{\gamma, \rho_2\}$ and $c_2 = \max\{\rho_1, \rho_2\}$. By assumption, the second term of (25) is finite which implies that the first term must also be finite. This immediately implies that $\boldsymbol{U}_{t+1} - \boldsymbol{U}_t \to 0, \boldsymbol{V}_{t+1} - \boldsymbol{V}_t \to 0$ and $\boldsymbol{Z}_{t+1} - \boldsymbol{Z}_t \to 0$ as desired.

We are now ready to prove the main result of the theorem. The (unaugmented) Lagrangian $\mathcal{L}$ for (10) is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}) = &\sum_{(i,j)\in\Omega} ((\boldsymbol{U}\boldsymbol{V}^T)_{ij} - A_{ij})^2 + \lambda \mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Y}\big) \\ &+ \frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) + \mathbb{I}_{\mathcal{P}_k}(\boldsymbol{P}) + \mathrm{Tr}(\boldsymbol{\Phi}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}) + \mathrm{Tr}(\boldsymbol{\Psi}^T(\boldsymbol{Z} - \boldsymbol{U})).\end{aligned} \tag{26}$$

The corresponding first order optimality conditions can be expressed as

$$[2\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V} + \gamma\boldsymbol{I}_k]U_{i,\star} = 2\boldsymbol{V}^T\boldsymbol{W}_i A_{i,\star} + \Psi_{i,\star} \quad i \in [n], \tag{27a}$$

$$[2\boldsymbol{U}^T\boldsymbol{W}_j\boldsymbol{U} + \gamma\boldsymbol{I}_k]V_{j,\star} = 2\boldsymbol{U}^T\boldsymbol{W}_j A_{\star,j} \quad j \in [m], \tag{27b}$$

$$\boldsymbol{P} = \boldsymbol{M}\boldsymbol{M}^T \text{where } \boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T \text{ is a rank } k \text{ SVD of } \lambda\boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{Z}^T + \boldsymbol{Z}\boldsymbol{\Phi}^T), \tag{27c}$$

$$\boldsymbol{\Phi} + \boldsymbol{\Psi} = \boldsymbol{P}\boldsymbol{\Phi}, \tag{27d}$$

$$\boldsymbol{Z} = \boldsymbol{P}\boldsymbol{Z}, \tag{27e}$$

$$\boldsymbol{Z} = \boldsymbol{U}, \tag{27f}$$

where the diagonal matrices $\boldsymbol{W}_i, \boldsymbol{W}_j$ are defined as in Propositions 8 and 9 respectively. Let $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ denote any limit point of $\{(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t, \boldsymbol{Z}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)\}$. Recalling the Algorithm 1 updates for $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$, the conditions $\boldsymbol{\Phi}_{t+1} - \boldsymbol{\Phi}_t \to 0$ and $\boldsymbol{\Psi}_{t+1} - \boldsymbol{\Psi}_t \to 0$ imply that (27e) and (27f) hold at $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$. Moreover, when (27f) holds the Algorithm 1 update for $\boldsymbol{U}$ given by Proposition 8 reduces to (27a) while the update for $\boldsymbol{V}$ given by Proposition 9 enforces (27b). From the proof of Propsition (11), we know that Algorithm 1 updates $\boldsymbol{Z}$ to satisfy the following

$$\Big(\rho_1(\boldsymbol{I}_n - \boldsymbol{P}) + \rho_2\boldsymbol{I}_n\Big)\boldsymbol{Z} = \Big(\rho_2\boldsymbol{U} - (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{\Phi} - \boldsymbol{\Psi}\Big). \tag{28}$$

Since (27e) and (27f) hold, (28) immediately implies (27d) is satisfied by $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$. It remains to verify that $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ satisfies (27c). By Proposition 10, we know that we have $\bar{\boldsymbol{P}} = \boldsymbol{L}\boldsymbol{L}^T$ where $\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T$ is a rank k truncated SVD of the matrix $[\lambda\boldsymbol{Y}\boldsymbol{Y}^T + \frac{\rho_1}{2}\bar{\boldsymbol{Z}}\bar{\boldsymbol{Z}}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$. If the set of k leading

eigenvectors of the matrix $\bar{\boldsymbol{Z}}\bar{\boldsymbol{Z}}^T$ is the same as the set of k leading eigenvectors of $[\lambda \boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$, it follows immediately that $\boldsymbol{L}\Sigma'\boldsymbol{L}^T$ will be a rank k SVD of $[\lambda \boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$ for some diagonal matrix Σ'. Thus, in this setting, $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ satisfies (27c). this completes the proof.

□

In words, Theorem 13 states that if the sequence of dual variable iterates produced by Algorithm 1 is bounded and the primal residuals converge to zero quickly enough (specifically, it is required that the norm of successive dual variable differences is summable), then any accumulation point $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ of the sequence of iterates produced by Algorithm 1 satisfies the first order optimality conditions of (10) if the rank k approximation of the matrix $[\lambda \boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$ shares the same column space as the matrix $\bar{\boldsymbol{Z}}$. We note that the assumption made in Theorem 13 is consistent with common assumptions made in the analysis of nonconvex ADMM [57–59]. We also note that this condition can only be verified upon termination of Algorithm 1 since it depends on the algorithm output in addition to the problem data. Accordingly, Theorem 13 provides an a posteriori convergence result.

We note that this condition was always satisfied by the output of Algorithm 1 in our synthetic numerical experiments. Specifically, letting $\boldsymbol{P}_1 \in \mathbb{R}^{n\times n}$ denote the orthogonal projection onto the k-dimensional column space of $\bar{\boldsymbol{Z}}$ and $\boldsymbol{P}_2 \in \mathbb{R}^{n\times n}$ denote the orthogonal projection onto the k leading eigenvectors of $[\lambda \boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$, the eigenvector condition stated in Theorem 13 is satisfied if and only if we have $(\boldsymbol{I}_n - \boldsymbol{P_1})\boldsymbol{P}_2 = \boldsymbol{0}_{n\times n}$. This suggests that the quantity $\boldsymbol{P}_2 - \boldsymbol{P_1}\boldsymbol{P}_2$ can be viewed as a dual residual for Algorithm 1 (recall that the primal residuals are given by $\boldsymbol{Z} - \boldsymbol{P}\boldsymbol{Z}$ and $\boldsymbol{Z} - \boldsymbol{U}$, which we term the $\boldsymbol{\Phi}$-residual and the $\boldsymbol{\Psi}$-residual respectively). In Figure 1, we plot the evolution of the primal and dual residuals over 500 iterations of Algorithm 1 for a single synthetic data run (see Section 6.1 for a specification of the data generation procedure) where we have fixed $\rho_1 = \rho_2 = 10$. Observe that after only a small number of iterations, the norm of the $\boldsymbol{\Phi}$-residual and the norm of the $\boldsymbol{\Psi}$-residual quickly approach 0, indicating that Algorithm 1 has arrived at a feasible solution. Moreover, the norm of the dual residual similarly quickly approaches 0 after a small number of iterations, indicating that the eigenvector condition from Theorem 13 is satisfied and that Algorithm 1 has therefore arrived at a solution satisfying first order optimality conditions.

6 Computational Results

We evaluate the performance of Algorithm 1 implemented in Julia 1.7.3. Throughout, we fix $\rho_1 = \rho_2 = 10$, set the maximum number of iterations $T = 20$ and set the number of compute threads $w = 24$. Note that given the novelty of Problem (1), there are no pre-existing specialized methods to benchmark against. Accordingly, we compare the performance of Algorithm 1 against well studied methods for the very closely related MC problem, a method designed for MC in the presence of side information and a highly performant generic method for low rank matrix optimization problems. The MC methods we consider are Fast-Impute [9], Soft-Impute [23] and Iterative-SVD [20] which we introduced formally in Section 2.1. We utilize the implementation

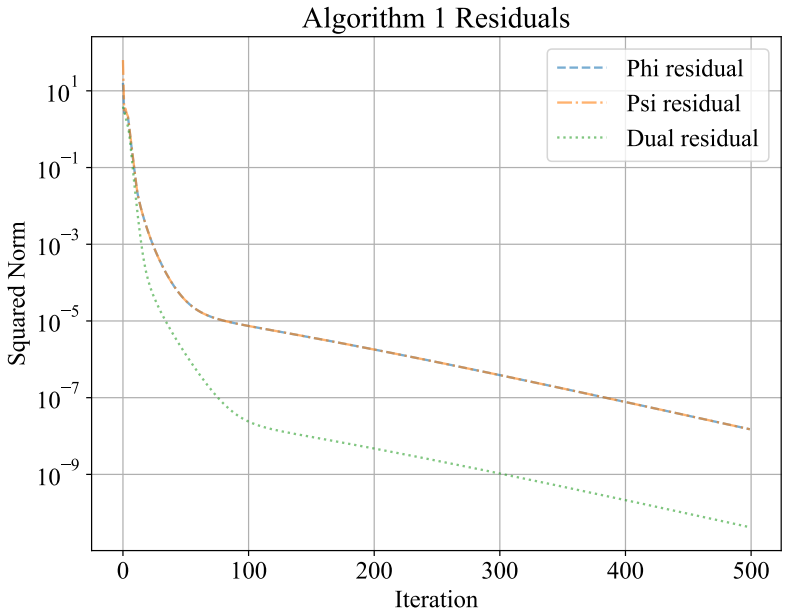

Fig. 1 Algorithm 1 primal and dual residual evolution versus iteration number for a single synthetic data run with $n = 1000, m = 100, k = 5$ and $d = 150$. Note that due to the logarithmic scale, the phi residual and psi residual lines are overlapping.

of Fast-Impute made publicly available by [9] while we use the implementation of Soft-Impute and Iterative-SVD from the python package fancyimpute 0.7.0 [60]. We additionally consider the variant of Fast-Impute that incorporates side information (denoted as Fast-Impute-Side) which is also made publicly available by [9]. The matrix optimization method we consider is ScaledGD (scaled gradient descent) [37] which we introduced formally in Section 2.2.1 and implement ourselves. We perform experiments using both synthetic data and real world data on MIT's Supercloud Cluster [61], which hosts Intel Xeon Platinum 8260 processors. To bridge the gap between theory and practice, we have made our code freely available on GitHub at https://github.com/NicholasJohnson2020/LearningLowRankMatrices.

To evaluate the performance of Algorithm 1, Fast-Impute, Fast-Impute-Side, Soft-Impute, Iterative-SVD and ScaledGD on synthetic data, we consider the objective value achieved by a returned solution in (1), the ℓ_2 reconstruction error between a returned solution and the ground truth, the coefficient of determination (R^2) when the returned solution is used as a predictor for the side information, the numerical rank of a returned solution and the execution time of each algorithm. Explicitly, let $\hat{\boldsymbol{X}} \in \mathbb{R}^{n \times m}$ denote the solution returned by a given method (where we define $\hat{\boldsymbol{X}} = \hat{\boldsymbol{U}}\hat{\boldsymbol{V}}^T$ if the method outputs low rank factors $\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}$) and let $\boldsymbol{A}^{true} \in \mathbb{R}^{n \times m}$ denote the ground

truth matrix. We define the the ℓ_2 reconstruction error of $\hat{\boldsymbol{X}}$ as

$$ERR_{\ell_2}(\hat{\boldsymbol{X}}) = \frac{\|\hat{\boldsymbol{X}} - \boldsymbol{A}^{true}\|_F^2}{\|\boldsymbol{A}^{true}\|_F^2}.$$

We compute the numerical rank of $\hat{\boldsymbol{X}}$ by calling the default rank function from the Julia LinearAlgebra package. We aim to answer the following questions:

1. How does the performance of Algorithm 1 compare to existing methods such as Fast-Impute, Fast-Impute-Side, Soft-Impute, Iterative-SVD and ScaledGD on synthetic and real world data?
2. How is the performance of Algorithm 1 affected by the number of rows n, the number of columns m, the dimension of the side information d and the underlying rank k of the ground truth?
3. Empirically, which subproblem solution update is the computational bottleneck of Algorithm 1?

6.1 Synthetic Data Generation

To generate synthetic data, we specify a number of rows $n \in \mathbb{Z}_+$, a number of columns $m \in \mathbb{Z}_+$, a desired rank $k \in \mathbb{Z}_+$ with $k < \min\{n, m\}$, the dimension of the side information $d \in \mathbb{Z}_+$, a fraction of missing values $\alpha \in (0, 1)$ and a noise parameter $\sigma \in \mathbb{R}_+$ that controls the signal to noise ratio. We sample matrices $\boldsymbol{U} \in \mathbb{R}^{n\times k}, \boldsymbol{V} \in \mathbb{R}^{m\times k}, \boldsymbol{\beta} \in \mathbb{R}^{m\times d}$ by drawing each entry $U_{ij}, V_{ij}, \beta_{ij}$ independently from the uniform distribution on the interval $[0, 1]$. Furthermore, we sample a noise matrix $\boldsymbol{N} \in \mathbb{R}^{n\times d}$ by drawing each entry N_{ij} independently from the univariate normal distribution with mean 0 and variance σ^2. We let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}^T$ and we let $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\beta} + \boldsymbol{N}$. Lastly, we sample $\lfloor \alpha \cdot n \cdot m \rfloor$ indices uniformly at random from the collection $\mathcal{I} = \{(i, j) : 1 \leq i \leq n, 1 \leq j \leq m\}$ to be the set of missing indices, which we denote by Γ. The set of revealed entries can then be defined as $\Omega = \mathcal{I} \setminus \Gamma$. We fix $\alpha = 0.9, \sigma = 2$ throughout our experiments and report numerical results for various different combinations of (n, m, d, k).

6.2 Sensitivity to Row Dimension

We present a comparison of Algorithm 1 with ScaledGD, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD as we vary the number of rows n. In these experiments, we fixed $m = 100, k = 5$, and $d = 150$ across all trials. We varied $n \in \{100, 200, 400, 800, 1000, 2000, 5000, 10000\}$ and we performed 20 trials for each value of n. For ScaledGD, we set the step size to be $\eta = \frac{1}{10\sigma_1(\boldsymbol{A})}$ where $\sigma_1(\boldsymbol{A})$ denotes the largest singular value of the input matrix $\boldsymbol{A}$ where we fill the unobserved entries with the value 0. Letting $f(\boldsymbol{U}_t, \boldsymbol{V}_t)$ denote the objective value achieved after iteration t of ScaledGD, we terminate ScaledGD when either $t > 1000$ or $\frac{f(\boldsymbol{U}_{t-1},\boldsymbol{V}_{t-1}) - f(\boldsymbol{U}_t,\boldsymbol{V}_t)}{f(\boldsymbol{U}_{t-1},\boldsymbol{V}_{t-1})} < 10^{-3}$. In words, we terminate ScaledGD after 1000 iterations or after the relative objective value improvement between two iterations is less than 0.1%.

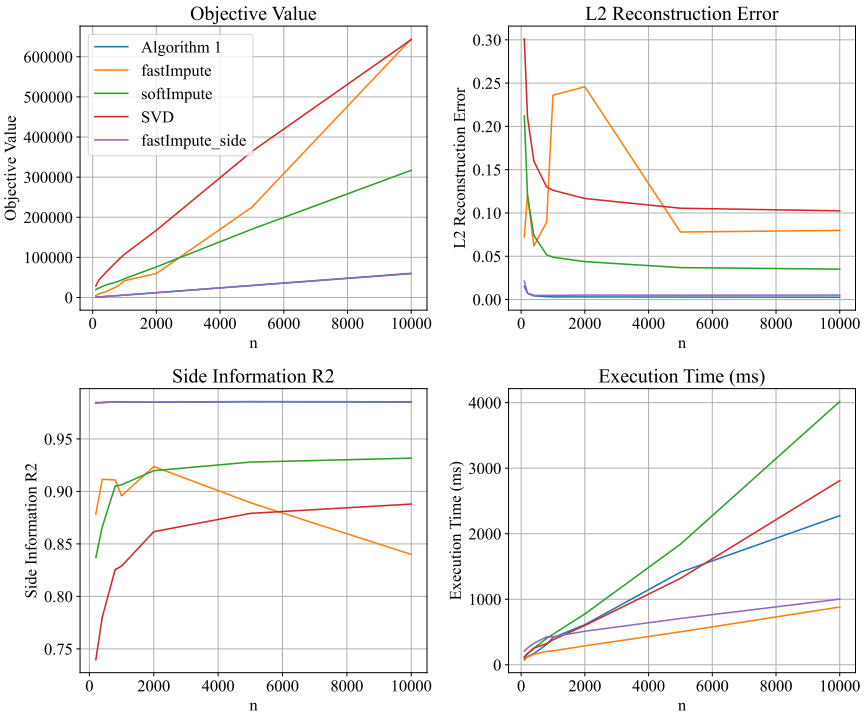

Fig. 2 Objective value (top left), ℓ_2 reconstruction error (top right), side information R^2 (bottom left) and execution time (bottom right) versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

We report the objective value, ℓ_2 reconstruction error, side information R^2 and execution time for Algorithm 1, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD in Figure 2. For ease of comparison between Algorithm 1 and Fast-Impute-Side, we plot only the performance of these two methods in Figure 3. We additionally report the objective value, reconstruction error, side information R^2 and execution time for ScaledGD, Algorithm 1, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD in Tables A1, A2, A3 and A4 of Appendix A. In Figure 4, we plot the average cumulative time spent solving subproblems (12), (14), (16), (18) during the execution of Algorithm 1 versus n. Our main findings from this set of experiments are:

1. Algorithm 1 and Fast-Impute-Side systematically produce higher quality solutions than ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD (see Table A1), sometimes achieving an objective value that is an order of magnitude superior than the next best method. We remind the reader that Fast-Impute, Soft-Impute and Iterative-SVD are methods designed for the generic MC problem and are not custom built to solve (1) so it should not come as a surprise that Algorithm 1 and Fast-Impute-Side (which leverages the side information $\boldsymbol{Y}$) significantly outperform these 3 methods in terms of objective value. ScaledGD however has explicit

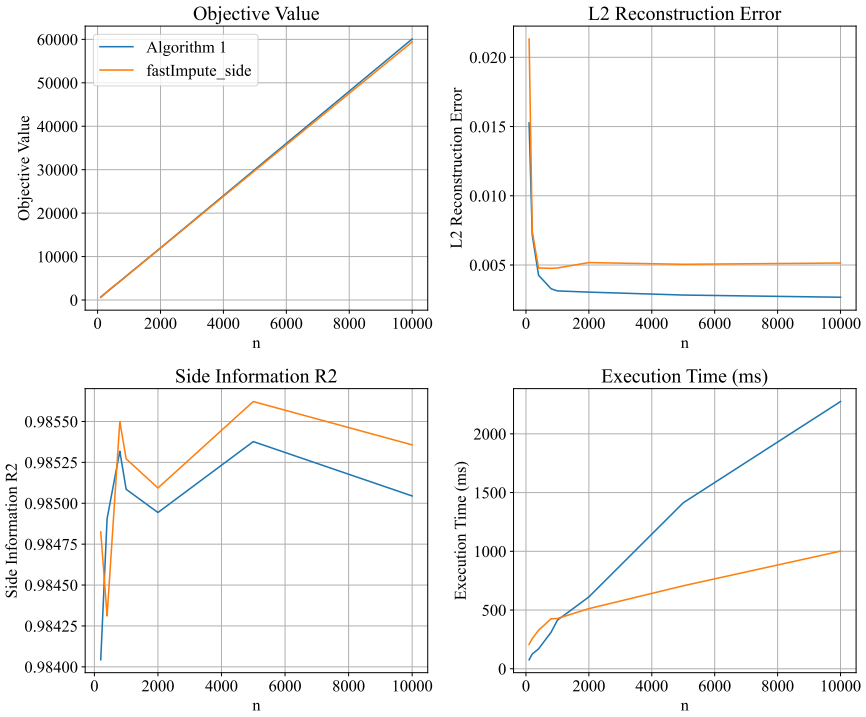

Fig. 3 Objective value (top left), ℓ_2 reconstruction error (top right), side information R^2 (bottom left) and execution time (bottom right) versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

knowledge of the objective function of (1) along with its gradient, yet surprisingly produces the weakest average objective value across these experiments. We note that we use the default hyperparameters for ScaledGD recommended by the authors of this method [37]. We observe that the objective value achieved by all methods increases linearly as the number of rows n increases. There is no significant difference between the objective value achieved by the output of Algorithm 1 compared to that of Fast-Impute-Side (Algorithm 1 is on average 1% higher).

2. In terms of ℓ_2 reconstruction error, Algorithm 1 systematically produces solutions that are of higher quality than ScaledGD, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD (see Table A2). On average, Algorithm 1 outputs a solution whose ℓ_2 reconstruction error is 30% lesser than the reconstruction error achieved by the best performing alternative method (Fast-Impute-Side). This is especially noteworthy since Algorithm 1 is not designed explicitly with reconstruction error minimization as the objective, unlike Fast-Impute, Fast-Impute-Side and Soft-Impute, and suggests that modeling the side information $\boldsymbol{Y}$ as a linear function of $\boldsymbol{X}$ is instrumental in recovering high quality low rank estimates of the partially observed data matrix.

3. With the exception of the experiments for which $n = 100$, Algorithm 1 and Fast-Impute-Side always produced solutions that achieved a superior R^2 value when used as a predictor for the side information compared to Fast-Impute, Soft-Impute, Iterative-SVD and ScaledGD. There is no significant difference between R^2 value achieved by the output of Algorithm 1 compared to that of Fast-Impute-Side.
4. The runtime of Algorithm 1 is competitive with that of the other methods. The runtime of Algorithm 1 is less than of Soft-Impute and Iterative-SVD but greater than that of Fast-Impute. The runtime for Algorithm 1 was less than that of Fast-Impute-Side for $n < 2000$ but greater than for $n \geq 2000$. For experiments with $n \leq 2000$, Table A4 illustrates that ScaledGD was the method with the fastest execution time (however as previously mentioned the returned solutions were of low quality). The runtime of Algorithm 1, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD appear to grow linearly with n.
5. Figure 4 illustrates that the computation of the solution for (12) is the computational bottleneck in the execution of Algorithm 1 in this set of experiments, followed next by the computation of the solution for (16). Empirically, we observe that the solution time of (12), (14), (16) and (18) appear to scale linearly with the number of rows n. This observation is consistent with the computational complexities derived for each subproblem of Algorithm 1 in Section 5.

6.3 Sensitivity to Column Dimension

Here, we present a comparison of Algorithm 1 with ScaledGD, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD as we vary the number of columns m. We fixed $n = 1000, k = 5$, and $d = 150$ across all trials. We varied $m \in \{100, 200, 400, 800, 1000, 2000, 5000, 10000\}$ and we performed 20 trials for each value of m.

We report the objective value, ℓ_2 reconstruction error, fitted rank and execution time for Algorithm 1, Fast-Impute, Fast-Impute-Side and Soft-Impute in Figure 5. For ease of comparison between Algorithm 1 and Fast-Impute-Side, we plot only the performance of these two methods in Figure 6. We additionally report the objective value, reconstruction error and execution time for ScaledGD, Algorithm 1, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD in Tables A5, A6 and A7 of Appendix A. In Figure 7, we plot the average cumulative time spent solving subproblems (12), (14), (16), (18) during the execution of Algorithm 1 versus m. Our main findings from this set of experiments are a follows:

1. Here again, Algorithm 1 and Fast-Impute-Side systematically produce higher quality solutions than ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD (see Table A5). For trials with $m \leq 5000$, Algorithm 1 outputs a solution whose objective value is on average 12% lesser than the objective value achieved by the best performing alternative method (Fast-Impute-Side). Fast-Impute-Side produced solutions with lower objective values for $m = 10000$. Here again, ScaledGD produces the weakest average objective value across these experiments.
2. In terms of ℓ_2 reconstruction error, Algorithm 1 again systematically produces solutions that are of higher quality than ScaledGD, Fast-Impute, Fast-Impute-Side,

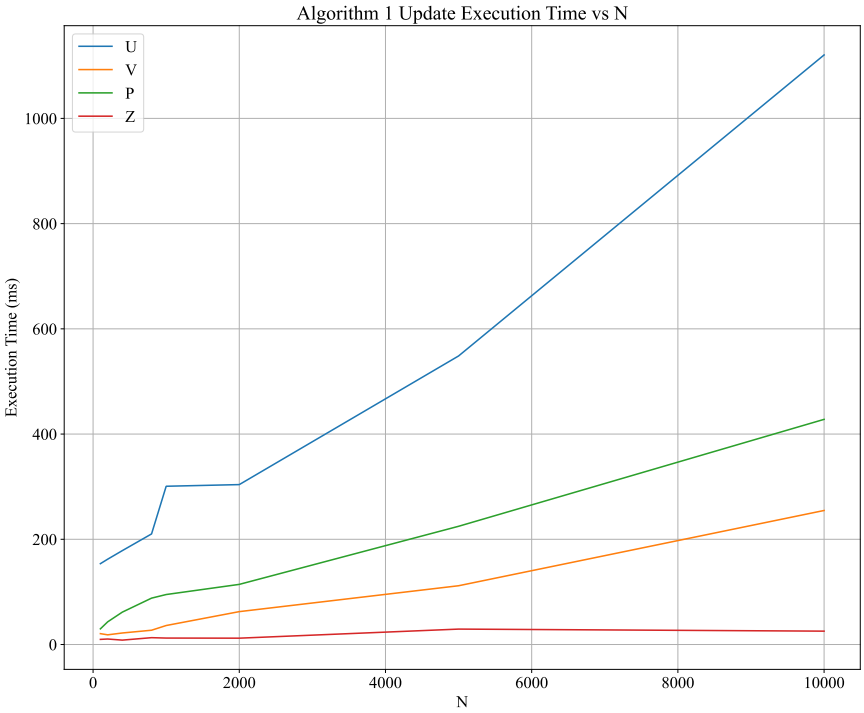

Fig. 4 Cumulative time spent solving each subproblem of Algorithm 1 versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

Soft-Impute and Iterative-SVD (see Table A6), often achieving an error that is an order of magnitude superior than the next best method. On average, Algorithm 1 outputs a solution whose ℓ_2 reconstruction error is 77% lesser than the reconstruction error achieved by the best performing alternative method (either Fast-Impute-Side or Soft-Impute).

3. We observe that the fitted rank of the solutions returned by Algorithm 1, ScaledGD and Fast-Impute always matched the specified target rank as would be expected, but surprisingly the solutions returned by Soft-Impute and Iterative-SVD were always of full rank despite the fact that these methods were provided with the target rank explicitly. This is potentially due to a numerical issues in the computation of the rank due to presence of extremely small singular values.
4. The runtime of Algorithm 1 exhibits the most favorable scaling behavior among the methods tested in these experiments. For instances with $m \geq 2000$, Table A7 shows that Algorithm 1 had the fastest runtime. For instances with $m < 2000$, ScaledGD had the fastest execution time but produced low quality solutions. The runtime of all methods tested grow super-linearly with m.
5. Figure 7 illustrates that the computation of the solution for (12) and (14) are the computational bottlenecks in the execution of Algorithm 1 in this set of experiments while the computation of the solution for (18) and (16) appear to be a constant

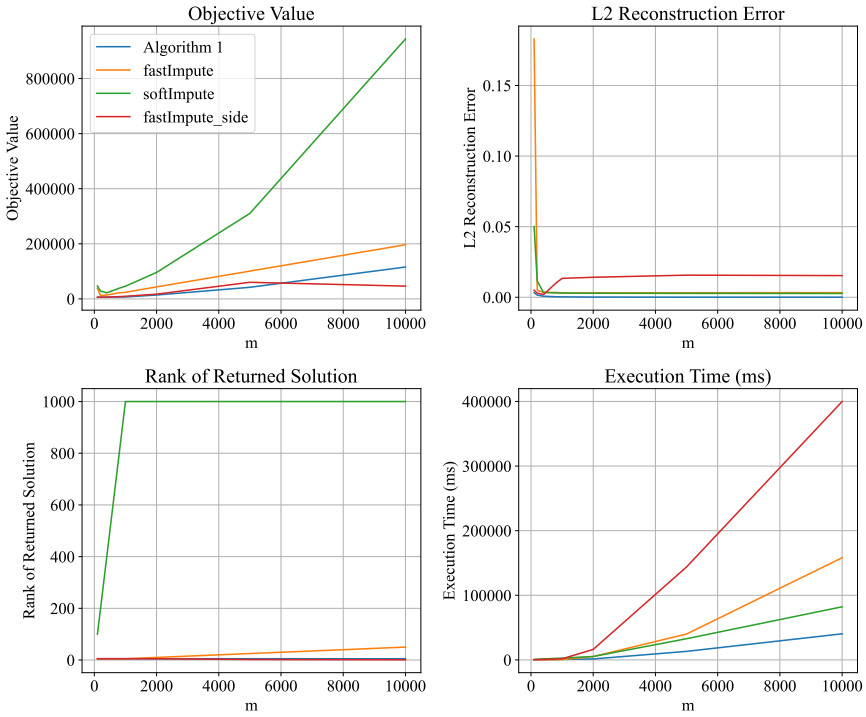

Fig. 5 Objective value (top left), ℓ_2 reconstruction error (top right), fitted rank (bottom left) and execution time (bottom right) versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

function of m. This observation is consistent with the complexity analysis performed for each subproblem of Algorithm 1 in Section 5. Indeed, this analysis indicated that solve times for (18) and (16) are independent of m while the solve times for (12) and (14) scale linearly with m when the number of threads w satisfies $w < m$.

6.4 Sensitivity to Side Information Dimension

We present a comparison of Algorithm 1 with ScaledGD, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD as we vary the dimension of the side information d. In these experiments, we fixed $n = 1000, m = 100$ and $k = 5$ across all trials. We considered values of d in the collection $\{10, 50, 100, 150, 200, 250, 500, 1000\}$ and we performed 20 trials for each value of d.

We report the objective value, ℓ_2 reconstruction error, side information R^2 and execution time for Algorithm 1, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD in Figure 8. For ease of comparison between Algorithm 1 and Fast-Impute-Side, we plot only the performance of these two methods in Figure 9. We additionally report the objective value, reconstruction error, side information R^2

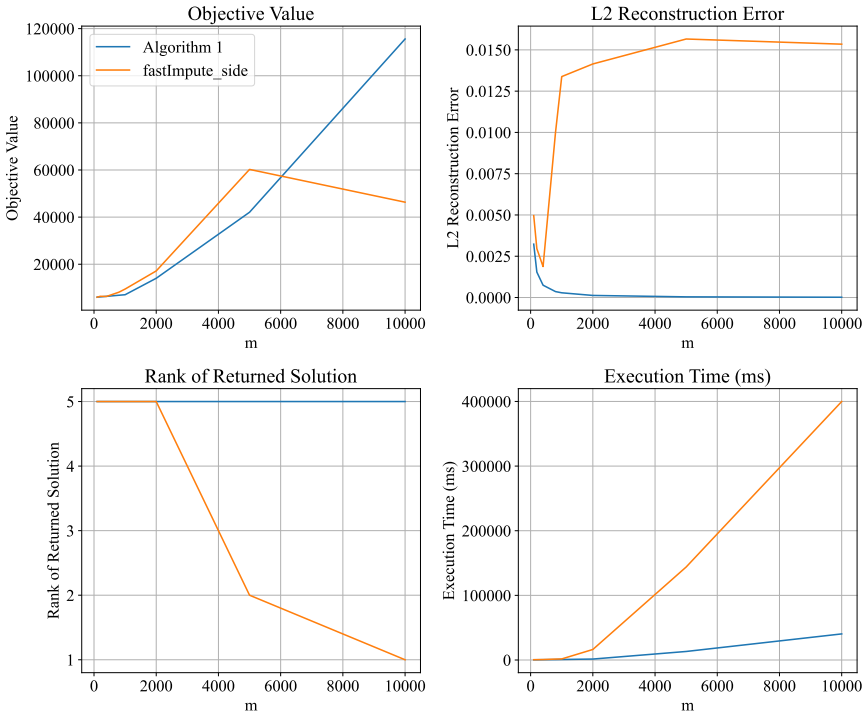

Fig. 6 Objective value (top left), ℓ_2 reconstruction error (top right), fitted rank (bottom left) and execution time (bottom right) versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

and execution time for ScaledGD, Algorithm 1, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD in Tables A8, A9, A10 and A11 of Appendix A. In Figure 10, we plot the average cumulative time spent solving subproblems (12), (14), (16), (18) during the execution of Algorithm 1 versus d. Our main findings from this set of experiments are:

1. Just as in Sections 6.2 and 6.3, Algorithm 1 and Fast-Impute-Side systematically produce higher quality solutions than ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD (see Table A8). ScaledGD produces the weakest average objective value across these experiments. The objective value achieved by each method appears to increase linearly as the dimension d of the side information increases. There is no significant difference between the objective value achieved by the output of Algorithm 1 compared to that of Fast-Impute-Side (Algorithm 1 is on average 2% lower).
2. In terms of ℓ_2 reconstruction error, Algorithm 1 and Fast-Impute-Side produce solutions that are of higher quality than ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD (see Table A9), often achieving an error that is an order of magnitude superior than the other methods. Relative to Fast-Impute-Side, Algorithm 1 outputs a solution whose ℓ_2 reconstruction error is on average 23% lesser across

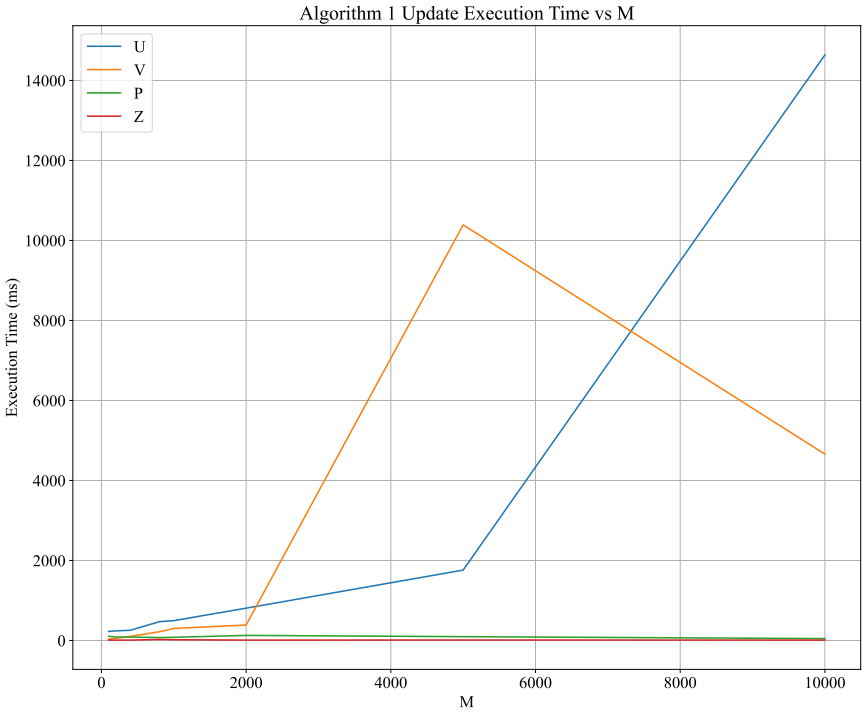

Fig. 7 Cumulative time spent solving each subproblem of Algorithm 1 versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

these trials. The performance of Algorithm 1 and Fast-Impute-Side improves as d increases, consistent with the intuition that recovering the partially observed matrix $\boldsymbol{A}$ becomes easier as more side information becomes available.

3. Algorithm 1 and Fast-Impute-Side always produced solutions that achieved a strictly greater R^2 value when used as a predictor for the side information compared to the other methods. The R^2 achieved by each method is roughly constant as the value of d increases. There is no significant difference between the R^2 achieved by the output of Algorithm 1 compared to that of Fast-Impute-Side.
4. The runtime of Algorithm 1 is competitive with that of the other methods. The runtime of Algorithm 1 is less than of Soft-Impute and Iterative-SVD but greater than that of Fast-Impute. Table A11 illustrates that ScaledGD was the fastest performing method, however its solutions were of the lowest quality. The runtimes of Algorithm 1, Fast-Impute-Side and ScaledGD grow with d while Fast-Impute, Soft-Impute and iterate SVD are constant with d which should be expected as these methods do not act on the side information matrix $\boldsymbol{Y}$. The runtime of Fast-Impute-Side exhibits particularly poor scaling behavior relative to the other methods as d increases.
5. Figure 10 illustrates that the computation of the solution for (16) is the computational bottleneck in the execution of Algorithm 1 in this set of experiments, followed

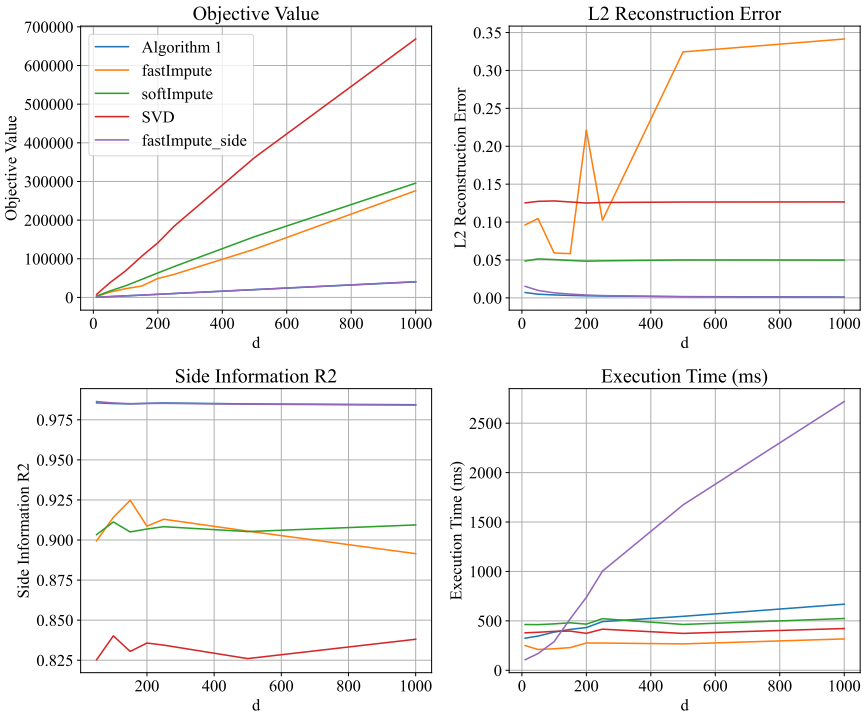

Fig. 8 Objective value (top left), ℓ_2 reconstruction error (top right), side information R^2 (bottom left) and execution time (bottom right) versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

next by the computation of the solution to (12). The solution times for (12), (14) and (18) appear constant as a function of d. This is consistent with the complexity analysis from Section 5 which found that the solve time for (16) is linear in d while the solve time for the 3 other subproblems are independent of d.

6.5 Sensitivity to Target Rank

We present a comparison of Algorithm 1 with ScaledGD, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD as we vary the rank of the underlying matrix k. In these experiments, we fixed $n = 1000, m = 100$ and $d = 150$ across all trials. We varied $k \in \{5, 10, 15, 20, 25, 30, 35, 40\}$ and we performed 20 trials for each value of d.

We report the objective value, ℓ_2 reconstruction error, fitted rank and execution time for Algorithm 1, Fast-Impute, Fast-Impute-Side, Soft-Impute and Iterative-SVD in Figure 11. For ease of comparison between Algorithm 1 and Fast-Impute-Side, we plot only the performance of these two methods in Figure 12. We additionally report the objective value, reconstruction error and execution time for ScaledGD, Algorithm 1, Fast-Impute, Fast-Impute-Side Soft-Impute and Iterative-SVD in Tables A12, A13 and A14 of Appendix A. In Figure 13, we plot the average cumulative time spent

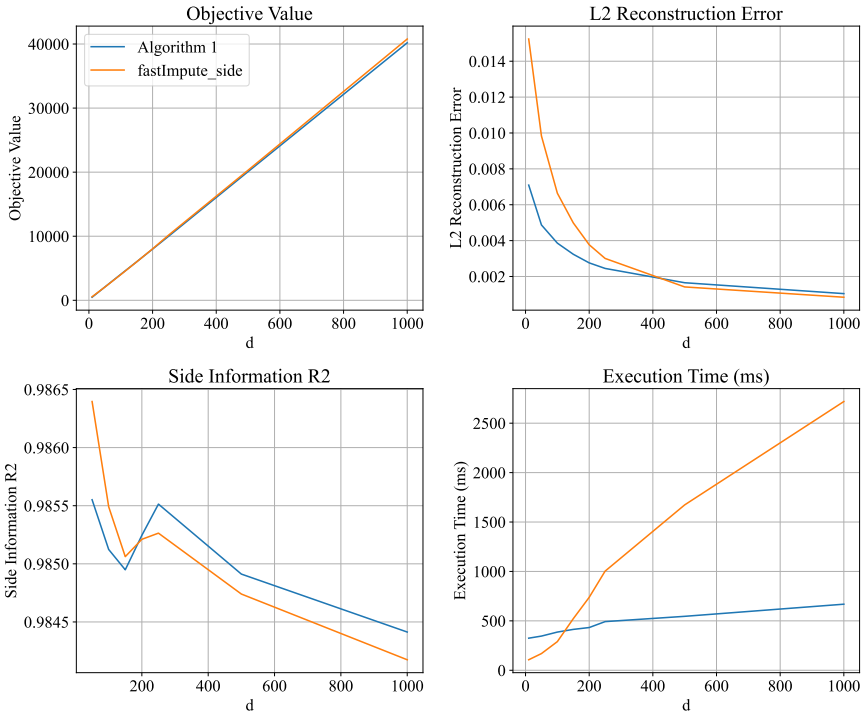

Fig. 9 Objective value (top left), ℓ_2 reconstruction error (top right), side information R^2 (bottom left) and execution time (bottom right) versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

solving subproblems (12), (14), (16), (18) during the execution of Algorithm 1 versus k. Our main findings from this set of experiments are as follows:

1. Unlike in Sections 6.2, 6.3 and 6.4, Algorithm 1 underperformed Fast-Impute-Side in terms of objective value (see Table A12). Fast-Impute-Side was the best performing method in 7 configurations and Soft-Impute was best in the remaining configuration. ScaledGD produces the weakest average objective value across these experiments.
2. In terms of ℓ_2 reconstruction error, Algorithm 1 produced higher quality solutions than all benchmark methods in 2 out of 8 of the tested parameter configurations where $k \leq 10$ (see Table A13). Fast-Impute-Side produced solutions achieving the lowest error in the other 6 parameter configurations.
3. The fitted rank of the solutions returned by Algorithm 1, ScaledGD, Fast-Impute and Fast-Impute-Side always matched the specified target rank, but the solutions returned by Soft-Impute and Iterative-SVD were always of full rank despite the fact that these methods were provided with the target rank explicitly.
4. The runtime of Algorithm 1 is competitive with that of the other methods. Table A14 illustrates that ScaledGD was the fastest performing method, however its

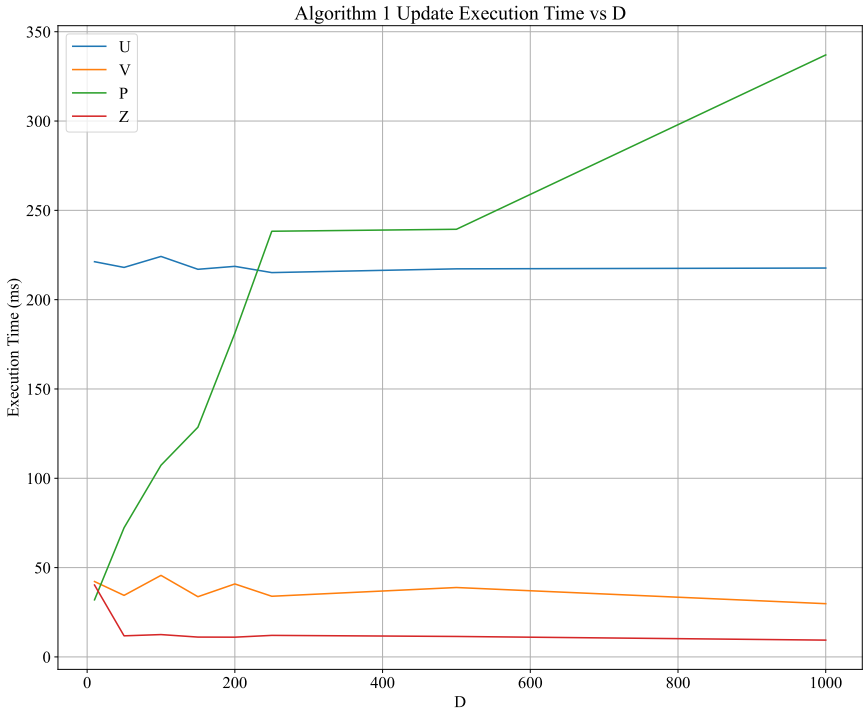

Fig. 10 Cumulative time spent solving each subproblem of Algorithm 1 versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

solutions were of the lowest quality. The runtime of Algorithm 1 is most competitive with Fast-Impute-Side, Soft-Impute and Iterative-SVD for small values of k.

5. Figure 13 illustrates that the computation of the solution for (12) is the computational bottleneck in the execution of Algorithm 1 in this set of experiments, followed next by the computation of the solution to (14) and (18).

6.6 Real World Data Experiments

We seek to answer the following question: how does the performance of Algorithm 1 compare to Fast-Impute and Fast-Impute-Side on real world data? We consider the Netflix Prize Dataset augmented with features from the TMDB Database as side information.

The Netflix Prize Dataset consists of greater than 10 million user ratings of movies spread across more than 450000 users and 17000 movies. To prepare data for our experiment, we first pull the following numerical features from the TMDB database:

1. Total Budget;
2. Revenue;
3. Popularity;
4. Average Vote;

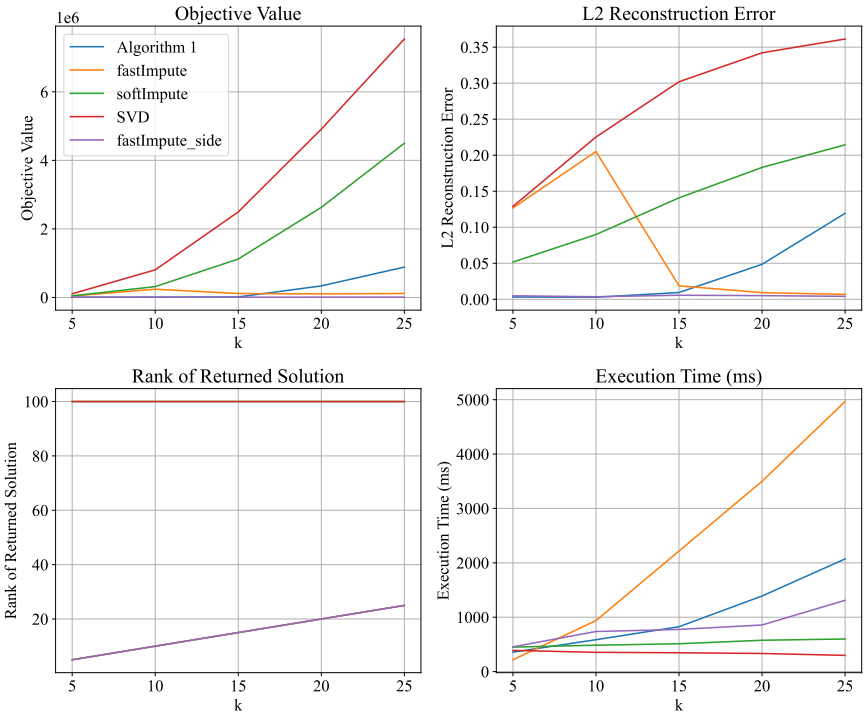

Fig. 11 Objective value (top left), ℓ_2 reconstruction error (top right), fitted rank (bottom left) and execution time (bottom right) versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

5. Vote Count;
6. Total Runtime.

Here, the matrix $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ from (1) corresponds to the matrix of movie-user ratings, $\Omega \subseteq [n] \times [m]$ corresponds to indices of $\boldsymbol{X}$ for which the user ratings of movies are observed and $\boldsymbol{Y} \in \mathbb{R}^{n \times d}$ corresponds to the TMDB Database features. Note that many movies did not have all 6 features available from TMDB. We constructed two datasets for our experimentation. In Dataset 1, we restricted the dataset to movies that had all 6 features present ($d = 6$) and to users who had given at least 5 ratings across those movies. After performing this filtering, we were left with $n = 3430$ movies and $m = 467364$ users. In Dataset 2, we considered the 4 most frequent features (popularity, average vote, vote count, total runtime) and restricted the dataset to movies that had all 4 of these features present ($d = 4$) and to users who had given at least 5 ratings to any of these movies. After performing this filtering, we were left with $n = 10574$ movies and $m = 470706$ users. For each dataset, we conducted experiments for values of the target rank k in the set $k \in \{3, 4, 5, 6, 7, 8, 9, 10\}$. For each value of k, we conducted 5 trials where a given trial consisted of randomly withholding 20% of the data as test data, estimating a low rank matrix on the 80% training data and evaluating the out of sample ℓ_2 reconstruction error on the withheld data.

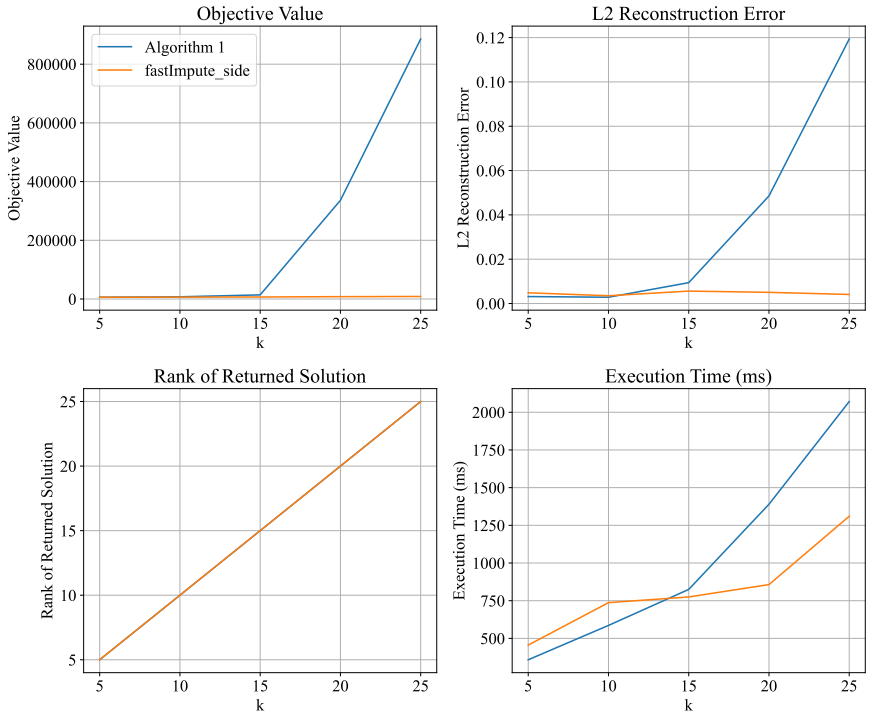

Fig. 12 Objective value (top left), ℓ_2 reconstruction error (top right), fitted rank (bottom left) and execution time (bottom right) versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

We report the in sample ℓ_2 reconstruction error, out of sample ℓ_2 reconstruction error and execution time for Algorithm 1, Fast-Impute and Fast-Impute-Side in addition to the average cumulative time spent solving subproblems (12), (14), (16), (18) during the execution of Algorithm 1 versus k on Dataset 1 and Dataset 2 in Figures 14 and 15 respectively. We additionally report the in sample ℓ_2 reconstruction error, out of sample ℓ_2 reconstruction error and execution time for Algorithm 1, Fast-Impute and Fast-Impute-Side on Dataset 1 and Dataset 2 in Tables A15 and A16 of Appendix A respectively. We report only results for Fast-Impute and Fast-Impute-Side as benchmarks because Soft-Impute, Iterative-SVD and ScaledGD failed to terminate after a 20 hour time limit across all experiments involving Dataset 1 and Dataset 2. Fast-Impute failed to terminate after a 20 hour time limit across all experiments involving Dataset 2 and across experiments involving Dataset 1 for which the target rank was greater than 6. Note that Fast-Impute and Fast-Impute-Side were the best performing benchmark methods across the synthetic data experiments so they consist of a reasonable method to compare against. In Figure 16, we report the coefficient of determination (R^2) achieved by Algorithm 1 on the side information both overall and on individual features in Dataset 1 and Dataset 2. Our main findings from this set of experiments are as follows:

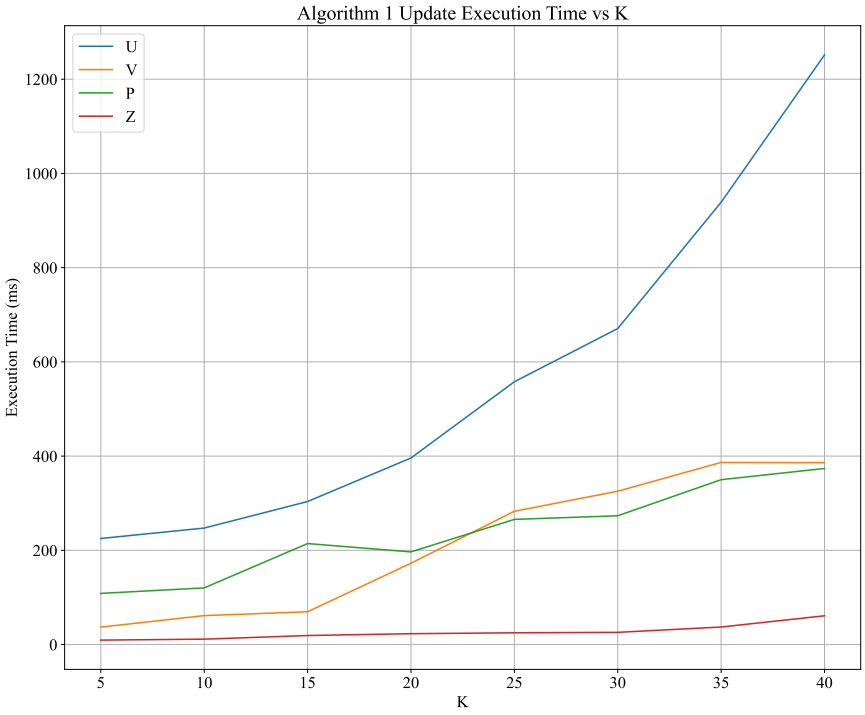

Fig. 13 Cumulative time spent solving each subproblem of Algorithm 1 versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

1. Fast-Impute in general produced solutions that achieved slightly lower in sample error but significantly higher out of sample error than the solutions produced by Algorithm 1. Out of sample error is a much more important metric than in sample error as out of sample error captures the ability of a candidate solution to generalize to unseen data. Fast-Impute-Side produced solutions that had both higher in sample error and out of sample error than the solutions produced by Algorithm 1. Across the experiments in which Fast-Impute terminated within the specified time limit, Algorithm 1 produced solutions that on average achieved 67% lower out of sample error than Fast-Impute. The high out of sample error (relative to in sample error) of Fast-Impute and Fast-Impute-Side suggests that these methods are likely over-fitting the training data. As is expected, in sample error decreased as the rank of the reconstruction increased. In the case of Algorithm 1, out of sample error increased as the reconstruction rank increased, suggesting that Algorithm 1 was over-fitting the data as rank increased.
2. Algorithm 1 exhibited significantly superior scalability than both Fast-Impute and Fast-Impute-Side. Across the experiments in which Fast-Impute terminated within the specified time limit, Algorithm 1 required on average 97% less time to execute than Fast-Impute (the faster of the methods between Fast-Impute and Fast-Impute-Side). The execution time of Algorithm 1 on the largest tested instance (Dataset

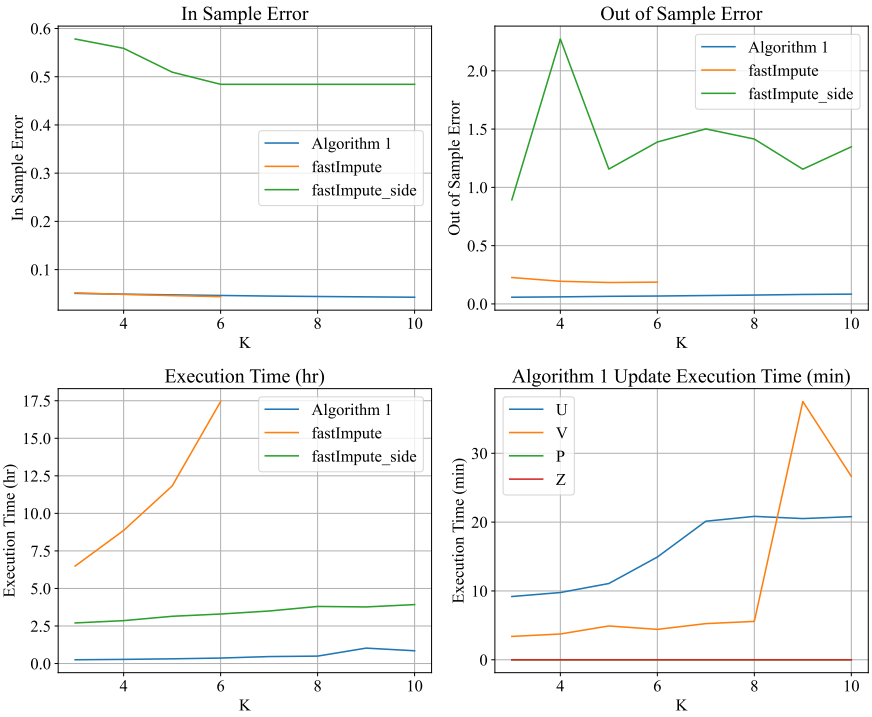

Fig. 14 In sample ℓ_2 reconstruction error (top left), out of sample ℓ_2 reconstruction error (top right), execution time (bottom left) and subproblem execution time (bottom right) versus k on Netflix Prize Dataset 1. Averaged over 5 trials.

$2, k = 10$) was less than the execution time of Fast-Impute on the smallest tested instance (Dataset $1, k = 3$).

3. The computational bottleneck of Algorithm 1 is the solution time of subproblems (12) and (14). Solving these two subproblems requires an order of magnitude more time than solving subproblems (16) and (18)
4. Figure 16 illustrates that the reconstructed matrix produced by Algorithm 1 becomes a better predictor of the side information as the value of k increases. This is to be expected as increasing k increases model complexity. We see that popularity and runtime are the most difficult features to predict as a linear function of the reconstructed matrix while vote average and budget are the easiest features to predict.

6.7 Summary of Findings

We now summarize our findings from our numerical experiments. In Sections 6.2-6.5, we see that across all experiments using synthetic data and target rank $k \leq 10$, Algorithm 1 produces solutions that achieve on average 2.3% lower objective value and 41% lower ℓ_2 reconstruction error than the solutions returned by the best performing benchmark method (usually Fast-Impute-Side). In the regime where $k > 10$,

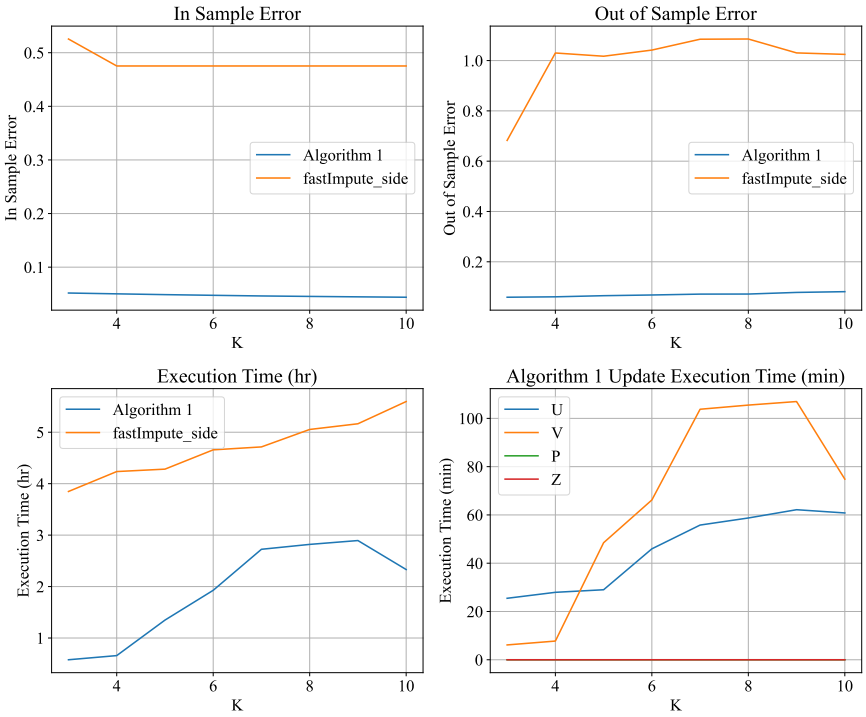

Fig. 15 In sample ℓ_2 reconstruction error (top left), out of sample ℓ_2 reconstruction error (top right), execution time (bottom left) and subproblem execution time (bottom right) versus k on Netflix Prize Dataset 2. Averaged over 5 trials.

we see in Section 6.5 that Fast-Impute-Side outperforms Algorithm 1. We see that the execution time of Algorithm 1 is competitive with and often notably faster than the benchmark methods on synthetic data. Our computational results are consistent with the complexity analysis performed in Section 5 for Problems (12), (14), (16) and (18). We observe that solution time for (16) becomes the bottleneck as the target rank k scales, otherwise the solution time for (12) is the bottleneck. On real world data comprised of the Netflix Prize Dataset augmented with features from the TMDB Database, Algorithm 1 produces solutions that achieve 67% lower out of sample error than Fast-Impute, the best performing benchmark, in 97% less execution time.

7 Conclusion

In this paper, we introduced Problem (1) which seeks to reconstruct a partially observed matrix that is predictive of fully observed side information. We illustrate that (1) has a natural interpretation as a robust optimization problem and can be reformulated as a mixed-projection optimization problem. We derive a semidefinite cone relaxation (9) to (1) and we present Algorithm 1, a mixed-projection alternating direction method of multipliers algorithm that obtains scalable, high quality solutions

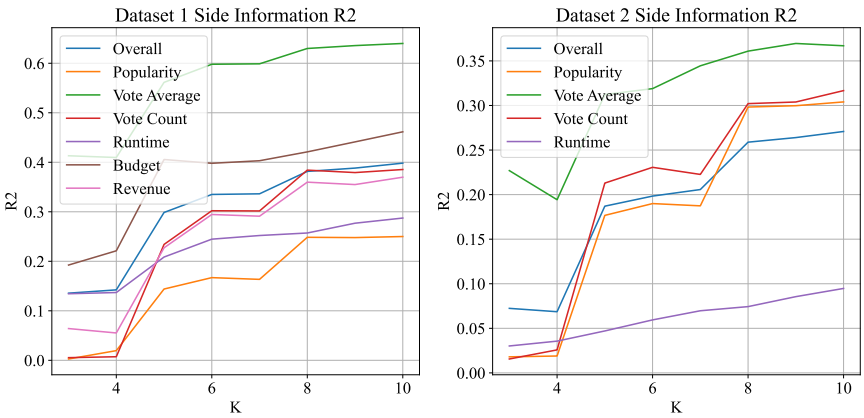

Fig. 16 Algorithm 1 side information R^2 on Netflix Prize Dataset 1 (left) and Dataset 2 (right) versus k. Averaged over 5 trials.

to (1). We rigorously benchmark the performance of Algorithm 1 on synthetic and real world data against benchmark methods Fast-Impute, Fast-Impute-Side, Soft-Impute, Iterative-SVD and ScaledGD. We find that across all synthetic data experiments with $k \leq 10$, Algorithm 1 outputs solutions that achieve on average 2.3% lower objective value in (1) and 41% lower ℓ_2 reconstruction error than the solutions returned by the best performing benchmark method. For the 5 synthetic data experiments with $k > 15$, Fast-Impute returns superior quality solutions than Algorithm 1, however the former takes on average 3 times as long as Algorithm 1 to execute. The runtime of Algorithm 1 is competitive with and often superior to that of the benchmark methods. Algorithm 1 is able to solve problems with $n = 10000$ rows and $m = 10000$ columns in less than a minute. On real world data from the Netflix Prize competition, Algorithm 1 produces solutions that achieve 67% lower out of sample error than benchmark methods in 97% less execution time.Future work could expand the mixed-projection ADMM framework introduced in this work to incorporate positive semidefinite constraints and general linear constraints. Additionally, future work could empirically investigate the strength of the semidefinite relaxation (9) and could explore how to leverage this lower bound to certify globally optimal solutions.

Declarations

Funding:

The authors did not receive support from any organization for the submitted work.

Conflict of interest/Competing interests:

The authors have no relevant financial or non-financial interests to disclose.

Ethics approval:

Not applicable.

Consent to participate:

Not applicable.

Consent for publication:

Not applicable.

Availability of data and materials:

The data employed in Section 6.6 was taken from the open source Netflix Prize competitions and the TMDB database.

Code availability:

To bridge the gap between theory and practice, we have made our code freely available on GitHub at github.com/NicholasJohnson2020/LearningLowRankMatrices

Authors' contributions:

Both authors contributed to the problem formulation and algorithmic ideation. Algorithm implementation, data collection, simulation and data analysis was performed by Nicholas Johnson. The first draft of the manuscript was written by Nicholas Johnson and both authors commented and edited subsequent versions of the manuscript. Both authors read and approved the final manuscript.

Appendix A Supplemental Computational Results

Table A1 Comparison of the objective value of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Objective					
N	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
100	249263	660	4890	**638**	19790	28678
200	306739	1283	9448	**1217**	23719	44055
400	417643	**2457**	13910	2549	30685	61113
800	421032	4844	28784	**4805**	40015	93119
1000	522586	6046	42010	**6002**	46788	107851
2000	563033	11975	59636	**11944**	76246	167459
5000	1226490	30040	225143	**29797**	170248	364066
10000	1973666	60083	644293	**59411**	317106	642760

Table A2 Comparison of the reconstruction error of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	ℓ_2 Reconstruction Error					
N	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
100	100.225	**0.015**	0.072	0.021	0.212	0.301
200	58.372	**0.007**	0.121	0.007	0.119	0.210
400	33.925	**0.004**	0.062	0.005	0.074	0.160
800	14.979	**0.003**	0.089	0.005	0.052	0.130
1000	12.665	**0.003**	0.236	0.005	0.049	0.126
2000	5.544	**0.003**	0.246	0.005	0.044	0.117
5000	2.473	**0.003**	0.078	0.005	0.037	0.105
10000	1.321	**0.003**	0.080	0.005	0.035	0.102

Table A3 Comparison of the side information R^2 of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Side Information R^2					
N	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
100	0.157	0.983	0.868	-	**1.000**	1.000
200	0.193	0.984	0.878	**0.985**	0.837	0.740
400	0.167	**0.985**	0.912	0.984	0.866	0.780
800	0.441	0.985	0.911	**0.985**	0.905	0.826
1000	0.328	0.985	0.896	**0.985**	0.906	0.829
2000	0.557	0.985	0.924	**0.985**	0.920	0.862
5000	0.525	0.985	0.889	**0.986**	0.928	0.879
10000	0.582	0.985	0.840	**0.985**	0.932	0.888

Table A4 Comparison of the execution time of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Execution Time (ms)					
N	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
100	**11.05**	76.74	100.68	207.84	120.84	115.21
200	**41.95**	125.53	116.53	256.05	169.37	169.00
400	**56.63**	169.16	159.47	327.89	251.79	254.37
800	**73.11**	311.63	204.53	425.74	402.47	323.79
1000	**48.05**	413.84	212.63	427.95	465.32	380.89
2000	**134.74**	611.63	288.95	511.84	775.95	600.05
5000	822.79	1413.00	**503.00**	706.37	1840.00	1318.32
10000	19707.21	2275.00	**881.58**	1001.53	4016.89	2810.84

Table A5 Comparison of the objective value of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Objective					
M	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
100	530097	6044	37924	**5997**	46917	103403
200	2483914	**6134**	12019	6388	28371	114448
400	14226535	6368	14060	**6329**	22009	90652
800	99356630	**6839**	22086	8133	38745	87895
1000	105452002	**7060**	23888	9470	45968	128500
2000	591164397	**14041**	43504	17158	96007	815807
5000	4002088274	**42069**	100878	60243	310417	11294105
10000	9826250213	115589	196625	**46335**	943496	60913874

Table A6 Comparison of the reconstruction error of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	ℓ_2 Reconstruction Error					
M	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
100	13.68740	**0.00323**	0.18290	0.00500	0.05010	0.12560
200	40.58900	**0.00154**	0.00500	0.00300	0.01150	0.06640
400	127.71450	**0.00075**	0.00340	0.00190	0.00340	0.02240
800	508.24550	**0.00036**	0.00340	0.01000	0.00310	0.00460
1000	443.26630	**0.00029**	0.00300	0.01340	0.00300	0.00350
2000	1292.61610	**0.00013**	0.00310	0.01410	0.00290	0.00300
5000	3658.18840	**0.00004**	0.00320	0.01570	0.00270	0.00540
10000	4559.27300	**0.00002**	0.00330	0.01530	0.00270	0.00650

Table A7 Comparison of the execution time of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Execution Time (ms)					
M	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
100	**45.26**	371.16	219.84	450.11	451.95	375.37
200	**56.79**	398.68	217.47	563.11	971.16	764.37
400	**79.89**	531.11	288.74	698.42	1503.68	1594.00
800	**131.16**	662.68	385.95	1402.89	2386.58	2721.89
1000	**159.79**	817.53	541.84	1651.00	2781.74	3140.95
2000	4698.95	**1459.00**	5125.47	16279.21	5282.89	6297.79
5000	29291.74	**13195.95**	40036.79	143865.75	32777.05	35080.95
10000	108192.58	**40368.16**	158083.84	399863.00	82190.32	89836.58

Table A8 Comparison of the objective value of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

	Objective					
D	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
10	11692	**478**	2648	553	3368	7710
50	96771	2070	13402	**2058**	16273	37204
100	229740	4051	22644	**4008**	30116	68634
150	532018	6010	29433	**5996**	46596	106504
200	734648	**7994**	49004	8029	63243	141252
250	1195066	**9984**	59470	10160	79625	183720
500	4165783	**20081**	124737	20302	157096	361741
1000	13578263	**40188**	276053	40781	295635	668550

Table A9 Comparison of the reconstruction error of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

	ℓ_2 Reconstruction Error					
D	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
10	0.60780	**0.00709**	0.09620	0.01520	0.04870	0.12530
50	1.41600	**0.00487**	0.10460	0.00980	0.05140	0.12730
100	5.03590	**0.00386**	0.05920	0.00660	0.05050	0.12790
150	14.00330	**0.00324**	0.05830	0.00500	0.04940	0.12650
200	22.26870	**0.00276**	0.22120	0.00380	0.04840	0.12510
250	39.41630	**0.00245**	0.10240	0.00300	0.04890	0.12580
500	177.68800	0.00170	0.32450	**0.00142**	0.05000	0.12640
1000	679.11770	0.00100	0.34150	**0.00084**	0.04980	0.12660

Table A10 Comparison of the side information R^2 of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

	Side Information R^2					
D	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
10	0.91180	0.98800	0.90730	**0.99199**	0.90330	0.82120
50	0.52800	0.98560	0.89950	**0.98640**	0.90340	0.82530
100	0.53300	0.98510	0.91420	**0.98549**	0.91130	0.84020
150	0.30270	0.98490	0.92480	**0.98506**	0.90500	0.83050
200	0.41070	**0.98525**	0.90860	0.98520	0.90690	0.83570
250	0.29080	**0.98551**	0.91300	0.98530	0.90830	0.83440
500	0.15090	**0.98491**	0.90560	0.98470	0.90530	0.82600
1000	0.29250	**0.98441**	0.89150	0.98420	0.90940	0.83810

Table A11 Comparison of the execution time of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

	Execution Time (ms)					
D	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
10	**81.73684**	324.73680	250.10530	106.31580	462.84210	379.00000
50	**91.47368**	345.84210	210.21050	168.89470	461.89470	384.42110
100	**91.89474**	386.47370	217.57890	289.89470	469.42110	393.78950
150	**98.73684**	413.57890	229.36840	521.73680	480.36840	396.36840
200	**99.00000**	432.36840	275.94740	738.00000	466.73680	373.57890
250	**151.73684**	492.31580	275.52630	1002.94740	522.05260	415.78950
500	**181.15789**	545.78950	266.78950	1673.31580	463.47370	373.00000
1000	**148.26316**	668.57890	317.10530	2719.10530	523.63160	422.42110

Table A12 Comparison of the objective value of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Objective					
D	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
5	514330	6040	40047	**6002**	46511	106391
10	1892279	7502	240910	**6315**	316063	805397
15	5213393	14022	113808	**6870**	1121725	2495972
20	10196280	336294	103289	**7952**	2629190	4910387
25	16816443	885813	115398	**8601**	4499282	7541301
30	27397868	65551564	128971	**8824**	6863908	10634437
35	39536651	160010082	144816	**8832**	9407573	14192828
40	-	320532712	114058512	19625866	**12538955**	18290215

Table A13 Comparison of the reconstruction error of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	ℓ_2 Reconstruction Error					
D	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
5	12.95500	**0.00314**	0.12670	0.00480	0.05160	0.12900
10	5.41720	**0.00284**	0.20510	0.00350	0.08990	0.22520
15	3.45210	0.00940	0.01870	**0.00561**	0.14080	0.30200
20	2.34590	0.04860	0.00920	**0.00505**	0.18300	0.34220
25	1.73610	0.11940	0.00680	**0.00409**	0.21440	0.36130
30	1.40170	0.19840	0.00570	**0.00350**	0.23740	0.37240
35	1.16780	0.17700	0.00480	**0.00301**	0.25270	0.37740
40	-	0.21870	0.00440	**0.00267**	0.26520	0.38020

Table A14 Comparison of the execution time of ScaledGD, Algorithm 1, Fast-Impute, Soft-Impute and SVD versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Execution Time (ms)					
D	ScaledGD	Algorithm 1	Fast-Impute	Fast-Impute-Side	Soft-Impute	SVD
5	**65**	358	217	455	450	389
10	**85**	586	938	737	485	354
15	**112**	825	2217	775	512	346
20	**110**	1390	3499	857	576	333
25	**118**	2071	4966	1310	599	298
30	**108**	2425	6381	1272	625	275
35	**120**	2653	8635	1508	641	280
40	4532	3092	11811	1797	689	**270**

Table A15 Comparison of the in sample ℓ_2 reconstruction error, out of sample ℓ_2 reconstruction error and execution time of Algorithm 1 and Fast-Impute versus k on Netflix Prize Dataset 1. Averaged over 5 trials.

	In Sample Error			Out of Sample Error			Execution Time (hr)		
K	Alg 1	FI	FIS	Alg 1	FI	FIS	Alg 1	FI	FIS
3	**0.0507**	0.0516	0.5782	**0.0573**	0.2264	0.8929	**0.2512**	6.4907	2.7021
4	0.0490	**0.0486**	0.5588	**0.0604**	0.1942	2.2728	**0.2760**	8.8641	2.8570
5	0.0476	**0.0460**	0.5095	**0.0651**	0.1835	1.1568	**0.3136**	11.8207	3.1463
6	0.0463	**0.0438**	0.4842	**0.0676**	0.1867	1.3890	**0.3654**	17.4471	3.2974
7	**0.0451**	-	0.4842	**0.0718**	-	1.5014	**0.4637**	-	3.4976
8	**0.0442**	-	0.4842	**0.0759**	-	1.4145	**0.4941**	-	3.8013
9	**0.0434**	-	0.4842	**0.0811**	-	1.1559	**1.0262**	-	3.7692
10	**0.0427**	-	0.4842	**0.0839**	-	1.3480	**0.8503**	-	3.9254

Table A16 In sample ℓ_2 reconstruction error, out of sample ℓ_2 reconstruction error and execution time of Algorithm 1 versus k on Netflix Prize Dataset 2. Averaged over 5 trials.

	In Sample Error		Out of Sample Error		Execution Time (hr)	
K	Algorithm 1	FIS	Algorithm 1	FIS	Algorithm 1	FIS
3	**0.0518**	0.5255	**0.0591**	0.6825	**0.5773**	3.8481
4	**0.0502**	0.4753	**0.0607**	1.0302	**0.6581**	4.2354
5	**0.0488**	0.4753	**0.0652**	1.0172	**1.3482**	4.2825
6	**0.0475**	0.4753	**0.0680**	1.0417	**1.9273**	4.6575
7	**0.0463**	0.4753	**0.0715**	1.0849	**2.7246**	4.7145
8	**0.0454**	0.4753	**0.0718**	1.0857	**2.8198**	5.0538
9	**0.0446**	0.4753	**0.0782**	1.0304	**2.8945**	5.1645
10	**0.0439**	0.4753	**0.0812**	1.0246	**2.3311**	5.5966

Table A17 Algorithm 1 side information R^2 on Netflix Prize Dataset 1. Averaged over 5 trials.

K	Overall	Popularity	Vote Average	Vote Count	Runtime	Budget	Revenue
3	0.136	0.003	0.413	0.005	0.134	0.192	0.064
4	0.142	0.02	0.41	0.007	0.137	0.221	0.055
5	0.299	0.144	0.561	0.234	0.209	0.406	0.227
6	0.335	0.167	0.598	0.302	0.245	0.398	0.295
7	0.336	0.163	0.599	0.302	0.252	0.403	0.291
8	0.382	0.248	0.63	0.384	0.257	0.421	0.36
9	0.388	0.248	0.636	0.379	0.277	0.441	0.355
10	0.398	0.25	0.64	0.385	0.287	0.462	0.37

Table A18 Algorithm 1 side information R^2 on Netflix Prize Dataset 2. Averaged over 5 trials.

K	Overall	Popularity	Vote Average	Vote Count	Runtime
3	0.072	0.018	0.227	0.016	0.03
4	0.069	0.019	0.194	0.026	0.036
5	0.187	0.177	0.312	0.213	0.047
6	0.198	0.19	0.319	0.231	0.059
7	0.206	0.187	0.344	0.223	0.07
8	0.259	0.298	0.361	0.302	0.074
9	0.264	0.3	0.37	0.304	0.086
10	0.271	0.304	0.367	0.317	0.095

Declarations

Funding:

The authors did not receive support from any organization for the submitted work.

Conflict of interest/Competing interests:

The authors have no relevant financial or non-financial interests to disclose.

Ethics approval:

Not applicable.

Consent to participate:

Not applicable.

Consent for publication:

Not applicable.

Availability of data and materials:

The data used in Section 6.6 was taken from the Netflix Prize Dataset and the TMDB Database, both of which are freely available online.

Code availability:

To bridge the gap between theory and practice, we have made our code freely available on GitHub at https://github.com/NicholasJohnson2020/LearningLowRankMatrices

Authors' contributions:

Both authors contributed to formulating the optimization problem. Algorithmic design and implementation along with data collection, simulation and data analysis was performed by Nicholas Johnson. The first draft of the manuscript was written by Nicholas Johnson and both authors commented and edited subsequent versions of the manuscript. Both authors read and approved the final manuscript.

References

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

[2] Liu, Q.: Power network system identification and recovery based on the matrix completion. In: Journal of Physics: Conference Series, vol. 1237, pp. 32–59 (2019). IOP Publishing

[3] Ji, H., Liu, C., Shen, Z., Xu, Y.: Robust video denoising using low rank matrix completion. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1791–1798 (2010). IEEE

[4] Ramlatchan, A., Yang, M., Liu, Q., Li, M., Wang, J., Li, Y.: A survey of matrix completion methods for recommendation systems. Big Data Mining and Analytics **1**(4), 308–323 (2018)

[5] Bell, R.M., Koren, Y.: Lessons from the netflix prize challenge. Acm Sigkdd Explorations Newsletter **9**(2), 75–79 (2007)

[6] Udell, M., Townsend, A.: Why are big data matrices approximately low rank? SIAM Journal on Mathematics of Data Science **1**(1), 144–160 (2019)

[7] Xu, M., Jin, R., Zhou, Z.-H.: Speedup matrix completion with side information: Application to multi-label learning. Advances in neural information processing systems **26** (2013)

[8] Chiang, K.-Y., Hsieh, C.-J., Dhillon, I.S.: Matrix completion with noisy side information. Advances in neural information processing systems **28** (2015)

[9] Bertsimas, D., Li, M.L.: Fast exact matrix completion: A unified optimization framework for matrix completion. Journal of Machine Learning Research **21**(231), 1–43 (2020)

[10] Bach, F.R., Jordan, M.I.: Predictive low-rank decomposition for kernel methods. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 33–40 (2005)

[11] Farhat, M.R., Shapiro, B.J., Kieser, K.J., Sultana, R., Jacobson, K.R., Victor, T.C., Warren, R.M., Streicher, E.M., Calver, A., Sloutsky, A., *et al.*: Genomic analysis identifies targets of convergent positive selection in drug-resistant mycobacterium tuberculosis. Nature genetics **45**(10), 1183–1189 (2013)

[12] Natarajan, N., Dhillon, I.S.: Inductive matrix completion for predicting gene–disease associations. Bioinformatics **30**(12), 60–68 (2014)

[13] Elmahdy, A., Ahn, J., Suh, C., Mohajer, S.: Matrix completion with hierarchical graph side information. Advances in neural information processing systems **33**,

9061–9074 (2020)

[14] Suh, G., Jeon, S., Suh, C.: When to use graph side information in matrix completion. In: 2021 IEEE International Symposium on Information Theory (ISIT), pp. 2113–2118 (2021). IEEE

[15] Owen, A.B., Perry, P.O.: Bi-cross-validation of the SVD and the nonnegative matrix factorization. The Annals of Applied Statistics **3**(2), 564–594 (2009)

[16] Bousquet, O., Elisseeff, A.: Stability and generalization. Journal of Machine Learning Research **2**, 499–526 (2002)

[17] Bertsimas, D., Cory-Wright, R., Pauphilet, J.: Mixed-projection conic optimization: A new paradigm for modeling rank constraints. Operations Research **70**(6), 3321–3344 (2022)

[18] Nguyen, L.T., Kim, J., Shim, B.: Low-rank matrix completion: A contemporary survey. IEEE Access **7**, 94215–94237 (2019)

[19] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the royal statistical society: series B (methodological) **39**(1), 1–22 (1977)

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

[21] Billsus, D., Pazzani, M.J., *et al.*: Learning collaborative information filters. In: Icml, vol. 98, pp. 46–54 (1998)

[22] Sarwar, B., Karypis, G., Konstan, J., Riedl, J.T.: Application of dimensionality reduction in recommender system-a case study. ACM WebKDD 2000 Workshop (2000)

[23] Mazumder, R., Hastie, T., Tibshirani, R.: Spectral regularization algorithms for learning large incomplete matrices. Journal of Machine Learning Research **11**(80), 2287–2322 (2010)

[24] Fazel, M.: Matrix rank minimization with applications. PhD thesis, PhD thesis, Stanford University (2002)

[25] Candès, E.J., Tao, T.: The power of convex relaxation: Near-optimal matrix completion. IEEE transactions on information theory **56**(5), 2053–2080 (2010)

[26] Candes, E., Recht, B.: Exact matrix completion via convex optimization. Communications of the ACM **55**(6), 111–119 (2012)

[27] Donoho, D.L., Johnstone, I.M., Kerkyacharian, G., Picard, D.: Wavelet shrinkage:

asymptopia? Journal of the Royal Statistical Society: Series B (Methodological) **57**(2), 301–337 (1995)

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

[29] Jain, P., Netrapalli, P.: Fast exact matrix completion with finite samples. In: Conference on Learning Theory, pp. 1007–1034 (2015). PMLR

[30] Zheng, Q., Lafferty, J.: Convergence analysis for rectangular matrix completion using burer-monteiro factorization and gradient descent. arXiv preprint arXiv:1605.07051 (2016)

[31] Jin, C., Kakade, S.M., Netrapalli, P.: Provable efficient online matrix completion via non-convex stochastic gradient descent. Advances in Neural Information Processing Systems **29** (2016)

[32] Chen, Y., Wainwright, M.J.: Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. arXiv preprint arXiv:1509.03025 (2015)

[33] Ge, R., Huang, F., Jin, C., Yuan, Y.: Escaping from saddle points—online stochastic gradient for tensor decomposition. In: Conference on Learning Theory, pp. 797–842 (2015). PMLR

[34] Sun, R., Luo, Z.-Q.: Guaranteed matrix completion via non-convex factorization. IEEE Transactions on Information Theory **62**(11), 6535–6579 (2016)

[35] Ma, C., Wang, K., Chi, Y., Chen, Y.: Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval and matrix completion. In: International Conference on Machine Learning, pp. 3345–3354 (2018). PMLR

[36] Nesterov, Y.: A method for solving the convex programming problem with convergence rate o (1/k2). In: Dokl Akad Nauk Sssr, vol. 269, p. 543 (1983)

[37] Tong, T., Ma, C., Chi, Y.: Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent. Journal of Machine Learning Research **22**(150), 1–63 (2021)

[38] Bertsimas, D., Cory-Wright, R., Pauphilet, J.: A new perspective on low-rank optimization. Mathematical Programming, articles in advance, 1–46 (2023)

[39] Boyd, S., Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge university press, USA (2004)

[40] Bertsimas, D., Cory-Wright, R., Johnson, N.A.G.: Sparse plus low rank matrix decomposition: A discrete optimization approach. Journal of Machine Learning

Research **24**(267), 1–51 (2023)

[41] Bertsimas, D., Cory-Wright, R., Lo, S., Pauphilet, J.: Optimal low-rank matrix completion: Semidefinite relaxations and eigenvector disjunctions. arXiv preprint arXiv:2305.12292 (2023)

[42] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., *et al.*: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning **3**(1), 1–122 (2011)

[43] Xu, Z., De, S., Figueiredo, M., Studer, C., Goldstein, T.: An empirical study of admm for nonconvex problems. arXiv preprint arXiv:1612.03349 (2016)

[44] Guo, K., Han, D., Wang, D.Z., Wu, T.: Convergence of admm for multi-block nonconvex separable optimization models. Frontiers of Mathematics in China **12**, 1139–1162 (2017)

[45] Wang, Y., Yin, W., Zeng, J.: Global convergence of admm in nonconvex nonsmooth optimization. Journal of Scientific Computing **78**, 29–63 (2019)

[46] Wang, J., Zhao, L.: Nonconvex generalization of alternating direction method of multipliers for nonlinear equality constrained problems. Results in Control and Optimization **2**, 100009 (2021)

[47] Xu, H., Caramanis, C., Mannor, S.: Robustness and regularization of support vector machines. Journal of Machine Learning Research **10**(7) (2009)

[48] Bertsimas, D., Hertog, D.: Robust and Adaptive Optimization. Dynamic Ideas LLC, ??? (2022)

[49] Bertsimas, D., Copenhaver, M.S.: Characterization of the equivalence of robustification and regularization in linear and matrix regression. European Journal of Operational Research **270**(3), 931–942 (2018)

[50] Günlük, O., Linderoth, J.: Perspective reformulation and applications. In: Mixed Integer Nonlinear Programming, pp. 61–89. Springer, ??? (2012)

[51] Boyd, S., El Ghaoui, L., Feron, E., Balakrishnan, V.: Linear Matrix Inequalities in System and Control Theory. SIAM, ??? (1994)

[52] Saxena, A., Bonami, P., Lee, J.: Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. Mathematical programming **124**(1), 383–411 (2010)

[53] Bertsimas, D., Pauphilet, J., Van Parys, B.: Sparse regression. Statistical Science **35**(4), 555–578 (2020)

[54] Bertsimas, D., Cory-Wright, R., Pauphilet, J.: A unified approach to mixed-integer optimization problems with logical constraints. SIAM Journal on Optimization **31**(3), 2340–2367 (2021)

[55] Bertsimas, D., Johnson, N.A.G.: Compressed sensing: A discrete optimization approach. Machine Learning, 1–40 (2024)

[56] Petersen, K.B., Pedersen, M.S., *et al.*: The matrix cookbook. Technical University of Denmark **7**(15), 510 (2008)

[57] Xu, Y., Yin, W., Wen, Z., Zhang, Y.: An alternating direction algorithm for matrix completion with nonnegative factors. Frontiers of Mathematics in China **7**, 365–384 (2012)

[58] Jiang, B., Ma, S., Zhang, S.: Alternating direction method of multipliers for real and complex polynomial optimization models. Optimization **63**(6), 883–898 (2014)

[59] Shen, Y., Wen, Z., Zhang, Y.: Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization. Optimization Methods and Software **29**(2), 239–263 (2014)

[60] Rubinsteyn, A., Feldman, S.: fancyimpute: An Imputation Library for Python (2016). https://github.com/iskandr/fancyimpute

[61] Reuther, A., Kepner, J., Byun, C., Samsi, S., Arcand, W., Bestor, D., Bergeron, B., Gadepally, V., Houle, M., Hubbell, M., Jones, M., Klein, A., Milechin, L., Mullen, J., Prout, A., Rosa, A., Yee, C., Michaleas, P.: Interactive supercomputing on 40,000 cores for machine learning and data analysis. In: 2018 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–6 (2018). IEEE