Efficient Algorithms on Robust Low-Rank Matrix Completion Against Outliers

Licheng Zhao, Prabhu Babu, and Daniel P. Palomar, *Fellow, IEEE*

***Abstract*—This paper considers robust low-rank matrix completion in the presence of outliers. The objective is to recover a low-rank data matrix from a small number of noisy observations. We exploit the bilinear factorization formulation and develop a novel algorithm fully utilizing parallel computing resources. Our main contributions are i) providing two smooth loss functions that promote robustness against two types of outliers, namely, dense outliers drawn from some elliptical distribution and sparse spike-like outliers with small additive Gaussian noise; and ii) an efficient algorithm with provable convergence to a stationary solution based on a parallel update scheme. Numerical results show that the proposed algorithm obtains a better solution with faster convergence speed than the benchmark algorithms in both synthetic and real data scenarios.**

***Index Terms*—Matrix completion, factorization formulation, parallel algorithm, robust loss functions.**

I. Introduction

PROCESSING high-dimensional and incomplete data matrices plays an important role in big-data system analytics. In real-life situations, datasets may contain numerous outliers because the data collection process contains noise and errors of different nature. A common problem faced in practice is the reconstruction of the dataset from just a few noisy observations. It is generally understood that in most engineering applications, the effective information of a high-dimensional data matrix lies in a low-dimensional subspace, which means the data matrix is low-rank in nature [2]. Take the Netflix problem as an example [3], [4]. The Netflix company wants to predict movie viewers' preferences and make recommendations for these customers based on an incomplete and inaccurate large dataset comprised of movie ratings from 1 to 5 done by other customers. It is known that people's preference for movies is related to only a few factors like the movie category, the starring cast, etc., which translates into a low-rank dataset. The low-rank characteristic lays a good foundation for the reconstruction task.

To formulate the matrix completion problem, we denote the original data matrix by $\mathbf{M} \in \mathbb{R}^{m \times n}$, and assume only those entries from the set $\Omega \subset \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$ are observed. Then, the data model for the observed matrix $\tilde{\mathbf{M}}$ is

$$\tilde{\mathbf{M}} = \boldsymbol{\Omega} \odot (\mathbf{M} + \mathbf{N}), \tag{1}$$

where $\boldsymbol{\Omega}_{ij} = \begin{cases} 1 & (i,j) \in \Omega \\ 0 & (i,j) \notin \Omega \end{cases}$, $\odot$ is the Hadamard product, and $\mathbf{N}$ stands for the noise matrix that models numerous outliers. Our objective is to find a low-rank matrix $\widehat{\mathbf{M}}$ to approximate $\tilde{\mathbf{M}}$, hoping to recover $\mathbf{M}$.

A. Related Work

In classical works, PCA[1] (Principal Component Analysis) [5]–[7] was proposed to recover $\mathbf{M}$. PCA finds a low-rank matrix that minimizes the squared estimation error to the given matrix subject to a rank upper bound l $(l < \min(m, n))$. This enables matrix factorization in $\widehat{\mathbf{M}}$ as $\widehat{\mathbf{M}} = \mathbf{X}^T \mathbf{Y}$, with $\mathbf{X} \in \mathbb{R}^{l \times m}$ and $\mathbf{Y} \in \mathbb{R}^{l \times n}$. The factorization technique is advantageous in that it not only removes the nonconvex rank constraint, but also reduces the complexity of data storage from $\mathcal{O}(mn)$ to $\mathcal{O}(l(m+n))$. However, the formulation remains nonconvex because of the bilinear decomposition of $\widehat{\mathbf{M}}$. Later, some research works add regularization terms to the objective function. One commonly used regularizer is of quadratic form in $\mathbf{X}$ and $\mathbf{Y}$ [8]–[10], and the problem is given as

$$\underset{\mathbf{X},\mathbf{Y}}{\text{minimize}} \underbrace{\sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{\Omega}_{ij}\left(\tilde{\mathbf{M}}_{ij} - \mathbf{x}_i^T \mathbf{y}_j\right)^2}_{\text{error loss}} + \gamma \underbrace{\left(\|\mathbf{X}\|_F^2 + \|\mathbf{Y}\|_F^2\right)}_{\text{low-rank index}}, \tag{2}$$

where $\mathbf{x}_i$ and $\mathbf{y}_j$ denote the ith and jth column of $\mathbf{X}$ and $\mathbf{Y}$, respectively, and $\gamma > 0$ is the regularization parameter. The quadratic regularizer in (2) promotes the low-rank characteristic [8]. The aforementioned problem is also named quadratically regularized PCA [9], which minimizes the weighted sum of the squared error loss and the low-rank index, indicating a tradeoff between observation error and low-rank complexity.

In [9, Sec. 4.2], the authors indicated a similar formulation with ℓ_1 error loss instead of ℓ_2:

$$\underset{\mathbf{X},\mathbf{Y}}{\text{minimize}} \sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{\Omega}_{ij}\left|\tilde{\mathbf{M}}_{ij} - \mathbf{x}_i^T \mathbf{y}_j\right| + \gamma\left(\|\mathbf{X}\|_F^2 + \|\mathbf{Y}\|_F^2\right). \tag{3}$$

The authors also pointed out that problem (3) is equivalent to the robust PCA problem [2], [11]–[16] except for an additional rank constraint. The robustness results from the ℓ_1 loss, which

Manuscript received December 08, 2015; revised April 18, 2016; accepted May 11, 2016. Date of publication May 24, 2016; date of current version July 26, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Cedric Fevotte. This work was supported by the Hong Kong RGC 2014/15-16207814 research grant. Part of the results in this paper were preliminarily presented at the Forty-Ninth IEEE Asilomar Conference on Signals, Systems and Computers 2015 [1].

The authors are with the Hong Kong University of Science and Technology (HKUST), Kowloon, Hong Kong (e-mail: lzhaoai@ust.hk; eeprabhubabu@ust.hk; palomar@ust.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2016.2572049

[1]PCA was originally designed for the full observation scenario, but can be trivially extended to missing data situations. Here we use the terminology in the general sense.

is less sensitive to large-valued outliers. The robust PCA problem has a rich literature and many algorithms have been put forward, such as APG (Accelerated Proximal Gradient) [17], ALM (Augmented Lagrange Multiplier) [2], LRSD (Low Rank and Sparse matrix Decomposition) [18], IALM (Inexact Augmented Lagrangian Method), and EALM (Exact Augmented Lagrangian Method) [19]. These algorithms adopt SVD (Singular Value Decomposition) operations or the like, which can be computationally expensive for large-scale problems. Besides that, the SVD operation is not friendly to multicore systems, which means the cost of SVD cannot be spread out by using a parallel computing machine.

Recently, several works [9], [8], [20] managed to handle these limitations by adopting the factorized formulation (like (2) and (3)). Moreover, they also considered general error loss functions. They generalized the aforementioned loss functions (ℓ_2 and ℓ_1) to any convex function $f:\ \mathbb{R}\rightarrow\mathbb{R}_+$. In terms of algorithm, [8] proposed JELLYFISH with convergence guarantee to a local minimum and [9] implemented Alternating Minimization. Both of them are gradient-based and enjoy good performance on multicore systems.

Understanding that outliers come from the additive noise matrix, we propose to use some particular loss functions to promote robustness. Previous works [9], [21], [22] showed that the loss functions have a probabilistic interpretation. For example, problem (2) is equivalent to

$$\underset{\mathbf{X},\mathbf{Y}}{\text{maximize}}\quad \exp\left(-\gamma\|\mathbf{X}\|_F^2\right)\exp\left(-\gamma\|\mathbf{Y}\|_F^2\right)\cdot\prod_{i=1}^{m}\prod_{j=1}^{n}\exp\left(-\boldsymbol{\Omega}_{ij}\left(\widetilde{\mathbf{M}}_{ij}-\mathbf{x}_i^T\mathbf{y}_j\right)^2\right), \quad (4)$$

which is the MAP (maximum a posteriori) estimator of $[\mathbf{X},\mathbf{Y}]$ under a Gaussian distribution. The ℓ_2 loss function is sensitive to outliers in that Gaussian distribution decays in an exponential square manner when the random variable deviates from the mean, which assumes low probability of far-away outliers. In order to promote robustness, we could choose loss functions from slowly-decaying or heavy-tailed distributions. However, those robust loss functions may not be convex, and thus the previously mentioned algorithms either are not applicable or lose convergence guarantee.

B. Contribution

In this paper, we propose an effective framework to promote robustness against two categories of outliers, namely dense outliers drawn from some elliptical distribution and sparse spike-like outliers with small additive Gaussian noise. The major contributions mainly lie in the following two aspects.

1) We provide two loss functions that promote robustness against two types of outliers. For dense elliptical-distributed outliers, we recommend the loss function $f_1(x)=\log\left(1+x^2/\nu\right)$ with $\nu>0$. For sparse spike-like outliers, we recommend the loss function $f_2(x)=1/\beta\cdot\log\left(\left(e^{\beta x}+e^{-\beta x}\right)/2\right)$ with $\beta>0$, which is a smooth approximation of ℓ_1 loss function.
2) We develop an efficient algorithm on the basis of [23] with provable convergence to a stationary solution, updating both factors $\mathbf{X}$ and $\mathbf{Y}$ in parallel instead of alternately. Furthermore, the proposed algorithm can be free of parameter tuning issues (unlike [23]). The proposed algorithm is simulated under synthetic and real data scenarios in Section V. Both the error level and convergence speed are more satisfactory than the benchmark algorithms. More impressively, in the real data scenario, the proposed algorithm improves the error level by 33.0% from the state-of-the-art online algorithm GRASTA and by 12.6% from the benchmark alternating minimization algorithm (intended for quadratically regularized PCA).

C. Organization and Notation

The rest of the paper is organized as follows. In Section II, we give the problem formulation and introduce the robust loss functions. In Section III, we borrow the framework of [23] and come up with the proposed Parallel Minimization algorithm. In Section IV, we provide the convergence and complexity analysis. Finally, Section V presents numerical simulations, and the conclusions are given in Section VI.

The following notation is adopted. Boldface upper-case letters represent matrices, boldface lower-case letters denote column vectors, and standard lower-case letters stand for scalars. $\mathbb{R}$ ($\mathbb{R}_+$) denotes the real (nonnegative real) field. $|\cdot|$ denotes the absolute value. $\log(\cdot)$ is the logarithm operation with base e. f' and f'' represent the first and second order derivatives of f, respectively, and $\|\cdot\|$ denotes a general norm of a vector or a matrix. For the vector case, $\|\cdot\|_p$ denotes the p-norm of a vector, with $p=1,2$ or $+\infty$ on most occasions. For the matrix case, $\|\cdot\|_*$, $\|\cdot\|_2$ and $\|\cdot\|_F$ stand for the nuclear, spectral and Frobenius norm, respectively. $\nabla(\cdot)$ represents the gradient of a vector or matrix function. $\mathbb{R}^{m\times n}$ represents the set of $m\times n$ real-valued matrices. $\mathbf{I}$ stands for the identity matrix. $\mathbf{X}_{ij}$ denotes the (i,j)th element of the matrix $\mathbf{X}$. $\mathbf{x}_i$ is the ith column of the matrix $\mathbf{X}$. $\mathbf{X}^T$, $\mathbf{X}^{-1}$, $Tr(\mathbf{X})$, and $\mathrm{vec}(\mathbf{X})$ denote the transpose, inverse, trace, and stacking vectorization of $\mathbf{X}$, respectively. Finally, $\odot$ and $\otimes$ stand for the Hadamard product and Kronecker product, respectively.

II. Problem Statement

A. Matrix Factorization Formulation

We consider the same formulation as has been proposed by [9], [8], [20]:

$$\underset{\mathbf{X}\in\mathbb{R}^{l\times m},\mathbf{Y}\in\mathbb{R}^{l\times n}}{\text{minimize}}\ \sum_{i=1}^{m}\sum_{j=1}^{n}\boldsymbol{\Omega}_{ij}f\left(\widetilde{\mathbf{M}}_{ij}-\mathbf{x}_i^T\mathbf{y}_j\right)+\gamma\left(\|\mathbf{X}\|_F^2+\|\mathbf{Y}\|_F^2\right). \quad (5)$$

For convenience, we denote

$$J(\mathbf{X},\mathbf{Y}) \triangleq \sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{\Omega}_{ij} f\left(\widetilde{\mathbf{M}}_{ij} - \mathbf{x}_i^T\mathbf{y}_j\right) + \gamma\left(\|\mathbf{X}\|_F^2 + \|\mathbf{Y}\|_F^2\right), \quad (6)$$

whose value is always nonnegative due to the nonnegativity of the loss function f. Note that f should promote robustness and is suggested to be chosen as the negative logarithm of the pdf of heavy-tailed distributions; hence, f may not be convex.

B. Robust Loss Functions

In this paper, we focus on two types of outliers: dense outliers drawn from some elliptical distribution and sparse spike-like outliers with small Gaussian noise. As for dense elliptical-distributed outliers, we consider fitting them to a particular heavy-tailed distribution. As is shown in the literature, heavy-tailed distributions yield formulations that enjoy more robustness than convex formulations [24]. In the existing literature for robust statistical modeling [25], Student's t distribution is adopted because of its polynomial-decaying thick tail which allows for high enough probability of outliers. It has been applied to a variety of statistical problems, like robust estimation of the covariance matrix [26]–[28]. Taking the negative logarithm of the pdf of Student's t distribution and neglecting the constant and scaling factor, we get the first considered robust loss function:

$$f_1(x) = \log\left(1 + x^2/\nu\right) \quad (7)$$

with $\nu > 0$. Note that f_1 is neither convex nor concave.

As for sparse spike-like outliers, we consider another distribution with slightly thinner tail: Laplace distribution. Taking the negative logarithm of its pdf, we get the absolute value loss function, the ℓ_1 loss. It is generally known that the ℓ_1-norm heuristic promotes sparsity, thus promoting robustness against sparse outliers. Note that these large sparse outliers are possibly accompanied by small Gaussian noise, Huber loss [9], [13], [29] is proposed in that it is robust both to large sparse outliers and to small dense Gaussian perturbations. Huber loss can be interpreted as a smooth approximation of the ℓ_1 loss. Observe that the Huber loss function is only first-order differentiable. To exploit higher-order properties in algorithm design, we need to further smoothen the Huber loss (a similar approach is mentioned in [30]) and hence we propose the second robust loss function:

$$f_2(x) = 1/\beta \cdot \log\left(\left(e^{\beta x} + e^{-\beta x}\right)/2\right) \quad (8)$$

with $\beta > 0$, which is a smoothed ℓ_1 loss function with arbitrary-order derivatives. In Fig. 1, we illustrate some traditional robust loss functions and the two proposed loss functions to provide insight.

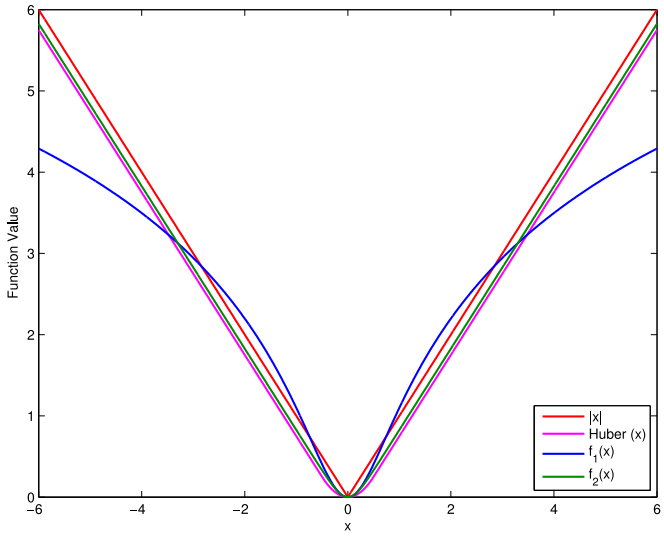


Fig. 1. Robust loss functions (traditional and proposed): $|x|$, $\mathrm{Huber}(x) = \begin{cases}(1/2)x^2 & |x| \le 1\\ |x| - 1/2 & |x| > 1\end{cases}$, $f_1(x)$ with $\nu = 0.5$, and $f_2(x)$ with $\beta = 4$.

III. Parallel Minimization Algorithm

In this section, we put forward a novel algorithm to solve (5). This algorithm can handle both proposed loss functions, i.e., f_1 and f_2, and we represent them with the general notation f. Traditional algorithms either involve costly SVD-like operations or are gradient-based methods. Thus, they can suffer from slow convergence in practice. Our proposed algorithm is based on [23], with an improvement in the step size selection that guarantees monotonicity and achieves notable progress at every iteration. The highlight of our proposed algorithm is threefold. First, we do a second-order convex approximation to $J(\mathbf{X},\mathbf{Y})$ [cf. (6)], exploiting higher-order information. Second, we propose a parallel update scheme on $\mathbf{X}$ and $\mathbf{Y}$, utilizing parallel computing resources. Third, we come up with a novel step size rule which can avoid parameter tuning and guarantee monotonic decrease, as opposed to [23].

We would like to regard $\mathbf{X}$ and $\mathbf{Y}$ as two separate agents and iteratively update both in parallel. Suppose we are at iteration k; in every iteration, we are going to carry out a two-step procedure. In the first step, we determine the descent direction of $J(\mathbf{X},\mathbf{Y})$ at $[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]$. For that we derive a second-order convex approximation of $J(\mathbf{X},\mathbf{Y})$ with respect to $\mathbf{X}$ and $\mathbf{Y}$ around $[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]$, and minimize the approximation function with respect to the same variable. This minimization problem is named the best-response problem, whose solution is denoted as $\widehat{\mathbf{X}}^{(k)}$ and $\widehat{\mathbf{Y}}^{(k)}$. The descent direction is given as $[\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}, \widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)}]$. In the second step, we construct the variable update for the next iteration $k+1$, denoted as $[\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)}]$, which takes the form $[\mathbf{X}^{(k)} + \alpha^{(k)}(\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}), \mathbf{Y}^{(k)} + \alpha^{(k)}(\widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)})]$ where $\alpha^{(k)}$ is the update step size at iteration k satisfying some step size rule. In the following, we elaborate on the two steps in detail.

A. Descent Direction Computation

We observe that $J(\mathbf{X},\mathbf{Y})$ takes on the same structure for fixed $\mathbf{X}$ and for fixed $\mathbf{Y}$. Thus, without loss of generality, we

concentrate on the derivation of $\widehat{\mathbf{X}}^{(k)}$, and $\widehat{\mathbf{Y}}^{(k)}$ can be analogously obtained. If $\mathbf{Y}$ is fixed at $\mathbf{Y}^{(k)}$, the optimization problem (5) reduces to (constant terms are removed for convenience)

$$\underset{\mathbf{X}}{\text{minimize}} \sum_{i=1}^{m}\left[\sum_{j=1}^{n}\mathbf{\Omega}_{ij}f\left(\widetilde{\mathbf{M}}_{ij}-\mathbf{x}_i^T\mathbf{y}_j^{(k)}\right)+\gamma\|\mathbf{x}_i\|_2^2\right], \quad (9)$$

which can be readily decomposed into the following subproblems: for each column of $\mathbf{X}$,

$$\begin{aligned}\underset{\mathbf{x}_i}{\text{minimize}} &\sum_{j=1}^{n}\mathbf{\Omega}_{ij}f\left(\widetilde{\mathbf{M}}_{ij}-\mathbf{x}_i^T\mathbf{y}_j^{(k)}\right)+\gamma\|\mathbf{x}_i\|_2^2\\ &\triangleq J_{\mathrm{x},i}\left(\mathbf{x}_i\right). \end{aligned}\quad (10)$$

We keep the convex term of $J_{\mathrm{x},i}\left(\mathbf{x}_i\right)$ and approximate the rest with its convex second-order Taylor expansion at $\mathbf{x}_i^{(k)}$:

$$\begin{aligned}J_{\mathrm{x},i}\left(\mathbf{x}_i\right)&\approx\gamma\mathbf{x}_i^T\mathbf{x}_i+\sum_{j=1}^{n}\mathbf{\Omega}_{ij}f\left(\widetilde{\mathbf{M}}_{ij}-\mathbf{x}_i^{(k)T}\mathbf{y}_j^{(k)}\right)\\ &+\mathbf{g}_{\mathrm{x},i}^{(k)T}\left(\mathbf{x}_i-\mathbf{x}_i^{(k)}\right)+\frac{1}{2}\left(\mathbf{x}_i-\mathbf{x}_i^{(k)}\right)^T\mathbf{H}_{\mathrm{x},i}^{(k)}\left(\mathbf{x}_i-\mathbf{x}_i^{(k)}\right)\end{aligned}\quad (11)$$

where $\mathbf{g}_{\mathrm{x},i}^{(k)}=\sum_{j=1}^{n}\mathbf{\Omega}_{ij}\mathbf{g}_{\mathrm{x},ij}^{(k)}$, $\mathbf{g}_{\mathrm{x},ij}^{(k)}=-f'(\widetilde{\mathbf{M}}_{ij}-\mathbf{x}_i^{(k)T}\mathbf{y}_j^{(k)})\mathbf{y}_j^{(k)}$, $\mathbf{H}_{\mathrm{x},i}^{(k)}=\left[\sum_{j=1}^{n}\mathbf{\Omega}_{ij}\mathbf{H}_{\mathrm{x},ij}^{(k)}\right]_+$, $\mathbf{H}_{\mathrm{x},ij}^{(k)}=f''(\widetilde{\mathbf{M}}_{ij}-\mathbf{x}_i^{(k)T}\mathbf{y}_j^{(k)})\mathbf{y}_j^{(k)}\mathbf{y}_j^{(k)T}$, and the operation $[\cdot]_+$ means taking the positive semidefinite part of a matrix. The best-response problem becomes the following QP (Quadratic Programming) (constant terms are removed for convenience):

$$\underset{\mathbf{x}_i}{\text{minimize}}\ \frac{1}{2}\mathbf{x}_i^T\left(2\gamma\mathbf{I}+\mathbf{H}_{\mathrm{x},i}^{(k)}\right)\mathbf{x}_i-\left(\mathbf{H}_{\mathrm{x},i}^{(k)}\mathbf{x}_i^{(k)}-\mathbf{g}_{\mathrm{x},i}^{(k)}\right)^T\mathbf{x}_i. \quad (12)$$

The optimal solution to (12) is

$$\widehat{\mathbf{x}}_i^{(k)}=\left(2\gamma\mathbf{I}+\mathbf{H}_{\mathrm{x},i}^{(k)}\right)^{-1}\left(\mathbf{H}_{\mathrm{x},i}^{(k)}\mathbf{x}_i^{(k)}-\mathbf{g}_{\mathrm{x},i}^{(k)}\right). \quad (13)$$

Similarly, the best-response solution $\widehat{\mathbf{y}}_j^{(k)}$ is given as

$$\widehat{\mathbf{y}}_j^{(k)}=\left(2\gamma\mathbf{I}+\mathbf{H}_{\mathrm{y},j}^{(k)}\right)^{-1}\left(\mathbf{H}_{\mathrm{y},j}^{(k)}\mathbf{y}_j^{(k)}-\mathbf{g}_{\mathrm{y},j}^{(k)}\right), \quad (14)$$

where $\mathbf{g}_{\mathrm{y},j}^{(k)}=\sum_{i=1}^{m}\mathbf{\Omega}_{ij}\mathbf{g}_{\mathrm{y},ij}^{(k)}$, $\mathbf{g}_{\mathrm{y},ij}^{(k)}=-f'(\widetilde{\mathbf{M}}_{ij}-\mathbf{x}_i^{(k)T}\mathbf{y}_j^{(k)})\mathbf{x}_i^{(k)}$, $\mathbf{H}_{\mathrm{y},j}^{(k)}=\left[\sum_{i=1}^{m}\mathbf{\Omega}_{ij}\mathbf{H}_{\mathrm{y},ij}^{(k)}\right]_+$ and $\mathbf{H}_{\mathrm{y},ij}^{(k)}=f''(\widetilde{\mathbf{M}}_{ij}-\mathbf{x}_i^{(k)T}\mathbf{y}_j^{(k)})\mathbf{x}_i^{(k)}\mathbf{x}_i^{(k)T}$. With $\widehat{\mathbf{x}}_i^{(k)}$ and $\widehat{\mathbf{y}}_j^{(k)}$ known, we get the descent direction as $[\widehat{\mathbf{X}}^{(k)}-\mathbf{X}^{(k)},\widehat{\mathbf{Y}}^{(k)}-\mathbf{Y}^{(k)}]$.

B. Step Size Computation

Now we construct the variable update and decide on the step size $\alpha^{(k)}$. The value of $\alpha^{(k)}$ measures how far we go along the descent direction just computed. To guarantee progress in every iteration, we require monotonic decrease in $J\left(\mathbf{X},\mathbf{Y}\right)$ after taking the update step. One natural choice is to adopt the Armijo step size rule, i.e., the backtracking line search method. Another choice is based on the minimization rule, i.e., the pseudo-exact line search. The prefix "pseudo" is used in that we do line search on the majorizing function (tight upper bound function) of the objective function. Although this line search method is not exact, monotonic decrease is still guaranteed.

1) Backtracking Line Search Method: The idea of the backtracking line search method follows the Armijo rule: for $k\geq 0$,

$$\begin{aligned}\text{set}\quad &\alpha^{(k)}=1;\\ \text{while}\quad &J\left(\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)}\right)-J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)>\\ &-\tau\alpha^{(k)}\left\|\left[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}\right]-\left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]\right\|_F^2\\ &\alpha^{(k)}=\alpha^{(k)}\cdot\rho;\end{aligned}\quad (15)$$

where $\alpha^{(k)}$ is the step size, $\rho\in(0,1)$ is the shrinkage parameter, and $\tau>0$ is the descent parameter chosen from $(0,2\gamma)$ controlling the descending progress of $J\left(\mathbf{X},\mathbf{Y}\right)$. More details on τ will be covered in the convergence analysis (Lemma 7 in Appendix).

Remark 1: For the backtracking line search method, we have to tune the parameter τ and ρ so as to achieve good performance. To avoid such trouble, we propose the following upper bound line search method, which is completely free of parameter tuning.

2) Upper Bound Line Search Method: To avoid the trouble of parameter tuning, we design $\alpha^{(k)}$ from another line search method. We could do exact line search on the objective function in the descent direction and set $\alpha^{(k)}$ to be the global minimizer. However, this approach can be costly due to lack of closed-form solution. To reduce the computational cost, we can alternatively do line search on an upper bound of the objective function. This upper bound function has to be carefully chosen for the sake of monotonic decrease, and the majorizing function in [31] can serve the purpose. We denote the majorizing function of $f\left(x\right)$ at $x=x_0$ as $\bar{f}\left(x,x_0\right)$. $\bar{f}\left(x,x_0\right)$ is a majorizing function of $f\left(x\right)$ at $x=x_0$ if 1) $\bar{f}\left(x,x_0\right)\geq f\left(x\right)$ for $\forall x\in\mathbb{R}$ and 2) $\bar{f}\left(x_0,x_0\right)=f\left(x_0\right)$. We can at least yield a decrease in the current function value $f\left(x_0\right)$ if we minimize $\bar{f}\left(x,x_0\right)$ with respect to x. The reason is shown as follows:

$$f\left(x_0\right)=\bar{f}\left(x_0,x_0\right)\geq\bar{f}\left(\widetilde{x},x_0\right)\geq f\left(\widetilde{x}\right), \quad (16)$$

where $\widetilde{x}\in\arg\min_x\bar{f}\left(x,x_0\right)$. To construct the majorizing function of $J\left(\mathbf{X},\mathbf{Y}\right)$, the fundamental issue is to design a majorizing function for $f\left(x\right)$. We introduce the following lemma to specify $\bar{f}\left(x,x_0\right)$.

Lemma 2: Let $f\left(x\right)$ denote either $f_1\left(x\right)$ or $f_2\left(x\right)$. The majorizing function of $f\left(x\right)$ at $x=x_0$ is $\bar{f}\left(x,x_0\right)=a\left(x_0\right)x^2+b\left(x_0\right)$ where

$$a\left(x_0\right)=\begin{cases}\frac{f'(x_0)}{2x_0} & x_0\neq 0\\ \frac{f''(0)}{2} & x_0=0\end{cases} \quad (17)$$

and

$$b\left(x_0\right)=\begin{cases}f\left(x_0\right)-\frac{f'(x_0)}{2}x_0 & x_0\neq 0\\ 0 & x_0=0.\end{cases} \quad (18)$$

Moreover, $\bar{f}'\left(x,x_0\right)|_{x=x_0}=f'\left(x_0\right)$.

Proof: The proof is trivial and omitted for lack of space. ■

It is not hard to verify that $a(x_0) \geq 0$ for all x_0; so, although $f(x)$ can be nonconvex, its majorizing function $\bar{f}(x, x_0)$ is always convex. With the help of $\bar{f}(x, x_0)$, the majorizing function of $J(\mathbf{X}, \mathbf{Y})$ is readily obtained:

$$\begin{aligned}
&\bar{J}\left(\mathbf{X}, \mathbf{Y}; \mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) \\
&\triangleq \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{\Omega}_{ij} \bar{f}\left(\widetilde{\mathbf{M}}_{ij} - \mathbf{x}_i^T \mathbf{y}_j, \widetilde{\mathbf{M}}_{ij} - \mathbf{x}_i^{(k)T} \mathbf{y}_j^{(k)}\right) \\
&\quad + \gamma\left(\sum_{i=1}^{m} \|\mathbf{x}_i\|_2^2 + \sum_{j=1}^{n} \|\mathbf{y}_j\|_2^2\right). \qquad (19)
\end{aligned}$$

We can also verify that $\nabla_{\mathbf{x}_i} \bar{J}\left(\mathbf{X}, \mathbf{Y}; \mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) = \nabla_{\mathbf{x}_i} J(\mathbf{X}, \mathbf{Y})$, $\forall i$ and $\nabla_{\mathbf{y}_j} \bar{J}\left(\mathbf{X}, \mathbf{Y}; \mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) = \nabla_{\mathbf{y}_j} J(\mathbf{X}, \mathbf{Y})$, $\forall j$ at $\left[\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right]$.

The minimization solution for $\alpha^{(k)}$ comes from the following inequality: for $k \geq 0$,

$$\begin{aligned}
&J\left(\mathbf{X}^{(k+1)}, \mathbf{Y}^{(k+1)}\right) \\
&\overset{(a)}{\leq} \bar{J}\left(\mathbf{X}^{(k+1)}, \mathbf{Y}^{(k+1)}; \mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) \\
&\overset{(b)}{=} \bar{J}\left(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}; \mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) + P_4^{(k)}\left(\alpha^{(k)}\right)^4 \\
&\quad + P_3^{(k)}\left(\alpha^{(k)}\right)^3 + P_2^{(k)}\left(\alpha^{(k)}\right)^2 + P_1^{(k)}\alpha^{(k)} \\
&\overset{(c)}{=} J\left(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) + P_4^{(k)}\left(\alpha^{(k)}\right)^4 + P_3^{(k)}\left(\alpha^{(k)}\right)^3 \\
&\quad + P_2^{(k)}\left(\alpha^{(k)}\right)^2 + P_1^{(k)}\alpha^{(k)}, \qquad (20)
\end{aligned}$$

where (a) is due to the definition of majorizing function; (b) comes from mathematical manipulation: recall $[\mathbf{X}^{(k+1)}, \mathbf{Y}^{(k+1)}] = [\mathbf{X}^{(k)} + \alpha^{(k)}(\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}), \mathbf{Y}^{(k)} + \alpha^{(k)}(\widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)})]$ and the expressions of the columns of $\widehat{\mathbf{X}}^{(k)}$ and $\widehat{\mathbf{Y}}^{(k)}$ [cf. (13) and (14)], expand $\bar{J}(\mathbf{X}^{(k+1)}, \mathbf{Y}^{(k+1)}; \mathbf{X}^{(k)}, \mathbf{Y}^{(k)})$, reorganize the terms and introduce the parameters $P_4^{(k)} \sim P_1^{(k)}$ as defined in (21) at the bottom of the page: $[\mathbf{A}_\Omega^{(k)}]_{ij} = \mathbf{\Omega}_{ij} a(\widetilde{\mathbf{M}}_{ij} - \mathbf{x}_i^{(k)T}\mathbf{y}_j^{(k)})$, $a(x)$ follows Lemma 2, $[\mathbf{B}_\Omega^{(k)}]_{ij} = 2[\mathbf{A}_\Omega^{(k)}]_{ij}(\widetilde{\mathbf{M}}_{ij} - \mathbf{x}_i^{(k)T}\mathbf{y}_j^{(k)})$, $\mathbf{C}^{(k)} = (\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)})^T(\widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)})$, $\mathbf{D}^{(k)} = \mathbf{X}^{(k)T}(\widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)}) + (\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)})^T\mathbf{Y}^{(k)}$ and the matrix operator $(\cdot)^2$ is an elementwise operator; (c) $\bar{J}\left(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}; \mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) = J\left(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right)$, following the definition of majorizing function. Now we set $\alpha^{(k)}$ to be

$$\alpha^{(k)} = \arg\min_{\alpha \in \mathbb{R}} \left\{P_4^{(k)}\alpha^4 + P_3^{(k)}\alpha^3 + P_2^{(k)}\alpha^2 + P_1^{(k)}\alpha\right\}, \qquad (22)$$

and we claim that $J(\mathbf{X}^{(k+1)}, \mathbf{Y}^{(k+1)}) \leq J(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)})$ holds because $\alpha^{(k)}$ is the global minimizer of $P_4^{(k)}\alpha^4 + P_3^{(k)}\alpha^3 + P_2^{(k)}\alpha^2 + P_1^{(k)}\alpha$ ($P_4^{(k)} \geq 0$), whose minimum is nonpositive. The global minimizer $\alpha^{(k)}$ is obtained as follows: we derive all the real zeros of its first order derivative (no more than three) and check for the global minimizer of the polynomial expression. The validity for this practice is that the global minimizer of a differentiable polynomial function must satisfy the zero derivative condition.

Remark 3: The upper bound line search method does not need a while loop and parameter tuning, but its performance is not guaranteed to be superior to the backtracking line search method. One can always apply upper bound line search method first and then do backtracking while tuning the parameters to see if the improvement is significant enough to make a switch.

Finally, we summarize the parallel minimization method in Algorithm 1.

Algorithm 1: Parallel Minimization Algorithm.

Require: $k = 0$, initial value $\mathbf{X}^{(0)}, \mathbf{Y}^{(0)}$.

1: **repeat**
2: Compute $\left[\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}, \widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)}\right]$ using (13) and (14);
3: Compute the step size $\alpha^{(k)}$ using backtracking line search method (15) or upper bound line search method (21) and (22);
4: $\left[\mathbf{X}^{(k+1)}, \mathbf{Y}^{(k+1)}\right] = \left[\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right] + \alpha^{(k)}\left[\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}, \widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)}\right]$;
5: $k = k + 1$;
6: **until** convergence

IV. Convergence and Complexity

A. Convergence Analysis

In this section, we provide some theoretical guarantee for our proposed algorithm. Previous works [8], [20] enjoyed convergence to a local minimum when the loss function f is convex and twice differentiable. The same result does not hold any more

$$\begin{cases}
P_4^{(k)} \triangleq \mathbf{1}^T\left(\mathbf{A}_\Omega^{(k)} \odot \mathbf{C}^{(k)2}\right)\mathbf{1} \\
P_3^{(k)} \triangleq 2 \cdot \mathbf{1}^T\left(\mathbf{A}_\Omega^{(k)} \odot \mathbf{C}^{(k)} \odot \mathbf{D}^{(k)}\right)\mathbf{1} \\
P_2^{(k)} \triangleq \mathbf{1}^T\left(\mathbf{A}_\Omega^{(k)} \odot \mathbf{D}^{(k)2} - \mathbf{B}_\Omega^{(k)} \odot \mathbf{C}^{(k)}\right)\mathbf{1} + \gamma\left\|\left[\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}, \widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)}\right]\right\|_F^2 \\
P_1^{(k)} \triangleq -\mathbf{1}^T\left(\mathbf{B}_\Omega^{(k)} \odot \mathbf{D}^{(k)}\right)\mathbf{1} + 2\gamma \mathrm{Tr}\left(\left[\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right]^T\left[\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}, \widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)}\right]\right) \\
\qquad = \mathrm{Tr}\left(\nabla_{[\mathbf{X},\mathbf{Y}]}^T J\left(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) \cdot \left(\left[\widehat{\mathbf{X}}^{(k)}, \widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right]\right)\right)
\end{cases} \qquad (21)$$

because our proposed robust loss function can be nonconvex. In this case, only stationary solutions can be guaranteed, as can be seen from the following theorem.

Theorem 4: Either Algorithm 1 converges to a stationary solution of (5) within a finite number of iterations, or every limit point of $\left\{\left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]\right\}_{k=1}^{+\infty}$ (assuming it exists) is a stationary solution of (5). Moreover, none of the limit-point stationary solutions is a local maximum.

Proof: Following the nature of monotonic decrease brought by both line search methods, we can see that $J\left(\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)}\right)\leq J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)$ holds for any iteration k. Also due to the nonnegativity of $J(\mathbf{X},\mathbf{Y})$, it is bounded below. Therefore, the sequence of the objective function value $\left\{J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)\right\}$ converges. The rest of the proof is dedicated to proving convergence to a stationary point; it consists of two parts which are elaborated in Appendix. ■

B. Computational Complexity

Now we discuss the computational complexity of Algorithm 1. The per-iteration computational cost comes from two sources: descent direction and step size. We analyze them separately. Recall that the size of $\mathbf{X}$ and $\mathbf{Y}$ is $l\times m$ and $l\times n$, respectively. When computing the descent direction, we need to solve $m+n$ systems of linear equations, each of size l. The cost of each is of complexity $\mathcal{O}\left(l^3\right)$; the total cost is $\mathcal{O}\left(l^3\left(m+n\right)\right)$. When computing the step size, we analyze the cost of upper bound line search. The dominating cost is (21): 1) computing $\mathbf{A}_\Omega^{(k)}$ and $\mathbf{B}_\Omega^{(k)}$ requires complexity $\mathcal{O}\left(lmn\right)+\mathcal{O}\left(mn\right)$: $\mathcal{O}\left(lmn\right)$ from matrix multiplication and $\mathcal{O}\left(mn\right)$ from Hadamard product operations, and computing $\mathbf{C}^{(k)}$ and $\mathbf{D}^{(k)}$ still requires $\mathcal{O}\left(lmn\right)+\mathcal{O}\left(mn\right)$: $\mathcal{O}\left(lmn\right)$ from matrix multiplication and $\mathcal{O}\left(mn\right)$ from matrix addition; 2) computing $P_4^{(k)}\sim P_1^{(k)}$ need additionally a few Hadamard product operations and summations, of complexity $\mathcal{O}\left(mn\right)$, and some other basic operations (norm, trace), of complexity $\mathcal{O}\left(\left(m+n\right)l^2\right)+\mathcal{O}\left(\left(m+n\right)l\right)$. The overall per-iteration cost is $\mathcal{O}\left(l^3\left(m+n\right)\right)+\mathcal{O}\left(lmn\right)$, neglecting the lower-order terms. Since l is assumed to be much smaller than m and n, $\mathcal{O}\left(l^3\left(m+n\right)\right)+\mathcal{O}\left(lmn\right)\approx\mathcal{O}\left(lmn\right)$. If we allow for parallel computation in solving the linear equation systems, the computational cost can be distributed and hence lowered to $\mathcal{O}\left(l^3\right)+\mathcal{O}\left(lmn\right)\approx\mathcal{O}\left(lmn\right)$. The complexity order cannot be lowered further because we cannot avoid matrix multiplication operations.

V. Numerical Simulations

In this section, we do numerical simulations on both synthetic data and real data. All simulations are performed on a PC with a 3.20 GHz i5-4570 CPU and 8 GB RAM. Parallel computing is realized by the "parfor" command in Matlab, with a total of four workers.

A. Synthetic Data Experiments

We generate the true data matrix $\mathbf{M}\left(\in\mathbb{R}^{m\times n}\right)=\mathbf{M}_1^T\mathbf{M}_2$ where $\mathbf{M}_1\in\mathbb{R}^{l\times m}$ and $\mathbf{M}_2\in\mathbb{R}^{l\times n}$ have i.i.d. Gaussian entries drawn from $\mathcal{N}\left(0,1\right)$ and then normalized so that the average squared magnitude of $\mathbf{M}$ is 1. The observation set is $\mathbf{\Omega}$, whose cardinality is some ratio of all the entries of mn. The default ratio is 50%, unless otherwise specified. With regard to outliers, i.e., noise matrix $\mathbf{N}$, we generate two types. The dense outliers are elliptically-distributed heavy-tailed noise, whose entries are i.i.d. following $\mathbf{N}_{ij}=\sqrt{\boldsymbol{\tau}_{ij}}\mathbf{U}_{ij}$ with $\boldsymbol{\tau}_{ij}\sim\chi^2$ and $\mathbf{U}_{ij}\sim\mathcal{N}\left(0,1\right)$. The sparse outliers are large spikes plus small Gaussian noise, whose entries are i.i.d. following $\mathbf{N}_{ij}=\sigma\mathbf{U}_{ij}+S\cdot\mathrm{Ind}_{ij}$ with $\sigma=0.1$, $S=80\gg0.1$, $\mathrm{Ind}_{ij}=0$ or 1, and the cardinality of Ind is $0.15\,mn$, i.e., 15% of all the entries. For convenience, we set $n=m$, and vary m in $\{250,300,\ldots,600\}$ for different matrix sizes. The true rank is set to be $m/50$. Our assumed rank upper bound $l=\left\lfloor\frac{|\mathbf{\Omega}|}{3(m+n)}\right\rfloor\approx\frac{m}{12}$ [8], more than 4 times larger than the true rank. As for the tuning parameters: γ and, ν or β, we do a grid search for each parameter on a particular range and pick the tuple (γ,ν) or (γ,β) that yields the smallest NMSE (normalized mean squared error, to be defined later). This is to eliminate the effect of parameter tuning. For all the iterative algorithms in the simulation, the stopping criterion is $\left\|\left[\mathbf{X}^{(k+1)}-\mathbf{X}^{(k)},\mathbf{Y}^{(k+1)}-\mathbf{Y}^{(k)}\right]\right\|_F/\left(\left(m+n\right)l\right)\leq\mathrm{Tol}$ with Tol being the tolerant precision.

1) NMSE of Various Loss Functions: First we justify why we use f_1 and f_2 to handle the outliers. For performance evaluation, we use NMSE, namely ($\widehat{\mathbf{M}}$ is the recovered matrix)

$$\mathrm{NMSE}(\widehat{\mathbf{M}})=\mathrm{E}\left[\left\|\mathbf{M}-\widehat{\mathbf{M}}\right\|_F^2\right]/\|\mathbf{M}\|_F^2. \tag{23}$$

The expectation is approximated by 30 Monte Carlo realizations. We compare the recommended loss function with some other loss functions: x^2, $|x|$, and $\mathrm{Huber}\left(x\right)=\begin{cases}\left(1/2\right)x^2 & |x|\leq1\\ |x|-1/2 & |x|>1\end{cases}$. The implementation for the non-recommended loss functions is either the alternating gradient method, cf. [9, Sec. 7, Algorithm 2], or the parallel gradient method, cf. [8] and [32, Algorithm 1]. To the best of our knowledge, we do not have other methods available in the existing literature, so we may as well choose the one that gives better performance. In Fig. 2, we present the error level of different loss functions under dense and sparse outliers. In the case of dense outliers, the proposed function $f_1\left(x\right)=\log\left(1+x^2/\nu\right)$ achieves the lowest NMSE at all matrix sizes; the error level stays almost steady, below 0.3. The error levels of the other loss functions are higher above: the quadratic loss obtains the highest NMSE, above 0.9; the Huber loss and absolute value loss produce similar NMSE which falls in the range 0.3–0.4, still not as good as the proposed function f_1. This is because none of these three loss functions are derived from the pdf's of heavy-tailed elliptical distributions; they are in nature non-robust to heavy-tailed elliptical noise, i.e., dense outliers. In the case of sparse outliers, the situation is slightly different. The proposed function $f_2\left(x\right)=1/\beta\cdot\log\left(\left(e^{\beta x}+e^{-\beta x}\right)/2\right)$ achieves the second lowest NMSE. The lowest NMSE is achieved by the absolute value loss. However, the gap of performance loss is not so significant and even shrinks as the matrix size increases. Besides

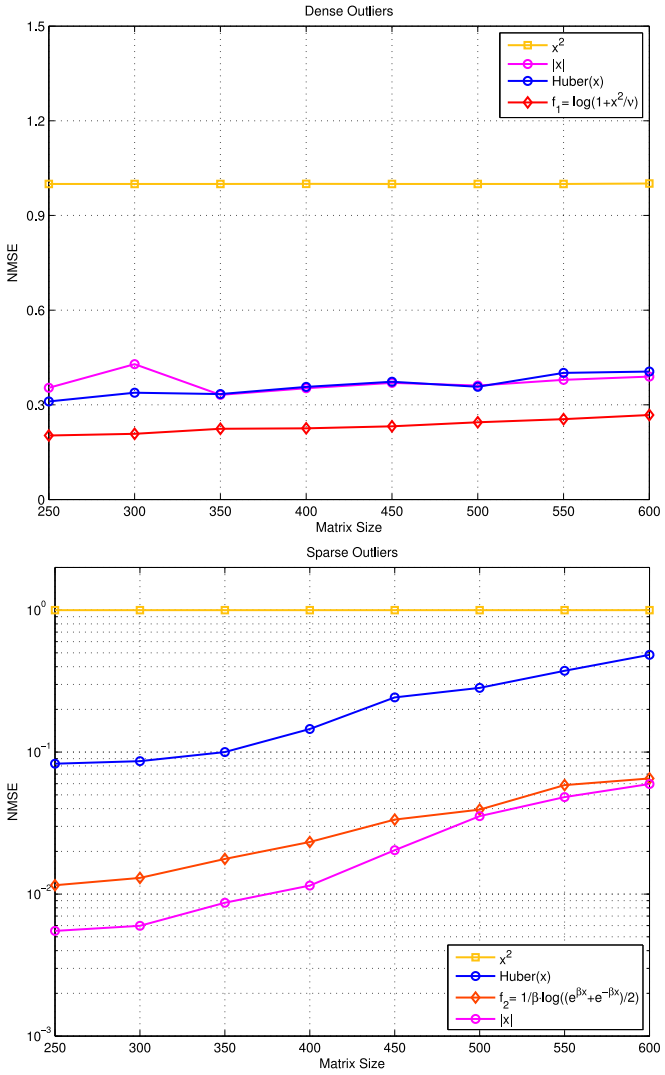


Fig. 2. NMSE of different loss functions under 1) dense outliers (upper) and 2) sparse outliers (lower).

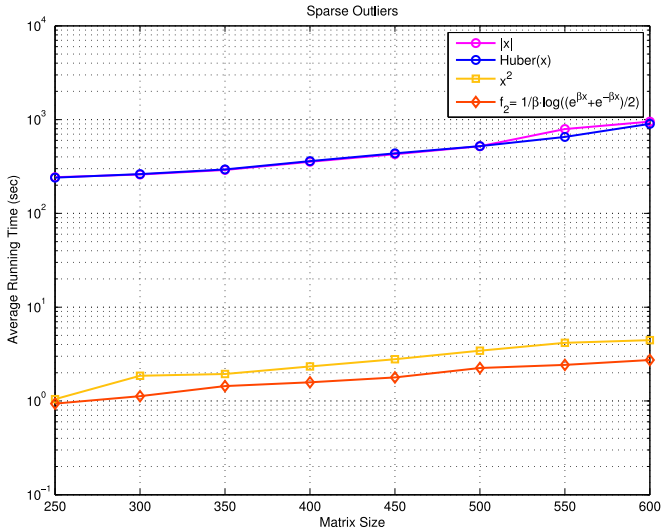


Fig. 3. Average running time (in seconds) of different loss functions.

that, we find the computational cost of using the absolute value loss is too high. In Fig. 3, we present the computational cost of using different loss functions. We see that using the absolute value loss and Huber loss is extremely costly; they need several hundred seconds to converge. The proposed function f_2 is much more efficient; it is more than two orders of magnitude faster than the absolute value loss. The proposed function f_2 achieves a good tradeoff between the NMSE and computational cost, so adopting f_2 makes more sense.

2) Computational Running Time: After having justified f_1 and f_2, we show the efficiency of our proposed algorithm. We compare our proposed algorithms with some existing methods under the same objective functions, and the performance evaluation is the average running time on the CPU. All the algorithms use the recommended loss function in the corresponding scenario and there is no need to compare the NMSE because the error levels achieved by those algorithms are more or less the same. There are two types of benchmark algorithms which can be applied to any general loss functions. One is the alternating gradient method, proposed by Udell *et al.* [9, Sec. 7, Algorithm 2]. The other type is the parallel gradient method, which is mentioned by Recht and Ré [8] and Sun and Luo [32, Algorithm 1]. In Fig. 4, we present the average running time of different algorithms. Under either dense or sparse outliers, our proposed algorithm converges faster than the benchmark algorithms; the average running time of the proposed algorithm (with either step size computation method) is more than one and a half orders of magnitude less than that of the benchmark algorithms. It can be ascribed to the nature of the benchmarks: they are first-order algorithms. Our algorithm exploits second-order information in computing descent direction and thus has faster convergence speed, despite slightly higher per-iteration computational cost. Another interesting fact is that the two line search methods produce very similar CPU time consumption, although in one case the upper bound line search is always slightly better. In terms of performance, either one is a good choice. But in view of parameter tuning issues, we may recommend the parameter-tuning-free upper bound line search method.

3) Miscellaneous Results: There are some other results worth looking at. The first one is how NMSE changes with the number of observed entries. We fix $n = m = 400$, and the true rank is 8. In Fig. 5, we plot the NMSE versus observation ratio from 0.2 to 0.9. Under dense outliers, we see at first a remarkable decrease in error level for more observed entries, but later on the NMSE stays almost steady. It indicates that within a certain range, more observations can significantly bring down NMSE; beyond a particular threshold, more observations will not help to reduce NMSE much. Under sparse outliers, the same situation also happens. So the message is, to achieve a satisfactory level of NMSE, observing 60% to 70% percent of the full matrix would be enough.

The second interesting result is how NMSE behaves as the true rank increases (still no more than the assumed upper bound l). We fix $n = m = 400$, the observation ratio is 50%, and rank upper bound $l = 33$. The true rank takes the range $\{2, 4, \ldots, 32\}$. In Fig. 6, we plot the NMSE versus the true rank

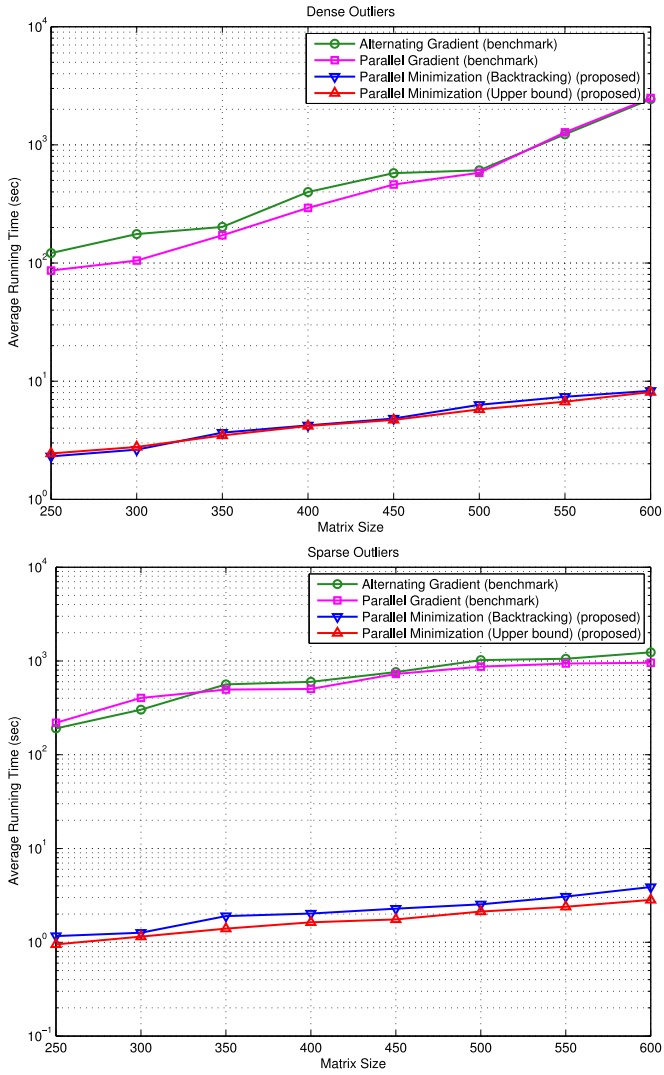


Fig. 4. Average running time (in seconds) of different algorithms under 1) dense outliers (upper), objective: f_1 and 2) sparse outliers (lower), objective: f_2.

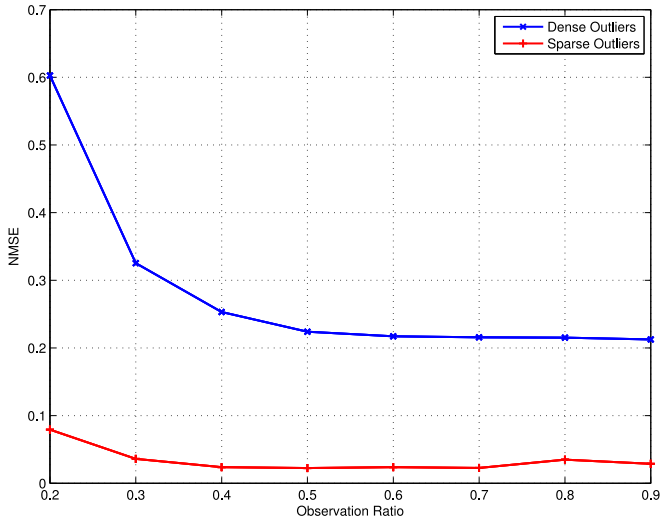


Fig. 5. NMSE of different observation ratios.

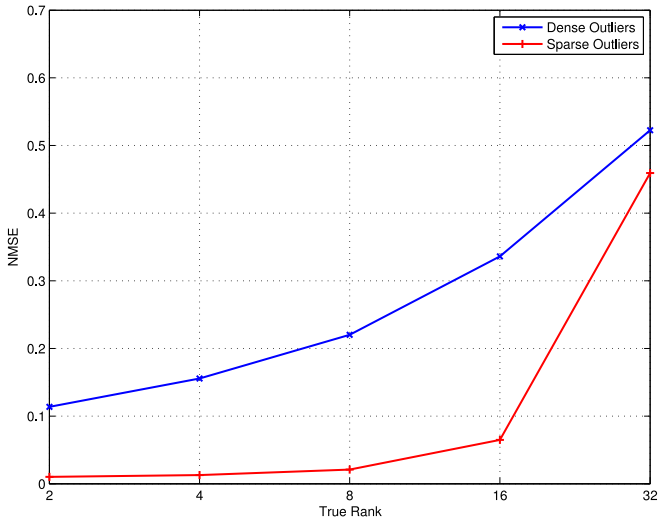


Fig. 6. NMSE versus the true rank of the original matrix.

of the original matrix. Under either dense or sparse outliers, we can see an increase in error level as the true rank is increased. When the true rank is no more than 8 ($\approx 33 \times 1/4$), the NMSE level grows moderately; when the true rank is larger than 8, the NMSE level increases remarkably. This phenomenon indicates that, if we are provided with some prior knowledge of the true rank, we should set the rank upper bound l to be at least 4 times the true value so as to get a satisfactory error level.

B. Real Data Experiments

Now we move on to real data experiments. We consider the MovieLens-100 K dataset [33]. This dataset consists of 100000 ratings (from 1 to 5) on 1 682 movies from 943 users. Each user has rated at least 20 movies. We use 80% of the data for training and the rest 20% for error testing. To ensure the existence of outliers, we manually set 15% of the training data to be either 5 or 1 with equal probability. This practice can be justified as malicious ratings in real life: a number of users may be hired to promote some movie and blindly give a 5 to that movie, or they are forced to give a 1 to its competitors. For performance evaluation, we modify the evaluation measure to be RMSE (root mean square error) of the available observations reserved for error testing:

$$\mathrm{RMSE}(\widehat{\mathbf{M}}) = \sqrt{\mathrm{E}\left[\left\|\mathbf{\Omega}_t \odot \left(\mathbf{M}_t - \widehat{\mathbf{M}}\right)\right\|_F^2\right] / \mathrm{card}\left(\mathbf{\Omega}_t\right)} \tag{24}$$

where $\mathbf{\Omega}_t$ is an indication matrix showing whether a certain entry is in the testing set or not, and $\mathbf{M}_t$ is the testing data matrix. The expectation is approximated by 10 Monte Carlo realizations.

Besides our algorithm, we also compare performance with the traditional Alternating Minimization algorithm (mentioned in [9, Sec. 7, Algorithm 1], used for solving the quadratically regularized PCA), the SVD-based benchmark APG (Accelerated Proximal Gradient) and the state-of-the-art online sequential algorithm GRASTA by He *et al.* [34]. The benchmark APG deals

TABLE I
RESULTS ON MOVIELENS-100 K DATASET. BOTH RMSE AND TIME (SEC) ARE AVERAGED FROM 10 MONTE CARLO REALIZATIONS. ALTERNATING MINIMIZATION USES QUADRATIC LOSS, CF. [9]; APG USES QUADRATIC LOSS, CF. [17]; GRASTA USES THE ℓ_1 LOSS, CF. [34]; PARALLEL MINIMIZATION USES $f_2(x) = 1/\beta \cdot \log\left(\left(e^{\beta x} + e^{-\beta x}\right)/2\right)$, ONE OF THE PROPOSED LOSS FUNCTIONS

	t_1		t_2		t_3		t_4		t_5	
	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time
Alternating Minimization (benchmark)	1.1840	4.4201	1.1417	4.5505	1.1397	4.1816	1.1438	4.1339	1.1785	4.3450
APG (benchmark)	1.0492	203.4381	1.0358	203.2578	1.0342	202.7188	1.0348	202.6786	1.0489	203.3632
GRASTA (benchmark)	1.2540	9.1171	1.8371	9.3238	1.3967	9.7546	1.4611	9.1017	1.6053	9.8277
Parallel Minimization (proposed)	1.0261	9.2506	1.0092	9.6388	1.0029	9.7511	1.0046	9.6928	1.0182	9.2790

with ℓ_2 loss, regularized by a nuclear norm term (no rank upper bound is imposed). We find more than one version of online algorithms; Balzano *et al.* [35] also proposed GROUSE before GRASTA. The loss function of GROUSE is also quadratic loss, which is not robust to outliers at all. In face of real dataset, the sample code of GROUSE suffers from singularity issues, possibly resulting from the highly incomplete nature of the data matrix. As for GRASTA, the loss function is the ℓ_1 loss. The algorithm is a two-step procedure: the first step is to get a low-rank subspace from the sequential training of online samples, i.e., the matrix columns. After a few rounds of training, the first step terminates with an output of a $m \times l$ subspace. The second step is to estimate the low-dimensional weight vectors with the output subspace. There are n weight vectors, each of length l. Holding all the weight vectors as columns, we get a $l \times n$ weight matrix. Multiplying the subspace with the weight matrix, we obtain the low-rank matrix. The weakness of GRASTA is twofold. First, this two-step algorithm does not have any convergence guarantee; the output result may not even be a stationary solution. Second, the termination of the subspace estimation step is rather tricky and involves a lot of parameter tuning work.

The results are shown as follows. We repeat the training-testing procedure 5 times. We denote them as $t_1, t_2, \ldots, t_5$. Since GRASTA uses the l_1 loss, we adopt the smoothened l_1 loss $f_2(x) = 1/\beta \cdot \log\left(\left(e^{\beta x} + e^{-\beta x}\right)/2\right)$ for fair comparison. Parameter tuning follows the practice in the synthetic data experiments. We fix the rank upper bound $l = 10$ for the proposed algorithm and the benchmark Alternating Minimization and GRASTA; APG does not impose rank upper bound. In Table I, we present the numerical results of the four algorithms.

The Alternating Minimization algorithm is superior to the other three algorithms in average running time (almost twice as fast as GRASTA and Parallel Minimization, almost two orders of magnitude faster than APG), but cannot get the lowest RMSE due to the non-robustness of quadratic error loss. The APG algorithm achieves a satisfactory RMSE level (still not the lowest) because its optimization problem adopts a convex formulation without a rank upper bound, but the running time is too long, at least one order of magnitude longer than the rest. The state-of-the-art GRASTA and our proposed Parallel Minimization spend almost the same amount of running time; their convergence speed is comparable, within 10 seconds. However, the proposed algorithm achieves lower RMSE than that provided by GRASTA. If we take the mean of the 5 procedures, the proposed algorithm improves the RMSE level by 33.0% from GRASTA, and by 12.6% from Alternating Minimization. Moreover, among the two algorithms on robust loss functions, the RMSE provided by GRASTA experiences more volatility, with standard deviation 0.2219; the RMSE provided by Parallel Minimization is much more stable, with standard deviation 0.0098.

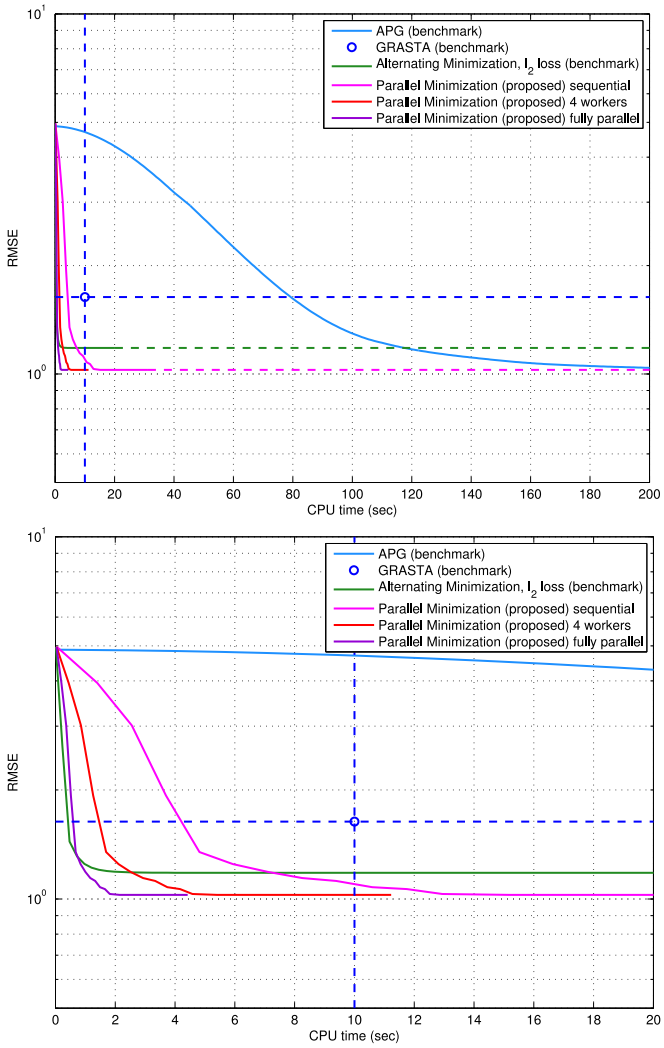


Fig. 7. RMSE versus CPU time (sec). $l = 10$. The lower plot is zoomed in from the upper plot within the time interval $[0, 20]$ seconds.

In order to better support our claims, we show the result in a particular training-testing procedure, e.g., t_2. In Fig. 7, we

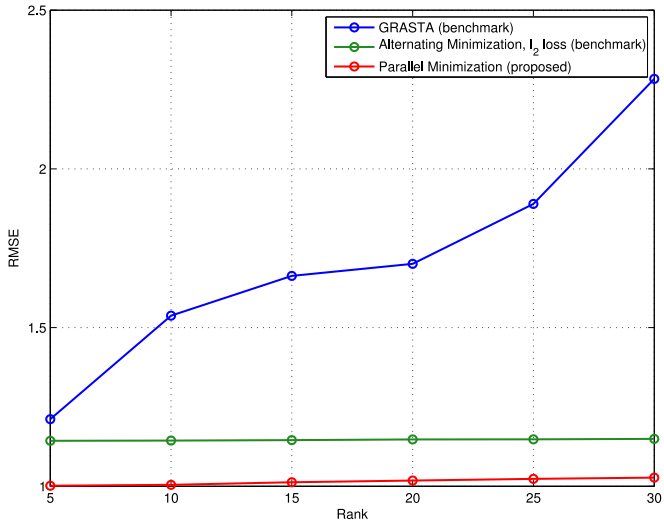


Fig. 8. RMSE versus rank upper bound.

present the convergence curve of RMSE versus CPU time. Alternating Minimization converges fast but eventually at a sub-optimal error level. APG achieves a satisfactory RMSE in the end, but needs 200 seconds to converge. GRASTA is not an iterative algorithm, and we simply put a dot showing its final RMSE and time. For detailed comparison, we look at the lower plot of Fig. 7. By default, we use "parfor" connecting 4 workers, and it reaches the dashed RMSE level in 2 seconds and the bottom in 4.5 seconds. If we use the plain "for", i.e., sequentially executing the for loop, it reaches the dashed line in 4.2 seconds (still better than GRASTA) and the bottom in 13 seconds (worse than GRASTA). Lastly, if we are allowed to execute the for loop fully in parallel, the dashed line can be reached in 0.8 second and the bottom, in 2 seconds, which is far better than GRASTA.

For the algorithms that need to be assigned rank upper bound l (all except APG), we vary it in $\{5, 10, \ldots, 30\}$. In Fig. 8, we present the RMSE versus rank upper bound. We observe that the RMSE level of GRASTA increases remarkably as the rank increases, which means GRASTA is sensitive to the choice of parameter l. The other two algorithms each can provide a stable level of RMSE, regardless of the choice of parameter l: the proposed Parallel Minimization algorithm provides an error level of about 1.01, while the Alternating Minimization reaches about 1.14.

VI. Conclusion

We have considered robust low-rank matrix completion in the presence of outliers. We have mainly focused on two types of outliers and have provided two loss functions to promote robustness. Then, we have solved the matrix completion problem using a parallel successive convex minimization algorithm. The method requires i) the computation of the descent direction via a convex second-order approximation and ii) the derivation of the step size with two line search methods, both of which enable monotonic decrease in the objective function value. We have also provided the convergence and complexity analysis for the proposed algorithm. Numerical simulations have shown that the proposed algorithm obtains a better solution with faster convergence speed than the benchmark algorithms.

Appendix
Supplement Proof of Theorem 4

Proof: Here we prove convergence to a stationary point. For clarity, the proof is divided into two parts:

Part 1) We introduce the following supporting lemma on descent direction.

Lemma 5: For every given $[\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}]$, $[\widehat{\mathbf{X}}^{(k)}, \widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}]$ is a descent direction of $J(\mathbf{X}, \mathbf{Y})$ at $\left[\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right]$ such that

$$\begin{aligned}&\mathrm{Tr}\left(\nabla^T_{[\mathbf{X},\mathbf{Y}]} J\left(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) \cdot \left(\left[\widehat{\mathbf{X}}^{(k)}, \widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right]\right)\right) \\ &\leq -2\gamma \left\| \left[\widehat{\mathbf{X}}^{(k)}, \widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right] \right\|_F^2 \end{aligned} \tag{25}$$

(γ is the same value as that in (5)).

Proof: The idea of the proof originates from [23]. Given $\left[\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right]$, for $\forall i$, $\widehat{\mathbf{x}}_i^{(k)}$ is the unique solution of the problem (12) and thus satisfies the minimum principle: for all $\mathbf{x}_i$,

$$\left(\mathbf{H}_{\mathrm{x},i}^{(k)} \left(\widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)}\right) + \mathbf{g}_{\mathrm{x},i}^{(k)} + 2\gamma \widehat{\mathbf{x}}_i^{(k)}\right)^T \left(\mathbf{x}_i - \widehat{\mathbf{x}}_i^{(k)}\right) \geq 0 \tag{26}$$

Choosing $\mathbf{x}_i = \mathbf{x}_i^{(k)}$, we can get

$$\begin{aligned} 0 \leq & \left(\mathbf{H}_{\mathrm{x},i}^{(k)} \left(\widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)}\right) + \mathbf{g}_{\mathrm{x},i}^{(k)} + 2\gamma \widehat{\mathbf{x}}_i^{(k)}\right)^T \left(\mathbf{x}_i^{(k)} - \widehat{\mathbf{x}}_i^{(k)}\right) \\ = & -\left(\widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)}\right)^T \mathbf{H}_{\mathrm{x},i}^{(k)} \left(\widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)}\right) - \mathbf{g}_{\mathrm{x},i}^{(k)T} \left(\widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)}\right) \\ & - 2\gamma \left(\widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)} + \mathbf{x}_i^{(k)}\right)^T \left(\widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)}\right) \\ \leq & -\left(\mathbf{g}_{\mathrm{x},i}^{(k)} + 2\gamma \mathbf{x}_i^{(k)}\right)^T \left(\widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)}\right) - 2\gamma \left\| \widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)} \right\|_F^2 \\ = & -\nabla^T_{\mathbf{x}_i} J\left(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) \left(\widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)}\right) - 2\gamma \left\| \widehat{\mathbf{x}}_i^{(k)} - \mathbf{x}_i^{(k)} \right\|_F^2. \end{aligned} \tag{27}$$

The same argument can be applied to $\mathbf{y}_j$, $\forall j$. Combining the summation over i and j, we obtain (25). ■

Now we are ready for the following proposition.

Proposition 6: For the backtracking line search method, there exists a constant $\eta > 0$ such that $J\left(\mathbf{X}^{(k+1)}, \mathbf{Y}^{(k+1)}\right) - J\left(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) \leq -\eta \|[\widehat{\mathbf{X}}^{(k)}, \widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}]\|_F^2$ for $k \geq 0$; For the upper bound line search method, either $\left[\mathbf{X}^{(0)}, \mathbf{Y}^{(0)}\right]$ is a stationary solution or there exists a constant $\eta > 0$ such that $J\left(\mathbf{X}^{(k+1)}, \mathbf{Y}^{(k+1)}\right) - J\left(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) \leq -\eta \|[\widehat{\mathbf{X}}^{(k)}, \widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}]\|_F^2$ for $k \geq 1$.

Proof: With Lemma 5, we can evaluate the decrease in objective function value. Our proof deals with the two line search methods separately in two cases.

Case 1: We consider the backtracking line search method. In this method, we aim to choose proper τ so that for all $k \geq 0$, the while loop is effective in gaining decrease. To proceed with the proof, we give another supporting lemma as follows.

Lemma 7: For any $k \geq 0$, there exists $\alpha^{(k)} = \rho^{s_k} \in (0, 1]$, $s_k = 0, 1, 2, \ldots$, with any given $\tau \in (0, 2\gamma)$, such that

$J(\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)})-J(\mathbf{X}^{(k)},\mathbf{Y}^{(k)})\le-\tau\alpha^{(k)}\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}]-[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2$.

Proof: The proof follows the general idea of ([36], Proposition 1.2.1) but we adapt it to this particular scenario. Denote $\mathbf{Z}\triangleq[\mathbf{X},\mathbf{Y}]$, $\mathbf{Z}^{(k)}\triangleq\left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]$, $\Delta\mathbf{Z}^{(k)}\triangleq\left[\Delta\mathbf{X}^{(k)},\Delta\mathbf{Y}^{(k)}\right]$, where $\Delta\mathbf{X}^{(k)}=\widehat{\mathbf{X}}^{(k)}-\mathbf{X}^{(k)}$ and $\Delta\mathbf{Y}^{(k)}=\widehat{\mathbf{Y}}^{(k)}-\mathbf{Y}^{(k)}$. Then we see (28) for $t\in(0,1]$

$$\begin{aligned}
&J\left(\mathbf{Z}^{(k)}+t\cdot\Delta\mathbf{Z}^{(k)}\right)-J\left(\mathbf{Z}^{(k)}\right)\\
&=\int_0^t\operatorname{vec}\left(\nabla_{\mathbf{Z}}J\left(\mathbf{Z}^{(k)}+\xi\cdot\Delta\mathbf{Z}^{(k)}\right)\right)^T d\xi\cdot\operatorname{vec}\left(\Delta\mathbf{Z}^{(k)}\right)\\
&=\int_0^t\operatorname{vec}\left(\nabla_{\mathbf{Z}}J\left(\mathbf{Z}^{(k)}+\xi\cdot\Delta\mathbf{Z}^{(k)}\right)-\nabla_{\mathbf{Z}}J\left(\mathbf{Z}^{(k)}\right)\right)^T d\xi\\
&\cdot\operatorname{vec}\left(\Delta\mathbf{Z}^{(k)}\right)+t\cdot\operatorname{vec}\left(\nabla_{\mathbf{Z}}J\left(\mathbf{Z}^{(k)}\right)\right)^T\cdot\operatorname{vec}\left(\Delta\mathbf{Z}^{(k)}\right)\\
&\overset{(a)}{=}\operatorname{vec}\left(\Delta\mathbf{Z}^{(k)}\right)^T\int_0^t\int_0^\xi D_{\mathbf{Z}}^T\left(\nabla_{\mathbf{Z}}J\left(\mathbf{Z}^{(k)}+\zeta\cdot\Delta\mathbf{Z}^{(k)}\right)\right)d\zeta d\xi\\
&\cdot\operatorname{vec}\left(\Delta\mathbf{Z}^{(k)}\right)+t\cdot\operatorname{vec}\left(\nabla_{\mathbf{Z}}J\left(\mathbf{Z}^{(k)}\right)\right)^T\cdot\operatorname{vec}\left(\Delta\mathbf{Z}^{(k)}\right)\\
&\overset{(b)}{\le}\operatorname{vec}\left(\Delta\mathbf{Z}^{(k)}\right)^T\cdot\frac{1}{2}Mt^2\cdot\operatorname{vec}\left(\Delta\mathbf{Z}^{(k)}\right)\\
&+t\cdot(-2\gamma)\operatorname{vec}\left(\Delta\mathbf{Z}^{(k)}\right)^T\\
&\cdot\operatorname{vec}\left(\Delta\mathbf{Z}^{(k)}\right)=-t\left(2\gamma-\frac{1}{2}Mt\right)\left\|\Delta\mathbf{Z}^{(k)}\right\|_F^2,
\end{aligned}\tag{28}$$

where (a) $D_{\mathbf{Z}}(\nabla_{\mathbf{Z}}J(\mathbf{Z}^{(k)}+\zeta\cdot\Delta\mathbf{Z}^{(k)}))$ stands for the Jacobian matrix of $\nabla_{\mathbf{Z}}J(\mathbf{Z}^{(k)}+\zeta\cdot\Delta\mathbf{Z}^{(k)})$ with respect to $\mathbf{Z}$; and (b) we define $M\triangleq\sup_{\zeta\in[0,1]}[\boldsymbol{\lambda}_{\max}(D_{\mathbf{Z}}(\nabla_{\mathbf{Z}}J(\mathbf{Z}^{(k)}+\zeta\cdot\Delta\mathbf{Z}^{(k)})))]$ and $\operatorname{vec}(\nabla_{\mathbf{Z}}J(\mathbf{Z}^{(k)}))^T\cdot\operatorname{vec}(\Delta\mathbf{Z}^{(k)})\le-2\gamma\operatorname{vec}(\Delta\mathbf{Z}^{(k)})^T\cdot\operatorname{vec}(\Delta\mathbf{Z}^{(k)})$ according to Lemma 5. If $M\le0$, then $J(\mathbf{Z}^{(k)}+t\cdot\Delta\mathbf{Z}^{(k)})-J(\mathbf{Z}^{(k)})\le-t(2\gamma-\frac{1}{2}Mt)\|\Delta\mathbf{Z}^{(k)}\|_F^2\le-t\cdot2\gamma\|\Delta\mathbf{Z}^{(k)}\|_F^2\le-\tau\cdot t\|\Delta\mathbf{Z}^{(k)}\|_F^2$ holds for all $t\in[0,1]$ and we can choose $\alpha^{(k)}=1=\rho^0\in(0,1]$. If $M>0$, then $J(\mathbf{Z}^{(k)}+t\cdot\Delta\mathbf{Z}^{(k)})-J(\mathbf{Z}^{(k)})\le-t(2\gamma-\frac{1}{2}Mt)\|\Delta\mathbf{Z}^{(k)}\|_F^2\le-\tau\cdot t\|\Delta\mathbf{Z}^{(k)}\|_F^2$ holds for $t\in[0,\min(\frac{4\gamma-2\tau}{M},1)]\subseteq[0,1]$, and it is always possible to choose some s_k such that $\alpha^{(k)}=\rho^{s_k}\le\min(\frac{4\gamma-2\tau}{M},1)$, with $\rho^{s_k}\in(0,1]$ for sure. ■

Now that the backtracking line search method has made possible a strict decrease in every iteration, we can conclude for $k\ge0$ with given descent parameter $\tau\in(0,2\gamma)$, $J(\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)})-J(\mathbf{X}^{(k)},\mathbf{Y}^{(k)})\le-\tau\alpha^{(k)}\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}]-[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2\le-\tau\cdot\min_{k\ge0}(\alpha^{(k)})\cdot\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}]-[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2$ and we can choose $\eta=\tau\cdot\min_{k\ge0}(\alpha^{(k)})>0$ because $\alpha^{(k)}>0$ for all k.

Case 2: We consider the upper bound line search method. We introduce a sublevel set based on the initial value $\mathbf{X}^{(0)}$ and $\mathbf{Y}^{(0)}$, given as $\mathcal{S}\triangleq\left\{[\mathbf{X},\mathbf{Y}]\,\middle|\,J(\mathbf{X},\mathbf{Y})\le J\left(\mathbf{X}^{(0)},\mathbf{Y}^{(0)}\right)\right\}$. Then we present a third supporting lemma on Lipschitz continuity.

Lemma 8: For $\forall\mathbf{Z}=[\mathbf{X},\mathbf{Y}]\in\mathcal{S}$, $\nabla_{\mathbf{Z}}J(\mathbf{X},\mathbf{Y})=\nabla_{\mathbf{Z}}J(\mathbf{Z})$ is Lipschitz continuous; i.e., for $\forall\mathbf{Z}_1,\mathbf{Z}_2\in\mathcal{S}$ and $\mathbf{Z}_1\ne\mathbf{Z}_2$, there exists a constant $L<+\infty$ such that $\|\nabla_{\mathbf{Z}}J(\mathbf{Z}_1)-\nabla_{\mathbf{Z}}J(\mathbf{Z}_2)\|_F\le L\|\mathbf{Z}_1-\mathbf{Z}_2\|_F$.

Proof: We denote $\mathbf{x}_i=\mathbf{Z}\mathbf{e}_{\mathbf{x}_i}$ and $\mathbf{y}_j=\mathbf{Z}\mathbf{e}_{\mathbf{y}_j}$, and $J(\mathbf{X},\mathbf{Y})$ becomes $J(\mathbf{Z})=\sum_{i=1}^m\sum_{j=1}^n\boldsymbol{\Omega}_{ij}f(\widetilde{\mathbf{M}}_{ij}-\mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j})+\gamma(\sum_{i=1}^m\mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{x}_i}+\sum_{j=1}^n\mathbf{e}_{\mathbf{y}_j}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j})$. We compute the gradient of $J(\mathbf{Z}):\nabla_{\mathbf{Z}}J(\mathbf{Z})=-\sum_{i=1}^m\sum_{j=1}^n\boldsymbol{\Omega}_{ij}f'(\widetilde{\mathbf{M}}_{ij}-\mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j})\mathbf{Z}(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T+\mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T)+2\gamma\mathbf{Z}$. Also, we get

$$\begin{aligned}
&\|\nabla_{\mathbf{Z}}J(\mathbf{Z}_1)-\nabla_{\mathbf{Z}}J(\mathbf{Z}_2)\|_F\\
&=\|\operatorname{vec}(\nabla_{\mathbf{Z}}J(\mathbf{Z}_1))-\operatorname{vec}(\nabla_{\mathbf{Z}}J(\mathbf{Z}_2))\|_2\\
&\overset{(a)}{=}\left\|\left[\int_0^1 D_{\mathbf{Z}}(\nabla_{\mathbf{Z}}J(\mathbf{Z}_2+\xi(\mathbf{Z}_1-\mathbf{Z}_2)))\,d\xi\right]\right.\\
&\left.\cdot(\operatorname{vec}(\mathbf{Z}_1)-\operatorname{vec}(\mathbf{Z}_2))\right\|_2\\
&\le\underbrace{\left\|\int_0^1 D_{\mathbf{Z}}(\nabla_{\mathbf{Z}}J(\mathbf{Z}_2+\xi(\mathbf{Z}_1-\mathbf{Z}_2)))\,d\xi\right\|_2}_{\text{spectral norm}}\\
&\cdot\|\operatorname{vec}(\mathbf{Z}_1)-\operatorname{vec}(\mathbf{Z}_2)\|_2\\
&\le\left[\int_0^1\|D_{\mathbf{Z}}(\nabla_{\mathbf{Z}}J(\mathbf{Z}_2+\xi(\mathbf{Z}_1-\mathbf{Z}_2)))\|_2 d\xi\right]\\
&\cdot\|\mathbf{Z}_1-\mathbf{Z}_2\|_F,
\end{aligned}\tag{29}$$

where (a) is due to the mean value theorem, $D_{\mathbf{Z}}(\nabla_{\mathbf{Z}}J(\mathbf{Z}))$ stands for the Jacobian matrix of $\nabla_{\mathbf{Z}}J(\mathbf{Z})$ with respect to $\mathbf{Z}$ and $0<\xi<1$. Then we must compute the Jacobian matrix:

$$\begin{aligned}
&D_{\mathbf{Z}}(\nabla_{\mathbf{Z}}J(\mathbf{Z}))\\
&=\sum_{i=1}^m\sum_{j=1}^n\boldsymbol{\Omega}_{ij}\left\{f''\left(\widetilde{\mathbf{M}}_{ij}-\mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j}\right)\right.\\
&\cdot\left[\left(\left(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T+\mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\right)\otimes\mathbf{I}\right)\operatorname{vec}(\mathbf{Z})\right]\\
&\cdot\left[\left(\left(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T+\mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\right)\otimes\mathbf{I}\right)\operatorname{vec}(\mathbf{Z})\right]^T\\
&-\left[f'\left(\widetilde{\mathbf{M}}_{ij}-\mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j}\right)\right.\\
&\left.\left.\cdot\left(\left(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T+\mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\right)\otimes\mathbf{I}\right)\right]\right\}+2\gamma\mathbf{I},
\end{aligned}\tag{30}$$

and its spectral norm has an upper bound [cf. (31) at the bottom of the next page], where (a) is because $f''(x)$ and $f'(x)$ are bounded; (b) is because $\|\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T+\mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\|_2\le\|\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T+\mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\|_F=\sqrt{2}$, $\|(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T+\mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T)\otimes\mathbf{I}\|_2\le\|(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T+\mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T)\otimes\mathbf{I}\|_F=\sqrt{2l}$ (l is the rank upper bound from rough estimation); and (c) is because $\|\mathbf{Z}\|_F^2=\|\mathbf{X}\|_F^2+\|\mathbf{Y}\|_F^2=\frac{1}{\gamma}(J(\mathbf{X},\mathbf{Y})-\sum_{i=1}^m\sum_{j=1}^n\boldsymbol{\Omega}_{ij}f(\widetilde{\mathbf{M}}_{ij}-\mathbf{x}_i^T\mathbf{y}_j))\le\frac{1}{\gamma}J(\mathbf{X}^{(0)},\mathbf{Y}^{(0)})$ in that $f(x)\ge0$ and the assumption $J(\mathbf{X},\mathbf{Y})\le J(\mathbf{X}^{(0)},\mathbf{Y}^{(0)})$. Therefore, $\|\nabla_{\mathbf{Z}}J(\mathbf{Z}_1)-\nabla_{\mathbf{Z}}J(\mathbf{Z}_2)\|_F\le[\int_0^1\|D_{\mathbf{Z}}(\nabla_{\mathbf{Z}}J(\mathbf{Z}_2+\xi(\mathbf{Z}_1-\mathbf{Z}_2)))\|_2 d\xi]\cdot\|\mathbf{Z}_1-\mathbf{Z}_2\|_F\le L\|\mathbf{Z}_1-\mathbf{Z}_2\|_F$ is obtained. ■

We already know that $J(\mathbf{X}^{(1)},\mathbf{Y}^{(1)}) \le J(\mathbf{X}^{(0)},\mathbf{Y}^{(0)})$. Now we look into two possibilities: $J(\mathbf{X}^{(1)},\mathbf{Y}^{(1)}) = J(\mathbf{X}^{(0)},\mathbf{Y}^{(0)})$ and $J(\mathbf{X}^{(1)},\mathbf{Y}^{(1)}) < J(\mathbf{X}^{(0)},\mathbf{Y}^{(0)})$. When $J(\mathbf{X}^{(1)},\mathbf{Y}^{(1)}) = J(\mathbf{X}^{(0)},\mathbf{Y}^{(0)})$, it means $\inf_{\alpha\in\mathbb{R}}\{P_4^{(0)}\alpha^4 + P_3^{(0)}\alpha^3 + P_2^{(0)}\alpha^2 + P_1^{(0)}\alpha\} = 0$ and $\alpha^{(0)} = 0$ must be among the global minimizers, which indicates $\alpha^{(0)} = 0$ satisfies the zero derivative property, giving $P_1^{(0)} = 0$. Recalling the expression of $P_1^{(0)}$ in (21) and following Lemma 5, we can deduce $P_1^{(0)} \le -2\gamma\|[\widehat{\mathbf{X}}^{(0)},\widehat{\mathbf{Y}}^{(0)}] - [\mathbf{X}^{(0)},\mathbf{Y}^{(0)}]\|_F^2$, which indicates $0 = P_1^{(0)} \le -2\gamma\|[\widehat{\mathbf{X}}^{(0)},\widehat{\mathbf{Y}}^{(0)}] - [\mathbf{X}^{(0)},\mathbf{Y}^{(0)}]\|_F^2 \le 0$, resulting in $[\widehat{\mathbf{X}}^{(0)},\widehat{\mathbf{Y}}^{(0)}] = [\mathbf{X}^{(0)},\mathbf{Y}^{(0)}]$. Putting it back to the minimum principle mentioned in Lemma 5, we have for $\forall[\mathbf{X},\mathbf{Y}]$, $\mathrm{Tr}(\nabla^T_{[\mathbf{X},\mathbf{Y}]}J(\mathbf{X}^{(0)},\mathbf{Y}^{(0)})\cdot([\mathbf{X},\mathbf{Y}] - [\mathbf{X}^{(0)},\mathbf{Y}^{(0)}])) \ge 0$, indicating $[\mathbf{X}^{(0)},\mathbf{Y}^{(0)}]$ is a stationary solution already and no extra iteration is needed. We should note that because $[\widehat{\mathbf{X}}^{(0)},\widehat{\mathbf{Y}}^{(0)}] = [\mathbf{X}^{(0)},\mathbf{Y}^{(0)}]$, $P_4^{(0)} = P_3^{(0)} = P_2^{(0)} = P_1^{(0)} = 0$ [cf. (21)], implying that $\alpha^{(0)}$ can be any real value. This will not cause any trouble because $[\mathbf{X}^{(0)},\mathbf{Y}^{(0)}]$ will not drift away with a zero-valued descent direction. Also note, this argument for zero-valued descent direction (thus stationary solution) actually holds for any $k \ge 0$.

When $J\left(\mathbf{X}^{(1)},\mathbf{Y}^{(1)}\right) < J\left(\mathbf{X}^{(0)},\mathbf{Y}^{(0)}\right)$, we provide the following argument [cf. (32)] for $k \ge 1$

$$
\begin{aligned}
& J\left(\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)}\right) \\
&\overset{(a)}{\le} \bar{J}\left(\mathbf{X}^{(k)} + \alpha^{(k)}\left(\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}\right), \mathbf{Y}^{(k)}\right. \\
&\quad \left. + \alpha^{(k)}\left(\widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)}\right); \mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) \\
&= \min_{\alpha\in\mathbb{R}} \bar{J}\left(\mathbf{X}^{(k)} + \alpha\left(\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}\right), \mathbf{Y}^{(k)}\right. \\
&\quad \left. + \alpha\left(\widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)}\right); \mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) \\
&\overset{(b)}{\le} \min_{\alpha\in\mathcal{S}_\alpha^{(k)}} \bar{J}\left(\mathbf{X}^{(k)} + \alpha\left(\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}\right), \mathbf{Y}^{(k)}\right. \\
&\quad \left. + \alpha\left(\widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)}\right); \mathbf{X}^{(k)}, \mathbf{Y}^{(k)}\right) \\
&\overset{(c)}{\le} \min_{\alpha\in\mathcal{S}_\alpha^{(k)}} \left\{\bar{J}\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right) + \alpha\mathrm{Tr}\left(\nabla^T_{[\mathbf{X},\mathbf{Y}]}\bar{J}\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)\right.\right. \\
&\quad \left.\cdot\left(\left[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]\right)\right) \\
&\quad \left. + \frac{1}{2}L\alpha^2\left\|\left[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]\right\|_F^2\right\} \\
&\overset{(d)}{=} \min_{\alpha\in\mathcal{S}_\alpha^{(k)}} \left\{J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right) + \alpha\mathrm{Tr}\left(\nabla^T_{[\mathbf{X},\mathbf{Y}]}J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)\right.\right. \\
&\quad \left.\cdot\left(\left[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]\right)\right) \\
&\quad \left. + \frac{1}{2}L\alpha^2\left\|\left[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]\right\|_F^2\right\} \\
&\overset{(e)}{\le} \min_{\alpha\in\mathcal{S}_\alpha^{(k)}} \left\{J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right) - \left(2\alpha\gamma - \frac{1}{2}L\alpha^2\right)\right. \\
&\quad \left.\cdot\left\|\left[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]\right\|_F^2\right\} \\
&= J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right) - \left[\max_{\alpha\in\mathcal{S}_\alpha^{(k)}}\left(2\alpha\gamma - \frac{1}{2}L\alpha^2\right)\right] \\
&\quad \cdot\left\|\left[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]\right\|_F^2
\end{aligned} \tag{32}
$$

$$
\begin{aligned}
& \left\|D_{\mathbf{Z}}\left(\nabla_{\mathbf{Z}}J\left(\mathbf{Z}\right)\right)\right\|_2 \\
&\le \sum_{i=1}^{m}\sum_{j=1}^{n}\boldsymbol{\Omega}_{ij}\left\{\left|f''\left(\widetilde{\mathbf{M}}_{ij} - \mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j}\right)\right|\left\|\left(\left(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T + \mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\right)\otimes\mathbf{I}\right)\mathrm{vec}\left(\mathbf{Z}\right)\right\|_2^2\right. \\
&\quad \left. + \left|f'\left(\widetilde{\mathbf{M}}_{ij} - \mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j}\right)\right|\left\|\left(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T + \mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\right)\otimes\mathbf{I}\right\|_2\right\} + 2\gamma \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n}\boldsymbol{\Omega}_{ij}\left\{\left|f''\left(\widetilde{\mathbf{M}}_{ij} - \mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j}\right)\right|\left\|\mathbf{Z}\left(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T + \mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\right)\right\|_F^2\right. \\
&\quad \left. + \left|f'\left(\widetilde{\mathbf{M}}_{ij} - \mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j}\right)\right|\left\|\left(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T + \mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\right)\otimes\mathbf{I}\right\|_2\right\} + 2\gamma \\
&\le \sum_{i=1}^{m}\sum_{j=1}^{n}\boldsymbol{\Omega}_{ij}\left\{\underbrace{\left|f''\left(\widetilde{\mathbf{M}}_{ij} - \mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j}\right)\right|}_{(a)\text{ bounded}}\underbrace{\left\|\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T + \mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\right\|_2}_{(b)\text{ bounded}}\underbrace{\|\mathbf{Z}\|_F^2}_{(c)\text{ bounded}}\right. \\
&\quad \left. + \underbrace{\left|f'\left(\widetilde{\mathbf{M}}_{ij} - \mathbf{e}_{\mathbf{x}_i}^T\mathbf{Z}^T\mathbf{Z}\mathbf{e}_{\mathbf{y}_j}\right)\right|}_{(a)\text{ bounded}}\underbrace{\left\|\left(\mathbf{e}_{\mathbf{y}_j}\mathbf{e}_{\mathbf{x}_i}^T + \mathbf{e}_{\mathbf{x}_i}\mathbf{e}_{\mathbf{y}_j}^T\right)\otimes\mathbf{I}\right\|_2}_{(b)\text{ bounded}}\right\} + 2\gamma \le L
\end{aligned} \tag{31}
$$

where (a) $J(\mathbf{X},\mathbf{Y}) \le \bar{J}\left(\mathbf{X},\mathbf{Y};\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)$; (b) we denote $\mathcal{S}_\alpha^{(k)} \triangleq \{\alpha | [\mathbf{X}^{(k)} + \alpha(\widehat{\mathbf{X}}^{(k)} - \mathbf{X}^{(k)}), \mathbf{Y}^{(k)} + \alpha(\widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^{(k)})] \in \mathcal{S}\}$ for $k \ge 0$; (c) with supporting Lemma 8, descent lemma in [36] is applied; (d) $J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right) = \bar{J}\left(\mathbf{X},\mathbf{Y};\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)$ and $\nabla_{[\mathbf{X},\mathbf{Y}]}\bar{J}\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right) = \nabla_{[\mathbf{X},\mathbf{Y}]}J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)$; and (e) supporting Lemma 5 is applied. Combining the conclusion in Step 1 and the assumption for this case, we get $J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right) \le J\left(\mathbf{X}^{(1)},\mathbf{Y}^{(1)}\right) < J\left(\mathbf{X}^{(0)},\mathbf{Y}^{(0)}\right)$, which tells us that there exists some α_0 such that $\alpha_0 > 0$ and $\alpha_0 \in \bigcap_{k=1}^{+\infty}\mathcal{S}_\alpha^{(k)} \neq \emptyset$. Moreover, we denote α_ϵ to be one element of the nonempty set $[\bigcap_{k=1}^{+\infty}\mathcal{S}_\alpha^{(k)}] \bigcap \left(0, \frac{4\gamma}{L}\right)$ (nonemptiness is because $[\bigcap_{k=1}^{+\infty}\mathcal{S}_\alpha^{(k)}] \bigcap \left(0, \frac{4\gamma}{L}\right) \supseteq \left(0, \min\left(\alpha_0, \frac{4\gamma}{L}\right)\right) \neq \emptyset$), and we get (continuing from (32))

$$\begin{aligned}
&J\left(\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)}\right)\\
&\le J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right) - \left[\max_{\alpha\in\mathcal{S}_\alpha^{(k)}}\left(2\alpha\gamma - \frac{1}{2}L\alpha^2\right)\right]\\
&\quad\cdot\left\|\left[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]\right\|_F^2\\
&\le J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right) - \left(2\alpha_\epsilon\gamma - \frac{1}{2}L\alpha_\epsilon^2\right)\\
&\quad\cdot\left\|\left[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}\right] - \left[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right]\right\|_F^2. \qquad (33)
\end{aligned}$$

We can choose $\eta = 2\alpha_\epsilon\gamma - \frac{1}{2}L\alpha_\epsilon^2$ because when $\alpha_\epsilon \in \left(0, \frac{4\gamma}{L}\right)$, $\eta = \alpha_\epsilon\left(2\gamma - \frac{1}{2}L\alpha_\epsilon\right) > 0$. Note that the positive scalar η can be very small, but it does not mean that it will cause slow convergence to our proposed algorithm; it is merely a worst-case guarantee. At this point, we have completed the proof of Proposition 6. ■

Part 2) Recall that the sequence $\left\{J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)\right\}$ converges, i.e., $\lim_{k\to+\infty}[J\left(\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)}\right) - J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)] = 0$, which means $\liminf_{k\to+\infty}[J\left(\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)}\right) - J(\mathbf{X}^{(k)},\mathbf{Y}^{(k)})] = 0$. Also, assuming $[\mathbf{X}^{(0)},\mathbf{Y}^{(0)}]$ is not a stationary solution, we recall Proposition 6: $J\left(\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)}\right) - J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right) \le -\eta\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2$ holds for both line search methods when k is large. Taking the limit inferior of both sides, we have $0 = \liminf_{k\to+\infty}[J\left(\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)}\right) - J\left(\mathbf{X}^{(k)},\mathbf{Y}^{(k)}\right)] \le \liminf_{k\to+\infty} -\eta\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2 \le -\eta\cdot\limsup_{k\to+\infty}\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2 \le 0$, so we can infer $\limsup_{k\to+\infty}\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2 = 0$. Because $0 \le \liminf_{k\to+\infty}\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2 \le \limsup_{k\to+\infty}\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2 = 0$, we can see $\liminf_{k\to+\infty}\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2 = 0$ and thus $\lim_{k\to+\infty}\|[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\|_F^2 = 0$. This indicates that the difference $[\widehat{\mathbf{X}}^{(k)},\widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]$ eventually vanishes as k goes to infinity, and $[\mathbf{X}^{(k+1)},\mathbf{Y}^{(k+1)}] = [\mathbf{X}^{(k)},\ \mathbf{Y}^{(k)}] + \alpha^{(k)}([\widehat{\mathbf{X}}^{(k)},\ \widehat{\mathbf{Y}}^{(k)}] - [\mathbf{X}^{(k)},\ \mathbf{Y}^{(k)}]) \to [\mathbf{X}^{(k)},\ \mathbf{Y}^{(k)}]$ holds when $k \to +\infty$, which means the sequence $\left\{[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\right\}$ also converges.

Now we still have to check whether $\{[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\}$ converges to a stationary point. We denote the limit point of the sequence $\{[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\}$ as $\{[\mathbf{X}^{(+\infty)},\mathbf{Y}^{(+\infty)}]\}$ and it has the property of $[\mathbf{X}^{(+\infty)},\mathbf{Y}^{(+\infty)}] = [\widehat{\mathbf{X}}^{(+\infty)},\widehat{\mathbf{Y}}^{(+\infty)}]$. Take one column of $[\mathbf{X}^{(+\infty)},\mathbf{Y}^{(+\infty)}]$, e.g., $\mathbf{x}_i^{(+\infty)}$, where $i \in \{1,2,\ldots,m\}$, for analysis. Recall the minimum principle: for all $\mathbf{x}_i$,

$$\begin{aligned}
0 &\le \left(\mathbf{H}_{\mathrm{x},i}^{(+\infty)}\left(\widehat{\mathbf{x}}_i^{(+\infty)} - \mathbf{x}_i^{(+\infty)}\right)\right.\\
&\quad\left. + \mathbf{g}_{\mathrm{x},i}^{(+\infty)} + 2\gamma\widehat{\mathbf{x}}_i^{(+\infty)}\right)^T\cdot\left(\mathbf{x}_i - \widehat{\mathbf{x}}_i^{(+\infty)}\right)\\
&= \left(\mathbf{H}_{\mathrm{x},i}^{(+\infty)}\left(\mathbf{x}_i^{(+\infty)} - \mathbf{x}_i^{(+\infty)}\right)\right.\\
&\quad\left. + \mathbf{g}_{\mathrm{x},i}^{(+\infty)} + 2\gamma\mathbf{x}_i^{(+\infty)}\right)^T\left(\mathbf{x}_i - \mathbf{x}_i^{(+\infty)}\right)\\
&= \nabla_{\mathbf{x}_i}^T J\left(\mathbf{X}^{(+\infty)},\mathbf{Y}^{(+\infty)}\right)\cdot\left(\mathbf{x}_i - \mathbf{x}_i^{(+\infty)}\right). \qquad (34)
\end{aligned}$$

Similarly, $\nabla_{\mathbf{y}_j}^T J\left(\mathbf{X}^{(+\infty)},\mathbf{Y}^{(+\infty)}\right)\cdot(\mathbf{y}_j - \mathbf{y}_j^{(+\infty)}) \ge 0, \quad \forall j$. Combining the summation over i and j, we obtain $\mathrm{Tr}(\nabla_{[\mathbf{X},\mathbf{Y}]}^T J(\mathbf{X}^{(+\infty)},\mathbf{Y}^{(+\infty)})([\mathbf{X},\mathbf{Y}] - [\mathbf{X}^{(+\infty)},\mathbf{Y}^{(+\infty)}])) \ge 0$, and therefore the limit point of the sequence $\{[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\}$ is a stationary point. Also note that $\{J(\mathbf{X}^{(k)},\mathbf{Y}^{(k)})\}$ is a nonincreasing sequence, which entails that no limit point of $\{[\mathbf{X}^{(k)},\mathbf{Y}^{(k)}]\}$ can be a local maximum. The whole proof is thus completed. ■

REFERENCES

[1] L. Zhao, P. Babu, and D. P. Palomar, "Robust low-rank optimization for large scale problems," in *Proc. 49th IEEE Asilomar Conf. Signals, Syst., Comput.*, 2015, pp. 391–395.

[2] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *J. ACM (JACM)*, vol. 58, no. 3, pp. 11, 2011.

[3] Y. Koren, "The Bellkor solution to the Netflix grand prize," *Netflix Prize Documentation*, vol. 81, 2009.

[4] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Largescale parallel collaborative filtering for the Netflix prize," *Algorithmic Aspects in Information and Management*. New York, NY, USA: Springer, 2008, pp. 337–348.

[5] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psychol.*, vol. 24, no. 6, pp. 417, 1933.

[6] I. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Wiley Online Library, 2002.

[7] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometr. Intell. Lab. Syst.*, vol. 2, no. 1, pp. 37–52, 1987.

[8] B. Recht and C. Ré, "Parallel stochastic gradient algorithms for large-scale matrix completion," *Math. Programm. Comput.*, vol. 5, no. 2, pp. 201–226, 2013.

[9] M. Udell, C. Horn, R. Zadeh, and S. Boyd, "Generalized low rank models," 2014, arXiv preprint arXiv:1410.0342.

[10] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Trans. Signal Process.*, vol. 63, no. 10, pp. 2663–2677, May 2015.

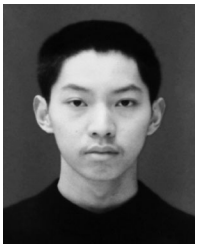
[11] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted lowrank matrices via convex optimization," *Adv. Neural Inf. Process. Syst.*, pp. 2080–2088, 2009.

[12] H. Xu, C. Caramanis, and S. Sanghavi, "Robust PCA via outlier pursuit," *Adv. Neural Inf. Process. Syst.*, pp. 2496–2504, 2010.

[13] G. Mateos and G. B. Giannakis, "Robust PCA as bilinear decomposition with outlier-sparsity regularization," *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5176–5190, Oct. 2012.

[14] J. Feng, H. Xu, and S. Yan, "Online robust PCA via stochastic optimization," *Adv. Neural Inf. Process. Syst.*, pp. 404–412, 2013.

[15] T. Bouwmans and E. H. Zahzah, "Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance," *Comput. Vis. Image Understand.*, vol. 122, pp. 22–34, 2014.
[16] X. Luan, B. Fang, L. Liu, W. Yang, and J. Qian, "Extracting sparse error of robust PCA for face recognition in the presence of varying illumination and occlusion," *Pattern Recognit.*, vol. 47, no. 2, pp. 495–508, 2014.
[17] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pac. J. Optim.*, vol. 6, no. 615–640, pp. 15, 2010.
[18] X. Yuan and J. Yang, "Sparse and low-rank matrix decomposition via alternating direction methods," 2009 preprint, [Online]. Available: https://www.semanticscholar.org/paper/Sparse-and-Low-rank-Matrix-Decomposition-via-Yuan-Yang/7431b21b56f320080fd6cd0d4d1df233de583ceb/pdf
[19] Z. Lin, M. Chen, and Y. Ma, "The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices," 2010, arXiv Preprint arXiv:1009.5055.
[20] F. Bach, J. Mairal, and J. Ponce, "Convex sparse matrix factorizations," 2008, arXiv Preprint arXiv:0812.1869.
[21] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *J. Roy. Statist. Soc. B (Statist. Methodol.)*, vol. 61, no. 3, pp. 611–622, 1999.
[22] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Comput.*, vol. 11, no. 2, pp. 443–482, 1999.
[23] G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang, "Decomposition by partial linearization: Parallel optimization of multi-agent systems," *IEEE Trans. Signal Process.*, vol. 62, no. 3, pp. 641–656, Feb. 2014.
[24] A. Aravkin, M. P. Friedlander, F. J. Herrmann, and T. Van Leeuwen, "Robust inversion, dimensionality reduction, and randomized sampling," *Math. Programm.*, vol. 134, no. 1, pp. 101–125, 2012.
[25] K. L. Lange, R. J. Little, and J. M. Taylor, "Robust statistical modeling using the t distribution," *J. Amer. Statist. Assoc.*, vol. 84, no. 408, pp. 881–896, 1989.
[26] D. E. Tyler, "A distribution-free M-estimator of multivariate scatter," *Ann. Statist.*, pp. 234–251, 1987.
[27] E. Ollila and V. Koivunen, "Robust antenna array processing using M-estimators of pseudo-covariance," in *Proc. 14th IEEE Personal, Indoor, Mobile Radio Communi. (PIMRC)*, 2003, vol. 3, pp. 2659–2663.
[28] Y. Sun, P. Babu, and D. P. Palomar, "Regularized robust estimation of mean and covariance matrix under heavy-tailed distributions," *IEEE Trans. Signal Process.*, vol. 63, no. 12, pp. 3096–3109, Jun. 2015.
[29] P. J. Huber, *Robust Statistics*. Springer, 2011.
[30] K. Fountoulakis and J. Gondzio, "A second-order method for strongly convex ℓ_1-regularization problems," *Math. Programm.*, pp. 1–31, 2013.
[31] D. R. Hunter and K. Lange, "A tutorial on MM algorithms," *Amer. Statistician*, vol. 58, no. 1, pp. 30–37, 2004.
[32] R. Sun and Z.-Q. Luo, "Guaranteed matrix completion via nonconvex factorization," 2014, arXiv Preprint arXiv:1411.8003.
[33] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proc. ACM Conf. Comput. Supported Cooperative Workshop*, 1994, pp. 175–186.
[34] J. He, L. Balzano, and J. Lui, "Online robust subspace tracking from partial information," 2011, arXiv Preprint arXiv:1109.3827.
[35] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Proc. 48th Annu. IEEE Allerton Conf. Commun., Control, Comput. (Allerton)*, 2010, pp. 704–711.
[36] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Scientific, 1999.



Licheng Zhao received the B.S. degree in information engineering from Southeast University (SEU), Nanjing, China, in 2014. He is currently pursuing the Ph.D. degree with the Department of Electronic and Computer Engineering at the Hong Kong University of Science and Technology (HKUST). His research interests are in optimization theory and fast algorithms, with applications in signal processing, machine learning, and financial engineering.

Prabhu Babu received the Ph.D. degree in electrical engineering from the Uppsala University, Sweden, in 2012. From 2013–2016, he was a post-doctorial fellow with the Hong Kong University of Science and Technology. He is currently with the Center for Applied Research in Electronics (CARE), Indian Institute of Technology Delhi.

Daniel P. Palomar (S'99–M'03–SM'08–F'12) received the Electrical Engineering and Ph.D. degrees from the Technical University of Catalonia (UPC), Barcelona, Spain, in 1998 and 2003, respectively.

He is a Professor in the Department of Electronic and Computer Engineering at the Hong Kong University of Science and Technology (HKUST), Hong Kong, which he joined in 2006. Since 2013 he is a Fellow of the Institute for Advance Study (IAS) at HKUST. He had previously held several research appointments, namely, at King's College London (KCL), London, UK; Stanford University, Stanford, CA; Telecommunications Technological Center of Catalonia (CTTC), Barcelona, Spain; Royal Institute of Technology (KTH), Stockholm, Sweden; University of Rome "La Sapienza", Rome, Italy; and Princeton University, Princeton, NJ. His current research interests include applications of convex optimization theory, game theory, and variational inequality theory to financial systems, big data systems, and communication systems.

Dr. Palomar is an IEEE Fellow, a recipient of a 2004/06 Fulbright Research Fellowship, the 2004 and 2015 (co-author) Young Author Best Paper Awards by the IEEE Signal Processing Society, the 2015–2016 HKUST Excellence Research Award, the 2002/03 best Ph.D. prize in Information Technologies and Communications by the Technical University of Catalonia (UPC), the 2002/03 Rosina Ribalta first prize for the Best Doctoral Thesis in Information Technologies and Communications by the Epson Foundation, and the 2004 prize for the best Doctoral Thesis in Advanced Mobile Communications by the Vodafone Foundation and COIT.

He is a Guest Editor of the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING 2016 Special Issue on "Financial Signal Processing and Machine Learning for Electronic Trading" and has been Associate Editor of IEEE TRANSACTIONS ON INFORMATION THEORY and of IEEE TRANSACTIONS ON SIGNAL PROCESSING, a Guest Editor of the IEEE SIGNAL PROCESSING MAGAZINE 2010 Special Issue on "Convex Optimization for Signal Processing", the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS 2008 Special Issue on "Game Theory in Communication Systems", and the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS 2007 Special Issue on "Optimization of MIMO Transceivers for Realistic Communication Networks".