Optimization and Engineering
https://doi.org/10.1007/s11081-019-09476-9

RESEARCH ARTICLE

Robust principal component analysis using facial reduction

Shiqian Ma[1] · Fei Wang[2] · Linchuan Wei[3] · Henry Wolkowicz[3]

Received: 8 January 2019 / Revised: 19 November 2019 / Accepted: 19 November 2019

Abstract
We introduce a novel approach for robust principal component analysis (RPCA) for a partially observed data matrix. The aim is to recover the data matrix as a sum of a low-rank matrix and a sparse matrix so as to eliminate erratic noise (outliers). This problem is known to be NP-hard in general. A classical approach to solving RPCA is to consider convex relaxations. One such heuristic involves the minimization of the (weighted) sum of a nuclear norm part, that promotes a low-rank component, with an ℓ_1 norm part, to promote a sparse component. This results in a well-structured convex problem that can be efficiently solved by modern first-order methods. However, first-order methods often yield low accuracy solutions. Moreover, the heuristic of using a norm consisting of a weighted sum of norms may lose some of the advantages that each norm had when used separately. In this paper, we propose a novel *nonconvex* and *nonsmooth* reformulation of the original NP-hard RPCA model. The new model adds a redundant semidefinite cone constraint and solves *small* subproblems using a *PALM* algorithm. Each subproblem results in an *exposing vector* for a facial reduction technique that is able to reduce the size significantly. This makes the problem amenable to efficient algorithms in order to obtain high-level accuracy. We include numerical results that confirm the efficacy of our approach.

Keywords Robust principal component analysis · Semidefinite cone · Facial reduction

Shiqian Ma: Research supported by a startup package from Department of Mathematics at UC Davis. Linchuan Wei: Research supported by the Undergraduate Student Research Awards Program, Natural Sciences and Engineering Research Council of Canada. Henry Wolkowicz: Research supported by The Natural Sciences and Engineering Research Council of Canada.

✉ Fei Wang
fewa@kth.se

[1] Department of Mathematics, University of California, Davis, CA 95616, USA

[2] Division of Optimization and System Theory, Department of Mathematics, Royal Institute of Technology, Stockholm, Sweden

[3] Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, ON N2L 3G1, Canada

Mathematics Subject Classification 90C26 · 65K10 · 90C27

1 Introduction

Principal component analysis (PCA) seeks a low-rank approximation to an input data matrix. It is arguably the most popular statistical tool in data analysis and is very effective for handling the effect of small random Gaussian noise. PCA can be accomplished easily via the singular value decomposition (SVD). However, it is also well-known that PCA lacks robustness with respect to large erratic noise, i.e., a single corrupted entry can result in an approximation that is far away from the true solution. Robust PCA has been proposed to remove the effect of sparse gross errors. In e.g., Candes et al. (2010), Chandrasekaran et al. (2009), Cheng et al. (2015), the intractable RPCA problem and its convex relaxation are introduced. It is shown that the two problems are equivalent, with high probability, under certain conditions of the input data. Specifically, RPCA aims to express a given data matrix $Z \in \mathbb{R}^{m\times n}$ as the sum $Z = L^* + S^*$, where L^* is the low-rank approximation to Z, and S^* is a sparse matrix that captures the additive erratic noise. Throughout this paper, we assume that $r = \operatorname{rank}(L^*) \ll \min(m, n)$, and $\mu > 0$ is a given positive parameter. RPCA is in a sense a multicriteria (two criteria) problem in that it attempts to find a low rank approximation as well a sparse approximation. Finding Pareto optimal points can be formulated as the following weighted optimization problem:

$$\begin{aligned} &\min \operatorname{rank}(L) + \mu\|S\|_0 \\ &\text{s.t. } L + S = Z, \end{aligned} \tag{1}$$

where the cardinality function $\|S\|_0$ counts the number of nonzeros of S. This problem is NP-hard due to the combinatorial nature of the rank and the cardinality functions. It is shown in e.g., Candes et al. (2010), Chandrasekaran et al. (2009), that under certain conditions (1) is equivalent, with high probability, to the following convex program:

$$\begin{aligned} &\min \|L\|_* + \mu\|S\|_1 \\ &\text{s.t. } L + S = Z. \end{aligned} \tag{2}$$

Here $\|L\|_*$ is the nuclear norm of L, the sum of the singular values of L, and the (vector) ℓ_1-norm is $\|S\|_1 := \sum_{ij} |S_{ij}|$. The convex program (2) is known as *robust principal component pursuit*, **RPCP**.

In practice, it is possible that the matrix Z is only partially observed. That is, there exists a subset $\hat{E}$ of the indices such that only entries Z_{ij} for $(i, j) \in \hat{E}$ are given. In this case, RPCA (1) and RPCP (2) need to be changed, respectively, to

$$(\mathbf{RPCA}) \qquad \begin{aligned} F_1(L, S) \; := \min \;& \operatorname{rank}(L) + \mu\|S\|_0 \\ \text{s.t.} \;& \mathcal{P}_{\hat{E}}(L + S) = z, \end{aligned} \tag{3}$$

and

$$\textbf{(RPCP)}\qquad \begin{array}{ll} \min & \|L\|_* + \mu\|S\|_1 \\ \text{s.t.} & \mathcal{P}_{\hat{E}}(L+S) = z, \end{array} \tag{4}$$

where $\mathcal{P}_{\hat{E}} : \mathbb{R}^{m\times n} \to \mathbb{R}^{\hat{E}}$ denotes the projection onto the components with indices in $\hat{E}$, and the data $z = \mathcal{P}_{\hat{E}}(Z)$.

The convex relaxations (2) and (4) can be reformulated as semidefinite programming problems (SDP), e.g., Recht et al. (2010), Candes et al. (2010). Thus, they can be efficiently solved by e.g., interior point methods. However, RPCA arising from real applications is usually of huge scale and interior point methods generally do not scale well. As a result, current research on algorithms for RPCA and RPCP is focused on first-order methods, see e.g., the recent survey papers (Bouwmans and Zahzah 2014; Aybat 2016; Ma and Aybat 2018). But, it is also known that the first-order methods cannot generally provide high accuracy solutions.

1.1 Main contributions

In this paper we propose a new approach to solve RPCA with partially observed data, (3). We use semidefinite programming and a novel cone facial reduction (FR) technique applied to a reformulation of (3). As a result, the size of the feasible region is significantly reduced and an efficient algorithm is used to obtain highly accurate solutions. This extends the approach in Huang and Wolkowicz (2018) for low-rank matrix completions where no sparse gross errors are considered. In addition, our facial reduction technique is applied on the non-convex model (3) directly, rather than within a nuclear norm mimimization model as done in Huang and Wolkowicz (2018). A key ingredient here is the use of the exposing vector approach for characterizing faces of the *semidefinite cone*, $\mathcal{S}^n_+$. see Drusvyatskiy and Wolkowicz (2017), Drusvyatskiy et al. (2017), Huang and Wolkowicz (2018). Here we develope a new technique to grow bicliques by submatrix completion. Our tests show that this new technique greatly increases the speed of the completion process and allows us to find more missing entries. In particular, we see that in many cases, and even under relatively low density for the sampled data, we get exact recovery without using any SDP solver. This provides a distinct *improvement* over simply using the convex relaxation approach, RPCP.

1.2 Connections and differences with previous work

Our work is an extension of the previous work in Huang and Wolkowicz (2018) that only considers facial reduction on the convex approximation of the rank minimization problem. In this paper, we show in Lemma 1 that the nonconex RPCA problem (3) is equivalent to the nonconvex SDP problem (5). And in Proposition 1 we get the exposing vectors for the optimal face of the nonconvex SDP problem, rather than the convex approximation in Huang and Wolkowicz (2018). Note that our goal is to solve the nonconvex problem in order to recover the *true* matrices. It is clear after facial reduction that the optimal solution lies in a *smaller* space. Hence the convex relaxation of the now smaller-size problem generally has a better chance to recover the true matrices than the convex relaxation of the original large-size problem. This is validated by the numerical tests in Sect. 6. Secondly, we extend the method in Huang and Wolkowicz (2018) to include the sparse component S and we show we can reduce the size of S using exposing vectors obtained from the low rank component L. Thirdly, we proposed a novel approach of growing bicliques in Sect. 5. The technique to grow bicliques is much more efficient when compared to the previous work in the literature.

1.2.1 Organization

Further background on RPCA is given in Sect. 2; this includes SDP reformulations and graph representations. In Sect. 3 we discuss how to obtain accurate solutions for small fully sampled submatrices. This is the key in obtaining high accurate exposing vectors for faces used in reducing the size of the complete problem, while still maintaining low error growth that could arise due to noise. Section 4 presents the main results used in the paper. This includes facial reduction, bicliques and exposing vectors, as well as the algorithm for solving the RPCA problem accurately. In Sect. 5 we discuss a heuristic on growing the size of bicliques using a submatrix approach. We present the numerical results in Sect. 6 for both noiseless and noisy data. We conclude in Sect. 7.

2 Background

In this paper we focus on the nonconvex and nonsmooth model **RPCA** in (3). Our approach starts with a reformulation using an additional semidefinite cone constraint.

2.1 Reformulating RPCA with SDP

In this section we show that (3) is equivalent to the following *nonconvex* and *nonsmooth* SDP problem:

$$\begin{aligned} F_2(Y,S) := \min_{Y,S} \ & \operatorname{rank}(Y) + \mu\|S\|_0 \\ \text{s.t.} \ & \mathcal{P}_{\hat{E}}(L+S) = z \\ & Y = \begin{bmatrix} W_1 & L \\ L^\top & W_2 \end{bmatrix} \succeq 0. \end{aligned} \tag{5}$$

Lemma 1 *Problems* (3) *and* (5) *are equivalent in the sense that they have the same optimal solution pair* (L^*, S^*) *and the same optimal objective value. In particular, the optimal* L^* *is a submatrix of the optimal* Y^*.

Proof Suppose (L^*, S^*) is the optimal solution of (3) with $\operatorname{rank}(L^*) = r < \min(m,n)$, and the compact SVD of L^* is given by $L^* = U\Sigma V^\top$, where both $U \in \mathbb{R}^{m\times r}$ and

$V \in \mathbb{R}^{n\times r}$ have orthonormal columns, and $\Sigma \in \mathbb{R}^{r\times r}_{++}$ is a diagonal matrix with the positive singular values of L^* on its diagonal. We now see that this is equivalent to (Y^*, S^*) is an optimal solution pair of (5) with

$$Y^* = \begin{bmatrix} U \\ V \end{bmatrix} \Sigma \begin{bmatrix} U \\ V \end{bmatrix}^\top = \begin{bmatrix} U\Sigma U^\top & U\Sigma V^\top \\ V\Sigma U^\top & V\Sigma V^\top \end{bmatrix} = \begin{bmatrix} U\Sigma U^\top & L^* \\ (L^*)^\top & V\Sigma V^\top \end{bmatrix}, \tag{6}$$

from which we get that $\operatorname{rank}(Y^*) = \operatorname{rank}(L^*) = r$, and L^* is the upper right corner block of Y^*. This shows that for an optimal Y^* we have $\operatorname{rank}(Y^*) = \operatorname{rank}(L^*) = r$; and it emphasizes that the only change in the two problems is the addition of the semidefinite constraint.

The equivalence follows from: if there exists a better solution $(\hat{Y}, \hat{S})$, with the spectral decomposition of $\hat{Y}$ given by

$$\hat{Y} = \begin{bmatrix} \hat{U} \\ \hat{V} \end{bmatrix} \hat{\Sigma} \begin{bmatrix} \hat{U} \\ \hat{V} \end{bmatrix}^\top = \begin{bmatrix} \hat{U}\hat{\Sigma}\hat{U}^\top & \hat{U}\hat{\Sigma}\hat{V}^\top \\ \hat{V}\hat{\Sigma}\hat{U}^\top & \hat{V}\hat{\Sigma}\hat{V}^\top \end{bmatrix},$$

then we have

$$F_1(\hat{U}\hat{\Sigma}\hat{V}^\top, \hat{S}) \le F_2(\hat{Y}, \hat{S}) < F_2(Y^*, S^*) = r + \mu\|S^*\|_0 = F_1(L^*, S^*),$$

where the first inequality is due to the fact $\operatorname{rank}(\hat{U}\hat{\Sigma}\hat{V}^\top) \le \operatorname{rank}(\hat{\Sigma}) = \operatorname{rank}(\hat{Y})$. This is a contradiction. □

Remark 1 Lemma 1 indicates that, in order to solve (3), we can solve (5) instead to obtain (Y^*, S^*). Then (L^*, S^*) solves (3), where L^* is obtained from the upper right corner block of Y^*. It appears that (5) is harder than (3) as it is much larger. However, the fact that the optimal L^* and Y^* are both of low rank enables us to reduce the size of (5) greatly to a new problem whose size is much smaller than (3). This is done by pursuing the facial structure of the semidefinite cone in (5). This is the main motivation for our approach.

2.2 Outline of our algorithm

We now present a brief outline of our algorithm based on the graph representation of the partially observed matrix Z and the facial reduction of the reformulation (5).

For a given partially observed data matrix $Z \in \mathbb{R}^{m\times n}$, we aim to find the low-rank-plus-sparse decomposition $Z = L^* + S^*$ with $r = \operatorname{rank}(L^*) \ll \min(m, n)$. The main steps of our algorithm are as follows:

(i) Associate a bipartite graph to Z using the observed entries, and find a biclique associated to a completely determined submatrix of Z. Denote this submatrix as $\bar{Z} \in \mathbb{R}^{p\times q}$, where p, q are *generally much smaller* than m, n, respectively.
(ii) Find the low-rank-plus-sparse decomposition for $\bar{Z}$, i.e., $\bar{Z} = \bar{L} + \bar{S}$. Note that since $p \ll m, q \ll n$, this problem is much easier than the original problem.

(iii) From $\bar{L}$ we are able to find an exposing vector for a face of the SDP cone that contains an optimal Y^* that solves (5), i.e., an exposing vector of the optimal face. Note that $\bar{L}$ is a submatrix of L^* and thus a submatrix of Y^*.

(iv) We use the fact that the sum of exposing vectors of faces is an exposing vector of the intersection of the faces, see (9) below. We can then reformulate (5) into a new problem whose size is much smaller than (3). This new problem can then be solved efficiently and accurately.

2.3 Graph representation of a partially observed matrix *Z*

We associate a bipartite graph $G_Z((U_m, V_n), \hat{E})$ to Z, whose node set corresponds to the union of the two sets of rows and columns of Z

$$U_m = \{1, \dots, m\}, \; V_n = \{1, \dots, n\},$$

and there is an edge $(i,j) \in \hat{E}$, with $i \in U_m$ and $j \in V_n$ if Z_{ij} is observed. Note that a biclique of G_Z corresponds to a submatrix of Z whose entries are all observed. To find a biclique of $G_Z((U_m, V_n), \hat{E})$, we can relate it to finding cliques in the graph $G = (V, E)$ whose node set is $V = \{1, \dots, m, m+1, \dots, m+n\}$ and the edge set E is

$$\begin{aligned} E := \{(i,j) \in V \times V : (i, j-m) \in \hat{E}\} \cup \{(i,j) \in V \times V : 1 \le i < j \le m\} \\ \cup \{(i,j) \in V \times V : m+1 \le i < j \le m+n\}. \end{aligned}$$

Suppose we find a non-trivial clique of G denoted by $C = \{i_1, \dots, i_k\}$ whose cardinality $k = p + q$ satisfies

$$|C \cap \{1, \dots, m\}| = p \neq 0, \quad |C \cap \{m+1, \dots, m+n\}| = q \neq 0.$$

By removing the edges in C that have both nodes in $\{1, \dots, m\}$ or $\{m+1, \dots, m+n\}$, we obtain a biclique of G_Z. We use $\bar{C}$ to denote this biclique. This biclique then corresponds to a submatrix of Z, with entries corresponding to edges in $\bar{C}$ being kept and other entries of Z are removed. We use $\bar{Z}$ to denote this matrix whose size is $p \times q$. Note that $\bar{Z}$ is a submatrix of Z, and all entries of $\bar{Z}$ are observed. Moreover, we generally maintain the size of $\bar{Z}$ much smaller than the size of Z, $p \ll m$ and $q \ll n$.

2.4 Heuristics for finding cliques

Finding all the cliques from a graph is a NP-hard problem, (Blair and Peyton 1993); the clique decision problem is one of Karp's 21 NP-complete problems, (Karp 1972). Moreover, the cost of finding the SVD needed for each submatrix corresponding to each clique found becomes expensive for large cliques and large submatrices. Therefore, we use a heuristic algorithm that is proposed in Krislock and Wolkowicz (2010) and Drusvyatskiy et al. (2017, Algorithm 2) to efficiently find many small cliques. This heuristic algorithm is briefly outlined in Algorithm 1.

Algorithm 1 Heuristics for finding a set of cliques

```
1: Input: A Graph $G = (V, E)$, maximum clique size $K_{max}$, minimum clique size $K_{min}$
2: Initialization: Set the clique set $\Theta$ as an empty set
3: for each node $v \in G$ do
4:     Initialize $\alpha \leftarrow \{v\}$
5:     while $|\alpha| < K_{max}$ and there is a $w$ that is connected to every node in $\alpha$ do
6:         $\alpha \leftarrow \alpha \cup \{w\}$
7:     end while
8:     if $|\alpha| \geq K_{min}$ then $\Theta \leftarrow \Theta \cup \{\alpha\}$
9:     end if
10: end for
11: Output: The clique set $\Theta$
```

Algorithm 1 is very efficient in practice because it only goes through each node in the graph G once. In total, we need to add at most $K_{max}|V|$ nodes which corresponds to $K_{max}|V|$ row operations on the adjacency matrix of the graph. Since it is a heuristic, it is not guaranteed that all the cliques are found. However, if we need more cliques, we could apply a random permutation to V and use Algorithm 1 again.

3 Decomposing the submatrix $\bar{Z}$ using PALM

From Sects. 2.3 and 2.4 we know that by finding a biclique of G_Z, we get a submatrix of Z, denoted by $\bar{Z} \in \mathbb{R}^{p \times q}$ with $K_{min} \leq p + q \leq K_{max}$, whose entries are all known (sampled). Now we want to find a low-rank-plus-sparse decomposition of $\bar{Z}$. This problem can be formulated as

$$\begin{aligned} &\min_{\bar{L},\bar{S}} \frac{1}{2}\|\bar{L} + \bar{S} - \bar{Z}\|_F^2 \\ &\text{s.t. } \operatorname{rank}(\bar{L}) \leq \bar{r}, \quad \|\bar{S}\|_0 \leq \bar{s}, \end{aligned} \tag{7}$$

where $\bar{r}$ and $\bar{s}$ are given parameters to control the rank of $\bar{L}$ and sparsity of $\bar{S}$, respectively. There are two reasons that (7) is much easier to solve than (3). The first reason is that all entries of $\bar{Z}$ are known, while those in Z are only partially known. The second reason is that the size of $\bar{Z}$ is much smaller than the size of Z. We adopt the proximal alternating linearization method (PALM) (Bolte et al. 2014) to solve (7). This is summarized in Algorithm 2.

Algorithm 2 PALM for Solving (7)

```
1: INPUT: $\bar{Z} \in \mathbb{R}^{p\times q}$, $\bar{r}$, $\bar{s}$, initial matrices $\bar{L}^0, \bar{S}^0$, step sizes $\gamma_1 > 1$, $\gamma_2 > 1$.
2: for $k = 0, 1, \ldots,$ do
3:     $G_L^k := \bar{L}^k - \frac{1}{\gamma_1}(\bar{L}^k + \bar{S}^k - \bar{Z})$.
4:     $\bar{L}^{k+1} := \mathrm{argmin}_{\bar{L}\in\mathbb{R}^{p\times q}}\{\|\bar{L} - G_L^k\|_F^2 : \mathrm{rank}(\bar{L}) \le \bar{r}\}$.
5:     $G_S^k := \bar{S}^k - \frac{1}{\gamma_2}(\bar{L}^{k+1} + \bar{S}^k - \bar{Z})$.
6:     $\bar{S}^{k+1} := \mathrm{argmin}_{\bar{S}\in\mathbb{R}^{p\times q}}\{\|\bar{S} - G_S^k\|_F^2 : \|\bar{S}\|_0 \le \bar{s}\}$.
7: end for
8: Set $\bar{L} := \bar{L}^k$ and $\bar{S} := \bar{S}^k$.
9: OUTPUT: Low-rank component $\bar{L}$ and sparse component $\bar{S}$.
```

By introducing indicator functions, (7) can be equivalently written as

$$\min_{\bar{L},\bar{S}} \Psi(\bar{L},\bar{S}) := \frac{1}{2}\|\bar{L}+\bar{S}-\bar{Z}\|_F^2 + \mathbb{1}(\bar{L} \mid \mathrm{rank}(\bar{L}) \le \bar{r}) + \mathbb{1}(\bar{S} \mid \|\bar{S}\|_0 \le \bar{s}), \quad (8)$$

where the indicator function $\mathbb{1}(X \mid \mathcal{X})$ equals 0 if $X \in \mathcal{X}$, and equals $+\infty$ otherwise. It is easy to verify that the objective function $\Psi(\bar{L},\bar{S})$ satisfies the so-called Kurdyka-ˇojasiewicz property (Kurdyka 1998; Łojasiewicz 1963; Bolte et al. 2014). As a result, we have the following convergence result for Algorithm 2 that follows directly from Bolte et al. (2014, Theorem 1).

Theorem 1 *The sequence $\{\bar{L}^k,\bar{S}^k\}_{k\in\mathbb{N}}$ generated by Algorithm 2 converges to a critical point of problem* (7).

Suppose we solve (7) and obtain the optimal value zero that guarantees a global optimum. To ensure the correct $\bar{L}$ is recovered, we need this global optimum to be unique. This happens with high probability when the rank of $\bar{L}$ is much smaller than the size of the matrix, and the incoherence conditions hold, see e.g., Candes et al. (2010), Chandrasekaran et al. (2009). Therefore, we make the following assumption throughout the remainder of the paper.

Assumption We assume that $\bar{r} \ll \min(p,q)$ and, in addition, that (7) has a unique global optimal solution $\bar{L}$ that is a submatrix of L^* and has the same rank as L^*.

The uniqueness also happens with higher probability when the sparsity parameter $\bar{s}$ is smaller. Let D_S denote the density of the sparse matrix S^*. We vary $\bar{s}$ from 1 to $\max\{1, pqD_S\}$ to control the sparsity in problem (7). We also set the target rank $\bar{r} = r$. Thus we have the following algorithm:

Algorithm 3 Decompose $\bar{Z}$

```
1: INPUT: $\bar{Z} \in \mathbb{R}^{p\times q}$, target rank $r$ such that $r < \min\{p,q\}$, density $D_S$ of the sparsity matrix.
2: $T := \max\{1, pqD_S\}$
3: for $\bar{s} = 1,\ldots,T$ do
4:     Decompose $\bar{Z}$ by solving (7) using PALM (Algorithm 2).
5:     if The optimal value of (7) is zero then
6:         Return success and $\bar{L}, \bar{S}$.
7:     end if
8: end for
9: Return failure.
10: OUTPUT: Success with $\bar{L}, \bar{S}$ or failure
```

4 Facial reduction, bicliques and exposing vectors

4.1 Preliminaries on faces

We now present some of the geometric facts we need. More details can be found in e.g., Drusvyatskiy and Wolkowicz (2017), Pataki (2000), Krislock and Wolkowicz (2010), Drusvyatskiy et al. (2017). Recall that the set K is a proper convex cone if

$$K + K \subseteq K, \;\; \lambda K \subseteq K, \forall \lambda \geq 0, \;\; \operatorname{int}(K) \neq \emptyset.$$

The *dual cone*, K^*, is defined by

$$K^* = \{\phi \in \mathbb{R}^n \,:\, \langle \phi, k\rangle \geq 0, \forall k \in K\}.$$

A subcone $F \subseteq K$ is a *face*, $F \trianglelefteq K$, of the convex cone K if

$$x, y \in K, x + y \in F \implies x, y \in F.$$

The *conjugate face*, F^*, is defined by $F^* = F^\perp \cap K^*$, where $F^\perp$ denotes the *orthogonal complement* of F. A face $F \trianglelefteq K$ is an *exposed face* if there exists $\phi \in K^*$ such that $F = \phi^\perp \cap K$; and ϕ is an *exposing vector*. Let T be a subset of the convex cone K, then face$\,(T)$ is the smallest face of K containing T. It is known that: a face of a face is a face; an intersection of faces is a face; and essential for our algorithm is the following for finding an intersection of exposed faces $F_i \trianglelefteq K, i = 1,\ldots,k$, see Drusvyatskiy et al. (2017),

$$F_i = K \cap \phi_i^\perp, \forall i \implies \cap_{i=1}^k F_i = \left(\sum_{i=1}^k \phi_i\right)^\perp \cap K. \tag{9}$$

If $K = \mathcal{S}_+^n$, i.e., the positive semidefinite cone, then the facial structure is well understood. Faces are characterized by the ranges or nullspaces of the matrices in the face. Let $X \in \mathcal{S}_+^n$ be rank r and

$$X = \begin{bmatrix} P & Q \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P & Q \end{bmatrix}^\top$$

be the (orthogonal) spectral decomposition with $D \in \mathcal{S}^r_{++}$ being a diagonal matrix. Then the smallest face of $\mathcal{S}^n_+$ containing X is

$$\operatorname{face}(X) = P\mathcal{S}^r_+ P^\top = \mathcal{S}^n_+ \cap (QQ^\top)^\perp.$$

The matrix $QQ^\top$ is an *exposing vector* for $\operatorname{face}(X)$. Moreover, the relative interior satisfies

$$\operatorname{relint}(\operatorname{face}(X)) = P\mathcal{S}^r_{++}P^\top = \operatorname{relint}(\operatorname{face}(\hat{X})), \quad \forall \hat{X} \in \operatorname{relint}(\operatorname{face}(X)),$$

i.e., the face and the exposing vectors are characterized by the eigenspace of any $\hat{X}$ in the relative interior of the face.

For our application we use the following view of facial reduction and exposed faces.

Theorem 2 (Drusvyatskiy et al. 2015, Theorem 4.1; Drusvyatskiy and Wolkowicz 2017) *Consider a linear transformation $\mathcal{M} : \mathcal{S}^n \to \mathbb{R}^m$ and a nonempty feasible set*

$$\mathcal{F} := \{X \in \mathcal{S}^n_+ : \mathcal{M}(X) = b\},$$

for some $b \in \mathbb{R}^m$. Then a vector v exposes a proper face of $\mathcal{M}(\mathcal{S}^n_+)$ containing b if, and only if, v satisfies the auxiliary system

$$0 \neq \mathcal{M}^* v \in \mathcal{S}^n_+ \quad \text{and} \quad \langle v, b \rangle = 0.$$

Let N denote the smallest face of $\mathcal{M}(\mathcal{S}^n_+)$ containing b. Then the following statements are true.

1. *We always have $\mathcal{S}^n_+ \cap \mathcal{M}^{-1}N =$ $\operatorname{face}(\mathcal{F})$, the smallest face containing $\mathcal{F}$.*
2. *For any vector $v \in \mathbb{R}^m$ the following equivalence holds:*

$$v exposes N \iff \mathcal{M}^* v exposes \operatorname{face}(\mathcal{F}). \tag{10}$$

The result in (10) details the facial reduction process for the matrix completion problem using exposing vectors. More precisely, if $B \succeq 0$ is a principal submatrix of the data and $\operatorname{trace}(VB) = 0, V \succeq 0, V \neq 0$, then V provides an exposing vector for the image of the coordinate projection onto the submatrix. We can then complete V with zeros (adjoint of the coordinate projection) to get $Y \in \mathcal{S}^n_+$, an exposing vector for $\mathcal{F}$. Define the *triangular number*, $t(n) = n(n+1)/2$, and the isometry $\operatorname{s2vec} : \mathcal{S}^n \to \mathbb{R}^{t(n)}$ that vectorizes the upper-triangular part of a symmetric matrix columnwise.

Corollary 1 *Suppose that $1 < k < n$ and $\mathcal{M}$ in Theorem 2 is the coordinate projection onto the leading principal submatrix of order $k, m = t(k)$. Let $B \in \mathcal{S}^k_+$, $b =$ $\operatorname{s2vec}(B) \in \mathbb{R}^{t(k)}$, i.e., for $X \in \mathcal{S}^n$, we have*

$$\mathcal{M}(X)_{ij} = b_{ij}, \quad \forall 1 \le i \le j \le k.$$

Let

$$0 \neq V \in \mathcal{S}^k_+, \ \text{trace}\,(VB) = 0, \ v = \text{ s2vec}\, V.$$

Then $Y = \mathcal{M}^* v$ *is an exposing vector for the feasible set* $\mathcal{F}$, *i.e.,*

$$\text{trace}\,(Y(\mathcal{F})) = 0.$$

Proof The proof follows immediately from Theorem 10 as v exposes N and $Y = \mathcal{M}^* v$ is an exposing vector for face $(\mathcal{F})$. $\square$

4.2 Exposing vector

The following lemma shows a generic rank property of a matrix with its submatrix.

Lemma 2 (Generic rank property, Lemma 3.6 in Huang and Wolkowicz (2018)) *Let r be a positive integer and $Z_1 \in \mathbb{R}^{m\times r}$ and $Z_2 \in \mathbb{R}^{n\times r}$ be continuous random variables with i.i.d. entries. Set $Z = Z_1 Z_2^\top$ and let $X \in \mathbb{R}^{p\times q}$ be any submatrix of Z with* $\min(p, q) \ge r$. *Then* $\text{rank}\,(X) = r$ *with probability* 1.

Based on Lemma 2, we can assume that $\bar{L}$ returned by Algorithm 3 has the same rank as the targeting low-rank matrix L^* if $\min(p, q) > r$, i.e.,

$$\bar{Z} = \bar{L} + \bar{S}, \qquad \text{rank}\,(\bar{L}) = r. \tag{11}$$

That is, we solved (7) to global optimality with objective value being 0.

Proposition 1 *Under Assumption* 3, *suppose that PALM (Algorithm 2) returns $\bar{L}$ and $\bar{S}$ such that* (11) *holds. Without loss of generality, we further assume the targeting low-rank matrix L^* can be partitioned as $L^* = \left[\begin{array}{c|c} L_1 & L_2 \\ \hline \bar{L} & L_3 \end{array}\right]$ (after a permutation if needed), where $\bar{L} \in \mathbb{R}^{p\times q}$ and $r \le$* $\min(p, q)$, *and the SVD of $\bar{L}$ is given by*

$$\bar{L} = \begin{bmatrix} \bar{P} & \bar{U} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{Q} & \bar{V} \end{bmatrix}^\top, \quad \Sigma_r \in \mathcal{S}^r_{++}. \tag{12}$$

By adding appropriate blocks of zeros to $\bar{U}\bar{U}^\top$ and $\bar{V}\bar{V}^\top$ (after a permutation if needed), we get the following matrix W, which is an exposing vector for face (Y^*), *i.e.,* $\text{trace}\,(Y^* W) = 0$, *where Y^* is the optimal solution of* (5).

$$W = \left[\begin{array}{c|cc|c} 0 & 0 & 0 & 0 \\ \hline 0 & \bar{U}\bar{U}^\top & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right] + \left[\begin{array}{c|cc|c} 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & \bar{V}\bar{V}^\top & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{c|cc|c} 0 & 0 & 0 & 0 \\ \hline 0 & \bar{U}\bar{U}^\top & 0 & 0 \\ 0 & 0 & \bar{V}\bar{V}^\top & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right].$$

Proof Lemma 1 shows the property

$$\operatorname{rank}(Y^*) = \operatorname{rank}(L^*) = r.$$

Without loss of generality, after a permutation if needed, we assume that

$$Y^* = \begin{bmatrix} U \\ P \\ Q \\ V \end{bmatrix} D \begin{bmatrix} U \\ P \\ Q \\ V \end{bmatrix}^\top = \left[\begin{array}{c|c|c|c} UDU^\top & UDP^\top & UDQ^\top & UDV^\top \\ \hline PDU^\top & PDP^\top & PDQ^\top & PDV^\top \\ \hline QDU^\top & QDP^\top & QDQ^\top & QDV^\top \\ \hline VDU^\top & VDP^\top & VDQ^\top & VDV^\top \end{array}\right],$$

$$L^* = \left[\begin{array}{c|c} UDQ^\top & UDV^\top \\ \hline PDQ^\top & PDV^\top \end{array}\right], \text{ with } PDQ^\top = \bar{L} \text{ and } D \in \mathcal{S}^r_{++}.$$

Now, since $PDQ^\top = \bar{L}$, we have $\operatorname{Range}(\bar{L}) \subseteq \operatorname{Range}(P)$, $\operatorname{Range}(\bar{L}^\top) \subseteq \operatorname{Range}(Q)$. Next, from $\bar{L} = \bar{P}\Sigma_r\bar{Q}^\top$, we have $\operatorname{Range}(\bar{L}) \subseteq \operatorname{Range}(\bar{P})$, $\operatorname{Range}(\bar{L}^\top) \subseteq \operatorname{Range}(\bar{Q})$. Therefore, since $\bar{P}, P, \bar{Q}, Q$ all have only r columns and $\operatorname{rank}(\bar{L}) = r$. It follows that $\operatorname{Range}(P) = \operatorname{Range}(\bar{P}) = \operatorname{Range}(\bar{L})$ and $\operatorname{Range}(Q) = \operatorname{Range}(\bar{Q}) = \operatorname{Range}(\bar{L}^\top)$. We conclude that $P^\top \bar{U}\bar{U}^\top = \bar{P}^\top \bar{U}\bar{U}^\top = 0$ and $Q^\top \bar{V}\bar{V}^\top = \bar{Q}^\top \bar{V}\bar{V}^\top = 0$, i.e., that $Y^* \cdot W = 0$. Therefore W is an exposing vector of the optimal face, the face that contains Y^*, the optimal solution of (5). □

Remark 2 The reason we apply facial reduction to (5) (or equivalently (3)) is because applying facial reduction to (5) is better than applying facial reduction to the convex relaxation formulation. Since the goal is to solve the nonconvex problem (5) to recover the true matrices, when we apply facial reduction to (5), we can show that the exposing vector is actually the optimal face of the nonconvex problem (5) in Proposition 1, therefore the optimal solution lies in a smaller space and we have a better chance to obtain the optimal solution of the nonconvex problem (5). If we apply facial reduction to the convex approximation, we may not obtain the exposing vector of the "optimal face".

4.3 Reducing problem size using FR

From Proposition 1, we get Algorithm 4 to find an exposing vector Y_{expo} for $\operatorname{face}(Y^*)$.

Algorithm 4 Finding the final exposing vector

```
1: INPUT: partially observed matrix $Z \in \mathcal{M}^{m\times n}$, target rank $r$, density $D_S$, a set of bicliques $\Theta$, minimum clique size
   $K_{min} > 2r$; set of bicliques $\bar{\Theta} \leftarrow \emptyset$.
2: OUTPUT: final block exposing vector $Y_{expo}$ and biclique set $\bar{\Theta}$.
3: for each biclique $\alpha \in \Theta$ and the corresponding submatrix $\bar{Z} \in \mathbb{R}^{p\times q}$ do
4:     if $\min\{p,q\} \geq \frac{K_{min}}{2}$ then
5:         Find the decomposition of $\bar{Z} = \bar{L} + \bar{S}$ satisfying (11) using Algorithm 3;
6:         $[\bar{U}, \bar{V}] \leftarrow$ from SVD of $\bar{L}$ (12)
7:         $W_\alpha^P \leftarrow \bar{U}\bar{U}^\top$;
8:         $W_\alpha^Q \leftarrow \bar{V}\bar{V}^\top$;
9:         add the biclique $\alpha$ to the set $\bar{\Theta}$
10:    end if
11: end for
12: sum over the bicliques with block submatrices filled in with appropriate size of zero matrices:
        $$0 \neq Y_{expo} \leftarrow \left[\begin{array}{c|c} \sum_{\alpha\in\bar{\Theta}} W_\alpha^P & 0 \\ \hline 0 & \sum_{\alpha\in\bar{\Theta}} W_\alpha^Q \end{array}\right].$$
    return $Y_{expo}$ and $\bar{\Theta}$.
```

Remark 3 We emphasize that the size of the bicliques in Θ are *small*. A large biclique means an expensive SVD when finding the corresponding exposing vector. Adding many exposing vectors results in one exposing vector corresponding to the union of the bicliques after completion.

From Y_{expo} we can also find the basis for Null$\,(Y_{expo})$ which is given by the columns of

$$V = \text{Null}\,(Y_{expo}) = \begin{bmatrix} V_P & 0 \\ 0 & V_Q \end{bmatrix}, \quad V_P^\top V_P = I_{r_p},\ V_Q^\top V_Q = I_{r_q}, \tag{13}$$

where $V_P \in \mathbb{R}^{m\times r_p}$ and $V_Q \in \mathbb{R}^{n\times r_q}$. We denote $r_v = r_p + r_q < m+n$. Therefore Y^* (optimal solution of (5)) can be expressed as

$$Y^* = VRV^\top = \begin{bmatrix} V_P R_p V_P^\top & V_P R_{pq} V_Q^\top \\ V_Q R_{pq}^\top V_P^\top & V_Q R_q V_Q^\top \end{bmatrix}, \quad \text{for some } R = \begin{bmatrix} R_p & R_{pq} \\ R_{pq}^\top & R_q \end{bmatrix} \in \mathcal{S}^{r_v}. \tag{14}$$

From Lemma 1 and (13) we have rank$\,(Y^*) = $ rank$\,(V_P R_{pq} V_Q^\top) = $ rank$\,(R_{pq})$. Therefore, (5) reduces to the following problem which has a much smaller size for the low-rank component:

$$\begin{array}{rl} \min\limits_{R_{pq}\in\mathbb{R}^{r_p\times r_q}, S\in\mathbb{R}^{m\times n}} & \text{rank}\,(R_{pq}) + \mu\|S\|_0 \\ \text{s.t.} & \mathcal{P}_{\hat{E}}(V_P R_{pq} V_Q^\top) + \mathcal{P}_{\hat{E}}(S) = z. \end{array} \tag{15}$$

4.4 Further reducing the size of the problem

Note that the size of S in (15) can be further reduced. In the decomposition (11), we know that $\bar{L}$ and $\bar{S}$ are exactly recovered by PALM when successful. Therefore, there is a subset of entries of S^* that has been successfully recovered. We can remove this subset of entries from the linear constraints in (15). We let $\hat{E}_S$ denote the set of indices of entries of S^* that are exactly recovered by PALM, and let $\hat{E}_{S^c} := \hat{E} \backslash \hat{E}_S$ to get:

$$\begin{aligned} &\min \operatorname{rank}(R_{pq}) + \mu \|s\|_0 \\ &\text{s.t. } \mathcal{P}_{\hat{E}_S}(V_P R_{pq} V_Q^\top) = z_{\hat{E}_S} \\ &\qquad \mathcal{P}_{\hat{E}_{S^c}}(V_P R_{pq} V_Q^\top) + s = z_{\hat{E}_{S^c}} \\ &\qquad R_{pq} \in \mathbb{R}^{r_p \times r_q},\ s \in \mathbb{R}^{\hat{E}_{S^c}}. \end{aligned} \tag{16}$$

Moreover, the number of linear constraints in (16) can, and should, be further reduced to remove any redundant linear constraints that arose. We use A and B to denote the matrix representations of the coefficient matrices in $\mathcal{P}_{\hat{E}_S}(V_P R_{pq} V_Q^\top)$ and $\mathcal{P}_{\hat{E}_{S^c}}(V_P R_{pq} V_Q^\top)$, respectively. If the ith row of B (denoted by B_i) is in the row space of A, then B_i can be removed so the size of B gets smaller. The algorithm that we use for this purpose is outlined in Algorithm 5.

```
Algorithm 5 Removing rows of $B$ that are in the row space of $A$
1: INPUT: two matrices $A \in \mathbb{R}^{s_1 \times t}$, $B \in \mathbb{R}^{s_2} \times t$
2: Compute the rank of $A$, denote $r_A = \operatorname{rank}(A)$.
3: Compute the LU decomposition of $[A^\top, B^\top]$
4: Remove the first $r_A$ rows and first $s_1$ columns from the upper triangular matrix $U$ and obtain $\bar{U}$.
5: Let $I_c$ be the index set of the nonzero columns in $\bar{U}$.
6: Let $\bar{B}$ be the submatrix of $B$ by removing rows of $B$ corresponding to indices in $I_c$.
7: OUTPUT: $\bar{B}$
```

Since FR typically results in very small values of r_p and r_q in (13), the first set of linear constraints in (16) , i.e.,

$$\mathcal{P}_{\hat{E}_S}(V_P R_{pq} V_Q^\top) = z_{\hat{E}_S}, \tag{17}$$

is usually an overdetermined linear system. As a result, we can obtain L^* by only solving this linear system.

Lemma 3 *The target matrix L^* is unique if $\mathcal{P}_{\hat{E}_S}(V_P R_{pq} V_Q^\top) = z_{\hat{E}_{S^c}}$ in* (16) *has a unique solution.*

Lemma 4 *If L^* is the unique optimal solution of* (3), *then* (16) *has a unique optimal solution R^*_{pq} that recovers L^*:*

$$L^* = V_P R^*_{pq} V_Q^\top.$$

When we have enough bicliques, our numerical tests generally successfully recover L^* by solving the linear system (17). If we do not have enough bicliques, then we have to solve the nonconvex and nonsmooth problem (16). In this case, we can solve the following convex relaxation of (16):

$$\begin{aligned} \min_{R_{pq}\in\mathbb{R}^{r_p\times r_q}, s\in\mathbb{R}^{\hat{E}_{S^c}}} \quad & \|R_{pq}\|_* + \mu\|s\|_1 \\ \text{s.t.} \quad & \mathcal{P}_{\hat{E}_S}(V_P R_{pq} V_Q^\top) = z_{\hat{E}_S} \\ & \mathcal{P}_{\hat{E}_{S^c}}(V_P R_{pq} V_Q^\top) + s = z_{\hat{E}_{S^c}}. \end{aligned} \tag{18}$$

Standard first-order methods such as alternating direction method of multipliers can be used to solve (18). Moreover, since the size of (18) is very small, we can also use interior point methods to solve it. The convex relaxation (18) after facial reduction is usually tighter than (4) and is more likely to successfully recover L^*. We do not go into more details in this paper.

4.5 Detecting the target rank

So far we have assumed that we know the target rank in advance, while in practice we may not know the target rank. We now assume that we know a *range* of the target rank and the density of the sparse part.

Algorithm 6 Detecting the target rank

```
1: INPUT: $\bar{Z} \in \mathbb{R}^{p\times q}$, range of the target rank $[r_{min}, r_{max}]$, density $D_S$ of the sparsity matrix.
2: $T := \max\{1, 2pqD_S\}$
3: for $r = r_{\min}, \dots, r_{\max}$ do
4:     for $\bar{s} = 1, \dots, T$ do
5:         Decompose $\bar{Z}$ by solving (7) using PALM (Algorithm 2).
6:         if The optimal value of (7) is zero then
7:             Return success and target rank $r$.
8:         end if
9:     end for
10: end for
11: Return failure.
12: OUTPUT: Success with the target rank $r$ or failure
```

To obtain the correct target, we can just sample many submatrices from the observed data and run Algorithm 6. We then choose the rank which appears most correct. We tested Algorithm 6 on random instances with different size, ranks and density of

Table 1 Rank estimation; average of 100 instances

Specifications			True r =	Percent of estimated ranks				
m	n	D_S		$r-1$	r	$r+1$	$r+2$	$r+3$
10	10	0.01	2	0	100	0	0	0
15	15	0.01	2	0	96	4	0	0
20	20	0.01	2	0	85	12	3	0
10	10	0.02	3	0	90	9	1	0
15	15	0.02	3	0	71	15	7	7
20	20	0.02	3	0	61	13	12	14
10	10	0.03	4	0	56	18	17	9
15	15	0.03	4	0	55	20	11	14

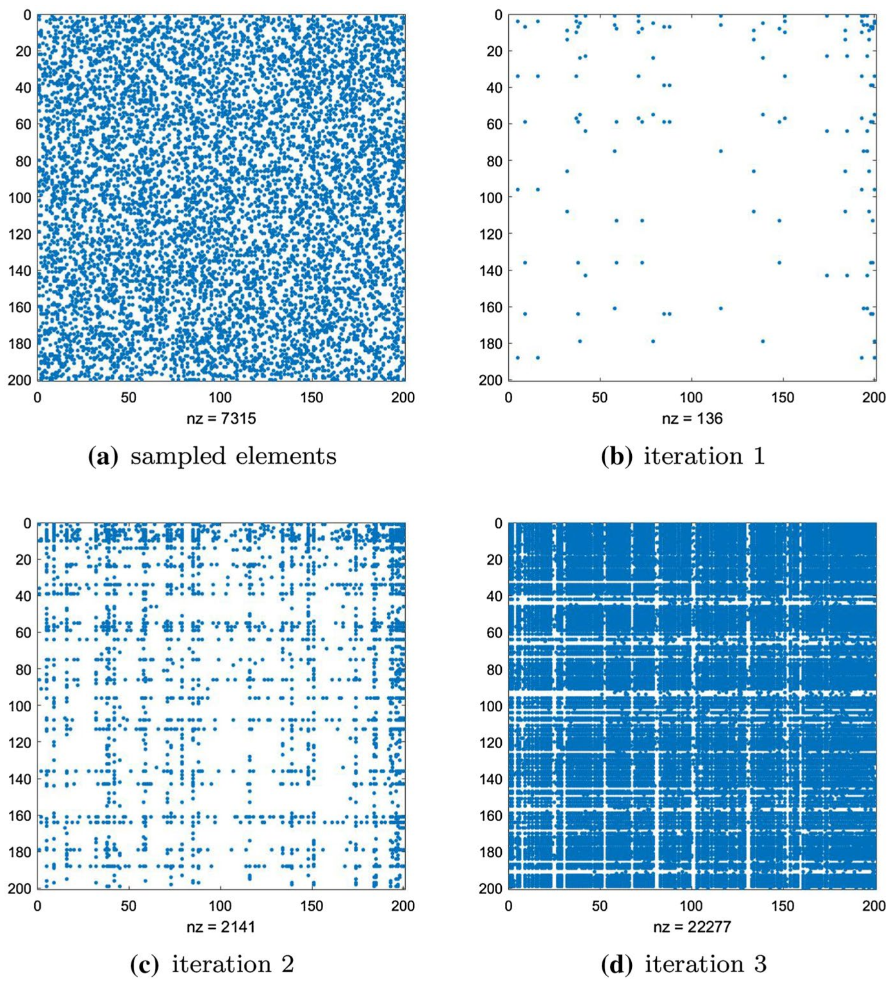

Fig. 1 Growth of set of bicliques; 3 iterations for sufficient growth

sparsity. The results are presented in Table 1. For example, the third row of Table 1 presents average results for 100 instances of size 20×20, density $D_S = 0.01$ and orignal rank $r = 2$. Algorithm 6 found estimated ranks $r = [1, 2, 3, 4, 5]$ in [0, 85, 12, 3, 0] percent, respectively. We conclude that Algorithm 6 estimates the correct rank in most cases.

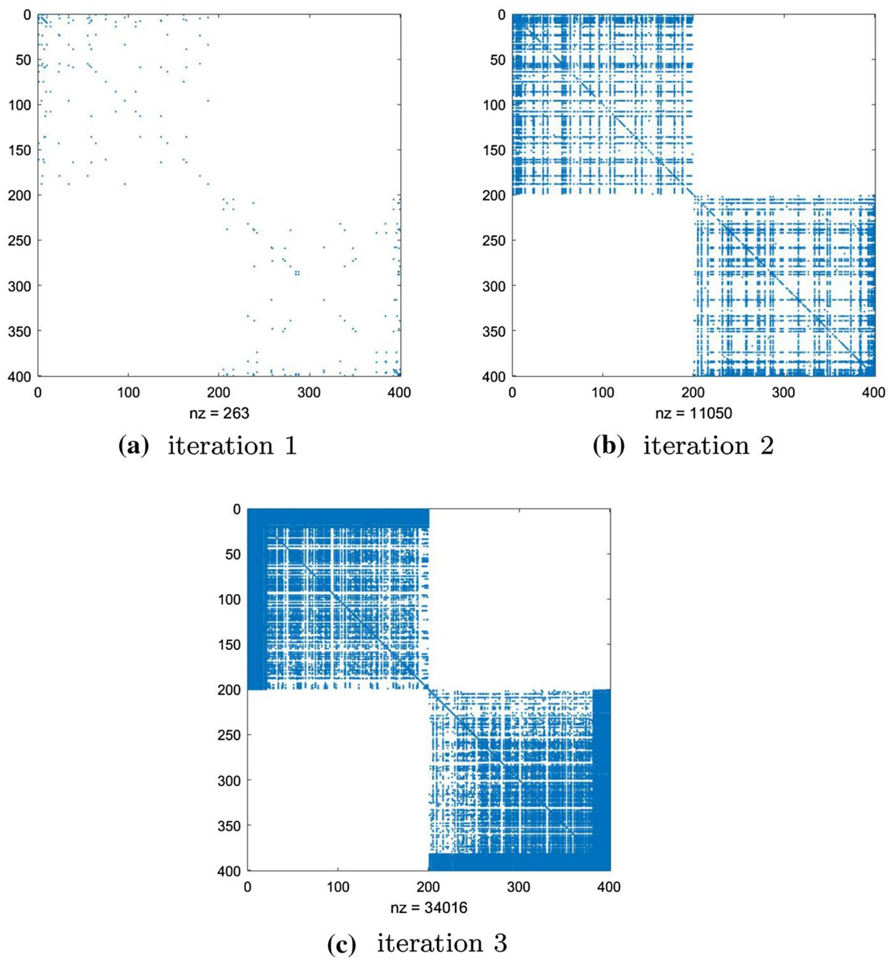

Fig. 2 Growth of Y_{expo}

5 Growing bicliques by submatrix completion

We observe that when there are not enough bicliques, we may have many zero rows and zero columns in the exposing vector Y_{expo}. Let the set of indices for nonzero rows and columns in the upper-left block of Y_{expo} be J_1 and the set of indices for nonzero rows and columns in the bottom-right block of Y_{expo} be J_2. Now, if we consider the submatrix Z_{J_1,J_2} which is obtained by taking the rows of Z whose indices are in J_1 and columns of Z whose indices are in J_2, we see that the size of the problem (16) is much smaller, i.e., since we remove all the zero rows and columns, we observe that the exposing vectors now cover all the remaining rows and columns, i.e.,those of Z_{J_1,J_2}. Hence we have a better chance of recovering Z_{J_1,J_2}. In fact, we often get a unique solution.

After we recover the submatrix Z_{J_1,J_2}, we can add the indices for all the entries in Z_{J_1,J_2} to the sampled indices set $\hat{E}$. The sampled indices set $\hat{E}$ contains a larger biclique $J_1 \times J_2$ and has more elements. We then use Algorithm 1 again to find more bicliques. This process can be repeated until all the rows and columns in Z are recovered.

We illustrate the growth of bicliques and the exposing vector Y_{expo} with an example. We choose the size of Z to be 200×200, the density for the sampled data is 0.18 and the minimum size for the bicliques is 4×4 with the target rank $r = 2$. The original low rank matrix L is recovered after 3 iterations by our algorithm. The example is shown in Figs. 1 and 2.

Our main algorithm for recovering the low rank matrix is summarized in Algorithm 7.

Algorithm 7 Recovery of the low rank component L in RPCA with partially observed data (3)

```
1: INPUT: Partial matrix $Z \in \mathbb{R}^{m\times n}$ with sampling indices set $\hat{E}$, target rank $r$, density $D_S$, limit of iteration $T$,
   tolerance for the rank $tol$, range of clique size $[K_{min}, K_{max}]$;
2: Set $J_1 = J_2 = \emptyset$, $\hat{E}_S = \emptyset$, $W_{expo} = 0$, $j = 0$
3: while $J_1 \times J_2$ is growing with $|J_1| + |J_2| < m + n$ and $\operatorname{rank}(L, tol) = r$ and $j < T$ do
4:     Find a set of bicliques $\Theta = \{\alpha_1, ..., \alpha_k\}$ in $\hat{E}$ by Algorithm 1.
5:     Calculate the exposing vector $Y_{expo}$ and biclique set $\bar{\Theta}$ by Algorithm 4.
6:     Update
           $$W_{expp} := W_{expo} + Y_{expo}.$$
7:     Set $Y_P = W_{expo}[1:m, 1:m]$ and $Y_Q = W_{expo}[m+1:m+n, m+1:m+n]$.
8:     Calculate the indices sets $J_1$ and $J_2$: $J_1 := \text{find}(\text{diag}(Y_P) > 0)$, $J_2 := \text{find}(\text{diag}(Y_Q) > 0)$.
9:     Update $Y_P = Y_P[J_1, J_1]$ and $Y_Q = Y_Q[J_2, J_2]$.
10:    Calculate $V_P = \text{Null}(Y_P)$ and $V_Q = \text{Null}(Y_Q)$ such that
           $$V_P^\top V_P = I_{r_p}, \; V_Q^\top V_Q = I_{r_q},$$
11:    Update $\hat{E}_S = \hat{E}_S \cup \bar{\Theta}$ and $\hat{E}_{S^c} = (\hat{E} \backslash \hat{E}_S) \cap \{J_1 \times J_2\}$.
12:    Solve problem (16) with the sampling indices $\hat{E}_S, \hat{E}_{S^c}$.
13:    Calculate the low rank matrix completion $L = V_P R_{pq} V_Q^\top$.
14:    Set $\hat{E} = \hat{E} \cup \{J_1 \times J_2\}$ and set $Z[J_1, J_2] = L$.
15:    $j = j + 1$.
16: end while
17: return $L$ or FAILURE if $|J_1| + |J_2| < m + n$ or $\operatorname{rank}(L, tol) \neq r$.
```

Remark 4 The motivation to grow bicliques arises from the desire to improve efficiency. We note that Algorithm 1 is only a heuristic and finding all the bicliques in a graph is an NP-hard problem. If there are not enough bicliques being identified, the equation in (17) may be underdetermined. Therefore we have to solve the convex approximation which can lead to a failure to recover the true low-rank matrix. The growing bicliques approach uses the existing bicliques to grow a big biclique. And when we add this big biclique to the sampled elements and run Algorithm 1 again, we are able to find many more bicliques efficiently. We note that the heuristic Algorithm 1 performs well when a big biclique exists. Our experiments show that this process is much more efficient than simply running Algorithm 1 many times. In fact this process is critical for the success of the method. Without the biclique growing procedure the proposed method takes much longer and will often fail to recover the target matrices if the density of the samplings is low.

Table 2 Results of Algorithm 7 for solving (3) with target rank $r = 2$

Specifications			Succ	CPU	$\hat{r}$	K_{min}	K_{max}	μ
m	n	δ						
800	800	0.18	14	11.55	2	10	50	0.88
1000	1000	0.18	19	14.55	2	10	50	0.79
1200	1200	0.18	18	23.52	2	10	50	0.29
1500	1500	0.18	17	27.77	2	10	50	0.26
1800	1800	0.18	17	33.43	2	10	50	0.12
2100	2100	0.18	19	37.84	2	10	50	0.11
3000	3000	0.18	17	54.61	2	10	50	0.09
5000	5000	0.18	16	106.87	2	10	50	0.07

Table 3 Results of Algorithm 7 for solving (3) with target rank $r = 3$

Specifications			Succ	CPU	$\hat{r}$	K_{min}	K_{max}	μ
m	n	δ						
800	800	0.26	18	17.74	3	10	60	0.12
1000	1000	0.26	14	21.20	3	10	60	0.11
1500	1500	0.26	14	30.32	3	10	60	0.09
2000	2000	0.26	18	41.83	3	12	60	0.07
2500	2500	0.26	18	48.74	3	12	60	0.07
3000	3000	0.26	19	67.17	3	12	60	0.06
4000	4000	0.26	16	97.61	3	12	60	0.05
5000	5000	0.26	16	119.99	3	12	60	0.05

6 Numerical experiments

6.1 Noiseless case

We applied our Algorithm 7 to solving random noiseless instances of (3). The input data were generated in the following manner. The low-rank matrix L^* was integer valued and obtained using $L^* = \text{round}(L_r L_c^\top)$, where $L_r \in \mathbb{R}^{m\times r}$ and $L_c \in \mathbb{R}^{r\times n}$ are from the standard normal distribution, $N(0, 10)$. The density of the sparse matrix $S^* \in \mathbb{R}^{m\times n}$ is denoted by D_S, and the nonzero elements were integer valued and obtained by rounding the uniformly distributed random elements. Matrix Z is set as $Z = L^* + S^*$. We then randomly sample elements in Z using the density δ, i.e., the ratio of the observed components of Z is δ. We use $\hat{L}$ to denote the recovery returned by our Algorithm 7.

For each set of data parameters (m, n, δ), we randomly generated 20 instances and reported the averaged performance in Tables 2 and 3. In these tables, $D_S = 0.01$, $\mu = c/(\delta\sqrt{m})$ where c is a chosen constant in [1, 10], the cpu times are in seconds, K_{min} denotes the smallest size of the cliques we found, K_{max} denotes the largest size of the cliques we found, $\hat{r} = \text{rank}\,(\hat{L})$ and "Succ" denotes the number of

Table 4 Comparison of Algorithm 7 and FALM

Specifications				Succ FR	Succ FALM	CPU FR	CPU FALM
m	n	δ	r				
500	500	0.33	4	18	0	15.88	11.07
500	500	0.36	4	19	0	9.33	11.11
500	500	0.45	4	20	8	5.00	10.92
500	500	0.5	4	20	18	5.38	11.66
600	600	0.22	3	18	0	25.62	15.03
1000	1000	0.22	3	16	0	18.16	72.22
1000	1000	0.33	3	17	0	23.81	72.61
1000	1000	0.45	3	20	20	11.73	76.22
1000	1000	0.36	5	20	1	29.36	75.49
1000	1000	0.39	5	20	14	27.88	77.53
2000	2000	0.28	5	14	6	67.76	615.62
2000	2000	0.33	5	20	18	56.74	651.81
2000	2000	0.26	3	18	0	41.83	662.70
2000	2000	0.36	3	20	20	36.74	673.05
2100	2100	0.18	2	19	0	37.84	728.17
3000	3000	0.18	2	17	0	54.61	2216.66

successfully recovered instances out of 20 randomly generated ones. Here we claim that matrix L^* is successfully recovered if there is no difference between $\hat{L}$ and L^*, i.e., $\|\hat{L} - L^*\|_F = 0$. Note that this is possible because all entries of L^* are integer valued. From these results we see that we can get exact recovery with very low density. Moreover, our algorithm is very efficient, and for very large problems with size 5000×5000, we can solve it in about two minutes.

We also compare our method with the fast alternating linearization method (FALM) proposed in Goldfarb et al. (2013). The results are reported in Table 4, where $D_S = 0.01$. From these results we see that after preprocessing by facial reduction, our reduced model (16) has a higher probability for recovering the low-rank matrix than solving the convex relaxation (4) by FALM. Moreover, our algorithm, although using a second-order interior point solver, is significantly faster than the first order method FALM. Our code is available at http://www.math.uwaterloo.ca/~hwolkowi//henry/reports/Codeforthetables.

Recently there has been a trend to use matrix factorization to solve the low-rank problem, such as Sun and Luo (2016), Yi et al. (2016). Therefore we also compared our method with the methods based on matrix factorization. Since in Sun and Luo (2016) no sparse gross errors are considered, we only compare the fast gradient descent method in Yi et al. (2016)denoted as FGD with our method. We note that in Yi et al. (2016) only the noiseless cases are considered therefore we only consider the noiseless case in the comparisons. We note in Yi et al. (2016), the authors make two assumptions about the data: (i) The low rank matrix

Table 5 Comparison of Algorithm 7 and FGD without assumptions (i), (ii)

Specifications				Succ FR	Succ FGD	CPU FR	CPU FGD
m	n	δ	r				
800	800	0.18	2	18	0	13.85	0.692172
1000	1000	0.18	2	20	0	18.53	1.293400
1200	1200	0.18	2	20	0	36.27	1.878267
1500	1500	0.18	2	16	0	41.98	2.596376
1800	1800	0.18	2	18	0	53.83	3.587537
2100	2100	0.18	2	18	0	63.12	5.182643
3000	3000	0.18	2	16	0	101.09	9.338740

Table 6 Comparison of Algorithm 7 and FGD with assumptions (i), (ii)

Specifications				Succ FR	Succ FGD	CPU FR	CPU FGD
m	n	δ	r				
800	800	0.18	2	20	0	12.14	0.761310
1000	1000	0.18	2	20	0	15.54	1.196388
1200	1200	0.18	2	20	10	30.32	1.729541
1500	1500	0.18	2	20	18	32.78	2.624523
1800	1800	0.18	2	20	20	39.92	3.770775
2100	2100	0.18	2	20	20	48.48	4.905493
3000	3000	0.18	2	12	20	51.20	10.816730

Table 7 Comparison of Algorithm 7 and FALM ($\sigma = 10^{-4}$, $D_S = 0.01$, CPU times in seconds)

Specifications			CPU FR	CPU FALM	Rank	Res (FR)	Res (FALM)
m	n	Mean(p)					
600	600	0.33	7.19	17.55	2	1.730694e−04	1.478195e−03
800	800	0.33	10.56	38.00	2	1.189773e−04	1.005278e−03
1000	1000	0.26	14.30	78.66	2	1.123756e−04	2.154113e−03
1500	1500	0.26	21.28	280.05	2	1.853607e−04	9.815458e−04
2000	2000	0.26	50.63	687.00	2	1.120183e−04	1.388883e−04

M needs to be μ-incoherent. (ii) The entries of the sparse component S are "spread out", i.e., for $\alpha \in [0, 1)$, S contains at most α-fraction nonzero entries per row and column. In this paper, we don't make these assumptions. Therefore we first compare our method with their method when these assumptions hold, secondly, we compare our method with their method without these assumptions being hold. The results are in Table 5 and 6. From the results in the tables we can see that the fast gradient descent method usually works better for large scale problems when these two assumptions hold. However when these two assumptions do not hold,

Table 8 Comparison of Algorithm 7 and FALM ($\sigma = 10^{-5}$, $D_S = 0.01$, CPU times in seconds)

Specifications			CPU FR	CPU FALM	Rank	Res (FR)	Res (FALM)
m	n	Mean(p)					
600	600	0.33	9.44	17.49	3	6.600662e−05	1.837060e−03
600	600	0.36	6.53	18.25	3	1.808533e−05	1.156141e−03
800	800	0.33	9.70	39.08	3	2.187891e−05	1.033397e−03
800	800	0.36	8.65	38.29	3	1.240057e−05	8.217820e−04
1000	1000	0.33	11.98	80.09	3	1.523253e−05	5.252964e−04
1000	1000	0.36	11.38	79.56	3	1.125439e−05	3.015739e−04
1500	1500	0.33	19.89	284.93	3	1.220386e−05	7.854144e−05
1500	1500	0.36	23.08	302.34	3	1.022679e−05	5.251847e−05
2000	2000	0.33	43.87	728.43	3	1.088412e−05	2.258433e−05

our method can recover the true low rank matrix with a high success rate while the fast gradient descent method fails to recover any of them.

6.2 Noisy case

In the *noisy* case, we consider $Z = L + S + \mathcal{E}$ where L, S are no longer integer and all the elements of the matrix $\mathcal{E}$ are from random noise, generated from a Gaussian distribution with standard deviation σ. We still assume S is sparse, with density D_S.

The results are in Tables 7 and 8. Each row in the tables presents the average of 20 instances.

We see that when the noise level is small ($\sigma = 10^{-4}$ or 10^{-5}), the CPU time of our Algorithm 7 is significantly smaller than that of FALM; and the residual is smaller as well which is pa great advantage of our algorithm.

In the case when the noise level is relatively large, our algorithm does not perform as well as FALM. This appears to be due to the failure of recovering the correct submatrices $\bar{L}, \bar{S}$ from $\bar{Z}$. However in general, if Algorithm 7 succeeds in finding a solution with the correct target rank, then our algorithm generally has a smaller residual than the first order method FALM. If Algorithm 7 fails to find the solution with the correct target rank, FALM is generally better. We conclude that we should use Algorithm 7 first and check if the correct target rank is achieved with a satisfactory tolerance. If not, then we should fall back on the first order method FALM.

We note there are popular algorithms such as the stochastic gradient descent method (SGD) (Gemulla et al. 2011; Recht and Ré 2013; Zhuang et al. 2013) and block coordinate descent type methods (Pilászy et al. 2010; Yu et al. 2012) to solve the large scale low-rank matrix completion problems. Our method proposed in the paper can be modified to solve the low-rank matrix completion problems as well and in the future it will be very interesting to compare our methods to those methods for quality improvement when solving the general low-rank matrix completion problems. Also in Sun and Luo (2016), a regularization term is added to control incoherence in the iterates, it will be very interesting to study whether a regularization term will help to improve the recovery quality for RPCA problems.

7 Conclusion

In this paper we have shown that we can apply a facial reduction approach in combination with the nuclear norm heuristic to efficiently solve the robust PCA problem with partially observed data. This exploits the implicit degeneracy at the optimal solution resulting from the low-rank and sparse structure.

Specifically, whenever enough complete bipartite subgraphs in the data are available, we are able to find a proper face of the semidefinite cone that contains the optimal solution and results in a significant reduction in dimension. If we cannot find enough bicliques, the matrix can still be partially completed. Having an insufficient number of bicliques is indicative of not having enough initial data to recover the unknown elements for our algorithm. This is particularly true for large rank r, where larger bicliques are needed. Throughout we see that the facial reduction both regularizes the problem and reduces the size and often allows for a solution without any refinement or need for an SDP solver.

Our preliminary numerical results are promising as they efficiently and accurately recover large scale problems. The numerical tests are ongoing with improvements in using biclique algorithms rather than clique algorithms. At the current stage, we are only testing randomly generated examples. In practice most of the problems from the real world, for example from surveillance video with moving objects, the gross errors are not sparse enough, or in other words, the random noise is not significantly smaller than the sparse gross errors. To solve practical problems from the real world, we need to further refine our algorithm to better separate the random noise from the sparse gross errors, which is part of our current ongoing work.

Theoretical results on exact recovery are discussed in many papers, e.g., Candes et al. (2010), Chandrasekaran et al. (2009). They use the so-called incoherence conditions, which are difficult to verify. It appears from our work above that exact recovery guarantees can be found from rigidity results in the graph of Z, i.e., in the number and density of the bicliques. Moreover, there are interesting questions on how to extend these results from the simple matrix completion to general solutions of linear equations, $\mathcal{A}(Z) = b$, where $\mathcal{A}$ is some linear transformation.

Acknowledgements Open access funding provided by Royal Institute of Technology.

References

Aybat NS (2016) Algorithms for stable PCA. In: Bouwmans T, Aybat NS, Zahzah E (eds) Handbook of robust low rank and sparse matrix decomposition: applications in image and video processing. CRC Press, Cambridge

Blair JRS, Peyton B (1993) An introduction to chordal graphs and clique trees. In: Graph theory and sparse matrix computation, volume 56 of IMA volumes in mathematics and its applications. Springer, New York, pp 1–29

Bolte J, Sabach S, Teboulle M (2014) Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Math Program 146(1–2):459–494

Bouwmans T, Zahzah EH (2014) Robust pca via principal component pursuit: a review for a comparative evaluation in video surveillance. Comput Vis Image Underst 122:22–34

Candes EJ, Li X, Ma Y, Wright J (2010) Robust principal component analysis. J ACM 58(1):1–37

Chandrasekaran V, Sanghavi S, Parrilo PA, Willsky AS (2009) Rank-sparsity incoherence for matrix decomposition. Technical Report, MIT, Boston, MA. arXiv:0906.2220

Cheng J, Chen K, Sacchi MD (2015) Application of robust principal component analysis (RPCA) to suppress erratic noise in seismic records. In: SEG technical program expanded, pp 4646–4651, SEG

Drusvyatskiy D, Wolkowicz H (2017) The many faces of degeneracy in conic optimization. Found Trends^ Optim 3(2):77–170

Drusvyatskiy D, Pataki G, Wolkowicz H (2015) Coordinate shadows of semidefinite and Euclidean distance matrices. SIAM J Optim 25(2):1160–1178

Drusvyatskiy D, Krislock N, Voronin Y-LC, Wolkowicz H (2017) Noisy Euclidean distance realization: robust facial reduction and the Pareto frontier. SIAM J Optim 27(4):2301–2331

Gemulla R, Nijkamp E, Haas PJ, Sismanis Y (2011) Large-scale matrix factorization with distributed stochastic gradient descent. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 69–77

Goldfarb D, Ma S, Scheinberg K (2013) Fast alternating linearization methods for minimizing the sum of two convex functions. Math Program 141(1–2, Ser. A):349–382

Huang S, Wolkowicz H (2018) Low-rank matrix completion using nuclear norm with facial reduction. J Glob Optim 72(1):5–26

Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of computer computation. Plenum Press, New York, pp 85–103

Krislock N, Wolkowicz H (2010) Explicit sensor network localization using semidefinite representations and facial reductions. SIAM J Optim 20(5):2679–2708

Kurdyka K (1998) On gradients of functions definable in o-minimal structures. Annales de l'institut Fourier 146:769–783

Łojasiewicz S (1963) Une propriété topologique des sous-ensembles analytiques réels, Les Équations aux Dérivées Partielles. Éditions du centre National de la Recherche Scientifique, Paris

Ma S, Aybat NS (2018) Efficient optimization algorithms for robust principal component analysis and its variants. Proc IEEE 106(8):1411–1426

Pataki G (2000) Geometry of semidefinite programming. In: Wolkowicz H, Saigal R, Vandenberghe L (eds) Handbook of semidefinite programming: theory, algorithms, and applications. Kluwer Academic Publishers, Boston, MA

Pilászy I, Zibriczky D, Tikk D (2010) Fast ALS-based matrix factorization for explicit and implicit feedback datasets. In: Proceedings of the fourth ACM conference on Recommender systems. ACM, pp 71–78

Recht B, Ré C (2013) Parallel stochastic gradient algorithms for large-scale matrix completion. Math Program Comput 5(2):201–226

Recht B, Fazel M, Parrilo P (2010) Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Rev 52(3):471–501

Sun R, Luo Z-Q (2016) Guaranteed matrix completion via non-convex factorization. IEEE Trans Inf Theory 62(11):6535–6579

Yi X, Park D, Chen Y, Caramanis C (2016) Fast algorithms for robust PCA via gradient descent. In: Advances in neural information processing systems, pp 4152–4160

Yu H, Hsieh C, Si S, Dhillon I (2012) Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In: 2012 IEEE 12th international conference on data mining. IEEE, pp 765–774. https://doi.org/10.1109/ICDM.2012.168

Zhuang Y, Chin W-S, Juan Y-C, Lin C-J (2013) A fast parallel SGD for matrix factorization in shared memory systems. In: Proceedings of the 7th ACM conference on Recommender systems. ACM, pp 249–256