Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 656074, 8 pages
http://dx.doi.org/10.1155/2014/656074

*Research Article*

# A Generalized Robust Minimization Framework for Low-Rank Matrix Recovery

**Wen-Ze Shao,[1] Qi Ge,[1] Zong-Liang Gan,[1] Hai-Song Deng,[2] and Hai-Bo Li[3]**

[1] *College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*
[2] *School of Mathematics and Statistics, Nanjing Audit University, Nanjing 211815, China*
[3] *School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm 10044, Sweden*

Correspondence should be addressed to Wen-Ze Shao; shaowenze1010@163.com

Received 18 January 2014; Accepted 23 April 2014; Published 6 May 2014

Academic Editor: Carlo Cattani



This paper considers the problem of recovering low-rank matrices which are heavily corrupted by outliers or large errors. To improve the robustness of existing recovery methods, the problem is solved by formulating it as a generalized nonsmooth nonconvex minimization functional via exploiting the Schatten $p$-norm ($0 < p \leq 1$) and $L_q$($0 < q \leq 1$) seminorm. Two numerical algorithms are provided based on the augmented Lagrange multiplier (ALM) and accelerated proximal gradient (APG) methods as well as efficient root-finder strategies. Experimental results demonstrate that the proposed generalized approach is more inclusive and effective compared with state-of-the-art methods, either convex or nonconvex.

## 1. Introduction

In many practical applications, such as removing shadows and specularities from face images, separating foregrounds and backgrounds from monitored videos, ranking, and collaborative filtering, the observed data matrix $C$ can naturally be decomposed into a low-rank matrix $A$ and a corrupted matrix $B$. That is, $C = A + B$, where $B$ can be arbitrarily large and is usually assumed to be sparse and unknown. The problem is whether it is possible to recover the low-rank matrix $A$ from the observed $C$. Recently, it has been shown that the answer is affirmative as long as the corrupted matrix $B$ is sufficiently sparse and the rank of the low-rank matrix $A$ is sufficiently low [1, 2]. That is to say, under certain conditions one can exactly recover from $C$ the low-rank matrix $A$ with high probability by solving the following convex optimization problem, that is, the idealization of robust principal component analysis (RPCA):

$$\begin{aligned} &\min_{A,B} \quad \|A\|_* + \lambda \|B\|_1, \\ &\text{s.t.} \quad C = A + B, \end{aligned} \tag{1}$$

where $\|A\|_*$ denotes the nuclear norm of the low-rank matrix $A$, that is, the sum of the singular values of $A$, $\|B\|_1$ denotes the $L_1$ norm of the matrix $B$ when seen as a vector, and $\lambda$ is a positive tuning parameter. Recently, there has been a lot of research focusing on solving the RPCA problem, for example, [3–6].

In spite of the great theoretical success and wide practical applications of RPCA (1), its major limitation should be claimed due to the use of nuclear and $L_1$ norms as regularizers. Specifically, compared with the intrinsic rank constraint, that is, rank $(A) < r_0$, the nuclear norm regularizer will not only do more harm to the large singular values of $A$ but also lead to weaker shrinkage of the disturbed small singular values. It is not hard to make a similar analysis with the case of the $L_1$ norm regularizer. Then, the performance of RPCA (1) in dimensionality reduction and outlier separation will not be as good as expected in some scenarios. In Section 3, it has been empirically demonstrated that RPCA (1) is not robust to the case that either the matrix $A$ is not sufficiently low-rank or the matrix $B$ is more grossly corrupted.

To improve the robustness of RPCA [3–6], this paper proposes a generalized nonsmooth nonconvex minimization

framework for low-rank matrix recovery by exploiting the Schatten $p$-norm ($0 < p \leq 1$) and $L_q$($0 < q \leq 1$) seminorm. And two numerical algorithms are deduced based on the augmented Lagrange multiplier (ALM) and accelerated proximal gradient (APG) methods as well as efficient root-finder strategies. Experimental results show that the proposed generalized approach is more inclusive and effective compared with state-of-the-art methods [3–6]. Notice that much recently a nonconvex relaxation approach for low-rank matrix recovery [7] is proposed exploiting a nonconvex penalty called minmax concave plus and also a nonconvex loss function. However, our approach is different from [7] and is better in the terms of recovery accuracy and robustness than [7] as well as the other two nonconvex methods [8, 9]. The paper is organized as follows. Section 2 provides the generalized nonsmooth nonconvex minimization framework, including the problem formulation and two numerical algorithms based on the ALM and APG methods. Section 3 verifies the recovery performance of the proposed method and compares it against state-of-the-art methods. Finally, the paper is concluded in Section 4.

## 2. Proposed Model and Algorithms

*2.1. Problem Formulation.* Taking into account both the recovery robustness and computational efficiency, the Schatten $p$-norm is used to better approximate the intrinsic rank constraint $\text{rank}(A) < r_0$; similarly, the $L_q$ seminorm is exploited to replace the $L_1$ norm of a matrix when seen as a vector. It is now intuitive to generalize RPCA as follows:

$$\begin{aligned} &\min_{A,B} \quad \|A\|_{Sp}^p + \lambda \|B\|_{Lq}^q, \\ &\text{s.t.} \quad C = A + B, \end{aligned} \tag{2}$$

where $0 < p, q \leq 1$, $\|A\|_{Sp}$, and $\|B\|_{Lq}$ are defined, respectively, in the following. Assume $A, B, C \in \mathbf{R}^{m\times n}$; then the $L_q$ seminorm of a matrix $B$ when seen as a vector can be defined as

$$\|B\|_{Lq} = \left( \sum_{k=1,l=1}^{m,n} |B_{k,l}|^q \right)^{1/q}, \tag{3}$$

where $B_{k,l}$ is the $k,l$th element of $B$; and the Schatten $p$-norm of a matrix $A$ can be defined as

$$\|A\|_{Sp} = \left( \sum_{j=1}^{\min\{m,n\}} \sigma_j^p \right)^{1/p}, \tag{4}$$

where $\sigma_j$ is the $j$th singular value of $A$ and the singular value decomposition (SVD) of $A$ is $A = U\Sigma V^T$. Clearly, as $p = q = 1$, (2) reduces to convex RPCA in (1); as $0 < p, q < 1$, (2) corresponds to a constrained nonsmooth and nonconvex minimization problem. Now, it comes to study the numerical iteration schemes of (2).

In recent several years, communities of signal processing and computational mathematics show more and more interests in developing efficient algorithms for nonlinear nonsmooth optimization problems [10], such as iterative soft thresholding, split Bregman iteration, accelerated proximal gradient, augmented Lagrange multiplier, and so on, which have significantly simplified sparse optimization problems including RPCA (1). In a similar spirit to [1, 3, 4, 7], this paper exploits the accelerated proximal gradient (APG) and augmented Lagrange multiplier (ALM) methods to solve the generalized minimization problem (2), considering the fact that APG and ALM are the two most popular numerical algorithms currently. As for ALM, it has been shown [11] that under certain general conditions, ALM converges $Q$ linearly to the optimal solution. As for APG, though little is known about the actual convergence of its produced sequences, the $O(1/k^2)$ rate of convergence of the objective function that they achieve is optimal [10]. However, we should note that the above mentioned convergence results are not applicable to the new problem (2) because of its nonsmooth and nonconvex properties. In spite of that, empirical studies in Section 3 demonstrate that the two deduced algorithms in the following can both solve (2) very well, with empirically fast convergence rate.

*2.2. ALM-Based Algorithm.* This subsection exploits ALM to solve problem (2), which is a nonconvex extension of [3]. First of all, define functions $f(A, B)$ and $h(A, B)$ as

$$\begin{aligned} f(A, B) &= \|A\|_{Sp}^p + \lambda \|B\|_{Lq}^q, \\ h(A, B) &= C - A - B. \end{aligned} \tag{5}$$

According to ALM, the Lagrange function for (2) is given as

$$L(A, B, X, \mu) = f(A, B) + \langle X, h(A, B) \rangle + \frac{\mu}{2} \|h(A, B)\|_{L2}^2, \tag{6}$$

where $X$ is a matrix of Lagrange multipliers and $\mu \geq 0$ is the augmented Lagrange penalty parameter. It is seen that $A$, $B$ can be solved iteratively by alternating minimization of $L(A, B, X, \mu)$. In the meanwhile, a continuation strategy is applied to $\mu \geq 0$ in order to improve both the accuracy and efficiency of low-rank matrix recovery. Specifically, the iteration process is described in Algorithm 1.

In the following, $A^{k+1}, B^{k+1}$ are solved, respectively, by

$$\min_A \frac{\mu^k}{2} \left\| \left( C - B^k + \frac{X^k}{\mu^k} \right) - A \right\|_{L_2}^2 + \|A\|_{Sp}^p, \tag{7}$$

$$\min_B \frac{\mu^k}{2} \left\| \left( C - A^{k+1} + \frac{X^k}{\mu^k} \right) - B \right\|_{L_2}^2 + \lambda \|B\|_{Lq}^q. \tag{8}$$

As for (7), suppose that $C - B^k + X^k/\mu^k = \Phi^k \Omega^k (\Psi^k)^T$; then (7) can be rewritten as

$$\min_{\Sigma \geq 0} \frac{\mu^k}{2} \left\| \Phi^k \Omega^k \left( \Psi^k \right)^T - U\Sigma V^T \right\|_{L_2}^2 + \text{trace}\left( \Sigma^p \right). \tag{9}$$

Set $1 < \rho < 2$, $\lambda, \overline{\mu}, \varepsilon$ and $K$. Initialize $k = 0$, $B^0, X^0, \mu^0$

**while** Residual error $= \dfrac{\left\|C - A^{k+1} - B^{k+1}\right\|_{L2}}{\max\left\{\|C\|_\infty/\lambda, \|C\|_2\right\}} > \varepsilon$

$A^{k+1} = \arg\min_A L(A, B^k, X^k, \mu^k)$;

$B^{k+1} = \arg\min_B L(A^{k+1}, B, X^k, \mu^k)$;

$X^{k+1} = X^k + \mu^k(C - A^{k+1} - B^{k+1})$;

$\mu^{k+1} = \min\{\rho\mu^k, \overline{\mu}\}$;

$k = k + 1$;

**if** $k > K$ *break*; **end**

**end**

Algorithm 1

According to von Neumann's theorem on singular values [12], we have

$$\begin{aligned}
&\left\|\Phi^k\Omega^k\left(\Psi^k\right)^T - U\Sigma V^T\right\|_{L_2}^2 \\
&\quad = \operatorname{trace}\left(\left(\Omega^k\right)^T\Omega^k\right) + \operatorname{trace}\left(\Sigma^T\Sigma\right) \\
&\qquad - 2\operatorname{trace}\left(\Phi^k\Omega^k\left(\Psi^k\right)^T U\Sigma V^T\right) \\
&\quad \geq \operatorname{trace}\left(\left(\Omega^k\right)^T\Omega^k\right) + \operatorname{trace}\left(\Sigma^T\Sigma\right) - 2\operatorname{trace}\left(\left(\Psi^k\right)^T\Sigma\right) \\
&\quad = \left\|\Omega^k - \Sigma\right\|_{L_2}^2.
\end{aligned} \tag{10}$$

Hence, the minimization problem (7) can be approximated by

$$\min_{\Sigma\geq 0}\frac{\mu^k}{2}\left\|\Omega^k - \Sigma\right\|_{L_2}^2 + \operatorname{trace}\left(\Sigma^p\right). \tag{11}$$

More concretely, that is,

$$\min_{\sigma^j\geq 0}\frac{\mu^k}{2}\left(\sigma^j - \omega_j^k\right)^2 + \left(\sigma^j\right)^p. \tag{12}$$

Let $\widetilde{B}^k = C - A^{k+1} + X^k/\mu^k$. It is not hard either to find that the minimization problem (8) can be reduced to

$$\min_{B_{i,j}}\frac{\mu^k}{2\lambda}\left(B_{i,j} - \widetilde{B}_{i,j}^k\right)^2 + \left(B_{i,j}\right)^q. \tag{13}$$

From (12) and (13), the kernel of Algorithm 1 is the following root-finder problem:

$$\min_{\varsigma}\frac{\beta}{2}(\varsigma - \xi)^2 + (\varsigma)^\alpha. \tag{14}$$

Here, we borrow the numerical idea in [13] to solve (14). For $\alpha = 1, 1/2, 2/3$, analytical solutions are used as calculated by Algorithms 2 and 3 in [13]. For all other $\alpha$ values, the numerical root-finder method Newton-Raphson is exploited to solve (14).

*2.3. APG-Based Algorithm.* This subsection exploits AGP as well as the continuation technique to solve problem (2), which is a nonconvex extension of [4]. First of all, a relaxed minimization problem is produced from (2); that is,

$$\min_{A,B} F(A, B) = \min_{A,B} \nu f(A, B) + g(A, B), \tag{15}$$

where $g(A, B) = \|h(A, B)\|_{L2}^2/2$ and $\nu \geq 0$ is a relaxation parameter. Obviously, $F(A, B)$ is different from the Lagrange function of (2) in Section 2.2. However, instead of directly minimizing $F(A, B)$, a sequence of separable quadratic approximations to $F(A, B)$ is minimized, denoted as $Q(A, B, \Lambda, \Gamma)$,

$$\begin{aligned}
Q(A, B, \Lambda, \Gamma, \nu) = {} & g(\Lambda, \Gamma) + \left\langle \nabla g(\Lambda, \Gamma), (A, B) - (\Lambda, \Gamma)\right\rangle \\
& + \frac{L_g}{2}\|(A, B) - (\Lambda, \Gamma)\|_{L2}^2 + \nu f(A, B),
\end{aligned} \tag{16}$$

where $(\Lambda, \Gamma)$ are specifically chosen points and $L_g = 2$. Then, $A, B$ can be solved iteratively by alternating minimization of $Q(A, B, \Lambda, \Gamma, \nu)$ with reasonable choices of $\Lambda, \Gamma$. That is,

$$\begin{aligned}
\min_{A,B} Q(A, B, \Lambda, \Gamma, \nu) = {} & \min_{A,B} \nu f(A, B) \\
& + \frac{L_g}{2}\|(A, B) - (G_\Lambda, G_\Gamma)\|_{L2}^2,
\end{aligned} \tag{17}$$

where

$$G_\Lambda = \Lambda - \frac{1}{2}\nabla g(\Lambda, \Gamma), \qquad G_\Gamma = \Gamma - \frac{1}{2}\nabla g(\Lambda, \Gamma). \tag{18}$$

To assure both the accuracy and efficiency of minimizing $Q(A, B, \Lambda, \Gamma, \nu)$, two key strategies are taken into account deliberately. For one thing, $(\Lambda, \Gamma)$ are determined by iterative smoothed computation as suggested in [14]. For another, the continuation technique is also applied to $\nu \geq 0$, just similar to Algorithm 1. Moreover, the stopping criterion is identical to the one proposed in [15] and utilized in [4]. The iteration scheme is presented in Algorithm 2 specifically.

In the following, $A^{j+1}, B^{j+1}$ can be solved, respectively, by

$$\min_A \frac{L_g}{2}\left\|A - G_\Lambda^j\right\|_{L_2}^2 + \nu^j\|A\|_{Sp}^p, \tag{19}$$

$$\min_B \frac{L_g}{2}\left\|B - G_\Gamma^j\right\|_{L_2}^2 + \nu^j\lambda\|B\|_{Lq}^q. \tag{20}$$

Set $0 < \eta < 1, \lambda, \overline{\nu}, \tau, J$. Initialize $j = 1, A^{0/-1}, B^{0/-1}, \Lambda^0, \Gamma^0, \nu^0$.

**while** Residual error $= \dfrac{\left\|(\Lambda^j - A^{j+1}) + (B^{j+1} - \Gamma^j)\right\|_{L2}}{\sqrt{2}\max\left(1, \sqrt{\left\|\Lambda^j\right\|_{L2}^2 + \left\|\Gamma^j\right\|_{L2}^2}\right)} > \tau$

$\Lambda^j = A^j + \dfrac{t_{j-1} - 1}{t_j}\left(A^j - A^{j-1}\right), \qquad \Gamma^j = B^j + \dfrac{t_{j-1} - 1}{t_j}\left(B^j - B^{j-1}\right);$

$G_\Lambda^j = \Lambda^j - \dfrac{1}{2}\left(\Lambda^j + \Gamma^j - C\right), \qquad G_\Gamma^j = \Gamma^j - \dfrac{1}{2}\left(\Lambda^j + \Gamma^j - C\right),$

$A^{j+1} = \arg\min_A Q\left(A, B^j, \Lambda^j, \Gamma^j, \nu^j\right);$

$B^{j+1} = \arg\min_B Q\left(A^{j+1}, B, \Lambda^j, \Gamma^j, \nu^j\right);$

$\nu^{j+1} = \max\left\{\eta\nu^j, \overline{\nu}\right\};$

$j = j + 1;$

**if** $j > J$ *break*; **end**

**end**

Algorithm 2

Similar to (12) and (13) in Section 2.2, both (19) and (20) are instances of rooter-find problems (14) and hence can be solved efficiently by borrowing the numerical idea in [13].

## 3. Experimental Results

*3.1. Experimental Settings.* In this section, simulation experiments are designed and conducted to show the validity of our proposed approach. It firstly needs to produce available data using $C = A^* + B^*$, in which $A^*$ and $B^*$ are, respectively, the true low-rank and sparse matrices that we wish to recover. Without loss of generality, $A^*$ is generated as a product of two $m \times r$ matrices whose entries are sampled i.i.d. from Gaussian distribution $N(0, 1/m)$, and the sparse matrix $B^*$ is constructed by setting a proportion of entries to be $\pm 1$ and the rest to be zeros. More specifically, if $r$ and spr represent, respectively, the matrix rank and sparsity ratio, then the MATLAB v7.0 scripts for generating $A^*$ and $B^*$ can be given as

(i) `A = 1/m*randn(m,r)*1/m*randn(r,m);`

(ii) `B = zeros(m,m);`

(iii) `p = randperm(m*m);`

(iv) `L = round(spr*m*m);`

(v) `B(p(1:L)) = sign(randn(L,1));`

In the following experiments, set $m$ to 500, $r$ to 50, 100, 150, and 200, and spr to 5%, 10%, 15%, and 20%. To be noted, the matrix recovery problem (2) roughly changes from easy to hard as $r$ or spr changes from small to large. To assess the accuracy of low-matrix recovery, the relative squared error (RSE) is used, defined as

$$\text{RSE} = \frac{\left\|\widetilde{A} - A^*\right\|_{L2}}{\|A^*\|_{L2}}, \tag{21}$$

where $\widetilde{A}$ is the recovered low-rank matrix. And the number of SVD's is used to evaluate the computational efficiency, since the running time of Algorithms 1 and 2 as well as [1, 3, 4, 7] is dominated by the SVD in each iteration. The experiments in this paper are conducted on a Lenovo computer equipped with an Intel Pentium (R) Core i5-3470 CPU (3.20 GHz) and 8 GB of RAM.

*3.2. Comparison between Algorithm 1 and RPCA [3].* In the literature, although several different numerical algorithms solving RPCA have been reported [3–6], the ALM method [3] is shown possessing the best performance in both accuracy and efficiency. Hence, this subsection compares Algorithm 1 with its convex and reduced version, that is, RPCA [3]. As implementing Algorithm 1, the parameters $\rho, \lambda, \varepsilon, K$ are uniformly set as $\rho = 1.5, \lambda = 1/\sqrt{500}, \varepsilon = 1e{-}7, K = 100$, and the parameter $\overline{\mu}$ is set as $\overline{\mu} = \mu^0 \cdot 1e{-}7$, where $\mu^0$ is set as $1.25/\widetilde{\sigma}$ and $\widetilde{\sigma}$ is the largest singular value of $C$. Besides, $B^0, X^0$ are both set as zero matrices. As for value choices of $p$ and $q$, we set them as 0.85 based on empirical studies, despite the fact that it may produce more accurate recovery with choices adaptive to different $r$ and spr.

Experimental results of Algorithm 1 and [3] are provided in Tables 1, 2, 3, and 4 corresponding to different settings. As the sparsity ratio spr equals 5%, it is obviously observed that Algorithm 1 performs perfectly in recovering the true rank of $A^*$ and is better than [3] in the term of RSE. It is also noticed that, as the sparsity ratio spr becomes larger, the recovery accuracy of both Algorithm 1 and [3] reduces, too. But it is still the case that Algorithm 1 behaves better than [3] no matter in the term of RSE or true rank recovery.

One more point should be claimed which is that slightly lower RSE's can be achieved by Algorithm 1 as setting $K = 200$. However, since the improvement in recovery accuracy is very limited, we just choose $K = 100$ for computational efficiency.

*3.3. Comparison between Algorithm 2 and RPCA [4].* As running Algorithm 2, the parameters $\eta, \lambda, \tau, J$ are uniformly

Table 1: Low-rank recovery with Algorithm 1 and [3] (spr = 5%).

| rank($A^*$) | RSE | | # SVD | | Rank ($\widetilde{A}$) | |
|---|---|---|---|---|---|---|
| $r$ | [3] | Algorithm 1 | [3] | Algorithm 1 | [3] | Algorithm 1 |
| 50 | $6.21e-6$ | $\mathbf{4.68e-6}$ | 20 | 30 | 48 | 50 |
| 100 | $8.31e-6$ | $\mathbf{5.99e-6}$ | 30 | 30 | 100 | 100 |
| 150 | $2.00e-3$ | $\mathbf{3.49e-5}$ | 30 | 100 | 270 | 150 |
| 200 | $7.03e-3$ | $\mathbf{3.85e-5}$ | 30 | 100 | 306 | 200 |

Table 2: Low-rank recovery with Algorithm 1 and [3] (spr = 10%).

| rank($A^*$) | RSE | | # SVD | | Rank ($\widetilde{A}$) | |
|---|---|---|---|---|---|---|
| $r$ | [3] | Algorithm 1 | [3] | Algorithm 1 | [3] | Algorithm 1 |
| 50 | $\mathbf{7.55e-6}$ | $9.22e-6$ | 30 | 30 | 50 | 50 |
| 100 | $3.32e-4$ | $\mathbf{2.81e-5}$ | 30 | 100 | 192 | 100 |
| 150 | $4.70e-3$ | $\mathbf{4.56e-5}$ | 30 | 100 | 309 | 150 |
| 200 | $9.30e-3$ | $\mathbf{1.30e-3}$ | 30 | 100 | 325 | 361 |

Table 3: Low-rank recovery with Algorithm 1 and [3] (spr = 15%).

| rank($A^*$) | RSE | | # SVD | | Rank ($\widetilde{A}$) | |
|---|---|---|---|---|---|---|
| $r$ | [3] | Algorithm 1 | [3] | Algorithm 1 | [3] | Algorithm 1 |
| 50 | $1.75e-5$ | $\mathbf{1.52e-5}$ | 30 | 30 | 50 | 50 |
| 100 | $2.50e-3$ | $\mathbf{3.87e-5}$ | 30 | 100 | 287 | 100 |
| 150 | $7.90e-3$ | $\mathbf{6.01e-5}$ | 30 | 100 | 328 | 151 |
| 200 | $1.23e-2$ | $\mathbf{3.60e-3}$ | 30 | 100 | 334 | 398 |

Table 4: Low-rank recovery with Algorithm 1 and [3] (spr = 20%).

| rank($A^*$) | RSE | | # SVD | | Rank ($\widetilde{A}$) | |
|---|---|---|---|---|---|---|
| $r$ | [3] | Algorithm 1 | [3] | Algorithm 1 | [3] | Algorithm 1 |
| 50 | $6.04e-5$ | $\mathbf{2.12e-5}$ | 30 | 30 | 176 | 50 |
| 100 | $5.70e-3$ | $\mathbf{4.35e-5}$ | 30 | 100 | 319 | 100 |
| 150 | $1.11e-2$ | $\mathbf{2.10e-3}$ | 30 | 100 | 334 | 333 |
| 200 | $1.56e-2$ | $\mathbf{6.60e-3}$ | 30 | 100 | 337 | 403 |

Table 5: Low-rank recovery with Algorithm 2 and [4] (spr = 5%).

| rank($A^*$) | RSE | | # SVD | | Rank ($\widetilde{A}$) | |
|---|---|---|---|---|---|---|
| $r$ | [4] | Algorithm 2 | [4] | Algorithm 2 | [4] | Algorithm 2 |
| 50 | $1.10e-3$ | $\mathbf{7.12e-4}$ | 100 | 100 | 50 | 50 |
| 100 | $1.10e-3$ | $\mathbf{4.99e-4}$ | 100 | 120 | 100 | 100 |
| 150 | $9.06e-4$ | $\mathbf{2.59e-4}$ | 100 | 120 | 150 | 150 |
| 200 | $2.90e-3$ | $\mathbf{1.61e-4}$ | 120 | 140 | 256 | 200 |

Table 6: Low-rank recovery with Algorithm 2 and [4] (spr = 10%).

| rank($A^*$) | RSE | | # SVD | | Rank ($\widetilde{A}$) | |
|---|---|---|---|---|---|---|
| $r$ | [4] | Algorithm 2 | [4] | Algorithm 2 | [4] | Algorithm 2 |
| 50 | $1.40e-3$ | $\mathbf{6.35e-4}$ | 100 | 120 | 50 | 50 |
| 100 | $1.40e-3$ | $\mathbf{4.29e-4}$ | 100 | 120 | 100 | 100 |
| 150 | $2.50e-3$ | $\mathbf{2.47e-4}$ | 120 | 120 | 241 | 150 |
| 200 | $7.80e-3$ | $\mathbf{2.91e-4}$ | 120 | 140 | 290 | 200 |

Table 7: Low-rank recovery with Algorithm 2 and [4] (spr = 15%).

| rank($A^*$) | RSE | | # SVD | | Rank ($\widetilde{A}$) | |
|---|---|---|---|---|---|---|
| $r$ | [4] | Algorithm 2 | [4] | Algorithm 2 | [4] | Algorithm 2 |
| 50 | $1.70e-3$ | $\mathbf{5.73e-4}$ | 100 | 120 | 50 | 50 |
| 100 | $2.10e-3$ | $\mathbf{3.36e-4}$ | 100 | 120 | 114 | 100 |
| 150 | $6.50e-3$ | $\mathbf{2.71e-4}$ | 100 | 140 | 283 | 150 |
| 200 | $1.21e-2$ | $\mathbf{7.00e-4}$ | 120 | 100 | 140 | 210 |

Table 8: Low-rank recovery with Algorithm 2 and [4] (spr = 20%).

| rank($A^*$) | RSE | | # SVD | | Rank ($\widetilde{A}$) | |
|---|---|---|---|---|---|---|
| $r$ | [4] | Algorithm 2 | [4] | Algorithm 2 | [4] | Algorithm 2 |
| 50 | $2.00e-3$ | $\mathbf{5.28e-4}$ | 100 | 120 | 70 | 50 |
| 100 | $4.10e-3$ | $\mathbf{3.31e-4}$ | 120 | 120 | 259 | 100 |
| 150 | $1.11e-2$ | $\mathbf{4.36e-4}$ | 120 | 140 | 297 | 150 |
| 200 | $1.62e-2$ | $\mathbf{2.10e-3}$ | 120 | 140 | 305 | 282 |

set as $\eta = 0.9 < 1, \lambda = 1/\sqrt{500}, \tau = 1e-7$, and $J = 200$, and the parameter $\overline{\nu}$ is set as $\overline{\nu} = \nu^0 \cdot 1e-9$, where $\nu^0$ is set as the largest singular value of $C$, that is, $\widetilde{\sigma}$. In addition, $A^{0/-1}, B^{0/-1}, \Lambda^0$, and $\Gamma^0$ are all set as zero matrices. As for $p$ and $q$, similar to the above manner, they are both set as 0.9 based on intensive empirical studies.

Experimental results of Algorithm 2 and [4] are provided in Table 5, 6, 7, and 8 corresponding to different settings. It is also remarkable that Algorithm 2 recovers the true rank of $A^*$ in almost all the scenarios, which is much superior to Algorithm 1. Its second advantage over Algorithm 1 is that it achieves slightly more robust recovery as $B^*$ is much grossly corruptedand$A^*$ is not sufficiently low-rank; for example, rank($A^*$) is 200. In spite of that, it is observed in other majority of cases that Algorithm 1 outperforms Algorithm 2 by $O(1e-1)$ in the term of RSE. Therefore, it can be concluded that both algorithms possess their own advantages and disadvantages, and on the whole, Algorithm 1 shows better performance in terms of both recovery accuracy and efficiency.

*3.4. Comparison between Proposed Approach and [7].* In the literature several nonconvex approaches for low-rank matrix recovery have also been proposed, for example, [7–9]. However, only [7] announces that it outperforms ALM-based RPCA [3] in the term of recovery accuracy.

Table 9 presents the RSE, number of SVD, and recovered rank achieved by [7] with sparsity ratios equal to 5% and 20%. Making comparison among Tables 1, 4, 5, 8, and 9, we can claim that both Algorithms 1 and 2 outperform [7] in terms of RSE and true rank recovery when $A^*$ is not sufficiently low-rank or $B^*$ is much grossly corrupted. In the meanwhile, we should also note that our method is computationally less efficient than [7] because of slightly more SVD's used in each iteration, which is one of the future works to be studied.

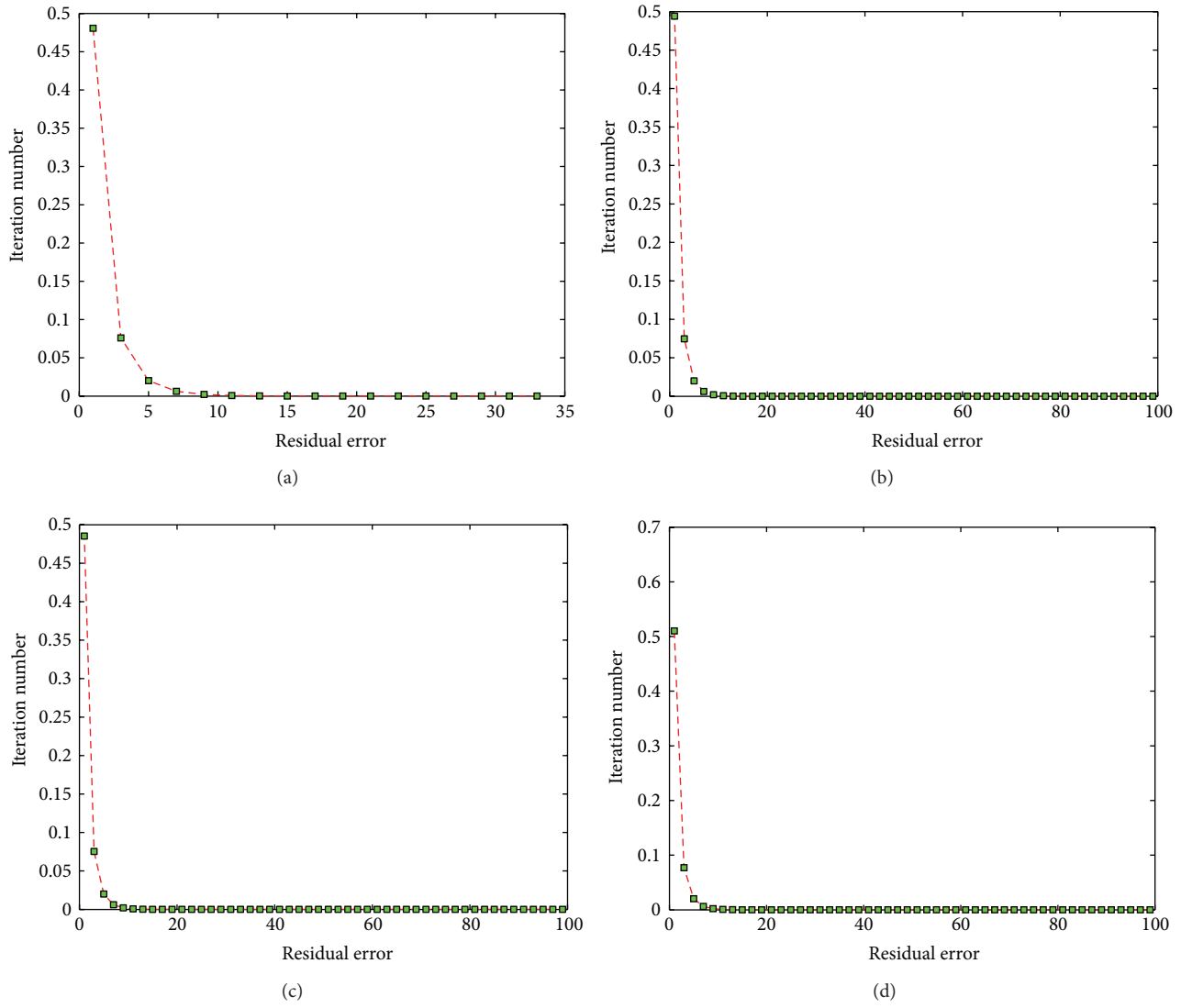


Figure 1: Residual error curves against iteration number for ALM-based Algorithm 1 (spr = 20%); (a) rank($A^*$) = 50, (b) rank($A^*$) = 100, (c) rank($A^*$) = 150, and (d) rank ($A^*$) = 200.

Table 9: Low-rank recovery with nonconvex approach [7].

| rank($A^*$) | RSE | | # SVD | | rank($\widetilde{A}$) | |
|---|---|---|---|---|---|---|
| $r$ | 5% | 20% | 5% | 20% | 5% | 20% |
| 50 | $8.29e-7$ | $2.91e-6$ | 40 | 60 | 50 | 50 |
| 100 | $9.97e-7$ | $1.67e-1$ | 68 | 70 | 100 | 195 |
| 150 | $7.39e-5$ | $3.94e-1$ | 72 | 73 | 419 | 259 |
| 200 | $1.19e-1$ | $5.36e-1$ | 73 | 73 | 471 | 266 |

*3.5. Empirical Analysis on the Convergence of Algorithms 1 and 2.* As mentioned earlier, existed convergence results on ALM and APG in [10, 11] are not applicable to problem (2) due to the usage of nonconvex $L_q$ seminorm and Schatten $p$-norm, which makes it difficult to conduct theoretical convergence analysis of the proposed algorithms, either. In spite of that, the empirical analysis can be made by plotting the residual error curve against iteration number for each algorithm. Specifically, the residual error curves are provided as the sparsity ratio equals 20% for both Algorithms 1 and 2, as shown, respectively, in Figures 1 and 2. It is obvious that the two deduced algorithms are of empirically fast convergence in each recovery scenario. Actually, this observation is also valid to other easier recovery cases with lower sparsity ratios. In addition, the number of iterations can be deduced from the residual error curves for each recovery problem.

## 4. Conclusions and Discussions

In this paper, a generalized robust minimization framework is proposed for low-rank matrix recovery by exploiting the Schatten $p$-norm ($0 < p \leq 1$) and $L_q$ ($0 < q \leq 1$) seminorm. And two numerical algorithms are deduced based

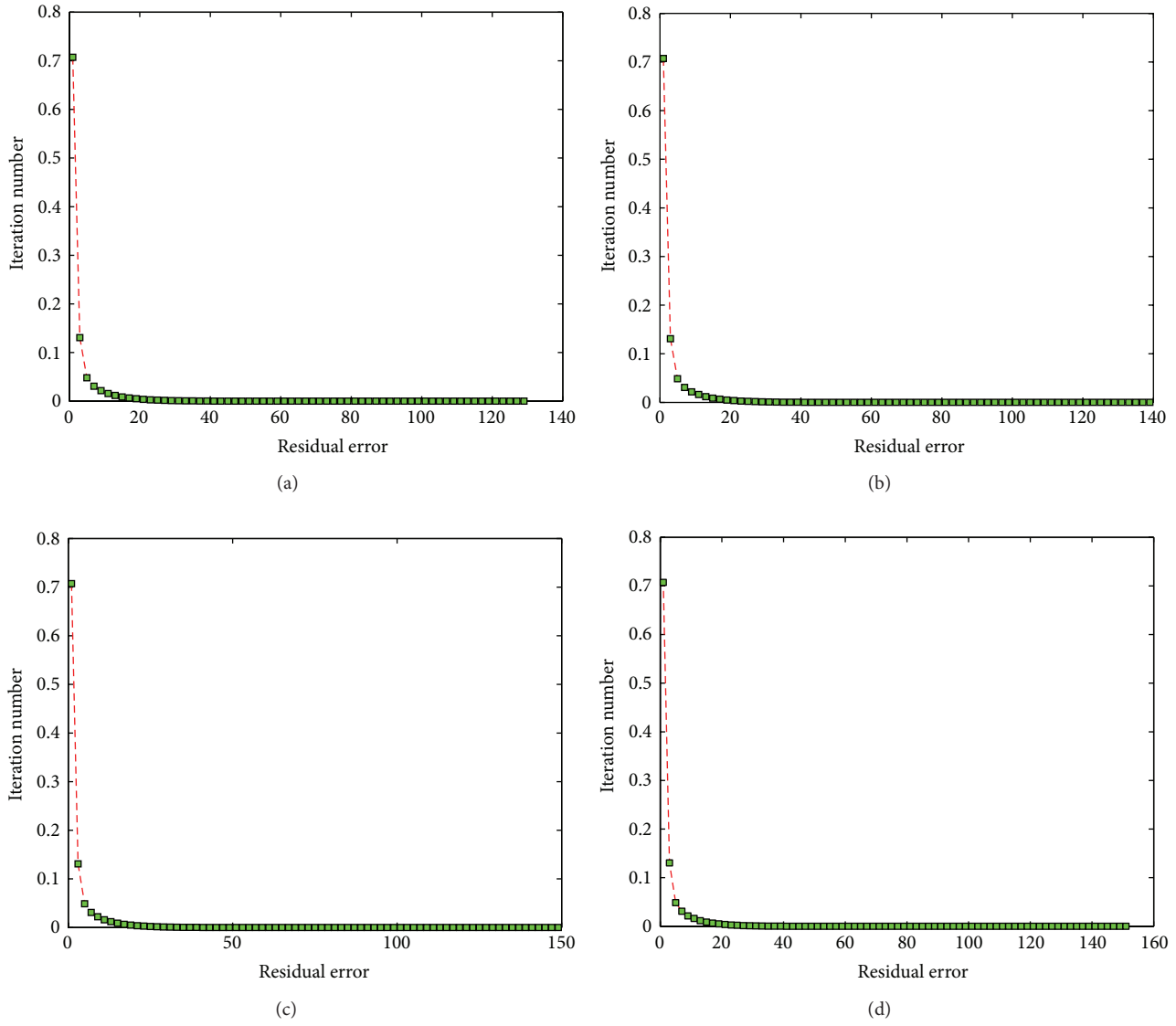


Figure 2: Residual error curves against iteration number for APG-based Algorithm 2 (spr = 20%); (a) rank($A^*$) = 50, (b) rank($A^*$) = 100, (c) rank($A^*$) = 150, and (d) rank($A^*$) = 200.

on the ALM and APG methods as well as efficient root-finder techniques. Experimental results demonstrate that the proposed algorithms possess their own advantages and disadvantages and both perform more effectively than state-of-the-art methods, either convex or nonconvex, in terms of both RSE and true rank recovery.

Note that this paper does not consider the influence of additive noise on the proposed algorithms, which actually corresponds to the problem of noisy RPCA [16, 17]. As claimed in [17], noisy RPCA is intrinsically different from the RPCA problem, that is, the focus of this paper. Indeed, the proposed algorithms in this paper are not quite robust to additive noise, just the same as many existing approaches to RPCA, for example, [1–4, 6–9]. To some degree, this observation coincides with the investigations in [18, 19]; that is, the $L_q$ seminorm as a sparsity-enforcing penalty is vulnerable against the influence of additive noise on the data, as it resembles the $L_0$ seminorm when $q$ approaches 0, in spite of the fact that $q$ in Algorithms 1 and 2 is chosen, respectively, as 0.85 and 0.9. Our future research topic is to extend the proposed algorithms to the noisy RPCA problem with applications to the field of image and vision computing.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgment

The authors would like to show gratitude to the anonymous reviewers for their serious, pertinent, and helpful comments.

Wen-Ze Shao is much grateful to Professor Zhihui Wei, Professor Yizhong Ma, and Dr. Min Wu for their kind supports in the past years. He also shows many thanks to Mr. Yatao Zhang and other kind people for helping him through his lost and sad years. The work is supported in part by the NSF of Jiangsu Province (BK20130868, BK20130883), the NSRF of Jiangsu Universities (13KJB510022, 13KJB120005), the TIF of NJUPT (NY212014, NY213007, NY213011, NY213066, and NY213139), and the NSF of China (61203270).

## References

[1] J. Wright, Y. Peng, Y. Ma, A. Ganesh, and S. Rao, "Robust principal component analysis: exact recovery of corrupted low-rank matrices by convex optimization," in *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS '09)*, pp. 2080–2088, December 2009.

[2] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM*, vol. 58, no. 3, article 11, 2011.

[3] Z. Lin, M. Chen, Y. Ma et al., "The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices," http://arxiv.org/abs/1009.5055.

[4] Z. Lin, A. Ganesh, J. Wright et al., "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," in *Proceedings of the International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, Aruba, Dutch Antilles, 2009.

[5] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[6] X. Yuan and J. Yang, "Sparse and low rank matrix decomposition via alternating direction method," *Pacific Journal of Optimization*, vol. 9, no. 1, pp. 167–180, 2013.

[7] S. Wang, D. Liu, and Z. Zhang, "Nonconvex relaxation approaches to robust matrix recovery," pp. 1764–1770, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, 2013.

[8] M. Fazel, H. Hindi, and S. P. Boyd, "Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices," in *Proceedings of the IEEE American Control Conference*, vol. 3, pp. 2156–2162, June 2003.

[9] A. Kyrillidis and V. Cevher, "Matrix ALPS: accelerated low rank and sparse matrix reconstruction," in *Proceedings of IEEE Statistical Signal Processing Workshop*, pp. 185–188, 2012.

[10] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 185–212, 2011.

[11] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Athena Scientific, 1996.

[12] J. von Neumann, "Some matrix-inequalities and metrization of matric-space," *Reviews of Tomsk Polytechnic University*, vol. 1, pp. 286–300, 1937.

[13] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-laplacian priors," in *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS '09)*, pp. 1033–1041, December 2009.

[14] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[15] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific Journal of Optimization*, vol. 6, no. 3, pp. 615–640, 2010.

[16] C. Qiu and N. Vaswani, "ReProCS: a missing link betweenrecursive robust PCA and recursive sparse recovery in large but correlated noise," http://arxiv.org/abs/1106.3286.

[17] T. Zhou and D. Tao, "GoDec: randomized low-rank & sparse matrix decomposition in noisy case," in *Proceedings of the 28th International Conference on Machine Learning (ICML '11)*, pp. 33–40, Bellevue, Wash, USA, July 2011.

[18] C. Hage and M. Kleinsteuber, "Robust PCA and subspace tracking from incomplete observations using L0-surrogates," *Computational Statistics*, 2013.

[19] D. Kong, M. Zhang, and C. Ding, "Minimal shrinkage for noisy data recovery using Schatten-$p$ norm objective," in *Machine Learning and Knowledge Discovery in Databases*, pp. 177–193, Springer, Berlin, Germany, 2013.