International Journal of Machine Learning and Cybernetics (2021) 12:3379–3395
https://doi.org/10.1007/s13042-021-01361-1

ORIGINAL ARTICLE

Performance guarantees of transformed Schatten-1 regularization for exact low-rank matrix recovery

Zhi Wang[1] · Dong Hu[1] · Xiaohu Luo[1] · Wendong Wang[2] · Jianjun Wang[3] · Wu Chen[1]

Received: 23 July 2020 / Accepted: 11 June 2021 / Published online: 29 June 2021

Abstract
Low-rank matrix recovery aims to recover a matrix of minimum rank that subject to linear system constraint. It arises in various real world applications, such as recommender systems, image processing, and deep learning. Inspired by compressive sensing, the rank minimization can be relaxed to nuclear norm minimization. However, such a method treats all singular values of target matrix equally. To address this issue, recently the transformed Schatten-1 (TS1) penalty function was proposed and utilized to construct low-rank matrix recovery models. Unfortunately, the method for TS1-based models cannot provide both convergence accuracy and convergence speed. To alleviate such problems, this paper further investigates the basic properties of TS1 penalty function. And we describe a novel algorithm, which we called ATS1PGA, that is highly efficient in solving low-rank matrix recovery problems at a convergence rate of $O(1/N)$, where N denotes the iterate count. In addition, we theoretically prove that the original rank minimization problem can be equivalently transformed into the TS1 optimization problem under certain conditions. Finally, extensive experimental results on real image data sets show that our proposed algorithm outperforms state-of-the-art methods in both accuracy and efficiency. In particular, our proposed algorithm is about 30 times faster than TS1 algorithm in solving low-rank matrix recovery problems.

Keywords Low-rank matrix recovery · Transformed Schatten-1 penalty function · Nonconvex model · Equivalence

1 Introduction

The problem of recovering a matrix of minimum rank subject to linear system constraint has attracted considerable attention in recent years. This problem arises in various real world applications, such as recommender systems [1, 2], image processing [3–5], quality-of-service (QoS) prediction [6], and deep learning [7, 8]. In general, such a task can be formulated as the following low-rank minimization problem [9, 10]:

$$\min_{X\in\mathbb{R}^{m\times n}} rank(X) \quad s.t. \quad \mathcal{A}(X) = b, \tag{1}$$

where X is the considered low-rank matrix in $\mathbb{R}^{m\times n}$, b is a given measurement in $\mathbb{R}^d$, and $\mathcal{A}$ denotes the linear transformation. By adopting the regularization method, the optimization problem (1) can be equivalently converted into the following unconstrained minimization problem:

$$\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{2} \,||\, \mathcal{A}(X) - b \,||_2^2 + \lambda rank(X), \tag{2}$$

where $\lambda > 0$ is a regularization parameter.

Unfortunately, the optimization problems (1) and (2) are computationally intractable due to the nonconvexity and discontinuous properties of the rank function. In order to overcome this difficulty, many researchers suggested to use the nuclear norm instead, which is known as the tightest convex proxy of the rank function [11, 12]. Theoretical analysis shows that, under some mild conditions, the low-rank matrix can be exactly recovered with high probability by using this

✉ Zhi Wang
chiw@swu.edu.cn

✉ Wu Chen
chenwu@swu.edu.cn

Jianjun Wang
wjj@swu.edu.cn

[1] College of Computer and Information Science, Southwest University, Chongqing 400715, People's Republic of China

[2] College of Artificial Intelligence, Southwest University, Chongqing 400715, People's Republic of China

[3] School of Mathematics and Statistics, Southwest University, Chongqing 400715, People's Republic of China

scheme. Thus, a large number of methods have been proposed for the resultant nuclear norm optimization problem, such as singular value thresholding (SVT) [13], accelerated proximal gradient with linesearch algorithm (APGL) [14], and accelerated inexact soft-impute (AIS-Impute) [15].

Since the methods mentioned above are simple and easy to use with theoretical guarantee, nuclear norm based model has recently attracted significant attention in the field of low-rank matrix recovery. However, the performance of such a convex relaxation is not good enough. In other words, the solutions of nuclear norm optimization problem may deviate from the solutions of the original optimization problem. The main reason is that the nuclear norm based model over-penalizes large singular values. To alleviate this limitation, a common used strategy is to use nonconvex surrogates to approximate the rank function, which make closer approximation than nuclear norm. Examples of these nonconvex surrogate functions include l_q-norm ($0 < q < 1$) [16–18], weighted nuclear norm (WNN) [19], smoothly clipped absolute deviation (SCAD) [20], mini-max concave penalty (MCP) [21], log-sum penalty (LSP) [22], and so on. Despite the resultant problem is nonconvex, non-smooth, and even non-Lipschitz, numerous methods have been proposed to handle it. In [16] and [23], the authors proposed fixed point iterative scheme with the singular value thresholding operator. The convergence analysis and empirical results show that these methods are fast and efficient. In [24], the iteratively reweighted nuclear norm (IRNN) method has been proposed by using the concavity and decreasing supergradients property of existing nonconvex regularizer. Since a computationally expensive singular value decomposition (SVD) step is involved in per iteration, the IRNN method converges slowly. In order to improve the speed and performance of IRNN method, fast nonconvex low-rank (FaNCL) [25] method was proposed. The empirical results of all the methods mentioned above illustrate that the nonconvex based model outperforms the convex based model.

Recently, the transformed Schatten-1 (TS1) penalty function [26], as a matrix quasi-norm defined on its singular values, has been successfully applied to low-rank matrix recovery. Actually, the TS1 penalty function is extended from the Transformed ℓ_1 (TL1) function. The TL1 function can be seen as a class of ℓ_1 based nonconvex penalty function, which was generalized by Lv and Fan in [27]. Kang et al. [28] have demonstrated the very high efficiency of TL1 function when applied to robust principal component analysis. However, the TL1 penalty function leads to a nonconvex optimization problem that is difficult to solve fast and efficient. Therefore, Zhang et al. continue such a study [29, 30] and point out that the TL1 proximal operator has closed form analytical solutions for all values of parameter. Based on this finding, in this paper, we consider the following TS1 penalty function:

$$T(X) = \sum_{i=1}^{rank(X)} \rho_a(\sigma_i(X)) = \sum_{i=1}^{rank(X)} \frac{(a+1)\sigma_i(X)}{a+\sigma_i(X)}, \tag{3}$$

where

$$\rho_a(|x|) = \frac{(a+1)|x|}{a+|x|}, \tag{4}$$

is a nonconvex function with parameter $a \in (0, +\infty)$, and $\sigma_i(X)$ denotes the ith singular value of matrix X. Therefore, the original problem (1) can be naturally converted into the following optimization problem:

$$\min_{X\in\mathbb{R}^{m\times n}} T(X) \quad s.t. \quad \mathcal{A}(X) = b. \tag{5}$$

More often, we focus on its regularization version, which can be formulated as:

$$\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{2} \,\|\, \mathcal{A}(X) - b \,\|_2^2 + \lambda T(X). \tag{6}$$

It should be noted that the TS1 penalty function is more general than the nonconvex penalty function in [31]. Besides, ρ_a with $a \in (0, +\infty)$ provides solutions satisfying the unbiasedness and low-rankness.

In this paper, we further investigate the basic properties of TS1 penalty function and propose a fast and efficient algorithm to solve problem (6). Specifically, we analyze the solution of rank minimization and prove that, under certain rank-RIP conditions, rank minimization can be equivalently transformed into unconstrained TS1 regularization, whose global minimizer can be obtained by TS1 thresholding functions. The TS1 thresholding functions are in closed analytical form for all parameter values. Thus, an exact mathematical analysis provides theoretical guarantee for the application of TS1 regularization in solving low-rank matrix recovery problems. We list the findings and contributions as follows:

(1) We first show that the minimizer of problem (5) is actually the optimal solution of problem (1).
(2) We further establish the relationship between problems (5) and (6), and prove that the solution of problem (5) can be obtained by solving problem (6).
(3) Nesterov's rule and inexact proximal strategies are adopted to achieve a novel algorithm highly efficient in solving the TS1 regularization at a convergence rate of $O(1/N)$.
(4) Extensive empirical studies regarding image inpainting tasks validate the advantages of our proposed method.

The remainder of this paper is organized as follows. Section 2 introduces the related works. In Sect. 3, the equivalence minimizers of the resultant optimization problem and the original optimization problem are established. Section 4

proposes an efficient optimization method with rigorous convergence guarantee. The experimental results are reported and analyzed in Sect. 5. Finally, we conclude this paper in Sect. 6.

2 Background

2.1 Proximal algorithms

In this paper, we consider the following low-rank optimization problem:

$$\min_{X\in\mathbb{R}^{m\times n}} F(X) = f(X) + \lambda h(X), \tag{7}$$

where f is bounded below and differentiable with L_f-Lipschitz continuous gradient, i.e., $\| \nabla f(X_1) - \nabla f(X_2) \|_F \leq L_f \| X_1 - X_2 \|_F$, h is low-rank regularizer, and $\lambda > 0$ is a given parameter. During the last decades, the proximal algorithm [32] has been received considerable attention and successfully applied to solve problem (7). Specifically, the proximal algorithm generates X_{k+1} as

$$X_{k+1} = prox_{\frac{\lambda}{\tau}h}\Big(X_k - \frac{1}{\tau}\nabla f(X_k)\Big), \tag{8}$$

where $\tau > 0$ is a parameter, and

$$prox_{\frac{\lambda}{\tau}h}(M) = \arg\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{2}\| X - M \|_F^2 + \frac{\lambda}{\tau}h(X) \tag{9}$$

denotes the proximal operator. Assume that f and h are convex, Toh and Yun [14] proposed an accelerated proximal gradient (APG) algorithm with a converge rate of $O(1/N^2)$. However, exactly solving the proximal operator may be expensive. With the aim of alleviating this difficulty, the inexact proximal gradient methods have been proposed and the theoretical analysis in [33] reveals that it shares the same convergence rate as APG. Most recently, the inexact proximal gradient method has been extended to nonconvex and nonsmooth problems [34]. It should be pointed out that the nmAIPG algorithm in [34] is nearly the same as the nmAPG algorithm in [35], but it is much faster. Unfortunately, both nmAIPG and nmAPG algorithms may involve two proximal steps in per iteration.

2.2 TS1 thresholding algorithm

We first define the proximal operator with TS1 regularizer, which can be found as follows:

$$prox_{\lambda}^{\rho_a}(y) = \arg\min_{x\in\mathbb{R}} \frac{1}{2}(y-x)^2 + \lambda\rho_a(|x|), \tag{10}$$

where $\rho_a(\cdot)$ is defined in (4). As shown in [30], this nonconvex function has a closed-form expression for its optimal solution and the following lemma addresses this issue.

Lemma 1 (see [30]) *For any $\lambda > 0$ and $y \in \mathbb{R}$, the solutions to nonconvex function (10) are*

$$prox_{\lambda}^{\rho_a}(y) = \begin{cases} 0, & \text{if } |y| \leq t \\ g_\lambda(y), & \text{if } |y| > t \end{cases} \tag{11}$$

where

$$g_\lambda(y) = sgn(y)\left\{\frac{2}{3}(a+|y|)cos\left(\frac{\phi(y)}{3}\right) - \frac{2a}{3} + \frac{|y|}{3}\right\}$$

with $\phi(y) = arccos(1 - \frac{27\lambda a(a+1)}{2(a+|y|)^3})$, and

$$t = \begin{cases} \frac{\lambda(a+1)}{a}, & \text{if } \lambda \leq \frac{a^2}{2(a+1)} \\ \sqrt{2\lambda(a+1)} - \frac{a}{2}, & \text{if } \lambda > \frac{a^2}{2(a+1)}. \end{cases} \tag{12}$$

Based on this finding, the optimal solutions to problem (6) can be obtained by the proximal operator.

Lemma 2 (see [26]) *Assume that $\tau > \| \mathcal{A} \|_2^2$, the optimal solutions to problem (6) are*

$$X^* = prox_{\frac{\lambda}{\tau}T(\cdot)}(B_\tau(X^*)) = UDiag\left(prox_{\frac{\lambda}{\tau}}^{\rho_a}(\sigma_i(B_\tau(X^*)))\right)V^T, \tag{13}$$

where $B_\tau(X^*) = X^* - \frac{1}{\tau}\mathcal{A}^*(\mathcal{A}(X^*) - b)$ and it admits SVD as $UDiag(\sigma(B_\tau(X^*)))V^T$.

Therefore, at kth iteration

$$X_{k+1} = prox_{\frac{\lambda}{\tau}T(\cdot)}\Big(X_k - \frac{1}{\tau}\mathcal{A}^*(\mathcal{A}(X_k) - b)\Big). \tag{14}$$

According to iteration (14), the TS1 algorithm is proposed.

3 Equivalence minimizers of problem (1) and problem (5)

In this section, we further investigate the basic properties of TS1 penalty function. Assume that X_0 and X_1 be the minimizers to the problems (1) and (5), respectively. Let $R = X_1 - X_0$, then partition matrix R into two matrices R_0 and R_c which are defined as follows.

Definition 1 Let R admits SVD as $UDiag(\sigma(R))V^T$, the matrices R_0 and R_c can be defined as:

$$R_0 = [U_{2K}\ 0]_{m\times m}\begin{bmatrix}\Sigma_1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}_{m\times n}[V_{2K}\ 0]_{n\times n}^T \tag{15}$$

and

$$R_c = [0\ U_{m-2K}]_{m\times m}\begin{bmatrix}0 & 0 & 0\\ 0 & \Sigma_2 & 0\end{bmatrix}_{m\times n}[0\ V_{m-2K}]^T_{n\times n}. \tag{16}$$

Definition 2 Define the index set $I_j = \{P(j-1)+2K+1,\dots,Pj+2K\}$ and partition R_c into a sum of matrices $R_1, R_2, \dots$, i.e.,

$$R_c = \sum_j R_j,$$

where

$$R_j = [0\ U_{I_j}\ 0]_{m\times m}\begin{bmatrix}0 & 0 & 0\\ 0 & \Sigma_{I_j} & 0\\ 0 & 0 & 0\end{bmatrix}_{m\times n}[0\ V_{I_j}\ 0]^T_{n\times n}. \tag{17}$$

First, we introduce the following useful results.

Lemma 3 *Assume that $K = rank(X_0)$, we have*

$$T(R_c) \le T(R_0). \tag{18}$$

Proof Since X_1 is the minimizer of (5), we have

$$\begin{aligned}T(X_0) \ge T(X_1) =& T(X_0+R) \ge T(X_0+R_c) - T(R_0)\\ =& T(X_0) + T(R_c) - T(R_0),\end{aligned}$$

the second inequality follows from Lemma 2.1 in [30] and the last equality follows from Lemma 3.4 in [12]. □

Theorem 1 *For any $a > 0$, we have*

$$\| R_0 + R_1 \|_F \ge \frac{aT(R_0)}{(a+1)\sqrt{2K}}. \tag{19}$$

Proof According to the definition of $\rho_a(\cdot)$, we have

$$\rho_a(|x|) = \frac{(a+1)|x|}{a+|x|} = \frac{(a+1)|x|}{a} - \frac{(a+1)x^2}{a(a+|x|)} \le \frac{(a+1)|x|}{a}.$$

Hence, we get

$$\begin{aligned}T(R_0) = \sum_i \frac{(a+1)\sigma_i(R_0)}{a+\sigma_i(R_0)} \le \frac{a+1}{a}\sum_i \sigma_i(R_0) = \frac{a+1}{a}\|\sigma(R_0)\|_1\\ \le \frac{a+1}{a}\sqrt{2K}\|R_0\|_F \le \frac{a+1}{a}\sqrt{2K}\|R_0+R_1\|_F.\end{aligned}$$

Thus, we get the desired result

$$\| R_0 + R_1 \|_F \ge \frac{aT(R_0)}{(a+1)\sqrt{2K}}.$$

□

Lemma 4 *Let $X \in \mathbb{R}^{m\times n}$ admits SVD as $UDiag(\sigma(X))V^T$. For any $a > 0$ and $\alpha > \alpha_1$, where*

$$\alpha_1 = \frac{ar + r - 1}{a}\sigma_1(X) \tag{20}$$

with $r = rank(X)$, we have

$$T(\alpha^{-1}X) \le 1. \tag{21}$$

Proof Using the fact that $\rho_a(x)$ is increasing in the range $[0, +\infty)$, we have

$$\begin{aligned}T(\alpha^{-1}X) =& \sum_{i=1}^{r}\rho_a(\sigma_i(\alpha^{-1}X)) \le r\rho_a(\sigma_1(\alpha^{-1}X))\\ =& \frac{\frac{1}{\alpha}r(a+1)\sigma_1(X)}{a+\frac{1}{\alpha}\sigma_1(X)} = \frac{r(a+1)\sigma_1(X)}{\sigma_1(X)+a\alpha}.\end{aligned}$$

To get $T(\alpha^{-1}X) \le 1$, it suffices to impose

$$\frac{r(a+1)\sigma_1(X)}{\sigma_1(X)+a\alpha} \le 1,$$

equivalently,

$$\alpha \ge \frac{ar + r - 1}{a}\sigma_1(X).$$

We complete the proof. □

Theorem 2 *For any $a > 0$ and $\alpha > \frac{ar_c+r_c-1}{a}\sigma_1(R_c)$, we have*

$$\sum_{i\ge 2}\|\alpha^{-1}R_i\|_F \le \sum_{i\ge 2}\frac{T(\alpha^{-1}R_{i-1})}{\sqrt{P}} \le \frac{T(\alpha^{-1}R_0)}{\sqrt{P}}, \tag{22}$$

where $r_c = rank(R_c)$.

Proof According to the definition of $T(\cdot)$ and Lemma 4, we have

$$\rho_a(\sigma_i(\alpha^{-1}R_j)) \le T(\alpha^{-1}R_j) \le 1,\ \forall i \in I_j.$$

Hence, we have

$$\rho_a(\sigma_i(\alpha^{-1}R_j)) = \frac{(a+1)\sigma_i(\alpha^{-1}R_j)}{a+\sigma_i(\alpha^{-1}R_j)} \le 1 \Leftrightarrow \sigma_i(\alpha^{-1}R_j) \le 1,$$

which is equivalent to

$$\sigma_i(\alpha^{-1}R_j) \le \rho_a(\sigma_i(\alpha^{-1}R_j)).$$

By Definition 2, we have

$$\sigma_i(R_j) \le \sigma_{i'}(R_{j-1}),\ \forall i \in I_j, i' \in I_{j-1},$$

and the rank of R_js is not greater than k.

Since $\rho_a(t)$ is increasing in the range $(0, \infty)$, we get

$$\sigma_i(\alpha^{-1}R_j) \le \rho_a(\sigma_i(\alpha^{-1}R_j)) \le \frac{T(\alpha^{-1}R_{j-1})}{P}.$$

Therefore, we obtain

$$\| \alpha^{-1}R_j \|_F \le \frac{T(\alpha^{-1}R_{j-1})}{\sqrt{P}},$$

and

$$\sum_{j\ge 2} \| \alpha^{-1}R_j \|_F \le \sum_{j\ge 2} \frac{T(\alpha^{-1}R_{j-1})}{\sqrt{P}}.$$

This is the first part of (22). By Lemma 3, the second part of (22)is obtained immediately.

We complete the proof. □

In order to show that the minimizers to problem (5) are the optimal solutions of the problem (1), we introduce the following rank restricted isometry property (rank-RIP) condition.

Definition 3 (*see* [36]) For every integer r with $1 \le r \le m$, let the restricted isometry constant $\delta_r(\mathcal{A})$ be the minimum number such that

$$(1-\delta_r(\mathcal{A}))\| X \|_F^2 \le \| \mathcal{A}(X) \|_2^2 \le (1+\delta_r(\mathcal{A}))\| X \|_F^2, \tag{23}$$

holds for all $X \in \mathbb{R}^{m\times n}$ of rank at most r.

Armed with rank-RIP condition, the following theorem reveals that the original problem (1) and its nonconvex relaxation problem (5) share the same solutions under the rank-RIP condition with $\delta_{3K} < \frac{a^2-4a-2}{5a^2+4a+2}$.

Theorem 3 *Assume that there is a number $P > 2K$, where $K = rank(X_0)$, such that*

$$\delta_P(\mathcal{A}) + \frac{P}{2K}\delta_{2K+P}(\mathcal{A}) < \frac{P}{2K} - 1. \tag{24}$$

Then, there exists $a^* > 0$, such that for any $a^* < a < +\infty$, $X_1 = X_0$.

Proof Let

$$f(a) = \frac{a^2}{(a+1)^2}\frac{P}{2K}\big(1-\delta_{2K+P}(\mathcal{A})\big) - \big(1+\delta_P(\mathcal{A})\big). \tag{25}$$

It is easy to verify that the function f is continuous and increasing in the range $[0, +\infty)$. Note that at $a = 0$,

$$f(0) = -1 - \delta_P(\mathcal{A}) < 0,$$

and

$$f(a) = \frac{P}{2K}\big(1-\delta_{2K+P}(\mathcal{A})\big) - \big(1+\delta_P(\mathcal{A})\big) > 0, \quad as \quad a \to +\infty.$$

Thus, there exists a constant $a^* > 0$ such that $f(a^*) = 0$. Then, for any $a^* < a < +\infty$, we get

$$\frac{a}{a+1}\sqrt{\frac{1-\delta_{2K+P}(\mathcal{A})}{2K}} - \sqrt{\frac{1+\delta_P(\mathcal{A})}{P}} > 0. \tag{26}$$

Since $\mathcal{A}(R) = \mathcal{A}(X_1 - X_0) = b - b = 0$, we obtain

$$\begin{aligned}
0 = \| \mathcal{A}(\alpha^{-1}R) \|_2 &= \| \mathcal{A}(\alpha^{-1}R_0 + \alpha^{-1}R_c) \|_2 \\
&= \| \mathcal{A}(\alpha^{-1}R_0 + \alpha^{-1}R_1) + \sum_{i\ge 2}\mathcal{A}(\alpha^{-1}R_i) \|_2 \\
&\ge \| \mathcal{A}(\alpha^{-1}R_0 + \alpha^{-1}R_1) \|_2 - &\#XArrowvert; \sum_{i\ge 2}\mathcal{A}(\alpha^{-1}R_i) &\#XArrowvert;_2 \\
&\ge \| \mathcal{A}(\alpha^{-1}R_0 + \alpha^{-1}R_1) \|_2 - \sum_{i\ge 2} \| \mathcal{A}(\alpha^{-1}R_i) \|_2 \\
&\ge \sqrt{1-\delta_{2K+P}(\mathcal{A})} \| \alpha^{-1}R_0 + \alpha^{-1}R_1 \|_F - \sqrt{1+\delta_P(\mathcal{A})} \sum_{i\ge 2} \| \alpha^{-1}R_i \|_F .
\end{aligned}$$

Using Theorems 1 and 2, we get

$$0 \geq \sqrt{1-\delta_{2K+P}(\mathcal{A})}\frac{aT(\alpha^{-1}R_0)}{(a+1)\sqrt{2K}} - \sqrt{1+\delta_P(\mathcal{A})}\frac{1}{\sqrt{P}}T(\alpha^{-1}R_0)$$
$$= \left(\frac{a}{a+1}\sqrt{\frac{1-\delta_{2K+P}(\mathcal{A})}{2K}} - \sqrt{\frac{1+\delta_P(\mathcal{A})}{P}}\right)T(\alpha^{-1}R_0).$$

According to the inequality (26) and the definition of $T(\cdot)$, we have $T(\alpha^{-1}R_0) = 0$, which implies that $R_0 = 0$. This, together with Lemma 3, yields $R_c = 0$. Therefore, $X^* = X_0$.

We complete the proof. □

Remark 1 In Theorem 3, if let $P = 3K$, the inequality (26) is

$$\frac{3a^2}{2(a+1)^2}(1-\delta_{5K}(\mathcal{A})) > 1+\delta_{3K}(\mathcal{A}). \tag{27}$$

This inequality will approach $\delta_{3T} < \frac{a^2-4a-2}{5a^2+4a+2}$. Moreover, we can obtain $\delta_{3T} < 0.2$ as $a \to \infty$.

Although the minimizers of problem (1) can be exactly obtained by solving the nonconvex relaxation problem (5), the resultant optimal problem is still NP-hard. To overcome this difficultly, we focus on its regularization version (6). The following theorem addresses this issue.

Theorem 4 *Let $\{\lambda_p\}$ be a decreasing sequence of positive numbers with $\lambda_p \to 0$, and X_{λ_p} be the optimal minimizer of the problem (6) with $\lambda = \lambda_p$. Assume that the problem (5) is feasible, then the sequence $\{X_{\lambda_p}\}$ is bounded and any of its accumulation points is the optimal minimizer of the problem (5).*

Proof Assume that the problem (5) is feasible and $\tilde{X}$ is any feasible point, then $\mathcal{A}(\tilde{X}) = b$. Since X_{λ_p} is the optimal minimizer of the problem (6) with $\lambda = \lambda_p$, we get

$$\begin{aligned}\max\left\{\lambda_p T(X_{\lambda_p}), \| \mathcal{A}(X_{\lambda_p}) - b \|_2^2\right\} &\leq \| \mathcal{A}(X_{\lambda_p}) - b \|_2^2 + \lambda_p T(X_{\lambda_p}) \\ &\leq \| \mathcal{A}(\tilde{X}) - b \|_2^2 + \lambda_p T(\tilde{X}) \\ &= \lambda_p T(\tilde{X}).\end{aligned} \tag{28}$$

Thus, the sequence $\{X_{\lambda_p}\}$ is bounded and has at least one accumulation point. Assume that X^* be any accumulation point of $\{X_{\lambda_p}\}$. From (28), we get $\mathcal{A}(X^*) = b$, which means that X^* is a feasible point of the problem (5). Together with $T(X^*) \leq T(\tilde{X})$ and the arbitrariness of $\tilde{X}$, we get that X^* is the optimal minimizer of the problem (5).

We complete the proof. □

4 The proposed algorithm and its convergence analysis

In order to solve the nonconvex relaxation problem (6) efficiently, in this section we propose a novel algorithm highly efficient to handle it at a convergence rate of $O(1/N)$. Specifically, the proposed algorithm has an improved convergence rate and improved recovery capability of the low-rank matrix over those of the state-of-the-art algorithms.

First, the power method [37] is adopted to obtain approximate SVD. As shown in [33] and [34], in real application it is often too expensive to compute the proximal operator. To speed up the convergence of the proposed algorithm, it is desirable to solve the proximal operator by computing the SVD on a smaller matrix. The following lemma addresses this issue.

Lemma 5 *For any fixed $\lambda > 0$, assume that $X = QQ^TX \in \mathbb{R}^{m\times n}$, where $Q \in \mathbb{R}^{m\times t}(t \ll n)$ is an orthogonal matrix. Then,*

$$prox_{\frac{\lambda}{\tau}T(\cdot)}(X) = Q prox_{\frac{\lambda}{\tau}T(\cdot)}(Q^TX). \tag{29}$$

Proof The proof can be followed the footsteps of Proposition 1 in [38], and we omit it here. □

Since the power method is employed in our algorithm to get approximate SVD, the results generated by the proximal operator may be inexact, meaning that

$$\begin{aligned}X_{k+1} &= prox_{\frac{\lambda}{\tau}T(\cdot)}(Y_k) \\ &= \Big\{U \,|\, \frac{\lambda}{\tau}T(U) + \frac{1}{2}\| U - Y_k \|_F^2 \leq \xi_k + \frac{\lambda}{\tau}T(V) \\ &\quad + \frac{1}{2}\| V - Y_k \|_F^2,\ \forall V \in \mathbb{R}^{m\times n}\Big\}.\end{aligned} \tag{30}$$

As indicated in [34], although the inexact proximal steps are employed in our algorithm, the basic properties stay the same.

Second, the convergence rate of our proposed algorithm is further accelerated by the Nesterov's rule [39] which is a widely used method to speed up the convergence rate of first-order algorithm. In this paper, we integrate APGnc$^+$ [40] with our proposed algorithm, and the details can be found in Algorithm 1. It should be pointed out that our proposed algorithm ATS1PGA is much different from APGnc$^+$. We focus on the transformed Schatten-1 regularizer which induces lower rank and achieves improved recovery performance than the nonconvex regularizer used in APGnc$^+$.

Algorithm 1 The Accelerated Transformed Achatten-1 Proximal Gradient Algorithm (ATS1PGA)

Input: $\mathscr{A}:\mathbb{R}^{m\times n}\mapsto\mathbb{R}^d$, $R\in\mathbb{R}^{n\times t}$, $b\in\mathbb{R}^d$, $\lambda>0$, $\tau\geq\|\mathscr{A}\|_2^2$, $\alpha,\beta\in(0,1)$.

Initialize: $X_1=Y_1=0$, $\lambda>0$;
for $k=1:K$ **do**
  $Y=Y_k$;
  $B=Y-\frac{1}{\tau}\mathscr{A}^*\mathscr{A}(Y)+\frac{1}{\tau}\mathscr{A}^*(b)$;
  $Q=\text{PowerMethod}(\text{B},\text{R})$;
  $Y=Qprox_{\frac{\lambda}{\tau}T(\cdot)}(Q^TB)$;
  $r=rank(Y)$, $X_{k+1}=Y$;
  $\lambda=\frac{2(\sigma_{r+1}(X_{k+1}))^2}{1+2\sigma_{r+1}(X_{k+1})}$, $a=\frac{\lambda}{\tau}+\sqrt{(\frac{\lambda}{\tau})^2+\frac{2\lambda}{\tau}}$, $t=\frac{\lambda(a+1)}{\tau a}$;
  $V_{k+1}=X_{k+1}+\beta(X_{k+1}-X_k)$;
  if $F_\lambda(X_{k+1})\leq F_\lambda(V_{k+1})$ **then**
    $Y_{k+1}=X_{k+1}$, $\beta\leftarrow\alpha\beta$;
  else
    $Y_{k+1}=V_{k+1}$, $\beta\leftarrow\min\{\frac{\beta}{\alpha},1\}$;
  end if
end for
return X_K

The following theorem shows that the objective function is always decreased and any accumulation point of the sequence generated by Algorithm 1 is a stationary point.

Theorem 5 *Let* $F(X)=\frac{1}{2}\|\mathcal{A}(X)-b\|_2^2+\lambda T(X)$ *and* $\{X_k\}$ *be the sequence generated via Algorithm 1 with* $\tau\geq\|\mathcal{A}\|_2^2$. *Assume that* $\xi_k\leq\delta\|X_{k+1}-Y_k\|_F^2$ *and* $\frac{\tau}{2}-\frac{\|\mathcal{A}\|_2^2}{2}-\tau\delta>0$, *then*

(1) *the sequence* $\{X_k\}$ *is bounded, and has at leat one accumulation point*;
(2) $F(X)$ *is monotonically decreasing and converges to* $F(X^*)$, *where* X^* *is any accumulation point of* $\{X_k\}$;
(3) $\lim_{k\to\infty}\|X_{k+1}-Y_k\|_F^2=0$;
(4) X^* *is a stationary point of* (6).

To prove Theorem 5, we first introduce the following lemma.

Lemma 6 *If* $\xi_k\leq\delta\|A-Y_k\|_F^2$, *then*

$$F(A)\leq F(Y_k)-\left(\frac{\tau}{2}-\frac{\|\mathcal{A}\|_2^2}{2}-\tau\delta\right)\|A-Y_k\|_F^2, \quad (31)$$

where $A=prox_{\frac{\lambda}{\tau}T(\cdot)}(Y_k-\frac{1}{\tau}\nabla f(Y_k))$.

Proof Let $f(A)=\frac{1}{2}\|\mathcal{A}(A)-b\|_2^2$, we have

$$f(A)\leq f(Y_k)+<A-Y_k,\nabla f(Y_k)>+\frac{\|\mathcal{A}\|_2^2}{2}\|A-Y_k\|_F^2.$$

According to the definition of the inexact proximal operator for each step, we have that

$$\frac{\lambda}{\tau}T(A)+\frac{1}{2}\|A-Y_k+\frac{1}{\tau}\nabla f(Y_k)\|_F^2 \leq\xi_k+\frac{\lambda}{\tau}T(Y_k)+\frac{1}{2}\|\frac{1}{\tau}\nabla f(Y_k)\|_F^2,$$

and thus we can simplify it as

$$\lambda T(A)\leq\tau\xi_k+\lambda T(Y_k)-\frac{\tau}{2}\|A-Y_k\|_F^2-<A-Y_k,\nabla f(Y_k)>.$$

Therefore, we have that

$$\begin{aligned}F(A)=&f(A)+\lambda T(A)\\ \leq&f(Y_k)+<\nabla f(Y_k),A-Y_k>+\frac{\|\mathcal{A}\|_2^2}{2}\|A-Y_k\|_F^2\\ &+\lambda T(Y_k)-<\nabla f(Y_k),A-Y_k>-\frac{\tau}{2}\|A-Y_k\|_F^2+\tau\xi_k\\ =&F(Y_k)-\left(\frac{\tau-\|\mathcal{A}\|_2^2}{2}\right)\|A-Y_k\|_F^2+\tau\xi_k\\ \leq&F(Y_k)-\left(\frac{\tau}{2}-\frac{\|\mathcal{A}\|_2^2}{2}-\tau\delta\right)\|A-Y_k\|_F^2.\end{aligned}$$

We complete the proof. □

Now, we begin prove Theorem 5.

Proof (1) and (2) By applying Lemma 6, we obtain that

$$F(X_{k+1})\leq F(Y_k)-\left(\frac{\tau}{2}-\frac{\|\mathcal{A}\|_2^2}{2}-\tau\delta\right)\|X_{k+1}-Y_k\|_F^2. \quad (32)$$

Since $\frac{\tau}{2}-\frac{\|\mathcal{A}\|_2^2}{2}-\tau\delta>0$, it follows that $F(X_{k+1})\leq F(Y_k)$. Besides, according to the the update rule of Algorithm 4.1,

we have $F(Y_{k+1}) \leq F(X_{k+1})$. In summary, for all k the following inequality holds:

$$F(Y_{k+1}) \leq F(X_{k+1}) \leq F(Y_k) \leq F(X_k), \tag{33}$$

which shows that $\{F(X_k)\}$ is monotonically decreasing and converges to a constant C^*. From $\{X_k\} \subset \{X : F(X) \leq F(X_0)\}$ which is bounded, it follows that $\{X_k\}$ is bounded and, therefore, there is at least one accumulation point X^*. Moreover, using the continuity and monotonicity of $\{F(\cdot)\}$, $F(X_k) \to F^* = F(X^*)$ as $k \to +\infty$.

(3) From (32) and (33), we have

$$\left(\frac{\tau}{2} - \frac{\| \mathcal{A} \|_2^2}{2} - \tau\delta\right) \| X_{k+1} - Y_k \|_F^2 \leq F(X_k) - F(X_{k+1}). \tag{34}$$

Summing (34) from $k = 1$ to m, we have

$$\sum_{k=1}^{m} \| X_{k+1} - Y_k \|_F^2 \leq \frac{1}{\frac{\tau}{2} - \frac{\|\mathcal{A}\|_2^2}{2} - \tau\delta}(F(X_1) - F(X_{m+1})). \tag{35}$$

Let $m \to \infty$, becomes

$$\sum_{k=1}^{\infty} \| X_{k+1} - Y_k \|_F^2 \leq \frac{1}{\frac{\tau}{2} - \frac{\|\mathcal{A}\|_2^2}{2} - \tau\delta}(F(X_1) - F(X^*)). \tag{36}$$

Thus,

$$\sum_{k=1}^{\infty} \| X_{k+1} - Y_k \|_F^2 < \infty, \tag{37}$$

Finally, we have that

$$\lim_{k\to\infty} \| X_{k+1} - Y_k \|_F^2 = 0, \tag{38}$$

This also implies that $\{X_k\}$ and $\{Y_k\}$ share the same set of limit points.

(4) Let $\{X_{k_j}\}$ and $\{Y_{k_j}\}$ be the convergent subsequences of $\{X_k\}$ and $\{Y_k\}$, respectively. Assuming X^* be their limit point, i.e.,

$$X_{k_j} \to X^*, \quad Y_{k_j} \to X^* \quad as \quad k_j \to +\infty. \tag{39}$$

From

$$\| X_{k_{j+1}} - X^* \|_F \leq \| X_{k_{j+1}} - X_{k_j} \|_F + \| X_{k_j} - X^* \|_F \to 0, \quad as \quad k_j \to +\infty,$$

we get

$$X_{k_{j+1}} \to X^* \quad as \quad k_j \to +\infty. \tag{40}$$

Since $X_{k_{j+1}} = prox_{\frac{\lambda}{\tau}T(\cdot)}(Y_{k_j} - \frac{1}{\tau}\nabla f(Y_{k_j}))$, we have

$$\frac{1}{2}\left\| X_{k_{j+1}} - \left[Y_{k_j} + \frac{1}{\tau}\mathcal{A}^*(b - \mathcal{A}(Y_{k_j}))\right]\right\|_F^2 + \frac{\lambda}{\tau}T(X_{k_{j+1}}) \leq \frac{1}{2}\left\| X - \left[Y_{k_j} + \frac{1}{\tau}\mathcal{A}^*(b - \mathcal{A}(Y_{k_j}))\right]\right\|_F^2 + \frac{\lambda}{\tau}T(X).$$

From (39) and (40), we immediately get for any $X \in \mathbb{R}^{m\times n}$ that

$$\frac{1}{2}\left\| X^* - \left[X^* + \frac{1}{\tau}\mathcal{A}^*(b - \mathcal{A}(X^*))\right]\right\|_F^2 + \frac{\lambda}{\tau}T(X^*) \leq \frac{1}{2}\left\| X - \left[X^* + \frac{1}{\tau}\mathcal{A}^*(b - \mathcal{A}(X^*))\right]\right\|_F^2 + \frac{\lambda}{\tau}T(X),$$

therefore, X^* is the minimizes of the following function:

$$X \to \frac{1}{2}\left\| X - \left[X^* + \frac{1}{\tau}\mathcal{A}^*(b - \mathcal{A}(X^*))\right]\right\|_F^2 + \frac{\lambda}{\tau}T(X).$$

We complete the proof. □

ATS1PA can be directly applied to solve matrix completion problems by replacing its step 4 with

$$B = Y - \frac{1}{\tau}\mathcal{P}_{\Omega}(W - Y), \tag{41}$$

where Ω denotes the indices of the observed entries, and $\mathcal{P}_{\Omega}$ is defined as

$$[\mathcal{P}_{\Omega}(M)]_{ij} = \begin{cases} M_{ij}, & \text{if } \Omega_{ij} = 1 \\ 0, & \text{if } \Omega_{ij} = 0. \end{cases}$$

The main computation cost of B is $\mathcal{P}_{\Omega}(W - Y)$ which takes $O(\| \Omega \|_1 \bar{r})$ time, where $\bar{r}$ is the rank of $(W - Y)$. The PowerMethod is performed in step 5 and takes $O(mnt_k)$ time, where t_k is the column number of R. At step 6, a SVD on a smaller matrix Q^TB is performed and SVD(Q^TB) takes only $O(mt_k^2)$ time. Besides, by using the "sparse plus low rank" structure of (41), the complexity of proximal step is $O((m+n)t_k^2 + \| \Omega \|_1 t_k)$ when $Y_{k+1} = V_{k+1}$ in step 13 or $O((m+n)\bar{r} + \| \Omega \|_1 t_k)$ when $Y_{k+1} = X_{k+1}$. Summarizing, the time complexity of ATS1PGA in each iteration is $O((m+n)t_k^2 + \| \Omega \|_1 t_k)$, where $t_k \ll n, \| \Omega \|_1 \ll mn$.

5 Numerical experiments

In this section, numerous experimental results on real-world data are presented to demonstrate that our proposed algorithm has an improved convergence rate and improved restoration capability of low-rank matrix over that of the state-of-the-art algorithms. We compare ATS1PGA algorithm with the following representative algorithms:

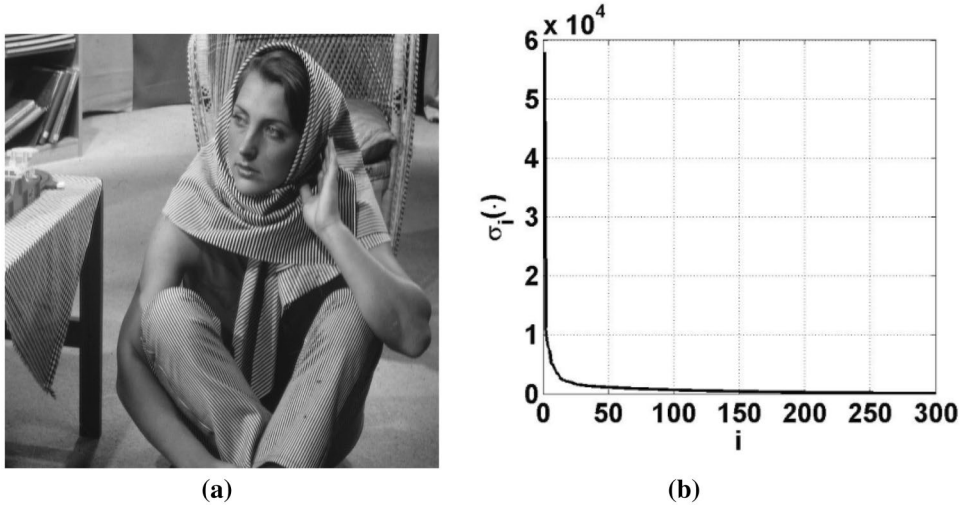

Fig. 1 Example of low-rank image. **a** One 512×512 image. **b** First 300 singular values of Barbara

Fig. 2 The 20 test grayscale images for image recovery

- SVT [13]: a widely used nuclear-norm-based algorithm which is inspired by the linearized Bregman iterations for compressed sensing.
- APG [14]: a nuclear-norm-based algorithm which extends a fast iterative shrinkage thresholding algorithm from the vector case to the matrix case.
- FPCA [41]: an algorithm deals with the same problem as APG while employing fast Monte Carlo algorithm for approximate SVD.
- R1MP [36]: an algorithm which extends the orthogonal matching pursuit algorithm from the vector case to the matrix case.
- IRNN [24]: a nonconvex algorithm which replaces the nuclear norm by reweighted nuclear norm. We choose SCAD in this work, as it always generates the best result in our test.
- FaNCL [25]: a nonconvex algorithm deals with the same problem as IRNN while achieving improved convergence rate. We select SCAD in this work.
- TS1 [26]: a nonconvex algorithm which is proposed by using transformed Schatten-1 penalty function.

As shown in Fig. 1, an image usually represents low-rank or approximately low-rank structure. Thus, the problem of recovering an incomplete image can be treated as the problem of recovering a low-rank matrix. To test effectiveness of our algorithm on real data, we compare it with the state-of-the-art algorithms on 20 widely used images represented in Fig. 2. The size of the first 7 images is 256×256, the size of the following 10 images is 512×512, and the size of the last 3 images is 1024×1024. In our tests, two different types of mask are considered.

- Random mask: given an image, we randomly exclude $\alpha\%$ pixels, and the remaining ones serve as the observations.
- Text mask: the text may cover some important texture information of a given image.

In the following experiments, the parameters in each algorithm are obtained by the recommended setting. For our proposed algorithm, we use the same parameter values as the TS1 algorithm [26]. Besides, all the algorithms are stopped when the difference in objective values between consecutive iterations becomes smaller than 10^{-4}. All the algorithms are implemented in MATLAB R2014a on a Windows server 2008 system with Intel Xeon E5-2680-v4 CPU(3 cores, 2.4GHz) and 256GB memory.

Table 1 Comparison of different algorithms on image datasets with $sr = 0.5$ and $s = 10$. CPU time is in seconds

	SVT		APG		FPCA		R1MP		IRNN		FaNCL		TS1		ATS1PGA	
	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
Image 1	28.12	7.0522	26.63	1.7294	26.44	3.9195	27.24	0.5628	26.67	0.6760	26.64	0.2991	27.85	8.0564	**29.15**	**0.1506**
Image 2	24.79	7.1050	24.23	1.7746	23.41	3.0221	24.51	0.5796	23.44	0.7260	23.39	0.3132	**25.46**	8.4647	25.23	**0.2271**
Image 3	25.66	7.8534	24.85	1.6502	24.41	3.1179	25.40	0.6396	24.35	0.5701	24.28	0.2166	26.35	8.4339	**26.37**	**0.1839**
Image 4	25.01	6.1852	24.58	1.7187	23.92	3.0520	24.89	0.6420	23.72	0.5366	23.71	0.2284	**25.84**	8.4713	25.38	**0.2003**
Image 5	22.94	10.7512	22.04	1.7674	20.91	3.0079	21.92	0.5750	21.46	0.8059	21.44	0.3129	**23.26**	8.4254	22.34	**0.1220**
Image 6	23.20	10.2381	22.39	1.7433	21.55	2.8939	22.33	0.5734	21.94	0.9809	21.99	0.2888	**23.75**	8.5683	22.25	**0.2178**
Image 7	24.18	10.6843	23.07	1.6810	22.69	2.8700	23.04	0.6220	22.95	0.7295	22.77	0.2938	**24.84**	8.4372	24.19	**0.2352**
Image 8	22.58	42.3117	22.98	7.3266	20.68	16.4599	21.69	2.9292	22.63	9.5122	22.64	4.6246	**22.91**	44.5014	22.68	**1.3937**
Image 9	24.57	27.9177	24.93	7.3271	23.28	16.1353	24.16	2.7988	24.63	6.9092	24.62	1.6469	**25.24**	45.1350	25.22	**0.6330**
Image 10	22.66	38.2290	23.04	7.8385	21.23	16.9078	22.07	2.7685	22.73	6.7341	22.75	3.1411	**23.36**	44.6797	22.73	**0.8809**
Image 11	25.77	26.1820	25.89	7.3974	24.48	15.9024	25.23	2.6248	25.92	4.2235	25.78	1.57111	26.20	43.5562	**26.66**	**0.5132**
Image 12	24.92	29.4253	25.23	7.4794	23.71	16.3034	24.67	2.7540	24.92	3.9365	24.93	1.8087	25.52	44.8915	**25.66**	**0.8790**
Image 13	24.23	38.6971	24.42	7.7039	22.39	15.8564	23.24	2.6455	24.35	4.4073	24.27	2.1387	**24.65**	44.5569	24.55	**0.6032**
Image 14	26.27	30.1202	26.45	7.3598	25.04	15.7296	26.00	2.4848	26.24	3.7275	26.37	1.2179	26.71	43.8446	**27.41**	**0.5545**
Image 15	21.80	51.1362	**21.91**	8.3499	19.47	16.1681	20.37	2.7977	21.82	27.2993	21.84	6.8433	21.89	44.1598	21.27	**0.7109**
Image 16	25.57	30.7280	25.86	7.4880	24.46	16.4550	25.19	2.7807	25.62	3.8177	25.53	1.6164	26.14	43.6423	**26.35**	**0.4293**
Image 17	28.01	25.6420	28.38	7.1608	27.57	15.9058	28.40	2.5718	28.10	3.7581	28.05	0.9712	28.33	45.7123	**29.81**	**0.9383**
Image 18	29.03	60.4977	30.34	30.3649	29.01	63.6952	29.84	14.3825	30.54	15.6403	30.52	5.3984	28.86	174.6819	**30.55**	**3.550**
Image 19	23.47	80.0213	**24.85**	32.7804	21.93	63.8732	23.12	14.2329	23.47	50.4676	23.49	33.1106	24.21	173.8045	23.48	**2.4335**
Image 20	24.34	82.4252	**25.67**	36.5820	22.26	66.5408	23.61	13.2689	24.44	45.1302	24.44	20.4999	24.92	186.6649	24.40	**1.9598**

The best results are hignlighted in bold

Table 2 Comparison of different algorithms on image datasets with $sr = 0.5$ and $s = 15$. CPU time is in seconds

	SVT		APG		FPCA		R1MP		IRNN		FaNCL		TS1		ATS1PGA	
	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
Image 1	**27.26**	7.0754	26.33	1.7875	24.45	3.1190	25.15	0.6559	26.37	0.7900	26.37	0.2491	25.40	8.1904	27.24	**0.1670**
Image 2	24.11	7.8826	24.04	1.7386	22.34	3.0952	22.37	0.6941	23.41	0.6996	23.43	0.3183	23.81	8.7700	**24.30**	**0.2396**
Image 3	24.98	8.4289	24.69	1.6836	23.08	2.9465	24.03	0.6239	24.33	0.6250	24.14	0.2237	24.49	8.8721	**25.31**	**0.1837**
Image 4	24.44	6.5536	24.49	1.6371	22.61	3.0976	23.52	0.6854	23.69	0.9070	23.66	0.2748	24.15	8.5616	**24.59**	**0.2079**
Image 5	**22.19**	12.5943	21.75	1.7638	20.18	2.9841	21.21	0.6708	21.25	1.7498	21.42	0.3314	22.12	8.4163	21.99	**0.1368**
Image 6	**22.59**	12.5106	22.22	1.7885	20.68	3.0374	21.62	0.5905	21.91	0.9223	21.85	0.3332	22.47	8.6626	22.02	**0.2567**
Image 7	23.49	11.2374	22.82	1.7581	21.57	3.0503	22.38	0.5532	22.62	0.9342	22.67	0.3128	23.29	8.6556	**23.53**	**0.2472**
Image 8	22.04	47.5567	**22.44**	7.8679	20.26	18.5199	20.20	2.8686	22.24	13.9046	22.21	6.6865	22.13	43.9176	22.08	**3.7260**
Image 9	24.02	30.0802	24.36	7.5350	22.50	16.8306	23.41	2.7679	24.37	5.6361	**24.38**	2.9922	23.66	44.7387	24.35	**0.6778**
Image 10	22.16	40.6173	**22.58**	8.0979	20.63	17.4876	21.62	3.3935	22.45	13.4186	22.39	8.8178	22.39	44.4329	22.24	**1.1338**
Image 11	25.17	32.0916	25.14	7.6827	23.47	16.2158	24.41	2.6611	**25.52**	4.6738	25.51	2.1584	24.39	45.4974	**25.52**	**0.6029**
Image 12	24.34	33.4320	24.57	7.9366	22.73	16.1386	23.83	3.0092	24.65	8.3647	**24.70**	2.3716	23.86	44.8954	**24.70**	**1.0065**
Image 13	23.66	42.9858	23.66	7.7556	21.52	16.2950	22.65	2.6455	**24.02**	7.7096	23.99	2.8660	23.22	44.8724	23.81	**0.9447**
Image 14	25.56	32.0559	25.57	7.5512	23.87	15.7938	24.90	2.7802	25.98	3.6653	25.87	2.2813	24.64	44.3576	**26.15**	**1.6192**
Image 15	21.20	56.7737	**21.38**	8.0317	19.06	16.4430	20.08	2.7662	21.26	23.8753	21.24	11.0893	21.11	45.8977	20.93	**1.3831**
Image 16	24.92	33.7499	25.11	7.4171	23.42	16.4019	24.29	2.5981	25.32	4.6269	**25.34**	1.9289	24.34	43.6238	25.32	**0.6053**
Image 17	27.54	26.1472	27.34	7.6436	25.58	17.0050	26.47	2.9997	27.79	3.1753	**27.85**	1.1448	25.63	44.3064	27.69	**1.0031**
Image 18	28.89	61.2627	28.53	31.4887	27.49	64.3037	28.57	12.7627	**29.29**	106.1543	29.28	27.4176	26.19	173.6727	29.02	**4.4138**
Image 19	23.20	89.6432	**23.91**	33.8283	21.53	63.2012	22.75	14.2583	23.03	105.9206	23.03	27.0927	23.07	173.4691	23.14	**3.5614**
Image 20	23.99	91.5974	**24.49**	37.0507	21.85	70.5371	23.19	13.7292	23.94	68.3029	23.94	28.4139	23.51	188.8545	24.00	**2.8048**

The best results are hignlighted in bold

Table 3 Comparison of different algorithms on image datasets with $sr = 0.5$ and $s = 20$. CPU time is in seconds

	SVT		APG		FPCA		R1MP		IRNN		FaNCL		TS1		ATS1PGA	
	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
Image 1	25.06	8.5915	25.57	1.8343	22.56	3.0590	23.40	0.6234	25.91	0.9292	25.98	0.2948	23.44	8.6004	**26.10**	**0.1668**
Image 2	22.66	10.8310	**23.54**	1.7522	21.11	3.0005	22.21	0.6961	23.27	0.7902	23.18	0.3642	22.31	8.8061	23.33	**0.3183**
Image 3	23.28	10.1649	24.12	1.6842	21.47	3.0674	22.64	0.6556	24.01	0.9961	24.08	0.2946	22.71	8.5985	**24.51**	**0.1970**
Image 4	22.92	9.8998	**24.17**	1.7188	21.41	3.4124	22.30	0.7878	23.63	0.9238	23.64	0.2142	22.56	8.5065	23.89	**0.2101**
Image 5	21.00	14.3899	**21.47**	1.7805	19.08	3.1253	20.58	0.5748	21.19	1.8967	21.24	0.6189	21.03	8.8110	21.31	**0.2042**
Image 6	21.33	13.9101	21.66	1.8880	19.67	3.0426	20.91	0.6523	21.73	1.4291	21.72	0.3834	21.26	8.6544	**21.81**	**0.2279**
Image 7	22.00	13.1601	22.24	1.8118	20.23	3.0141	21.33	0.6857	22.40	1.0575	22.39	0.3879	21.83	8.5125	**22.73**	**0.2746**
Image 8	21.13	51.9788	**21.64**	8.1407	19.77	16.5176	20.63	2.9439	21.32	29.6646	21.32	15.5575	21.03	46.0052	21.37	**4.3629**
Image 9	22.79	37.5269	23.31	7.6974	21.69	16.3428	22.48	3.0272	23.56	12.2459	**23.62**	4.3179	22.23	44.2934	23.57	**1.5234**
Image 10	21.14	50.2153	**21.87**	7.8387	20.01	17.0072	20.99	3.0246	21.56	38.5842	21.49	15.2198	21.25	45.0953	21.63	**2.7375**
Image 11	23.64	36.0988	23.97	7.6217	22.16	16.7545	23.40	2.7268	24.61	10.8370	**24.64**	3.4134	22.64	44.2307	24.31	**1.5466**
Image 12	23.02	40.6698	23.59	7.9074	21.80	16.2900	22.80	2.8834	**23.92**	8.3893	23.88	5.7960	22.33	45.1900	23.67	**1.2327**
Image 13	22.32	49.5075	22.65	8.2963	20.89	16.1462	21.94	3.0360	**23.07**	15.4198	23.06	8.0790	21.79	46.3659	22.93	**1.0082**
Image 14	24.05	38.1440	24.35	7.5623	22.72	16.8214	23.72	2.8140	25.09	13.6789	**25.11**	2.8991	22.87	44.8490	24.75	**1.6319**
Image 15	20.24	60.2615	**20.69**	8.0638	18.60	16.9443	19.69	2.7648	20.45	36.2165	20.46	14.1827	20.25	44.8731	20.48	**1.8071**
Image 16	23.59	40.2224	23.97	7.6546	22.30	16.9502	23.18	2.9089	24.49	7.4147	**24.56**	3.3182	22.63	45.2543	24.20	**0.6872**
Image 17	25.85	30.7902	25.81	7.7276	24.14	16.3700	24.64	3.0404	26.92	7.3013	**26.93**	2.3102	23.60	45.023	**26.93**	**1.2476**
Image 18	**28.14**	67.8806	26.32	31.2338	26.24	65.6523	27.42	11.3425	26.92	344.5795	26.94	57.9225	24.11	175.9683	27.64	**10.5524**
Image 19	22.44	113.8239	**22.78**	33.8090	21.14	63.2387	22.20	14.6544	22.37	114.4254	22.39	57.2202	21.84	173.1046	22.72	**3.7059**
Image 20	23.18	109.0291	23.12	38.0988	21.31	68.6569	22.62	16.6918	23.25	116.5822	23.24	38.6683	22.14	186.8347	**23.44**	**3.0409**

The best results are hignlighted in bold

Table 4 Comparison of different algorithms on image datasets with $sr = 0.5$ and $s = 30$. CPU time is in seconds

	SVT		APG		FPCA		R1MP		IRNN		FaNCL		TS1		ATS1PGA	
	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
Image 1	20.11	14.3610	23.20	1.8522	20.00	3.0965	20.61	0.6953	24.49	1.5899	24.51	0.4640	20.38	8.3933	**24.52**	**0.1812**
Image 2	19.38	15.3012	21.90	1.7767	19.09	2.9461	19.89	0.8097	22.17	1.7014	22.14	0.7116	19.87	8.5216	**22.25**	**0.3357**
Image 3	19.69	15.8596	22.26	2.1045	19.12	3.0053	20.10	0.7347	22.80	1.5227	22.90	0.6428	20.01	8.6046	**23.06**	**0.2136**
Image 4	19.68	16.1042	22.45	1.8133	19.29	3.1799	19.88	0.7912	22.69	1.7021	22.70	0.5357	20.03	8.3516	**22.81**	**0.3969**
Image 5	18.49	17.6919	20.12	1.7781	17.57	2.9548	18.80	0.7135	20.12	1.7981	20.09	1.2375	18.82	8.6311	**20.16**	**0.2816**
Image 6	18.72	17.6077	20.43	1.8103	17.74	2.9793	19.06	0.6814	20.45	1.9326	20.42	0.8847	19.03	8.4985	**20.50**	**0.5845**
Image 7	18.91	18.1794	20.43	1.9956	18.21	3.0377	19.25	0.6706	**21.07**	2.4327	21.06	0.7646	19.32	8.7427	21.06	**0.6599**
Image 8	18.64	69.5181	19.93	7.9818	18.57	16.5543	19.33	3.1697	**20.26**	81.1044	18.86	31.9728	19.02	44.9981	19.77	**3.9213**
Image 9	19.47	61.2749	21.15	8.4589	19.95	16.2363	20.52	3.2452	20.09	63.8744	20.06	31.8371	19.73	43.9468	**21.43**	**0.9740**
Image 10	18.72	67.0737	20.10	7.7532	18.80	16.2424	19.46	3.1780	18.93	80.8163	18.89	31.1456	19.18	44.1960	**21.19**	**1.0960**
Image 11	19.76	60.5257	21.39	7.6255	20.23	16.4330	21.32	2.8015	20.66	74.9666	20.72	31.7775	19.96	44.2506	**22.04**	**1.0628**
Image 12	19.62	63.0445	21.18	8.1335	20.06	17.1576	20.69	3.1226	20.29	46.1151	20.29	31.3270	19.83	45.7640	**21.70**	**3.1464**
Image 13	19.13	70.7935	20.47	7.7552	19.16	16.2298	20.25	2.8607	19.93	73.0650	19.91	16.6163	19.38	43.6602	**21.13**	**1.3068**
Image 14	19.94	63.2063	21.58	7.7174	20.61	15.9440	21.59	2.8529	20.97	80.2288	20.92	24.2662	20.09	44.3834	**22.34**	**1.2416**
Image 15	18.08	80.0374	19.07	8.0368	17.49	17.1265	18.59	2.9195	18.32	52.9593	18.16	29.6874	18.39	44.6563	**19.39**	**2.7629**
Image 16	19.73	64.4273	21.41	7.8360	20.28	16.2231	20.88	3.0687	20.70	36.8793	20.61	20.4402	19.96	45.4466	**21.92**	**0.9543**
Image 17	20.54	57.8760	22.49	8.5481	21.59	16.2335	21.91	3.2788	21.79	34.4166	21.74	24.4827	20.52	44.6804	**23.16**	**1.5515**
Image 18	21.29	174.4495	22.44	32.8983	23.91	63.1408	24.56	**10.8614**	23.03	358.2839	23.03	121.5080	20.96	177.2554	**25.18**	11.0252
Image 19	19.57	240.4020	20.46	34.2310	20.24	64.6557	20.91	15.8955	20.64	270.3954	20.58	121.9631	19.63	175.1030	**21.64**	**14.5619**
Image 20	19.79	237.7699	20.52	37.8127	20.33	68.4339	21.35	13.1204	21.37	297.6245	21.41	133.4120	19.71	185.3365	**22.13**	**3.8326**

The best results are hignlighted in bold

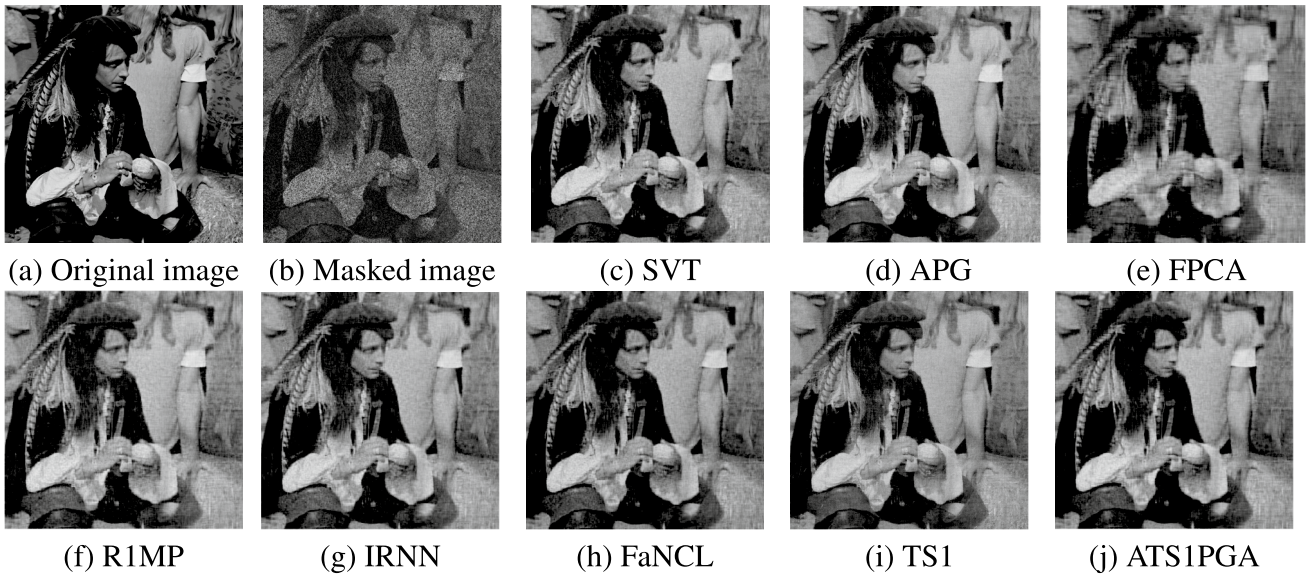

Fig. 3 Recovered images by using different algorithms on image 20 with $sr = 0.5$ and $s = 20$

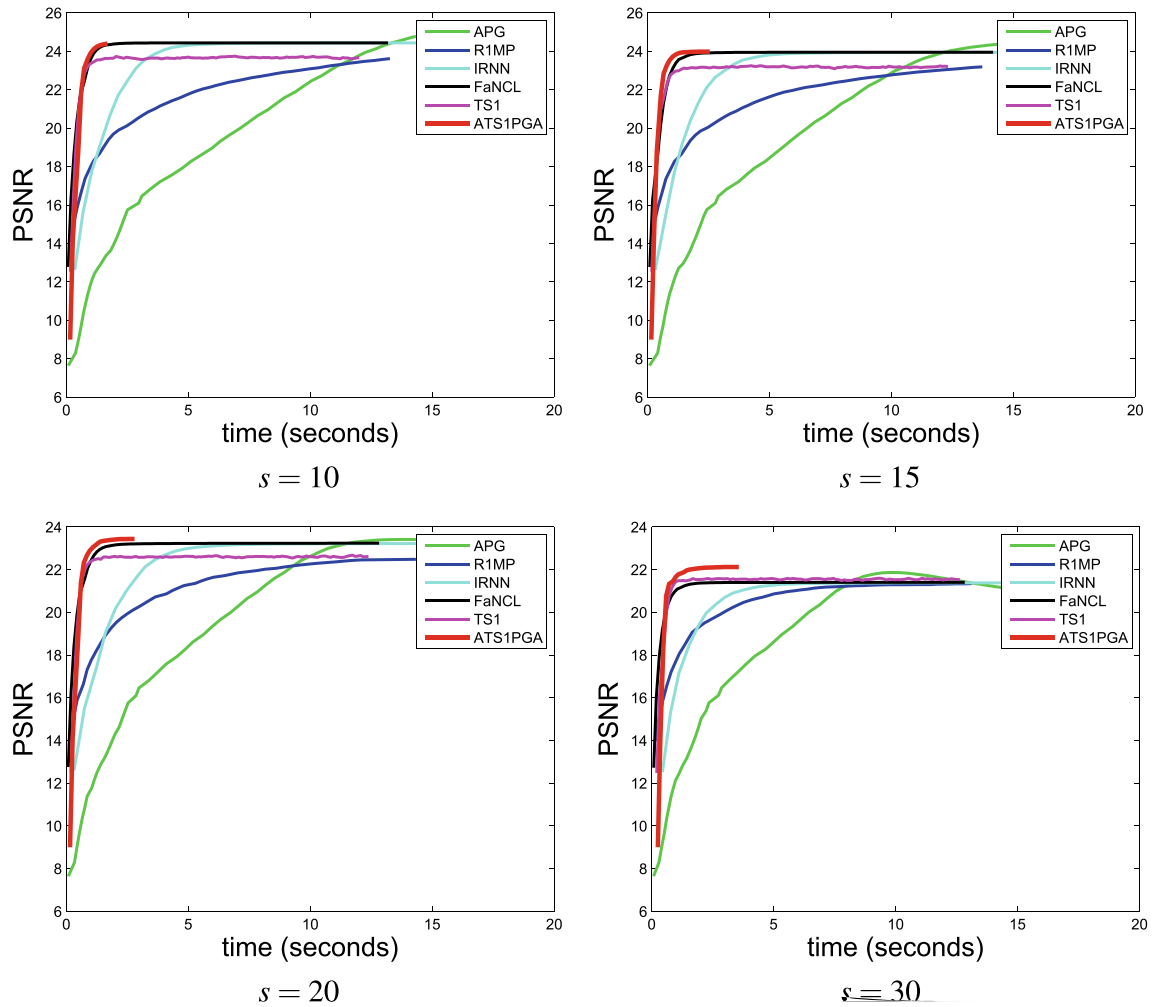

Fig. 4 Low-rank matrix recovery results on image data. We depict the PSNR along the running time

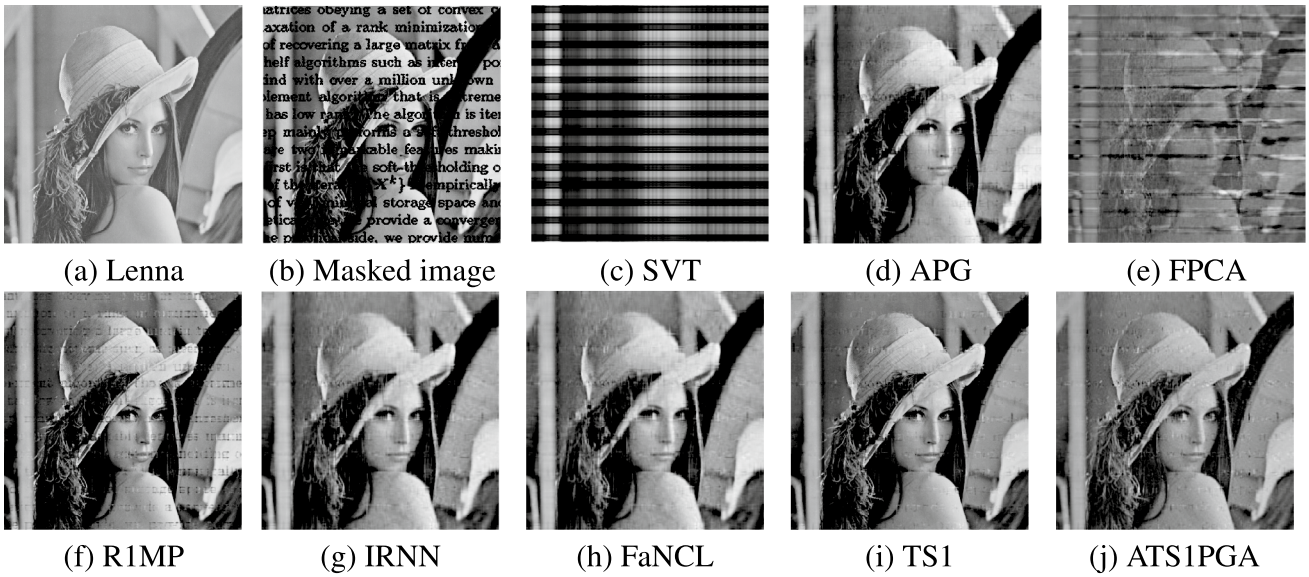

Fig. 5 Recovery results of the eight methods on Lenna with text noise. CPU time is in seconds. **a** Lenna. **b** Lenna with text. **c** SVT, Time = 14.08. **d** APG, Time = 9.37. **e** FPCA, Time = 18.36. **f** R1MP, Time = 6.96. **g** IRNN, Time = 9.38. **h** FaNCL, Time = 4.42. **i** TS1, Time = 55.98. **j** ATS1PGA, Time = 2.79. These results show that our proposed algorithm outperforms the competing methods

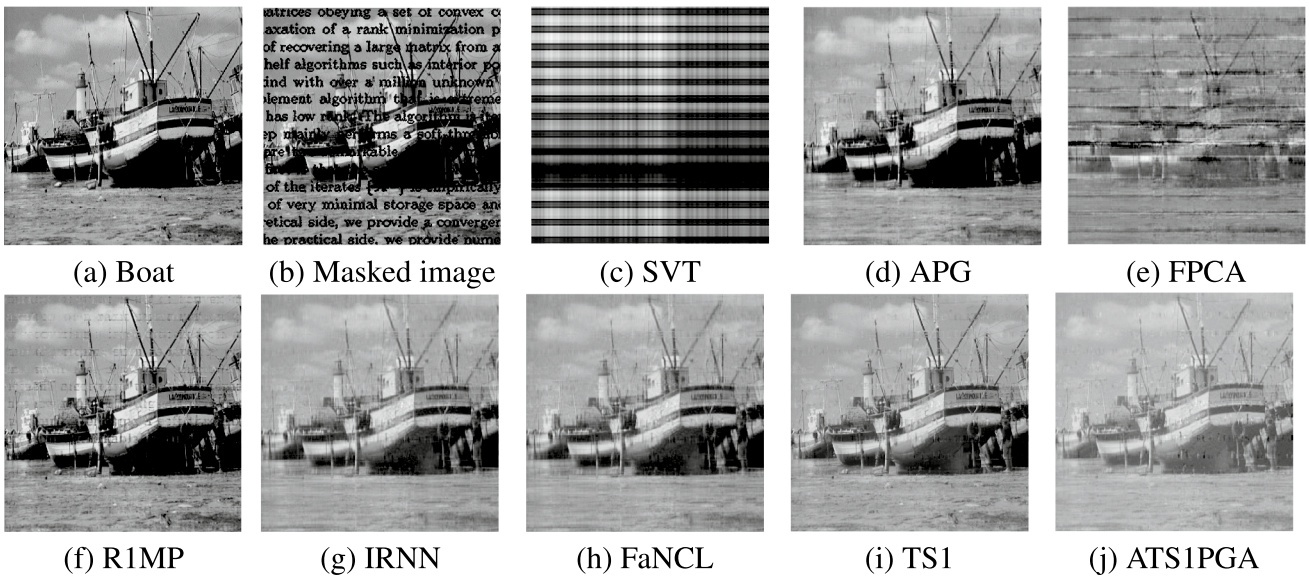

Fig. 6 Recovery results of the eight methods on Boat with text noise. CPU time is in seconds. **a** Boat. **b** Boat with text. **c** SVT, Time = 14.78. **d** APG, Time = 8.51. **e** FPCA, Time = 18.34. **f** R1MP, Time = 8.93. **g** IRNN, Time = 10.76. **h** FaNCL, Time = 5.40. **i** TS1, Time = 54.54. **j** ATS1PGA, Time = 3.15. These results show that our proposed algorithm outperforms the competing methods

5.1 Image inpainting with random mask

Image inpainting is one of the most basic problems in the field of image processing, which aims to find out the missing pixels from very limited information of an incomplete image. In the following tests, we first consider a relatively easy low-rank matrix recovery problem. We assume that the incomplete image data is corrupted with noise. Specifically, let matrix X represents an incomplete image data, before sampling missing pixels we first generate a noise matrix N with i.i.d. elements drawn form Gaussian distribution $\mathcal{N}(0, s)$. Then, we set $X = X + N$ as the observed matrix. By using the same setup as in [16], we randomly exclude 50% of

the pixels in each image, and the remaining ones are used as the observations. We also use sr to denote the sample ratio.

The performance of all algorithms are evaluated as: (1) peak signal-to-noise ration (PSNR) [42]; (2) the running time. We vary s in the range $\{10, 15, 20, 30\}$. The results with different levels of noise, the average of 10 times experiments, are reported.

It is seen from Table 1 that ATS1PGA algorithm achieves higher PSNR values than other alternative algorithms except TS1. We also find that TS1 achieves most accurate solutions 9 times on all images. However, ATS1PGA runs much faster than TS1. Actually, our proposed algorithm is the fastest. From Tables 2 and 3, we can observe that the noise degenerates the performance of TS1. For ATS1PGA, we can obtain the similar trend in Table 1. From Table 4, we can find that our proposed algorithm outperforms all competing algorithms on nearly all images. We also present the recovered images in Fig. 3 by using different algorithms and show that the images obtained by ATS1PGA contain more details than those obtained by other stat-of-the-art algorithms. The convergence efficiency of ATS1PGA is also investigated and the empirical results are presented in Fig. 4. The results in Fig. 4 show that our ATS1PGA algorithm performs best among all competing algorithms. Therefore, taking both accuracy and efficiency into consideration, our ATS1PGA algorithm has the best recovery performance among all stat-of-the-art algorithms.

5.2 Image inpainting with text mask

In this section, we will consider the text removal problem, where some of the pixels of one image are masked in a non-random fashion, such as texts on the image. Text removal is a tough task in the field of image processing. To deal with such a problem, the position of the text should be detected first, and then the corresponding task turns into recovering a low-rank matrix problem. Figures 5 and 6 represent the empirical results of the eight low-rank matrix recovery methods. Specifically, for the example Lenna in Fig. 5, the PSNR values for SVT, APG, FPCA, R1MP, IRNN, FaNCL, TS1, and ATS1PGA are 5.53, 27.04, 13.54, 27.08, 26.63, 26.63, 29.79, and 27.76, respectively. And for the example Boat in Fig. 6, the PSNR values for SVT, APG, FPCA, R1MP, IRNN, FaNCL, TS1, and ATS1PGA are 5.4, 26.04, 14.27, 26.57, 25.65, 25.65, 28.55, and 26.22, respectively. These results show that our ATS1PA algorithm is better than SVT, APG, FPCA, R1MP, IRNN, and FaNCL for both Lenna and Boat but only slightly worse than TS1. In terms of speed among eight methods, our ATS1PGA is the fastest. In particular, our ATS1PGA is at least 15 times faster than TS1. Thus, we can conclude that our ATS1PGA algorithm is competitive in handling the text removal task.

6 Conclusion and future work

This paper further investigated the basic properties of TS1 penalty function and utilized it to deal with the problem of low-rank matrix recovery. Specifically, we theoretically proved that the original low-rank problem (1) can be equivalently transformed into the problem (5) under certain conditions. Since the resulting optimization problem (5) is still NP-hard, we proved that the solutions of (5) can be obtained by solving its regularization problem. To provide an efficient and fast low-rank matrix recovery method, an algorithm with inexact proximal steps and Nesterov's rule was proposed. Besides, a convergence analysis of the proposed algorithm demonstrated that any accumulation point of the sequence generated by our algorithm is a stationary point. Finally, numerical experiments on real-world data sets demonstrated that our proposed algorithm much faster than the SVT, APG, FPCA, R1MP, IRNN, FaNCL, and TS1 algorithms. The experiments results also showed that our proposed algorithm achieves comparable recovery performance. We can conclude that our algorithm leads to impressive improvements over the state-of-the-art methods in the field of low-rank matrix recovery.

In many real-world applications, it is not appropriate to assume the observed entries are corrupted by only white Gaussian noise. This means that there is still much follow-up work to be done in future, such as developing more flexible model with TS1 regularizer to deal with low-rank matrix recovery problem when observed entries are corrupted by non-Gaussian noise. Furthermore, developing fast, robust, scalable and reliable algorithm is also a central issue in the future.

Acknowledgements This work is supported in part by the Natural Science Foundation of China under Grant 11901476, and in part by the Fundamental Research Funds for the Central Universities under Grant SWU120036.

References

1. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 426–434
2. Luo X, Zhou M, Li S, You Z, Xia Y, Zhu Q (2016) A non-negative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method. IEEE Trans Neural Netw Learn Syst 27(3):579–592
3. Huang C, Ding X, Fang C, Wen D (2014) Robust image restoration via adaptive low-rank approximation and joint kernel regression. IEEE Trans Image Process 23(12):5284–5297
4. Chen B, Yang Z, Yang Z (2018) An algorithm for low-rank matrix factorization and its applications. Neurocomputing 275:1012–1020

5. Zhao F, Peng J, Cui A (2020) Design strategy of thresholding operator for low-rank matrix recovery problem. Signal Process 171:1–10
6. Luo X, Zhou M, Li S, Xia Y, You Z, Zhu Q, Leung H (2018) Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data. IEEE Trans Cybern 48(4):1216–1228
7. Fan J, Chow T (2017) Deep learning based matrix completion. Neurocomputing 266:791–803
8. Peng X, Zhang Y, Tang H (2016) A unified framework for representation-based subspace clustering of out-of-sample and large-scale data. IEEE Trans Neural Netw Learn Syst 27(12):2499–2512
9. Liu G, Liu Q, Yuan X (2017) A new theory for matrix completion. In: Proceedings of the advances in neural information processing systems, pp 785–794
10. Liu G, Liu Q, Li P (2016) Low-rank matrix completion in the presence of high coherence. IEEE Trans Signal Process 64(21):5623–5633
11. Fazel M (2002) Matrix rank minimization with applications. Ph.D. thesis, Stanford University
12. 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
13. Cai J-F, Candès EJ, Shen Z (2010) A singular value thresholding algorithm for matrix completion. SIAM J Optim 20(4):1956–1982
14. Toh K-C, Yun S (2010) An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. Pac. J Optim 6(3):615–640
15. Yao Q, Kwok J (2015) Accelerated inexact soft-impute for fast large scale matrix completion. In: Proceedings of the international joint conference on artificial intelligence, pp 4002–4008
16. Wang Z, Wang W, Wang J, Chen S (2019) Fast and efficient algorithm for matrix completion via closed-form 2/3-thresholding operator. Neurocomputing 330:212–222
17. Wang Z, Gao C, Luo X, Tang M, Wang J, Chen W (2020) Accelerated inexact matrix completion algorithm via closed-form q-thresholding ($q = 1/2, 2/3$) operator. Int J Mach Learn Cybern 11:2327–2339
18. Wang Z, Liu Y, Luo X, Wang J, Gao C, Peng D, Chen W (2021) Large-scale affine matrix rank minimization with a novel nonconvex regularizer. IEEE Trans Neural Netw Learn Syst (**to be published**)
19. Gu S, Zhang L, Zuo W, Feng X (2014) Weighted nuclear norm minimization with application to image denoising. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2862–2869
20. Fan J, Li R (2001) Variable selection via nonconcal penalized likelihood and its oracle properties. J Am Stat Assoc 96(456):1348–1360
21. Zhang C (2010) Nearly unbiased variable selection under minimax concave penalty. Ann Stat 38(2):894–942
22. Candès EJ, Wakin M, Boyd S (2008) Enhancing sparsity by reweighted l_1 minimization. J Fourier Anal Appl 14:877–905
23. Peng D, Xiu N, Yu J (2017) $S_{1/2}$ regularization methods and fixed point algorithms for affine rank minimization problems. Comput Optim Appl 67:543–569
24. Lu C, Tang J, Yan S, Lin Z (2016) Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm. IEEE Trans Image Process 25(1):829–839
25. Yao Q, Kwok J, Wang T, Liu T (2019) Large-scale low-rank matrix learning with nonconvex regularizers. IEEE Trans Pattern Anal Mach Intell 41(11):2628–2643
26. Zhang S, Yin P, Xin J (2017) Transformed Schatten-1 iterative thresholding algorithms for low rank matrix completion. Commun Math Sci 15(3):839–862
27. Lv J, Fan Y (2009) A unified approach to model selection and sparse recovery using regularized least squares. Ann Stat 37(6):3498–3528
28. Kang Z, Peng C, Cheng Q (2015) Robust PCA via nonconvex rank approximation. In: Proceedings of IEEE international conference on data mining, pp 211–220
29. Zhang S, Xin J (2017) Minimization of transformed L_1 penalty: closed form representation and iterative thresholding algorithms. Commun Math Sci 15(2):511–537
30. Zhang S, Xin J (2018) Minimization of transformed L_1 penalty: theory, difference of convex function algorithm, and robust application in compressed sensing. Math Progr 169(1–2):307–336
31. Cui A, Peng J, Li H (2018) Exact recovery low-rank matrix via transformed affine matrix rank minimization. Neurocomputing 319:1–12
32. Parikh N, Boyd S (2014) Proximal algorithms. Found Trends Optim 1(3):127–239
33. Schmidt M, Roux N, Bach F (2011) Convergence rates of inexact proximal gradient methods for convex optimization. In: Proceedings of the advances in neural information processing systems, pp 1458–1466
34. Gu B, Huo Z, Huang H (2018) Inexact proximal gradient methods for non-convex and non-smooth optimization. In: Proceedings of the twenty-second AAAI conference on artificial intelligence, pp 3093–3100
35. Li H, Lin Z (2015) Accelerated proximal gradient methods for nonconvex programming. In: Proceedings of the advances in neural information processing systems, pp 379–387
36. Wang Z, Lai M, Lu Z, Fan W, Davulcu H, Ye J (2015) Orthogonal rank-one matrix pursuit for low rank matrix completion. SIAM J Sci Comput 37(1):A488–A514
37. Halko N, Martinsson P, Tropp J (2011) Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev 53(2):217–288
38. Oh T, Matsushita Y, Tai Y, Kweon I (2018) Fast randomized singular value thresholding for low-rank optimization. IEEE Trans Pattern Anal Mach Intell 40(2):376–391
39. Nesterov Y (1983) A method for solving the convex programming problem with convergence rate $O(1/k^2)$. Dokl Akad Nauk SSSR 27(2):543–547
40. Li Q, Zhou Y, Liang Y, Varshney P (2017) Convergence analysis of proximal gradient with momentum for nonconvex optimization. In: Proceedings of the 34th international conference on machine learning, pp 2111–2119
41. Ma S, Goldfarb D, Chen L (2011) Fixed point and Bregman iterative methods for matrix rank minimization. Math Progr 128(1–2):321–353
42. Thu Q, Ghanbari M (2008) Scope of validity of PSNR in image/video quality assessment. Electron Lett 44(13):800–801