A Parameterized Proximal Point Algorithm for Separable Convex Optimization

Jianchao Bai · **Hongchao Zhang** ·
Jicheng Li

Abstract In this paper, we develop a Parameterized Proximal Point Algorithm (P-PPA) for solving a class of separable convex programming problems subject to linear and convex constraints. The proposed algorithm is provable to be globally convergent with a worst-case $O(1/t)$ convergence rate, where t denotes the iteration number. By properly choosing the algorithm parameters, numerical experiments on solving a sparse optimization problem arising from statistical learning show that our P-PPA could perform significantly better than other state-of-the-art methods, such as the Alternating Direction Method of Multipliers (ADMM) and the Relaxed Proximal Point Algorithm (R-PPA).

Jianchao Bai
School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, P.R. China
E-mail: bjc1987@163.com

Hongchao Zhang
Department of Mathematics, Louisiana State University, Baton Rouge, LA 70803-4918, USA
E-mail: hozhang@math.lsu.edu

Jicheng Li
School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, P.R. China
E-mail: jcli@mail.xjtu.edu.cn

1 Introduction

This article aims to study a novel parameterized proximal point method for solving the following two-block separable convex optimization problem

$$
\begin{aligned}
\min \ & f(x) + g(y) \\
\text{s.t.} \ & Ax + By = c, \\
& x \in \mathcal{X}, y \in \mathcal{Y},
\end{aligned}
\tag{1}
$$

where $f(x) : \mathcal{R}^m \to \mathcal{R}$ and $g(y) : \mathcal{R}^n \to \mathcal{R}$ are closed and proper convex functions, but not necessarily smooth; $A \in \mathcal{R}^{l \times m}, B \in \mathcal{R}^{l \times n}$ and $c \in \mathcal{R}^l$ are given matrices and vectors, respectively; $\mathcal{X} \subset \mathcal{R}^m$ and $\mathcal{Y} \subset \mathcal{R}^n$ are closed convex sets. Throughout the paper, we assume the solution set of the problem (1) is nonempty and the matrices A and B have full column rank.

It is well-known in the literature that proximal point methods are a class of benchmark methods for solving the problem (1). The Proximal Point Algorithm (PPA) was originally proposed for solving monotone operator inclusion problems [21,22] and then became popularized to convex programmings by Rockafellar [24] and Eckstein[10]. As demonstrated in [24], the augmented Lagrangian method [25] for solving the problem (1) is actually an application of PPA to its dual problem. And the recently very popular Alternating Direction Method of Multipliers (ADMM) can be also regarded as another special variant of PPA to the dual problem [9]. Due to its simplicity for implementation, efficiency and strong theoretical background, the PPA has attracted extensive researches in recent years for solving structured convex optimization problems, especially by the optimizers from the areas involving lots of structured data, such as compressed sensing, image processing and machine learning, etc.

There is very rich literature on PPA. Combettes and Pennanen [6] showed some conditions for the viability and weak convergence of an inexact Relaxed PPA (R-PPA) for finding a common zero of countably many cohypomonotone operators in Hilbert space. Later, based on the closed-form expressions for the proximity operators [7], Combettes et al.[2] still derived expressions of new proximity operators in product spaces and presented an extension of PPA for solving the multicomponent signal/image processing problems. Recently, PPA was extensively studied for a class of multi-criteria optimization problems with the difference of convex objective functions, whose efficiency was demonstrated by testing a multi-period portfolio minimization problem, see [19] for more details. More recently, by using proximal regularization techniques and partially parallel splitting schemes, Wang et al.[28] developed a proximal partially parallel splitting method for a multi-objective convex minimization problem. For an extensive review on PPA, one may refer [11,15] and the references therein. In what follows, we simply mention a few works closely related to the development of our proposed method. He et al.[15] investigated a customized application of the classical PPA for the convex programming with linear constraints, where some image processing problems were tested to show the efficiency of their method. Cai et al.[5] also proposed a R-PPA for solving (1) and analyzed its global convergence with a worst-case linear convergence

rate. Based on the results of [5, 11, 15], Ma and Ni [23] recently revisited the application of PPA for solving the basis pursuit and matrix completion problem. Our new proposed Parameterized PPA (**P-PPA**) can be actually regarded as more general extensions of the algorithms developed in [15] and [23] which does not make use of the separable structure of the objective function in (1).

Major contributions of this paper are summarized in the following. Firstly, the proximal matrix in our proposed P-PPA is more general and flexible than those in the previous work [15, 23], due to more induced parameters to take consideration of the problem structure instead of a unique objective function. Secondly, by properly choosing the algorithm parameters, the new P-PPA could significantly outperform some state-of-the-art methods, such as ADMM [1] and R-PPA [11], for solving the separable convex optimization, especially when the problem size is large and high accurate solutions are required.

The remaining parts are organized as follows. In Section 2, we characterize the solution of the problem (1) as the solution of proper variational inequalities and review the unified framework of PPA. In Section 3, we derive the new P-PPA and discuss its global convergence and worst-case convergence rate in an ergodic sense. At the end of Section 3, we still present a P-PPA with a relaxation step. Some preliminary numerical experiments are performed in Section 4 for comparing our proposed methods with two benchmark methods. We finally conclude the paper in Section 5.

2 Preliminaries

In this section, we first introduce some necessary notations used throughout the paper. Then, we characterize the solution of the problem (1) by the aid of an equivalent variational inequality. Similar approaches have been widely used in the literature, e.g. [15, 18].

For the sake of convenience, let $\mathcal{R}$, $\mathcal{R}^n$ and $\mathcal{R}^{n \times m}$ denote the set of real numbers, the set of n dimensional real column vectors and the set of $n \times m$ real matrices, respectively. For any $x, y \in \mathcal{R}^n$, $\langle x, y \rangle = x^T y$ denotes the standard inner product in $\mathcal{R}^n$ and $\|x\| = \sqrt{\langle x, x \rangle}$ is the Euclidean norm. Given any symmetric positive definite matrix $G \in \mathcal{R}^{n \times n}$, the weighted norm $\|x\|_G = \sqrt{\langle x, Gx \rangle}$. In addition, we use $\mathbf{I}$ and $\mathbf{0}$ to stand for the identity matrix and the zero vector with proper dimension, respectively.

The following basic lemma given in [18] will be used as a tool for analyzing the primal-dual solution pair of the problem (1).

Lemma 1 *Let $\varphi : \mathcal{R}^m \to \mathcal{R}$ and $\psi : \mathcal{R}^m \to \mathcal{R}$ be two convex functions defined on a closed convex set $\Omega \subset \mathcal{R}^m$ and ψ is differentiable. Suppose the solution set $\Omega^* = \arg \min_{x \in \Omega} \{\varphi(x) + \psi(x)\}$ is nonempty. Then we have*

$$x^* \in \Omega^* \quad \text{if and only if} \quad x^* \in \Omega, \ \varphi(x) - \varphi(x^*) + \langle x - x^*, \nabla \psi(x^*) \rangle \geq 0, \forall x \in \Omega.$$

Now, given any $\tau \in \mathcal{R}\backslash\{0\}$, that is $\tau \neq 0$, a Lagrangian function of problem (1) can be written as

$$L(x, y, \lambda) = f(x) + g(y) - \langle \lambda, \tau(Ax + By - c)\rangle, \qquad (2)$$

where $\lambda \in \mathcal{R}^l$ is the Lagrange multiplier. Then, for any primal-dual solution pair (x^*, y^*, λ^*) of (1), we have

$$L(x^*, y^*, \lambda) \leq L(x^*, y^*, \lambda^*) \leq L(x, y, \lambda^*),$$

which is equivalent to

$$\begin{cases} x^* = \arg\min\{f(x) - \langle \lambda, \tau Ax\rangle | x \in \mathcal{X}\}, \\ y^* = \arg\min\{g(y) - \langle \lambda, \tau By\rangle | \ y \in \mathcal{Y}\}, \\ \lambda^* = \arg\max\{- \langle \lambda, \tau(Ax + By - c)\rangle | \ \lambda \in \mathcal{R}^l\}. \end{cases}$$

By applying Lemma 1, the optimality conditions of the above equations are

$$\begin{cases} x^* \in \mathcal{X}, \ \ f(x) - f(x^*) + \langle x - x^*, -\tau A^T\lambda^*\rangle \geq 0, \ x \in \mathcal{X}, \\ y^* \in \mathcal{Y}, \ \ g(y) - g(y^*) + \langle y - y^*, -\tau B^T\lambda^*\rangle \geq 0, \ \ y \in \mathcal{Y}, \\ \lambda^* \in \mathcal{R}^l, \langle \lambda - \lambda^*, \tau(Ax^* + By^* - c)\rangle \geq 0, \qquad \lambda \in \mathcal{R}^l, \end{cases} \qquad (3)$$

which can be rewritten as a variational inequality (**VI**)

$$\text{VI}(\phi, \mathcal{J}, \mathcal{M}): \quad \phi(u) - \phi(u^*) + \langle w - w^*, \mathcal{J}(w^*)\rangle \geq 0, \quad \forall w \in \mathcal{M}, \quad (4)$$

where

$$\phi(u) = f(x) + g(y), \quad \mathcal{M} = \mathcal{X} \times \mathcal{Y} \times \mathcal{R}^l,$$

$$u = \begin{pmatrix} x \\ y \end{pmatrix}, \ w = \begin{pmatrix} x \\ y \\ \lambda \end{pmatrix} \text{ and } \mathcal{J}(w) = \tau \begin{pmatrix} -A^T\lambda \\ -B^T\lambda \\ Ax + By - c \end{pmatrix}.$$

Clearly, the solution set of $\text{VI}(\phi, \mathcal{J}, \mathcal{M})$, denoted by $\mathcal{M}^*$, is nonempty by the assumption of nonempty solution set of the problem (1). Since the affine mapping $\mathcal{J}$ is skew-symmetric, we can obtain

$$\langle w - \widehat{w}, \mathcal{J}(w)\rangle = \langle w - \widehat{w}, \mathcal{J}(\widehat{w})\rangle, \quad \forall \ w, \widehat{w} \in \mathcal{M}. \qquad (5)$$

Hence, the variational inequality (4) is also rewritten as

$$\phi(u) - \phi(u^*) + \langle w - w^*, \mathcal{J}(w)\rangle \geq 0, \quad \forall w \in \mathcal{M}. \qquad (6)$$

When the proximal point algorithms are applied to solve the variational inequality (6) or equivalently (4), they often take the following unified approach: at the k-th iteration, find iterate w^{k+1} satisfying

$$\phi(u) - \phi(u^{k+1}) + \langle w - w^{k+1}, \mathcal{J}(w^{k+1}) + G\left(w^{k+1} - w^k\right)\rangle \geq 0, \ \ \forall w \in \mathcal{M}, \ (7)$$

where the above matrix $G \in \mathcal{R}^{(m+n+l)\times(m+n+l)}$ called proximal matrix is a positive definite, and

$$u^{k+1} = \begin{pmatrix} x^{k+1} \\ y^{k+1} \end{pmatrix}, \ w^{k+1} = \begin{pmatrix} x^{k+1} \\ y^{k+1} \\ \lambda^{k+1} \end{pmatrix}, \ \mathcal{J}(w^{k+1}) = \tau \begin{pmatrix} -A^T\lambda^{k+1} \\ -B^T\lambda^{k+1} \\ Ax^{k+1} + By^{k+1} - c \end{pmatrix}.$$

Obviously, different choices of G would result in different proximal point algorithms. We would provide a new choice of G for our proposed P-PPA.

3 Main results

In this section, we first develop the P-PPA for solving (1) in detail and discuss its convergence properties. Then, it is straightforward to extend the method to the case with a relaxation step.

3.1 Development of P-PPA with convergence

Mainly motivated by the proximal matrix of PPA in Eq.(2.5) of [15] and Eq.(3.1) of [23], we would design the matrix G in (7) having the following structure:

$$
G = \left[\begin{array}{cc|c} \left(\sigma + \frac{\varepsilon^2 - 1}{s}\right) A^T A & & -\varepsilon A^T \\ & \left(\rho + \frac{\tau^2 - 1}{s}\right) B^T B & -\tau B^T \\ \hline -\varepsilon A & -\tau B & s\mathbf{I} \end{array} \right],
\tag{8}
$$

where $(\sigma, \rho, s, \tau, \varepsilon)$ are parameters satisfying

$$
s > 0, \ \sigma > \frac{1}{s}, \ (\sigma s - 1)(\rho s - 1) - \tau^2 \varepsilon^2 > 0, \varepsilon \in \mathcal{R} \text{ and } \tau \in \mathcal{R} \backslash \{0\}.
\tag{9}
$$

For convenience, let us define

$$
\bar{\sigma} = \sigma + \frac{\tau^2 - 1}{s} \quad \text{and} \quad \bar{\rho} = \rho + \frac{\tau^2 - 1}{s}.
\tag{10}
$$

Then, from later analysis we can see that $1/s$ would play a role of penalty parameter for the equality constraint of (1), while $\bar{\sigma}$ and $\bar{\rho}$ can be regarded as the proximal parameters as those used in the customized PPA [15].

The following lemma ensures that under proper conditions of the parameters, G is a positive definite matrix.

Lemma 2 *Suppose that the matrices A and B have full column rank. For any $(\sigma, \rho, s, \tau, \varepsilon)$ satisfying (9), the matrix G defined in (8) is positive definite.*

Proof Clearly, the matrix G is symmetric and can be decomposed into

$$
G = D^T G_0 D,
$$

where

$$
D = \left[\begin{array}{ccc} A & & \\ & B & \\ & & \mathbf{I} \end{array} \right] \quad \text{and} \quad G_0 = \left[\begin{array}{cc|c} \left(\sigma + \frac{\varepsilon^2 - 1}{s}\right) \mathbf{I} & & -\varepsilon \mathbf{I} \\ & \left(\rho + \frac{\tau^2 - 1}{s}\right) \mathbf{I} & -\tau \mathbf{I} \\ \hline -\varepsilon \mathbf{I} & -\tau \mathbf{I} & s\mathbf{I} \end{array} \right].
\tag{11}
$$

By the full column rank assumption of A and B, the matrix G is positive definite if and only if G_0 is positive definite. Noting that

$$
\left[\begin{array}{ccc} \mathbf{I} & & \frac{\varepsilon}{s}\mathbf{I} \\ & \mathbf{I} & \frac{\tau}{s}\mathbf{I} \\ & & \mathbf{I} \end{array} \right] G_0 \left[\begin{array}{ccc} \mathbf{I} & & \frac{\varepsilon}{s}\mathbf{I} \\ & \mathbf{I} & \frac{\tau}{s}\mathbf{I} \\ & & \mathbf{I} \end{array} \right]^T = \left[\begin{array}{cc|c} \left(\sigma - \frac{1}{s}\right)\mathbf{I} & -\frac{\tau\varepsilon}{s}\mathbf{I} & \\ -\frac{\tau\varepsilon}{s}\mathbf{I} & \left(\rho - \frac{1}{s}\right)\mathbf{I} & \\ \hline & & s\mathbf{I} \end{array} \right] =: \widetilde{G}_0.
$$

Hence, G_0 is positive definite if and only if $\widetilde{G}_0$ is positive definite, which is guaranteed if condition (9) holds. Therefore, the proof is completed. $\diamondsuit$

In what follows, we develop our P-PPA in detail. Substituting the matrix G into (7), we have $\lambda^{k+1} \in \mathcal{R}^l$ and

$$\langle \lambda - \lambda^{k+1}, R_\lambda \rangle \geq 0, \ \forall \lambda \in \mathcal{R}^l,$$

where

$$R_\lambda = \tau \left(Ax^{k+1} + By^{k+1} - c \right) - \varepsilon A \left(x^{k+1} - x^k \right) - \tau B \left(y^{k+1} - y^k \right)$$
$$+ s \left(\lambda^{k+1} - \lambda^k \right).$$

Hence, we have $R_\lambda = 0$ which leads to

$$\lambda^{k+1} = \lambda^k - \frac{1}{s} \left[(\tau - \varepsilon)Ax^{k+1} + \varepsilon Ax^k + \tau By^k - \tau c \right]. \tag{12}$$

Meanwhile, by (7) and (12), we also have

$$x^{k+1} \in \mathcal{X}, \quad f(x) - f(x^{k+1}) + \left\langle x - x^{k+1}, R_x \right\rangle \geq 0, \quad \forall x \in \mathcal{X}, \tag{13}$$

where

$$R_x = -\tau A^T \lambda^{k+1} - \varepsilon A^T \left(\lambda^{k+1} - \lambda^k \right) + \left(\sigma + \frac{\varepsilon^2 - 1}{s} \right) A^T A \left(x^{k+1} - x^k \right)$$

$$= -(\tau + \varepsilon)A^T \left\{ \lambda^k - \frac{1}{s} \left[(\tau - \varepsilon)Ax^{k+1} + \varepsilon Ax^k + \tau By^k - \tau c \right] \right\}$$

$$+ \varepsilon A^T \lambda^k + \left(\sigma + \frac{\varepsilon^2 - 1}{s} \right) A^T A \left(x^{k+1} - x^k \right)$$

$$= -\tau A^T \bar{\lambda}^k + \bar{\sigma} A^T A \left(x^{k+1} - x^k \right)$$

with $\bar{\sigma}$ being defined in (10) and

$$\bar{\lambda}^k = \lambda^k - \frac{\tau + \varepsilon}{s} \left(Ax^k + By^k - c \right). \tag{14}$$

Notice that by (9), we get $\bar{\sigma} = (\sigma s + \tau^2 - 1)/s > 0$. Hence, by Lemma 1, x^{k+1} is the solution of the following optimization problem

$$x^{k+1} = \arg\min_{x \in \mathcal{X}} \left\{ f(x) + \frac{\bar{\sigma}}{2} \left\| A(x - x^k) \right\|^2 - \left\langle Ax, \tau \bar{\lambda}^k \right\rangle \right\}$$

$$= \arg\min_{x \in \mathcal{X}} \left\{ f(x) + \frac{\bar{\sigma}}{2} \left\| A(x - x^k) - \frac{\tau}{\bar{\sigma}} \bar{\lambda}^k \right\|^2 \right\}. \tag{15}$$

Similarly, we can also derive from (7) and (12) that

$$y^{k+1} \in \mathcal{Y}, \quad g(y) - g(y^{k+1}) + \left\langle y - y^{k+1}, R_y \right\rangle \geq 0, \quad \forall y \in \mathcal{Y}, \tag{16}$$

where

$$R_y = -\tau B^T \lambda^{k+1} - \tau B^T \left(\lambda^{k+1} - \lambda^k \right) + \left(\rho + \frac{\tau^2 - 1}{s} \right) B^T B \left(y^{k+1} - y^k \right)$$

$$= -2\tau B^T \left\{ \lambda^k - \frac{1}{s} \left[(\tau - \varepsilon) A x^{k+1} + \varepsilon A x^k + \tau B y^k - \tau c \right] \right\}$$

$$+ \tau B^T \lambda^k + \left(\rho + \frac{\tau^2 - 1}{s} \right) B^T B \left(y^{k+1} - y^k \right)$$

$$= -\tau B^T \bar{\lambda}^{k+\frac{1}{2}} + \bar{\rho} B^T B \left(y^{k+1} - y^k \right),$$

and

$$\bar{\lambda}^{k+\frac{1}{2}} = \lambda^k - \frac{2}{s} \left[(\tau - \varepsilon) A x^{k+1} + \varepsilon A x^k + \tau B y^k - \tau c \right]. \tag{17}$$

Furthermore, it follows from (14) and (17) that

$$\bar{\lambda}^{k+\frac{1}{2}} = \bar{\lambda}^k + \frac{\tau + \varepsilon}{s} \left(A x^k + B y^k - c \right)$$

$$- \frac{2}{s} \left[(\tau - \varepsilon) A x^{k+1} + \varepsilon A x^k + \tau B y^k - \tau c \right]$$

$$= \bar{\lambda}^k - \frac{\tau - \varepsilon}{s} \left[A(2 x^{k+1} - x^k) + B y^k - c \right]. \tag{18}$$

Hence, by Lemma 1, y^{k+1} is the solution of the following optimization problem

$$y^{k+1} = \arg\min_{y \in \mathcal{Y}} \left\{ g(y) + \frac{\bar{\rho}}{2} \left\| B(y - y^k) \right\|^2 - \left\langle By, \tau \bar{\lambda}^{k+\frac{1}{2}} \right\rangle \right\}$$

$$= \arg\min_{y \in \mathcal{Y}} \left\{ g(y) + \frac{\bar{\rho}}{2} \left\| B(y - y^k) - \frac{\tau}{\bar{\rho}} \bar{\lambda}^{k+\frac{1}{2}} \right\|^2 \right\}. \tag{19}$$

In addition, it follows from (12) and (14) that

$$\bar{\lambda}^{k+1} = \lambda^{k+1} - \frac{\tau + \varepsilon}{s} \left(A x^{k+1} + B y^{k+1} - c \right)$$

$$= \lambda^k - \frac{1}{s} \left[(\tau - \varepsilon) A x^{k+1} + \varepsilon A x^k + \tau B y^k - \tau c \right]$$

$$- \frac{\tau + \varepsilon}{s} \left(A x^{k+1} + B y^{k+1} - c \right)$$

$$= \bar{\lambda}^k + \frac{\tau + \varepsilon}{s} \left(A x^k + B y^k - c \right) - \frac{\tau + \varepsilon}{s} \left(A x^{k+1} + B y^{k+1} - c \right)$$

$$- \frac{1}{s} \left[(\tau - \varepsilon) A x^{k+1} + \varepsilon A x^k + \tau B y^k - \tau c \right]$$

$$= \bar{\lambda}^k - \frac{\tau}{s} (A x^{k+1} + B y^{k+1} - c)$$

$$- \frac{1}{s} \left[\tau A(x^{k+1} - x^k) + \varepsilon B(y^{k+1} - y^k) \right]. \tag{20}$$

Summarizing all the above discussions, we propose P-PPA as the following algorithm:

Algorithm 1 (P-PPA for solving Problem (1))

---

1 Choose parameters $(\sigma, \rho, s, \tau, \varepsilon)$ satisfying (9);

2 Initialize $(x^0, y^0, \lambda^0) \in \mathcal{R}^m \times \mathcal{R}^n \times \mathcal{R}^l$ and $r^0 = Ax^0 + By^0 - c$;

3 $\bar{\lambda}^0 = \lambda^0 - \frac{\tau + \varepsilon}{s} r^0$;

4 **For** $k = 0, 1, \cdots,$ **do**

5      $x^{k+1} = \arg\min\limits_{x \in \mathcal{X}} \left\{ f(x) + \frac{\bar{\sigma}}{2} \left\| A(x - x^k) - \frac{\tau}{\bar{\sigma}} \bar{\lambda}^k \right\|^2 \right\}$;

6      $\bar{\lambda}^{k+\frac{1}{2}} = \bar{\lambda}^k - \frac{\tau - \varepsilon}{s} \left[A(2x^{k+1} - x^k) + By^k - c \right]$;

7      $y^{k+1} = \arg\min\limits_{y \in \mathcal{Y}} \left\{ g(y) + \frac{\bar{\rho}}{2} \left\| B(y - y^k) - \frac{\tau}{\bar{\rho}} \bar{\lambda}^{k+\frac{1}{2}} \right\|^2 \right\}$;

8      $r^{k+1} = Ax^{k+1} + By^{k+1} - c$;

9      $\bar{\lambda}^{k+1} = \bar{\lambda}^k - \frac{\tau}{s} r^{k+1} - \frac{1}{s} \left[\tau A(x^{k+1} - x^k) + \varepsilon B(y^{k+1} - y^k) \right]$.

For the above P-PPA, we have the following remarks.

Remark 1 By taking $\tau = \varepsilon = 1$, the matrix G in (8) would become

$$G = \begin{bmatrix} \sigma A^T A & & -A^T \\ & \rho B^T B & -B^T \\ -A & -B & s\mathbf{I} \end{bmatrix},$$

where the parameters (σ, ρ, s) satisfy

$$s > 0, \quad \sigma > \frac{1}{s}, \quad (\sigma s - 1)(\rho s - 1) > 1.$$

In such case, Algorithm 1 would be reduced to

$$\begin{cases} \text{Let } \bar{\lambda}^0 = \lambda^0 - \frac{2}{s} \left(Ax^0 + By^0 - c \right); \\ \text{For } k = 0, 1, \cdots, \text{do} \\ \quad x^{k+1} = \arg\min\limits_{x \in \mathcal{X}} \left\{ f(x) + \frac{\bar{\sigma}}{2} \left\| A(x - x^k) - \frac{\tau}{\bar{\sigma}} \bar{\lambda}^k \right\|^2 \right\}; \\ \quad y^{k+1} = \arg\min\limits_{y \in \mathcal{Y}} \left\{ g(y) + \frac{\bar{\rho}}{2} \left\| B(y - y^k) - \frac{\tau}{\bar{\rho}} \bar{\lambda}^k \right\|^2 \right\}; \\ \quad \bar{\lambda}^{k+1} = \bar{\lambda}^k - \frac{1}{s} \left[A(2x^{k+1} - x^k) + B(2y^{k+1} - y^k) - c \right]. \end{cases}$$

This algorithm can be considered as a direct extension of the customized PPA [15], which only considers problem with one block structure, to solve the two-block structured problem (1).

Remark 2 The freedom of choosing the parameters $(\sigma, \rho, s, \tau, \varepsilon)$ would allow P-PPA to have more flexibility to select the proximal parameters $\bar{\sigma}$ and $\bar{\rho}$. However, note that they are proportional to the parameters $(\sigma, \tau, 1/s)$ and $(\rho, \tau, 1/s)$, respectively. As commented in the final part of [17], if they are too large, then slow convergence will occur in terms of solving the subproblems, which can significantly affect the overall efficiency of the algorithm.

Next, we discuss the convergence properties of P-PPA in a more general proximal point setting given in (7).

Lemma 3 *The sequence $\{w^{k+1}\}$ generated by Algorithm 1 satisfies*

$$\|w^{k+1} - w^*\|_G^2 \le \|w^k - w^*\|_G^2 - \|w^{k+1} - w^k\|_G^2, \quad \forall w^* \in \mathcal{M}^*. \qquad (21)$$

Proof Setting $w = w^*$ in (7), we have

$$\phi(u^*) - \phi(u^{k+1}) + \left\langle w^* - w^{k+1}, \mathcal{J}(w^{k+1}) + G\left(w^{k+1} - w^k\right)\right\rangle \ge 0,$$

which, by (4)-(5), implies

$$\left\langle w^* - w^{k+1}, G\left(w^{k+1} - w^k\right)\right\rangle \ge 0.$$

By the above inequality, we obtain

$$\|w^k - w^*\|_G^2 = \|w^k - w^{k+1} + w^{k+1} - w^*\|_G^2$$
$$\ge \|w^k - w^{k+1}\|_G^2 + \|w^{k+1} - w^*\|_G^2,$$

which immediately gives the inequality (21). $\quad \diamond$

Based on the above lemma, we have the following global convergence theorem.

Theorem 1 *Suppose that the condition (9) holds and the sequence $\{w^{k+1}\}$ is generated by Algorithm 1. Then, there exists a $w^\infty \in \mathcal{M}^*$ such that*

$$\lim_{k\to\infty} w^k = w^\infty. \qquad (22)$$

Proof Since condition (9) holds, we have by Lemma 2 that G is positive definite. Then, it follows from (21) that $\{w^k\}$ is bounded and

$$\lim_{k\to\infty} \|w^k - w^{k+1}\| = 0. \qquad (23)$$

Let w^∞ be any accumulation point of $\{w^k\}$. By taking a subsequence of w^k in (7) if necessary, it follows from (23) that

$$\phi(u) - \phi(u^\infty) + \langle w - w^\infty, \mathcal{J}(w^\infty)\rangle \ge 0, \ \forall w \in \mathcal{M}.$$

Hence, $w^\infty \in \mathcal{M}^*$. So, by (21) again, we have

$$\|w^k - w^\infty\|_G \le \|w^j - w^\infty\|_G \quad \text{for all } k \ge j.$$

Then, it follows from w^∞ being an accumulation point that (22) holds. $\quad \diamond$

Now, we establish the worst-case $\mathcal{O}(1/t)$ ergodic convergence rate for solving the variational inequality (6). Let

$$\mathbf{w}_t := \frac{1}{t+1}\sum_{k=0}^t w^{k+1} \quad \text{and} \quad \mathbf{u}_t := \frac{1}{t+1}\sum_{k=0}^t u^{k+1}. \qquad (24)$$

Theorem 2 *Suppose that the condition (9) holds and the sequence $\{w^{k+1}\}$ is generated by Algorithm 1. Then, we have*

$$\phi(\boldsymbol{u}_t) - \phi(u) + \langle \boldsymbol{w}_t - w, \mathcal{J}(w) \rangle \leq \frac{1}{2(t+1)} \left\| w^0 - w \right\|_G^2, \ \forall \ w \in \mathcal{M}. \quad (25)$$

Proof By (7) and the property (5), it holds that

$$\phi(u) - \phi(u^{k+1}) + \langle w - w^{k+1}, \mathcal{J}(w) \rangle \geq \langle w^{k+1} - w, G(w^{k+1} - w^k) \rangle, \forall \ w \in \mathcal{M}. \quad (26)$$

Then, applying the identity

$$2\langle a - b, G(a - c) \rangle = \|a - c\|_G^2 + \|a - b\|_G^2 - \|c - b\|_G^2$$

with

$$a = w^{k+1}, \quad b = w, \quad c = w^k,$$

we can obtain

$$\begin{aligned}
\langle w^{k+1} - w, G(w^{k+1} - w^k) \rangle &= \tfrac{1}{2} \left(\left\| w^{k+1} - w^k \right\|_G^2 + \left\| w^{k+1} - w \right\|_G^2 - \left\| w^k - w \right\|_G^2 \right) \\
&\geq \tfrac{1}{2} \left(\left\| w^{k+1} - w \right\|_G^2 - \left\| w^k - w \right\|_G^2 \right),
\end{aligned}$$

which together with (26) imply

$$\phi(u) - \phi(u^{k+1}) + \langle w - w^{k+1}, \mathcal{J}(w) \rangle + \frac{1}{2} \left\| w^k - w \right\|_G^2 \geq \frac{1}{2} \left\| w^{k+1} - w \right\|_G^2.$$

Summing the above inequality over $k = 0, 1, \cdots, t$, we get

$$(t+1)\phi(u) - \sum_{k=0}^{t} \phi(u^{k+1}) + \left\langle (t+1)w - \sum_{k=0}^{t} w^{k+1}, \mathcal{J}(w) \right\rangle + \frac{1}{2} \left\| w^0 - w \right\|_G^2 \geq 0,$$

which by the definition of $\mathbf{w}_t$ in (24) gives

$$\frac{1}{t+1} \sum_{k=0}^{t} \phi(u^{k+1}) - \phi(u) + \langle \mathbf{w}_t - w, \mathcal{J}(w) \rangle \leq \frac{1}{2(t+1)} \left\| w^0 - w \right\|_G^2. \quad (27)$$

By the convexity of the function $\phi(u)$ and the definition of $\mathbf{u}_t$ in (24), we have

$$\phi(\mathbf{u}_t) \leq \frac{1}{t+1} \sum_{k=0}^{t} \phi(u^{k+1}).$$

Then, (25) is true by substituting the above inequality into (27). $\diamond$

3.2 P-PPA with a relaxation step

Applying a relaxation step for PPA is a standard technique to accelerate its convergence [11,12,15]. Combining the PPA approach (7) together with a relaxation step would give the following procedure: at the k-th iteration, find $\widetilde{w}^{k+1}$ satisfying

$$\phi(u) - \phi(\widetilde{u}^{k+1}) + \left\langle w - \widetilde{w}^{k+1}, \mathcal{J}(\widetilde{w}^{k+1}) + G\left(\widetilde{w}^{k+1} - w^k\right)\right\rangle \geq 0, \ \ \forall w \in \mathcal{M}, \ (28)$$

and then let

$$w^{k+1} = w^k + \gamma(\widetilde{w}^{k+1} - w^k),$$

where $\gamma \in (0,2)$ is the relaxation parameter. When $\gamma = 1$, the above relaxed PPA will be reduced to the standard PPA. For the relaxed PPA, analogous to (21), it is not difficult (for details, see [11,12]) to show

$$\|w^{k+1} - w^*\|_G^2 \leq \|w^k - w^*\|_G^2 - \gamma(\gamma - 2)\|\widetilde{w}^{k+1} - w^k\|_G^2, \ \ \ \forall w^* \in \mathcal{M}^*. \ (29)$$

Then, based on (29), it is straightforward to have global convergence and convergence rate analogous to Theorem 1 and Theorem 2. More precisely, combining Algorithm 1 with a relaxation step would give the following Relaxed P-PPA (RP-PPA).

Algorithm 2 (RP-PPA for solving Problem (1))

---

1 Choose parameters $(\sigma, \rho, s, \tau, \varepsilon)$ satisfying (9) and $\gamma \in (0,2)$;
2 Initialize $(x^0, y^0, \lambda^0) \in \mathcal{R}^m \times \mathcal{R}^n \times \mathcal{R}^l$ and $r^0 = Ax^0 + By^0 - c$;
3 $\bar{\lambda}^0 = \lambda^0 - \frac{\tau + \varepsilon}{s} r^0$;
4 **For** $k = 0, 1, \cdots,$ **do**
5    $\widetilde{x}^k = \arg\min\limits_{x \in \mathcal{X}} \left\{ f(x) + \frac{\bar{\sigma}}{2} \left\| A(x - x^k) - \frac{\tau}{\sigma}\bar{\lambda}^k \right\|^2 \right\}$;
6    $\bar{\lambda}^{k+\frac{1}{2}} = \bar{\lambda}^k - \frac{\tau - \varepsilon}{s}\left[A(2\widetilde{x}^k - x^k) + By^k - c\right]$;
7    $\widetilde{y}^k = \arg\min\limits_{y \in \mathcal{Y}} \left\{ g(y) + \frac{\bar{\rho}}{2} \left\| B(y - y^k) - \frac{\tau}{\rho}\bar{\lambda}^{k+\frac{1}{2}} \right\|^2 \right\}$;
8    $r^{k+1} = A\widetilde{x}^k + B\widetilde{y}^k - c$; $\Delta x^k = \widetilde{x}^k - x^k$; $\Delta y^k = \widetilde{y}^k - y^k$;
9    $\widetilde{\lambda}^k = \bar{\lambda}^k - \frac{\tau}{s}r^{k+1} - \frac{1}{s}\left[\tau A\Delta x^k + \varepsilon B\Delta y^k\right]$;
10   Relaxation step: compute

$$\begin{pmatrix} x^{k+1} \\ y^{k+1} \end{pmatrix} = \begin{pmatrix} x^k \\ y^k \end{pmatrix} + \gamma \begin{pmatrix} \Delta x^k \\ \Delta y^k \end{pmatrix},$$

and

$$\bar{\lambda}^{k+1} = \bar{\lambda}^k + \gamma(\widetilde{\lambda}^k - \bar{\lambda}^k) - \frac{(1-\gamma)(\tau + \varepsilon)}{s}(A\Delta x^k + B\Delta y^k).$$

---

4 Numerical experiments

In this section, we would perform some numerical experiments for solving the following *lasso* model problem arising from statistical learning [27]:

$$\min_{x \in \mathcal{R}^n} \nu \|x\|_1 + \frac{1}{2}\|Dx - b\|^2, \tag{30}$$

where $\|x\|_1 = \sum_{i=1}^n |x_i|$; $\nu > 0$ is a scalar regularization parameter; $b \in \mathcal{R}^l$ is a response vector; $D \in \mathcal{R}^{l \times n}$ is a design matrix with l and n denoting the number of data points and the number of features, respectively. In typical applications, there are usually many more features than training examples [1], that is $l \ll n$, and the goal is to find a parsimonious model for the data. We refer the readers to [3,27,1,13] for more backgrounds on the *lasso* model. In the numerical experiments, all algorithms are coded in MATLAB 7.10(R2010a) and run on a PC with Intel Core i5 processor (3.3GHz) with 4 GB memory.

By introducing an auxiliary variable $y \in \mathcal{R}^n$, the problem (30) is obviously equivalent to

$$\begin{aligned} &\min \nu \|x\|_1 + \frac{1}{2}\|Dy - b\|^2 \\ &\text{s.t.} \quad x - y = \mathbf{0}, \\ &\qquad x \in \mathcal{R}^n, \ y \in \mathcal{R}^n, \end{aligned} \tag{31}$$

which is a special case of (1) with

$$f(x) = \nu \|x\|_1, \ g(y) = \frac{1}{2}\|Dy - b\|^2, \ A = \mathbf{I}, \ B = -\mathbf{I}, \ c = \mathbf{0}.$$

Applying Algorithm 1 to solve (31), we have

$$x^{k+1} = \arg \min_{x \in \mathcal{R}^n} \left\{ \nu \|x\|_1 + \frac{\bar{\sigma}}{2} \left\| x - x^k - \frac{\tau}{\bar{\sigma}} \bar{\lambda}^k \right\|^2 \right\},$$

for which x^{k+1} can be explicitly obtained by a soft shrinkage operator [8] and $\bar{\sigma}$ is defined in (10). In addition, we can deduce that

$$y^{k+1} = \left[D^T D + \bar{\rho}\mathbf{I}\right]^{-1} \left[D^T b + \bar{\rho}\left(y^k - \frac{\tau}{\bar{\rho}}\bar{\lambda}^{k+\frac{1}{2}}\right)\right],$$

where $\bar{\rho}$ is defined in (10). Notice that the matrix $D^T D + \bar{\rho}\mathbf{I}$ is positive definite, since $\bar{\rho} = (\rho s + \tau^2 - 1)/s > 0$. Though it maybe quite time consuming to compute $(D^T D + \bar{\rho}\mathbf{I})^{-1}$ and $D^T b$ when the problem scale is large, they only need to be computed once before the iteration starts. To save computation, we can actually compute once and cache the Cholesky Factorization of the much smaller matrix $DD^T/\bar{\rho} + \mathbf{I}$ (note $l \ll n$), which takes about $(l^2 n + l^3/3)$ flops, including the cost of forming DD^T and the Cholesky Factorization. Then, all the subsequent y-updates can be calculated by the Sherman-Morrison inversion matrix formula together with forward-backward substitutions.

The problem data are generated by the following way. Each entry of the feature matrix D is draw from the standard normal distribution $\mathcal{N}(0,1)$ and

then each of its column is normalized. A random sparse vector $x^{true} \in \mathcal{R}^n$ with 100 nonzero entries is also drawn from an $\mathcal{N}(0,1)$ distribution. The vector b is computed via $b = Dx^{true} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 10^{-3}\mathbf{I})$, and the regularization parameter is set as $\nu = 0.12\nu_{max}$ with $\nu_{max} = \|D^T b\|_\infty$. The following stopping criterion is used for all the comparison algorithms

$$\text{IRE(k)} := \frac{\|x^k - y^k\|}{\max\{\|x^k\|, \|y^k\|\}} \le \text{Tol} \quad \text{and} \quad \frac{\phi(u^k) - \phi^*}{\phi^*} \le 1.0 \times 10^{-8},$$

where Tol is a given tolerance, $\phi(u^k) = f(x^k) + g(y^k)$, and ϕ^* is the approximate optimal objective function value obtained by running P-PPA after 2000 iterations. Then, all the comparison algorithms are set to have the maximum 2000 number of iterations and they all use the same starting point $(x^0, y^0, \lambda^0) = (\mathbf{0}, \mathbf{0}, \mathbf{0})$.

4.1 Effects of parameters $(\sigma, \rho, s, \tau, \varepsilon)$

The aim of this subsection is to investigate how the five parameters $(\sigma, \rho, s, \tau, \varepsilon)$ would influence the performance of P-PPA. For this purpose, we first fix the free parameters (τ, ε) as $(3, 1.5)$ and then change other distinctively constrained parameters (σ, ρ, s) to investigate their effects on P-PPA.

Table 1 presents the numerical results of Algorithm 1 (i.e. P-PPA) with different parameters for solving the test problem (31) with dimension $(l, n) = (1800, 4000)$. For this set of tests, we fix the tolerance Tol $= 1.0 \times 10^{-6}$. And in all the numerical Tables, "Iter", "CPU" and "DRN" denote the iteration numbers, the CPU time in seconds and the dual residual norm $\|y^k - y^{k-1}\|$, respectively. We can observe from Table 1 that:

- For the parameters (σ, ρ, s), the reported results in each column of IRE, DRN and $\phi(u^k)$ are nearly the same when fixed any two parameters with one parameter changing.
- With the increase of the parameter σ or ρ, both the iteration number and the CPU time tend to increase(denoted by $\downarrow$);
- With the increase of the parameter s, both the iteration number and the CPU time decrease firstly and then increase(denoted by $\updownarrow$).

These changing trends identify with Remark 2. Reported results of Table 1 indicate that the choice of the parameters (σ, ρ, s) could have a great effect on the performance of P-PPA, the value in the place of the arrow is better than others in each subtable, and it seems that setting $(\sigma, \rho, s) = (0.8, 6, 3)$ would be a reasonable choice for solving the test problem (31).

Remark 3 Noting from Table 1 that if any four parameters are fixed, then by (9) the remaining one is clearly subjected to a given domain. For instance, in the top subtable of Table 1 we have from (9) that $\sigma > \frac{37.25}{51} \approx 0.73$. Therefore, we can randomly choose some values in such region to do experiments to find out which one gives relatively better performance.

Parameters	Iter	CPU	IRE	DRN	$\phi(u^k)$
$\sigma(\rho = 6, s = 3)$					
0.8	$\mathbf{134}_\downarrow$	$\mathbf{3.60}_\downarrow$	9.8125e-7	1.0932e-5	19.2402
1	137	3.70	9.3824e-7	1.0508e-5	19.2402
2	149	4.00	9.3133e-7	1.0673e-5	19.2402
4	174	4.49	8.5235e-7	1.0120e-5	19.2402
6	199	5.08	7.8227e-7	9.5316e-6	19.2402
8	223	5.55	7.5285e-7	9.3548e-6	19.2402
10	248	6.25	6.9449e-7	8.7643e-6	19.2402
$\rho(\sigma = 6, s = 3)$					
0.8	$\mathbf{137}_\downarrow$	$\mathbf{3.61}_\downarrow$	6.2401e-7	8.0288e-6	19.2402
1	140	3.70	6.0813e-7	7.8031e-6	19.2402
2	152	3.94	6.5624e-7	8.3108e-6	19.2402
4	176	4.56	7.1727e-7	8.8891e-6	19.2402
7	210	5.22	8.2280e-7	9.9542e-6	19.2402
8	222	5.47	8.2070e-7	9.8666e-6	19.2402
11	255	6.22	8.9283e-7	1.0567e-5	19.2402
$s(\sigma = 6, \rho = 6)$					
1	263	6.48	9.7027e-7	1.4122e-5	19.2402
2	211	5.25	9.8606e-7	1.3624e-5	19.2402
5	197	4.99	4.3585e-7	3.8039e-6	19.2402
7	192	4.84	3.9519e-7	2.4871e-6	19.2402
11	$\mathbf{181}_\downarrow^\uparrow$	$\mathbf{4.67}_\downarrow^\uparrow$	9.9008e-7	2.9591e-6	19.2402
12	195	4.86	8.9998e-7	2.8168e-6	19.2402
14	221	5.47	9.6407e-7	3.3572e-6	19.2402
16	255	6.39	8.7023e-7	2.4315e-6	19.2402
18	282	6.92	8.5695e-7	2.7583e-6	19.2402

Table 1: Results of problem (31) by P-PPA with different parameters (σ, ρ, s).

Remark 4 In a similar way as mentioned in Remark 3, we have

$$\tau^2 < \frac{(\sigma s - 1)(\rho s - 1)}{\varepsilon^2} \quad \text{and} \quad \varepsilon^2 < \frac{(\sigma s - 1)(\rho s - 1)}{\tau^2}.$$

Then, by testing some values of the parameter $\tau(\varepsilon)$ and observing which one performs approximately better, our tuned results are $(\tau, \varepsilon) = (3, 1.5)$. Hence, for further comparative experiments with some state-of-the-art methods, we would use the tuned results $(0.8, 6, 3, 3, 1.5)$ as the default parameter setting for both P-PPA and RP-PPA.

4.2 Comparative experiments

Now, we would like to compare P-PPA and RP-PPA with other two popular methods for solving the problem (31): ADMM[1] [1] and R-PPA [11].

---

[1] Available at http://web.stanford.edu/∼boyd/papers/admm/.

P-PPA(l,n)	Iter	CPU	IRE	$\phi(u^k)$
(1000,4000)	313	4.01	9.7448e-11	19.3556
(1800,4000)	265	6.27	9.5478e-11	19.2402
(1000,10000)	387	11.25	9.6822e-11	19.0072
(1800,10000)	260	13.16	9.6308e-11	20.0529
(1000,16000)	**279**	**11.71**	9.8237e-11	18.6698
(1600,16000)	247	17.24	9.5727e-11	19.7810
(1000,20000)	**316**	**16.43**	9.2613e-11	19.9294
(1800,20000)	**196**	**19.80**	8.9340e-11	19.2038
(1000,24000)	**369**	**22.62**	9.8638e-11	18.5250
(2000,26000)	**212**	**29.26**	9.4504e-11	19.0005
RP-PPA(l,n)	Iter	CPU	IRE	$\phi(u^k)$
(1000,4000)	260	3.28	9.3533e-11	19.3556
(1800,4000)	219	5.20	9.7552e-11	19.2402
(1000,10000)	322	9.45	9.4616e-11	19.0072
(1800,10000)	216	11.09	9.2376e-11	20.0529
(1000,16000)	**232**	**10.00**	9.2286e-11	18.6698
(1600,16000)	206	14.49	9.5818e-11	19.7810
(1000,20000)	**259**	**13.70**	9.3551e-11	19.9294
(1800,20000)	**173**	**17.58**	9.1322e-11	19.2038
(1000,24000)	**302**	**18.64**	9.6840e-11	18.5250
(2000,26000)	**174**	**24.40**	9.5930e-11	19.0005
R-PPA(l,n)	Iter	CPU	IRE	$\phi(u^k)$
(1000,4000)	284	3.67	9.7350e-11	19.3556
(1800,4000)	238	5.68	9.3094e-11	19.2402
(1000,10000)	371	10.85	9.9245e-11	19.0072
(1800,10000)	243	12.27	9.5912e-11	20.0529
(1000,16000)	336	13.99	9.8769e-11	18.6698
(1600,16000)	256	17.74	9.5437e-11	19.7810
(1000,20000)	339	17.47	9.3942e-11	19.9294
(1800,20000)	236	22.80	9.9142e-11	19.2038
(1000,24000)	374	22.92	9.7664e-11	18.5250
(2000,26000)	226	31.13	9.7351e-11	19.0005
ADMM(l,n)	Iter	CPU	IRE	$\phi(u^k)$
(1000,4000)	**100**	**1.36**	9.2692e-11	19.3556
(1800,4000)	**71**	**2.16**	9.2988e-11	19.2402
(1000,10000)	**189**	**5.62**	9.1517e-11	19.0072
(1800,10000)	**126**	**6.80**	8.9039e-11	20.0529
(1000,16000)	285	11.95	9.8136e-11	18.6698
(1600,16000)	**193**	**13.69**	9.2166e-11	19.7810
(1000,20000)	351	18.04	9.1173e-11	19.9294
(1800,20000)	208	20.33	9.9380e-11	19.2038
(1000,24000)	426	26.06	9.5935e-11	18.5250
(2000,26000)	236	32.28	9.7005e-11	19.0005

Table 2: Comparative results of problem (31) with different dimensions[2].

Table 2 reports the numerical results of all comparison methods for solving the problem (31) with different dimensions (l, n). Actually, ADMM uses the downloaded codes but with penalty parameter 1 and a widely used step-length 1.618 when updating the lagrangian multipliers. For R-PPA, the penalty parameter is set as 10, which is reasonably good for this method. Both RP-PPA and R-PPA use the same relaxation factor $\gamma = 1.2$. The tolerance Tol $= 1.0 \times 10^{-10}$ is used for all the testing problems in Table 2. The numerical results of solving the test problem with fixed dimensions $(l, n) = (1800, 20000)$, but under different accurate tolerances, are presented in Table 3. Moreover, the comparative convergence curves of the objective function $\phi(u^k)$, the residual error IRE(k) and the dual residual norm $\|y^k - y^{k-1}\|$ against the number of iterations are shown in Fig. 1.

Tol $= 10^{-5}$	P-PPA	RP-PPA	R-PPA	ADMM
Iter	100	**86**	102	88
CPU	10.53	**9.31**	10.68	9.54
IRE	9.9112e-6	9.0542e-6	9.5777e-6	9.2727e-6
$\phi(u^k)$	19.2038	19.2038	19.2038	19.2037
Tol $= 10^{-8}$	P-PPA	RP-PPA	R-PPA	ADMM
Iter	159	**137**	182	158
CPU	16.80	**14.27**	18.64	16.65
IRE	8.9178e-9	9.7009e-9	9.5103e-9	9.3515e-9
$\phi(u^k)$	19.2038	19.2038	19.2038	19.2038
Tol $= 10^{-11}$	P-PPA	RP-PPA	R-PPA	ADMM
Iter	**214**	**190**	264	234
CPU	**21.72**	**18.48**	27.03	24.08
IRE	9.2468e-12	9.8367e-12	9.5013e-12	9.5918e-12
$\phi(u^k)$	19.2038	19.2038	19.2038	19.2038
Tol $= 10^{-14}$	P-PPA	RP-PPA	R-PPA	ADMM
Iter	**274**	**244**	347	2000
CPU	**26.79**	**24.18**	32.24	192.33
IRE	9.9348e-15	8.9322e-15	9.7695e-15	1.3963e-14
$\phi(u^k)$	19.2038	19.2038	19.2038	19.2038

Table 3: Comparative results of problem (31) under different tolerance errors[3].

We can observe from Tables 2-3 that both P-PPA and RP-PPA perform better than ADMM and R-PPA for the relatively large size problems in terms of both the number of iterations and the CPU time. Another outstanding observation is that RP-PPA can clearly shorten the number of iterations and the CPU time of P-PPA. Besides, ADMM performs the worst for large-size

---

[2] The bold value excluding that of P-PPA is of the smallest in each experiment with respect to (l,n), and the bold value of P-PPA is smaller than that of R-PPA and ADMM.

[3] The bold value of P-PPA is smaller than that of R-PPA and ADMM, and the bold value of RP-PPA is of the smallest.

Fig. 1 Comparative convergence curves of the objective function $\phi(u^k)$, the residual IRE(k) and the dual residual norm $\|y^k - y^{k-1}\|$ against the number of iterations under tolerance Tol $= 1.0 \times 10^{-11}$.

problems, while it performs the best for small-size problems (e.g. $n = 4000$). Both Table 3 and Fig. 1 illustrate that as the tolerance becomes smaller, P-PPA and RP-PPA could perform significantly better than ADMM and R-PPA. In addition, note from Table 3 that ADMM fails to solve the problem with dimensions $(l, n) = (1800, 20000)$ in 2000 iterations to achieve the accuracy Tol $= 1.0 \times 10^{-14}$. Reported results show that our proposed methods are efficient when properly choosing the algorithmic parameters.

Remark 5 Although the tuned values of the parameters in the proposed algorithms are not proved theoretically to be the best, the reported numerical

results of comparative experiments are sufficient to show that such a choice can make our algorithms outperform the other two algorithms.

5 Conclusion and discussion

By introducing several parameters to the proximal matrix in the framework of the traditional proximal point algorithm, we propose a new Parameterized Proximal Point Algorithm (P-PPA) for solving the separable convex minimization problem. Under certain conditions on these parameters, we show that the P-PPA is globally convergent and would maintain a worst-case $O(1/t)$ ergodic convergence rate. By properly choosing the parameters, the numerical experiments of solving the classical *lasso* problem in statistical learning indicate our P-PPA and the Relaxed P-PPA (RP-PPA) could perform significantly better than the other two benchmark methods: ADMM and R-PPA, especially for solving large scale problems and high accurate solutions are required.

For the case that the subproblems are not easy to solve, inexact ADMMs[14] are recently developed for solving the general separable convex optimization problems with a linear constraint and with an objective including smooth plus nonsmooth terms, which is particularly useful when the ADMM's subproblems do not have closed solutions or when the solution of the subproblem is expensive. Also, there are other works in references [16,26] which discuss how to solve the subproblems inexactly.

Finally, observe that the P-PPA and RP-PPA can be naturally extended to to solve the problem (1) with inequality constraints or with matrix variables such as

$$
\begin{aligned}
\min\ &f(X) + g(Y) \\
\text{s.t.}\ \ &AX + BY = C, \\
&X \in \mathcal{X}, Y \in \mathcal{Y},
\end{aligned}
\tag{32}
$$

where both f and g are proper closed convex functions over the matrix variables X and Y, A and B are coefficient matrices, $\mathcal{X}$ and $\mathcal{Y}$ are certain closed convex sets. The model problem (32) also arises very often in many important applications in data analysis [4,20], for example, the robust principal component analysis in image processing, etc. One of the following research tasks could be to extend the P-PPA to solve the multi-block separable convex/nonconvex programming problems.

Acknowledgements

References

1. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Machine Learning, 3, 1-122 (2010)
2. Briceño-Arias, L.M., Combettes, P.L., Pesquet, J.-C., Pustelnik, N.: Proximal algorithms for multicomponent image recovery problems. J. Math. Imaging Vis. 41, 3-22 (2011)
3. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. SIAM Rev. 43, 129-159 (2001)
4. Candès, E.J., Li, X.D., Ma, Y., Wright, J.: Robust principal component analysis? J. ACM, 58, Article 11, (2011)
5. Cai, X.J., Gu, G., He, B.S., Yuan, X.M.: A proximal point algorithm revisit on alternating direction method of multipliers. Sci. China Math. 56, 2179-2186 (2013)
6. Combettes, P.L., Pennanen, T.: Proximal methods for cohypomonotone operators. SIAM J. Control Optim. 43, 731-742 (2004)
7. Combettes, P.L., Pesquet, J.-C.: Proximal thresholding algorithm for minimization over orthonormal bases. SIAM J. Optim. 18, 1351-1376 (2007)
8. Donoho, D.L., Tsaig, Y.: Fast solution of l_1-norm minimization problems when the solution may be sparse. IEEE Trans. Inform. Theory, 54, 4789-4812 (2008)
9. Eckstein, J., Bertsekas, D.P.: On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. 55, 293-318 (1992)
10. Eckstein, J.: Nonlinear proximal point algorithms using Bregman functions, with applications to convex programming. Math. Oper. Res. 18, 202-226 (1993)
11. Gu, G.Y., He, B.S., Yuan, X.M.: Customized proximal point algorithms for linearly constrained convex minimization and saddle-point problems: a unified approach. Comput. Optim. Appl. 59, 135-161 (2014)
12. Gol'shtein, E.G., Tret'yakov, N.V.: Modified Lagrangian in convex programming and their generalizations. Math. Program. Stud. 10, 86-97 (1979)
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, second ed. 2009.
14. Hager, W.W., Zhang, H.C.: Inexact alternating direction multiplier methods for separable convex optimization. arXiv:1604.02494v1, 8 Apr. (2016)
15. He, B.S., Yuan, X.M., Zhang, W.X.: A customized proximal point algorithm for convex minimization with linear constraints. Comput. Optim. Appl. 56, 559-572 (2013)
16. He, B.S., Yuan, X.M.: Linearized alternating direction method of multipliers with Gaussian back substitution for separable convex programming. Numer. Algebra Control Optim. 3, 247-260 (2013)
17. He, B.S., Xu, H.K., Yuan, X.M.: On the proximal Jacobian decomposition of ALM for multiple-block separable convex minimization problems and its relationship to ADMM. J. Sci. Comput. 66, 1204-1217 (2016)
18. He, B.S., Ma, F., Yuan, X.M.: Convergence study on the symmetric version of ADMM with larger step sizes. SIAM J. Imaging Sci. 9, 1467-1501 (2016)
19. Ji, Y., Goh, M., Souz, R.: Proximal point algorithms for multi-criteria optimization with the difference of convex objective functions. J. Optim. Theory Appl. 169, 280-289 (2016)
20. Liu, Z.S., Li, J.C., Li, G., Bai, J.C., Liu, X.N.: A new model for sparse and low rank matrix decomposition. J. Appl. Anal. Comput. 7, 600-616 (2017)
21. Moreau, J.J.: Proximité et dualité dans un espace hilbertien. Bull. Soc. Math. Fr. 93, 273-299 (1965)
22. Martinet, B.: Regularisation, d'inéquations variationelles par approximations succesives. Rev. Fr. Inform. Rech. Oper. 4, 154-159 (1970)
23. Ma, F., Ni, M.F.: A class of customized proximal point algorithms for linearly constrained convex optimization. Comp. Appl. Math. (2016) doi:10.1007/s40314-016-0371-3
24. Rockafellar, R.T.: Augmented Lagrangians and applications of the proximal point algorithm in convex programming. Math. Oper. Res. 1, 97-116 (1976)
25. Powell, M.J.D.: A method for nonlinear constraints in minimization problems. In optimization, Fletcher, R. (ed.), Academic Press, N. Y. 283-298 (1969)
26. Solodov, M.V., Svaiter, B.F.: An inexact hybrid generalized proximal point algorithm and some new results on the theory of Bregman functions. Math. Oper. Res. 25, 214-230 (2000)

27. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Roy. Statist. Soc. 58, 267-288 (1996)
28. Wang, K., Desai, J., He, H.: A proximal partially parallel splitting method for separable convex programs. Optim. Method. Soft. 32, 39-68 (2017)