Computational Optimization and Applications (2021) 80:831–851
https://doi.org/10.1007/s10589-021-00321-3

A parallel splitting ALM-based algorithm for separable convex programming

Shengjie Xu[1,2] · **Bingsheng He**[3]

Received: 9 March 2021 / Accepted: 16 September 2021 / Published online: 25 September 2021

Abstract
The augmented Lagrangian method (ALM) provides a benchmark for solving the canonical convex optimization problem with linear constraints. The direct extension of ALM for solving the multiple-block separable convex minimization problem, however, is proved to be not necessarily convergent in the literature. It has thus inspired a number of ALM-variant algorithms with provable convergence. This paper presents a novel parallel splitting method for the multiple-block separable convex optimization problem with linear equality constraints, which enjoys a larger step size compared with the existing parallel splitting methods. We first show that a fully Jacobian decomposition of the regularized ALM can contribute a descent direction yielding the contraction of proximity to the solution set; then, the new iterate is generated via a simple correction step with an ignorable computational cost. We establish the convergence analysis for the proposed method, and then demonstrate its numerical efficiency by solving an application problem arising in statistical learning.

Keywords Convex programming · Augmented Lagrangian method · Jacobian decomposition · Parallel computing · Operator splitting methods

Mathematics Subject Classification 90C25 · 90C06 · 90C33

✉ Shengjie Xu
xsjnsu@163.com

Bingsheng He
hebma@nju.edu.cn

[1] Department of Mathematics, Harbin Institute of Technology, Harbin, China

[2] Department of Mathematics, Southern University of Science and Technology, Shenzhen, China

[3] Department of Mathematics, Nanjing University, Nanjing, China

1 Introduction

Our discussion starts with the following canonical convex minimization problem with linear equality constraints:

$$\min \{\theta(x) \mid \mathcal{A}x = b,\ x \in \mathcal{X}\}, \tag{1.1}$$

where $\theta : \mathfrak{R}^n \to \mathfrak{R} \cup \{+\infty\}$ is a closed proper convex but not necessarily smooth function; $\mathcal{X} \subseteq \mathfrak{R}^n$ is a closed convex set; $\mathcal{A} \in \mathfrak{R}^{l\times n}$ and $b \in \mathfrak{R}^l$. Among algorithms for solving (1.1), the augmented Lagrangian method (ALM) introduced in [24, 29] turns out to be a fundamental tool in both theoretical and algorithmic aspects, and its associated iterative scheme reads as

$$(\text{ALM})\begin{cases} x^{k+1} = \arg\min \left\{\mathcal{L}_\beta(x, \lambda^k) \mid x \in \mathcal{X}\right\}, \\ \lambda^{k+1} = \lambda^k - \beta(\mathcal{A}x^{k+1} - b), \end{cases} \tag{1.2}$$

where

$$\mathcal{L}_\beta(x, \lambda) := \theta(x) - \lambda^T(\mathcal{A}x - b) + \frac{\beta}{2}\|\mathcal{A}x - b\|_2^2$$

denotes the augmented Lagrangian function of (1.1); $\lambda \in \mathfrak{R}^l$ is the Lagrange multiplier and $\beta > 0$ is a penalty parameter for the linear constraints. As analyzed in [30], the classic ALM is indeed an application of the proximal point algorithm (PPA) in [26] to the dual of (1.1). Throughout our discussion, the penalty parameter β is assumed to be fixed for simplification.

In this paper, we focus on a special case of (1.1), where its objective function is the sum of m subfunctions without coupled variables:

$$\min \left\{ \sum_{i=1}^m \theta_i(x_i) \,\middle|\, \sum_{i=1}^m A_i x_i = b;\ x_i \in \mathcal{X}_i,\ i = 1, 2, \ldots, m \right\}. \tag{1.3}$$

Here, $\theta_i : \mathfrak{R}^{n_i} \to \mathfrak{R} \cup \{+\infty\}$ $(i = 1, \ldots, m)$ are closed proper convex functions and they are not necessarily smooth; $\mathcal{X}_i \subseteq \mathfrak{R}^{n_i}$ $(i = 1, \ldots, m)$ are closed convex sets; $\sum_{i=1}^m n_i = n$; $A_i \in \mathfrak{R}^{l\times n_i}$ $(i = 1, \ldots, m)$ and $b \in \mathfrak{R}^l$. Obviously, the model (1.3) corresponds to (1.1) with $x = (x_1; \ldots; x_m)$, $\mathcal{A} = (A_1, \ldots, A_m)$ and $\theta(x) = \sum_{i=1}^m \theta_i(x_i)$. In practice, the generic model (1.3) finds a multitude of applications in, e.g., [4, 31] for some image reconstruction problems, [32, 35] for the matrix recovery model and the Potts based image segmentation problem, and [5, 6, 33] for a number of problems arising in distributed optimization and statistical learning. We also refer to, e.g., [25, 34], for more applications in other communities. Throughout our discussion, the solution set of (1.3) is assumed to be nonempty and the matrices A_i $(i = 1, \ldots, m)$ are assumed to be full column-rank. In addition, we assume each x_i-subproblem

$$x_i^* = \arg\min \left\{ \theta_i(x_i) + \frac{\beta}{2}\|A_i x_i - q_i\|_2^2 \ \mid\ x_i \in \mathcal{X}_i \right\}, \tag{1.4}$$

with any given $q_i \in \Re^l$ and $\beta > 0$, has a closed-form solution or can be easily solved with a high precision (see, e.g., [27], for specific supported examples).

Applying the classical ALM (1.2) straightforwardly to (1.3), the resulting scheme then reads as

$$\begin{cases} (x_1^{k+1}, \ldots, x_m^{k+1}) = \arg\min \left\{ \mathcal{L}_\beta(x_1, x_2, \ldots, x_m, \lambda^k) \,|\, x_i \in \mathcal{X}_i,\ i = 1, \ldots, m \right\}, \\ \lambda^{k+1} = \lambda^k - \beta\left(\sum_{i=1}^m A_i x_i^{k+1} - b\right), \end{cases}$$

where

$$\mathcal{L}_\beta(x_1, x_2, \ldots, x_m, \lambda) = \sum_{i=1}^m \theta_i(x_i) - \lambda^T \left(\sum_{i=1}^m A_i x_i - b \right) + \frac{\beta}{2} \left\| \sum_{i=1}^m A_i x_i - b \right\|_2^2. \tag{1.5}$$

However, note there are coupled terms in the quadratic term $\| \sum_{i=1}^m A_i x_i - b\|_2^2$. The x_i-subproblems ($i = 1, \ldots, m$) generally can not be tackled simultaneously. Exploiting the separable structure of the objective function θ, a common-used separable strategy for x-subproblem is the following Gaussian decomposition iterative scheme:

$$\begin{cases} \begin{cases} x_1^{k+1} = \arg\min \left\{ \mathcal{L}_\beta(x_1, x_2^k, \ldots, x_{m-1}^k, x_m^k, \lambda^k) \mid x_1 \in \mathcal{X}_1 \right\}, \\ x_2^{k+1} = \arg\min \left\{ \mathcal{L}_\beta(x_1^{k+1}, x_2, \ldots, x_{m-1}^k, x_m^k, \lambda^k) \mid x_2 \in \mathcal{X}_2 \right\}, \\ \qquad \vdots \\ x_m^{k+1} = \arg\min \left\{ \mathcal{L}_\beta(x_1^{k+1}, x_2^{k+1}, \ldots, x_{m-1}^{k+1}, x_m, \lambda^k) \mid x_m \in \mathcal{X}_m \right\}, \end{cases} \\ \lambda^{k+1} = \lambda^k - \beta\left(\sum_{i=1}^m A_i x_i^{k+1} - b\right). \end{cases} \tag{1.6}$$

Clearly, each x_i-subproblem in (1.6) possesses the favorable composition (1.4) and can be treated individually. In particular, for the case $m = 2$, the method (1.6) corresponds to the classical alternating direction method of multipliers (ADMM) proposed in [9, 10], which is indeed an application of the Douglas-Rachford splitting method (see [8]). The generic scheme (1.6) is thus named the direct extension of ADMM (D-ADMM) in [7, 10]. For the case $m \geq 3$, the D-ADMM has been shown to be empirically efficient for various applications, see, e.g., [28, 32] and references cited therein. However, a counterexample in [7] reveals that the D-ADMM (1.6) is not necessarily convergent. It has immediately inspired a number of works such as [11, 13, 17–19, 23, 25, 31–33]. Especially, the ADMM with Gaussian back substitution in [12, 17, 19], which needs only an additional back substitution step based on the D-ADMM, turns out to be a convergent yet numerically well-preforming method.

We now turn our attention to another common-used splitting strategy for the x-subproblem in the generic ALM scheme (1.2). Applying the full Jacobian-decomposition of x-subproblem to (1.2), it immediately leads to the following so-named direct extension of ALM (abbreviated as D-ALM):

$$\text{(D-ALM)}\begin{cases} \begin{cases} x_1^{k+1} = \arg\min\left\{\mathcal{L}_\beta(x_1, x_2^k, \ldots, x_{m-1}^k, x_m^k, \lambda^k) \mid x_1 \in \mathcal{X}_1\right\}, \\ x_2^{k+1} = \arg\min\left\{\mathcal{L}_\beta(x_1^k, x_2, \ldots, x_{m-1}^k, x_m^k, \lambda^k) \mid x_2 \in \mathcal{X}_2\right\}, \\ \qquad \vdots \\ x_m^{k+1} = \arg\min\left\{\mathcal{L}_\beta(x_1^k, x_2^k, \ldots, x_{m-1}^k, x_m, \lambda^k) \mid x_m \in \mathcal{X}_m\right\}, \end{cases} \\ \lambda^{k+1} = \lambda^k - \beta\big(\sum_{i=1}^m A_i x_i^{k+1} - b\big). \end{cases}$$

It is obvious that each x_i-subproblem in the above D-ALM can be solved in a parallel way. In practice, there are many applications (see [36]) in the medical image processing field such as the Potts-model based image segmentation problem and its variants, can be solved efficiently by using the D-ALM. However, as analyzed in [14], the D-ALM is proved to be not necessarily convergent in theory. Let $w := (x_1; \ldots; x_m; \lambda)$. To ensure the convergence, in [14], the authors regard first the output of the D-ALM as a predictor $\tilde{w}^k$ (i.e., $(\tilde{x}_1^k; \ldots; \tilde{x}_m^k; \tilde{\lambda}^k) := (x_1^{k+1}; \ldots; x_m^{k+1}; \lambda^{k+1})$), then the new iterate w^{k+1} is updated via

$$w^{k+1} = w^k - \alpha(w^k - \tilde{w}^k) \quad \text{with} \quad \alpha \in (0,\ 2(1 - \sqrt{m/(m+1)})).$$

Since the modification is simple, the variant is also named the parallel splitting ALM (denoted by P-ALM) in [14]. It is easy to verify that, as the increase of m, the step size α would be tiny (for example, if $m = 3$, then $\alpha \in (0, 0.268)$), which would adversely affect the convergence efficiency in the real computations. In [18], by adding extra regularization terms to the D-ALM, it directly leads to the following parallel splitting ADMM-like scheme (denoted by PS-ADMM):

$$\begin{cases} x_1^{k+1} = \arg\min\left\{\mathcal{L}_\beta(x_1, x_2^k, \ldots, x_{m-1}^k, x_m^k, \lambda^k) \mid x_1 \in \mathcal{X}_1\right\}, \\ \begin{cases} x_2^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1^{k+1}, x_2, \ldots, x_{m-1}^k, x_m^k, \lambda^k) \\ \qquad + \dfrac{\tau}{2}\beta\|A_2(x_2 - x_2^k)\|_2^2 \mid x_2 \in \mathcal{X}_2\}, \\ \qquad \vdots \\ x_m^{k+1} = \arg\min\left\{\mathcal{L}_\beta(x_1^{k+1}, x_2^k, \ldots, x_{m-1}^k, x_m, \lambda^k) \right. \\ \qquad \left. + \dfrac{\tau}{2}\beta\|A_m(x_m - x_m^k)\|_2^2 \mid x_m \in \mathcal{X}_m\right\}, \end{cases} \\ \lambda^{k+1} = \lambda^k - \beta(\sum_{i=1}^m A_i x_i^{k+1} - b), \end{cases}$$

in which $\tau > m - 2$. Clearly, it would correspond to a huge regularization term (or equivalently, a tiny step size) as the increase of m. Recent works such as [15, 16, 20] have demonstrated that relaxation of regularization term allows a bigger step size to potentially accelerate the convergence. Hence, it is necessary to reduce such a regularization factor τ.

The primary purpose of the paper is to present a novel parallel splitting ALM-based algorithm for the multiple-block separable convex programming problem (1.3). More concretely, our new method begins with the following fully parallel regularized ALM-based scheme:

$$\begin{cases} \tilde{x}_1^k = \arg\min\{\mathcal{L}_\beta(x_1, x_2^k, \dots, x_{m-1}^k, x_m^k, \lambda^k) \\ \qquad + \frac{\tau}{2}\beta\|A_1(x_1 - x_1^k)\|_2^2 \mid x_1 \in \mathcal{X}_1\}, \\ \tilde{x}_2^k = \arg\min\{\mathcal{L}_\beta(x_1^k, x_2, \dots, x_{m-1}^k, x_m^k, \lambda^k) \\ \qquad + \frac{\tau}{2}\beta\|A_2(x_2 - x_2^k)\|_2^2 \mid x_2 \in \mathcal{X}_2\}, \\ \qquad\qquad \vdots \\ \tilde{x}_m^k = \arg\min\left\{\mathcal{L}_\beta(x_1^k, x_2^k, \dots, x_{m-1}^k, x_m, \lambda^k) \right. \\ \qquad \left. + \frac{\tau}{2}\beta\|A_m(x_m - x_m^k)\|_2^2 \mid x_m \in \mathcal{X}_m\right\}, \\ \tilde{\lambda}^k = \lambda^k - \beta\left(\sum_{i=1}^m A_i x_i^k - b\right), \end{cases} \tag{1.7}$$

where $\tau > (m-4)/4$. As can be seen easily, the scheme (1.7) possesses a fully parallel splitting structure for both x_i-subproblems ($i = 1, \dots, m$) and λ-subproblem. It thus belongs to the category of parallel operator splitting schemes. Furthermore, as will be shown in Sect. 3.1, the scheme (1.7) can provide a descent direction yielding the contraction of proximity to the solution set of (1.3). It is thus inspired to generate the new iterate w^{k+1} via

$$w^{k+1} = w^k + \alpha d(w^k, \tilde{w}^k) := w^k - \alpha M(w^k - \tilde{w}^k), \tag{1.8}$$

where $\alpha \in (0, 1)$ and the correction matrix M will be defined in (2.11). Note that the step size α is a constant. The correction step (1.8), as the initialization of next iteration, could be handled easily. Moreover, we give a more practical and succinct scheme (3.7) to replace the method (1.7)–(1.8) in Sect. 3.3.

The novel algorithm (1.7)–(1.8) enjoys great advantages in mainly two folds: first, compared with the P-ALM, the constant step size $\alpha \in (0, 1)$ can be taken more broadly; second, it reduces the regularization factor τ from $(m-2)$ to $(m-4)/4$ compared with the PS-ADMM, so it provides a wider choice of the regularization term to potentially accelerate the convergence. Also, we show the global convergence of the proposed method, and then illustrate its efficiency by extensive numerical experiments.

The rest of the paper is organized as follows. In Sect. 2, we summarize some preliminaries and fundamental matrices, which will be useful for further analysis. Then, we propose the novel method in Sect. 3 and establish its convergence analysis in Sect. 4. The numerical efficiency of the proposed method is further demonstrated in Sect. 5. Some conclusions are drawn in Sect. 6.

2 Preliminaries

In this section, we summarize some preliminary results that will be used frequently throughout our discussion. The analysis of the paper is based on the following elementary lemma (its proof can be found in, e.g., [2]).

Lemma 1 *Let $\mathcal{X} \subseteq \Re^n$ be a closed convex set, $\theta(x)$ and $\varphi(x)$ be convex functions. If $\varphi(x)$ is differentiable on an open set which contains $\mathcal{X}$, and the solution set of the convex minimization problem* $\min\{\theta(x) + \varphi(x) \mid x \in \mathcal{X}\}$ *is nonempty, then we have*

$$x^* \in \arg\min\{\theta(x) + \varphi(x) \mid x \in \mathcal{X}\}$$

if and only if

$$x^* \in \mathcal{X}, \quad \theta(x) - \theta(x^*) + (x - x^*)^T \nabla\varphi(x^*) \geq 0, \quad \forall\, x \in \mathcal{X}.$$

2.1 A variational characterization of (1.3)

To begin with, using the similar techniques in, e.g., [22], we show first how to stand for the optimality condition of (1.3) in the variational inequality context. By attaching the Lagrange multiplier $\lambda \in \Re^l$ to the linear constraints, the Lagrange function of (1.3) reads as

$$L(x_1, x_2, \ldots, x_m, \lambda) = \sum_{i=1}^{m} \theta_i(x_i) - \lambda^T \left(\sum_{i=1}^{m} A_i x_i - b \right), \tag{2.1}$$

and it is defined on the set $\Omega := \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_m \times \Re^l$. Suppose the pair $(x_1^*, x_2^*, \ldots, x_m^*, \lambda^*) \in \Omega$ is a saddle point of (2.1). Then, for any $\lambda \in \Re^l$ and $x_i \in \mathcal{X}_i\,(i = 1, \ldots, m)$, we have

$$L(x_1^*, x_2^*, \ldots, x_m^*, \lambda) \leq L(x_1^*, x_2^*, \ldots, x_m^*, \lambda^*) \leq L(x_1, x_2, \ldots, x_m, \lambda^*).$$

In view of Lemma 1, it is equivalent to finding $(x_1^*, \ldots, x_m^*, \lambda^*)$ such that

$$\begin{cases} \theta_1(x_1) - \theta_1(x_1^*) + (x_1 - x_1^*)^T(-A_1^T\lambda^*) \geq 0, & \forall\, x_1 \in \mathcal{X}_1, \\ \qquad\qquad \vdots \\ \theta_m(x_m) - \theta_m(x_m^*) + (x_m - x_m^*)^T(-A_m^T\lambda^*) \geq 0, & \forall\, x_m \in \mathcal{X}_m, \\ (\lambda - \lambda^*)^T\left(\sum_{i=1}^{m} A_i x_i^* - b\right) \geq 0, & \forall\, \lambda \in \Re^l. \end{cases} \tag{2.2}$$

Furthermore, if we set

$$x=\begin{pmatrix}x_1\\ \vdots\\ x_m\end{pmatrix},\quad \theta(x)=\sum_{i=1}^m\theta_i(x_i),\quad \mathcal{A}=(A_1,A_2,\ldots,A_m),$$
$$w=\begin{pmatrix}x_1\\ \vdots\\ x_m\\ \lambda\end{pmatrix}\quad\text{and}\quad F(w)=\begin{pmatrix}-A_1^T\lambda\\ \vdots\\ -A_m^T\lambda\\ \sum_{i=1}^m A_ix_i-b\end{pmatrix}, \tag{2.3}$$

the inequalities (2.2) can be compactly rewritten as

$$\mathrm{VI}(\Omega,F,\theta):\quad w^*\in\Omega,\quad \theta(x)-\theta(x^*)+(w-w^*)^TF(w^*)\geq 0,\quad \forall\, w\in\Omega. \tag{2.4}$$

In the later sections, we denote by Ω^* the solution set of (2.4), which is assumed to be nonempty. On the one hand, since the operator $F(w)$ in (2.3) is affine with a skew-symmetric matrix, we obtain

$$(w-\bar{w})^T(F(w)-F(\bar{w}))\equiv 0,\quad \forall\, w,\ \bar{w}\in\Omega, \tag{2.5}$$

which means F is monotone. On the other hand, the objective function $\theta(x)$ is not necessarily differentiable. With this regard, we also name (2.4) the mixed monotone variational inequality. In the following, we say (2.4) the variational inequality (VI) for short.

2.2 A variational characterization of (1.7)

The associated VI-structure of the introduced parallel splitting scheme (1.7) can be deduced by the following fundamental lemma.

Lemma 2 *Let $\{\tilde{w}^k\}$ be the sequence generated by (1.7) for solving (1.3). Then, we have*

$$\theta(x)-\theta(\tilde{x}^k)+(w-\tilde{w}^k)^TF(\tilde{w}^k)\geq(w-\tilde{w}^k)^TQ(w^k-\tilde{w}^k),\quad \forall\, w\in\Omega, \tag{2.6}$$

where

$$Q=\begin{pmatrix}(1+\tau)\beta A_1^TA_1 & \cdots & 0 & 0\\ \vdots & \ddots & \vdots & \vdots\\ 0 & \cdots & (1+\tau)\beta A_m^TA_m & 0\\ -A_1 & \cdots & -A_m & \frac{1}{\beta}I_l\end{pmatrix}. \tag{2.7}$$

Proof To begin with, for each x_i-subproblem in (1.7), it follows from the fundamental Lemma 1 that $\tilde{x}_i^k\in\mathcal{X}_i$, and $\tilde{x}_i^k$ satisfies

$$\theta_i(x_i) - \theta_i(\tilde{x}_i^k) + (x_i - \tilde{x}_i^k)^T \left\{ -A_i^T \left(\lambda^k - \beta \left(A_i \tilde{x}_i^k + \sum_{j \neq i} A_j x_j^k - b \right) \right) + \tau \beta A_i^T A_i (\tilde{x}_i^k - x_i^k) \right\} \geq 0, \ \forall\, x_i \in \mathcal{X}_i.$$

Since $\tilde{\lambda}^k = \lambda^k - \beta(\mathcal{A}x^k - b)$ in (1.7), we obtain

$$\begin{aligned} & - A_i^T \left(\lambda^k - \beta \left(A_i \tilde{x}_i^k + \sum_{j \neq i} A_j x_j^k - b \right) \right) + \tau \beta A_i^T A_i (\tilde{x}_i^k - x_i^k) \\ & \quad = -A_i^T [\lambda^k - \beta(\mathcal{A}x^k - b)] + \beta A_i^T A_i (\tilde{x}_i^k - x_i^k) + \tau \beta A_i^T A_i (\tilde{x}_i^k - x_i^k) \\ & \quad = -A_i^T \tilde{\lambda}^k + (1+\tau)\beta A_i^T A_i (\tilde{x}_i^k - x_i^k). \end{aligned}$$

Therefore, the x_i-subproblem in (1.7) satisfies

$$\begin{aligned} & \theta_i(x_i) - \theta_i(\tilde{x}_i^k) \\ & \quad + (x_i - \tilde{x}_i^k)^T \left\{ -A_i^T \tilde{\lambda}^k + (1+\tau)\beta A_i^T A_i (\tilde{x}_i^k - x_i^k) \right\} \geq 0, \quad \forall\, x_i \in \mathcal{X}_i. \end{aligned} \tag{2.8}$$

For the λ-subproblem in (1.7), it is easy to see that $\mathcal{A}x^k - b + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k) = 0$, which is also equivalent to

$$(\lambda - \tilde{\lambda}^k)^T \{ \mathcal{A}\tilde{x}^k - b - \mathcal{A}(\tilde{x}^k - x^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k) \} = 0, \quad \forall\, \lambda \in \mathfrak{R}^l. \tag{2.9}$$

Using the notations in (2.3), the assertion of the lemma follows immediately by adding (2.8) and (2.9). □

2.3 Some basic matrices

We list some fundamental matrices which will be useful for further discussion. Recall the matrix Q defined in (2.7). Setting $S = Q^T + Q$, we get

$$\begin{aligned} S = Q^T + Q &= \begin{pmatrix} 2(1+\tau)\beta A_1^T A_1 & \cdots & 0 & -A_1^T \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 2(1+\tau)\beta A_m^T A_m & -A_m^T \\ -A_1 & \cdots & -A_m & \frac{2}{\beta} I_l \end{pmatrix} \\ &= \operatorname{diag}\left(\begin{pmatrix} A_1^T \\ \vdots \\ A_m^T \\ I_l \end{pmatrix} \right) \begin{pmatrix} 2(1+\tau)\beta I_l & \cdots & 0 & -I_l \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 2(1+\tau)\beta I_l & -I_l \\ -I_l & \cdots & -I_l & \frac{2}{\beta} I_l \end{pmatrix} \operatorname{diag}\left(\begin{pmatrix} A_1 \\ \vdots \\ A_m \\ I_l \end{pmatrix} \right). \end{aligned} \tag{2.10}$$

Note that the matrices A_i $(i = 1, \ldots, m)$ are assumed to be full column-rank. Clearly, the matrix S is symmetric, and $S \succ 0$ provided $\tau > (m-4)/4$. Throughout,

$\tau > (m-4)/4$ is required to guarantee the positive definiteness of the matrix S. Furthermore, we define $M = Q^{-T}S$, and thus

$$M = \frac{1}{1+\tau}\mathrm{diag}((A_1^TA_1)^{-1}, \ldots, (A_m^TA_m)^{-1}, I_l) \begin{pmatrix} (2\tau+1)A_1^TA_1 & -A_1^TA_2 & \cdots & -A_1^TA_m & \frac{1}{\beta}A_1^T \\ -A_2^TA_1 & (2\tau+1)A_2^TA_2 & \cdots & -A_2^TA_m & \frac{1}{\beta}A_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -A_m^TA_1 & -A_m^TA_2 & \cdots & (2\tau+1)A_m^TA_m & \frac{1}{\beta}A_m^T \\ -(1+\tau)\beta A_1 & -(1+\tau)\beta A_2 & \cdots & -(1+\tau)\beta A_m & 2(1+\tau)I_l \end{pmatrix}. \tag{2.11}$$

Note that these inverses $(A_i^TA_i)^{-1}$ $(i = 1, \ldots, m)$ are just for algebraically showing the correction step explicitly, and they can be completely avoided (see Sect. 3.3). We also set

$$H = QS^{-1}Q^T, \tag{2.12}$$

and

$$\bar{M} = \begin{pmatrix} \frac{2\tau+1}{1+\tau}I_l & -\frac{1}{1+\tau}I_l & \cdots & -\frac{1}{1+\tau}I_l & \frac{1}{(1+\tau)\beta}I_l \\ -\frac{1}{1+\tau}I_l & \frac{2\tau+1}{1+\tau}I_l & \cdots & -\frac{1}{1+\tau}I_l & \frac{1}{(1+\tau)\beta}I_l \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\frac{1}{1+\tau}I_l & -\frac{1}{1+\tau}I_l & \cdots & \frac{2\tau+1}{1+\tau}I_l & \frac{1}{(1+\tau)\beta}I_l \\ -\beta I_l & -\beta I_l & \cdots & -\beta I_l & 2I_l \end{pmatrix}. \tag{2.13}$$

3 Algorithm

Based on the framework of contraction method (see, e.g., [3]), we present the novel parallel algorithm (1.7)–(1.8) in this section.

3.1 A descent direction of the distance function induced by (1.7)

Recall the VI-structure of (1.7). The predictor $\tilde{w}^k$ generated by (1.7) satisfies

$$\theta(x) - \theta(\tilde{x}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) \geq (w - \tilde{w}^k)^T Q(w^k - \tilde{w}^k), \quad \forall\, w \in \Omega. \tag{3.1}$$

Without loss of generality, we assume $w^k \neq \tilde{w}^k$ throughout our discussion. Otherwise, it follows from (2.4) that $\tilde{w}^k$ would be an optimal solution. Let w^* be an arbitrary solution of (2.4). Setting $w = w^*$ in (3.1), we have

$$\begin{aligned}(\tilde{w}^k - w^*)^T Q(w^k - \tilde{w}^k) &\geq \theta(\tilde{x}^k) - \theta(x^*) + (\tilde{w}^k - w^*)^T F(\tilde{w}^k) \\ &\overset{(2.5)}{=} \theta(\tilde{x}^k) - \theta(x^*) + (\tilde{w}^k - w^*)^T F(w^*) \geq 0.\end{aligned}$$

Using $\tilde{w}^k - w^* = (\tilde{w}^k - w^k) + (w^k - w^*)$ and $w^T Q w = \frac{1}{2} w^T (Q^T + Q) w$, the above inequality can be rewritten as

$$(w^k - w^*)^T Q(w^k - \tilde{w}^k) \geq (w^k - \tilde{w}^k)^T Q(w^k - \tilde{w}^k) \overset{(2.10)}{=} \frac{1}{2}\|w^k - \tilde{w}^k\|_S^2. \tag{3.2}$$

Taking the nonsingularity of the matrices Q and H (see (2.7) and (2.12)) into consideration, the inequality (3.2) can be further reformulated as

$$\begin{aligned}&\left\langle \nabla\Big(\frac{1}{2}\|w - w^*\|_H^2\Big)\Big|_{w=w^k} \,,\, -H^{-1}Q(w^k - \tilde{w}^k)\right\rangle \\ &\quad = -(w^k - w^*)^T Q(w^k - \tilde{w}^k) \leq -\frac{1}{2}\|w^k - \tilde{w}^k\|_S^2.\end{aligned}$$

Therefore, for any fixed $w^* \in \Omega^*$,

$$d(w^k, \tilde{w}^k) := -H^{-1}Q(w^k - \tilde{w}^k) = -Q^{-T}S(w^k - \tilde{w}^k) = -M(w^k - \tilde{w}^k)$$

is a descent direction of the unknown distance function $\|w - w^*\|_H^2$ at the point w^k. According to the same analysis in [21], such a direction is able to yield the contraction of proximity to the solution set of (1.3) if an appropriate step size is taken. Hence, we generate the new iterate w^{k+1} via

$$w^{k+1} = w^k + \alpha_k d(w^k, \tilde{w}^k) = w^k - \alpha_k M(w^k - \tilde{w}^k). \tag{3.3}$$

It remains to find a step size α_k to make the update w^{k+1} closer to Ω^*.

3.2 A befitting step size α_k in the correction step (3.3)

We now turn to choose an appropriate step size α_k in the correction step (3.3). Since

$$\begin{aligned}&\|w^{k+1} - w^*\|_H^2 \\ &\quad = \|w^k - \alpha Q^{-T}S(w^k - \tilde{w}^k) - w^*\|_H^2 \\ &\quad = \|w^k - w^*\|_H^2 - 2\alpha(w^k - w^*)^T Q(w^k - \tilde{w}^k) + \alpha^2\|w^k - \tilde{w}^k\|_S^2 \\ &\quad \overset{(3.2)}{\leq} \|w^k - w^*\|_H^2 - \alpha\|w^k - \tilde{w}^k\|_S^2 + \alpha^2\|w^k - \tilde{w}^k\|_S^2,\end{aligned}$$

we have

$$\|w^k - w^*\|_H^2 - \|w^{k+1} - w^*\|_H^2 \geq \alpha\|w^k - \tilde{w}^k\|_S^2 - \alpha^2\|w^k - \tilde{w}^k\|_S^2 =: q(\alpha).$$

Maximizing the quadratic term $q(\alpha)$, it follows that

$$\alpha_k^* = \frac{\|w^k - \tilde{w}^k\|_S^2}{2\|w^k - \tilde{w}^k\|_S^2} = \frac{1}{2}.$$

Meanwhile, note that $q(\alpha)$ is a lower bounded quadratic contraction function. Refer to the similar technique in [14], we can introduce a relaxation factor $\gamma \in (0,2)$ and thus $\alpha_k := \gamma\alpha_k^* \in (0,1)$, which means a fixed step size $\alpha_k \in (0,1)$ can be taken in (3.3). Consequently, the correction step (3.3), as an update process, can be tackled easily.

3.3 The practical computing scheme for the new method (1.7)–(1.8)

Since the matrix M defined in (2.11) contains some inverses $(A_i^TA_i)^{-1}$ $(i = 1,\dots,m)$, the corrector (3.3) is relatively complicated. Refer to the similar analysis in [23], these inverses are just for algebraically showing the correction step explicitly, and they can be completely avoided in computation empirically. More concretely, note that the variables $(A_1x_1^k,\dots,A_mx_m^k,\lambda^k)$ is essentially required in each iteration of (1.7). Using the same technique in, e.g., [19, 23], the correction step (3.3) can be replaced with

$$\begin{pmatrix} A_1x_1^{k+1} \\ \vdots \\ A_mx_m^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} A_1x_1^{k} \\ \vdots \\ A_mx_m^{k} \\ \lambda^{k} \end{pmatrix} - \alpha\bar{M}\begin{pmatrix} A_1x_1^{k} - A_1\tilde{x}_1^{k} \\ \vdots \\ A_mx_m^{k} - A_m\tilde{x}_m^{k} \\ \lambda^{k} - \tilde{\lambda}^{k} \end{pmatrix}, \tag{3.4}$$

where the matrix $\bar{M}$ is defined in (2.13), and it can be concretely written as

$$\begin{cases} A_ix_i^{k+1} = A_ix_i^k - \alpha\Big\{\frac{2\tau+1}{1+\tau}A_i(x_i^k-\tilde{x}_i^k) - \frac{1}{1+\tau}\Big[\sum_{j\neq i}A_j(x_j^k-\tilde{x}_j^k) \\ \qquad\qquad -\frac{1}{\beta}(\lambda^k-\tilde{\lambda}^k)\Big]\Big\}, \quad i=1,\dots,m, \\ \lambda^{k+1} \quad = \lambda^k - \alpha\big\{-\beta\mathcal{A}(x^k-\tilde{x}^k) + 2(\lambda^k-\tilde{\lambda}^k)\big\}. \end{cases} \tag{3.5}$$

We now give a succinct representation of (3.5) for easy-implementation. Recall $\tilde{\lambda}^k = \lambda^k - \beta(\mathcal{A}x^k - b)$ in (1.7). By substituting it into (3.5), it follows that

$$\begin{aligned} &A_ix_i^{k+1} \\ &= A_ix_i^k - \alpha\left\{\frac{2\tau+1}{1+\tau}A_i(x_i^k-\tilde{x}_i^k) - \frac{1}{1+\tau}\left[\sum_{j\neq i}A_j(x_j^k-\tilde{x}_j^k) - \frac{1}{\beta}(\lambda^k-\tilde{\lambda}^k)\right]\right\} \\ &= A_ix_i^k - \alpha\left\{\frac{2\tau+1}{1+\tau}A_i(x_i^k-\tilde{x}_i^k) - \frac{1}{1+\tau}\left[\sum_{j\neq i}A_j(x_j^k-\tilde{x}_j^k) - (\mathcal{A}x^k-b)\right]\right\} \\ &= A_ix_i^k - \alpha\Big\{2A_i(x_i^k-\tilde{x}_i^k) + \frac{1}{1+\tau}\big(\mathcal{A}\tilde{x}^k-b\big)\Big\}, \end{aligned}$$

and

$$\begin{aligned}\lambda^{k+1} &= \lambda^k - \alpha\{-\beta\mathcal{A}(x^k - \tilde{x}^k) + 2(\lambda^k - \tilde{\lambda}^k)\}\\ &= \lambda^k - \alpha\{-\beta\mathcal{A}(x^k - \tilde{x}^k) + 2\beta(\mathcal{A}x^k - b)\}\\ &= \lambda^k - \alpha\beta\{\mathcal{A}x^k + \mathcal{A}\tilde{x}^k - 2b\}.\end{aligned}$$

Therefore, the practical correction step (3.5) can be reformulated concisely as

$$\begin{cases}\begin{cases}\text{for} \quad i = 1, \ldots, m, \quad \text{do:}\\ A_i x_i^{k+1} \;=\; A_i x_i^k - \alpha\Big\{2A_i(x_i^k - \tilde{x}_i^k) + \frac{1}{1+\tau}(\mathcal{A}\tilde{x}^k - b)\Big\},\end{cases}\\ \lambda^{k+1} \;=\; \lambda^k - \alpha\beta(\mathcal{A}x^k + \mathcal{A}\tilde{x}^k - 2b).\end{cases} \tag{3.6}$$

It further leads to the following more practical and succinct scheme to replace the proposed algorithm (1.7)–(1.8):

$$\begin{cases}\text{(Parallel)}\begin{cases}\tilde{x}_1^k = \arg\min\Big\{\mathcal{L}_\beta(x_1, x_2^k, \ldots, x_{m-1}^k, x_m^k, \lambda^k)\\ \qquad +\frac{\tau}{2}\beta\|A_1(x_1 - x_1^k)\|_2^2 \mid x_1 \in \mathcal{X}_1\Big\},\\ \tilde{x}_2^k = \arg\min\Big\{\mathcal{L}_\beta(x_1^k, x_2, \ldots, x_{m-1}^k, x_m^k, \lambda^k)\\ \qquad +\frac{\tau}{2}\beta\|A_2(x_2 - x_2^k)\|_2^2 \mid x_2 \in \mathcal{X}_2\Big\},\\ \qquad\vdots\\ \tilde{x}_m^k \;= \arg\min\Big\{\mathcal{L}_\beta(x_1^k, x_2^k, \ldots, x_{m-1}^k, x_m, \lambda^k)\\ \qquad +\frac{\tau}{2}\beta\|A_m(x_m - x_m^k)\|_2^2 \mid x_m \in \mathcal{X}_m\Big\},\end{cases}\\ \text{(Update)}\begin{cases}\text{for} i = 1, \ldots, m, \quad \text{do:}\\ A_i x_i^{k+1} \;=\; A_i x_i^k - \alpha\Big\{2A_i(x_i^k - \tilde{x}_i^k) + \frac{1}{1+\tau}(\mathcal{A}\tilde{x}^k - b)\Big\},\\ \lambda^{k+1} \;=\; \lambda^k - \alpha\beta(\mathcal{A}x^k + \mathcal{A}\tilde{x}^k - 2b),\end{cases}\end{cases} \tag{3.7}$$

in which $\alpha \in (0, 1)$ and $\tau > (m-4)/4$. Figure 1 shows an ideal parallel implementation mechanism of the proposed algorithm (3.7) for the case where each x_i-subproblem needs a high computational cost. In such a case, every x_i-subproblem can be solved in an independent CPU platform.

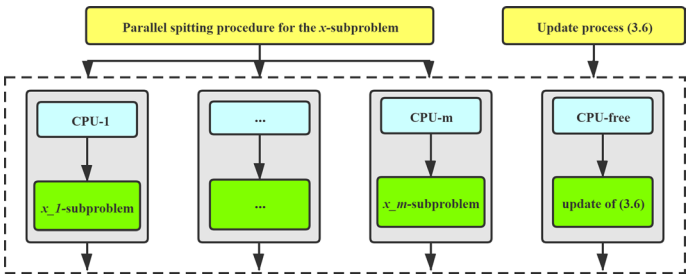

Fig. 1 An ideal parallel implementation mechanism of the proposed algorithm (3.7)

4 Convergence

We establish the convergence analysis of the proposed method (1.7)–(1.8) in this section. The technique of analysis is motivated by the work in [19, 23]. As we have mentioned, $\tau > (m-4)/4$ is required to guarantee the positive definiteness of the matrices S and H. To show the global convergence of the novel algorithm (1.7)–(1.8), we first prove a lemma, followed by a theorem.

Lemma 3 *Let $\{\tilde{w}^k\}$ and $\{w^k\}$ be the sequences generated by the algorithm (1.7)–(1.8) for (1.3). Then, for any constant $\alpha \in (0,1)$, we have*

$$\|w^k - w^*\|_H^2 - \|w^{k+1} - w^*\|_H^2 \;\geq\; \alpha(1-\alpha)\|w^k - \tilde{w}^k\|_S^2. \tag{4.1}$$

Proof It follows from (3.3) that

$$\begin{aligned}
&\|w^{k+1} - w^*\|_H^2 \\
&= \|w^k - \alpha Q^{-T} S(w^k - \tilde{w}^k) - w^*\|_H^2 \\
&= \|w^k - w^*\|_H^2 - 2\alpha(w^k - w^*)^T Q(w^k - \tilde{w}^k) + \alpha^2\|w^k - \tilde{w}^k\|_S^2 \\
&\overset{(3.2)}{\leq} \|w^k - w^*\|_H^2 - \alpha(1-\alpha)\|w^k - \tilde{w}^k\|_S^2,
\end{aligned}$$

and the proof is complete. □

Theorem 1 *Let $\{\tilde{w}^k\}$ and $\{w^k\}$ be the sequences generated by the method (1.7)–(1.8) for solving (1.3). Then, for any $\alpha \in (0,1)$, we obtain $\lim_{k\to\infty} w^k = w^\infty$ where w^∞ belongs to Ω^*.*

Proof First of all, it follows from (4.1) that the generated sequence $\{w^k\}$ satisfies

$$\begin{aligned}
\|w^{k+1} - w^*\|_H^2 &\leq \|w^k - w^*\|_H^2 - \alpha(1-\alpha)\|w^k - \tilde{w}^k\|_S^2 \\
&\leq \|w^k - w^*\|_H^2 \leq \cdots \leq \|w^0 - w^*\|_H^2,
\end{aligned}$$

which means the sequence $\{w^k\}$ is bounded. Furthermore, summing (4.1) over $k = 0, 1, \ldots, \infty$, we obtain

$$\begin{aligned}
\sum_{k=0}^{\infty} \alpha(1-\alpha)\|w^k - \tilde{w}^k\|_S^2 &\leq \sum_{k=0}^{\infty}(\|w^k - w^*\|_H^2 - \|w^{k+1} - w^*\|_H^2) \\
&\leq \|w^0 - w^*\|_H^2,
\end{aligned}$$

and thus

$$\lim_{k\to\infty} \|w^k - \tilde{w}^k\|_S^2 = 0. \tag{4.2}$$

This indicates the sequence $\{\tilde{w}^k\}$ is also bounded. Let w^∞ be a cluster point of $\{\tilde{w}^k\}$, where $\{\tilde{w}^{k_j}\}$ is a subsequence which converges to w^∞. Then, it follows from (2.6) that

$$\theta(x) - \theta(\tilde{x}^{k_j}) + (w - \tilde{w}^{k_j})^T F(\tilde{w}^{k_j}) \geq (w - \tilde{w}^{k_j})^T Q(w^{k_j} - \tilde{w}^{k_j}), \quad \forall\, w \in \Omega.$$

Note the matrix Q is nonsingular. It follows from the continuity of $\theta(x)$ and $F(w)$ that

$$w^\infty \in \Omega, \quad \theta(x) - \theta(x^\infty) + (w - w^\infty)^T F(w^\infty) \geq 0, \quad \forall\, w \in \Omega,$$

which means w^∞ is a solution of the VI(Ω, F, θ) (2.4). On the one hand, according to (4.2), we have $\lim_{k\to\infty} w^{k_j} = w^\infty$. On the other hand, it follows from (4.1) that

$$\|w^{k+1} - w^\infty\|_H^2 \leq \|w^k - w^\infty\|_H^2, \tag{4.3}$$

which means it is impossible that the sequence $\{w^k\}$ has more than one cluster point. Therefore, we have $\lim_{k\to\infty} w^k = w^\infty$ and the proof is complete. □

Remark 1 Refer to the same techniques in, e.g., [19, 23], we can easily show a worst-case $\mathcal{O}(1/N)$ convergence rate measured by iteration complexity for the novel method (1.7)–(1.8) by using the essential Lemma 2. Since the techniques are completely similar, the analysis of the convergence rate is omitted here.

5 Numerical experiments

We report the numerical results of the proposed algorithm (1.7)–(1.8) (practical scheme is indeed (3.7)) in this section. All codes are written in a Python 3.9 and implemented in a laptop with Intel Core CPU 2.20 GHz (i7-8750H) and 16 GB memory. The preliminary numerical results affirmatively demonstrate that the proposed method can perform more efficiently than some known parallel spitting algorithms.

5.1 The test model

The latent variable Gaussian graphical model selection (LVGGMS) problem in [6] is a typical three-block separable convex minimization model arising in statistical learning. As stated in [1], the mathematical form of the LVGGMS model reads as

$$\begin{aligned} \min \;\; & F(X, Y, Z) := \langle X, C\rangle - \log\det(X) + \nu\|Y\|_1 + \mu\,\mathrm{tr}(Z) \\ \text{s.t.} \;\; & X - Y + Z = 0, \;\; Z \succeq 0, \end{aligned} \tag{5.1}$$

where $C \in \Re^{n\times n}$ is the covariance matrix obtained from the observation, ν and μ stand for two given positive weight parameters, $\|Y\|_1 = \sum_{i,j} |Y_{i,j}|$ and $\mathrm{tr}(\cdot)$ represents the trace of a matrix.

We first show the implementation of the proposed scheme (3.7) for tackling the LVGGMS model (5.1). Since the update (3.6) is simple, we only focus on the detail of the resulting x_i-subproblems ($i = 1, 2, 3$). Let

$$\begin{aligned}\mathcal{L}_\beta(X,Y,Z,\Lambda) =& \langle X, C\rangle - \log\det(X) + \nu\|Y\|_1 + \mu\mathrm{tr}(Z)\\ &- \langle\Lambda, X - Y + Z\rangle + \frac{\beta}{2}\|X - Y + Z\|_F^2\end{aligned}$$

be the augmented Lagrangian function of (5.1). The corresponding scheme of the x-subproblem in (3.7) then reads as

$$\left\{\begin{array}{rcl}\tilde{X}^k &=& \arg\min\left\{\mathcal{L}_\beta(X, Y^k, Z^k, \Lambda^k) + \dfrac{\tau\beta}{2}\|X - X^k\|_F^2 \;\middle|\; X \in \Re^{n\times n}\right\},\\ \tilde{Y}^k &=& \arg\min\left\{\mathcal{L}_\beta(X^k, Y, Z^k, \Lambda^k) + \dfrac{\tau\beta}{2}\|Y - Y^k\|_F^2 \;\middle|\; Y \in \Re^{n\times n}\right\},\\ \tilde{Z}^k &=& \arg\min\left\{\mathcal{L}_\beta(X^k, Y^k, Z, \Lambda^k) + \dfrac{\tau\beta}{2}\|Z - Z^k\|_2^2 \;\middle|\; Z \succeq 0,\ Z \in \Re^{n\times n}\right\}.\end{array}\right. \tag{5.2}$$

For the X-subproblem in (5.2), according to the first-order optimal condition, it is equivalent to solving the nonlinear equation system

$$C - X^{-1} + \beta\left(X - Y^k + Z^k - \frac{1}{\beta}\Lambda^k\right) + \tau\beta(X - X^k) = 0.$$

Multiplying X to both sides, it follows that

$$(1+\tau)\beta X^2 + (C - \tau\beta X^k - \beta Y^k + \beta Z^k - \Lambda^k)X - I = 0. \tag{5.3}$$

Let $UDU^T = C - \tau\beta X^k - \beta Y^k + \beta Z^k - \Lambda^k$ be an eigenvalue decomposition. Plugging it back into (5.3) and setting $P = U^T XU$, we have

$$(1+\tau)\beta PP + DP - I = 0,$$

and thus obtain

$$P_{ii} = \frac{1}{2\beta(1+\tau)}\left(-D_{ii} + \sqrt{D_{ii}^2 + 4(1+\tau)\beta}\right).$$

Therefore, we can derive that $\tilde{X}^k = U\,\mathrm{diag}(P)\,U^T$ is a solution of (5.3). For the Y-subproblem in (5.2), since

$$\begin{aligned}\tilde{Y}^k &= \arg\min_Y\left\{\nu\|Y\|_1 + \frac{\beta}{2}\|X^k - Y + Z^k - \frac{1}{\beta}\Lambda^k\|_F^2 + \frac{\tau}{2}\beta\|Y - Y^k\|_F^2\right\}\\ &= \arg\min_Y\left\{\|Y\|_1 + \frac{\beta(1+\tau)}{2\nu}\left\|Y - \frac{1}{1+\tau}(X^k + \tau Y^k + Z^k - \frac{1}{\beta}\Lambda^k)\right\|_F^2\right\},\end{aligned}$$

the Y-subproblem can be solved via the soft shrinkage operator (see [32]) immediately. Finally, for the Z-subproblem in (5.2), note that

$$\tilde{Z}^k = \arg\min_{Z\succeq 0}\left\{\mu\mathrm{tr}(Z) + \frac{\beta}{2}\left\|X^k - Y^k + Z - \frac{1}{\beta}\Lambda^k\right\|_F^2 + \frac{\tau}{2}\beta\|Z - Z^k\|_2^2\right\}$$
$$= \arg\min_{Z\succeq 0}\left\{\left\|Z - \frac{1}{1+\tau}\left(-X^k + Y^k + \tau Z^k + \frac{1}{\beta}(\Lambda^k - \mu I)\right)\right\|_2^2\right\}.$$

Let $VD_1V^T = \frac{1}{1+\tau}(-X^k + Y^k + \tau Z^k + \frac{1}{\beta}\Lambda^k - \frac{\mu}{\beta}I)$ be an eigenvalue decomposition. Then, we can easily verify that $\tilde{Z}^k = V\max(D_1, 0)\,V^T$ is a solution of the Z-subproblem (where $\max(D_1, 0)$ is taken component-wisely).

5.2 Numerical results

We evaluate the numerical performance of the proposed method (3.7) in this subsection. Some parallel splitting algorithms including the parallel augmented Lagrangian method (denoted by P-ALM) proposed in [14] and the parallel splitting ADMM-like method (denoted by PS-ADMM) proposed in [18] are compared. In addition, as a reference, the numerical results of the direct extension of ADMM (denoted by D-ADMM) is also listed.

In our experiments, we take $\nu = 0.005$ and $\mu = 0.05$ (refer to [1]), and the covariance matrix C is randomly generated according to S. Boyd's Homepage (see http://web.stanford.edu/~boyd/papers/admm/covsel/covsel_example.html). Moreover, refer to [1], the following two stopping conditions are taken:

$$\begin{aligned}\mathrm{IER}(k) &:= \max\left\{\|X^k - X^{k+1}\|_\infty, \|Y^k - Y^{k+1}\|_\infty, \|Z^k - Z^{k+1}\|_\infty\right\} < \mathrm{Tol},\\ \mathrm{CER}(k) &:= \|X^k - Y^k + Z^k\|_F < \mathrm{Tol}.\end{aligned}$$

Clearly, the feasibility errors IER and CER can be regarded as the primal and dual errors of the mentioned algorithms, respectively. When the proposed method (3.7) is applied for solving (5.1), it is convergent provided $\tau > (3-4)/4$, and we consider three special cases: $\tau = 0$, $\tau = 1/3$ and $\tau = 1$. Moreover, the initial values are chosen as $(X_0, Y_0, Z_0, \Lambda_0) = (I_n, 2I_n, I_n, \mathbf{0}_{n\times n})$ in our experiments. Meanwhile, taking the high communication cost between different CPU platforms into consideration, what we emphasize here is that the tested algorithms are implemented by a serial way, rather than a parallel way. In the following tables and figures, the numerical performance of the tested methods is evaluated by the following parameters:

- Iter: the required iteration number;
- Time: the total computing time in seconds;
- CPU: the maximum total running time of subproblems in seconds.

Note that the D-ADMM enjoys a Gaussian decomposition structure. The CPU time is indeed the total computing time for the D-ADMM.

In Table 1, we report the numerical results of the proposed method (3.7) for solving the LVGGMS model (5.1) with $n = 100$ when various stop stopping conditions are adopted. As can be seen easily, both the feasibility errors IER and CER have a

Table 1 Numerical results of the novel method (3.7) for solving the LVGGMS model (5.1) with $n = 100$

τ	β	IER$(k) < 10^{-9}$				CER$(k) < 10^{-9}$			
		Iter	CPU	Time	CER	Iter	CPU	Time	IER
$\tau = 0$	0.05	344	3.90	9.77	7.57e−8	487	5.55	13.91	1.29e−11
	0.08	232	2.61	6.58	4.54e−8	311	3.52	8.81	2.08e−11
	0.09	210	2.34	5.86	4.15e−8	279	3.13	7.92	2.29e−11
	0.10	193	2.14	5.44	3.56e−8	252	2.82	7.14	2.63e−11
	0.11	178	1.98	5.01	3.18e−8	230	2.58	6.49	2.87e−11
	0.12	165	1.84	4.63	2.93e−8	211	2.35	5.94	3.23e−11
	0.13	154	1.72	4.32	2.65e−8	195	2.18	5.55	3.52e−11
	0.14	144	1.60	4.00	2.50e−8	182	2.04	5.12	3.54e−11
	0.15	135	1.50	3.77	2.40e−8	169	1.87	4.71	4.09e−11
	0.16	128	1.41	3.58	2.09e−8	159	1.75	4.41	4.05e−11
	0.17	121	1.34	3.38	2.03e−8	149	1.65	4.19	4.61e−11
	0.18	117	1.29	3.25	1.56e−8	141	1.56	3.88	4.74e−11
	0.19	113	1.24	3.12	1.29e−8	134	1.47	3.75	6.13e−11
	0.20	109	1.20	3.04	1.21e−8	127	1.40	3.54	1.19e−10
	0.25	150	1.63	4.18	3.46e−9	162	1.77	4.43	2.61e−10
$\tau = \frac{1}{3}$	0.05	206	2.31	5.84	1.43e−8	257	2.89	7.38	6.62e−11
	0.08	140	1.56	3.94	8.08e−9	165	1.85	4.65	1.09e−10
	0.09	126	1.39	3.52	7.47e−9	147	1.62	4.16	1.26e−10
	0.10	115	1.27	3.18	6.56e−9	133	1.48	3.72	1.34e−10
	0.11	106	1.18	3.07	5.87e−9	121	1.35	3.41	1.52e−10
	0.12	98	1.07	2.73	5.38e−9	111	1.21	3.06	1.64e−10
	0.13	**92**	1.01	2.55	4.72e−9	103	1.13	2.85	1.86e−10
	0.14	97	1.06	2.68	1.92e−9	**101**	1.11	2.82	4.89e−10
	0.15	114	1.25	3.13	9.45e−10	114	1.24	3.17	9.64e−10
	0.16	128	1.40	3.59	7.27e−10	126	1.39	3.49	1.15e−9
	0.17	140	1.54	3.91	6.60e−10	137	1.52	3.81	1.29e−9
	0.18	151	1.66	4.20	6.33e−10	147	1.61	4.05	1.47e−9
	0.19	162	1.80	4.54	5.85e−10	157	1.73	4.35	1.58e−9
	0.20	173	1.90	4.84	5.29e−10	167	1.84	4.69	1.62e−9
	0.25	223	2.47	6.25	4.17e−10	211	2.31	5.90	2.26e−9
$\tau = 1$	0.05	156	1.74	4.38	1.66e−8	193	2.14	5.43	5.56e−11
	0.08	103	1.13	2.87	1.05e−8	122	1.35	3.40	7.98e−11
	0.09	94	1.02	2.57	8.48e−9	108	1.20	3.06	1.06e−10
	0.10	111	1.22	3.06	2.20e−9	117	1.29	3.25	3.95e−10
	0.11	132	1.47	3.72	1.53e−9	136	1.52	3.83	5.56e−10
	0.12	150	1.64	4.12	1.28e−9	153	1.68	4.25	6.51e−10
	0.13	166	1.83	4.68	1.21e−9	168	1.85	4.71	7.95e−10
	0.14	182	2.01	5.08	1.08e−9	183	2.01	5.12	8.74e−10
	0.15	197	2.18	5.57	1.02e−9	198	2.17	5.56	9.09e−10
	0.16	212	2.32	5.94	9.45e−9	212	2.33	5.95	9.84e−10
	0.17	227	2.50	6.37	8.62e−9	225	2.47	6.33	1.11e−9
	0.18	241	2.68	6.78	8.33e−9	239	2.65	6.68	1.13e−9

Table 1 (continued)

τ	β	IER(k) $< 10^{-9}$				CER(k) $< 10^{-9}$			
		Iter	CPU	Time	CER	Iter	CPU	Time	IER
	0.19	256	2.83	7.25	7.49e−9	252	2.76	6.98	1.21e−9
	0.20	270	2.97	7.56	7.14e−9	265	2.82	7.24	1.28e−9
	0.25	338	3.75	9.52	5.85e−9	327	3.64	9.38	1.64e−9

Bold value denote the required minimum number of iterations under various stopping criteria

Table 2 Toned values of parameters of the above mentioned operator spiting methods for the LVGGMS model (5.1)

Method	Other parameters	β	
		IER	CER
New method	Regularization factor $\tau = 1/3$	$\beta = 0.13$	$\beta = 0.14$
D-ADMM	–	$\beta = 0.15$	$\beta = 0.15$
GS-ADMM	Back-substitution factor $\nu = 0.95$	$\beta = 0.15$	$\beta = 0.15$
PS-ADMM	Regularization factor $\tau = 1.001$	$\beta = 0.07$	$\beta = 0.08$
P-ALM	Relaxation factor $\gamma = 1.95$	$\beta = 0.10$	$\beta = 0.11$

similar changing pattern with the change of β for the various regularization parameters τ. In particular, the cases where $\tau = 1/3$ and $\beta \in (0, 12, 0.14)$ perform more stably and efficiently, and they need only a reduced iterations and computational time (both Time and CPU) in terms of the feasibility errors IER and CER.

To show the comparisons between the novel method (3.7) and other operator spitting methods, we list first some efficient parameters for the above mentioned splitting algorithms. Using the same parameter adjustment strategy (trial-and-error) as the novel method (3.7), the toned parameters of the mentioned methods for solving the LVGGMS problem are given in Table 2.

In Table 3, we report the numerical results of the above tested algorithms for the LVGGMS problem (5.1) with various settings of n. As shown in Table 3, it is clear that the D-ADMM needs less required iterations and the proposed method (3.7) needs less CPU time than other algorithms. Compared with P-ALM and PS-ADMM, the novel method needs also fewer required iterations to reach different stopping criteria, which verifies empirically our theoretical motivation of this study (our purpose is to present a more efficient parallel spitting method). In addition, it can be seen easily from Table 3 that the D-ALM is divergent for the LVGGMS model (5.1), which can be further numerically illustrated that the direct extension of ALM is not necessarily convergent.

To further visualize the numerical comparison between the new method and the other splitting algorithms, in Fig. 2, we plot their respective feasibility errors with respect to iterations and CPU time for the case $n = 100$. Computational results in Fig. 2 demonstrate that the proposed method has steeper convergence curves than PS-ADMM and P-ALM both in iterations and CPU time. In particular, it can bee seen from Fig. 2 that the new method is faster than D-ADMM, even though the latter needs a less required iterations. For other settings of n, since their convergence curves are similar as the case $n = 100$, we opt to skip them for succinctness.

Table 3 Numerical results of the tested algorithms for solving the LVGGMS problem (5.1) with various n

n	New method		D-ADMM		PS-ADMM		P-ALM		D-ALM
	Iter	CPU	Iter	CPU	Iter	CPU	Iter	CPU	
IER $< 10^{-9}$									
100	92	1.01	67	1.81	121	1.35	323	3.62	div
300	228	27.19	141	34.12	278	34.10	689	83.11	div
400	165	36.62	104	45.77	205	47.81	520	114.88	div
500	107	33.88	75	47.53	120	38.83	382	136.32	div
CER $< 10^{-9}$									
100	101	1.11	79	2.22	143	1.58	436	4.91	div
300	241	28.32	179	43.33	299	36.07	933	117.25	div
400	176	38.74	132	58.19	240	54.20	693	158.18	div
500	110	34.45	81	51.55	139	44.38	475	164.59	div

Here, "div" is used to denote the tested method is numerically divergent

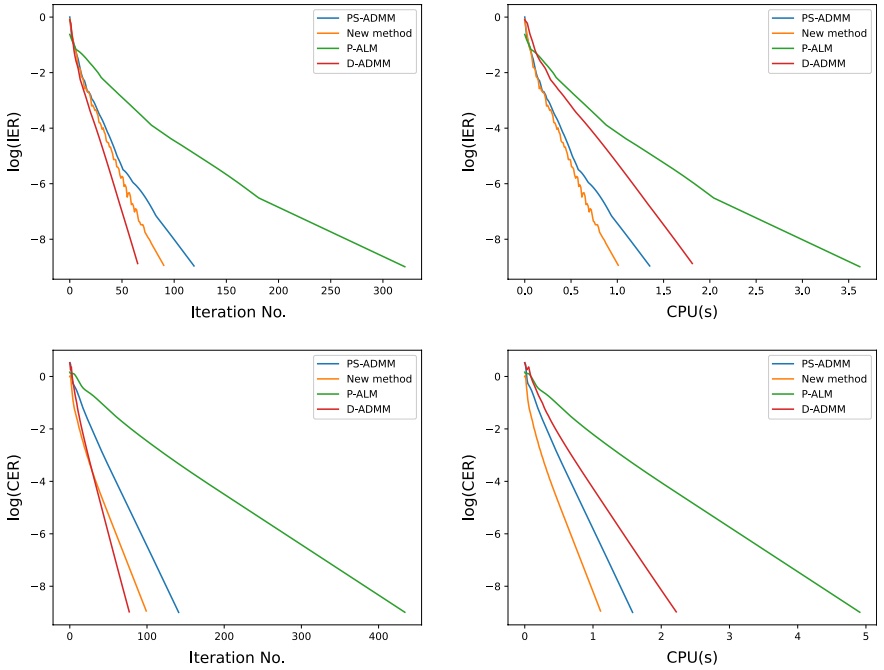

Fig. 2 First column: convergence curves of the IER and CER with iteration numbers; second column: convergence curves of the IER and CER with CPU time ($n = 100$)

6 Conclusions

We propose a novel parallel splitting ALM-based algorithm for solving the multiple-block separable convex programming problem with linear equality constraints, where the objective function can be expressed as the sum of some individual subfunctions without coupled variables. By adding a simple update (1.8) to the fully parallel regularized ALM (1.7), convergence of the novel method can be guaranteed provided a reduced parameter of the regularization terms, which allows bigger step sizes and thus potentially results in fewer required iterations in the real computations. The numerical results on the LVGGMS problem affirmatively illustrates that the proposed method has an acceleration effect compared with some known parallel splitting algorithms.

Acknowledgements Funding was provided by National Natural Science Foundation of China (Grant No.11871029).

References

1. Bai, J., Li, J., Xu, F., Zhang, H.: Generalized symmetric ADMM for separable convex optimization. Comput. Optim. Appl. **70**(1), 129–170 (2018). https://doi.org/10.1007/s10589-017-9971-0
2. Beck, A.: First-Order Methods in Optimization, vol. 25. SIAM, Philadelphia (2017)
3. Blum, E., Oettli, W.: Mathematische Optimierung. Grundlagen und Verfahren. Ökonometrie und Unternehmensforschung. Springer, Berlin (1975)
4. Bose, N., Boo, K.: High-resolution image reconstruction with multisensors. Int. J. Imaging Syst. Technol. **9**(4), 294–304 (1998) https://doi.org/10.1002/(SICI)1098-1098(1998)9:4%3c294::AID-IMA11%3e3.0.CO;2-X
5. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2010). https://doi.org/10.1561/2200000016
6. Chandrasekaran, V., Parrilo, P.A., Willsky, A.S.: Latent variable graphical model selection via convex optimization. Ann. Stat. **40**(4), 1935–1967 (2012). https://doi.org/10.1214/11-AOS949
7. Chen, C., He, B., Ye, Y., Yuan, X.: The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. Math. Program. **155**(1–2), 57–79 (2016). https://doi.org/10.1007/s10107-014-0826-5
8. Facchinei, F., Pang, J.S.: Finite-Dimensional Variational Inequalities and Complementarity Problems. Springer Series in Operations Research, vol. I. Springer, New York (2003)
9. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximation. Comput. Math. Appl. **2**(1), 17–40 (1976). https://doi.org/10.1016/0898-1221(76)90003-1
10. Glowinski, R.: Numerical Methods for Nonlinear Variational Problems. Springer, Berlin (1984)
11. Hager, W.W., Zhang, H.: Convergence rates for an inexact ADMM applied to separable convex optimization. Comput. Optim. Appl. **77**(3), 729–754 (2020). https://doi.org/10.1007/s10589-017-9971-0
12. He, B.: My 20 years research on alternating directions method of multipliers. Oper. Res. Trans. **22**, 1–31 (2018)
13. He, B.: Study on the splitting methods for separable convex optimization in a unified algorithmic framework. Anal. Theory Appl. **36**, 262–282 (2020). https://doi.org/10.4208/ata.OA-SU13
14. He, B., Hou, L., Yuan, X.: On full Jacobian decomposition of the augmented Lagrangian method for separable convex programming. SIAM J. Optim. **25**(4), 2274–2312 (2015). https://doi.org/10.1137/130922793

15. He, B., Ma, F., Yuan, X.: Optimal proximal augmented Lagrangian method and its application to full Jacobian splitting for multi-block separable convex minimization problems. IMA J. Numer. Anal. **40**(2), 1188–1216 (2020). https://doi.org/10.1093/imanum/dry092
16. He, B., Ma, F., Yuan, X.: Optimally linearizing the alternating direction method of multipliers for convex programming. Comput. Optim. Appl. **75**(2), 361–388 (2020). https://doi.org/10.1007/s10589-019-00152-3
17. He, B., Tao, M., Yuan, X.: Alternating direction method with Gaussian back substitution for separable convex programming. SIAM J. Optim. **22**(2), 313–340 (2012). https://doi.org/10.1137/110822347
18. He, B., Tao, M., Yuan, X.: A splitting method for separable convex programming. IMA J. Numer. Anal. **35**(1), 394–426 (2015). https://doi.org/10.1093/imanum/drt060
19. He, B., Tao, M., Yuan, X.: Convergence rate analysis for the alternating direction method of multipliers with a substitution procedure for separable convex programming. Math. Oper. Res. **42**(3), 662–691 (2017). https://doi.org/10.1287/moor.2016.0822
20. He, B., Xu, S., Yuan, J.: Indefinite linearized augmented Lagrangian method for convex optimization with linear inequality constraints. arXiv preprint arXiv:2105.02425 (2021)
21. He, B., Yuan, X.: Convergence analysis of primal-dual algorithms for a saddle-point problem: from contraction perspective. SIAM J. Imaging Sci. **5**(1), 119–149 (2012). https://doi.org/10.1137/100814494
22. He, B., Yuan, X.: On the $O(1/n)$ convergence rate of the Douglas–Rachford alternating direction method. SIAM J. Numer. Anal. **50**(2), 700–709 (2012). https://doi.org/10.1137/110836936
23. He, B., Yuan, X.: A class of ADMM-based algorithms for three-block separable convex programming. Comput. Optim. Appl. **70**(3), 791–826 (2018). https://doi.org/10.1007/s10589-018-9994-1
24. Hestenes, M.R.: Multiplier and gradient methods. J. Optim. Theory Appl. **4**(5), 303–320 (1969). https://doi.org/10.1007/BF00927673
25. Kiwiel, K.C., Rosa, C.H., Ruszczynski, A.: Proximal decomposition via alternating linearization. SIAM J. Optim. **9**(3), 668–689 (1999). https://doi.org/10.1137/S1052623495288064
26. Martinet, B.: Régularisation d'inéquations variationnelles par approximations successives. Rev. Fr. Inform. Rech. Oper. **4**(R3), 154–158 (1970)
27. Parikh, N., Boyd, S.: Proximal algorithms. Found. Trends Optim. **1**(3), 127–239 (2014). https://doi.org/10.1561/2400000003
28. Peng, Y., Ganesh, A., Wright, J., Xu, W., Ma, Y.: RASL: robust alignment by sparse and low-rank decomposition for linearly correlated images. IEEE Trans. Pattern Anal. Mach. Intell. **34**(11), 2233–2246 (2012)
29. Powell, M.J.: A Method for Nonlinear Constraints in Minimization Problems. In: Fletcher, R. (ed.) Optimization, pp. 283–298. Academic Press, New York (1969)
30. Rockafellar, R.T.: Monotone operators and the proximal point algorithm. SIAM J. Control Optim. **14**(5), 877–898 (1976). https://doi.org/10.1137/0314056
31. Setzer, S., Steidl, G., Teuber, T.: Deblurring Poissonian images by split Bregman techniques. J. Visual Commun. Image Represent. **21**(3), 193–199 (2010). https://doi.org/10.1016/j.jvcir.2009.10.006
32. Tao, M., Yuan, X.: Recovering low-rank and sparse components of matrices from incomplete and noisy observations. SIAM J. Optim. **21**(1), 57–81 (2011). https://doi.org/10.1137/100781894
33. Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., Knight, K.: Sparsity and smoothness via the fused lasso. J. R. Stat. Soc. **67**(1), 91–108 (2005). https://doi.org/10.1111/j.1467-9868.2005.00490.x
34. Yuan, J., Bae, E., Tai, X.C.: A study on continuous max-flow and min-cut approaches. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2217–2224. IEEE (2010)
35. Yuan, J., Bae, E., Tai, X.C., Boykov, Y.: A continuous max-flow approach to Potts model. In: Computer Vision-ECCV 2010, pp. 379–392. Springer (2010)
36. Yuan, J., Fenster, A.: Modern convex optimization to medical image analysis. arXiv preprint arXiv:1809.08734 (2018)