SCHUR COMPLEMENT BASED MULTICOLORING LOW-RANK CORRECTION PRECONDITIONER FOR SPARSE MATRICES *

QINGQING ZHENG †, YUANZHE XI ‡, YOUSEF SAAD §, AND LINZHANG LU ¶

Abstract. This paper proposes an efficient Schur complement based parallel preconditioner for solving general large sparse linear systems which include the nonsymmetric and highly indefinite problems that are difficult to solve by using iterative methods. The basic idea is that the original problem is solved after it is transfered to several smaller systems based on the multicoloring ordering and the Schur complement combined with low-rank correction. Each smaller system corresponds to one color and one sub-solution vector. The low-rank correction matrix is obtained by several steps of Lanczos process and Arnoldi process for the symmetric and nonsymmetric matrices, respectively. Numerical examples illustrate that, when combined with Krylov subspace accelerators, the new preconditioner is efficient to solve large sparse symmetric and nonsymmetric linear systems.

Key words. Low-rank approximation; Schur complement; multicoloring; parallel preconditioner; Krylov subspace method

1. Introduction.

Consider the solution of the following linear system

$$A_0 x_0 = b_0, \tag{1.1}$$

where $A_0 \in \mathbb{R}^{n\times n}$ and $b_0 \in \mathbb{R}^n$. Iterative methods based on the Krylov subspace methods can be quite efficient for solving the above linear system especially when a good preconditioner is provided. A preconditioner to matrix A_0 is just a way to approximately solve $A_0 z = r$, where r is the residual vector. This way can be a matrix M which is a good approximation of A_0 or an effective function F, which satisfies $z = M^{-1}r$ or $z = F(r)$ is a good approximation to the solution of $A_0 z = r$. A basic property of the preconditioner is that the system with coefficient being the matrix M or function F should be inexpensive to solve.

ILU preconditioners [10, 15] based on the Incomplete LU (ILU) factorization of A_0 can be quite effective for solving certain kinds of linear systems. However, the preconditioners can not handle problems with coefficient matrix being highly indefinite. In addition, the ILU preconditioners have very poor performance on high-performance computers equipped with massively parallel coprocessors due to their sequential nature. The algebraic multigrid (AMG) is a popular technique to solve the problems coming from the discretized PDEs, which performs well for a large class of SPD matrices and also works efficiently in parallel. But multigrid method without specialization will fail on indefinite problems. Sparse approximate inverses [1, 8, 3] can overcome these situations. However, these preconditioners may result in high cost to the construction and application.

In Recent years, a new class of approximate inverses preconditioners rely on low-rank approximations were presented. For example, the divide-conquer based: Multilevel Low-Rank (MLR) preconditioner presented in [13]. The basic idea of this preconditioner is that a low-rank approximation is recursively applied after the problem is divided in two smaller parts. In addition, combining domain decomposition

*This work was supported by NSF under grant NSF/DMS-1521573 and by the Minnesota Supercomputing Institute

†School of Mathematical Science, Xiamen University. {zhengxmu@gmail.com}

‡Department of Mathematics, Emory University. {yxi26@emory.edu}

§Computer Science & Engineering, University of Minnesota, Twin Cities. {saad@umn.edu}

¶School of Mathematical Science, Guizhou Normal University, China & School of Mathematical Science, Xiamen University, China {lzlu@xmu.edu.cn }

with greedy multicoloring algorithm, the Multicoloring based Low-Rank (MCLR) preconditioner [18] was presented which can be used to solve the symmetric and nonsymmetric matrices. Another class of preconditioners are the Schur complement based: the Schur complement low-rank (SLR) preconditioner [14], the Multilevel Schur complement Low-Rank (MSLR) preconditioner [17] based on a multilevel Hierarchical interface decomposition (HID) ordering [17, 5, 9], the Generalized Multilevel Schur complement Low-Rank (GMSLR) preconditioner [5] presented to generalize MSLR preconditioner to solve nonsymmetric systems. In this paper, we present an efficient Schur complement based preconditioner by combining the multicoloring reordering and low-rank correction. Approximate inverses method is considered to apply to the left-top submatrices of the reordered matrix corresponding to each color. The resulted preconditioner is called Schur complement based Multicoloring Low-Rank (SMLR) correction preconditioner. Low-rank corrections are used for the submatrices in the left-top part of the reordered matrices for each color.

The SMLR method can solve symmetric and nonsymmetric large sparse linear systems. However, the previous methods such as SLR, MLR and MSLR can only be used for solving symmetric problems. In contrast with the GMSLR and MCLR technics which can also handle the nonsymmetric matrices have only one low-rank approximation scheme, SMLR method has two different schemes for the symmetric system and nonsymmetric system, respectively. The scheme can be chosen depend on the problem we need to solve. Similar to the MCLR method, we only need a very small rank to reach the same approximation accuracy for the SMLR preconditioner especially for the symmetric positive definite matrices. The cost for the preconditioner construction will be reduced dramatically due to this good property, since the cost to computing and applying the low-rank correction terms can be pretty high. The rank can just be taken as a value smaller than 5, and this is verified via numerical results presented in Section 5. To obtain each sub-solution according to the color, we only need to solve two linear systems with coefficient being block diagonal matrices. An efficient Block-Jacobi type correction technique presented in [18] is used to correct the solution to boost the convergence of the SMLR preconditioner. Here, the application of the Block-Jacobi type correction to the SMLR preconditioner is different with the one used in the MCLR preconditioner where the Block-Jacobi correction technique is applied to construct the MCLR preconditioner. In addition, a sufficient condition to converge for this corrected technique is presented in Section 4, which has not been discussed in the MCLR preconditioner [18].

The rest part of this paper is organized as follows: In Section 2, we present the SMLR preconditioner by transfering the multicoloring reordered system to an equivalent block diagonal linear system. This system is establish to put the solutions corresponding to different colors together, which are approximately obtained from some smaller linear systems with coefficient matrices being Schur complements. Section 3 shows some properties of the SMLR method. Section 4 studies the improvement of the SMLR preconditioner and presents the convergence property of the Block-Jacobi type correction which was not proved in [18]. In Section 5, we propose some numerical experiments to illustrate the efficiency of the new method. Section 6 gives some concluding remarks and presents some future works.

2. Schur complement based Multicoloring Low-Rank Preconditioner. This section presents the construction and application of SMLR preconditioner. We begin by briefly introducing the domain multicoloring ordering [18] which was presented to keep the balance between performance of convergence and parallel efficiency

of the MCLR preconditioner: first some domain decomposition methods are used to partition the graph, then the standard greedy multicoloring algorithm is applied to color the partition. This domain multicoloring is different from the classical multicoloring in which the standard greedy algorithm is applied to the adjacency graph of subdomains rather than vertices. The reordered matrix obtained by applying domain multicoloring ordering has block structure and the submatrices in the diagonal part of the reordered matrix are block diagonal matrices. Furthermore, if we have c diagonal blocks (here $c > 0$ is the number of colors), then each diagonal block is a block diagonal matrix. Figure 2.1 is an illustration of this multicoloring ordering for a general matrix. For this case, the number of colors is $c = 4$ and there are 3 subdomains in each color.

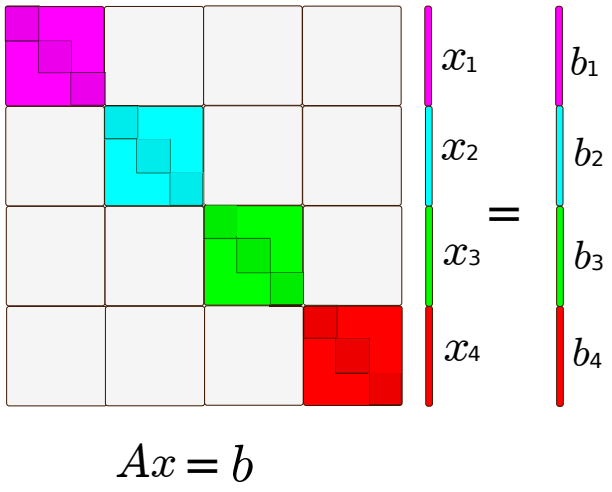

FIG. 2.1. *A four colors reordering with* 12 *subdomains for a general matrix by using domain multicoloring ordering. Here, A is the reordered matrix and there are* 3 *submatrices in each color.*

Equation (1.1) can be rewritten as the following permuted system

$$PA_0P^TPx_0 = Pb_0, \tag{2.1}$$

where the matrix P is the permutation matrix obtained by applying the domain multicoloring algorithm introduced above. Denote $PA_0P^T = A$, $Px_0 = x$ and $Pb_0 = b$, then linear system (2.1) reduces to

$$Ax = b, \tag{2.2}$$

where

$$x = (x_1^T, x_2^T, \ldots, x_c^T)^T, \tag{2.3}$$

$$b = (b_1^T, b_2^T, \ldots, b_c^T)^T$$

and

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & \ldots & A_{1,c} \\ A_{2,1} & A_{2,2} & \ldots & A_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ A_{c,1} & A_{c,2} & \ldots & A_{c,c} \end{pmatrix}.$$

As mentioned above, here $A_{1,1}, A_{2,2}, \ldots, A_{c,c}$ are block diagonal matrices and c is the number of colors.

2.1. Equivalent solution. In this section, we study the solution of (2.2) from the view of an equivalent solution. Firstly, we transfer the original system to c (the number of color) smaller equivalent systems according to the colors, and then solve each sub-solution x_i $(i = 1, 2, \ldots, c)$ base on a "local" Schur complement. Here, x_i $(i = 1, 2, \ldots, c)$ are column vectors definite in (2.3).

For $i = 1, 2, \ldots, c$, the system $Ax = b$ can be rewritten as the following block two-by-two linear systems:

$$\begin{pmatrix} B_i & F_i \\ E_i & C_i \end{pmatrix} \begin{pmatrix} x_i^o \\ x_i \end{pmatrix} = \begin{pmatrix} b_i^o \\ b_i \end{pmatrix}, \tag{2.4}$$

where

$$B_i = \begin{pmatrix} A_{1,1} & \cdots & A_{1,i-1} & A_{1,i+1} & \cdots & A_{1,c} \\ \vdots & & \vdots & \vdots & & \vdots \\ A_{i-1,1} & \cdots & A_{i-1,i-1} & A_{i-1,i+1} & \cdots & A_{i-1,c} \\ A_{i+1,1} & \cdots & A_{i+1,i-1} & A_{i+1,i+1} & \cdots & A_{i+1,c} \\ \vdots & & \vdots & \vdots & & \vdots \\ A_{c,1} & \cdots & A_{c,i-1} & A_{c,i+1} & \cdots & A_{c,c} \end{pmatrix},$$

$$E_i = \begin{pmatrix} A_{i,1} & \ldots & A_{i,i-1} & A_{i,i+1} & \ldots & A_{i,c} \end{pmatrix}, \; C_i = A_{i,i},$$

and

$$F_i = \begin{pmatrix} A_{1,i} \\ \vdots \\ A_{i-1,i} \\ A_{i+1,i} \\ \vdots \\ A_{c,i} \end{pmatrix}, \; x_i^o = \begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_{i+1} \\ \vdots \\ x_c \end{pmatrix}, \; b_i^o = \begin{pmatrix} b_1 \\ \vdots \\ b_{i-1} \\ b_{i+1} \\ \vdots \\ b_c \end{pmatrix}.$$

In fact, the above matrix B_i is just the submatrix of A obtained by deleting the elements in ith row and ith column. Moreover, the coefficient matrix of linear system (2.4) is obtained by moving the elements in ith row to the last row and then putting the elements in the ith column to the last column. Since

$$\begin{pmatrix} B_i & F_i \\ E_i & C_i \end{pmatrix} = \begin{pmatrix} I & 0 \\ E_i B_i^{-1} & I \end{pmatrix} \begin{pmatrix} B_i & F_i \\ 0 & S_i \end{pmatrix}, \tag{2.5}$$

where $S_i = C_i - E_i B_i^{-1} F_i$ is the 'local' Schur complement of color i, we can obtain

$$\begin{pmatrix} B_i & F_i \\ 0 & S_i \end{pmatrix} \begin{pmatrix} x_i^o \\ x_i \end{pmatrix} = \begin{pmatrix} b_i^o \\ b_i - E_i B_i^{-1} b_i^o \end{pmatrix}$$

or equivalently,

$$\begin{cases} S_i x_i = b_i - E_i B_i^{-1} b_i^o, \\ B_i x_i^o = b_i^o - F_i x_i. \end{cases} \tag{2.6}$$

If we accurately solve one system in (2.5) for arbitrary i ($i = 1, 2, \ldots, c$), we can get the solution of linear system (2.1). The iteration step will be only one if this solver is used as a preconditioner. However, the cost is very expensive since we need to solve three linear systems with coefficient matrix B_i. Now we consider replacing the above procedure by partitioning the whole solution to c "local" solutions according to the colors, which can be solve at the same time based on the "local" Schur complement.

Note the first equation of (2.6), we can obtain x_i by solving a linear system with coefficient matrix being a "local" Schur complement S_i. Denote

$$f_i = b_i - E_i B_i^{-1} b_i^o \ (i = 1, 2, \ldots, c),$$

then the linear system (2.2) has the same solution with the following linear system whose coefficient matrix is block diagonal:

$$\begin{pmatrix} S_1 & 0 & \cdots & 0 \\ 0 & S_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & S_c \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_c \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_c \end{pmatrix}. \tag{2.7}$$

To illustrate the above process, we present a simple example below. EXAMPLE 2.1. Consider the case for $c = 3$, the reordered linear system with three colors has the following block structure:

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

which is equal to all the following three linear systems:

$$\begin{cases} \begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} A_{21} \\ A_{31} \end{pmatrix} x_1 = \begin{pmatrix} b_2 \\ b_3 \end{pmatrix}, \\ \begin{pmatrix} A_{12} & A_{13} \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} + A_{11} x_1 = b_1, \end{cases} \tag{2.8}$$

$$\begin{cases} \begin{pmatrix} A_{11} & A_{13} \\ A_{31} & A_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} + \begin{pmatrix} A_{12} \\ A_{32} \end{pmatrix} x_2 = \begin{pmatrix} b_1 \\ b_3 \end{pmatrix}, \\ \begin{pmatrix} A_{21} & A_{23} \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} + A_{22} x_2 = b_2, \end{cases} \tag{2.9}$$

$$\begin{cases} \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} A_{13} \\ A_{23} \end{pmatrix} x_3 = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \\ \begin{pmatrix} A_{31} & A_{32} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + A_{33} x_3 = b_3. \end{cases} \tag{2.10}$$

We can solve x_1 from the following linear system:

$$\left[A_{11} - \begin{pmatrix} A_{12} & A_{13} \end{pmatrix} \begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix}^{-1} \begin{pmatrix} A_{21} \\ A_{31} \end{pmatrix} \right] x_1 = b_1 - \begin{pmatrix} A_{12} & A_{13} \end{pmatrix} \begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix}^{-1} \begin{pmatrix} b_2 \\ b_3 \end{pmatrix},$$

whose coefficient matrix is a Schur complement:

$$S_1 = A_{11} - \begin{pmatrix} A_{12} & A_{13} \end{pmatrix} \begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix}^{-1} \begin{pmatrix} A_{21} \\ A_{31} \end{pmatrix}.$$

Similarly, we can obtain x_2 and x_3 by solving the other two linear systems with the coefficient matrix being 'local' Schur complement. These three linear systems can be solved at the same time.

Figure 2.2 shows the above procedure to get the solution for the system presented in Figure 2.1.

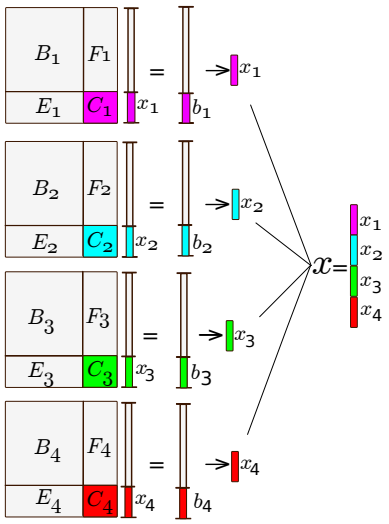

FIG. 2.2. *Solving 4 "local" solutions to get the whole solution of the reordered system with coefficient matrix given in Figure 2.1.*

For the systems

$$S_i x_i = f_i, \ (i = 1, 2, \ldots, c), \tag{2.11}$$

we just need to solve two linear systems with coefficient matrix B_i for each color. In next section, we consider some matrix $\overline{B}_i^{-1}$ to approximate B_i^{-1} to further reduce the cost. Here, the linear system with coefficient matrix $\overline{B}_i$ is more easier to solve than that with B_i.

2.2. Low-rank approximation. In this section, we consider approximating the matrix B_i by low-rank corrected methods. Here, the low-rank corrections presented have two different schemes for the symmetric matrices and nonsymmetric matrices, respectively. In the MCLR preconditioner [18], we have shown the low rank property associated with the difference between the inverse of a general matrix and the inverse of its block diagonal part if the matrix is partitioned into a block two-by-two matrix. And then the application of the MCLR method is based on the low-rank decay property of matrix

$$I - B_i B_{i0}^{-1}.$$

That is,

$$\begin{aligned} B_i^{-1} &\approx B_{i0}^{-1}(I - V_{ik}H_{ik}V_{ik}^T)^{-1} \\ &= B_{i0}^{-1}(I + V_{ik}G_{ik}V_{ik}^T), \end{aligned} \tag{2.12}$$

where V_{ik}, H_{ik} are matrices with rank k which are obtained by taking several steps of Arnoldi procedure and $G_{ik} = (I - H_{ik})^{-1} - I \in \mathbb{R}^{k\times k}$ is also a matrix with rank k. In this paper, we will just use one level to get the approximation of B_i^{-1} by applying the MCLR method [18]. That is, B_{i0} will be replaced by the block diagonal part of matrix B_i:

$$B_{i0} = \begin{pmatrix} A_{1,1} & & & & & \\ & \ddots & & & & \\ & & A_{i-1,i-1} & & & \\ & & & A_{i+1,i+1} & & \\ & & & & \ddots & \\ & & & & & A_{c,c} \end{pmatrix}.$$

However, we should note that the preconditioner obtained by using the above low-rank approximation is nonsymmetric. How can it be applied to a symmetric matrix by some symmetric forms? Note the matrix $I - L_{i0}^{-1}B_iL_{i0}^{-T}$ is similar to the matrix $I - B_iB_{i0}^{-1}$, where L_{i0} is a lower triangular matrix (Cholesky factor) obtained by the Cholesky decomposition of B_{i0}, i.e., $L_{i0}L_{i0}^T = B_{i0}$. This means $I - L_{i0}^{-1}B_iL_{i0}^{-T}$ can also be approximated by a low rank matrix. We can apply the Lanczos algorithm to obtain the low-rank approximation of this matrix

$$I - L_{i0}^{-1}B_iL_{i0}^{-T} \approx V_{ik}T_{ik}V_{ik}^T$$

with $T_{ik} \in \mathbb{R}^{k\times k}$ be a tridiagonal matrix. Then, we have

$$\begin{aligned} B_i^{-1} &\approx L_{i0}^{-T}(I - V_{ik}T_{ik}V_{ik}^T)^{-1}L_{i0}^{-1} \\ &= L_{i0}^{-T}(I + V_{ik}T_{ik}(I - T_{ik})^{-1}V_{ik}^T)L_{i0}^{-1} \\ &= L_{i0}^{-T}(I + V_{ik}[(I - T_{ik})^{-1} - I]V_{ik}^T)L_{i0}^{-1} \\ &= L_{i0}^{-T}(I + V_{ik}G_{ik}V_{ik}^T)L_{i0}^{-1}, \end{aligned} \tag{2.13}$$

where $G_{ik} = (I - T_{ik})^{-1} - I \in \mathbb{R}^{k\times k}$ is a symmetric matrix with rank k.

If B_i is symmetric positive definite, denote

$$R_{ik} = L_{i0}^{-T}V_{ik}G_{ik}V_{ik}^TL_{i0}^{-1},$$

which is a symmetric matrix. Otherwise, let

$$R_{ik} = B_{i0}^{-1}V_{ik}G_{ik}V_{ik}^T.$$

Obviously, R_{ik} is a matrix with rank k and we have

$$B_i^{-1} \approx B_{i0}^{-1} + R_{ik} = \overline{B}_i^{-1},$$

where $\overline{B}_i = (B_{i0}^{-1} + R_{ik})^{-1}$. Therefore, one can choose the low-rank approximation scheme base on the matrix need to be solved.

For the Schur complement matrix S_i $(i = 1, 2, \ldots, c)$, we use

$$\overline{S}_i = C_i - E_i \overline{B}_i^{-1} F_i$$

to approximate it. Then, for the linear systems with coefficient matrices $\overline{S}_i$ $(i = 1, 2, \ldots, c)$ we can use some preconditioned Krylov subspace methods to solve them. In this case, we just need to know the result of matrix vector product $y = \overline{S}_i v$ for arbitrary vector v. This implies that we just need to solve two linear systems with coefficient matrix $\overline{B}_i$ for each color, which can be solved in parallel by using ILU or IC method (SPD case).

2.3. Approximated solution. Based on Section 2.1 and Section 2.2, in this section, we solve the linear system (2.7) by considering an approximated solution obtained from an approximated system whose coefficient matrix also has a block diagonal structure.

Obviously, the solution of linear system (2.7) can be solved approximately from

$$\overline{S}\overline{x} = \begin{pmatrix} \overline{S}_1 & 0 & \cdots & 0 \\ 0 & \overline{S}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & \cdots & \overline{S}_c \end{pmatrix} \begin{pmatrix} \overline{x}_1 \\ \overline{x}_2 \\ \vdots \\ \overline{x}_c \end{pmatrix} = \begin{pmatrix} \overline{f}_1 \\ \overline{f}_2 \\ \vdots \\ \overline{f}_c \end{pmatrix}, \tag{2.14}$$

where $\overline{f}_i = b_i - E_i \overline{B}_i^{-1} b_i^o$. Here, the matrix $\overline{S}_i$ is the Schur complement of the block two-by-two matrix

$$\begin{pmatrix} \overline{B}_i & F_i \\ E_i & C_i \end{pmatrix}, \ i = 1, 2, \ldots, c. \tag{2.15}$$

For an arbitrary vector v, if B_i is symmetric positive definite, then $y = \overline{B}_i^{-1} v$ can be obtained by

$$\begin{aligned} y &= L_{i0}^{-T}(v_1 + V_{ik} G_{ik} V_{ik}^T v_1) \\ &= L_{i0}^{-T}(v_1 + V_{ik}(G_{ik}(V_{ik}^T v_1))) \\ &= L_{i0}^{-T} \widehat{v} \end{aligned}$$

where v_1 is computed from $v_1 = L_{i0}^{-1} v$ and $\widehat{v} = v_1 + V_{ik}(G_{ik}(V_{ik}^T v_1))$. If B_i is not symmetric positive definite, then $y = \overline{B}_i^{-1} v$ can be computed as follows

$$\begin{aligned} y &= B_{i0}^{-1}(v + V_{ik} G_{ik} V_{ik}^T v) \\ &= B_{i0}^{-1}(v + V_{ik}(G_{ik}(V_{ik}^T v))) \\ &= B_{i0}^{-1} \widehat{v}, \end{aligned}$$

where $\widehat{v} = v + V_{ik}(G_{ik}(V_{ik}^T v))$.

From (2.14), we can see that the SMLR preconditioner we have presented is a function which can approximately solve the linear system (2.2). Here, we denote this function by F, then the performance of F on a residual vector $\mathbf{r} = \begin{pmatrix} \mathbf{r}_1^T, & \mathbf{r}_2^T, & \ldots, & \mathbf{r}_c^T \end{pmatrix}^T$, i.e., $\mathbf{z} = F(\mathbf{r})$ can be obtained by solving

$$\overline{S}\mathbf{z} = \begin{pmatrix} \overline{S}_1 & 0 & \cdots & 0 \\ 0 & \overline{S}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & \cdots & \overline{S}_c \end{pmatrix} \begin{pmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_c \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 - E_1 \overline{B}_1^{-1} \mathbf{r}_1^o \\ \mathbf{r}_2 - E_2 \overline{B}_2^{-1} \mathbf{r}_2^o \\ \vdots \\ \mathbf{r}_c - E_c \overline{B}_c^{-1} \mathbf{r}_c^o \end{pmatrix},$$

where

$$\mathbf{r}_i^o = (\mathbf{r}_1^T, \ldots, \mathbf{r}_{i-1}^T, \mathbf{r}_{i+1}^T \ldots, \mathbf{r}_c^T)^T.$$

That is,

$$\mathbf{z} = \overline{S}^{-1}(\mathbf{r} - E\overline{B}^{-1}\mathbf{r}^o) = F(\mathbf{r}), \tag{2.16}$$

with

$$E = \begin{pmatrix} E_1 & & & \\ & E_2 & & \\ & & \ddots & \\ & & & E_c \end{pmatrix}, \ \overline{B} = \begin{pmatrix} \overline{B}_1 & & & \\ & \overline{B}_2 & & \\ & & \ddots & \\ & & & \overline{B}_c \end{pmatrix}, \ \mathbf{r}^o = \begin{pmatrix} \mathbf{r}_1^o \\ \mathbf{r}_2^o \\ \vdots \\ \mathbf{r}_c^o \end{pmatrix}.$$

Since the matrix E and matrix $\overline{B}$ are block diagonal, the above procedure is highly parallelizable. Algorithm 1 presents the detail description of this process.

ALGORITHM 1
Computing $\mathbf{z} = F(\mathbf{r})$

1: **For** $i = 1 : c$
2: Solve $\overline{B}_i\mathbf{v}_i = \mathbf{r}_i^o$
3: Compute $\mathbf{y}_i = \mathbf{r}_i - E_i\mathbf{v}_i$
4: Solve $\overline{S}_i\mathbf{z}_i = \mathbf{y}_i$
5: **EndFor**
6: Obtain solution $\mathbf{z} = (\mathbf{z}_1^T, \mathbf{z}_2^T, \ldots, \mathbf{z}_c^T)^T$

2.4. Spectral property. This section studies the eigenvalues property of the preconditioned matrix for the SMLR preconditioner. For matrix A, it is easy to see that the preconditioned matrix is $Y = F(A)$, so we just need to analyse the spectral property of the matrix Y. Here, for simplicity, we consider the case that $c = 2$, i.e., the number of colors after A_0 being reordered is 2. Then, we have

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}.$$

Observe matrices B_i, C_i, E_i, F_i with $i = 1, 2$ introduced in (2.4), we can obtain

$$B_1 = A_{22}, \ C_1 = A_{11}, \ E_1 = A_{12}, \ F_1 = A_{21}$$

and

$$B_2 = A_{11}, \ C_2 = A_{22}, \ E_2 = A_{21}, \ F_2 = A_{12}.$$

Hence, the matrix A can be rewritten as

$$A = \begin{pmatrix} C_1 & E_1 \\ F_1 & B_1 \end{pmatrix} \text{ and } A = \begin{pmatrix} B_2 & F_2 \\ E_2 & C_2 \end{pmatrix},$$

which implies $F_1 = E_2$ and $B_1 = C_2$. So we have $A = \begin{pmatrix} C_1 & E_1 \\ E_2 & C_2 \end{pmatrix}$.

Denote $A^o = \begin{pmatrix} F_1 & B_1 \\ B_2 & F_2 \end{pmatrix}$, then from (2.16) we can get

$$\begin{aligned} Y &= \overline{S}^{-1}(A - E\overline{B}^{-1}A^o) \\ &= \overline{S}^{-1}\left[\begin{pmatrix} C_1 & E_1 \\ E_2 & C_2 \end{pmatrix} - \begin{pmatrix} E_1\overline{B}_1^{-1} & 0 \\ 0 & E_2\overline{B}_2^{-1} \end{pmatrix} \begin{pmatrix} F_1 & B_1 \\ B_2 & F_2 \end{pmatrix} \right] \\ &= \begin{pmatrix} \overline{S}_1^{-1} & 0 \\ 0 & \overline{S}_2^{-1} \end{pmatrix} \begin{pmatrix} C_1 - E_1\overline{B}_1^{-1}F_1 & E_1 - E_1\overline{B}_1^{-1}B_1 \\ E_2 - E_2\overline{B}_2^{-1}B_2 & C_2 - E_2\overline{B}_2^{-1}F_2 \end{pmatrix} \\ &= \begin{pmatrix} I & \overline{S}_1^{-1}E_1(I - \overline{B}_1^{-1}B_1) \\ \overline{S}_2^{-1}E_2(I - \overline{B}_2^{-1}B_2) & I \end{pmatrix} \\ &= I + \begin{pmatrix} 0 & Z_1 \\ Z_2 & 0 \end{pmatrix}, \end{aligned}$$

where $Z_1 = \overline{S}_1^{-1}E_1(I - \overline{B}_1^{-1}B_1)$ and $Z_2 = \overline{S}_2^{-1}E_2(I - \overline{B}_2^{-1}B_2)$. Let

$$Z = \begin{pmatrix} 0 & Z_1 \\ Z_2 & 0 \end{pmatrix},$$

then the eigenvalues of Y satisfy

$$\lambda(Y) = 1 + \lambda(Z). \tag{2.17}$$

Furthermore, if $\overline{B}_i$ $(i = 1, 2)$ are taken as full rank matrices, then $\overline{B}_i = B_i$ $(i = 1, 2)$, which means $Z_1 = Z_2 = 0$. So Y is an identity matrix in this case, and the eigenvalues of Y are 1 with multiplicity n.

3. Properties of SMLR. In this section, we study some properties of the SMLR preconditioner.

PROPOSITION 3.1. *If $\overline{B}_i$ is taken as the full matrix for $i = 1, 2, \ldots, c$, then the iteration step will be only one when the SMLR preconditioner is applied to solve (2.14).*

Proof. If the matrix $\overline{B}_i$ $(i = 1, 2, \ldots, c)$ are taken as the full matrices, then $\overline{B}_i = B_i$. This means (2.14) is just equal to (2.7). Hence, the result holds true directly. ∎

PROPOSITION 3.2. *For the matrix S_i and its approximate matrix $\overline{S}_i$ we have*

$$S_i - \overline{S}_i = E_i\Delta_{ik}F_i = \Delta_i, \tag{3.1}$$

where $\Delta_{ik} = B_i^{-1} - \overline{B}_i^{-1}$. Moreover, we can obtain

$$\Delta S = S - \overline{S} = diag(\Delta_1, \Delta_2, \ldots, \Delta_c).$$

Proof. Obviously, we have

$$S_i - \overline{S}_i = E_iB_i^{-1}F_i - E_i\overline{B}_i^{-1}F_i = E_i\Delta_iF_i.$$

So we can obtain $S - \overline{S} = \mathrm{diag}(\Delta_1, \Delta_2, \ldots, \Delta_c)$. ∎

PROPOSITION 3.3. *The condition number of the coefficient matrix of system (2.14) is*

$$\kappa(\overline{S}) = \frac{\lambda_{max}(\overline{S}_i)}{\lambda_{min}(\overline{S}_j)}, \ i, j = 1, 2, \cdots, c. \tag{3.2}$$

In addition, the condition number of the coefficient matrix of (2.7) is:

$$\kappa(S) = \frac{\lambda_{max}(S_i)}{\lambda_{min}(S_j)},\ i, j = 1, 2, \cdots, c. \tag{3.3}$$

Here, $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$ denote the largest and smallest module of the eigenvalues for the corresponding matrix, respectively.

Proposition 3.3 studies the condition number of the matrices S and $\overline{S}$. From Proposition 3.3, we can see

$$\kappa(S_i) \leq \kappa(S),\ \text{and}\ \kappa(\overline{S}_i) \leq \kappa(\overline{S}),\quad (i = 1, 2, \ldots, c).$$

Now, we study the eigenvalues property of S and $\overline{S}$.

LEMMA 3.4. *If η is an eigenvalue of $A + \triangle A \in C^{n\times n}$ and $X^{-1}AX = D = diag(\xi_1, \xi_2, \cdots, \xi_n)$, then*

$$\min_{\xi\in\lambda(A)} |\xi - \eta| \leq \kappa_p(X)\|\triangle A\|_p,$$

where $\|\cdot\|_p$ denotes any of the p-norms.

Proof. See [6, 7] for detail. □

From the result of Lemma 3.4, we can get the following Corollary, which shows the eigenvalue perturbation property of the matrix S_i for $i = 1, 2, \ldots, c$.

COROLLARY 3.5. *If μ_j $(j = i_1, i_2, \cdots, i_n)$ are eigenvalues of $\overline{S}_i \in \mathbb{R}^{i_n\times i_n}$ and*

$$X_i^{-1} S_i X_i = \Lambda_i = diag(\lambda_{i_1}, \lambda_{i_2}, \cdots, \lambda_{i_n}),$$

then

$$\min_{i_1\leq j\leq i_n} |\lambda_j - \mu_j| \leq \kappa_p(X_i)\|\Delta_i\|_p.$$

Here, λ_j $(j = i_1, i_2, \cdots, i_n)$ are the eigenvalues of S_i. Moreover, if S_i is symmetric, then there is an orthogonal matrix Q_i such that

$$Q_i^{-1} S_i Q_i = \Lambda_i = diag(\lambda_{i_1}, \lambda_{i_2}, \cdots, \lambda_{i_n}),$$

so we have

$$\min_{i_1\leq j\leq i_n} |\lambda_j - \mu_j| \leq \|\Delta_i\|.$$

By making use of Corollary 3.5, we can obtain the following result.

PROPOSITION 3.6. *For the matrices $S, \overline{S} \in R^{n\times n}$, we have*

$$\min_{\lambda\in\lambda(S),\mu\in\lambda(\overline{S})} |\lambda - \mu| \leq \min_{1\leq i\leq c} \kappa_p(X_i)\|\Delta_i\|_p. \tag{3.4}$$

Moreover, if A is a symmetric matrix, then we have

$$\min_{\lambda\in\lambda(S),\mu\in\lambda(\overline{S})} |\lambda - \mu| \leq \min_{1\leq i\leq c} \|\Delta_i\|. \tag{3.5}$$

Proof. Since

$$\begin{aligned}\min_{\lambda\in\lambda(S),\mu\in\lambda(\overline{S})}|\lambda-\mu| &\le \min_{1\le i\le c}\min_{i_1\le j\le i_n}|\lambda_j-\mu_j|\\ &\le \min_{1\le i\le c}\kappa_p(X_i)\|\Delta_i\|_p,\end{aligned}$$

the result in equation (3.4) holds true directly. Here, λ_j and μ_j $(j = i_1, i_2, \cdots, i_n)$ are the eigenvalues of S_i and $\overline{S}_i$ $(1 \le i \le c)$, respectively. Moreover, if the matrix A is symmetric, then $S_i = C_i - E_i B_i^{-1} E_i^T$ $(i = 1, 2, \cdots, c)$ is also a symmetric matrix. So we can obtain (3.5) based on the result in Corollary 3.5. This completes the proof. □

Above proposition illustrates the eigenvalue perturbation property of the approximated system (2.14) of linear system (2.7). Following results show the upper and lower bounds of relative errors between the approximated solutions obtained from (2.14) and the exact solutions of (2.7) for each color.

THEOREM 3.7. *For $i = 1, 2, \ldots, c$, the following results hold true:*

$$\frac{\|\overline{x}_i\|}{\|x_i\|} \le \kappa(\overline{S_i})\frac{\|S_i\|}{\|\overline{S_i}\|}\frac{\|\overline{f_i}\|}{\|f_i\|}, \tag{3.6}$$

where $\overline{x}_i$ satisfies $\overline{S_i}\overline{x_i} = \overline{f_i}$. Moreover,

$$1-\kappa(\overline{S_i})\frac{\|S_i\|}{\|\overline{S_i}\|}\frac{\|\overline{f_i}\|}{\|f_i\|} \le \frac{\|\Delta x_i\|}{\|x_i\|} \le 1+\kappa(\overline{S_i})\frac{\|S_i\|}{\|\overline{S_i}\|}\frac{\|\overline{f_i}\|}{\|f_i\|}. \tag{3.7}$$

Here, $\Delta x_i = x_i - \overline{x}_i$ is the error between x_i and $\overline{x}_i$.

Proof. Note $\overline{S_i}\overline{x_i} = \overline{f_i}$, we can obtain

$$\|\overline{x_i}\| = \|\overline{S_i}^{-1}\overline{f_i}\| \le \|\overline{S_i}^{-1}\|\|\overline{f_i}\|.$$

On the other hand, we have $S_i x_i = f_i$ which means

$$\frac{1}{\|x_i\|} \le \frac{\|S_i\|}{\|f_i\|}.$$

So we have

$$\begin{aligned}\frac{\|\overline{x}_i\|}{\|x_i\|} &\le \|S_i\|\|\overline{S_i}^{-1}\|\frac{\|\overline{f_i}\|}{\|f_i\|}\\ &= \|\overline{S_i}^{-1}\|\|\overline{S_i}\|\frac{\|S_i\|}{\|\overline{S_i}\|}\frac{\|\overline{f}_i\|}{\|f_i\|}\\ &= \kappa(\overline{S}_i)\frac{\|S_i\|}{\|\overline{S_i}\|}\frac{\|\overline{f_i}\|}{\|f_i\|}.\end{aligned}$$

Now, we prove the rest conclusion of this theorem. From the triangle inequalities below

$$\pm(\|x_i\| - \|\Delta x_i\|) \le \|x_i - \Delta x_i\|,$$

we can obtain

$$1-\frac{\|x_i-\Delta x_i\|}{\|x_i\|} \le \frac{\|\Delta x_i\|}{\|x_i\|} \le 1+\frac{\|x_i-\Delta x_i\|}{\|x_i\|}.$$

Note $x_i - \Delta x_i = \overline{x}_i$, then apply (3.6) which we have proved to the above inequality, the result in (3.7) sets up directly. The proof is completed. □

4. Improvement. In this section, we consider improve the preconditioner by using some corrected technics. Here, we will apply the Block-Jacobi type correction method presented in [18] to correct the solution $x = (x_1^T, x_2^T, \ldots, x_c^T)^T$ obtained from the approximated system (2.14) of (2.7). The application of Block-Jacobi type correction here is different with the one used for MCLR preconditioner studied in [18], since the Block-Jacobi corrected method is used to construct the MCLR preconditioner in [18] rather than correct the solution obtained by the MCLR preconditioner. In addition, the convergence property of the Block-Jacobi type corrected method, which has not been analysed in [18] is discuss in this section. Note we have

$$\begin{pmatrix} E_i & C_i \end{pmatrix} \begin{pmatrix} x_i^o \\ x_i \end{pmatrix} \approx b_i,$$

with $x_i^o = (x_1^T, \ldots, x_{i-1}^T, x_{i+1}^T, \ldots, x_c^T)^T$, where x_i $(i = 1, 2, \ldots, c)$ are the solutions we have obtained after solving (2.14). Let the corrected solution of x_i be $x_i + \delta_i$ and let

$$\begin{pmatrix} E_i & C_i \end{pmatrix} \begin{pmatrix} x_i^o \\ x_i + \delta_i \end{pmatrix} = b_i$$

for color i, then we have

$$\begin{aligned} C_i\delta_i &= b_i - E_i x_i^o - C_i x_i \\ &= b_i - A_i x \end{aligned}$$

with $A_i = \begin{pmatrix} A_{i,1} & A_{i,2} & \cdots & A_{i,c} \end{pmatrix}$. Hence, for color $i = 1, 2, \cdots, c$, to obtain the corrected solution of x_i, we just need to solve a linear system with coefficient matrix being the block diagonal C_i. Figure 4.1 illustrates the Block-Jacobi correction step for the problem given in Figure 2.1. In addition, the above correction step can be applied

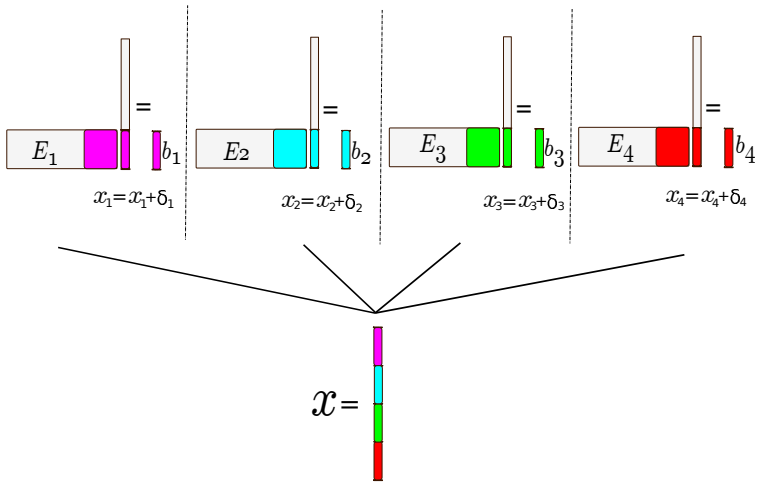

FIG. 4.1. *Block-Jacobi type correction for the system given in Figure 2.1.*

more than once, i.e., after obtaining a corrected solution x_i^c of x_i, we can use the Block-Jacobi type corrected method again to correct x_i^c to get next corrected solution of color i. We call the method combined SMLR preconditioner with the Block-Jacobi

type correction SMLR preconditioner with *correction*. Algorithm 2 illustrates the SMLR preconditioner with *correction* method. We denote this method by function F_{cor} and m is the number of Block-Jacobi correction steps.

ALGORITHM 2
Computing $x = F_{cor}(b)$

1: Applying Algorithm 1 to get solution $x = (x_1^T, x_2^T, \ldots, x_c^T)^T$
2: **For** $j = 1 : m$ Do
3: **For** $i = 1 : c$
4: Update local residual: $r_i = b_i - A_i x$
5: Solve $C_i \delta_i = r_i$
6: Get corrected solution $y_i = x_i + \delta_i$
7: **EndFor**
8: Update solution $x = y$
9: **EndDo**

In the rest part of this section we will study the convergence property of the Block-Jacobi corrected method. In fact, the Block-Jacobi correction step can be rewritten as

$$\begin{cases} A_{ii}\delta_i = b_i - A_i x, \\ x_i = x_i + \delta_i, \\ i = 1, 2, \ldots, c, \end{cases} \tag{4.1}$$

where $A_i = \begin{pmatrix} A_{i1} & A_{i2} \cdots & A_{ic} \end{pmatrix}$. This is equal to

$$\begin{cases} \begin{pmatrix} A_{11} & 0 & \cdots & 0 \\ 0 & A_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & A_{cc} \end{pmatrix} \begin{pmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_c \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{pmatrix} - \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1c} \\ A_{21} & A_{22} & \cdots & A_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ A_{c1} & A_{c1} & \cdots & A_{cc} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_c \end{pmatrix}, \\ x = x + \delta, \end{cases} \tag{4.2}$$

where $x = (x_1^T, x_2^T, \ldots, x_c^T)^T$ and $\delta = (\delta_1^T, \delta_2^T, \ldots, \delta_c^T)^T$. Let A_d be the block diagonal part of matrix A, i.e., $A_d = \mathrm{diag}(A_{1,1},\ A_{2,2},\ \ldots,\ A_{c,c})$, then (4.1) can be obtained by

$$x^{i+1} = x^i + A_d^{-1} r^i = (I - A_d^{-1} A) x^i + A_d^{-1} b, \tag{4.3}$$

where x^i and r^i are definite below

$$x^i = \begin{pmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_c^i \end{pmatrix} \text{ and } r^i = \begin{pmatrix} r_1^i \\ r_2^i \\ \vdots \\ r_c^i \end{pmatrix}.$$

Hence, the iteration matrix of (4.3) is $T = I - A_d^{-1} A$, which implies that the iteration (4.1) is convergent if $\rho(T) < 1$, where $\rho(T)$ is the spectral radius of the matrix T. Following theorem proposes a sufficient condition to converge for the iteration (4.1).

THEOREM 4.1. *Suppose that A is a SPD matrix, then we have $\rho(T) < 1$ if there exist a block two-by-two partition of A:*

$$A = \begin{pmatrix} L_{11} & L_{12} \\ L_{12}^T & L_{22} \end{pmatrix},\ L_{11} \in R^{p \times p},\ L_{22} \in R^{q \times q},$$

which satisfies $A_d = \begin{pmatrix} L_{11} & 0 \\ 0 & L_{22} \end{pmatrix}$. *Here,* $p + q = n$.

Proof. The eigenvalues of T are equal to that of matrix $J = I - A_d^{-\frac{1}{2}} A A_d^{-\frac{1}{2}}$, because T is similar to J. Since

$$\begin{aligned} J &= I - \begin{pmatrix} L_{11}^{-\frac{1}{2}} & 0 \\ 0 & L_{22}^{-\frac{1}{2}} \end{pmatrix} \begin{pmatrix} L_{11} & L_{12} \\ L_{12}^T & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^{-\frac{1}{2}} & 0 \\ 0 & L_{22}^{-\frac{1}{2}} \end{pmatrix} \\ &= \begin{pmatrix} 0 & L_{11}^{-\frac{1}{2}} L_{12} L_{22}^{-\frac{1}{2}} \\ L_{22}^{-\frac{1}{2}} L_{12}^T L_{11}^{-\frac{1}{2}} & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & \Phi \\ \Phi^T & 0 \end{pmatrix}, \end{aligned}$$

where $\Phi = L_{11}^{-\frac{1}{2}} L_{12} L_{22}^{-\frac{1}{2}} \in R^{p \times q}$. So the eigenvalues of T are the t pairs $\pm\sigma_i$, with $n - 2t$ additional zero eigenvalues if $n > 2t$; see [18, 7]. Here, σ_i $(i = 1, 2, \ldots, t)$ are the non-zero singular values of the matrix Φ. From the result of Theorem 3.2 in [18] we can see $\sigma_i \in [0, 1)$ for $i = 1, 2, \ldots, t$, so we have

$$\rho(T) = \max |\lambda(T)| \leq \max_{1 \leq i \leq t} \sigma_i < 1.$$

The proof is completed. □

5. Numerical experiments. In this section, we present some numerical tests to illustrate the efficiency of the SMLR preconditioner with *correction* for solving 2D and 3D large sparse linear systems. The codes were written by C++ and compiled by the Intel C compiler with -O3 optimization and all the experiments were run on the `Mesabi` Linux cluster (with a single node) at the Minnesota Supercomputing Institute. In our tests, the `PartGraphKway` comes from METIS [12] package was used to partition the original coefficient matrix A_0. In the numerical results, the time to build the preconditioner reported comes from the factorization of block diagonal submatrices of the reordered coefficient matrices and the computation of the low-rank correction terms. The reordering time for the original matrix was regarded as preprocessing and was not reported in the numerical results.

In actual computations, the right-hand-side vector b was chosen such that $Ae = b$ with e being a random vector. Moreover, the initial iteration vectors used were zero vectors for all the tests. We need to use the following notations in the coming numerical tests:

- fill: $\frac{nnz(prec)}{nnz(A)}$;
- its: number of iterations of preconditioned GMRES or preconditioned CG required for relative residual smaller than 10^{-6}. In addition, we use "F" to indicate that preconditioned GMRES method or preconditioned CG method can not converge after 300 iterations;
- p-t: CPU time to build the preconditioner (in seconds);
- i-t: CPU time to iteration phase. We denote the iteration time by "–" when the preconditioned GMRES or preconditioned CG method can not converge after 300 iteration (in seconds);
- t-t: CPU time to build and apply SMLR preconditioner with *correction* (in seconds);
- r_k: rank which is used in the low-rank corrections term;
- n_d: number of subdomains;

- tol_0: threshold used to solve the linear systems with coefficient matrices being the Schur complements in (2.14);
- its_0: number of iterations for solving the linear systems with coefficient matrices being the Schur complements in (2.14);
- m: number of Block-Jacobi type correction steps.

For the SPD matrices, the SMLR preconditioner with *correction* was compared with the incomplete Cholesky factorization with threshold dropping (ICT) and the MSLR preconditioner [17]. The accelerator we used for these three methods is the conjugate gradient (CG) method [11]. Otherwise, the SMLR method was compared with the incomplete LU factorization with threshold dropping (ILUT) and the GMSLR preconditioner [5]. Here, the accelerator we used for the SMLR, MSLR and ILUT methods is the generalized minimal residual (GMRES) [16] method. In all the tests, OpenMP [2] was used to parallelize the applications of SMLR, MSLR and GMSLR preconditioners. The number of threads used here is the number of cores which is 24.

5.1. Test 1. First, we consider the following nonsymmetric problem:

$$\begin{aligned} -\triangle u - \alpha \cdot \nabla u - \beta u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned} \tag{5.1}$$

which is a shifted convection-diffusion equation. Here, $\Omega = (0,1)^3$ and $\alpha \in R^3$. This equation is discreted by using finite differences with the standard 7-point stencil in 3 dimensions.

TABLE 5.1
The iteration counts and CPU time for solving (5.1) with $s = 0.16$ and $\alpha = [.1, .1, .1]$ on a 32^3 grid by applying the GMRES-SMLR method with correction *($m = 5$). Here, n_d is taken as 20 and threshold used in the incomplete LU factorization is fixed to 10^{-2}. The rank for the low-rank correction is fixed to $r_k = 2$.*

tol_0	its	fill (ILU)	fill (Low-rank)	p-t	i-t	t-t
.2	67	2.64	1.76	.09	4.10	4.19
.1	62	2.64	1.76	.09	3.83	3.92
.05	58	2.64	1.76	.10	3.65	3.75
.01	54	2.64	1.76	.15	3.50	3.65
.005	52	2.64	1.76	.10	3.55	3.65
.001	49	2.64	1.76	.09	3.67	3.76
.0005	47	2.64	1.76	.09	3.80	3.89
.0001	46	2.64	1.76	.10	3.87	3.97

5.1.1. Effect of tol_0 and its_0. One important effect we need to consider is the threshold used for solving the linear systems with coefficient matrices being Schur complements in (2.14). We study this factor on the SMLR preconditioner with *correction* by solving problem (5.1) with $s = 0.16$ and $\alpha = 0.1^3$ on a 32^3 grid. Here, we shift the discretized convection-diffusion operator by sI with $s = 5 * h$, where h is the mesh size. Moreover, the number of Block-Jacobi correction steps is set as $m = 5$. The numerical results are shown in Table 5.1, from which we can see a steady improvement of the number of iteration steps when the threshold reduces from 0.2 to 0.0001. This is because that a smaller threshold always results in a better approximation to the solution of equation (2.14) which leads to a smaller number of iteration steps. In addition, the time to compute the solution of (2.14) increases at the same time, so it is unworthy to use a lower tolerance to reduce the smaller decrease of the iteration steps. Moreover, the effect of the number of iteration steps its_0 used to solve

(2.14) on the SMLR preconditioner with *correction* is similar to that of tol_0. This is verified in Table 5.2, where the fill factors for the ILU factorizations and low-rank correction terms were omitted since these results are the same as the results in Table 5.1.

TABLE 5.2
The iteration counts and CPU time for solving (5.1) with $s = 0.16$ *and* $\alpha = [.1, .1, .1]$ *on a* 32^3 *grid with the GMRES-SMLR method with* correction *(*$m = 5$*). Here,* n_d *is taken as* 20 *and threshold used in the incomplete LU factorization is fixed to* 10^{-2}.

its_0	its	p-t	i-t	t-t
0	53	.17	4.83	5.00
1	42	.15	3.80	3.95
2	39	.15	3.55	3.70
3	35	.15	3.58	3.73
4	30	.16	3.62	3.78

5.1.2. Effect of m. In this section, we study the performance of SMLR preconditioner with *correction* when the number of Block-Jacobi correction varies. We study this effect by solving the same problem as in the Section 5.1.1. As is see in Table 5.3, the iteration number decreases form 109 to 33 when m increases from 0 to 13, which illustrates the efficiency of the Block-Jacobi correction to improve the SMLR preconditioner. Meanwhile, the cost to apply this Block-Jacobi correction increases with higher m, which is further illustrated by Figure 5.1, from which one can see the iteration time first decreases when m increase from 0 to 5, and then increases as m increases from 5 to 13. For this test problem, the optimal number of correction steps is $m = 5$, which results in a smallest iteration time and total time. In the remaining tests, we set m as $m = 5$ for simplicity, if not special specified.

TABLE 5.3
The iteration counts and CPU time for solving (5.1) with $s = 0.16$ *and* $\alpha = [.1, .1, .1]$ *on a* 32^3 *grid with the GMRES-SMLR method. Here,* n_d *is taken as 20 and threshold used in the incomplete LU factorization and* tol_0 *are fixed to* 10^{-2}.

m	its	p-t	i-t	t-t
0	109	.18	4.83	6.80
1	85	.18	4.83	5.02
3	59	.17	3.35	3.52
5	49	.19	2.84	3.04
7	42	.18	2.98	3.16
9	36	.16	3.01	3.17
11	34	.18	3.29	3.37
13	33	.17	3.88	3.64

5.2. Test 2. The second test is the symmetric problem:

$$\begin{aligned} -\triangle u - \beta u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned} \tag{5.2}$$

5.2.1. Effect of n_d. The number of the subdomain is also an important factor we need to consider. Here, the test matrix was obtained by discretizing (5.2) with $\beta = 0.0$ on a 50^3 grid. As the number of subdomain increases from 10 to 60, one can see from Table 5.4 that the iteration number has a slight fluctuation. Moreover, the fill factor from the low-rank term increases monotonously while the fill factor from the

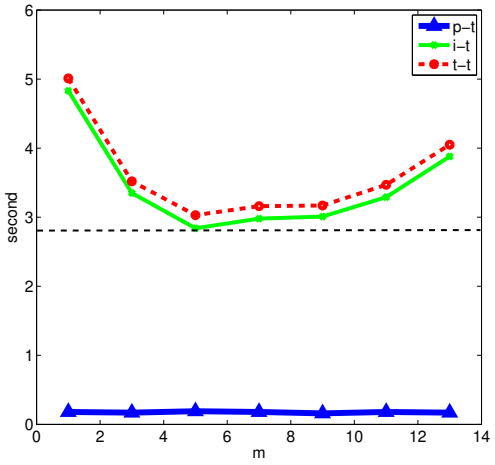

FIG. 5.1. *The time to build the preconditioner and the iteration time with respect to different m.*

ILU decomposition decreases. This is because a bigger n_d leads to a larger number of colors, which will cause more low-rank correction terms. On the other hand, a larger n_d results in a smaller size of each block diagonal matrix, which reduces the storage of the ILU factorizations. In addition, we find $n_d = 30$ is usually the optimal value for this problem based on various tests, so we set n_d to 30 in the remaining experiments for simplicity.

TABLE 5.4

The fill factor, iteration counts and CPU time for solving (5.2) with $s = 0.0$ on a 50^3 grid with the GMRES-SMLR method with correction *($m = 5$). Here, the rank for the low-rank correction matrices is fixed at 2, and the thresholds used in the incomplete LU factorization and tol_0 are taken as 10^{-2}.*

nd	fill (ILU)	fill (Low-rank)	fill (total)	its	p-t	i-t
10	2.81	1.16	3.97	10	.47	2.21
20	2.75	1.43	4.18	10	.70	2.37
30	2.70	1.74	4.44	10	.70	2.19
40	2.66	2.03	4.69	10	.71	2.39
50	2.63	2.12	4.75	11	.80	3.06
60	2.61	2.18	4.79	10	.75	2.75

5.2.2. Effect of r_k. Now we study the effect of the low-rank correction on the performance of the SMLR preconditioner with *correction*. To illustrate this, we solve the same test problem as in the Section 5.2.1 with different values for the rank r_k. Here, the number of Block-Jacobi correction steps was taken as 5. The test problem (5.2) becomes more difficult to solve with a larger s since the indefiniteness of the matrix becomes higher, so we expected that the iteration steps and computing time should increase. This is verified with the numerical results in Table 5.5 when s is increase from 0 to 0.2. As one can see from Table 5.5, as the rank increases from 0 to 10, the iteration number and the iteration time decreases. This is because a larger rank results in a more accurate approximation to the coefficient matrix of linear system (2.7). In addition, we can see from Table 5.5 that the preconditioner building

time increases when the rank becomes larger, this is because more time is needed to compute the low-rank correction terms for higher rank. Meanwhile, a larger rank also results in more storage to the low-rank correction terms especially for the indefinite matrices. Hence, we don't need to take a higher rank ($r_k \leq 5$) to correct the Schur complements in actual application.

TABLE 5.5
The fill factor, iteration counts and CPU time for solving (5.2) on a 50^3 grid with the GMRES-SMLR method with correction *($m = 5$). Here, the thresholds used in the incomplete LU factorization and tol_0 are taken as 10^{-2}.*

s	$r_k = 0$				$r_k = 2$				$r_k = 5$				$r_k = 10$			
	fill	its	p-t	i-t	fill	its	p-t	i-t	fill	its	p-t	i-t	fill	its	p-t	i-t
.00	2.68	10	.40	1.15	4.44	9	.65	.98	7.06	8	.78	.85	11.43	7	.90	.69
.05	2.70	38	.68	6.50	4.78	34	.76	5.87	7.81	30	.99	5.44	12.90	26	1.12	4.80
.15	2.75	116	.91	12.03	4.80	107	1.11	11.68	7.83	100	1.25	10.48	12.93	93	1.32	9.47

To further illustrate the effect of the value of r_k on the SMLR preconditioner, we present the numerical results for the same tests where the number of Block-Jacobi correction steps are taken as $m = 0$. The corresponding fill factors, iteration counts and iteration time are given in Table 5.6. As we can see from this table, the iteration number reduces faster than that of the results in 5.5 when r_k increases from 0 to 10 and s are fixed. So the low-rank correction terms is more important to the SMLR preconditioner without Block-Jacobi correction. In addition, the iteration number and the iteration time in Table 5.5 are both less than that of the results in Table 5.6 when r_k and s are fixed, which implies that the Block-Jacobi correction can further improve the SMLR preconditioner when the same rank is used to solve the problem.

TABLE 5.6
The fill factor, iteration counts and CPU time for solving (5.2) on a 50^3 grid with the GMRES-SMLR method ($m = 0$). Here, the thresholds used in the incomplete LU factorization and tol_0 are taken as 10^{-2}.

s	$r_k = 0$				$r_k = 2$				$r_k = 5$				$r_k = 10$			
	fill	its	p-t	i-t	fill	its	p-t	i-t	fill	its	p-t	i-t	fill	its	p-t	i-t
.00	2.68	18	.42	1.89	4.44	15	.67	1.57	7.06	12	.77	1.26	11.43	10	.92	.1.06
.05	2.70	52	.66	8.50	4.78	44	.78	7.21	7.81	37	.99	6.15	12.90	30	1.15	5.07
.15	2.76	157	.93	15.06	4.80	140	1.10	13.34	7.83	125	1.27	11.90	12.93	100	1.33	10.16

In the rest part of this section, we present more numerical results to the SMLR preconditioner with *correction* for solving the 2D and 3D Laplacian matrices. Here, we solve (5.2) for the positive case with $\beta = 0$ and indefinite case with $\beta > 0$. We set r_k to $r_k = 2$ and $r_k = 5$ for the matrices with $\beta = 0$ and $\beta > 0$, respectively. Also, for the indefinite case, we fixed s to be $s = 5 * h$ for mesh size h. The numerical results for 2D problems are given in Table 5.7, from which we can see that the new method outperforms the ICT and MSLR preconditioners when they have almost the same fill-factors since the iteration number and computing time to converge for the SMLR preconditioner with *correction* are less than that of the other two methods.

The numerical results for the 3D cases are given in Table 5.8, which illustrates that the SMLR preconditioner with *correction* is more efficient than the ILUT and GMSLR methods for solving indefinite problems. The ILUT and GMSLR methods even can not converge for the matrices with grid 1024×1024 ($s = 0.005$) and grid $128\times128\times128$ ($s = 0.04$) which have 394 and 217 negative eigenvalues, respectively.

TABLE 5.7
Comparison among SMLR with *correction* ($m = 5$), ICT/ILUT *and* MSLR/GMSLR *preconditioners for solving symmetric positive definite/indefinite linear systems from the 2D PDEs with the* CG/GMRES *method.*

Mesh	s	SMLR				ICT				MSLR					
		fill	its	p-t	i-t	fill	its	p-t	i-t	lev	r_k	fill	its	p-t	i-t
256^2	0	3.09	20	.05	.12	3.37	55	.03	.43	5	8	3.11	35	.05	.21
512^2	0	3.19	32	.50	2.67	3.38	55	.26	3.81	7	8	3.15	72	.13	3.20
1024^2	0	3.37	52	1.92	10.23	3.39	122	0.64	19.97	9	8	3.18	117	0.59	15.99
Mesh	s	SMLR				ILUT				GMSLR					
		fill	its	p-t	i-t	fill	its	p-t	i-t	lev	r_k	fill	its	p-t	i-t
256^2	0.02	3.09	171	.19	0.69	3.37	181	.05	.1.63	5	16	3.18	270	.06	1.25
512^2	0.01	3.14	208	.53	11.65	3.08	298	.20	17.90	10	16	3.30	279	.22	15.88
1024^2	0.005	3.37	167	1.52	14.01	3.30	F	2.24	–	13	16	3.38	F	2.30	–

TABLE 5.8
Comparison among SMLR with *correction* ($m = 5$), ICT/ILUT *and* MSLR/GMSLR *preconditioners for solving symmetric positive definite/indefinite linear systems from the 3D PDEs with the* CG/GMRES *method.*

Mesh	s	SMLR				ICT				MSLR					
		fill	its	p-t	i-t	fill	its	p-t	i-t	lev	rk	fill	its	p-t	i-t
32^3	0	3.08	8	.09	.12	3.06	23	.29	.30	7	8	3.10	15	.10	.20
64^3	0	2.80	12	.12	.98	2.96	26	.26	2.03	10	8	3.09	24	.30	1.53
128^3	0	2.87	18	1.46	7.09	2.98	41	2.16	20.02	13	8	3.03	27	2.08	11.08
Mesh	s	SMLR				ILUT				GMSLR					
		fill	its	p-t	i-t	fill	its	p-t	i-t	lev	r_k	fill	its	p-t	i-t
32^3	0.16	3.51	47	.18	.24	3.06	85	.03	.51	7	16	3.57	75	.11	.35
64^3	0.08	2.80	127	.39	8.09	3.00	208	.45	19.89	10	16	3.19	153	1.04	12.63
128^3	0.04	3.33	233	3.50	12.01	2.98	F	10.23	–	13	16	3.43	F	5.32	–

5.3. Test 3. In this section, we apply our SMLR preconditioner with *correction* to some general matrices (Table 5.9) come from SuiteSparse Matrix Collection [4] to further illustrate the efficiency of this new method for solving general large sparse linear systems. The numerical results for these matrices are presented in Table 5.10.

From this table we can see that the SMLR preconditioner with *correction* outperforms the ICT and MSLR methods for solving the symmetric matrices and outperforms the ILUT and GMSLR methods to solve the nonsymmetric matrices.

TABLE 5.9
Names, orders (N), numbers of nonzeros (nnz) and short descriptions of the test matrices.

Matrix	Order	nnz	symmetric	Description
Dubcova3	146,689	3,636,643	yes	2D/3D Problem
ecology1	1,000,000	4,996,000	yes	landscape ecology problem
thermal1	82,654	574,458	yes	thermal problem
thermal2	1,228,045	8,580,313	yes	thermal problem
Atmosmodd	1,270,432	8,814,880	no	atmospheric model
Atmosmodl	1,489,752	10,319,760	no	atmospheric model
Transport	1,602,111	23,500,731	no	CFD problem

TABLE 5.10
Comparison among SMLR with *correction* ($m = 5$), ICT/ILUT *and* MSLR/GMSLR *preconditioners for solving general sparse linear systems along with* CG *or* GMRES.

Matrix	SMLR				ICT				MSLR					
	fill	its	p-t	i-t	fill	its	p-t	i-t	lev	rk	fill	its	p-t	i-t
Dubcova3	1.57	19	.76	1.60	1.59	38	1.52	3.34	8	64	1.60	23	1.58	2.61
ecology1	2.58	10	1.06	1.07	2.60	20	.60	3.31	11	50	2.68	16	1.11	2.43
thermal1	2.43	34	.23	.99	2.52	79	.16	2.41	6	64	2.60	69	.40	1.72
thermal2	2.94	66	3.62	6.96	2.54	171	3.01	14.12	8	180	2.33	122	4.02	9.98
Matrix	SMLR				ILUT				GMSLR					
	fill	its	p-t	i-t	fill	its	p-t	i-t	lev	rk	fill	its	p-t	i-t
Atmosmodd	2.85	21	2.94	4.71	2.97	47	1.27	11.53	10	16	2.98	38	4.78	8.89
Atmosmodl	2.89	11	1.19	2.90	2.97	27	2.51	7.71	11	4	2.82	21	2.77	4.34
Transport	1.39	41	3.67	11.89	1.41	77	2.33	29.65	11	4	1.46	58	2.09	17.90

6. Conclusion. In this paper, we present an efficient low-rank correction parallel preconditioner for solving general large sparse linear systems based on the Schur complement combined with multicoloring ordering. The original problem is solved after it is transfered to some smaller linear systems which can be solved at the same time. The inverse of Schur complement corresponding to each color is approximated via a low-rank correction term.

The SMLR preconditioner with *correction* is very efficient for solving symmetric and nonsymmetric problems. In addition, the new method is robust to solve the highly indefinite matrices which stand techniques (ICT and ILUT) can not handle.

In the future, we may extend the application of SMLR method to the eigenvalues problems, where the highly indefinite systems have to be solved.

Acknowledgements.

REFERENCES

[1] M. Benzi and M. Tuma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.

[2] OpenMP Architecture Review Board, *OpenMP Application Program Interface*, Version 3.1, 2011.

[3] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[4] T. A. Davis and Y. Hu, *The university of florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011).

[5] G. Dillon, V. Kalantzis, Y. Xi, and Y. Saad, *A hierarchical low-rank schur complement preconditioner for indefinite linear systems*, SIAM J. Sci. Comput., 40 (2018), pp. A2234–A2252.

[6] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins Studies in Mathematical Sciences, 2013.

[7] ———, *Matrix Computations, 4th edition*, Johns Hopkins University Press, Baltimore, MD, 4th ed., 2013.

[8] M. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[9] P. Hénon and Y. Saad, *A parallel multistage ILU factorization based on a hierarchical graph decomposition*, SIAM J. Sci. Comput., 28 (2006), pp. 2266–2293.

[10] J. C. Haws, M. Benzi, and M. Tuma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.

[11] M. R. Hestenes and E. L. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.

[12] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[13] R. Li and Y. Saad, *Divide and conquer low-rank preconditioners for symmetric matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A2069–A2095.

[14] R. Li, Y. Xi, and Y.Saad, *Schur complement-based domain decomposition preconditioners with low-rank corrections*, Numer. Linear Algebra Appl., 23 (2016), pp. 706–729.

[15] Y. Saad, *ILUT: a dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[16] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[17] Y. Xi, R. Li, and Y. Saad, *An algebraic multilevel preconditioner with low-rank corrections for sparse symmetric matrices*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 235–259.

[18] Q. Zheng, Y. Xi, and Y. Saad, *Multicoloring low-rank correction preconditioner for general sparse linear systems.*