Vol.2019-HPC-172 No.18
2019/12/19

Block Multicolor Gauss-Seidelにおけるブロッキングとカラーリングの影響分析

依田 凌[1,a)]　藤井 昭宏[1,b)]　田中 輝雄[1,c)]

概要：Block Multicolor Gauss-Seidel (BMC-GS) は，高い並列性と収束性を持つ有力な smoother のひとつである．本研究は，Gauss-Seidel に焦点をあてた，ブロッキングとカラーリングアルゴリズムが与える影響を分析する．カラーリングには，Welsh-Powell，修正 Cuthill-McKee，Algebraic Multicolor の三種を用いる．行列分析では，ブロッキングとカラーリングを含めた BMC-SGS の前処理行列を導出し，BMC-GS 前処理による固有値分布の改善を確認した．数値実験では，BMC-SGS 前処理付き CG 法をソルバとして，各ブロッキングとカラーリングを含めた計算コストを見積もり，並列性と収束性を分析した．Algebraic Multicolor による色付けが問題サイズが大きくなるほど優位になり，並列性を踏まえた最大ブロック分割数によるブロッキングが，計算コストの見積もりを最小にした．

キーワード：Block Multicolor Gauss-Seidel，グラフカラーリング，線形ソルバ，前処理

Analysis of the blocking and coloring in Block Multicolor Gauss-Seidel

1. はじめに

科学技術計算において，有限要素法や有限差分法により離散化された線形システムの係数行列は，大規模でスパースな性質を持つ．この性質よりシステムの求解には，前処理付き Krylov 部分空間法が広く使用される [18]．大規模問題に適した強力な前処理のひとつとして，反復回数が問題サイズに依存しないというスケーラブル性を持つ Multigrid 法が挙げられる [5], [17], [20]．Multigrid 法は粗いレベルを構築するマルチレベル解法であり，各レベルで smoother を適用するため，smoother 自体の性能が重要である．

Multigrid 法には従来，point-wise 型の Jacobi smoother や Gauss-Seidel smoother が使用され，並列性を抽出するカラーリングと，収束性を高めるブロッキングを行う Block Multicolor Gauss-Seidel (BMC-GS) は，有力な smoother のひとつである [1], [4], [15]．また別種として，多項式を利用した polynomial smoother が注目されている．Adams らは Multigrid 前処理付き CG (MGCG) の smoother として，Chebyshev 多項式を問題行列の固有値分布にスケーリングした Chebyshev smoother と Gauss-Seidel smoother を比較し，Chebyshev smoother が収束性と並列性の両面で優れている，と報告した [2]．Chebyshev smoother は疎行列ベクトル積 (SpMV) と同じ高い並列性を持ち，収束性でも Gauss-Seidel smoother より優れているのであれば，Gauss-Seidel を積極的に用いる理由は無い．しかしながら，我々は Chebyshev smoother の収束性が著しく悪化し，高い並列性を加味しても Gauss-Seidel smother に劣る例を確認した．図 1 は，4 次の Chebyshev smoother と Gauss-Seidel smoother を用いた MGCG の残差履歴を表す．Suitesparse Matrix Collection[7] より，対象とする行列は nasa4704，右辺ベクトルは nasa4704_b を用い，実装は PyAMG[3] を利用した．Gauss-Seidel smoother を用いたソルバは 698 回で収束したのに対し，Chebyshev smoother を用いたソルバは 100 反復後で残差減少の傾きが悪化し，行列サイズ 4704 回反復しても収束しなかった．このように Chebyshev smoother が有効ではない問題も多くあるため，Gauss-Seidel smoother に関連する収束性，並列性の向上を図る研究は依然として重要である．

[1] Kogakuin University, 1–24–2 Nishi-shinjuku, Shinjuku-ku, Tokyo, 163–8677, Japan
[a)] em18017@ns.kogakuin.ac.jp
[b)] fujii@cc.kogakuin.ac.jp
[c)] teru@cc.kogakuin.ac.jp

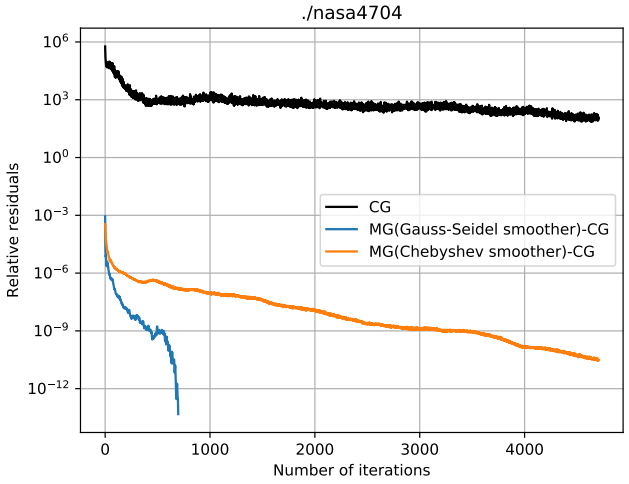


図 1　Suitesparse nasa4704 に対する残差履歴

問題行列の非ゼロ構造から代数グラフを構築し頂点彩色により並列性を抽出する手法は，BMC-GS だけでなく，Incomplete Cholesky (IC) または Incomplete LU (ILU) にも用いられる [10], [12], [14], [16]．IC/ILU に対しては，incompatible nodes と呼ばれる独立点によりカラーリングが分析されており，incompatible nodes が少ないほど収束性が高く，従来よりも色数を多く塗るカラーリングが良いと報告されている [8], [9]．しかし，同じグラフに対する同じ色付けの場合でも，Gauss-Seidel に対する効果と IC/ILU に対する効果は異なる．そのため Gauss-Seidel に焦点をあてたブロッキングとカラーリングの分析が必要である．本研究では，問題領域を METIS[13] を用いて複数のサブ領域に分割し各領域に 2 回の SGS を行う，Multiplicative Schwartz type BMC-SGS (MS-BMC-SGS)[14], [15] 単体を前処理とした CG 法を対象とする．Welsh-Powell (WP)[19]，修正 Cuthill-McKee (MCM)[6]，Algebraic Multicolor (AMC) [11] を用いて複数のブロッキングとカラーリングによる影響を分析する．

本稿は，次のように構成される．2 章では本研究に用いるカラーリングアルゴリズムを述べる．3 章では BMC-GS の概要を説明する．4 章では BMC-GS の行列を明示し，前処理による固有値分布の改善を確認する．5 章では Suitesparse Matrix Collection の行列を対象に行なった実験結果を示す．6 章で本研究の結論を述べる．

2. カラーリングアルゴリズム

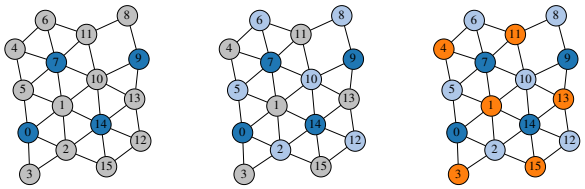
問題行列 A の非ゼロ構造から代数グラフを構築する．頂点は未知数番号に対応し，$A_{i,j}$ が非ゼロ要素である時，頂点 i から頂点 j に辺が張られる．この代数グラフに対して，一般的なグラフカラーリングアルゴリズムが適用可能である．本研究では Welsh-Powell (WP)，修正 Cuthill-McKee (MCM)，Algebraic Multicolor (AMC) の三種類のカラーリングアルゴリズムを用いる．

2.1 Welsh-Powell

Welsh-Powell (WP) は 1967 年に Welsh と Powell により提案されたカラーリングであり，最小彩色数を近似する色付けを行う [19]．つまり WP はなるべく少ない色数で彩色を行う．Algorithm1 に，WP アルゴリズムを示す．また図 2 に，WP を用いた色付けの例を示す． 次数降順の

Algorithm 1 Welsh-Powell

(1) Sort the vertices in descending order of the degree of a vertex. Let i-color be 0.
(2) Assign the i-color to the first vertex in the sorted list.
(3) Go down the sorted list and assign i-color to every vertex which is not connected the i-color assigined vertices.
(4) Increment i-color to i-color+1.
(5) Repeat the process from (2) to (4) until there aren't the color assigined vertices.



(a) Assign 0-color　(b) Assign 1-color　(c) Assign 2-color

図 2　Welsh-Powell coloring process for a graph. Coloring process proceeds (a) to (c).

頂点リストは，[14, 10, 7, 1, 2, 13, 11, 5, 0, 15, 12, 9, 6, 4, 8, 3] である．まず始めに，ステップ (2) より頂点 14 を色 0 で塗る．次にステップ (3) より，頂点リストを前から走査し，未だ塗られていないかつ隣接が同色でない頂点 7, 9, 0 を色 0 で塗る．走査が終了した後に，ステップ (4) より i-color をインクリメントし，色 1 とする．この処理を 2 回繰り返すことで，図 2 のグラフは 3 色で塗られる．WP は最小彩色数である保証はないが，なるべく少ない色数で彩色を行うアルゴリズムである．

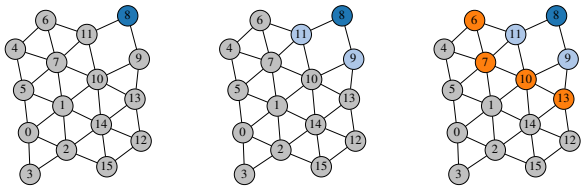
2.2 修正 Cuthill-McKee

Cuthill-McKee (CM) は行列のバンド幅をなるべく小さくするオーダリングアルゴリズムとして，1969 年に Cuthill と McKee により提案された手法である [6]．この手法は，幅優先探索の変種として解釈でき，CM を彩色に用いる場合，幅優先の深さを色として用いる．Algorithm2 に，CM アルゴリズムを示す．また図 3 に，CM を用いた色付けの例を示す．

まず始めに，{ 頂点番号，深さ } を状態とする空のキューを作る．次数が最も小さい頂点として頂点 8 を選び，深さ 0 として状態 {8, 0} をキューに入れる．ステップ (3) で，キューから取り出された状態 {8，0} より頂点 8 を色 0 で

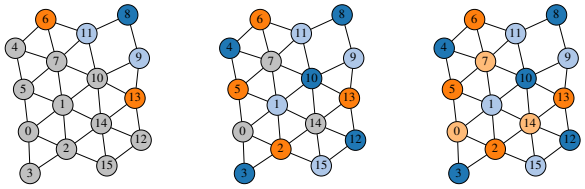
Algorithm 2 Cuthill-Mckee

(1) Prepare the empty queue of {a vertex, its depth from the root of the BFS-tree}. Let the depth be 0 and set the number of colors as C.
(2) Enqueue {a peripheral vertex (the vertex with the lowest degree of a vertex), itss depth}.
(3) Dequeue {dequeued vertex, its depth}. And assign the color of its depth to the dequeued vertex.
(4) Visit the adjacent vertices of the dequeued vertex in (3), and Enqueue the uncolored vertices as {the adjacent vertex, the dequeued vertex depth+1 mod C } in ascending order of the degree of a vertex.
(5) Repeat the process from (2) to (4) until there are no the color assigined vertices.



(a) Assign 0-color (b) Assign 1-color (c) Assign 2-color

図 3 Cuthill-Mckee coloring process for a graph. Coloring process proceeds (a) to (c).



(a) Assign 2-color (b) Assign 2-color (c) Assign 3-color

図 4 Modified Cuthill-Mckee coloring process for a graph. Coloring process proceeds (a) to (c).

塗る．次にステップ (4) で，隣接頂点 9, 11 を深さ 1 として状態 {9, 1}，{10, 1} をキューに入れる．ステップ (3) に戻り，キューから取り出された状態 {9, 1} より頂点 9 を色 1 で塗る．次のステップ (4) で，未だ塗られていない頂点 10, 13 を深さ 2 としてキューに入れる．この処理を深さ 2 まで繰り返した色付けが，図 3 (c) である．

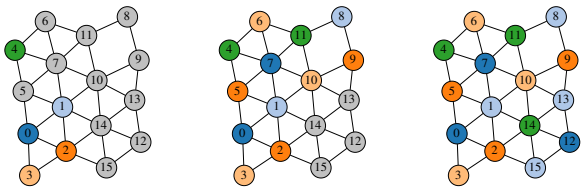
しかしながら我々は，依存関係を切り並列性を抽出するために CM カラーリングを用いる．図 3 の場合，色 2 で塗られている頂点が隣接しており，依存関係が切られていない．そのためアルゴリズムに次の修正を行う．キューから取り出し，頂点を塗る時に，隣接が同色でないか確認を行ない，もし同色でない場合はステップ (3) へ，同色の場合はステップ (2) へ，という処理を追加する．これより，隣接が同色であるカラーリングは行われないが，キューが空になった場合でも，全ての頂点に色が塗られていない可能性がある．よって処理の最後に，未だ塗られていない頂点は，その隣接の色と同色にならない最小の色で塗る処理も追加する．これを修正 Cuthill-McKee (MCM) と呼び，図 4 に，MCM を用いた色付けの例を示す．先ほどの CM とは異なり，修正を加えたことで頂点 7, 10 に色 2 が塗られていないことが確認できる．またこの例はキューが空になった後に，未だ塗られていない頂点を持つ (図 4 (b))．この頂点を隣接と同色でない色で塗る処理の際に，定めた色数で塗れない場合は，色数を増やしてカラーリングする．例では 3 色では塗りきれず，4 色目を用いている．MCM は，MC が解決できなかった依存関係を解決し，並列性を抽出するカラーリングである．

2.3 Algebraic Multicolor

Algebraic Multicolor (AMC) は，2002 年に岩下と島崎により提案され，それぞれの色が持つ頂点数のバランスが良い性質を持つ [11]．Algorithm3 に，AMC アルゴリズムを示す．また図 5 に，AMC を用いた色付けの例を示す．

Algorithm 3 Algebraic Multicolor

(1) Let i-color be 0. Set the graph-size as n and the number of colors as C.
(2) Iterate from 1 to n with j.
  (a) Check the colors of the adjacency vertices of vertex j. If there is the same color as i-color, then let i-color be (i-color + 1) mod C and repeat (a).
  (b) Assign i-color to vertex j.



(a) Color-loop 1 (b) Color-loop 3 (c) Color-loop 5

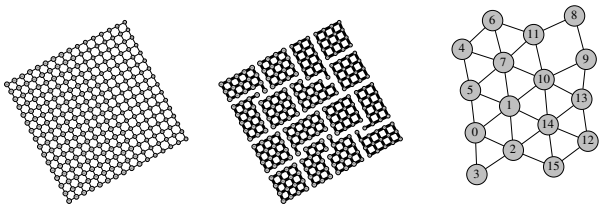
図 5 Algebraic Multicolor coloring process for a graph. Coloring process proceeds (a) to (c).

頂点番号順に走査し，一色一頂点ずつ塗る処理を繰り返すため，各色の頂点数の極端な不均衡はなくなり，なるべくバランスを良くする．図 5 の各色の頂点数は，[3, 4, 3, 3, 3] となり，その性質が伺える．指定した色数 C で塗りきれない場合は，塗られていない頂点が無くなるまで C を増やしてカラーリングを行う．

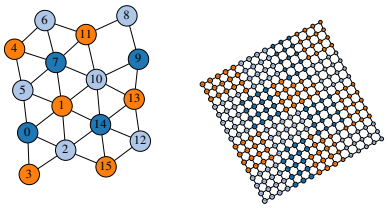
3. Block Multicolor Gauss-Seidel

古典的定常反復法として挙げられる Gauss-Seidel は収束性が高いものの，更新順序が依存関係を持つため，そのまま並列化を行うことは難しい．依存関係を解決するために，カラーリング情報を用いて更新順序を決める Multicolor (MC) が提案されており，この手法は同色内で並列実行が

可能である．しかし Multicolor の並列性抽出による収束性の悪化が問題になる場合がある．これを解決するために，複数の頂点を一つのブロックにまとめるブロッキングを取り入れた Block Multicolor (BMC) がある．BMC は各ブロック内の依存関係は維持されるため収束性が向上するだけでなく，キャッシュの恩恵を受けて高速になることが期待される．図 6 に二次元のグリッドグラフに対する BMC の処理フローを示す．まず始めに，与えられたグラフを，



(a) A 16 by 16 grid graph (b) Divide the graph into 16 parts (c) A blocked graph based on (b)



(d) Apply the coloring algorithm (e) Assign the color based on (d)

図 6 Block Multicolor for a 16 by 16 grid.

グラフ分割ライブラリを用いて分割する．次に，分割された一つの領域を一つのブロック頂点とみなして，ブロック化グラフを構築する．ブロック化グラフに対してカラーリングアルゴリズムを適用し，そのブロック化グラフの色付けを用いて，各ブロックに属する頂点を同色で塗る．

3.1 Multiplicative Schwartz type Gauss-Seidel

BMC-GS の各ブロックが実行環境のキャッシュサイズより十分小さい場合，一度ブロックに対して GS を行うと，更新するために参照した情報はキャッシュに収まっていると仮定できる．また並列性を十分抽出した BMC-GS は，ブロック内の更新処理よりも隣接通信が支配的な処理である場合がある．この状況を利用して各ブロック内の GS を複数回繰り返す乗法シュワルツ型の BMC-GS (MS-BMC-GS) が提案されている．MS-BMC-GS は，通常の BMC-GS と同じ並列性を持ち，キャッシュヒット率を高め，より高い収束性を実現する．

4. BMC-SGS 前処理付き CG

本研究では BMC-GS を CG 法の前処理として用いる．CG 法は正定値対称行列に対する Krylov 部分空間法であるため，前処理も対称性を持つ必要がある．そのため対称な smoother である BMC-Symmetric GS (BMC-SGS) を用いる．BMC-SGS は，BMC-GS の処理後，更新順番を逆にした BMC-GS を行う方法を指す．Algorithm4 に前処理付き CG 法のアルゴリズムを示す．前処理部分は 2 行目と 8 行目であり，$\mathbf{z}_k$ の初期値を $\mathbf{0}$ として，$A\mathbf{z}_k = \mathbf{r}_k$ に BMC-SGS を行なった近似解を用いる．この手法を BMC-SGS-CG と記述する．

Algorithm 4 preconditioned CG

1: $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$
2: $\mathbf{z}_0 = M^{-1}\mathbf{r}_0$
3: $\mathbf{p}_0 = \mathbf{z}_0$
4: **for** $k = 0$ to maxiter **do**
5: $\quad \alpha_k = \frac{\mathbf{r}_k^T \mathbf{z}_k}{\mathbf{p}_k^T A \mathbf{p}_k}$
6: $\quad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
7: $\quad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A \mathbf{p}_k$
8: $\quad \mathbf{z}_{k+1} = M^{-1}\mathbf{r}_{k+1}$
9: $\quad \mathbf{b}_k = \frac{\mathbf{z}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{z}_k^T \mathbf{r}_k}$
10: $\quad \mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k$
11: **end for**

CG 法の収束性は問題行列の固有値分布に依存する．前処理の目的は固有値分布を改善しソルバの収束性を高めることである．ここでは BMC-SGS の反復行列と前処理行列を明示的に導出し，前処理後の行列を求め，実際に固有値分布が改善するかを確認する．

Gauss-Seidel は係数行列 $A \in \mathbb{R}^{n\times n}$ を LDU 分解を行ない，$(L+D)$ について解く手法であり，式 3 で表される．L と $U \in \mathbb{R}^{n\times n}$ はそれぞれ，A の厳密な下三角行列と上三角行列であり，$D \in \mathbb{R}^{n\times n}$ は A の対角行列である．

$$A = L + D + U \tag{1}$$
$$A\mathbf{x} = \mathbf{b} \tag{2}$$
$$\mathbf{x}_{k+1} = -(L+D)^{-1}U\mathbf{x}_k + (L+D)^{-1}\mathbf{b} \tag{3}$$

BMC-GS は色順にオーダリングされた行列に対する Gauss-Seidel と考えると，色 i ごとの反復行列と前処理行列の乗法系で記述することができる．各色の反復式は次式となる．

$$A_i\mathbf{x} = (I - I_i)\mathbf{x} + I_i\mathbf{b} \tag{4}$$
$$\mathbf{x}_{k+1} = (L_{A_i} + D_{A_i})^{-1}U_{A_i}(I - I_i)\mathbf{x}_k + (L_{A_i} + D_{A_i})^{-1}I_i\mathbf{b} \tag{5}$$
$$\mathbf{x}_{k+1} = (D_{A_i} + U_{A_i})^{-1}L_{A_i}(I - I_i)\mathbf{x}_k + (D_{A_i} + U_{A_i})^{-1}I_i\mathbf{b} \tag{6}$$

$I \in \mathbb{R}^{n\times n}$ は単位行列，$I_i \in \mathbb{R}^{n\times n}$ は色 i に対応する対角要素にのみ 1 が，その他には全て 0 が入る行列である．$A_i \in \mathbb{R}^{n\times n}$ は色 i に対応する行が A と同じであり，その他の行には対角に 1 が入る行列である．さらに A_i に対する LDU 分解のそれぞれを $L_{A_i}, D_{A_i}, U_{A_i} \in \mathbb{R}^{n\times n}$ と記す．

式 5 は，各色に関して，未知数番号が小さい頂点から大きい頂点にかけて更新する反復式である．式 6 は，更新順番が逆の反復式であり，未知数番号が大きい頂点から小さい頂点にかけて更新する．上記より，色 0 から色 C にかけて式 5 で更新，その後に色 C から色 0 の式 6 で更新を行う BMC-SGS は，次で形でまとめることができる．

$$\mathbf{x}_{k+1} = M_{\text{BMC-SGS}}\mathbf{x}_k + N_{\text{BMC-SGS}}\mathbf{b} \tag{7}$$

CG 法は両側前処理を行うため，前処理行列 $N_{\text{BMC-SGS}}$ に対する Cholesky 分解 $N = L^T L$ を用いて，前処理後の行列は $L^T AL$ ととなる．図 7 に，係数がランダムに変化する 3 次元ポアソン問題に 7 点差分で離散化された行列に対して，ブロック数 $32, 64, 128$，WP カラーリングを用いた各領域で 2 回の SGS を行う MS-BMC-SGS 前処理の固有値分布を示す．縦軸は固有値の実数，横軸は固有値番号を表す．MS-BMC-SGS 前処理のブロック数の違いに注目するため，最大固有値は 18.69 である元の問題行列の固有値分布はプロットしていない．前処理は固有値が 1.0 に近いほど良い性能といえる．図 7 より，各種ブロック数が小さいほど，つまり 1 ブロック内の自由度が多いほど，固有値を 1.0 に集め固有値分布を改善したことを確認した．

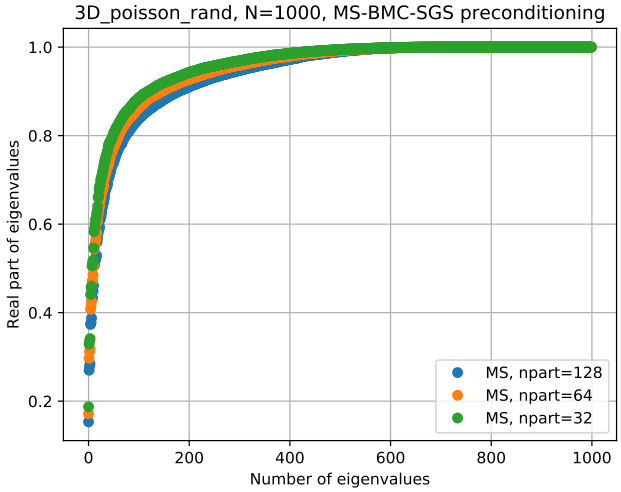


図 7　MS-BMC-SGS による固有値分布の改善

5. 数値実験

本章では，MS-BMC-SGS-CG の各種ブロック数と各カラーリングアルゴリズムによる影響分析を行う．分割された各領域で行う SGS の回数は 2 回とした．SuiteSparse Matrix Collection の行列サイズ 1000 以上かつ非ゼロ要素数 100 万以下の行列の中から，正定値対称かつ，SGS 前処理 CG 法が行列サイズの反復回数で収束した行列を評価対象とした．全ての問題で初期値ベクトル $\mathbf{x}_0$ を $\mathbf{0}$，右辺ベクトル $\mathbf{b}$ は $\mathbf{1}$ とした．ブロッキングのための領域分割には METIS Library を用いた．カラーリングは WP，MCM，AMC の三種類を用い，WP の色数を基準に MCM と ACM の色数を決定した．

5.1 ブロッキングの収束性分析

この実験では，ブロッキングによる収束性の改善を確認する．テスト行列に対して，ブロック数を 128, 256, 512, 1024, 2048 としたブロッキングを行ない，各カラーリングごとに反復回数の削減率を計測した．行列 Pres_Poisson(行数 14,822，非ゼロ要素数 715,804) に適用した結果を図 8 に示す．縦軸は反復回数を表し，横軸は各カラーリングのブロック数と色数を表す．青色は WP，緑色は MCM，橙色は AMC である．各カラーリングとも，ブロック数を減らすとともに，つまりブロック内の自由度を増やすとともに，反復回数を削減した．

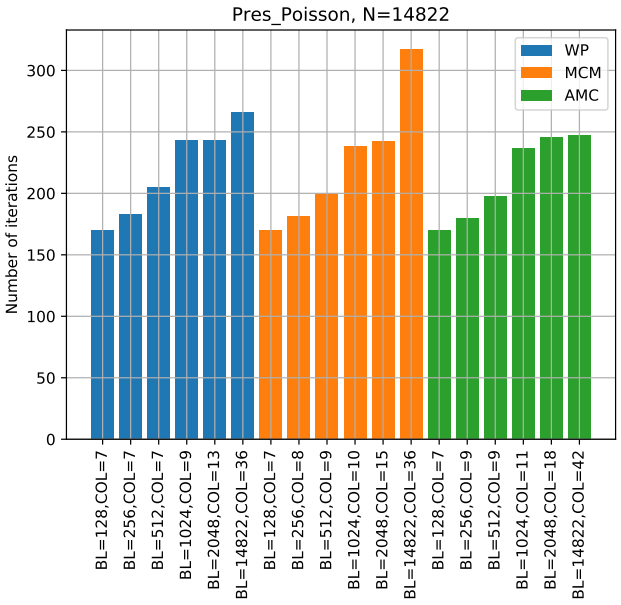


図 8　行列 Pres_Poisson に対する MS-BMC-SGS-CG の反復回数

次に，テスト行列全てに各ブロッキング・カラーリングを適用し，非ブロッキングに対する反復回数の削減率を図に示す．最大で行列 aft01 に対して約 58%削減，また約 8 割の行列に対して 20%以上削減し，ブロッキングによる収束性の向上を確認した．

5.2 カラーリングの収束性分析

この実験ではカラーリングの影響に焦点をあてるため，非ブロッキングの MS-MC-SGS-CG の反復回数を注目する．図 10 に，行列 bcsstk13 (行列数 2,003，非ゼロ要素数 83,883) に対して各カラーリングを適用した際の，各色ごとの自由度分布を示す．横軸は各色を表し，縦軸は各色で塗られた未知数の数を表し，上から順番に WP，MCM，ACM となっている．WP は依存関係が無い箇所を全て塗る操作を最初の色から繰り返すため，色番号が増えるにつれて各

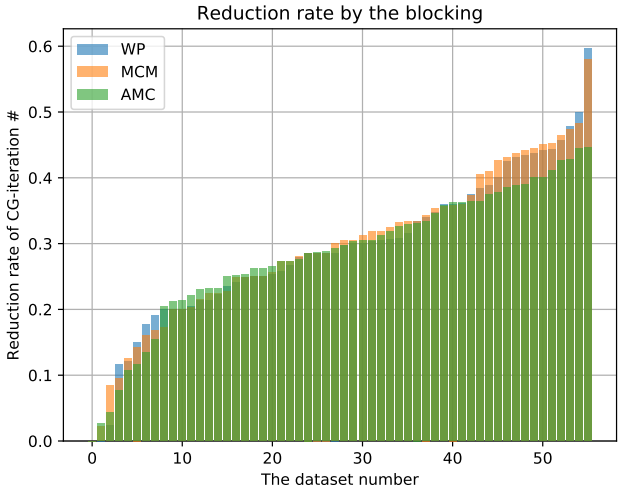


図 9　ブロッキングによる BMC-SGS-CG の最大削減率

色が持つ自由度が少なくなっている．MCM は幅優先探索を行うため，色番号が増えるにつれて各色が持つ自由度が増加し，中盤から減少している．ACM はどの色もバランスよく自由度を持っていることが特徴的である．この分布

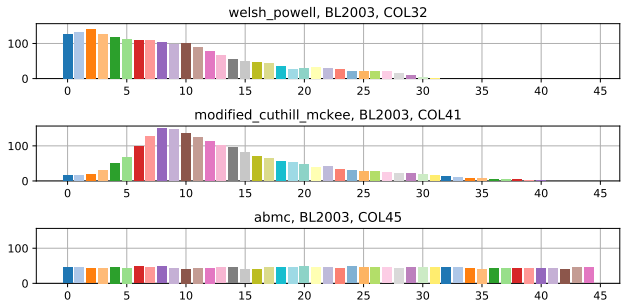


図 10　行列 bcsstk13 に対する BMC-SGS の色付け分布

は各カラーリングごとに異なり，この違いがどのように収束性に影響するかを分析する．全テスト行列を行列サイズ N に応じて $N \leq 5000$，$5000 < N \leq 10000$，$10000 < N$ の 3 クラスに分類し，最も反復回数が少ない各カラーリングのケース数を表 1 に示す．表 1 より，ACM が行列サイズが増えるに従い，最も反復回数が少ないケースの割合が 29%，56%，70%と増加し，高い収束性を確認した．よって，行列サイズがより大きい場合に，他カラーリング法に比べて，ACM が収束性が高いと推察される．

表 1　全テスト行列 56 個における色数を同程度に揃えたカラーリングアルゴリズムの収束性

	WP	MCM	ACM
$N \leq 5000$: 24 cases	9(38%)	13(54%)	7(29%)
$5000 < N \leq 10000$: 9 cases	5(56%)	2(22%)	5(56%)
$10000 < N$: 23 cases	9(39%)	8(35%)	16(70%)

5.3　MS-BMC-SGS-CG の計算コスト見積もり

本章では，MS-BMC-SGS-CG の並列環境における計算量を見積もり，その大小でブロッキングとカラーリングを議論する．最も多くの未知数を持つコアが，反復が終了するまでに未知数を参照した回数の近似を式 8 に示す．この式は 3 項で構成される．第 1 項の 1 ブロック内の未知数を表す．npart は領域分割数で，METIS を用いて分割しているため，平均的に分割されることを仮定している．第 2 項は各コアに割り振られる各色の自由度の最大値である．parallelism は並列度を表す．第 3 項は MS-BMC-SGS-CG の反復回数である．

$$\frac{N}{\text{npart}} \cdot \sum_{i=0}^{\text{colors}} \left\lceil \frac{\text{DoFs of color}i}{\text{parallelism}} \right\rceil \cdot \text{iterations} \qquad (8)$$

本研究で扱う各領域で 2 回の SGS を行う MS-BMC-SGS は，2 回目の SGS をキャッシュヒット率を高める手法である．そのため，各ブロックの自由度に対してキャッシュが十分大きいと仮定をおいて，通常の BMC-SGS と同等の計算量と見なしてコストを計算した．

行列サイズ $N > 10000$ 以上のテスト行列に対して，各ブロッキング・カラーリングによる MS-BMC-SGS-CG の計算コストを見積もり，並列度 8，32，128，512 における最もコストが小さかったケースの累積を表 2~5 に示す．並列度 8 の場合の表 2 では，非ブロッキングのコストが最小にならず，全ての行列が各分割数でブロッキングを行う手法がコストを最小にした．表 2~ 表 5 より，並列度を挙げるに従い，よりブロック数が多い，つまり 1 ブロック内の自由度が少ないブロッキングが最小コストとなった．これは表内のケース累積数が右にシフトしていく様子に対応する．少ないブロック数により実行並列度での並列性抽出ができない場合は最小コストにならず，実行並列度の並列性が十分抽出できるより少ないブロック数によるブロッキングが最小コストになることを表す．収束性と並列性のバランスが取れたブロッキングが最適であることを示す一例を確認した．

我々が興味があるのは各問題に対して，どのような設定

表 2　並列度 8: 計算コストが最小となった各ブロッキング・カラーリングにおけるのケース数

\ npart	128	256	512	1024	2014	N	sum(rate)
WP	2	5	3	1	1	0	12(52%)
MCM	0	3	4	2	1	0	10(43%)
AMC	2	4	1	0	1	0	8(35%)

表 3　並列度 32: 計算コストが最小となった各ブロッキング・カラーリングにおけるのケース数

\ npart	128	256	512	1024	2014	N	sum(rate)
WP	0	4	1	1	5	1	12(52%)
MCM	0	2	0	3	4	0	9(39%)
AMC	0	5	1	3	3	0	12(52%)

表 4　並列度 128: 計算コストが最小となった各ブロッキング・カラーリングにおけるのケース数

\ npart	128	256	512	1024	2014	N	sum(rate)
WP	0	0	0	3	5	5	13(57%)
MCM	0	0	0	0	3	0	3(13%)
AMC	0	0	0	4	4	3	11(48%)

表 5　並列度 512: 計算コストが最小となった各ブロッキング・カラーリングにおけるのケース数

\ npart	128	256	512	1024	2014	N	sum(rate)
WP	0	0	0	0	2	7	9(39%)
MCM	0	0	0	0	0	4	4(17%)
AMC	0	0	0	0	0	12	12(52%)

でソルバを実行するかである．多くの場合で解きたい問題と実行環境は決まっているため，行列サイズ N と実行最大並列数 P は固定と仮定すると，未知の設定はブロッキングサイズとカラーリングアルゴリズムである．カラーリングには，5.2 章より大きな問題に対しても収束性の面で優れると推察される AMC を用いる．最後はブロッキングサイズを決める．5.1 章より，例外はあるものの多くの場合でブロックサイズを大きくするほどソルバの収束性向上を確認した．さらに AMC の場合，各色に対する自由度分布がバランス良い性質を持つため，各色が持つ自由度を $\frac{\text{npart}}{\text{色数}}$ と近似できる．これより，$\frac{\text{npart}}{\text{色数}} < P$ を満たす最大の npart が，収束性と並列性の双方において良いブロッキングであり，最も計算コストが低いと推定される．行列 Pres_Poisson に対してこの予想を確かめる．ACM より各ブロック化されたグラフに対して $\frac{\text{npart}}{\text{色数}}$ を求め，それを超える並列度で見積もった計算コストを表 6，7 に示す．赤文字は計算コストが最小のケースを表す．表 6，7 より，各並列度において $\frac{\text{npart}}{\text{色数}} < P$ を満たす最大の npart が選ばれることを確認した．また各ブロッキングの列に注目すると，並列度をあげる一方で計算コストの削減は頭打ちになることが確認できる．これは $\frac{\text{npart}}{\text{色数}}$ 以上の並列性は抽出できていないことを示す．Pres_Poisson 以外のテスト行列においても同様の結果が得られることを確認した．

表 6　行列 Pres_Poisson に対する各並列度における計算コスト (1)

P \ npart	128	256	512
20	1.38e+5	1.88e+5	1.60e+5
30	1.38e+5	1.04e+5	1.09e+5
60	1.38e+5	9.38e+4	5.73e+4
100	1.38e+5	9.38e+4	5.16e+4
120	1.38e+5	9.38e+4	5.16e+4
360	1.38e+5	9.38e+4	5.16e+4
$\frac{\text{npart}}{\text{色数}}$	18.29	28.44	56.89

表 7　行列 Pres_Poisson に対する各並列度における計算コスト (2)

P \ npart	1024	2048	Nonblocking
20	1.89e+5	1.92e+5	1.87e+5
30	1.44e+5	1.28e+5	1.25e+5
60	7.55e+4	6.41e+4	6.32e+4
100	3.77e+4	6.41e+4	4.15e+4
120	3.77e+4	3.20e+4	3.21e+4
360	3.77e+4	3.20e+4	1.14e+4
$\frac{\text{npart}}{\text{色数}}$	93.09	113.78	352.90

6. おわりに

本研究では CG 法に対する MS-BMC-SGS 前処理のブロッキングとカラーリングの影響分析を行なった．SuiteSparse Matrix Collection の行列サイズ 1000 以上かつ非ゼロ要素数 100 万以下の行列の中から，正定値対称かつ，SGS 前処理 CG 法が行列サイズの反復回数で収束した行列を評価対象とした．ブロッキングサイズを大きくするに従い，我々のテスト行列の約 8 割で収束性を 2 割以上向上させ，ブロッキングの有効性を確認した．またカラーリングアルゴリズムに WP，MCM，AMC を用いて反復回数を比較し，行列サイズが大きいほど AMC がテスト問題中の最小反復回数である割合が増加することを確認した．これより，AMC がより大きな行列サイズに対して収束性が高いと考えられる．

並列環境における 1 プロセスの計算コストを見積もり，収束性と並列性を評価した．並列数を増やすに従って，計算コストが最小となるブロッキングサイズは小さくなり，収束性と並列性のバランスが取れたブロッキングが良いとわかった．これより，事前に複数のブロッキングサイズを試し，実行環境の並列数を固定し，$\frac{\text{ブロック数}}{\text{色数}}$ 以下である最大のブロック数を用いた設定について分析を行なった．上記の評価に対して，AMC かつ上記を満たす最大ブロック数が最小コストになる例を確認した．

今後の課題として，Multigrid 法への拡張と，実環境での並列分散実装が挙げられる．通常 MS-BMC-SGS は Multigrid 法の smoother として用いられる．本研究では smoother 単体による前処理の評価だったため，MS-BMC-SGS を smoother とした Multigrid 前処理に対しての評価が必要である．また本研究での計算コストの見積もりは，通信コストやキャッシュの影響を含んでいないため，実環境において並列分散実装を行なう必要がある．MS-BMC-GS を smoother とする MGCG をソルバとした，より大規模な問題に対する並列分散環境での評価を今後の課題として研究を続けていく．

謝辞　理化学研究所計算科学研究センターの河合直聡氏に多くの有益な助言をいただきました．ここに感謝の意を表します．This work is supported by “Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures” and “High Performance Computing Infrastructure” in Japan (Project ID: jh180022-NAHI).

参考文献

[1] Adams, L. and Jordan, H.: Is SOR Color-Blind?, *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 2, pp. 490–506 (online), DOI: 10.1137/0907033 (1986).

[2] Adams, M., Brezina, M., Hu, J. and Tuminaro, R.: Parallel multigrid smoothing: polynomial versus Gauss-Seidel, *Journal of Computational Physics*, Vol. 188, No. 2, pp. 593–610 (online), DOI: 10.1016/S0021-9991(03)00194-3 (2003).

[3] Bell, W. N., Olson, L. N. and Schroder, J.: PyAMG: Algebraic Multigrid Solvers in Python (2013). Version 2.1.

[4] Block, U., Frommer, A. and Mayer, G.: Block colouring schemes for the SOR method on local memory parallel computers, *Parallel Computing*, Vol. 14, No. 1, pp. 61 – 75 (online), DOI: https://doi.org/10.1016/0167-8191(90)90096-R (1990).

[5] Briggs, W. L., Henson, V. E. and McCormick, S. F.: *A Multigrid Tutorial: Second Edition*, Society for Industrial and Applied Mathematics (2000).

[6] Cuthill, E. and McKee, J.: Reducing the Bandwidth of Sparse Symmetric Matrices, *Proceedings of the 1969 24th National Conference*, ACM '69, New York, NY, USA, ACM, pp. 157–172 (online), DOI: 10.1145/800195.805928 (1969).

[7] Davis, T. A. and Hu, Y.: The University of Florida Sparse Matrix Collection, *ACM Trans. Math. Softw.*, Vol. 38, No. 1, pp. 1:1–1:25 (online), DOI: 10.1145/2049662.2049663 (2011).

[8] de Recherche en Informatique et en Automatique, I. N., Doi, S. and Lichnewsky, A.: *A Graph-theory Approach for Analyzing the Effects of Ordering on ILU Preconditioning*, Rapports de recherche, Institut National de Recherche en Informatique et en Automatique (1991).

[9] Doi, S. and Washio, T.: Ordering Strategies and Related Techniques to Overcome the Trade-off Between Parallelism and Convergence in Incomplete Factorizations, *Parallel Comput.*, Vol. 25, No. 13-14, pp. 1995–2014 (online), DOI: 10.1016/S0167-8191(99)00064-2 (1999).

[10] Iwashita, T., Nakashima, H. and Takahashi, Y.: Algebraic Block Multi-Color Ordering Method for Parallel Multi-Threaded Sparse Triangular Solver in ICCG Method, *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, pp. 474–483 (online), DOI: 10.1109/IPDPS.2012.51 (2012).

[11] Iwashita, T. and Shimasaki, M.: Algebraic multicolor ordering for parallelized ICCG solver in finite-element analyses, *IEEE Transactions on Magnetics*, Vol. 38, No. 2, pp. 429–432 (online), DOI: 10.1109/20.996114 (2002).

[12] Iwashita, T. and Shimasaki, M.: Block Red-Black Ordering: A New Ordering Strategy for Parallelization of ICCG Method, *International Journal of Parallel Programming*, Vol. 31, No. 1, pp. 55–75 (online), DOI: 10.1023/A:1021738303840 (2003).

[13] Karypis, G. and Kumar, V.: *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices* (1998).

[14] Kawai, M., Ida, A. and Nakajima, K.: Hierarchical Parallelization of Multi-coloring Algorithms for Block IC Preconditioners, *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 138–145 (online), DOI: 10.1109/HPCC-SmartCity-DSS.2017.18 (2017).

[15] Kawai, M., Iwashita, T. and Nakashima, H.: SIMD Implementation of a Multiplicative Schwarz Smoother for a Multigrid Poisson Solver on an Intel Xeon Phi Coprocessor, *High Performance Computing for Computational Science – VECPAR 2014* (Daydé, M., Marques, O. and Nakajima, K., eds.), Cham, Springer International Publishing, pp. 57–65 (2015).

[16] Poole, E. L. and Ortega, J. M.: Multicolor ICCG Methods for Vector Computers, *SIAM Journal on Numerical Analysis*, Vol. 24, No. 6, pp. 1394–1418 (online), available from ⟨http://www.jstor.org/stable/2157341⟩ (1987).

[17] Tatebe, O.: The Multigrid Preconditioned Conjugate Gradient Method, Copper Mountain, CO, April 4-9, In 6th Copper Mountain Conference on Multigrid Methods (1993).

[18] van der Vorst, H. A.: *Iterative Krylov Methods for Large Linear Systems*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press (2003).

[19] Welsh, D. J. A. and Powell, M. B.: An upper bound for the chromatic number of a graph and its application to timetabling problems, *The Computer Journal*, Vol. 10, No. 1, pp. 85–86 (online), DOI: 10.1093/comjnl/10.1.85 (1967).

[20] Yavneh, I.: Why Multigrid Methods Are So Efficient, *Computing in Science Engineering*, Vol. 8, No. 6, pp. 12–22 (online), DOI: 10.1109/MCSE.2006.125 (2006).