Compas'2022 : Parallélisme / Architecture/ Système
MIS/UPJV - Amiens France, 5-8 juillet 2022

A multi-coloring ordering for preconditioned Krylov solvers

Aboul-Karim MOHAMED EL MAAROUF *, Luc GIRAUD, Abdou GUERMOUCHE, Thomas GUIGNON

Université de Bordeaux, Laboratoire Bordelais de Recherche en Informatique, Inria Bordeaux Sud-Ouest, IFPEN

Résumé
The trend of processor hardware evolution shows an increasing support of fine grain parallelism via SIMD vector instruction sets and hardware threading. We move from small SIMD vectors (x86 SSE2) to much larger vectors (x86 AVX512). In addition, all processors can handle at least two hardware threads. For example, the Intel KNL processor (AVX512) can handle 8 double precision vector sizes per hardware thread and up to 4 hardware threads. This provides a potential fine grain parallelism of degree 32, which becomes twice larger for single precision floating point arithmetic calculation. Also, the new ARM SVE instruction set potentially allows for hardware implementation up to 32 double precision SIMD vector sizes per hardware thread.
In this work, we are interested in using this fine grain parallelism for the implementation of iterative sparse linear system solvers based on Krylov subspace methods preconditioned with incomplete LU factorizations, mainly ILU0. For these calculations, the triangular solves involved when applying the preconditioner is a major computation kernel. Therefore, it is important to efficiently use fine grain parallelism in these sparse triangular solutions. One possible approach, that has been used for long time, consists in using MULTI-COLORING (MC) of the matrix adjacency graph which allows us to identify independent equations in the reordered matrix that can be solved simultaneously. This computational efficiency often comes at a cost of a numerical deterioration of the preconditioner quality that leads to a slower convergence of the iterative solvers. However, another ordering called REVERSE CUTHILL-MCKEE (RCM) often enhances the convergence of the numerical solver but does not naturally provide fine-grained parallelism for ILU0 calculations or the solution of sparse triangular systems. In order to alleviate the MC numerical penalty, we propose an algorithm that performs a matrix reordering by combining RCM ordering and MC called COLORRCM. The goal of the resulting ordering is to exhibit enough fine-grained parallelism to feed the computing units (SIMD units) and eventually hardware threading, while improving the convergence of the Krylov solver compared to the classical MC ordering. We investigate the performance of the novel algorithm on a large set of matrices from the SuiteSparse Matrix Collection as well as on a suite of matrices extracted from IFPEN porous media flow simulations. The COLORRCM approach shows an improvement of the ILU0-BICGSTAB convergence compared to the MC ordering.

Mots-clés : Graph multi-coloring, Reverse Cuthill-McKee, Ordering, Incomplete factorization

1. Introduction

To be able to simulate large and complex phenomena, we need a lot of computational power. This power is mainly related to the number of computational units that can work at the same time and the frequency with which they are clocked. Since the 90's with the advent of supercomputers with vector processors, advanced techniques to vectorize intensive computations, such as Gaussian elimination, have been intensively studied [10]. Due to the data dependencies in computations, the methods based on Gaussian elimination approach are hard to vectorize without having a prior knowledge of the problem

*. **Corresponding email** : aboul-karim.mohamed-el-maarouf@ifpen.fr

to be solved. This issue is still relevant today because modern processors are becoming more and more sophisticated with larger and larger register sizes. The trend of processor hardware evolution shows an increasing support of fine-grained parallelism via SIMD vector instruction sets and hardware threading. We move from small SIMD vectors (x86 SSE2) to much larger vectors (x86 AVX512). A good strategy for exploiting fine-grained parallelism is to group the unknowns with no direct relationship between each other and order them first to obtain blocks of independent unknowns. This approach can be recast into a problem of graph coloring, that is encountered in many different scientific computing applications. In our case, the graph, referred to as the adjacency graph of the sparse matrix, is defined as follows. Let $\mathbf{A} \in \mathbf{R}^{n\times n}$ be a structurally symmetric matrix, i.e., if $a_{ij} \neq 0$ then $a_{ji} \neq 0$ too. We can define its adjacency graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the vertices $\mathcal{V} = \{v_0, v_1, ..., v_{n-1}\}$ are associated with the matrix rows (or columns) and there exists an edge in $\mathcal{E}$ between v_i and v_j if $a_{ij} \neq 0$. Finding independent unknowns in the linear system reduces to finding vertices in $\mathcal{G}$ that are not connected and changing the ordering of the vertices of $\mathcal{G}$ translates into applying a symmetric permutation to $\mathbf{A}$. Preprocessing the system by applying a reordering method can bring many advantages, like robustness of the preconditioner or enabling parallelism in the computation. So, instead of solving :

$$\mathbf{Ax} = \mathbf{y} \tag{1}$$

we rather solve

$$\mathbf{PAP}^T\mathbf{z} = \mathbf{Py}, \quad \mathbf{x} = \mathbf{P}^T\mathbf{z} \tag{2}$$

where $\mathbf{P}$ is a permutation matrix associated with the chosen reordering, so that $\mathbf{PAP}^T$ exhibits better structure for the parallelism and its incomplete factorization will be more stable than the one applied directly to $\mathbf{A}$. Reordering techniques are widely used in numerical linear algebra, both for direct and iterative solvers. In particular, to find fine-grained parallelism, graph coloring methods are commonly used. The main challenge of the parallelization of the preconditioned Krylov solver is that the preconditioners are sensitive to the ordering equations. Before trying to parallelize the computations, we must make sure that our methods do not deteriorate the convergence of the solver. For this reason, we focus in this work on the effect of multi-coloring methods on the numerical convergence of the Krylov solver, and we propose a multi-coloring (MC) method based on the RCM approach which improves the numerical convergence of the Krylov solver preconditioned with ILU0. The rest of the paper is organized as follows : Section 2 presents a state of art of the ordering techniques and their effect on incomplete factorization. The first two parts of Section 3 are devoted to the two orderings on which our work is based, RCM and MC. In the last part of Section 3 we describe our graph color algorithm called ColorRCM. Section 4 contains the numerical results and we discuss the impact of ColorRCM in the convergence of the iterative solver.

2. Related work

Originally, the ordering methods were mainly studied in the context of sparse factorizations for direct methods since it is a way to reduce the fill-in that appears in the triangular factors. Ordering methods like Cuthill-McKee (CM) [6] and Reverse Cuthill-McKee (RCM) [13], Sloan [19] or Gibbs-Poole-Stockmeyer (GPS) [14] are known for their ability to reduce the bandwidth and the profile of the computed factors. Other methods like Minimum Degree (MD), Nested Dissection (ND) or Red-Black (a graph multi-coloring method with two colors), which are efficient in minimizing the fill-in in the exact factorization of matrices, have also been widely analyzed. The impact of the orderings in the preconditioned subspace method was first studied in [11] for symmetric positive definite (SPD) matrices. The authors reported that the use of orderings like MD, ND or Red-Black for an incomplete Cholesky factorization without fill-in (IC0) sometimes strongly degrades the convergence of the Conjugate Gradient solver (CG). According to [11] observation, RCM is the ordering that degrades the least, and sometimes it is even better than the original order of the matrix. In many studies for non SPD cases, RCM is often the ordering that improves the most the convergence of the Krylov solver [2, 3, 5, 17]. The authors in [3] recommend it as the default ordering with an ILU if we do not have any a prior knowledge of the problem to be solved. However, RCM does not provide naturally a parallelism for the computation of the ILU. The effect of multi-coloring methods on iterative solvers for structured grid methods has been

widely studied since the study in [11] which points out the link between the convergence of the Conjugate Gradient solver preconditioned with IC0 (IC0-CG) and the degree of parallelism. It seems that the more parallelism we have, the more we degrade the convergence of the solver [11]. Many reasons may explain this degradation, or even the failure of convergence. In particular, the instability of the factors, which could be caused by too small pivots or ill-conditioned factors, or simply when there have been too many value drops during the incomplete factorization. The first two points are often symptomatic of non-diagonal dominant matrices. For more details, we refer the readers to [1, 5, 3]. However, in the case of ILU0 or IC0 preconditioning, a good degree of parallelism can be achieved through MC methods. In the MC methods, groups of vertices that are not connected by an edge can be assigned the same color, and the vertices of the same color can be scheduled to be solved simultaneously. In a similar fashion, orderings such as LEVEL SCHEDULING or ND [12] are used to find vertices or sets of vertices that may be factorized in parallel. The idea of LEVEL SCHEDULING is to identify vertices that do not have any incoming edges and place them together at the same level. Those vertices can be factorized concurrently. In some case, MC provides a higher degree of parallelism than LEVEL SCHEDULING. More recently, a novel approach to compute the ILU has demonstrated a fine-grained parallelism [4]. Their method is an approximation of the factors **L** and **U** by solving in "sweeps" a system of nonlinear equations for each nonzero element. The authors have shown that their method needs a few sweeps to provide an efficient ILU preconditioner. Lastly, [16] proposed the Algebraic Block Multi Coloring (BMC) method for multi-threaded parallelization of triangular resolution in IC0-CG. In [15] the Hierarchical Block Multi Color method for SIMD parallelization of the triangular solver in IC0-CG showed better numerical performance than BMC while both methods are equivalent. However, the Hierarchical Block Multi Color method still strongly degrades the convergence of the IC0-CG solver. Very few algorithms have been developed allowing fine-grained parallelization of computations with MC while providing a numerically efficient ILU factorization like RCM. Although the effects of orderings on Krylov iterative solvers have been widely studied, most of the related work has been done on IC0-CG solver. The motivations sustaining this work are to reduce the negative impact of the MC methods on the number of iterations of the ILU0-BICGSTAB solver while providing enough fine-grained SIMD parallelism.

3. The orderings

3.1. Greedy multi-color graph ordering : MC

The idea of graph coloring is an assignment of colors to the vertices of the graph $\mathcal{G}$, such that no two adjacent vertices have the same color. The consequence is that the matrix resulting from the multi-color ordering will have diagonal blocks which are diagonal sub-matrices. The diagonal blocks represent the color classes, which are computed as follows :

1. $\mathcal{C}_j = \{v \in \mathcal{V} \mid \mathtt{color}(v) = j\}$, where $\mathtt{color}(v) = \min(c \geq 0 \mid c \neq \mathtt{color}(u), \forall u \in \mathtt{Adj}(v))$

In other words, finding a coloration of $\mathcal{G}$ is finding a partition $\{\mathcal{C}_0, \mathcal{C}_1, ..., \mathcal{C}_{\mathtt{ncolor}-1}\}$ of $\mathcal{V}$, such that the vertices of each $\mathcal{C}_l$ are independent from each other. Usually the graph coloring problem try to find the chromatic number $\chi(\mathcal{G})$, i.e, the minimum number of colors. The problem of finding $\chi(\mathcal{G})$ may not guarantee the balance of color, and usually there is a strong imbalance between the first and the last colors (see Figure 1c).

3.2. REVERSE CUTHILL-MCKEE(RCM)

Incomplete factorization is sensitive to the ordering of the unknowns. RCM is referenced as one of the orderings that can provide a better preconditioner [13]. The RCM algorithm can be described as follows :

1. Choose an initial vertex u as the pseudo-peripheral node of the graph $\mathcal{G}$. $\mathcal{R} = \mathcal{L}_0 = \{u\}$
2. For $i = 1 : nlevel$
   $\mathcal{L}_i = \mathtt{Adj}(\mathcal{L}_{i-1}) \backslash \mathcal{R}$ and sort the elements of $\mathcal{L}_i$ in ascending order of their degree.
   $\mathcal{R} = \mathcal{R} \bigcup \mathcal{L}_i$.
3. Reverse the order of the vertices in $\mathcal{R}$.

Then the set $\mathcal{L}_i$ is just connected with $\mathcal{L}_{i-1}$ and $\mathcal{L}_{i+1}$. Therefore, the matrix coming from the RCM ordering will be a block-tridiagonal matrix, but the blocks do not necessarily have a particular structure.

3.3. New multi-coloring : COLORRCM

The aim now is to find a particular coloring of the graph that follows the RCM ordering. If we compute the coloring of the vertices in each set $\mathcal{L}_i$, we will get a permuted matrix with block-tridiagonals and diagonal sub-matrices inside the diagonal blocks. We call this ordering the COLORRCM. The order provided by the graph coloring often affects the numerical convergence of the solver negatively. The coloring of the vertices in the $\mathcal{L}_i$ sets cannot be performed randomly. Thus, the graph coloring will follow the RCM order to assign colors to the vertices. The COLORRCM algorithm can be described as follows :

1. Compute the level-set $\mathcal{L} = \bigcup_{0=i}^{nlevel} \mathcal{L}_i$ from RCM.
2. Compute $\mathcal{C}_{i,j} = \{v \in \mathcal{L}_i \mid color(v) = j\}, 0 \leq i < nlevel$,
   where $color(v) = \min(c \geq 0 \mid c \neq color(u), \forall u \in Adj(v) \bigcap \mathcal{L}_i)$

$\mathcal{C}_{i,j}$ is the color class j of the level i, i.e., the set of vertices in level i with the same color. This algorithm gives the structure shown in Figure 1d.

3.4. Graph coloring with restricted the size of color class

We want to use COLORRCM to vectorize the incomplete factorization computations and also the triangular resolution in the solver. The approach of simply minimizing the number of colors is commonly used to achieve fine-grained parallelism. This results in a large color size, i.e., many vertices in the colors, but with the disadvantage that often the last colors are very small compared to the first ones (Figure 1c). Furthermore, we need a degree of parallelism that corresponds to a multiple of the size of the SIMD units. The degree of parallelism is the number of operations that can be computed simultaneously by a processor during the parallel execution of an application. So, we need vectors of SIMD units size with independent elements from each other. To do this, a standard trick is to define a maximum color size, i.e., a maximum number of vertices that a color class cannot exceed. So, the graph coloring COLORRCM with the restricted color size, denoted COLORRCM(p), can be described as follows :

1. Compute the level-set $\mathcal{L} = \bigcup_{0=i}^{nlevel} \mathcal{L}_i$ from RCM.
2. Compute $\mathcal{C}_{i,j} = \{v \in \mathcal{L}_i \mid j = color(v) \text{ and } |\mathcal{C}_{i,j}| \leq p\}, \quad 0 \leq i < nlevel \text{ and } p \in \mathbb{N}$

Where p is the maximum number of vertices in the set $\mathcal{C}_{i,j}$, i.e., the size of color. The idea is simply to try to minimize the number of colors, but once a color class reaches the maximum number of vertices p set by the user, we switch to another color. This will have the effect of significantly increasing the number of colors. For our goal of reducing the deterioration of the convergence of the solver introduced by multi-coloring, increasing the number of colors is rather beneficial. Indeed, it increases the global dependence of the Gauss elimination [8] because the colors are dependent on each other. Figure 1d illustrates an example of a sparse matrix pattern which is ordered with COLORRCM(4) and a color size sets to 4 vertices. The same approach was applied with MC, noted MC(p). It is represented in the Figure 1e, MC(4).

4. Experimental results

In this section, we study the numerical convergence of the ILU0-BICGSTAB iterative method. For more information on the preconditioned BICGSTAB algorithm, we refer the readers to the following papers [18, 21, 22]. The solver was run with the following parameters : a tolerance error of 10E-4 and a maximum number of iterations set to 500. For this experimental work, we used 227 matrices structurally symmetric with one strongly connected component from the SuiteSparse Matrix Collection [7] and 124 matrices from different time steps of IFPEN's porous media flow simulator with SPE10 model 2 [20] thus providing a total of 351 matrices. The solutions of the problems were performed sequentially in order to observe the effect of the ordering on the convergence of the solver. To visualize the results, we used the performance profile tool [9]. This tool allows to have a general classification of the performances of the different algorithms tested on a given set of problems. The idea is to calculate for a given test the ratio between the iterations of each algorithm compared to the smallest iteration (called

the performance ratio of the number of iterations). This gives us the percentage of tests for each algorithm performance ratio. For example, in Figure 2a, the point (1 ; 0.55) on the RCM plot means that in 55% of the tests, RCM has reached the minimum number of iterations compared to other algorithms. In contrast, the point (1 ; 0.22) on MC plot indicates that it is only in 22% of the cases that MC has reached the minimum number of iterations. Thus, the Figures 2a and 2b can be understood as follows : the smaller the performance ratio of an algorithm, the higher the proportion of problems solved are, and the better the performance of the method is. The performance profile allows us to compare the preconditioned solver ILU0-BiCGStab with Initial Order (without ordering), RCM, ColorRCM(8), MC(8) and MC orderings in terms of their number of iterations over the set of test cases. ColorRCM(8), MC(8) and MC refer respectively to the coloring combined with RCM with a maximum number of 8 vertices per color, to the classical Multi-Coloring with a maximum number of 8 vertices per color and to the classical Multi-Coloring without any restriction of the number of vertices per color. We test for k = 8, which corresponds to the maximum capacity of a cache line for double precision values.

4.1. Effect of orderings

We compare the algorithms between them according to their performance ratio of their number of iterations. We can see in Figure 2a that ordering the matrix with RCM before solving the problem with ILU0-BiCGStab allows in more than 90% of the cases to converge faster than with the initial ordering of the matrix. However, ColorRCM(8) is also, in more than 70% of the cases, more efficient than the initial order. In contrast, MC is the one that most strongly degrades the convergence of the solver. Although MC(8) is slightly better than MC but still performs worse than ColorRCM(8), Figure 2a. With very small colors, like p=8, the matrix resulting from the permutation with MC(8) will be structurally close to the initial structure of the matrix. Thus, the preconditioner ILU0 with MC(8) will be slightly closer to ILU0 with initial order compared to ILU0 with MC without limit. This explains the improvement of the solver performance for MC(8) compared to MC. This degradation phenomenon will be more pronounced if the initial order is already a "good" order. In Figure 2b, the initial order of SPE10 coming from the IFPEN simulator already has a certain preferential order. We can see that RCM is not performing better than the initial order of the matrices. It even seems to be the ordering which most degrades the convergence of the solver after MC. Also, MC(8) has almost the same performance as RCM. However, even if the gap is not as large as in Figure 2a, in Figure 2b ColorRCM(8) seems to perform better than the other orderings. For example, with a performance ratio of roughly 1.3, i.e. if we accept a degradation of the number of iterations of approximately 1.3 times more than the minimum number of iterations for ColorRCM(8), then we observe that in almost 95% of the tests, ColorRCM(8) is better than all other orderings. In contrast, MC, with its large color size, totally breaks the initial ordering and we can see that it has the worst performance. Finally, ColorRCM achieves the goals of our work by mitigating the negative impact of graph coloring on the solver convergence while providing a fine-grained parallelism for SIMD solver parallelization.

5. Conclusion

The aim of this work was to have an ordering that provides enough fine-grain parallelism for SIMD computational units while improving the numerical convergence of the solver compared to the MC ordering. Based on published studies on the effect of ordering on the convergence of Krylov type methods, the majority of them indicate that RCM often brings the best performance regarding the number of iterations. On the other hand, MC type algorithms tend to degrade the convergence of the solver compared to the natural order. Our results with BiCGStab preconditioned with ILU0 also go in this direction. The ColorRCM algorithm presented in Section 3.3 combines the RCM approach with MC and a parameter for the maximum color size. Bounding the size of the colors increases the number of colors, but ColorRCM often improves the numerical convergence of the solver compared to MC. Even if RCM remains the best ordering, ColorRCM has the advantage to bring a particular structure to the matrix allowing to exploit a fine grain parallelism for SIMD type computing units. In a future work, we will use ColorRCM ordering for ILU0 factorization and solving sparse triangular systems with AVX2 or AVX512. Moreover, the ColorRCM method as introduced in Section 3.3 can be partially parallelized. The computation of the RCM level-set is intrinsically sequential but the second step, the coloring of the vertices, can be done in parallel.

A. Figures

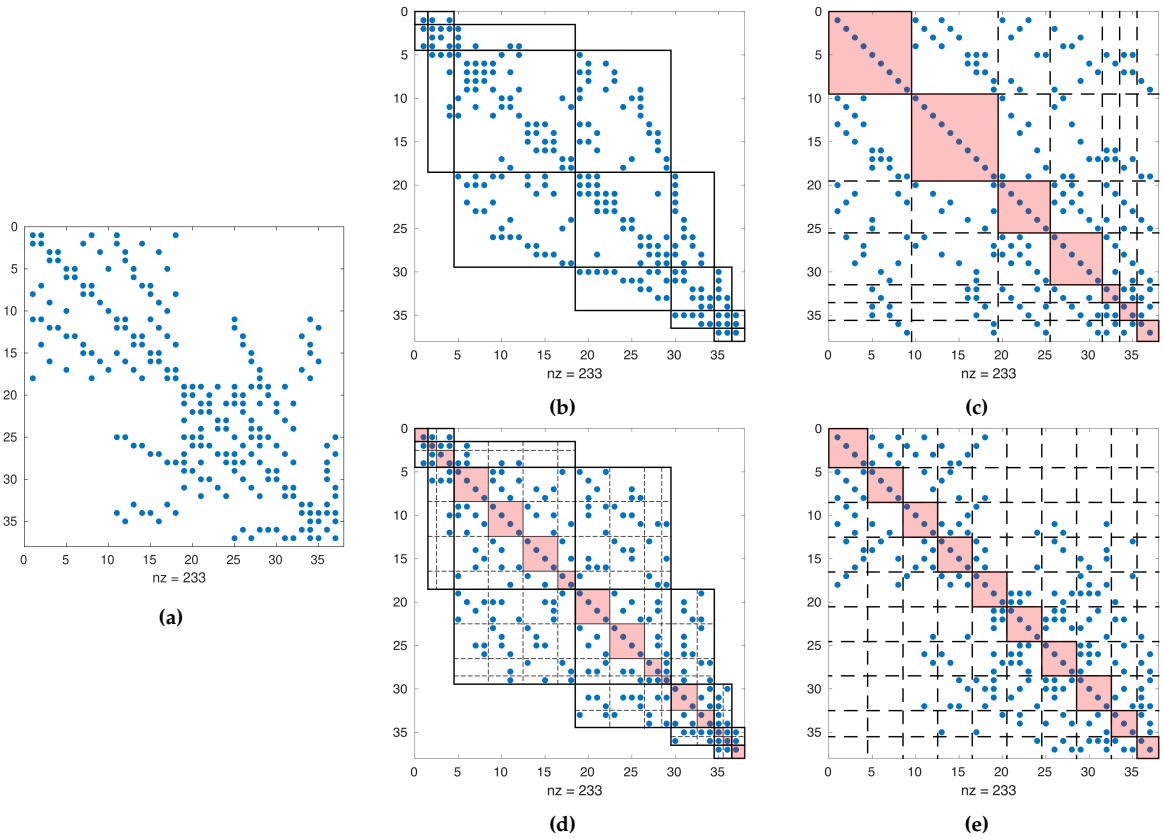
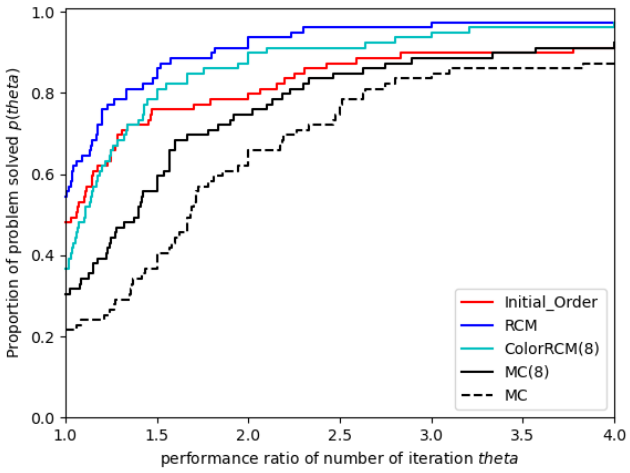
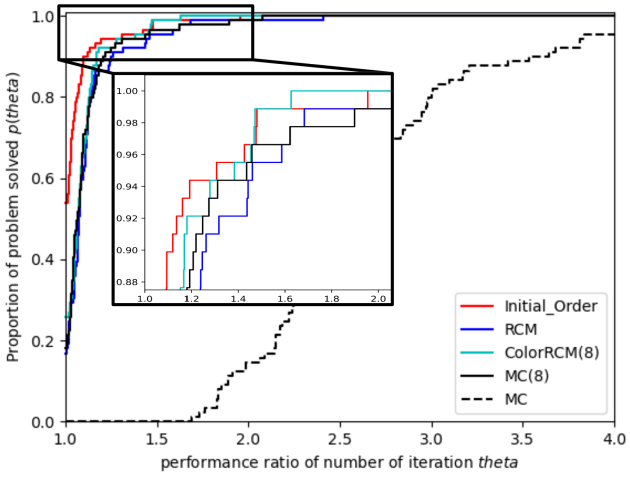


FIGURE 1 – *Representation of the cage5 matrix sparse profile. The blue bullets are the non-zero elements. The dark lines define the* RCM *levels, the red blocks illustrate the color classes and the dashed lines define the color blocks. (1a) initial ordering. (1b)* RCM *ordering, 7 levels. (1c)* MC *ordering, 7 colors. (1d)* COLORRCM*(4), 7 levels, 16 colors. (1e)* MC*(4) ordering, 10 colors.*

B. Experimental Results



(a)



(b)

FIGURE 2 – *Comparison of the evolution of the performance ratio of the number of iterations from* ILU0-BICGSTAB *for different orderings. Figure (2a) over the 227 matrices the SuiteSparse Matrix Collection. Figure (2b) over 124 matrices from IFPEN's porous media flow simulator.*

Bibliographie

1. Benzi (M.). – Preconditioning Techniques for Large Linear Systems : A Survey. *Journal of Computational Physics*, vol. 182, n2, novembre 2002, pp. 418–477.
2. Benzi (M.), Joubert (W.) et Mateescu (G.). – Numerical experiments with parallel orderings for ILU preconditioners. *Electronic Transactions on Numerical Analysis*, vol. Volume 8, 1999, pp. 88–114.
3. Benzi (M.), Szyld (D. B.) et van Duin (A.). – Orderings for Incomplete Factorization Preconditioning of Nonsymmetric Problems. *SIAM Journal on Scientific Computing*, vol. 20, n5, janvier 1999, pp. 1652–1670. – Publisher : Society for Industrial and Applied Mathematics.
4. Chow (E.) et Patel (A.). – Fine-Grained Parallel Incomplete LU Factorization. *SIAM Journal on Scientific Computing*, vol. 37, n2, janvier 2015, pp. C169–C193. – Publisher : Society for Industrial and Applied Mathematics.
5. Chow (E.) et Saad (Y.). – Experimental study of ILU preconditioners for indefinite matrices. *Journal of Computational and Applied Mathematics*, vol. 86, n2, décembre 1997, pp. 387–414.
6. Cuthill (E.) et McKee (J.). – Reducing the bandwidth of sparse symmetric matrices. – In *Proceedings of the 1969 24th national conference*, *ACM '69*, ACM '69, pp. 157–172, New York, NY, USA, août 1969. Association for Computing Machinery.
7. Davis (T. A.) et Hu (Y.). – The university of florida sparse matrix collection. *ACM Trans. Math. Softw.*, vol. 38, n1, dec 2011.
8. Doi (S.). – On parallelism and convergence of incomplete lu factorizations. *Applied Numerical Mathematics*, vol. 7, n5, 1991, pp. 417–436.
9. Dolan (E. D.) et Moré (J. J.). – Benchmarking optimization software with performance profiles. *Math. Program.*, 2001.
10. Dongarra (J. J.), Duff (L. S.), Sorensen (D. C.) et Vorst (H. A. V.). – *Numerical Linear Algebra for High Performance Computers*. – USA, SIAM, 1998.
11. Duff (I. S.) et Meurant (G. A.). – The effect of ordering on preconditioned conjugate gradients. *BIT Numerical Mathematics*, vol. 29, n4, décembre 1989, pp. 635–657.
12. George (A.). – Nested Dissection of a Regular Finite Element Mesh. *SIAM Journal on Numerical Analysis*, vol. 10, n2, avril 1973, pp. 345–363. – Publisher : Society for Industrial and Applied Mathematics.
13. George (J. A.). – *Computer Implementation of the Finite Element Method*. – Rapport technique, Stanford Univ. CA dept of science, février 1971. Section : Technical Reports.
14. Gibbs (N. E.), Poole, William G. (J.) et Stockmeyer (P. K.). – An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix. *SIAM Journal on Numerical Analysis*, vol. 13, n2, avril 1976, pp. 236–250. – Publisher : Society for Industrial and Applied Mathematics.
15. Iwashita (T.), Li (S.) et Fukaya (T.). – Hierarchical block multi-color ordering : A new parallel ordering method for vectorization and parallelization of the sparse triangular solver in the ICCG method. *CoRR*, vol. abs/1908.00741, 2019.
16. Iwashita (T.), Nakashima (H.) et 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*, 2012, pp. 474–483.
17. Saad (Y.). – Highly Parallel Preconditioners for General Sparse Matrices. – In Golub (G.), Luskin (M.) et Greenbaum (A.) (édité par), *Recent Advances in Iterative Methods*, *The IMA Volumes in Mathematics and its Applications*, The IMA Volumes in Mathematics and its Applications, pp. 165–199, New York, NY, 1994. Springer.
18. Saad (Y.). – *Iterative Methods for Sparse Linear Systems : Second Edition*. – SIAM, avril 2003.
19. Sloan (S. W.). – An algorithm for profile and wavefront reduction of sparse matrices. *International Journal for Numerical Methods in Engineering*, vol. 23, n2, 1986, pp. 239–251.
20. SPE. – The 10th spe comparative solution project, 2000.
21. van der Vorst (H. A.). – Bi-CGSTAB : A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, vol. 13, n2, mars 1992, pp. 631–644. – Publisher : Society for Industrial and Applied Mathematics.
22. Vuik (C.). – Krylov Subspace Solvers and Preconditioners. *ESAIM : Proceedings and Surveys*, vol. 63, 2018, pp. 1–43. – Publisher : EDP Sciences.