Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

THE EFFECT OF SYMMETRIC PERMUTATIONS ON THE CONVERGENCE OF A RESTARTED GMRES SOLVER WITH ILU-TYPE PRECONDITIONERS

Sanderson L. Gonzaga de Oliveira
Cláudio Carvalho

Departamento de Ciência da Computação
Universidade Federal de Lavras
Câmpus Universitário, C.P. 3037
Lavras, MG, 37200-000 BRAZIL

Carla Osthoff

Laboratório Nacional de
Computação Científica
Av. Getulio Vargas, 333, Quitandinha
Petrópolis, RJ, 25651-075 BRAZIL

ABSTRACT

This paper is concerned with applying heuristics for bandwidth reduction as a preprocessing step of a restarted Generalized Minimal Residual (GMRES for short) solver preconditioned by ILU-type preconditioners. Hundreds of heuristics have been proposed to solve the problem of bandwidth reduction since the mid-1960s. Previous publications have reviewed several heuristics for bandwidth reduction. Based on this experience, this paper evaluates nine low-cost symmetric permutations. Numerical simulations are presented to investigate the influence of these orderings on the convergence of the preconditioned GMRES solver restarted every 50 steps when applied to large-scale nonsymmetric and not positive definite matrices. This paper shows the most promising combination of preconditioner and ordering for each linear system used.

1 INTRODUCTION

The solution of large-scale sparse linear systems in the form $Ax = b$ where $A = [a_{ij}]$ is an $n \times n$ sparse matrix, x is the unknown n-vector solution, and b is a known n-vector is a fundamental step in several applications in science and engineering. It is typically a step of the simulation that demands high running times.

We concentrate our simulations on real nonsymmetric (structurally as well as numerically) matrices that are not positive definite. Such matrices arise from the analysis of power system networks, circuit simulation, computational fluid dynamics, electromagnetics, optimization problems, and elsewhere. These problems can cause severe difficulties for preconditioned iterative methods.

Iterative methods are fundamental tools to the solution of several large-scale problems in scientific computing (Benzi 2002). Krylov subspace iterative methods are now extensively employed in conjunction with preconditioning techniques for large-scale instances in scientific and engineering applications. The GMRES method of Saad and Schultz (1986) is a prominent Krylov subspace method for solving non-Hermitian linear systems $Ax = b$ where $A \in \mathbb{C}^{n \times n}$ is a nonsingular matrix and $b \in \mathbb{C}^n$ (Saad 2003). The GMRES method terminates with the exact solution in at most n steps. However, it uses the Arnoldi algorithm and, consequently, its execution costs increases with each iteration. Thus, practitioners usually restart the GMRES(m) method after m iterations, using the last computed residual as the next initial one. This restarted version may stagnate (Saad and Schultz 1986). Convergence can be slow even if stagnation does not occur, and researchers have proposed several remedies for this problem (see Güttel and Pestana (2014) and references therein). Thereby, preconditioning is a crucial ingredient for the success of the restarted GMRES method. Saad (2003) describes the restarted GMRES solver and incomplete LU factorizations in detail (see also Saad (2016)).

Incomplete factorization preconditioners also depend on the ordering of unknowns and equations. It is a common experience that if the matrix has nonzero coefficients close to the main diagonal, the number of floating point operations in the ILU factorization can be smaller and, consequently, reordering the unknowns and equations influence the rate of convergence of preconditioned Krylov subspace methods (Benzi et al. 1999; Benzi 2002).

We are particularly interested in the effects of symmetric permutations of the large-scale matrix A to further improve incomplete (approximate) factorization preconditioning for sequential calculations with the restarted GMRES solver. Thus, this paper shows a set of numerical simulations on the influence of nine reordering approaches on the convergence of the preconditioned GMRES solver restarted every 50 steps when using the ILU (Meijerink and van der Vorst 1977), ILUT (Saad 1994), VBILU, and VBILUT (Saad 2002) methods as preconditioners for a set of large-scale sparse matrices (ranging from 1,000,005 to 4,690,002 unknowns and from 3,105,536 to 27,130,349 nonzero coefficients). Specifically, we evaluate nine heuristics for bandwidth reduction of matrices in a preprocessing step for the restarted preconditioned GMRES solver.

There are significant motives for finding an adequate order for the coefficient matrix. Modern hierarchical memory architecture and paging policies benefit programs that take locality of reference into account. Spatial locality is a relevant aspect when designing an algorithm. An adequate nodal ordering is essential for the low-cost solution of large and sparse linear systems. Heuristics for bandwidth reduction are typically employed to obtain a sequence of graph (associated with the matrix A) vertices with spatial locality. Thus, heuristics for bandwidth reduction are commonly used to obtain low processing times for solving large sparse linear systems by iterative methods (Gonzaga de Oliveira et al. 2018b).

Due to its importance in numerical mathematics, important publications have investigated this subject, as described below. Nevertheless, our contribution is to evaluate various low-cost heuristics for bandwidth reduction along with several other variants of symmetric permutations in conjunction with the restarted GMRES(50) solver preconditioned by four ILU-type preconditioners when applied to large-scale non-symmetric linear systems. Specifically, we employed in our simulations both the preconditioners and the Flexible GMRES solver available in ITSOL v.2.0, a package of iterative solvers (Saad 2016). In particular, this is a state-of-the-art implementation of the restarted GMRES method and its preconditioners.

The remainder of this manuscript is structured as follows. Section 2 reviews important works in this field. Section 3 discusses the heuristics for bandwidth reduction evaluated in this study. Section 4 describes how the simulations were conducted in this study. Section 5 describes in detail the parameters evaluated in the restarted preconditioned GMRES solver. Section 6 presents and discusses the results. Finally, Section 7 addresses the conclusions.

2 RELATED WORK

In this field, significant works have addressed the influence of permutations on the convergence of the restarted preconditioned GMRES solver (Saad and Schultz 1986). Camata et al. (2012) reviewed the most important publications in this subject. We provide only a brief review of the most important contributions in the field.

Benzi et al. (1999) published one of the most important articles on reordering nonsymmetric matrices for solving linear systems by the GMRES method. These authors presented a very detailed review of the subject. We reproduce here their main results. For the highly nonsymmetric test matrices used, i.e., when the nonsymmetric part is large, Benzi et al. (1999) concluded that level set reorderings favor the convergence of the preconditioned GMRES solver. These researchers used several instances originating from petroleum engineering, metal forming simulation, neutron diffusion, plasma physics, and incompressible flow applications. The authors used small instances for today's standards, i.e., the linear systems used were lower than 6,000 unknowns and 84,000 nonzero coefficients. Benzi et al. (1999) employed the GMRES(20) solver preconditioned by the ILU(0), ILU(1), ILUT(1e-2 ,5), and ILUT(1e-3,10) preconditioners. These authors investigated the results achieved by the (reverse) BFS, (reverse) CM, and multiple minimum degree

orderings. The authors did not describe the pseudoperipheral vertex finder used, but probably the RCM method (George 1971) was used with the starting vertices given by the George-Liu algorithm George and Liu (1979). Benzi et al. (1999) concluded that the Reverse Cuthill-McKee method (George and Liu 1981) provided the best performance and robustness results, particularly alongside the ILUT preconditioning when applied to finite difference approximations of convection-dominated convection-diffusion equations. Benzi (2002) provided a detailed survey of preconditioned techniques for large linear systems.

Among several experiments, Camata et al. (2012) conducted a specific serial simulation. These authors investigated the use of ordering strategies along with the ILU algorithm used as a preconditioner to the restarted GMRES solver. Specifically, these authors compared the results obtained using natural, quotient minimum degree, and RCM-GL (George and Liu 1981) orderings when applied to a 2-D serial adaptive mesh refinement and coarsening problems arising from finite element discretizations. To provide further details, the authors studied a mixed vector field problem. Specifically, this vector field problem is a variable transient incompressible Navier–Stokes problem. They used an instance composed of 16,384 vertices and concluded that the RCM ordering generally obtains faster solves.

The results presented by Benzi et al. (1999) and Camata et al. (2012) showed that the selection of the ordering strategy strongly influences the CPU time and the number of GMRES iterations. Additionally, their results show that the use of the natural (or original) ordering provides experiments substantially more expensive than using the RCM-GL ordering (George and Liu 1981). In general, these authors recommended employing the RCM-GL (George and Liu 1981) as a preprocessing step for the restarted preconditioned GMRES solver.

3 HEURISTICS FOR BANDWIDTH REDUCTION EVALUATED

The bandwidth minimization problem consists of labeling the vertices of a graph with integer labels aiming at minimizing the maximum absolute difference between the labels of adjacent vertices. The problem is isomorphic to the problem of reordering the rows and columns of a symmetric matrix in such a way that its non-null coefficients are maximally located as close as along the main diagonal (Koohestani and Poli 2011). Let $A = [a_{ij}]$ be an $n \times n$ symmetric adjacency matrix associated with an undirected graph $G = (V,E)$ composed of a set of vertices V and a set of edges E. The bandwidth of line i is $\beta_i(A) = i - \min_{1 \leq j \leq i}[j : a_{ij} \neq 0]$. The bandwidth of A is defined as $\beta(A) = \max_{1 \leq i \leq n}[\beta_i(A)]$. Equivalently, the bandwidth of G for a vertex labeling $S = \{s(v_1), s(v_2), \cdots, s(v_{|V|})\}$ (i.e. a bijective mapping from V to the set $\{1, 2, \cdots, |V|\}$) is $\beta(G) = \max_{v \in V}[(\{v,u\} \in E)\ |s(v) - s(u)|]$. The bandwidth minimization problem is NP-hard (Papadimitriou 1976). Thus, since the mid-1960s, several heuristics for bandwidth reduction have been proposed.

Previous publications (see Gonzaga de Oliveira et al. (2018b) and references therein) have identified promising low-cost heuristics for bandwidth reduction: KP-band (Koohestani and Poli 2011), reverse breadth-first search (BFS) with the starting vertex given by the George-Liu (GL) algorithm, and RCM-GL (George and Liu 1981) heuristics. The Reverse Cuthill-McKee method (George 1971) labels the vertices of a graph $G(V,E)$ in order of increasing distance from a given pseudoperipheral vertex. Specifically, the method labels vertices with the same distance from the vertex starting vertex in order of increasing degree. Finally, the ordering is reversed. The KP-band heuristic labels the vertices of a graph using the formula $0.179492928171 \cdot (\varsigma(v))^3 + 0.292849834929 \cdot (\varsigma(v))^2 - 0.208926175433 \cdot n - 0.736485142138 \cdot n \cdot \varsigma(v) - 1.77524579882 \cdot \varsigma(v) - 1.75681383404$ where n is the dimension of matrix A and $\varsigma(v)$ is the sum of degrees of vertices connected to the vertex v.

A low-cost heuristic that reasonably reduces the bandwidth of the coefficient matrix A is more relevant in reducing the computing time required by an iterative linear solver than using a high-cost heuristic that reduces the bandwidth of A to a considerable extent (Gonzaga de Oliveira et al. 2018b). Thus, in addition to these three reordering algorithms, we also include in our simulations the BFS, reverse BFS (see Benzi

et al. (1999)), BFS-GL, Cuthill–McKee (CM) (Cuthill and McKee 1969), Reverse Cuthill–McKee (RCM) (George 1971), and CM-GL procedures.

We selected these nine low-cost heuristics for bandwidth reduction because an adequate vertex labeling of a graph associated with a matrix contained in a linear system may reduce the computational times of an iterative linear solver, such as the GMRES method (Benzi et al. 1999). The bandwidth reduction obtained is not directly proportional to the computational time reduction of solving linear systems by an iterative solver. Furthermore, at least when only a single linear system is to be solved (Gonzaga de Oliveira et al. 2018b), the total simulation time (i.e., including the time required by the reordering procedure) should present low cost. Thus, an application should employ a low-cost reordering algorithm. The orderings evaluated in this study are graph theoretical algorithms that perform only symmetric permutations and preserve the set of diagonal entries. Additionally, these orderings neither affect the coefficients nor the eigenvalues of the nonsymmetric matrix A.

4 DESCRIPTION OF THE TESTS

We wrote the codes of the heuristics for bandwidth reduction in the C++ programming language using the g++ version 5.4.0 compiler, with the optimization flag -O3. As previously mentioned, we applied both preconditioners and the Flexible GMRES solver available in ITSOL v.2.0 (Saad 2016). The implementations use double-precision floating-point arithmetic.

To evaluate the performance of the restarted preconditioned GMRES solver (Saad and Schultz 1986) computed after executing reordering algorithms, we used 11 linear systems contained in the SuiteSparse matrix collection (Davis and Hu 2011). We selected real nonsymmetric instances with size greater than 1,000,000 from this sparse matrix collection. Thus, this search in the sparse matrix collection returns 14 nonsymmetric instances. However, we did not include the instances *circuit5M*, *cage15*, and *HV15R* in our simulations. The amount of memory on our machine is insufficient to compute these instances. Because the sizes of the problems are greater than 1,000,000 unknowns, we apply an iterative method to solve the linear systems. Thus, the preconditioned GMRES(50) method solves the linear systems in this study. The matrices are not positive definite, but real nonsymmetric instances.

We used the structure of $A+A^T$ in our simulations. There are other graphs used for preordering A, including the row graph A^TA and bipartite graph. Nevertheless, we computed $A+A^T$ because it is a simple strategy that presents lower storage and execution costs than the other strategies do. In particular, the MATLAB software (The MathWorks, Inc. 2019) includes this strategy along with the RCM-GL method (George and Liu 1981).

The vectors b and x_0 do not define if the GMRES converge or not, but can influence the number of iterations and the CPU time is used to evaluate the results of algorithms in our simulations. We used CPU time as the primary metric in the simulations because different iterations can be either time-consuming or fast.

When the collection does not supply the right-hand side of a linear system, we generated a vector b with random entries in the range $(-1,1)$. We initialized the vector x_0 with zeros. The performance of the GMRES solver depends on the choice of the initial ordering, and we considered the original sequence of unknowns and equations given in the linear systems.

A suitable parameter m could be very different for each matrix. The optimal value is profoundly problem dependent. Thus, it is not straightforward to find an optimal value for each matrix. The sizes of the matrices used in our simulations increase this issue. Nevertheless, since our interest is in the study of the influence of symmetric permutations on the convergence of the restarted preconditioned GMRES method, we fixed for all matrices the number of restart vector for the GMRES solver. Section 5 shows how we chose the initial parameters for the GMRES(50) solver preconditioned by the ILU(ℓ) and VBILU(ℓ) algorithms, for ℓ established as 1, 2, 4, and 12, and ILUT(τ,p) and VBILUT(τ,p) preconditioners with (1e-3,50), (1e-6,75), (1e-9,100), (1e-12,200), and (1e-15,250). Section 5.2 defines the parameters τ and p.

If the GMRES(50) solver does not converge with a specific preconditioner, our program repeats its execution with different parameters. For example, if the GMRES(50) solver preconditioned by the ILU(1) method does not converge when applied to a specific instance, then the program employs the ILU(2) method as the preconditioner. If it does not converge again, the program applies the ILU(4) preconditioner. In our simulations, the GMRES(50) solver stops either when the norm of the residual vector is less than 1e-8 or when a maximum number of iterations ($|n|$) was reached.

Since the preconditioned GMRES method with 50 as the restart parameter generally takes few seconds to solve a linear system, we aborted an execution that computed more than 20 minutes supposing that either the solver would not converge or the solver would converge after a much longer time. We arbitrarily chose this timeout. Nevertheless, the GMRES solver (along with the preconditioners used with different parameters) did not converge when applied to some linear systems. When using these most difficult linear systems, we performed another execution with the preconditioned solver applied to the instance with a timeout of 3,600 seconds.

The solver executed if the condest associated with the matrix is smaller than 1e+15. The FGMRES solver allows changes in the preconditioning every GMRES step, but we did not change the parameters during execution.

The workstation used to compute the simulations featured an Intel® Core™ i7-4770 (CPU 3.40 GHz, 8 MB of L3 cache, 16 GB of main memory DDR3 1.333 GHz) (Intel; Santa Clara, CA, United States). This machine uses Ubuntu 16.04.3 LTS 64-bit operating system with Linux kernel-version 4.13.0-39-generic.

5 PARAMETERS EVALUATED IN THE RESTARTED PRECONDITIONED GMRES SOLVER

This section describes how we chose the initial parameters in the restarted GMRES solver preconditioned by four ILU-type preconditioners. It can be found in the literature the use of the restarted GMRES(m) solver employed with several parameters. Essentially, the greater this parameter is, the larger the amount of storage required. This characteristic can be a limiting factor for using this parameter in large-scale instances. Even though we use large-scale linear systems, we studied greater values for the parameter in the GMRES solver. In exploratory investigations, we studied the use of the values 30, 50, 100, and 250 for this parameter.

Table 1 shows the parameters used for the four preconditioners evaluated in this study. We considered that the best combinations of parameters are those that lead to convergence of the GMRES solver at low cost.

Table 1: Preliminary parameters studied in the restarted preconditioned GMRES solver.

Method	Parameter	Values
GMRES	m	30, 50, 100, 250
ILU, VBILU	ℓ	0, 1, 2, 3, 4
ILUT,	τ	1e-1, 1e-3, 1e-6, 1e-9
VBILUT	p	15, 50, 100, maximum degree

We applied each preconditioner with each combination of parameters to 35 linear systems ranging from 12,111 to 150,102 unknowns. Table 2 shows these (arbitrarily chosen) instances.

Table 3 shows the values ρ calculated for each parameter m evaluated, where $\rho = \sum_{i=1}^{N} \frac{\beta_H(i) - \beta_{min}(i)}{\beta_{min}(i)}$ (Gonzaga de Oliveira et al. 2018a), $\beta_H(i)$ is the bandwidth obtained when using an algorithm H applied to the instance i, $\beta_{min}(i)$ is the lowest bandwidth obtained in the instance i (considering the algorithms evaluated and the original bandwidth of the matrix), and N is the number of matrices. The metric ρ in Table 3 shows that the best results were obtained when using the restarted preconditioned GMRES solver with 50 vectors in the Krylov basis, denoted GMRES(50) solver, among the three other parameters evaluated in this study. With this value for the GMRES solver, we also evaluated the best parameters on average for the four preconditioners when applied to the instances exhibited in Table 2.

Table 2: Thirty-five instances used in exploratory investigations.

Instance	n	$\|E\|$	Instance	n	$\|E\|$	Instance	n	$\|E\|$
ncvxqp1	12,111	73,963	minsurfo	40,806	203,622	cont-201	80,595	438,795
cbuckle	13,681	676,515	vanbody	47,072	2,329,056	apache1	80,800	542,184
olafu	16,146	1,015,156	gridgena	48,962	512,084	shallow_water1	81,920	327,680
bodyy6	19,366	134,208	cvxbqp1	50,000	349,968	consph	83,334	6,010,480
raefsky4	19,779	1,316,789	sparsine	50,000	1,548,988	ASIC_100ks	99,190	578,890
qpband	20,000	45,000	dixmaanl	60,000	299,998	lung2	109,460	492,564
msc23052	23,052	1,142,686	blockqp1	60,012	640,033	torso2	115,967	1,033,473
bcsstk37	25,503	1,140,977	venkat01	62,424	1,717,792	cop20k_A	121,192	2,624,331
smt	25,710	3,749,582	cant	62,451	4,007,383	cfd2	123,440	3,085,406
brainpc2	27,607	179,395	Dubcova2	65,025	1,030,225	Dubcova3	146,689	363,6643
ship_001	34,920	3,896,496	bcircuit	68,902	375,558	G2_circuit	150,102	726,674
c-57	37,833	403,373	cfd1	70,656	1,825,580			

Table 3: Metric ρ calculated for four parameters evaluated in the GMRES solver.

m	30	50	100	250
ρ	4.47	**2.25**	2.29	3.74

It is natural that, due to the variety of characteristics present in the instances used in this study (i.e., the large number of vertices, large number of nonzero coefficients, etc.) arising from different application areas, the choice of parameters is not ideal for every matrix used. This generic choice of parameters means that a generic application of the restarted preconditioned GMRES solver will be unsuccessful for some instances contained in the dataset used. Nevertheless, these values made effective the set of algorithms for eight out of 11 large-scale nonsymmetric linear systems used in this study, recalling that our objective is to investigate the performance of reordering algorithms employed as a preprocessing step for the restarted preconditioned GMRES solver. Thus, we decided to set a "generic" restarted preconditioned GMRES solver to study the performance of the reordering algorithms.

For each parameter, we calculated the average times of the restarted preconditioned GMRES solver. Then, we calculated the metric ρ (Gonzaga de Oliveira et al. 2018a) with these average times. Table 4 shows the definitive parameters that we used for the preconditioners. As previously mentioned, we used the parameters described in the following round if the preconditioned GMRES(50) solver did not converge with the parameters set in the previous experiment. Section 5.1 [5.2] explains how we chose the parameters listed in Table 4 for the (VB)ILU [(VB)ILUT] preconditioners.

Table 4: Parameters used in the simulations with the preconditioned GMRES(50). The program applied values of the next trial only to instances that the solver did not converge in the previous experiment.

Preconditioner	Parameter	1st round	2nd round	3rd round	4th round
ILU, VBILU	ℓ	1	2	4	12
ILUT,	τ	1e-3	1e-6	1e-9	1e-15
VBILUT	p	50	75	100	250

5.1 ILU and VBILU Preconditioners

The ILU preconditioner available in the ITSOL package (Saad 2016) is an ILU preconditioner with level ℓ of fill. The VBILU preconditioner available in the ITSOL package is a variable block preconditioner with a

level of fill and with automatic block detection. Because we use large-scale instances, these preconditioners could compute for a long time if setting large values for the parameter ℓ.

The metric ρ in Table 5 shows that the ILU and VBILU preconditioners yielded better results when using the parameter ℓ established as 1 and 2. In particular, the computation of the no-fill ILU preconditioner is inexpensive. This preconditioner is effective for significant problems, such as low-order discretizations of scalar elliptic partial differential equations leading to diagonally dominant nonsingular matrices (Meijerink and van der Vorst 1977; Benzi 2002). However, the metric ρ in Table 5 shows that for harder and more practical problems, the no-fill factorizations result is a too simple approximation of A, and preconditioners that allow fill-in in the incomplete factors are required (see Benzi (2002)). Benzi (2002) explains that this is the case for highly nonsymmetric and indefinite matrices, such as those originating from many computational fluid dynamics applications. The same author also describes that high values of ℓ hardly pay the increasing computing times, except maybe for very complicated problems. In the cases where instability occurs, the problem may disappear by allowing more fill-in in the incomplete factors (Benzi et al. 1999). However, according to these authors, there is no guarantee that this strategy will always work. Remarkably, these authors showed cases where increasing the amount of fill-in provided worse results by increasing the instability of the factors. A preconditioner that returns a dense matrix may demand few iterations to converge at the cost of high CPU time (Benzi 2002). Thus, we studied only a few sets of parameters for the preconditioners, keeping in mind that the objective of this work is to investigate the performance of reordering algorithms when applied as a preprocessing step of the restarted preconditioned GMRES solver. Although the goal is to evaluate the effects of reordering algorithms, large values for the parameter ℓ cause the resulting matrix to become denser. On the other hand, we used large-scale linear systems in the simulations. Thus, we also used a high value for the parameter ℓ. For instance, we used the GMRES(50) preconditioned by the (VB)ILU(12) algorithm in the cases that the solver did not converge when preconditioned by the (VB)ILU(1), (VB)ILU(2), and (VB)ILU(4) algorithms.

Table 5: Metric ρ calculated for five parameters for the ILU(ℓ) and VBILU(ℓ) algorithms when preconditioning the GMRES solver.

GMRES	ℓ	0	1	2	3	4
Four parameters in Table 3	ρ	6.77	5.29	**5.28**	5.64	10.16
GMRES(50)	ρ	5.67	**2.33**	4.33	7.24	9.24

5.2 ILUT(τ, p) and VBILUT(τ, p) Preconditioners

The ILUT preconditioner available in the ITSOL package (Saad 2016) is an ILU preconditioner with a threshold. The preconditioner drops an element if it is less than the relative tolerance τ_i (i.e., τ multiplied by the original norm of the i-th row, e.g., the 2-norm). The preconditioner also applies another dropping rule. It keeps only the p highest elements in each L and U parts of the row. The preconditioner always keeps the diagonal element (Saad 2003). The VBILUT preconditioner available in the ITSOL package is a variable block preconditioner with a threshold and with automatic block detection.

Usually, researchers empirically choose the parameters τ and p for a few sample matrices from a specific application when searching for satisfactory values. The optimal values, however, are heavily problem dependent.

The literature includes several combinations for the parameters τ and p. Researchers have established the parameter τ as inversely proportional to the parameter p. Usually, the parameter τ is set ranging from 1e-7 to 1e-1. We studied this parameter defined as 1e-1, 1e-3, 1e-6, and 1e-9 (combined with several values for the parameter p). We studied the parameter p set as 15, 50, 100, and the maximum degree found in the instance.

The metric ρ in Table 6 shows that the preconditioners achieved better results when using the parameter τ established as 1e-3 both when using results obtained by applying only the GMRES(50) solver and the

solver with the parameters presented in Table 3. The values ρ in Table 6 show that the preconditioners obtain better results when using the parameter p established as 50. Thus, we employed the ILUT and VBILUT methods with the parameters (1e-3, 50), (1e-6, 75), (1e-9, 100), and (1e-15,250) when preconditioning the GMRES(50) solver applied to large-scale linear systems. We employed the preconditioners established with the parameters (1e-15,250) to the matrices in the cases that the preconditioned iterative linear solver did not converge when setting the preconditioners with the other parameters described.

Table 6: Metric ρ calculated for several combinations of parameters in the ILUT(τ, p) and VBILUT(τ, p) algorithms when preconditioning the GMRES solver.

GMRES	τ	1e-1	1e-3	1e-6	1e-9
Four parameters in Table 3	ρ	18	**10**	30	97
GMRES(50)	ρ	20	**12**	33	101
GMRES	p	15	50	100	maximum degree
Four parameters in Table 3	ρ	12	**10**	15	15
GMRES(50)	ρ	12	**11**	17	17

6 RESULTS AND ANALYSIS

Tables 7 and 8, built from a wide variety of references that were part of this computational experiment, show the instance's name, preconditioner used, average running time required by the preconditioned GMRES(50) solver without the use of a reordering algorithm (see column Orig.), and running times of this preconditioned solver along with several reordering algorithms. The symbol * indicates that the preconditioned solver did not converge because of some inconsistency during preconditioning, such as a diagonal or pivot null or exceedingly small pivots, which is a possibility for matrices that do not have diagonal dominance and for highly unstructured problems. An inaccurate factorization can also happen even when the pivots are large. This kind of failure occurs when the preconditioner drops many large fill-ins from the incomplete factors. Ill-conditioning of the triangular factors also results in the instability of the recurrences involved in the forward and backward solves when executing the preconditioning (Benzi et al. 1999). The symbol † means that we aborted the run because of it computed for 20 minutes. A superscript indicates the trial that the solver converged. It also means that the solver did not converge in the previous experiment(s). The symbol ‡ denotes that we performed the fourth trial with the preconditioned solver applied to the instance, and the simulation terminated for exceeding 3,600 seconds. Numbers in dark gray color indicate the run that required the least amount of work when using the preconditioner. This case is not always the same as the run that demanded the lowest number of iterations (in light gray color) because, in general, different orderings result in a preconditioner with a different number of nonzero coefficients (Benzi et al. 1999). Therefore, numbers in dark gray color are the best results yielded in the instance.

The solver alongside the ILU(1) preconditioner achieved the best results when applied to the original ordering in three linear systems (*atmosmodd*, *atmosmodj*, and *atmosmodl*) included in our simulations. The RBFS ordering (see Benzi et al. (1999)) benefited the convergence of the GMRES(50) solver when applied in conjunction with the ILUT(1e-3,50) preconditioner in one case (instance *memchip*). In particular, the preconditioned GMRES(50) solver did not converge when applied to the original ordering of this instance.

The GMRES(50) solver preconditioned with the ILUT(1e-3,50) algorithm applied to the original matrix achieved the best results to the instances *cage14*, *circuit5M_dc*, and *rajat31*. The GMRES(50) solver preconditioned with the ILUT(1e-6,75) algorithm alongside the BFS ordering achieved the best results when applied to the instance *Freescale1*. The GMRES(50) solver preconditioned by the ILUT preconditioner converged much faster after a permutation of the coefficient matrix (see Table 7) than using the original ordering of the instance *Freescale1*. The George-Liu algorithm (George and Liu 1979) did not favor the reordering algorithms in the simulations performed in this study.

Table 7: Execution times (s) of the GMRES(50) solver preconditioned by the (VB)ILU and (VB)ILUT methods applied to nine nonsymmetric linear systems (continue on Table 8).

I.	Preconditioner		Orig.	RBFS-GL	RCM-GL	RBFS	RCM	BFS-GL	CM-GL	BFS	CM	KP
webbase-1M	β		987649	979,737	979,948	979,737	979,948	985,437	999,148	979,712	998,678	979,799
	(VB)ILU(T)		*	*	*	*	*	*	*	*	*	*
	β		21,904	7,772	7,772	7,773	7,773	7,773	7,772	7,773	7,773	7,775
atmosmodd	ILUT	t(s)	39	31	38	26	33	30	38	26	33	38
	(1e-3,50)	iter.	41	31	31	34	34	34	31	34	34	34
	ILU	t(s)	18	24	33	22	29	26	32	22	29	34
	(1)	iter.	75	71	75	75	75	75	71	75	75	77
	VBILU(T)		†	†	†	†	†	†	†	†	†	†
	β		21,904	7,773	7,772	7,773	7,773	7,772	7,772	7,773	7,773	7,775
atmosmodj	ILUT	t(s)	40	30	37	26	33	30	38	26	33	38
	(1e-3,50)	iter.	41	32	35	35	35	32	32	35	35	35
	ILU	t(s)	18	26	31	22	29	24	31	22	29	34
	(1)	iter.	75	75	66	75	75	66	66	75	75	75
	VBILU(T)		†	†	†	†	†	†	†	†	†	†
	β		39,204	7,182	7,183	7,183	7,183	7,182	7,183	7,183	7,183	7,185
atmosmodl	ILUT	t(s)	28	31	39	27	36	30	40	28	36	41
	(1e-3,50)	iter.	19	19	19	19	19	19	19	19	19	19
	ILU	t(s)	10	20	28	14	23	19	28	14	22	29
	(1)	iter.	35	38	37	34	35	37	38	35	34	36
	VBILU(T)		†	†	†	†	†	†	†	†	†	†
Hamrle3	β		1,442,878	1,446,303	1,442,845	1,442,845	1,442,845	1,446,680	1,446,908	1,442,845	1,445,832	1,445,433
	ILUT, ILU		*	*	*	*	*	*	*	*	*	*
	VBILU(T)		‡	‡	‡	‡	‡	‡	‡	‡	‡	‡
cage14	β		676,026	200,002	214,595	198,281	200,311	204,409	203,775	198,281	200,311	207,827
	ILUT	t(s)	32	65	84	51	67	64	81	52	68	87
		iter.	4	4	4	4	4	4	4	4	4	4
	ILU	t(s)	50	82	103	66	86	82	99	69	83	109
		iter.	4	4	4	4	4	4	4	4	4	4
	VBILU(T)		†	†	†	†	†	†	†	†	†	†
memchip	β		1,647,939	2,321,802	2,707,524	2,706,113	2,705,988	2,188,134	1,187,256	2,707,524	2,705,988	2,188,871
	ILUT	t(s)/iter.		18/5		9/5			28/6		21/6	
	ILU	t(s)	*	30	*	21	*	*	41	*	33	*
		iter.		39		39			40		39	
	VBILU(T)		†	†	†	†	†	†	†	†	†	†
FullChip	ILUT		*	*	*	*	*	*	*		*	*
	ILU									‡		
	VBILU(T)		‡	‡	‡	‡	‡	‡	‡		‡	‡
Freescale1	β		3,425,041	162,269	162,269	185,194	184,847	162,269	162,269	185,194	184,847	164,757
	ILUT	t(s)	303	50	$51^{/2}$	33	$36^{/2}$	$29^{/2}$	69	$17^{/2}$	55	$53^{/2}$
		iter.	565	51	10	47	10	10	50	10	52	11
	ILU	t(s)	$448^{/3}$	$360^{/3}$		$948^{/3}$			$380^{/3}$		$906^{/3}$	
		iter.	782	570	†	1,605	†	†	575	†	1,509	†
	VBILU(T)		†	†		†			†		†	

Table 9 provides the characteristics (name, size, number of nonzero coefficients ($|E|$), the best preconditioner employed, the most promising reordering algorithm applied to the instance, and the best time obtained) of the linear systems composed of real nonsymmetric matrices that were used in this computational simulation. The "—" symbol in Table 9 indicates the occurrences that could not be solved using the preconditioned GMRES(50) solver.

Table 8: Execution times (s) of the GMRES(50) solver preconditioned by the (VB)ILU and (VB)ILUT methods applied to two other nonsymmetric linear systems (continued from Table 7).

I.	Preconditioner		Orig.	RBFS-GL	RCM-GL	RBFS	RCM	BFS-GL	CM-GL	BFS	CM	KP
circuit5M_dc	β		2,832,158	165,214	164,338	122,060	122,141	165,214	164,338	122,060	122,141	165,937
	ILUT	t(s)	3	25	*	10	*	23	*	11	*	44
		iter.	3	4		2		3		3		3
	ILU	t(s)	7	26	*	12	*	30	*	16	*	51
		iter.	9	4		4		4		9		9
	VBILU(T)		†	†	†	†	†	†	†	†	†	†
rajat31	β		4,688,751	7,481	7,482	3,753	3,752	7,482	7,486	3,753	3,752	7,484
	ILUT	t(s)/iter.	4/6	*	$53^{/2}$/3	*	34/3	$250^{/3}$/3	*	10/5	*	*
	ILU	t(s)/iter.	17/12		$123^{/2}$/7		44/4	29/8		22/10		57/13
	VBILU(T)		†	†	†	†	†	†	†	†	†	†

Table 9: Best orderings and preconditioners used in conjunction with the GMRES solver restarted every 50 steps when applied to 11 linear systems composed of real nonsymmetric matrices. The simulations evaluated the (VB)ILU and (VB)ILUT preconditioners with several parameters in conjunction with nine reordering algorithms and the original ordering of the instance.

Instance	Size	$\|E\|$	Preconditioner	Ordering	t(s)
webbase-1M	1,000,005	3,105,536		—	
atmosmodd	1,270,432	8,814,880	ILU(1)	original	18
atmosmodj	1,270,432	8,814,880	ILU(1)	original	18
atmosmodl	1,489,752	10,319,760	ILU(1)	original	10
Hamrle3	1,447,360	5,514,242		—	
cage14	1,505,785	27,130,349	ILUT(1e-3,50)	original	32
memchip	2,707,524	13,343,948	ILUT(1e-3,50)	RBFS	9
FullChip	2,987,012	26,621,983		—	
Freescale1	3,428,755	17,052,626	ILUT(1e-6,75)	BFS	17
circuit5M_dc	3,523,317	14,865,409	ILUT(1e-3,50)	original	3
rajat31	4,690,002	20,316,253	ILUT(1e-3,50)	original	4

7 CONCLUSIONS

Among several reordering algorithms for bandwidth reduction identified in the literature (see Gonzaga de Oliveira et al. (2018a), Gonzaga de Oliveira et al. (2018b) and references therein), we applied nine low-cost symmetric permutations to solve these problems. Consistent with the current findings, the simulations conducted in this study show that the choice of the best combination of reordering algorithm and preconditioned GMRES solver is highly problem dependent.

Even applying nine orderings and four preconditioners (ILU, VBILU, ILUT, and VBILUT) with different parameters (see the parameters used in Table 4), the GMRES(50) solver did not converge when applied to three out of 11 linear systems (instances *webbase-1M*, *Hamrle3*, and *FullChip*). One could investigate better parameters for the restarted preconditioned GMRES solver (Saad 2016; Saad and Schultz 1986) to achieve convergence with these instances, but our principal objective here is to study reordering algorithms used as

a preprocessing step of the preconditioned iterative solver. A suitable parameter *m* for the restarted GMRES solver could be very different for each matrix analyzed. Nevertheless, similarly to several publications in this field, and since our objective is to evaluate the influence of reordering algorithms on the convergence of the restarted GMRES solver, after exploratory investigations (see Section 5), we fixed the number of restart vectors for the GMRES solver for all matrices used.

The George-Liu algorithm (George and Liu 1979) did not favor the reordering algorithms employed in our simulations. Furthermore, the GMRES(50) solver preconditioned with the VBILU and VBILUT algorithms did not converge to the 11 nonsymmetric linear systems used in this study.

The simulations show that the number of iterations is not the best metric to analyze the results. For example, the solver preconditioned by the ILUT method, when applied to the instance *rajat31*, took 4 and 250 seconds with the original and BFS-GL orderings in six and three iterations, respectively.

Relevant publications in this field (Benzi et al. 1999; Saad 2003; Camata et al. 2012) affirm that level set reorderings (such as the heuristics included in our simulations) can be useful for preprocessing iterative methods applied to nonsymmetric linear systems. However, the use of a reordering algorithm in our study led to the best results only when using two out of 11 linear systems, at least with the parameters evaluated.

It is a common experience that an important metric to be evaluated is the bandwidth of the matrices before and after the reordering. In this sense, reordering cannot be useful for preprocessing matrices that have small bandwidth. However, in most of the cases (instances *atmosmodd*, *atmosmodj*, *atmosmodl*, *cage14*, *rajat31*, and *circuit5M_dc*), although the heuristics have reduced the bandwidth of the matrices, the best results were obtained with the original ordering.

The execution costs of the preconditioned GMRES(*m*) method may depend more on the structure of the matrix *A*, the matrix spectrum, and spatial locality than the bandwidth of the instance. Nevertheless, if the user realizes that a reordering algorithm can help, a simple BFS procedure or a variant (i.e., a reverse BFS) should be preferred. The (reverse) breadth-first search ordering yielded, in general, better performance than did the classical Reverse Cuthill-McKee ordering (George 1971). For example, the GMRES(50) solver, even preconditioned by four algorithms, did not converge when applied to the instance *memchip* with the original ordering. Nevertheless, the solver preconditioned by the ILUT algorithm converged in 9 seconds to this instance when preprocessing the matrix with the RBFS algorithm. As another example, the solver preconditioned by the ILUT algorithm took 303 seconds to solve the instance *Freescale1* when using the original ordering, but when preprocessing the matrix by the BFS procedure, the preconditioned solver converged in 17 seconds.

We intend to investigate the effects of parallel orderings on the convergence of parallel implementations of the restarted preconditioned GMRES solver using OpenMP, Galois, and Message Passing Interface systems. We found no study on this subject. The literature on heuristics for bandwidth reduction presents many works with sequential algorithms and just a small fraction of parallel strategies. A systematic review of parallel heuristics for bandwidth reduction is another future step in our study. Concerning massively parallel computing, we plan to evaluate heuristics for bandwidth reduction implemented within the Intel® Math Kernel Library running on Intel® Scalable processors.

REFERENCES

Benzi, M. 2002. "Preconditioning Techniques for Large Linear Systems: A Survey". *Journal of Computational Physics* 182(2):418–477.

Benzi, M., D. B. Szyld, and A. Van Duin. 1999. "Orderings for Incomplete Factorization Preconditioning of Nonsymmetric Problems". *SIAM Journal on Scientific Computing* 20(5):1652–1670.

Camata, J. J., A. L. Rossa, A. M. P. Valli, L. Catabriga, G. F. Carey, and A. L. G. A. Coutinho. 2012. "Reordering and Incomplete Preconditioning in Serial and Parallel Adaptive Mesh Refinement and Coarsening Flow Solutions". *International Journal for Numerical Methods in Fluids* 69(4):802–823.

Cuthill, E., and J. McKee. 1969. "Reducing the Bandwidth of Sparse Symmetric Matrices". In *Proceedings of the 1969 24th International Conference, August 26–28*, 157–172. New York, NY: ACM.

Davis, T. A., and Y. Hu. 2011. "The University of Florida Sparse Matrix Collection". *ACM Transactions on Mathematical Software* 38(1):1–25.

George, A. 1971. *Computer Implementation of the Finite Element Method*. Ph. D. thesis, Stanford University, Stanford.

George, A., and J. W. Liu. 1981. *Computer Solution of Large Sparse Positive Definite Systems*. Englewood Cliffs: Prentice-Hall.

George, A., and J. W. H. Liu. 1979. "An Implementation of a Pseudoperipheral Node Finder". *ACM Transactions on Mathematical Software* 5(3):284–295.

Gonzaga de Oliveira, S. L., J. A. B. Bernardes, and G. O. Chagas. 2018a. "An Evaluation of Low-cost Heuristics for Matrix Bandwidth and Profile Reductions". *Computational & Applied Mathematics* 37(2):1412–1471.

Gonzaga de Oliveira, S. L., J. A. B. Bernardes, and G. O. Chagas. 2018b. "An Evaluation of Reordering Algorithms to Reduce the Computational Cost of the Incomplete Cholesky-Conjugate Gradient Method". *Computational & Applied Mathematics* 37(3):2965–3004.

Güttel, S., and J. Pestana. 2014. "Some Observations on Weighted GMRES". *Numerical Algorithms* 67(4):733–752.

Koohestani, B., and R. Poli. 2011. "A Hyper-heuristic Approach to Evolving Algorithms for Bandwidth Reduction Based on Genetic Programming". In *Research and Development in Intelligent Systems XXVIII*, 93–106. London, UK: Springer London.

Meijerink, J. A., and H. A. van der Vorst. 1977. "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric *M*-matrix". *Mathematics of Computation* 31:148–162.

Papadimitriou, C. H. 1976. "The NP-completeness of Bandwidth Minimization Problem". *Computing Journal* 16(3):177–192.

Saad, Y. 1994. "ILUT: A Dual Threshold Incomplete LU Factorization". *Numerical Linear Algebra with Applications* 1(4):387–402.

Saad, Y. 2002. "Finding Exact and Approximate Block Structures for ILU Preconditioning". *SIAM Journal on Scientific Computing* 24(4):1107–1123.

Saad, Y. 2003. *Iterative Methods for Sparse Linear Systems*. 2nd ed. Philadelphia: SIAM.

Saad, Y. 2016. "ITSOL v.2.0: Iterative Solvers Package". http://www-users.cs.umn.edu/ saad/software/ITSOL/index.html.

Saad, Y., and M. H. Schultz. 1986. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". *SIAM Journal on Scientific and Statistical Computing* 7(3):856–869.

The MathWorks, Inc. 2019. "MATLAB". http://www.mathworks.com/products/matlab/index.html.

AUTHOR BIOGRAPHIES

SANDERSON L. GONZAGA DE OLIVEIRA is an Assistant Professor of Computer Science at the Universidade Federal de Lavras, Brazil. He holds a DSc degree in Computer Science from the Universidade Federal Fluminense, Brazil. His research interests include heuristics for bandwidth and profile reductions, numerical methods, and parallel computing. His email address is sanderson@ufla.br.

CLÁUDIO CARVALHO holds a M.Sc. degree in Computer Science from the Universidade Federal de Lavras, with a dissertation in the fields of heuristics for bandwidth and profile of matrices and the restarted preconditioned GMRES solver. His email address is claudiovicar@gmail.com.

CARLA OSTHOFF is a researcher at National Laboratory for Scientific Computing (LNCC-Brazil). Her main research areas are: high performance computing, cluster computing, hybrid parallel computing, high performance scientific computing applications and parallel programming models. Prof. Osthoff received her BSc in Electronics Engineering from PUC/Rio de Janeiro, a MSc and a PhD in Computer Science from Universidade Federal do Rio de Janeiro, UFRJ, Brazil. She is currently the coordinator from LNCC High Performance Computing Center. Her email address is osthoff@lncc.br.