Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 2323-2343 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

A STUDY OF THE INFLUENCE OF SPARSE MATRICES REORDERING ALGORITHMS ON KRYLOV-TYPE PRECONDITIONED ITERATIVE METHODS

Kamila Ghidetti, Lucia Catabriga, Maria Claudia Boeres and Maria Cristina Rangel

Laboratory of High Performance Computing - LCAD, Federal University of Espírito Santo - UFES, Department of Informatics, Av. Fernando Ferrari, 514 - Goiabeiras - 29075-910 - Vitória - ES - Brazil, rg.kamila@gmail.com, luciac@inf.ufes.br, boeres@inf.ufes.br, crangel@inf.ufes.br

Keywords: Minimizing Bandwidth and Reduced Envelope, Matrices Reordering, Algorithms on Graphs, Combinatorial Optimization

Abstract. This work analyzes the influenc of matrices reordering algorithms on solving linear systems using non-stationary iterative methods GMRES and Conjugate Gradient, both with and without preconditioning. The algorithms referenced most often in the literature for the reordering of matrices are Reverse Cuthill-McKee (RCM), Gibbs-Poole-Stockmeyer (GPS), Nested Dissection (ND) and Spectral (ES). We analyze these algorithms and propose some modification comparing their solution qualities (minimizing bandwidth and minimizing envelope) and CPU times. Moreover, the linear systems associated with sparse matrices are solved via preconditioned Krylov-type iterative methods considering the incomplete LU factorization preconditioners. For the computational tests, we consider a set of structurally symmetric matrices that can come from various field of knowledge. We conclude that the reordering of matrices, in most cases, reduces the number of iterations in the iterative methods, but that reducing the CPU time depends on the size and conditioning of the matrix.

1 INTRODUCTION

A significan portion of scientifi problems include the solution of sparse and large linear systems. In these cases, the minimizing of the bandwidth (Carvalho et al. (2009)) and reducing of the envelope (Barnad et al. (1995)). are ways to simplify the solution of these systems. These pre-processing methods consist of performing permutations between rows and columns having nonzero elements of the matrix closest to the main diagonal. In the context of the solution systems via direct methods, minimizing the bandwidth reduces the fillin that occurs in LU decomposition. Large linear systems, however, are usually resolved by non-stationary iterative methods (Saad (2003)) These methods do not change the sparsity of the matrix but require convergence criteria. Generally, a process to accelerate convergence, called preconditioning, is necessary. The preconditioners, based on incomplete LU decomposition (Benzi et al. (1999); Camata et al. (2010)), are widely used to sharply accelerate the convergence rate. Such operations alter the sparseness of the matrix and consequently the effectiveness of the preconditioners depends on matrix reordering.

From the 60's through the 80's, a great deal of research was conducted into minimizing bandwidth, with researchers emphasizing RCM and GPS methods. These algorithms are based on graph search strategies and provide a high quality solution (Cuthill and McKee (1969); Gibbs et al. (1976)). Following this period, most of the literature is divided mainly into either proposing improvements to these algorithms or applying new heuristics to the problem. Also proposed (Martí et al. (2008); Caprara and Salazar (2005)), however, were exact methods of solving these problems; the tests were conducted, however, on relatively small instance problems; i.e., dimensions of under 1000. Marti et al. (2001) proposed the use of the Tabu Search with movements based on the elements most distant from the diagonal. This work was performed to compare this algorithm, the GPS, and an implementation of Simulated Annealing (SA) proposed by Dueck and Jeffs (1995). The researchers concluded that the Tabu Search had the highest solution quality; however the computational time was much longer than that of heuristic GPS (at its longest 50 times as long). The Tabu Search also showed a better solution quality than the SA, with an implementation time of around 20 times less. Using the same set of matrices in their experiments, Piñana et al. (2004) proposed applying GRASP with path relinking. Computational results show that GRASP achieved better solution and reduced computational time compared to the Tabu Search by Marti et al. (2001). Nevertheless, compared to GPS, computational time is thousands of times more.Lim et al. (2006) presented two new approaches to the problem: a method called *node shift*, a genetic algorithm. A comparison of these methods with GPS and GRASP presented in Piñana et al. (2004) was performed by computational tests using the same set of matrices. Results showed that both approaches, compared to the GPS and GRASP, improved the quality of solution. With respect to processing time, however, the node shift method runs about three times longer than the genetic algorithm, making it longer than the heuristic GPS.

A generalization of heuristic GPS was proposed by Wang et al. (2009). Computational tests showed that the proposal, with only a small increase in processing time, slightly improved the quality of the solution. In a novel approach,Carvalho et al. (2009) proposed a heuristic that did not consider the problem represented by graphs; the heuristic can be applied to both symmetrical and non-symmetrical matrices. Computational tests comparing this heuristic algorithm with the Cuthill-McKee (CM) show that the heuristic improves the quality of solution, at a processing time, however, hundreds of times longer than the CM.

Considering studies that use the reordering of matrices to be a preprocessing step to solving

linear systems, we highlight the work of Benzi et al. (1999). They tested the algorithms for reordering, RCM and ND, and the algorithms for solving systems, Bi-CGStab, GMRES, and TFQMR. They found the computational tests for the reordering of the coefficien matrix reduced the number of iterations of the algorithms. Moreover, some matrices that had not converged before reordering converged afterwards. Recently, Portugal et al. (2009) used the RCM algorithm, the preconditioner ILUT(T, p) (Saad (2003))and the GMRES algorithm. They concluded that the reordering improved the quality of the preconditioning ILUT.

In summary, all the works cited improved the solution quality when compared to the more classical algorithms and with those used more in the literature (RCM and GPS). None of these works, however, could improve the quality of solution and at the same time obtain a comparable computational time to those algorithms. Clearly, much remains to be done to obtain a better compromise between quality solution and run time.

To compare solution quality and processing time, we implemented four algorithms from the literature to reorder sparse matrices and their modifie versions proposed in this work. To analyze the influenc of the reordering algorithms in the solution of systems, we also implemented two algorithms for solving linear systems and the preconditioner ILU(p).

The paper is organized as follows. The next section summarizes the main mathematical defi nitions necessary to understand the algorithms implemented. Sections 3, 4 and 5 show, respectively, methods for reordering and improvements proposed for the algorithms, methods for solving linear systems, and the preconditioners implemented in this work. Section 6 shows the details of implementation and Section 7 the computational tests. In the fina section we offer our main conclusions and proposals for future work.

2 IMPORTANT DEFINITIONS

Let A be a structurally symmetric matrix; i.e., if $a_{ij} \neq 0$ then $a_{ji} \neq 0$, but not necessarily $a_{ij} = a_{ji}$. The bandwidth of A denoted by $lb(A)$ is define as the greatest distance from the firs nonzero element to the diagonal, considering all lines of matrix and the envelope of A denoted by $env(A)$, it is the sum of the distances of the firs element non-zero until the diagonal, also considering all the rows of the matrix (Coleman (1984)). In other words:

$$\begin{aligned} b_i &= (i-j) \ \forall \ a_{ij} \neq 0, \quad i = 2, \ldots, n \\ lb(A) &= \max_{i=2,\ldots,n} \{b_i\} \end{aligned} \tag{1}$$

$$env(A) = \sum_{i=2}^{n} b_i \tag{2}$$

where j is the column index of the firs nonzero element of row i. The problem of minimizing the bandwidth is NP-complete (Papadimitriou (1976)). Because of this characteristic, findin an exact solution to the problem in a reasonable computational time is difficult thus justifying the use of heuristic methods. If we propose a solution to this problem from a Combinatorial optimization viewpoint, we must reformulate it through graphs: the matrix associated with the linear system can be directly represented by the adjacency matrix of a graph and its reordering. This is represented by the problem of reordering the labels of the vertices of graph. To clarify the problem represented by the graphs, we will summarize the main concepts and properties of this mathematical structure.

Let $G = (V, E)$, a non-oriented graph consisting of a set of vertices V and a set of edges E. Two vertices are adjacent if there is an edge between them in G. The edge is incident to a vertex

if it is one of the ends of the edge. The degree of a vertex is the number of edges incident to it. A *path* in G is define as a sequence of edges of E connecting two vertices, called the initial and fina respectively. A *cycle* is a path in which the initial vertex coincides with the fina vertex. It is said that G is connected if there is a path between each pair of vertices of G. Otherwise, G is said to be disconnected and consists of connected subgraphs, called *connected components*. A graph T is a *tree* that is connected and has no cycles. The vertices of a tree with unit degree are called *leaves*. *Distance*(u, v) is define as the number of edges (length) or the cost associated with the shortest path between the vertices u and v of V. It define $associatednumbere(v)$ as the greatest distance of v to all other vertices of G. The associated number $e(v)$ for $v \in T$, where T is a tree, corresponds to the length or cost of the path from v to a leaf of the tree. *Diameter*(G) is define as the value of the higher associated number of the graph (Narsingh (1974)). *Pseudo-diameter* corresponds to a high associated number, but not necessarily the highest. *Peripheral vertices* of G are vertices whose associated number is equal to the diameter of the graph and *pseudo-peripheral* vertices are those with high associated numbers, but not necessarily the highest. *Structure of levels with root associated* G, $SL(G) = \{L_1, L_2 \dots L_k\}$, consists of a rooted tree with levels. A tree is rooted when it has a vertex that stands out from another (root). A vertex v_i is at level i of a rooted tree if v_i is distance i from the root. In this type of tree, a vertex can not be part of two distinct levels (Menezes (1995)). The bandwidth of matrix A $A\ lb(A) \leq 2w - 1$, where w level with the greater number of vertices (Gibbs et al. (1976)).

Other important information for understanding this work is related to spectral graph theory. Let G be a graph that can be represented by the Laplacian matrix obtained through the operation $L(G) = D(G) - A(G)$, $D(G)$ is a diagonal matrix and each element of row i corresponds to the degree of its vertex i and $A(G)$ is the adjacency matrix of G. The spectrum of $L(G)$ is the set of eigenvalues $(\lambda_1, \lambda_2, \dots, \lambda_k)$ obtained by solving $det(L - \lambda I) = 0$. Considering $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_k$, the smallest second eigenvalue define the *algebraic connectivity* where each component of the associated eigenvector $v(\lambda_2)$ is related to connectivity between the vertices (Fiedler (1973)).

3 HEURISTIC ALGORITHMS TO MINIMIZE BANDWIDTH AND REDUCE ENVELOPE

What heuristic can solve the problem of minimizing bandwidth and reducing the envelope, providing good solution quality and low execution time? In the literature, the heuristic most often referenced for being able to do this is the Cuthill McKee (CM), (Cuthill and McKee (1969)). This method was designed to reorder the rows and columns of a matrix represented by a graph. It applies a breadth-firs search procedure to visit all the vertices of the graph in a organized way: starting from an initial vertex and searching the neighborhood of the vertices in increasing order of degrees, the heuristic visits all vertices of the graph. The order of the vertices visited corresponds to the permutation of lines an columns to be held in the matrix after the procedure. Shortly after Cuthill and McKee came up with this heuristic, George (1971) observed that if we reversed the order numbering obtained by CM (lines 15-22 do Algorithm 1) we could reduce the matrix's envelope without affecting the bandwidth. Thus the Reverse Cuthill McKee algorithm was set (RCM). Several years later, Liu and Sherman (1976) mathematically proved George (1971) conjecture. The good results of RCM algorithms motivated the treatment of the reordering of matrices problem using graphs, Menezes (1995). Thus, we can say that the matrix-reordering algorithms generally have, as input, a graph $G(V, E)$ and the set of adjacent vertices of a vertex x in G ($Adj(x)$) and as output, a new labeling of the vertices of G. We also

Algorithm 1: **RCM**

```
1  Find an initial vertex (x)
2  SortG (Adj(x))
3  z = 1, F = Adj(x), i = |Adj(x)|, y = F[z], z=z+1
4  while (i ≤ |V|) do  //Home breadth-first search
5      SortG (Adj(y))
6      for j = 1, ..., |Adj(y)| do
7          if Adj(y)[j] ∉ F then
8              F[i] = Adj(y)[j]
9              i = i + 1
10         endif
11     endfor
12     y=F[z]
13     z = z+1
14 endw
15 m = 1  //Home of the reversed of the numbering
16 while (m < i) do
17     aux = F[m]
18     F[m] = F[i]
19     F[i] = aux
20     i = i − 1
21     m = m + 1
22 endw
```

defin *SortG* as a function that orders the vertices of the graph in increasing order of degrees. In this context, the RCM heuristic is presented in Algorithm 1.

The firs step of the algorithm is to search for an initial vertex. The choice of this vertex is closely related to the quality of the solution. Some studies indicate that an initial choice of vertices with high associated numbers results in good solution quality. So the firs approach taken by Cuthill and McKee (1969) proposed starting the search for the initial vertex by examining each vertex in with degree D ranging from $D_{min} \leq D < D_{min} + \frac{D_{max}}{2}$, where D_{min} is the lowest degree of the graph and D_{max}, the highest. Using this approach makes the RCM heuristic slow, as many vertices can have their degrees in this interval. George and Liu (1979) proposed a heuristic based on the level structure of a minimum degree's vertex. This heuristic compares values of associated number for each new level structure of the vertices of the last level. Algorithm 2 describes the computation of pseudo-peripheral vertices. We propose two modification to Algorithm 1. Our objective, of course, is to improve the quality of the solution and possibly reduce the processing CPU time. The firs modification (a) to obtain the initial vertex, consider the Dijkstra algorithm rather than Algorithm 2. This new algorithm will be called Algorithm RCM-P1. It will not be shown here because it is less than one step and is identical to Algorithm 1. The second modificatio (b) use the level structure of the vertex obtained by the heuristic proposed by George and Liu (1979), performing the new labeling of the vertices from their positions in levels, in increasing order of degrees (Algorithm 3, RCM-P2).

Algorithm 2: **Choice of pseudo-peripheral vertex**

Choice a vertex (x) of minimal degree.
for $j = 1, \ldots, n$ **do**
    SortG ($\overrightarrow{\text{Adj}}$(j))
endfor
$SL(x) = \{L_1, L_2, \ldots, L_k\}$
while $i \leq |L_k|$ **do**
    SortG (L_k)
    $y = NL_k[i]$
    $SL(y) = \{Ly_1, Ly_2, \ldots, Ly_k\}$
    if $e(y) > e(x)$ *and* $w(y) < w(x)$ **then**
        $x = y$
        $L_k = Ly_k$
        $i = 1$
    else
    endif
    $i = i + 1$
endw
x is the pseudo-peripheral vertex.

Algorithm 3: **RCM-P2**

Choose a initial vertex (x)
$j = 1, m = 1$
$SL(x) = \{L_1, L_2, \ldots, L_k\}$ *//Level structure of the vertex x.*
for $i = 1, \ldots, k$ **do**
    while $j \leq |L_i|$ **do**
        $\overrightarrow{\text{F}}[m] = \text{L}_i[j]$
        $m = m + 1$
    endw
endfor
$m = 1$
$i = |V|$
while $(m < i)$ **do**
    aux = $\overrightarrow{F}$[m]
    $\overrightarrow{\text{F}}[m] = \overrightarrow{\text{F}}[i]$
    $\overrightarrow{\text{F}}[i] = aux$
    $i = i - 1$
    $m = m + 1$
endw

Another algorithm quite often referenced in the literature is the GPS, Gibbs et al. (1976). Their idea, similar to RCM, uses for its initial vertices two vertices with the distance between them equal to the pseudo-diameter. The algorithm find the initial vertices and then creates a level structure for each. The two-level structures are joined by a process define as minimizing the width of level. In the end, it labels the vertices similarly to that of RCM. The main steps are described in Algorithm 4. We can verify that the same algorithm used to fin the pseudo-peripheral vertex, described above (Algorithm 2), can also be used to fin the pseudo-diameter. The main difference between the GPS and RCM algorithms is in Step 2 of Algorithm 4. The minimization of the wide level requires additional processing (details can be found in Gibbs et al. (1976)). In Step 3 of that algorithm, the difference lies in the fact that the relabeling was done on the new structure obtained in Step 2. In the algorithm proposed in Gibbs et al. (1976), a comparison is made between the GPS and RCM algorithm heuristics. The GPS algorithm obtained a solution quality comparable to that of RCM. Its CPU time, however, was around ten to twenty times less than RCM's. Using George and Liu (1979) heuristic to fin a pseudo-peripheral vertex (Algorithm 2)we fin that the RCM algorithm is faster than the GPS algorithm.

Algorithm 4: **GPS**

1 Find a pseudo-diameter, i.e., the initial vertices x and y.
2 Minimize the width of level through joining the structures x and y.
3 Renumber the graph similarly to that of RCM.

Algorithm 5: **ND**

1 $i = 0, m = 1$
2 **repeat**
3 Partition G
4 Get S, K_i, K_{i+1}
5 **for** $j = 1, \ldots, |K_i|$ **do**
6 $\overrightarrow{F}[m] = K_i[j]$
7 $m = m + 1$
8 **endfor**
9 **for** $j = 1, \ldots, |K_{i+1}|$ **do**
10 $\overrightarrow{F}[m] = K_{i+1}[j]$
11 $m = m + 1$
12 **endfor**
13 **for** $j = 1, \ldots, |S|$ **do**
14 $\overrightarrow{F}[m] = S[j]$
15 $m = m + 1$
16 **endfor**
17 $p = p + 1$
18 $m = m + 1$
19 **until** $i < p$;

Between the publication of the RCM and GPS algorithms, another algorithm was introduced by George (1973) the ND (Nested dissection) algorithm. It is a process of reordering a set of

columns that divides the matrix into $K \geq two$ disjointed matrices (Algorithm 5, ND). Note that findin a set of columns that divides the matrix into K matrices can be interpreted as a problem of graph partitioning. In this problem, we want to achieve the lowest subset of edges (cut) that splits the graph into disjointed sets K (K connected components of the original graph with aproximadament same cardinality). The set S of Algorithm 5 corresponds to the set of vertices formed by only one end of each edge of this cut (vertex separators). The sets K_i and K_{i+1} are the partitions i and $i+1$ obtained by partitioning the graph and p is the number of times you should see the graph partitioned. Due to this characteristic, several versions of the ND algorithm have been proposed and graph partitioning heuristics are widely used. The ND algorithm implemented in this work uses a heuristic based on the adjacencies of the graph. It performs a breadth-firs search that introduces each visited vertex into a partition of the graph. When the value $|V|/2$ is reached, the search process ends. Moreover, to perform refinemen partitions

Algorithm 6: **ES**

1 Calculate λ_2 and $v(\lambda_2)$ of (L) *// λ_2 algebraic connectivity and $v(\lambda_2)$ eigenvector associated.*
2 Sort $v(\lambda_2)$ in ascending order, generating $\overrightarrow{F}_1$
3 Apply the permutation ($\overrightarrow{F}_1$) on L, generating L_1.
4 Calculate $x = env(L_1)$.
5 Sort $v(\lambda_2)$ in descending order, generating $\overrightarrow{F}_2$
6 Apply the permutation ($\overrightarrow{F}_2$) on L, generating L_2.
7 Calculate $y = env(L_2)$.
8 **if** $x < y$ **then**
9 $\quad \overrightarrow{F} = \overrightarrow{F}_1$
10 **else**
11 $\quad \overrightarrow{F} = \overrightarrow{F}_2$
12 **endif**

for all partitions obtained, we use the heuristic Fiduccia Mattheyses (Fiduccia and Mattheyses (1982)) based on cuts by edges. We propose for the ND algorithm two modifications (a) if we label the vertices so that the labels of K_i are smaller than the labels of S which in turn are smaller than the labels K_{i+1}, we defin a new algorithm called ND-P3; (b) if we re-label the partitions that consider the set S to consist of all vertices of extreme edges belonging to cut edges, we defin another algorithm, called in this work, ND-P4.

Finally, the last algorithm implemented was the Spectral (ES), proposed by Barnad et al. (1995). It is based on the use of eigenvalues and eigenvectors of the Laplacian matrix of graph $G(V, E)$ (Algorithm 6, ES). The firs step of this algorithm is highly complex. The complexity can be minimized, however, by an optimized storage form of the Laplacian matrix. Details are given in Section 6.

4 NON-STATIONARY ITERATIVE METHOD

Iterative methods generate successive approximations at every step, aiming for more accurate solutions to a linear system. We divide iterative methods into two groups, stationary iterative and non-stationary iterative methods. Stationary iterative methods are expressed as $x^{(k)} = Bx^{(k-1)} + c$, where B and c not depend on the calculation of the iteration k, These methods are the easiest to implement, but convergence is slower. Non-stationary iterative methods differ in that with

Algorithm 7: **GMRES**

Data $A, b, lmax, tol, x = 0, k$
$\epsilon = tol\|b\|_2$
$l = 1$
repeat
  $i = 1, u_i = b - Ax, e_i = \|u_i\|, u_i = \dfrac{u_i}{e_i}, \rho = e_i$
  while $(\rho > \epsilon) and (i < k)$ **do**
    $u_{i+1} = Au_i$
    for $j = 1, \ldots, i$ **do**
      $h_{ji} = u_{i+1}^t u_j$
      $\mathrm{u}_{i+1} = u_{i+1} - h_{ji}u_j$
    endfor
    $h_{i+1,i} = \|u_{i+1}\|_2$
    $u_{i+1} = \dfrac{u_{i+1}}{h_{i+1,i}}$
    for $j = 1, \ldots, i-1$ **do**
      $h_{ji} = c_j h_{ji} + s_j h_{j+1,i}$
      $\mathrm{h}_{j+1,i} = -s_j h_{ji} + c_j h_{j+1,i}$
    endfor
    $r = \sqrt{h_{ii}^2 + h_{i+1,i}^2}$
    $c_i = \dfrac{h_{ii}}{r}$
    $s_i = \dfrac{h_{i+1,i}}{r}$
    $h_{ii} = r\ (h_{i+1,i=0})$
    $e_{i+1} = -s_i e_i$
    $e_i = c_i e_i$
    $\rho = \mid e_{i+1} \mid$
    $i = i + 1$
    for $j = i, \ldots, 1$ **do**
      $$y_j = \frac{e_j - \sum_{l=j+1}^{i} h_{jl} y_l}{h_{jj}}$$
    endfor
    $$x = \sum_{j=1}^{i} y_j u_j$$
    $l = l + 1$
  endw
until $(\rho < \epsilon) or (l \geq lmax)$;

each iteration information varies. Its strategy is to transform the system $Ax = b$ into a problem of minimizing the residual $r_k = min\{b - Ax\}$ (Barrett et al. (1994)).

This paper presents two non-stationary iterative methods, Conjugate Gradient (CG) and the

Generalized Minimum Residue (GMRES). Both perform each iteration, an approximation to the solution of the system $x_i = x_0 + z$ where $z \in K_i = span\{r_0, Ar_0, A^2r_0, ..., A^{i-1}r_0\}$, define as the Krylov subspace, i is the dimension of subspace and $r_0 = b - Ax_0$ is the residual of the approximation x_0.

The GMRES method is applied to nonsymmetrical and sparse matrices. Its objective is to minimize the norm of the residual system, to fin an approximate solution that uses a z vector of the Krylov subspace. The solution of the system $x = x_0 + z$ such that $\|b - A(x_o + z)\|$ should be minimal. Algorithm 7 describes the GMRES(k) method. The idea is to build in this base an orthogonal basis of Krylov subspace U. It is built through the Modifie Gram-Schmidt process (lines 6-13). This process, in addition to generating the orthogonal U basis, generates Matrix H, known as the Hessemberg matrix. This matrix is almost an upper triangular except for a diagonal lying below the main diagonal. To eliminate this bias and make the system trivial, a process define as Givens rotation, lines 14-23, is accomplished by eliminating elements of the matrix. This process, through back substitution, then resolves the system $Hy = e$ (lines 26-27). Finally, in line 29 a new approach is calculated by considering the vector y and u obtained by multiplying the base U. By default, the algorithm GMRES is computationally costly, since for each iteration i, it should calculate and store the product matrix vector. For this reason, we implement the GMRES(k), where k is the size of the subspace and consequently the number of subspace vectors to be stored.

The conjugate gradient method, CG, is named for generating a sequence of vectors with a descending orthogonal direction. CG is applied when A is symmetric, $a_{ij} = a_{ji}$, and define positive, $\forall x \in \Re^n x^t Ax > 0$ and if $x = 0, x^t Ax = 0$. The basic idea of the CG method is that through the choice of n linearly independent directions, $v_1, v_2, \ldots, v_n$, and through the minimization of the function $F(x^{(j)} + \alpha v_j)$ in each direction, we construct a sequence of approximations that provides the minimal quadratic function $F(x) = \frac{1}{2}x^T Ax - b^T x$. Algorithm 8 describes the CG method.

Algorithm 8: **CG**

1 Data $A, b, Nmax, tol, x_0 = 0$
2 $v_0 = r_0 = b$
3 $\delta_{novo} = \|r_0\|_2^2, \delta_0 = \delta_{novo}, j = 0$
4 **while** $(j < Nmax) and (\delta_{novo} > tol^2 \delta_0)$ **do**
5 $p_j = Av_j$
6 $\alpha = \frac{\delta_{novo}}{v_j^t p_j}$
7 $x_{j+1} = x_j + \alpha v_j$
8 $r_{j+1} = r_j - \alpha p_j$
9 $\delta_{velho} = \delta_{novo}$
10 $\delta_{novo} = \|r_{j+1}\|_2^2$
11 $\beta = \frac{\delta_{novo}}{\delta_{velho}}$
12 $v_{j+1} = r_{j+1} + \beta v_j$
13 $j = j + 1$
14 **endw**

5 PRECONDITIONING

The convergence of iterative methods depends significantl on the spectrum (set of eigenvalues) of the coefficients matrix. When the eigenvalues approach zero or present different orders of magnitude, the matrix has a strong tendency for ill-conditioning (Thibes (2002)). To produce a well-conditioned matrix, a technique called preconditioning can firs be applied to the system. This technique aims to accelerate the convergence of iterative methods. It replaces the system $Ax = b$ with the equivalent system $M^{-1}Ax = M^{-1}b$, where M is called the preconditioner matrix. As we can see in Saad (2003), to fin a good preconditioner to solve a linear system is often viewed as a combination of art and science.

Algorithm 9: **ILU(0)**

for $i = 2, \ldots, n$ **do**
  for $k = 1, \ldots, i-1$ **do**
    if $(i,k) \in NZ$ **then**
      $a_{ik} = \dfrac{a_{ik}}{a_{kk}}$
    endif
    for $j = k+1, \ldots, n$ **do**
      if $(i,j) \in NZ$ **then**
        $a_{ij} = a_{ij} - (a_{ik} * a_{kj})$
      endif
    endfor
  endfor
endfor

Algorithm 10: **ILU(p)**

for $\forall a_{ij} \neq 0$**do**
  $Lev_{ij} = 0$
endfor
for $i = 2, \ldots, n$ **do**
  for $k = 1, \ldots, i-1$ **do**
    if $Lev_{ij}k$ **then**
      $a_{ik} = \dfrac{a_{ik}}{a_{kk}}$
    endif
    for $j = k+1, \ldots, n$ **do**
      $a_{ij} = a_{ij} - (a_{ik} * a_{kj})$
      $Lev_{ij} = min\{Lev_{ij}, Lev_{ik} + Lev_{kj} + 1\}$
    endfor
  endfor
  if $Lev_{ij} > k$ **then**
    $Lev_{ij} = 0$
  endif
endfor

Preconditioners are active during the iterative process. Those for M can be define as good preconditioners if they are a matrix close to A and satisfy two conditions: *(a)* $M^{-1}A$ is well conditioned; *(b)* the system $Mc = b$ is trivial (Cunha (2000)). Some of the most widely referenced preconditioners in the literature are based on the incomplete LU factorization (ILU). This is where the LU factorization is performed, but a fil level *p* is determined, thereby halting the factorization. Hence, if $p = 0$ there will be no fills If $p = 1$ there will be fill up to the firs level of the LU factorization and so on. In this work, we implement preconditioners based on the LU decomposition, where two algorithms are define for an ILU(0) and another for ILU(p) when $p \neq 0$. To construct this type of preconditioning, we do the usual factorization where the order of the loops controlled by the indices *i* (line), *j* (column) and *k* (step factoring) are crucial to the performance of the preconditioner. For example, the algorithm *k, i, j* is difficul to implement since in each step *k* must be modifie for all the remaining lines from $k + 1$ to n. The algorithm *i, k, j* however, is characterized, at every step i by replacing the values of the elements of row i of the coefficien matrix by triangular factors L and U (Saad (2003)). We can see in Algorithms 9 and 10 both preconditioners being implemented. In Algorithm 10 Lev_{ij} indicates the fil level of each element a_{ij} of the matrix assigned $Lev_{ij} = 0$ where $a_{ij} \neq 0$ and $Lev_{ij} = \infty$ where $a_{ij} = 0$.

6 COMPUTATIONAL IMPLEMENTATION

All algorithms implemented in this work used a matrix stored in a structure called *Compress Sparse Row* (CSR), Saad (2003). To choose the pseudo-peripheral vertex and the pseudo-diameter, we implemented the RCM and GPS algorithms using the heuristic proposed by George and Liu (1979). Please note that the RCM implemented in this paper considers the algorithm proposed by Cuthill and McKee (1969), with the reverse described by George (1971). The GPS algorithm was implemented as described in Gibbs et al. (1976). All the processes of sorting the algorithms required for RCM, GPS, ES and ND were performed using the *QuickSort* algorithm. The sparsity calculation is done as a percentage, i.e., $esp(A) = ((n*n) - nnz)/(n*n) * 100$, where n is the total number of vertices and nnz is the total number of edges of the graph. That is, n is the number of lines, and nnz is the number of non-zero elements of Matrix A. The envelope size and bandwidth are given as a percentage of how much they reduced from their initial size. To calculate the reduction:

$$red(env(A)) = 100 - ((env(A)_{\textit{after reordering}} * 100)/env(A)_{\textit{initial}}) \tag{3}$$

where $env(A)_{\textit{initial}}$ is the size of the original envelope and $env(A)_{\textit{after reordering}}$ is the size of the envelope after the reordering of Matrix A. The same calculation is done to reduce bandwidth ($red(lb(A))$ exchanging env $env(A)$ per $lb(A)$.

The matrices used in the computational tests are in the Matrix Market format as described by Boisvert et al. (1996) and available on the websites http://math.nist.gov/MatrixMarket/ and http://www.cise.ufl.edu/research/sparse/matrices/ These matrices come from a variety of application field such as computational fluid dynamics (CFD), electromagnetism, chemistry, and so forth. To calculate the eigenvalue and eigenvector associated required in the ES algorithm some libraries were tested: Clapack, Blopex, Trilinos and Chaco 2.0. The Chaco library was chosen due to optimized storage matrix and low processing time. We underscore the fact that, for this purpose, the Chaco implements two algorithms: Lanczos, suitable for small- and medium-sized matrices and Multilevel Symmlq/RQI used in this work, suitable for large-sized matrices(Hendrickson and Leland (1995)). This algorithm considers the RQI (Rayleigh Quotient Iteration) iterative method, which solves a linear system using the LQ decomposition in

each iteration, Barnad and Simon (1993). It is worth noting that the quality of solution obtained depends strongly on the tolerance considered for the iterative process. We also point out that we have changed the source code for Chaco. To reduce the processing time we calculate only the eigenvector associated with the second smallest eigenvalue.

7 COMPUTATIONAL TESTING

In this section we compare the reordering algorithms RCM, RCM-P1, RCM-P2, ND, ND-P3-P4 ND, GPS and ES (subsection 7.1) and present the action of these algorithms, when ILU(p) preconditioners , are applied to GMRES and CG iterative methods (subsection 7.2).

Table 1: Characteristics of the chosen set of matrices

Matrix (A)	Application Area	$a_{ij} = a_{ji}$	nnz	n	$esp(.)$
Na5	Quantum Chemistry	Sim	305630	5832	99,10
FEM_3D_t1	Thermal Problem	Não	430740	17880	99,87
wathen100	Random 2D/3D	Sim	471601	30401	99,95
wathen120	Random 2D/3D	Sim	565761	36441	99,96
qa8fm	Acoustics Problem	Sim	1660579	66127	99,96
Baumann	Chemical Problem	Não	748331	112211	99,99
FEM_3D_t2	Thermal Problem	Não	3489300	147900	99,98
thermomech_dK	Thermal Problem	Não	2846228	204316	99,99
cage13	DNA electrophoresis	Não	7479343	445315	99,99
tmt_sym	Electromagnetics	Sim	5080961	726713	99,99
atmosmodl	CDF Problem	Não	10319760	1489752	99,99

The metrics analyzed were the size of the envelope ($env(.)$), the bandwidth ($lb(.)$), the processing time (CPU), and the number of iterations of iterative methods. Tests were performed on a computer with an Intel DualCore 2.2GHz, with 3GB of RAM and Ubuntu 9.04 operating system.

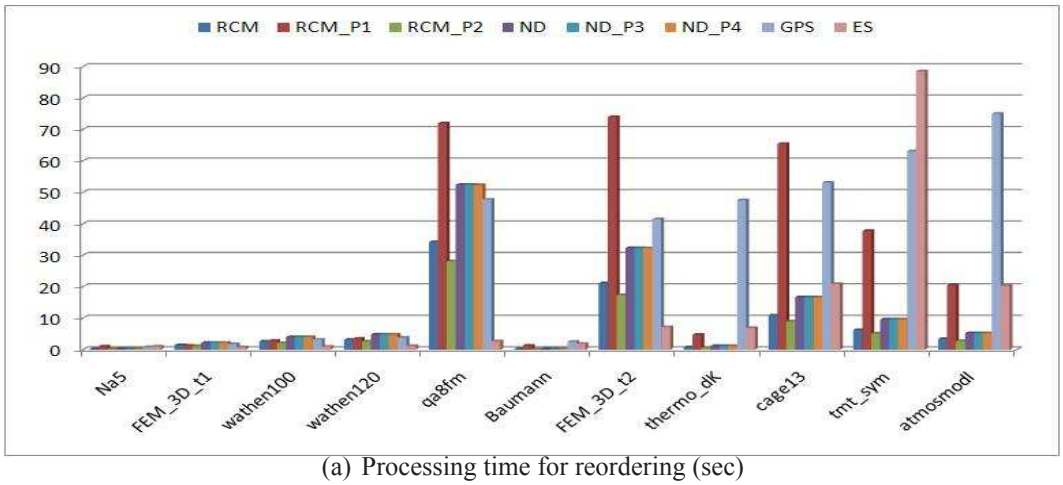
Table 1 shows the main features of the set of matrices chosen to examine the methods implemented. The columns, Matrix (A), Application Area, $a_{ij} = a_{ji}$, nnz, n and $esp(.)$ indicate, respectively, the name of the matrix, the area of application that generated the matrix, if the non-zero values of the matrix are symmetrical, non-zero number of elements, matrix size and the sparsity percentage, described in Section 6. We note that all the chosen matrices have a sparsity greater than 99%.

7.1 Analysis of the algorithms for reordering

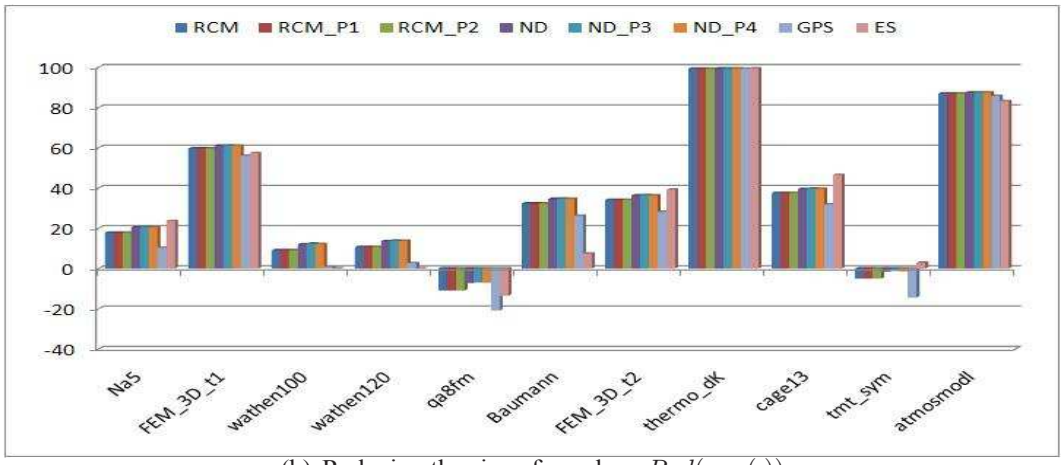
Figure 1(a) presents the CPU time (in seconds) of algorithms RCM, RCM-P1, RCM-P2, ND, ND-P3, ND-P4, GPS and ES for the set of matrices shown in Table 1. Figure 1(b) shows the reduction in size of the envelope $red(env(.))$ and Fig. 1(c), the reduction in size of the bandwith $red(lb(.))$ for all matrices analyzed. We emphasize that in Figure 1 the negative percentages indicate that the algorithms have not reduced the $(env(.))$ and $red(lb(.))$; in fact they have increased them.

We can observe from Fig. 1(a)that the processing times of the algorithm RCM-P1 for all matrices tested were greater than those obtained by the RCM. Such a result is caused by the need, in the RCM-P1, to run Dijkstra's algorithm for all vertices of the graph associated with the ma-

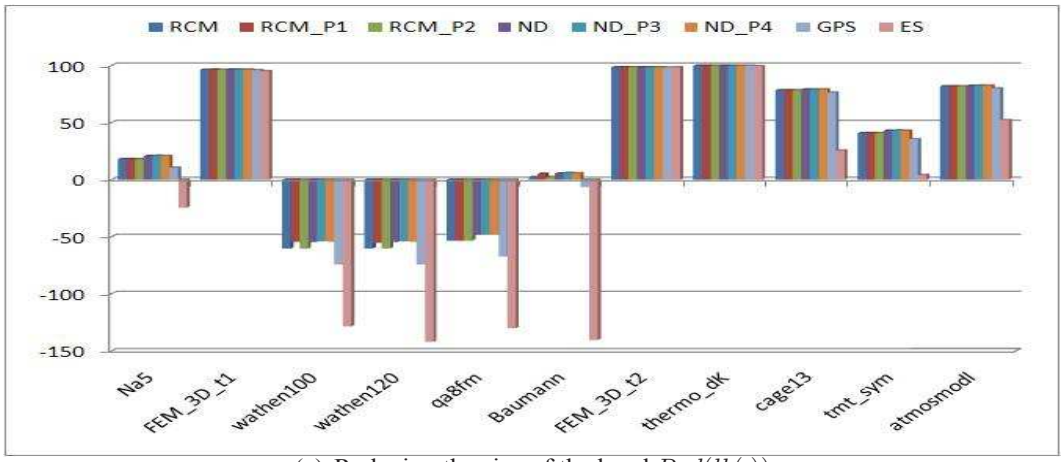
trix, increasing its computational complexity from $O(nnz * log(n))$ to $O(n * (nnz * log(n)))$. The cases with comparable times occur when the heuristic to fin the initial vertex (step 1 of Algorithm RCM) is performed at worst case. We further note that in 85% of the tests, the reduction achieved by algorithm RCM-P1 equaled that of the RCM algorithm for $env(.)$ (Fig. 1(b))



(a) Processing time for reordering (sec)



(b) Reducing the size of envelope $Red(env(.))$



(c) Reducing the size of the band $Red(lb(.))$

Figure 1: Processing time and reducing the size of the envelope and the band's set of matrices analyzed for algorithms RCM, RCM-P1, RCM-P2, ND, ND-P3, ND-P4, GPS, and ES

and $lb(.)$ (Fig. 1(c)). This fact occurs because, for most matrices tested, Algorithm 2 find the vertex of greatest associated number (peripheral). The proposed RCM-P2 algorithm has, for all tested matrices, a smaller processing time than does the RCM but with a quality of solution similar to it. The reduction in processing time may be explained by no long needing to execute the breadth-firs search of the RCM algorithm. Since Algorithm 3 is directly applied to the graph with vertices sorted in increasing order of degrees, making the level structure obtained equivalent to breadth-firs search of RCM algorithm. The ND algorithm has a processing time,

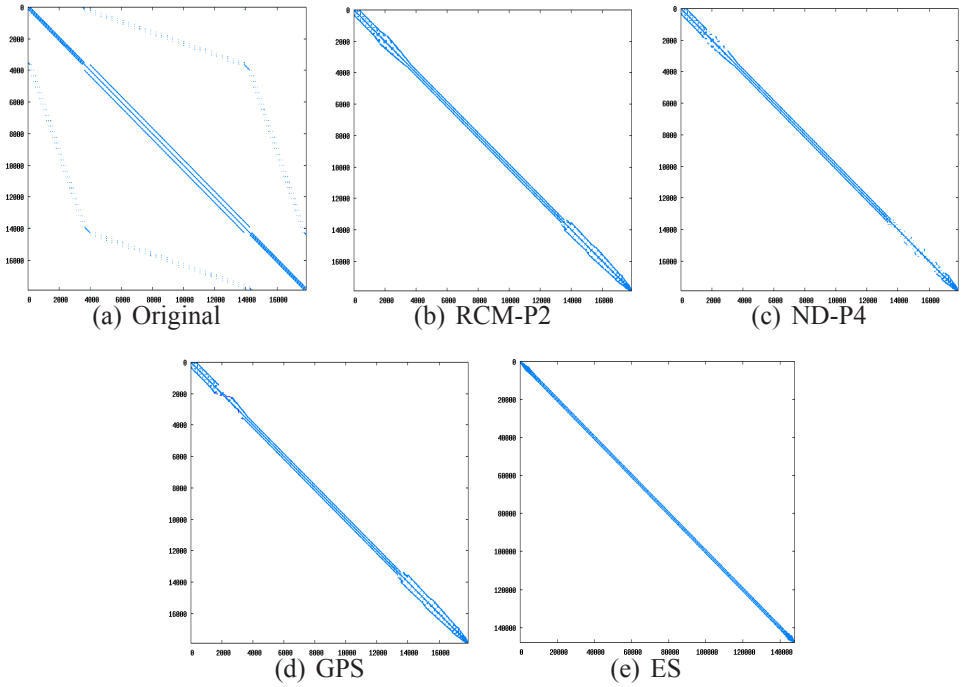


Figure 2: Matrix configuratio of the FEM-3D-t1 for algorithms RCM-P2, ND-P4, GPS e ES

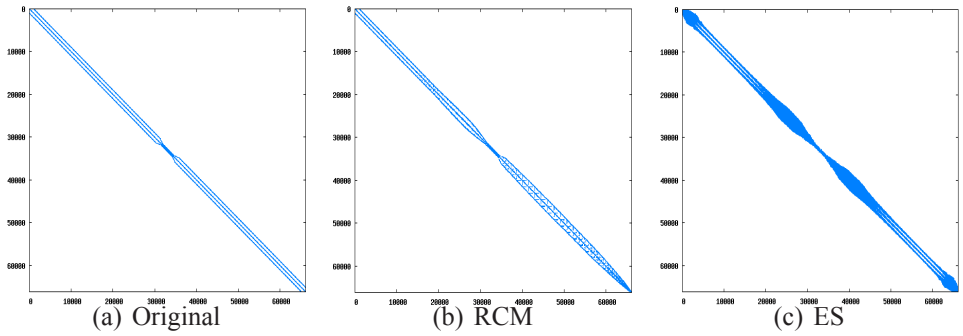


Figure 3: Matrix configuratio of the qa8fm for algorithms RCM e ES

on average, around 50% higher than that of the RCM, showing a slight improvement in quality of solution. We observed that the processing times of the proposed adaptations (algorithms ND-P3 and ND-P4) correspond to those of ND, with the ND-P4 being slightly faster. This fact may be explained by it no longer being necessary to identify all the vertices' cuts. In this case, the extreme vertices of all cutting edges were used. A small improvement in solution quality was also observed. We would also emphasize that the solution quality and the processing

time of algorithms ND, ND-P3, and ND-P4 were directly related to the graph partitioning algorithm used. The GPS algorithm was much slower than the RCM, with no significan difference in solution quality. This fact may be explained by the fact that the second step of the GPS algorithm is computationally more costly and requires more memory usage. For some of the tested matrices, the ES algorithm has shorter processing times than the RCM (FEM_3D_t1, wathen120, qa8fm, FEM_3D_t2) and longer processing times than some of the others (tmt_sym, atmosmodl). Cases where the ES outperformed the others are characterized by matrices with less sparsity . In general, we observed that the algorithms based on search strategies in width (RCM, RCM-P1, RCM-P2, ND, ND-P3, ND-P4 and GPS) were extremely sensitive to the sparsity of the matrix. This behavior may be explained by the computational complexity of the breadth-firs search, which is $O(n + nnz)$. For ES, on the other hand, the computational bottleneck is the computation of the eigenvalue and eigenvector. Besides being a costly operation, the greater the precision required for components of the eigenvector, the greater the calculating processing time and the better the quality of the solution. The greater the precision, the greater the possibility that components of the eigenvector are distinct, a fact contributing, due to the results' better ordering, to the solution quality's improvement.

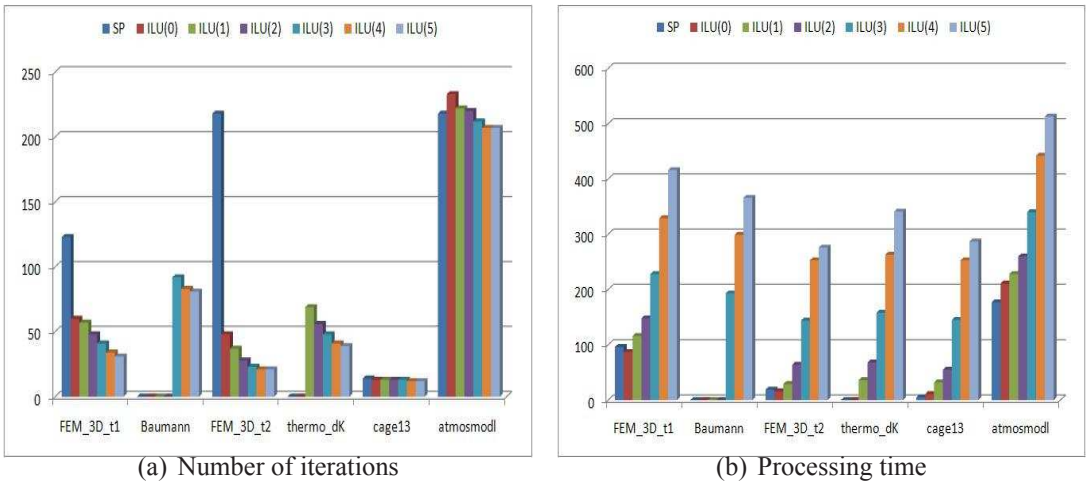
Figure 2(a) shows the original configuratio of the sparse matrix FEM_3D_t1 and Figures 2(b), 2(c), 2(d), 2(e) show, respectively, the configuration obtained by the algorithms RCM-P2, ND-P4, GPS, and ES. We can observe that all algorithms reduce the bandwidth and the envelope matrix. In Figures 1(b) and 1(c) however, we fin a group of matrices that, for all algorithms tested, the quality of the initial solution obtained was worse, such as Matrix qa8fm. For the breadth-firs search-based algorithms, we noticed this actually happens for graphs with label sequences next to each other. We fin in the ES algorithm no improvement in quality of solution when the values of eigenvector components are very close to each other. This behavior comes up when the distances between the vertices are small. Figure 3 shows the original configuratio of Matrix qa8fm and settings for the RCM and ES algorithms. We observe here that there was no reduction in bandwidth or envelope.

7.2 Analysis of ILU(p) Preconditioners

Figures 4(a), 5(a) and 6(a) show the number of iterations for the GMRES(k) method without preconditioning and with preconditioner ILU(p), $p = 0, 1, 2, 3, 4, 5$, respectively, without reordering, with RCM-P2 and ES for a set of nonsymmetrical matrices presented in Table 1. Figures 4(b), 5(b) and 6(b) show the processing time for the same set of matrices.

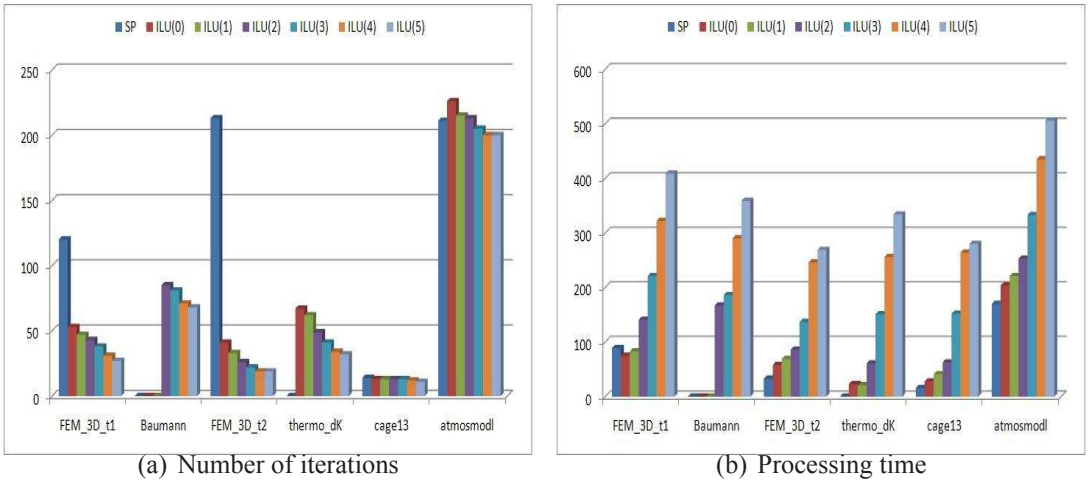
When the system is solved without applying a reordering algorithm, Figure 4, we found that the number of iterations performed in most cases was greater without preconditioning. We must point out, however, that in applying preconditioning, reducing the number of iterations does not necessarily imply a faster processing time. The time reduction is significan when there is a big reduction in number of iterations, as is the case with matrices FEM_3D_t1 and FEM_3D_t2 for the preconditioner ILU(0), The preconditioners with $p > 0$ and without reordering were not efficien in terms of processing time. This inefficien y is due to the computational cost of obtaining M given that $lb(A)$ and $env(A)$ are, most often, higher without the reordering, most of the time. We also found that Baumann matrices and thermo_dk, start to converge only after applying preconditioners ILU(3) and ILU(1), respectively.

In Figures 5 and 6, we analyze the RCM-P2 and ES algorithms, respectively. Compared to the results without reordering (Figure 4), the number of iterations and processing time were, by and large, reduced. In reordering RCM-P2 (Figure 5) note that the Baumann matrix is converging with the preconditioner ILU(2) and the thermo_dk is beginning to do so with the pre-



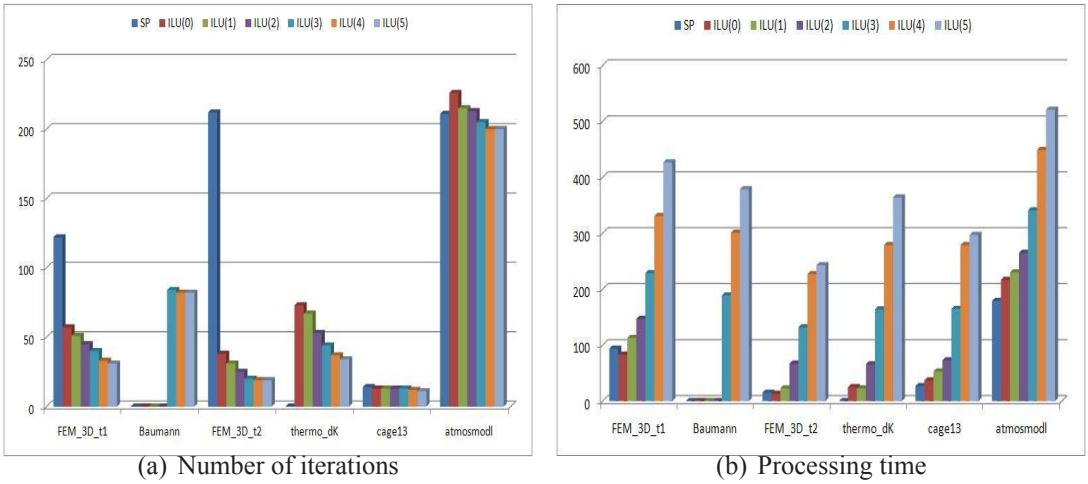
(a) Number of iterations (b) Processing time

Figure 4: Number of iterations and processing time for the GMRES(k) algorithm without reordering



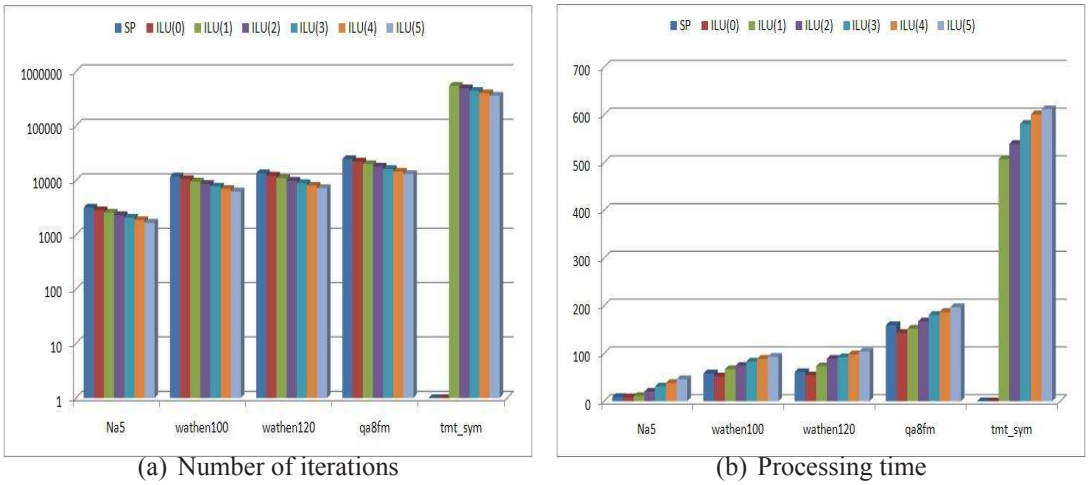
(a) Number of iterations (b) Processing time

Figure 5: Number of iterations and processing time for the GMRES(k) algorithm with reordering RCM-P2



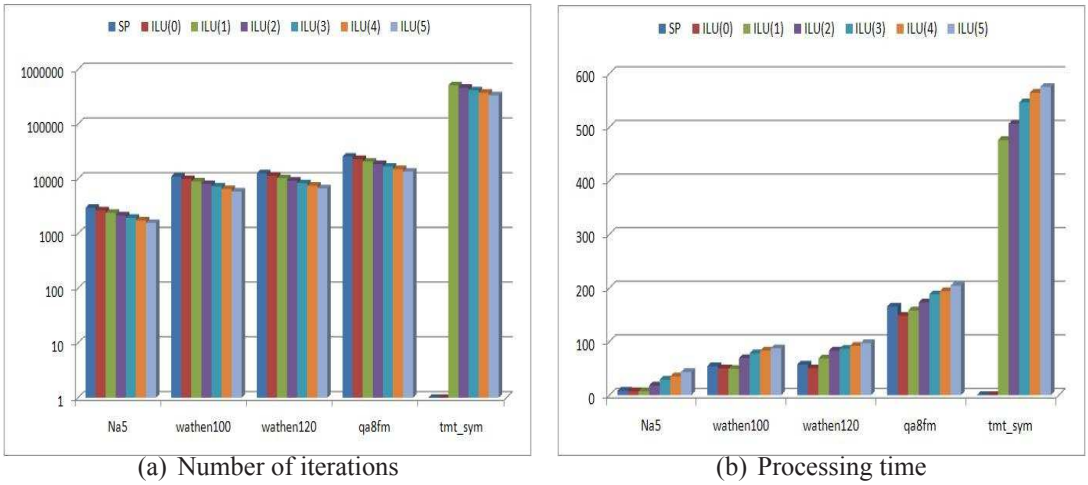
(a) Number of iterations (b) Processing time

Figure 6: Number of iterations and processing time for the algorithm GMRES(k) with reordering ES



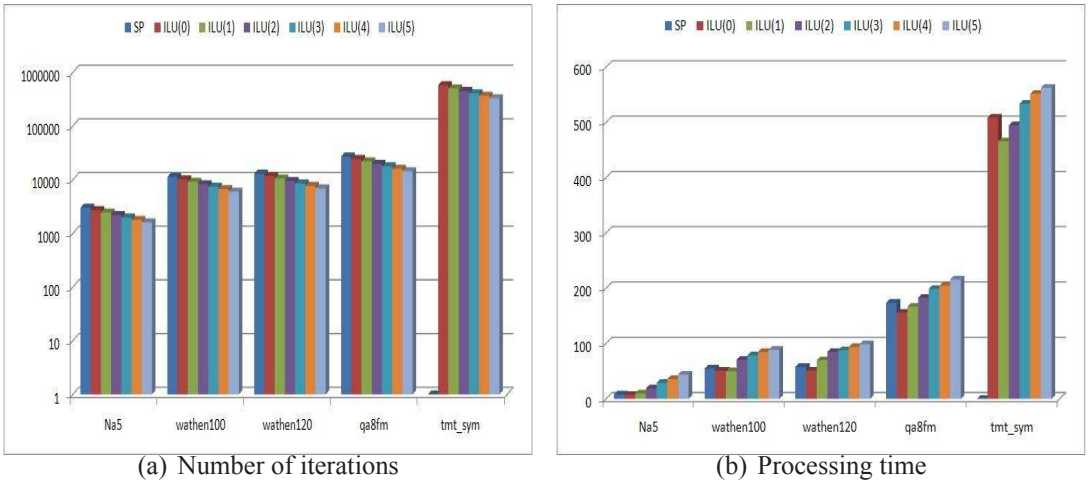
(a) Number of iterations (b) Processing time

Figure 7: Number of iterations and processing time for the CG algorithm without reordering



(a) Number of iterations (b) Processing time

Figure 8: Number of iterations and processing time for the CG algorithm with reordering RCM-P2



(a) Number of iterations (b) Processing time

Figure 9: Number of iterations and processing time for the CG algorithm with reordering ES

conditioner ILU(0). The matrix FEM_3D_t2 lowers the number of iterations, but increases the processing time, since the algorithm reduces the RCM-P2 $env(A)$ and $lb(A)$, but finishin the reordering requires a long processing time. In the ES algorithm for matrix FEM_3D_t2, the number of iterations and processing time are reduced, since for this matrix the ES reordering is fast.

Figures 7(a), 8(a) and 9(a) show the number of iterations for the CG method without preconditioning and with ILU(p), $p = 0, 1, 2, 3, 4, 5$, respectively, without reordering, with RCM-P2 and ES for the set of symmetrical matrices presented in Table 1. Figures 7(b), 8(b) and 9(b) show the processing time for the same set of matrices. We should point out that, to help with visualization, the number of iterations is on a logarithmic scale; indeed, the values were far and wide. The results show that when reordering is implemented a slight reduction in both the number of iterations and processing time results. As was expected, the reduction in the number of iterations and processing time for the ES algorithm was, because of its reordering, smaller (80% of tests) than that of the RCM-P2 algorithm. The tmt_sym matrix only converges to the ILU(1) without reordering but, when ES is considered, converges to ILU(0). This occurs once the ES is the only algorithm that can reduce the env to that matrix. As for the qa8fm matrix, there was no reduction in the number of iterations or in the processing time because in none of the reordering algorithms was there a reduction in the $env(A)$ and $lb(A)$, as noted in section 7.1

8 CONCLUSIONS AND FUTURE WORK

Having taken from the literature four major matrix-reordering algorithms – RCM, GPS, ND and ES – this paper compared their solution qualities and processing times. We proposed modification to the RCM (RCM-P1 and RCM-P2) and the ND (ND-P3 and ND-P4) and, for the CG and GMRES methods, studied the influenc of reordering algorithms on ILU(p) preconditioners. We carried out computational tests on a set of 11 large-sized sparse matrices with sparsities over 99%.

Our modificatio to the RCM algorithm, RCM-P2, produced, for all cases tested, the lowest processing time with a quality of solution identical to that of RCM. Our modificatio to the ND algorithm, ND-P4, showed the best results of all other ND algorithms, though it was slower than the RCM. The ES algorithm, only for less sparse matrices, had a solution comparable to that of RCM and with a better processing time. For all matrices tested, GPS had a faster processing time than the RCM. This is due to the fact that minimizing the level width (step 2 of Algorithm 4)is more costly, computationally, than performing a breadth-firs search (lines 4-14 of Algorithm 1). These facts justify the widespread use of the RCM algorithm for reordering sparse matrices. We also note that the proposed modification presented here showed better solutions, considering the balance between solution quality and computational time, than those obtained by standard algorithms.

The ILU(p) preconditioners with $p > 1$ tend to have computational costs high taking long processing times and heavy memory consumption. On the other hand, this increases the possibility of convergence, a fact explained by the proximity of M the A with the increasing of p. We stress that for well-conditioned matrices, in most cases, the use of preconditioning does not help reduce processing time, while matrix reordering improves the conditioning of the matrix, influencin the number of iterations and reducing the processing time.

For future work, we want to investigate the behavior of Minimum Degree and ant colony algorithms and apply these to the problem of matrix reordering. Another study provided concerns choosing the initial vertex of the RCM algorithm: one can vary the choice of this vertex based on

the vertices of minimum, maximum, and random degrees and investigate other preconditioners.

REFERENCES

Barnad S., Photen A., and Simon H. A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Aplications*, 3:317–334, 1995.

Barnad S. and Simon H. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. In *Proceedings of the 6th SIAM Conf. Parallel Processing for Scientific Computing*, pages 711–718. SIAM, 1993.

Barrett R., Berry M., Chan T.F., Demmel J., Donato J., Dongarra J., Eijkhout V., Pozo R., Romine C., and der Vorst H.V. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, 1994.

Benzi M., Szyld D.B., and van Duin A. Ordering for incomplete factorization preconditioning of nonsymmetric problems. *SIAM J. Sci. Comput*, 20(5):1652–1670, 1999.

Boisvert R., Pozo R., and Remington K. The matrix market exchange formats: Initial design. Technical report, National Institute of Standards and Technology, 1996.

Camata J., Rossa A., Valli A., Catabriga L., Carey G., and Coutinho A. Reordering in incomplete factorization preconditioning for serial and parallel amr solutions, 2010. Applied to *Int. J. Numer. Meth. Engng.*

Caprara A. and Salazar G. Laying out sparse graphs with provably minimum bandwidth. *INFORMS Journal on Computing*, 17(3):356–373, 2005.

Carvalho M., Junqueira N., and Soma N. Uma heurística para o problema de minimização de largura de banda em matrizes. *Pesquisa operacional*, pages 2886–2897, 2009.

Coleman T. *Large sparse numerical optimization*. Springer-Verlag Berlin Heidelberg, 1984.

Cunha C. *Métodos numéricos*. Editora da Unicamp, Segunda edição, New York, NY, USA, 2000.

Cuthill E. and McKee J. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 24th ACM national conference*, pages 157–172. ACM Press, New York, NY, USA, 1969.

Dueck G. and Jeffs J. A heuristic bandwidth reduction algorithm. *Journal of Combinatorial Mathematics and Combinatorial Computing*, pages 97–108, 1995.

Fiduccia C. and Mattheyses R. A linear-time heuristic for improving network partitions. In *Proceedings of the IEEE 19th Design Automation Conference,*, pages 175–181. 1982.

Fiedler M. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

George A. Nested dissection of a regular finit element mesh. *SIAM Journal of Numerical Analysis*, 10:345–363, 1973.

George A. and Liu W. An implementation of a pseudoperipheral node finde . *ACM Transactions on Mathematical Software*, 5:236–250, 1979.

George J.A. *Computer implementation of the finite element method*. Ph.D. thesis, Stanford University, Stanford, CA, USA, 1971.

Gibbs N.E., Poole W.G., and Stockemeyer P.K. An algorithm for reducing the bandwidth and profil of a sparse matrix. *SIAM Journal of Numerical Analysis*, 13(2):236–250, 1976.

Hendrickson B. and Leland R. The chaco user's guide version 2.0. Technical Report SAND-95-2344, Sandia National Laboratories, New Maxico, USA, 1995.

Lim A., Rodrigues B., and Xiao F. Heuristics for matrix bandwidth reduction. *European Journal of Operational Research*, 174:69–91, 2006.

Liu W. and Sherman A. Comparative analysis of the cuthill-mckee and the reverse cuthill-mckee

ordering algorithms for sparse matrices. *SIAM J. Num. Anal.*, 13:198–213, 1976.

Martí R., Campos V., and Piñana E. A branch and bound algorithm for the matrix bandwidth minimization. *European Journal of Operational Research*, 186(2):513–528, 2008.

Marti R., Laguna M., Gloves F., and Campos V. Reducing the bandwidth of a sparse matrix with tabu search. *European Journal of Operational Research*, 135:450–459, 2001.

Menezes I. *Técnicas de reordenação para solução de sistemas esparsos*. Ph.D. thesis, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, 1995.

Narsingh D. *Graph theory with applications to Engineering and Computer Science*. Prentice-Hall, 1974.

Papadimitriou C. The np-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.

na04

Piñana E., Plana I., Campos V., and Marti R. Grasp and path relinking for the matrix bandwidth minimization. *European Journal of Operacional Research*, 153:200–210, 2004.

Portugal C., Pessanha J., and Saavedra O. Investigação crítica do desempenho do gmres pré-condicionado via fatoração incompleta lu em estudos de flux de carga. *Sba Controle & Automação*, 20:564–572, 2009.

Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS Publishing, Second edition, Boston, 2003.

Thibes H. *Um estudo da fatoração incompleta LU e Cholesk como précondicionadores nos métodos iterativos*. Ph.D. thesis, Programa de Pós Graduação em Matemática Aplicada, Universidade Federal do Rio Grande do Sul, 2002.

Wang Q., Guo Y., and Shi X. A generalized gps algorithm for reducing the bandwidth and profil of a sparse matrix. *Progress In Electromagnetics Research*, 153(PIER 90):121–136, 2009.