LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2020

A comparison of pseudoperipheral vertex finders for the Reverse Cuthill-McKee method

A. A. A. M. Abreu
Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina - IFSC
Canoinhas, SC
alexandre.abreu@ifsc.edu.br

J. A. B. Bernardes, S. L. Gonzaga de Oliveira
Universidade Federal de Lavras - UFLA
Lavras, MG
{juniorbernardes,sanderson}@ufla.br

ABSTRACT

Previous publications reviewed several pseudoperipheral vertex finders in graphs for the Reverse Cuthill-McKee method. Based on this experience, this paper evaluates the George-Liu algorithm against three other methods for solving the problem of finding a proper pseudoperipheral vertex in a graph. The results show that the George-Liu algorithm remains in the state of the practice for providing pseudoperipheral vertices for the Reverse Cuthill-McKee method when applied to matrices with symmetric sparsity patterns.

KEYWORDS. Bandwidth reduction. Sparse matrices. Graph labeling.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

1. Introduction

Various applications in modern engineering require the analysis and solution of large and complex problems defined by a set of linear equations in the form $Ax = b$, where $A = [a_{ij}]$ is an $n \times n$ large-scale sparse matrix, b is a known n-vector, and x is an unknown n-vector. Some techniques can be employed to reduce the computational efforts in solving systems of linear equations with large-scale sparse matrices. Particularly, an efficient solution employs a proper order for the variables of the problem. Reordering the row and columns of the matrix A is equivalent to reorder the equations in the case of using one degree-of-freedom per node in the finite-element analysis. Similarly, reordering the row and columns of the matrix A is equivalent to reorder the equations in partial differential equation finite-volume discretizations.

A technique commonly used to reduce the computational efforts in solving systems of linear equations intend to reduce matrices bandwidths. Furthermore, applications reduce the execution costs of iterative linear system solvers by applying a heuristic for bandwidth reduction (see Gonzaga de Oliveira et al. [2018b] and references therein). The transfer of information to and from memory is associated with the number of non-null coefficients in the rows of the matrix system A. The algorithm reduces cache miss rates if the non-null coefficients in each row lie in the same level of the memory hierarchy. In this context, cache hits depend on the bandwidth of A [Gonzaga de Oliveira e Abreu, 2018a].

Among the algorithms that have been proposed to reduce matrices bandwidths, the Reverse Cuthill-McKee (RCM) method is possibly the most successful low-cost heuristic. Systematic reviews [Gonzaga de Oliveira e Chagas, 2015; Gonzaga de Oliveira et al., 2018b,a; Gonzaga de Oliveira e Abreu, 2018b,a] reported the Reverse Cuthill-McKee method with starting vertex given by the George-Liu (GL) algorithm [George e Liu, 1981] as one of the most successful low-cost heuristics for bandwidth reductions. The Reverse Cuthill-McKee method is employed in this present computational experiment because this algorithm is the most commonly used vertex renumbering strategy used to reduce the bandwidth of matrices (see Gonzaga de Oliveira et al. [2018b,a]; Gonzaga de Oliveira e Abreu [2018a] and references therein). Indeed, the RCM-GL method is possibly the most frequently used graph labeling strategy used in the literature (e.g., see Gonzaga de Oliveira e Chagas [2015]; Gonzaga de Oliveira et al. [2018b]; Boost [2017]; Gonzaga de Oliveira et al. [2018a] and references therein). Few algorithms can compete with the RCM-GL method [George e Liu, 1981] in terms of bandwidth results when taking computing times into account [Gonzaga de Oliveira et al., 2018b,a] because the method yields consistent bandwidth results at a low cost. This may be the reason why this method is available on the MATLAB [The MathWorks, Inc., 1994–2020] and GNU Octave [Eaton et al., 2015] mathematical softwares as the function *symrcm*[1], and on Boost C++ Library [Boost, 2017][2].

The Reverse Cuthill-McKee method belongs to a class of heuristics known as level set reorderings, where the procedure labels the vertices in a graph partitioning them into level sets. In this process, the algorithm takes into account the distance from a given starting vertex. Thus, the selection of the starting vertex significantly influences the quality of the resulting ordering delivered by the algorithm. In general, the method yields better bandwidth results when the width of a level structure rooted at the starting vertex is narrow, and its eccentricity is very close to the diameter of the graph (see definitions of these properties below) [Gonzaga de Oliveira e Abreu, 2018a].

Previous publications [Gonzaga de Oliveira e Abreu, 2018b,a] showed that the George-

[1] https://www.mathworks.com/help/matlab/ref/symrcm.html?requestedDomain=www.mathworks.com, https://octave.sourceforge.io/octave/function/sym rcm.html.

[2] http://www.boost.org/doc/libs/1_38_0/libs/graph/doc/cuthill_mckee_order ing.html.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

Liu algorithm dominated several algorithms for finding pseudoperipheral vertices for the Reverse Cuthill-McKee method. Along the same lines, this present paper appraises the George-Liu algorithm against a method based on evaluating the diagonal dominance of the matrix A [Zhang et al., 2013] and two other variants of pseudoperipheral vertex finders proposed by Kaveh [Kaveh, 1986]. Specifically, this paper compares the performance of the Reverse Cuthill-McKee method when varying the initialization vertex scheme. To this end, this paper uses the George-Liu (GL) algorithm and two proposed algorithms (AVKB and MKD) that are variants of the Kaveh's B and D algorithms [Kaveh, 1986]. The criterion used to analyze the results was the bandwidth reduction. After that, the GL algorithm was compared with the diagonal dominance (DD) algorithm Zhang et al. [2013], both joined with the Reverse Cuthill-McKee method, in the context of improving the performance of the the zero-fill incomplete Cholesky preconditioned conjugate gradient method when applied to some systems of linear equations with sparse matrices.

In Section 2, background information is reviewed. In Section 3, the Reverse Cuthill-McKee method is described. In Section 4, the algorithms evaluated in this computational experiment are reported. In Section 5, the simulations conducted in this study are described. In Section 6, the results are discussed. Finally, in Section 7, the conclusions are addressed.

2. Definitions and notation

Let $G = (V, E)$ be a connected undirected graph, where V and E are sets of vertices and edges, respectively. 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 defined as $\beta(G) = \max_{v \in V}[(\{v, u\} \in E)\ |s(v) - s(u)|]$, where $s(v)$ and $s(u)$ are labels of vertices v and u, respectively. Equivalently, define $A = [a_{ij}]$ as a symmetric $n \times n$ matrix associated with a connected undirected graph $G = (V, E)$. The bandwidth of row i of matrix A is given by $\beta_i(A) = i - \min_{1 \leq j \leq i}[j : a_{ij} \neq 0]$ and the bandwidth of a matrix A is defined as $\beta(A) = \max_{1 \leq i \leq n}[\beta_i(A)]$. The bandwidth minimization problem is $\mathcal{NP}$-hard [Papadimitriou, 1976] and, since the mid-1960s, several heuristics for bandwidth reductions have been proposed [Gonzaga de Oliveira e Chagas, 2015].

Many algorithms for bandwidth reductions are based on a data structure known as *rooted level structure*. Given a vertex $v \in V$, the level structure $\mathscr{L}(v)$ rooted at vertex v, with depth $\ell(v)$, is the partitioning $\mathscr{L}(v) = \{L_0(v), L_1(v), \ldots, L_{\ell(v)}(v)\}$, where $L_0(v) = \{v\}$ and $L_i(v) = Adj(L_{i-1}(v)) - \bigcup_{j=0}^{i-1} L_j(v)$, for $i = 1, 2, 3, \ldots, \ell(v)$, and $Adj(U) = \{w \in V : (u \in U \subseteq V)\ \{u, w\} \in E\}$. In particular, $\ell(v) = \max_{u \in V}[d(v, u)]$ denotes the *eccentricity* of the vertex v and the *distance* $d(v, u)$ is the length of a shortest path connecting vertices v and u. The *width* of a rooted level structure $\mathscr{L}(u)$ is defined as $b(\mathscr{L}(u)) = \max_{0 \leq i \leq \ell(u)}[|L_i(u)|]$.

3. The Reverse Cuthill-McKee method

The Reverse Cuthill-McKee method uses a level structure rooted at a vertex selected by a pseudoperipheral vertex finder. Algorithm 1 shows the pseudocode of the Reverse Cuthill-McKee method. The method receives as input the graph $G = (V, E)$ and a starting vertex v.

The Reverse Cuthill-McKee method is a greedy variant of the breadth-first search procedure, where the ordering chosen to reduce the bandwidth of a matrix is obtained by labeling the vertices in increasing-distance order from the starting vertex (see line 2 in Algorithm 1), and it labels vertices from an adjacency list in ascending-degree order (see line 6). Then, the method reverses the final labeling (see line 10).

As mentioned, the Reverse Cuthill-McKee method labels the vertices of the graph by increasing distance from a chosen starting pseudoperipheral vertex. Similar to the rooted level

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

structure defined above, ordering the vertices in this way partitions the vertices into level sets that are at the same distance from the starting pseudoperipheral vertex. Hence, considering the rooted level structure $\mathscr{L}(v)$, a small width $b(\mathscr{L}(v))$ is desirable, and this characteristic depends on the choice of an adequate starting vertex. Therefore, the bandwidth reduction provided by the Reverse Cuthill-McKee method depends on the starting vertex.

Input: connected graph $G = (V, E)$; starting vertex $v \in V$;
Output: reordering $S = \{s(1), s(2), \cdots, s(|V|)\}$;
1 **begin**
2 $s(1) \leftarrow v$;
3 $i \leftarrow 1$;
4 $j \leftarrow 1$;
5 **while** $i < |V|$ **do**
6 **foreach** *vertex* $w \in Adj(G, s(j)) - \{s(1), \cdots, s(i)\}$, *in order of increasing degree* **do**
7 $i \leftarrow i + 1$;
8 $s(i) \leftarrow w$;
9 $j \leftarrow j + 1$;
10 **return** *S in reversed order*;

Algorithm 1: Reverse Cuthill–McKee method [George e Liu, 1981].

4. Pseudoperipheral vertex finders evaluated

The Algorithm 2 shows the pseudocode of the George-Liu algorithm. The algorithm begins with an arbitrary vertex v (see line 2 in Algorithm 2) and builds its rooted level structure $\mathscr{L}(v)$ (see line 3). Then, the George-Liu algorithm builds the rooted level structure $\mathscr{L}(u)$ of a vertex u with the minimum degree belonging to $L_{\ell(v)}(v)$ (see lines 5 and 6). If $\ell(u) > \ell(v)$ (see lines 7), then u is attributed to v (i.e. $v \leftarrow u$) as well as $\mathscr{L}(v) \leftarrow \mathscr{L}(u)$ is computed (see line 8), and the process is repeated; otherwise, the process terminates and v is the pseudoperipheral vertex found (see line 10 in Algorithm 2).

The George-Liu algorithm is available on the MATLAB software [The MathWorks, Inc., 1994–2020], in conjunction with the Reverse Cuthill-McKee method. This algorithm obtains low computational costs in practical tests [Gonzaga de Oliveira et al., 2018b,a].

We evaluate another existing pseudoperipheral vertex finder in this study. Considering the matrix A, Zhang et al. [2013] selected the row $\min\limits_{1 \leq i \leq n} \left\lceil \frac{|a_{ii}|}{\sum\limits_{j=1, j \neq i}^{n} |a_{ij}|} \right\rceil$ with the lowest measure of diagonal dominance (DD) as the starting vertex for the Reverse Cuthill-McKee method, i.e., the last vertex in the Reverse Cuthill-McKee ordering. The time complexity of the DD algorithm is $\mathcal{O}(|V| + |E|)$. The reason is that the method evaluates each line (vertex) and column (edge) of the matrix A (underlying graph $G = (V, E)$) once. In particular, we refer to the DD algorithm together with the Reverse Cuthill-McKee method as the RCM-DD method.

Zhang et al. [2013] showed that this approach yielded better results than selecting the highest measure of diagonal dominance. The authors used the DD algorithm when providing starting vertices for the Reverse Cuthill-McKee method. However, the authors did not evaluate the results yielded by the RCM-DD method against the RCM-GL method.

Input: graph $G = (V, E)$;
Output: pseudoperipheral vertex $v \in V$;
1 **begin**
2 $v \leftarrow ArbitraryVertex(V)$;
// build a rooted level structure
3 $\mathscr{L}(v) \leftarrow$ Breadth-First-Search(v);
4 **repeat**
5 $u \leftarrow MinimumDegreeVertex(L_{\ell(v)}(v))$;
// build a rooted level structure
6 $\mathscr{L}(u) \leftarrow$ Breadth-First-Search(u);
7 **if** $\ell(u) > \ell(v)$ **then**
8 $v \leftarrow u$; $\mathscr{L}(v) \leftarrow \mathscr{L}(u)$;
9 **until** $u \neq v$;
10 **return** v;

Algorithm 2: George-Liu algorithm [George e Liu, 1981].

A previous publication [Gonzaga de Oliveira e Abreu, 2018a] showed that the George-Liu algorithm dominated Kaveh's B algorithm [Kaveh, 1986]. Thus, we included a slight variant of this algorithm in the experiments. Algorithm 3 shows the pseudocode of a variant of Kaveh's B algorithm. The original Kaveh's B algorithm begins with a vertex with the minimum degree. Instead, the variant begins with an arbitrary vertex v (see line 2 in Algorithm 3), as the George-Liu algorithm performs. Afterward, the algorithm builds the rooted level structure $\mathscr{L}(v)$ (see line 3). Then, the algorithm builds the rooted level structure $\mathscr{L}(u)$ of each vertex u belonging to $L_{\ell(v)}(v)$ (see lines 4 and 5). If $b(\mathscr{L}(u))$ is smaller than $b(\mathscr{L}(v))$ (see line 6), then u is attributed to v (i.e. $v \leftarrow u$) as well as $\mathscr{L}(v) \leftarrow \mathscr{L}(u)$ is computed (see line 7), and the process is repeated; otherwise, the process terminates and v is the pseudoperipheral vertex found (see line 8 in Algorithm 3). We refer to this variant as the Arbitrary-Vertex Kaveh's B (AVKB) algorithm.

Input: graph $G = (V, E)$;
Output: pseudo-peripheral vertex $v \in V$;
1 **begin**
2 $v \leftarrow$ ArbitraryVertex(V);
// build the rooted level structure $\mathscr{L}(v)$
3 $\mathscr{L}(v) \leftarrow$ Breadth-First-Search(v);
4 **foreach** vertex $u \in L_{\ell(v)}(v)$ **do**
// build the rooted level structure $\mathscr{L}(u)$
5 $\mathscr{L}(u) \leftarrow$ Breadth-First-Search(u);
6 **if** $b(\mathscr{L}(u)) < b(\mathscr{L}(v))$ **then**
7 $v \leftarrow u$; $\mathscr{L}(v) \leftarrow \mathscr{L}(u)$;
8 **return** v;

Algorithm 3: A variant of Kaveh's B algorithm.

Kaveh's D algorithm [Kaveh, 1986] selects the vertex that forms the rooted level structure with the narrowest width. The time complexity of this algorithm is $\mathcal{O}(|V|(|V| + |E|))$. Thus,

this computational experiment evaluates a low-cost algorithm based on Kaveh's D algorithm. Algorithms 4 shows the pseudocode of the variant. Unlike Kaveh's D algorithm, the variant terminates as soon as the width of a rooted level structure associated with a vertex is not smaller then the width of the narrowest rooted level structure found so far (see line 6). We refer to this variant as the Modified Kaveh's D (MKD) algorithm.

Input: graph $G = (V, E)$;
Output: pseudoperipheral vertex $u \in V$;
1 **begin**
  // v_1 is the first vertex established in the
2 $u \leftarrow v_1 \in V$; // entry file of the instance $G = (V, E)$
  // build the rooted level structure $\mathscr{L}(u)$
3 $\mathscr{L}(u) \leftarrow$ Breadth-First-Search(v_1);
  // vertices are simply evaluated sequentially
  // in the order set in the instance
4 **for** $i \leftarrow 2; i \leq |V|; i \leftarrow i + 1$ **do**
    // build the rooted level structure $\mathscr{L}(v_i)$
5    $\mathscr{L}(v_i) \leftarrow$ Breadth-First-Search(v_i);
    // go to line 9 if the condition is satisfied
6   **if** $b(\mathscr{L}(v_i)) \geq b(\mathscr{L}(u)$ **then** break ;
7    $u \leftarrow v_i$;
8    $\mathscr{L}(u) \leftarrow \mathscr{L}(v_i)$;
9 **return** u;

Algorithm 4: A variant of Kaveh's D algorithm.

5. Description of the tests

We implemented the algorithms using the C++ programming language. The g++ version 5.4.0 compiler was used, with the optimization flag -O3. Furthermore, we used a hybrid data structure based on the Compressed Row Storage and Skyline storage scheme [Gkountouvas et al., 2013].

The workstations used in the executions of the simulations with small-sized (Tables 1 and 2) and large-scale (Tables 3 and 4) matrices featured an Intel® Core™ i5-3570 (6144KB Cache, CPU @ 3.40GHz, 12GB of main memory DDR3 1333MHz, termed M1 machine) and an Intel® Core™ i7-4770 (CPU 3.40GHz, 8MB Cache, 12GB of main memory DDR3 1.6GHz, termed M2 machine) (Intel; Santa Clara, CA, United States), respectively. The Slackware 14.1.64 64-bit operating system with Linux kernel-version 3.10.17 was used on the M1 machine. The M2 machine uses an Ubuntu 16.04.4 LTS 64-bit operating system with Linux kernel-version 4.2.0-36-generic. The implementation of the incomplete Cholesky-preconditioned conjugate gradient method uses double-precision floating-point arithmetic and converges with a precision of 10^{-8} or terminates when its number of iterations is the size of the linear system.

This computational experiment uses symmetric matrices contained in the SuiteSparse matrix collection [Davis e Hu, 2011] to evaluate the results yielded by the algorithms. The bandwidth reductions depend on the choice of the initial ordering. Thus, this paper considers the original ordering given in the matrix contained in the matrix collection.

6. Results and analysis

In this section, we show the bandwidth results of the Reverse Cuthill-McKee method executed after four pseudoperipheral vertex finders. Tables in this section show the instance's name and size (n), the value of the initial bandwidth (β_0) of the matrix, and the average values of bandwidth yielded by each algorithm in 10 (3) executions carried out in each small-sized (large-scale) instance.

In Section 6.1, we report the bandwidth results yielded by the algorithms when applied to 60 matrices. In Section 6.2, we describe experiments using the zero-fill incomplete Cholesky-preconditioned conjugate gradient method.

6.1. Bandwidth results yielded by pseudoperipheral vertex finders applied to 60 matrices

Tables 1 and 2 show the bandwidth results yielded by the Reverse Cuthill-McKee method executed after three pseudoperipheral vertex finders. The tables show that the George-Liu and AVKB algorithms delivered a higher number of best results than the MKD algorithm when providing pseudoperipheral vertices for the Reverse Cuthill-McKee method. Specifically, the AVKB and George-Liu algorithms respectively led to 34 and 25 best bandwidth results, whereas the MKD algorithm conducted the Reverse Cuthill-McKee method to 20 best results. The tables show that the running times of the George-Liu and MKD algorithms are similar, wheres the AVKB algorithm takes longer execution times than the George-Liu algorithm. Since Tables 1 and 2 show that the George-Liu and AVKB algorithms dominated the MKD algorithm regarding the output quality of the bandwidth results, we did not apply this algorithm to larger matrices.

Tabela 1: Bandwidths given by the Reverse Cuthill-McKee method joined with GL, AVK and MKD algorithms (s denotes seconds$\cdot 10^{-5}$).

Matrix	n	β_0	GL	t(s)	AVKB	t(s)	MKD	t(s)
ash85	85	39	10	6	10	7	16	4
bcspwr01	39	38	7	2	7	2	8	1
bcspwr02	49	34	13	2	13	2	13	1
bcspwr03	118	115	21	6	19	7	23	4
bcsstk01	48	35	26	3	26	9	26	2
bcsstk04	132	47	54	17	53	33	54	14
bcsstk05	153	28	26	16	30	24	24	9
bcsstk22	138	111	14	8	12	7	14	5
can_144	144	142	18	9	18	12	18	6
can_161	161	79	30	9	24	13	34	5
dwt_234	234	48	19	16	19	12	25	9
lund_A	147	23	23	11	23	9	23	9
lund_B	147	23	23	10	23	9	23	9
nos1	237	4	4	10	4	7	4	6
nos4	100	13	12	5	12	4	13	3
Number of best results / average time		4	10	9	13	10	7	6

Table 3 shows the bandwidth results yielded by the Reverse Cuthill-McKee method executed after the George-Liu and AVKB algorithms when applied to large-scale matrices. Column $|E|$ in the table shows the number of non-null coefficients of the matrix. The table shows that the George-Liu algorithm delivered the same number of best results than the AVKB algorithm at shorter running times when providing pseudoperipheral vertices for the Reverse Cuthill-McKee method.

Tabela 2: Bandwidths given by the Reverse Cuthill-McKee method joined with GL, AVK and MKD algorithms (s denotes seconds·10^{-4}).

Matrix	n	β_0	GL	t(s)	AVKB	t(s)	MKD	t(s)
494_bus	494	428	62	3	62	2	82	2
662_bus	662	335	135	4	125	3	99	2
685_bus	685	550	79	4	75	3	93	3
ash292	292	24	34	1	32	2	34	1
bcspwr04	274	265	65	2	65	1	56	1
bcspwr05	443	435	59	2	59	2	74	1
bcsstk06	420	47	50	4	49	4	50	3
bcsstk19	817	567	21	7	20	4	32	4
bcsstk20	485	20	19	3	19	3	19	2
bcsstm07	420	47	523	3	28	4	506	3
can__292	292	282	65	1	71	8	65	1
can__445	445	403	92	3	88	5	119	2
can__715	715	611	157	6	120	37	202	4
can__838	838	837	155	9	153	41	175	7
dwt__209	209	184	35	1	33	1	35	1
dwt__221	221	187	16	1	16	1	15	1
dwt__245	245	115	45	1	55	1	45	1
dwt__310	310	28	13	1	14	2	13	1
dwt__361	361	50	25	1	15	1	25	1
dwt__419	419	356	34	3	34	7	73	2
dwt__503	503	452	58	5	56	4	99	4
dwt__592	592	259	42	4	42	7	42	2
dwt__878	878	519	37	4	33	14	37	4
dwt__918	918	839	72	5	50	12	72	4
dwt__992	992	513	63	8	53	53	63	8
gr_30_30	900	31	59	4	48	39	59	4
jagmesh1	936	778	27	4	27	3	27	3
nos2	957	4	4	4	4	3	4	3
nos3	960	43	79	8	65	63	79	7
nos5	468	178	88	3	96	5	88	3
nos6	675	30	16	4	30	2	30	2
nos7	729	81	65	3	65	2	65	2
plat362	362	249	56	4	39	7	41	2
plskz362	362	248	24	2	24	2	25	1
sherman1	1000	100	57	118	62	115	57	140
Number of best results / average time		5	15	7	21	13	13	7

6.2. Experiments using the zero-fill incomplete Cholesky-preconditioned conjugate gradient method

The conjugate gradient (CG) method [Hestenes e Stiefel, 1952; Lanczos, 1952] is a distinguished approach for solving large sparse linear systems when the matrix A is symmetric and positive definite. However, the number of iterations of the CG method depends on some charac-

Tabela 3: Bandwidths given by the Reverse Cuthill-McKee method joined with GL and AVKB algorithms (s denotes seconds).

Matrix	n	$\|E\|$	β_0	GL	t(s)	AVKB	t(s)
msdoor	415863	19173163	291114	6558	8	8385	33
bone010	986703	47851783	13016	13055	1942	13055	7677
ecology2	999999	4995991	1000	1000	1	1000	1
nlpkkt80	1062400	28192672	550481	38716	6	22953	9175
thermal2	1228045	8580313	1226000	925	243	925	203
Serena	1391349	64131971	81578	81125	40	84947	65
Geo_1438	1437960	60236322	26018	25994	96	21266	15
StocF-1465	1465137	21005389	36616	39638	7799	38402	9421
Hook_1498	1498023	59374451	29036	28952	35	29144	131
G3_circuit	1585478	7660826	947128	5068	4	5068	3
Number of best results / average time (in seconds)				7	1017	7	2672

teristics. Among these are the structure of the matrix A, the matrix spectrum, and the accuracy to reach. Furthermore, the number of iterations of a preconditioned CG method depends on the characteristics of the preconditioning used. This paper employs a data structure that does not rely on the bandwidth of the matrix A. Thus, the most relevant aspect of a reordering algorithm, in conjunction with the preconditioner, is to generate a matrix with these characteristics. Also, the resulting matrix should favor spatial locality. Reordering the rows and columns of sparse matrices also contributes to improving the arithmetic intensity of the sparse matrix-vector multiplication [Williams et al., 2009], which is the most critical CG kernel. Thereby, even when multiple right-hand side vectors comprise the linear system, the resulting linear system is much easier to compute than is the original linear system [Gonzaga de Oliveira et al., 2018b].

Users apply a specific preconditioner in conjunction with the conjugate gradient method depending on the problem in context. Researchers have proposed several preconditioners for the CG method. Among them, a relevant preconditioner is the incomplete Cholesky (IC) factorization [Gonzaga de Oliveira et al., 2018b]. The zero-fill incomplete Cholesky-preconditioned conjugate gradient (ICCG) method [Ajiz e Jennings, 1984] has been used successfully in the literature (e.g., see Gonzaga de Oliveira et al. [2018b] and references therein).

An efficient solution of linear systems minimizes the total computing time, including the reordering time, at least when the simulation solves only a single linear system. Therefore, reordering algorithms must be capable of obtaining high-quality solutions at a low cost.

In this section, we evaluate the George-Liu and DD algorithms when providing starting vertices for the Reverse Cuthill-McKee method. Table 4 shows the results yielded by the RCM-GL and RCM-DD algorithms for reducing the running times of the ICCG method when applied to nine linear systems. The first row in each experiment in Table 4 shows the results of the ICCG method without executing a reordering algorithm.

Table 4 shows that the RCM-GL and RCM-DD heuristics yielded similar bandwidth results (see column β). The running times of the algorithms were also similar (see column t(s):heur.).

Table 4 shows that the number of iterations does not provide the most significant impact on the computing time of the ICCG method since iterations can have different times (see column iter.). Moreover, the same table indicates that reordering the row and columns of the linear system influences the execution costs of the ICCG method. The noteworthy in Table 4 is that the ICCG

Tabela 4: Speedups given by the Reverse Cuthill-McKee method joined with GL and DD algorithms applied to reduce the execution costs of the ICCG method (s denotes seconds).

Linear system	n	heuristic	heur.:t(s)	CG:t(s)	IC:t(s)	ICCG:t(s)	β	iter.	Sp.-up/dn
shallow_water1	81920	—	—	0.01	36	36	40959	1	—
		RCM-GL	0.1	0.06	33	33	345	7	1.07
		RCM-DD	0.1	0.01	35	35	331	1	1.02
shallow_water2	81920	—	—	0.01	34	34	40959	1	—
		RCM-GL	0.1	0.09	34	34	336	12	0.99
		RCM-DD	0.1	0.01	34	34	327	1	0.99
thermo mech_dM	204316	—	—	0.70	374	374	204276	23	—
		RCM-GL	0.2	0.30	403	403	261	19	0.93
		RCM-DD	0.2	0.30	534	534	295	13	0.70
af_4_k101	503625	—	—	11054.00	1507	9147	859	10653	—
		RCM-DD	1.0	520.66	8240	8761	1279	6141	1.04
		RCM-GL	2.0	9767.00	1360	9032	1277	10393	1.01
Inline_1	503712	—	—	1688.60	11530	13219	502403	8617	—
		RCM-GL	3.0	996.70	12897	13893	5996	7479	0.95
		RCM-DD	2.6	866.93	13090	13957	9428	6406	0.95
af_shell7	504855	—	—	85.70	8309	8394	4909	993	—
		RCM-GL	1.3	74.60	8288	8363	2539	851	1.00
		RCM-DD	1.0	2.39	8352	8355	2359	27	1.00
af_shell8	504855	—	—	53.80	6123	6177	4909	840	—
		RCM-GL	1.2	48.10	6132	6180	2497	731	1.00
		RCM-DD	1.0	1.18	8323	8324	2359	13	0.74
parabolic_fem	525825	—	—	1636.20	3229	3394	525820	1619	—
		RCM-GL	0.6	29.76	3333	3363	513	688	1.01
		RCM-DD	0.6	25.89	4307	4332	1466	446	0.78
Serena	1391349	—	—	251205.10	73488	324693	81578	692102	—
		RCM-GL	6.2	0.70	74010	74011	84947	1	4.39
		RCM-DD	5.9	0.75	74821	74821	84779	1	4.34

method yielded shorter running times when associated with the RCM-GL ordering for the nine test problems used (see column Sp.-up/dn). Figure 1 also shows the speed-up/down of the ICCG method provided by the Reverse Cuthill-McKee method with starting vertices given by the George-Liu and DD algorithms.

7. Conclusions

Practitioners have used the George-Liu algorithm for providing pseudoperipheral vertices for the Reverse Cuthill-McKee method since the 1980s. On the other hand, practitioners have proposed pseudoperipheral vertex finders in the last 40 years. This paper evaluated the George-Liu algorithm against two modified algorithms when providing starting vertices for the Reverse Cuthill-McKee method. The first was a variant of Kaveh's B algorithm (AVKB), and the second was a variant of Kaveh's D algorithm (MKD) [Kaveh, 1986]. The results showed that the George-Liu algorithm dominated the MKD algorithm. The results also showed that the George-Liu algorithm yielded similar results to the AVKB algorithm at lower execution costs.

This paper also evaluated the George-Liu algorithm against a recently proposed algorithm.

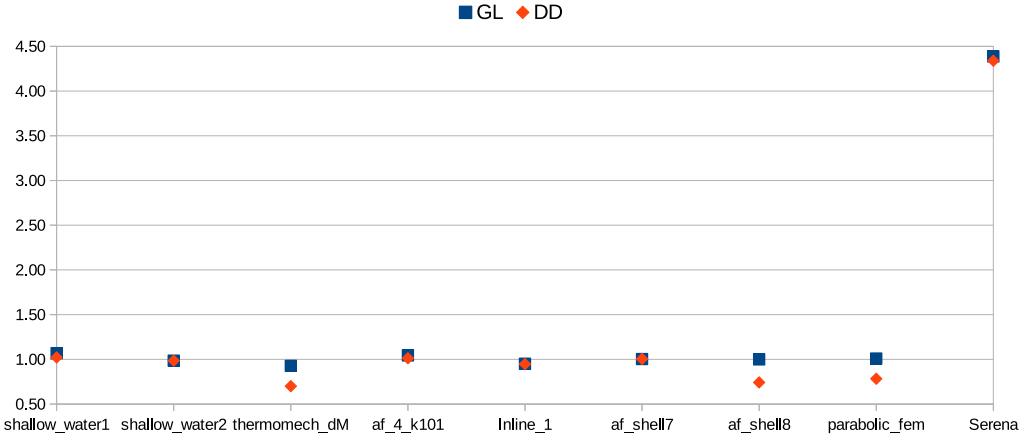


Figura 1: Speed-up/down of the ICCG method when applying the Reverse Cuthill-McKee method in conjunction with the George-Liu and DD algorithms.

The DD algorithm proved to be as fast as the George-Liu algorithm. However, the ICCG method yielded lower execution costs when associated with the RCM-GL method for the nine linear systems used in the experiments. Thus, the results showed that the George-Liu algorithm remains in the state of the practice when providing starting vertices for the Reverse Cuthill-McKee method. The parallel approaches of these algorithms are the next step in this investigation.

Referências

Ajiz, M. A. e Jennings, A. (1984). A robust incomplete Choleski-conjugate gradient algorithm. *International Journal for Numerical Methods in Engineering*, 20:949–966.

Boost (2017). Boost C++ libraries. http://www.boost.org. Accessed: 2017-06-28.

Davis, T. A. e Hu, Y. (2011). The University of Florida sparse matrix collection. *ACM Trans. on Mathematical Software*, 38(1):1–25.

Eaton, J. W., Bateman, D., Hauberg, S., e Wehbring, R. (2015). GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations. URL `http://www.gnu.org/software/octave/doc/interpreter`.

George, A. e Liu, J. W. (1981). *Computer solution of large sparse positive definite systems*. Prentice-Hall, Englewood Cliffs.

Gkountouvas, T., Karakasis, V., Kourtis, K., Goumas, G., , e Koziris, N. (2013). Improving the performance of the symmetric sparse matrix-vector multiplication in multicore. In *27th International Symposium on Parallel & Distributed Processing (IPDPS)*, p. 273–283, Boston, MA. IEEE Computer Society.

Gonzaga de Oliveira, S. L. e Abreu, A. A. A. M. (2018a). An evaluation of pseudoperipheral vertex finders for the reverse cuthill-mckee method for bandwidth and profile reductions of symmetric matrices. In *The 37th International Conference of the Chilean Computer Science Society (SCCC)*, Santiago, Chile. IEEE.

Gonzaga de Oliveira, S. L. e Abreu, A. A. A. M. (2018b). A pseudoperipheral vertex finder based on the George-Liu and Kaveh's B algorithms for small symmetric matrices. In *The 50th Brazilian Symposium of Operational Research (SBPO)*, Rio de Janeiro–RJ, Brazil. Sobrapo.

Gonzaga de Oliveira, S. L., Bernardes, J. A. B., e Chagas, G. O. (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., Bernardes, J. A. B., e Chagas, G. O. (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.

Gonzaga de Oliveira, S. L. e Chagas, G. O. (2015). A systematic review of heuristics for symmetric–matrix bandwidth reduction: methods not based on metaheuristics. In *The XLVII Brazilian Symposium of Operational Research (SBPO)*, Ipojuca–PE, Brazil. Sobrapo.

Hestenes, M. R. e Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(36):409–436.

Kaveh, A. (1986). Ordering for bandwidth reduction. *Computer & Structure*, 24:413–420.

Lanczos, C. (1952). Solutions of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards*, 49(1):33–53.

Papadimitriou, C. H. (1976). The NP-completeness of bandwidth minimization problem. *Computing Journal*, 16:177–192.

The MathWorks, Inc. (1994–2020). MATLAB. http://www.mathworks.com/products/ matlab.

Williams, S., Waterman, A., e Patterson, D. (2009). Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76.

Zhang, Y., Huang, T.-Z., Shao, W., e Lai, S.-J. (2013). Some new strategies for rcm ordering in solving electromagnetic scattering problems. *Computer Physics Communications*, 184(4):1161–1164.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings