Tema
Tendências em Matemática Aplicada e Computacional, **20**, N. 3 (2019), 497-507

www.scielo.br/tema
doi: 10.5540/tema.2019.020.03.0497

An Experimental Analysis of Three Pseudo-peripheral Vertex Finders in conjunction with the Reverse Cuthill-McKee Method for Bandwidth Reduction

S. L. GONZAGA DE OLIVEIRA[1*] and A. A. A. M. ABREU[2]

Received on December 13, 2017 / Accepted on May 16, 2019

ABSTRACT. The need to determine pseudoperipheral vertices arises from several graph-theoretical approaches for ordering sparse matrix equations. The results of two algorithms for finding such vertices, namely, the George-Liu and Kaveh-Bondarabady algorithms, are evaluated in this work along with a variant of the Kaveh-Bondarabady algorithm. The results suggest that the well-know George-Liu algorithm dominates the other two pseudoperipheral vertex finders mainly when considering the computational times of the algorithms.

Keywords: sparse matrices, graph labeling, graph algorithm, Reverse Cuthill-McKee method, bandwidth reduction, graph theory.

1 INTRODUCTION

Reordering a sparse matrix to reduce its bandwidth can accelerate many sparse matrix computations [7]. For example, several real-world scientific and engineering applications demand 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, x is the unknown n-vector solution, and b is a known n-vector. In particular, an efficient solution using a direct method requires to order the variables of the problem. Moreover, two other methods for solving these types of linear equations, which have found wide use in finite element analysis, are the profile and frontal solution schemes. These methods demand to process the equations in a proper order to compute the solution efficiently. In finite element analysis, in the case of one degree-of-freedom per node, performing a vertex reordering is equivalent to reorder the equations. Conducting a vertex reordering is also equivalent to rearrange the equations in partial differential equations finite-volume discretizations.

*Corresponding author: Sanderson L. Gonzaga de Oliveira – E-mail: sanderson@dcc.ufla.br – https://orcid.org/0000-0003-4863-542X
[1]Universidade Federal de Lavras, Lavras-MG, Brazil – E-mail: sanderson@dcc.ufla.br
[2]Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina - IFSC, Canoinhas-SC, Brazil – E-mail: alexandre.abreu@ifsc.edu.br

In general, the use of a heuristic for matrix bandwidth reductions reduces the computational cost of iterative solvers for the numerical solution of a sparse linear system of equations [5, 16, 17]. Specifically, the transfer of information to and from memory is related to the number of nonzero coefficients in the lines of the matrix system A. If the nonzero coefficients in each line lie at the same level of the memory hierarchy, then cache misses are reduced. These characteristics link cache misses with the bandwidth of A.

This paper concentrates on the use of the Reverse Cuthill-McKee method (RCM) [11] for reducing the bandwidth of matrices. The George-Liu algorithm (GL) [10] is the state-of-the-practice method for providing pseudoperipheral (see definitions below) vertices to the Reverse Cuthill-McKee method. Thus, this matrix bandwidth method starting with a pseudoperipheral vertex given by the George-Liu algorithm [10] is one of the best-known and widely used heuristics for bandwidth reductions of matrices (see [3, 4, 9, 12, 14, 16, 17] and references therein). For instance, the method is available on MATLAB [9, 12, 22] and GNU Octave [8, 9] as the function *symrcm*[1], and on Boost C++ Library [20][2]. The method is identified as one of the most important and amply used bandwidth reduction method because it reaches accurate approximations to the solution at low execution times. However, it is well-known that the Reverse Cuthill-McKee method is not trouble-free [3]. The choice of an initial pseudoperipheral vertex v profoundly influences the quality of the method. Thus, efforts should be continued to develop algorithms for the identification of pseudoperipheral vertices so that the Reverse Cuthill-McKee method [11] (and other graph-theoretical reordering heuristics) would be capable of producing solutions of excellent quality.

This paper analizes experimentally the George-Liu [10] and Kaveh-Bondarabady [19] algorithms for finding pseudoperipheral vertices along with a modified Kaveh-Bondarabady algorithm when applied in conjunction with the Reverse Cuthill-McKee method [11] for bandwidth reductions of matrices. This work is a revised and expanded version of a paper presented at the XXXVII Brazilian National Congress in Applied and Computational Mathematics (CNMAC 2017) [13].

In Section 2, background information with respect to the bandwidth reduction problem is provided. In Section 3, we describe the algorithms for determining the starting vertex to the Reverse Cuthill-McKee method evaluated in this computational experiment. In Section 4, we describe how the simulations were conducted. In Section 5, the results are discussed. Finally, In Section 6, the conclusions are addressed.

2 THEORETICAL BACKGROUND, DEFINITIONS, AND GRAPH NOTATION

In this section, we provide definitions necessary to support the discussion in the following sections. In this section, we give some basic background concepts concerning the characteristics of the graph-theoretical description of ordering.

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

[2] `http://www.boost.org/doc/libs/1\_38\_0/libs/graph/doc/cuthill\_mckee\_ordering.html`.

Let $A = [a_{ij}]$ be an $n \times n$ symmetric matrix associated with a connected undirected graph $G = (V,E)$, where V and E are sets of vertices and edges, respectively. The overall bandwidth of a matrix A is defined as $\beta(A) = \max_{1\leq i\leq n}\left[i - \min_{1\leq j<i}(j \mid a_{ij} \neq 0)\right]$ [18]. The vertices $v,u \in V$ with labels $s(v) = i$ and $s(u) = j$ are associated with lines i and j of A, respectively. Therefore, $a_{ii} \neq 0$, $a_{ij} \neq 0$ if $\{v,u\} \in E$, and $a_{ij} = 0$ if $\{v,u\} \notin E$. Thus, 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 $\beta(G) = \max_{\{v,u\}\in E}[|s(v) - s(u)|]$.

The Reverse Cuthill-McKee method [11] is based on graph-theoretical concepts. Essentially, the method labels the vertices of a graph $G(V,E)$ in order of increasing distance from a given initial vertex v. Ordering the vertices in such a manner partitions them into *level sets* according to the distance from the vertex v. Given a vertex $v \in V$, the level structure $\mathscr{L}(v)$ rooted at vertex v, with depth $\ell(v)$, is the partitioning $\mathscr{L}(v)$ of V satisfying $\mathscr{L}(v) = \{L_0(v), L_1(v), \ \ldots, L_{\ell(v)}(v)\}$, where $L_0(v) = \{v\}$ and $L_i(v) = Adj(L_{i-1}(v)) - \bigcup_{j=0}^{i-1} L_j(v)$, for $i = 1,2,3, \ldots, \ell(v)$ [2], $\ell(v) = \max_{u\in V}[d(v,u)]$ denotes the *eccentricity* of the vertex v, $Adj(U) = \{w \in V : (u \in U \subseteq V)\ \{u,w\} \in E\}$, and the *distance* $d(v,u)$ is the length of a shortest path connecting vertices v and u [18]. In particular, the *width* of a rooted level set is defined as $b(\mathscr{L}(v)) = \max_{0\leq i\leq \ell(v)} |L_i(v)|$.

The diameter of a graph is defined as $\Phi(G) = \max_{v\in V}[\ell(v)]$. A vertex $v \in V$ with $\ell(v) = \Phi(G)$ is a peripheral vertex. On first consideration, starting the Reverse Cuthill-McKee procedure with a peripheral vertex seems to be a compelling idea. The reason is that the heuristic approach would use a vertex with maximum eccentricity, in the hope that on average the rooted level set $\mathscr{L}(v)$ would also have small width. Finding peripheral vertices in graphs, however, is computationally expensive [21], that is, for a graph $G = (V,E)$, one can find a peripheral vertex by executing $|V|$ breadth-first search procedures. However, it takes $O(|V|(|V| + |E|))$ time in these operations. There exist alternative algorithms for finding a peripheral vertex, including Arany's algorithm [1], but these algorithms are still computationally expensive when compared with a pseudoperipheral vertex finder. Consequently, many heuristics for bandwidth reductions require as a first step the determination of a pseudoperipheral vertex instead of determining a peripheral vertex as the initial vertex of the approach [15]. If, for a given $u \in V$, a vertex $v \in L_{\ell(u)}(u)$ has $\ell(v) = \max_{x\in L_{\ell(u)}(u)}[\ell(x)]$, and $\ell(v) = \ell(u)$, then Smyth [21] defined u as a pseudoperipheral vertex. On the other hand, Kaveh and Bondarabady [19] defined a pseudoperipheral vertex regarding the width $b(\mathscr{L}(v))$ of a vertex $v \in V$. It is, therefore, desirable to find a pseudoperipheral vertex v with large $\ell(v)$ at a low cost. Additionally, the rooted level set $\mathscr{L}(v)$ must have small width.

Algorithm 1 shows the Reverse Cuthill-McKee method [11]. As previously mentioned, the method labels the vertices of a graph (see line 8) with the same distance from the pseudoperipheral vertex v in order of increasing degree (see line 6). Finally, the ordering is reversed. Therefore, the final label of vertex v is $|V|$. Thus, Algorithm 1 begins the numbering with the label $s(|V|)$ (see lines 2–4 in Algorithm 1). Algorithm 1 returns the new labeling at line 12.

Algorithm 1 Reverse Cuthill-McKee method [11].

Input: a connected graph $G = (V,E)$; a vertex $v \in V$;

Output: a labeling $S = \{s(1), s(2), ..., s(|V|)\}$.

```
1  begin
2      s(|V|) ← v;
3      i ← |V|;
4      j ← |V|;
5      while (i > 0) do
6          foreach (vertex w ∈ Adj(s(j)) − {s(|V|), ..., s(i)}, in order of ascending degree) do
7              i ← i − 1;
8              s(i) ← w;
9          end
10         j ← j − 1;
11     end
12     return S;
13 end
```

3 ALGORITHMS FOR FINDING PSEUDO-PERIPHERAL VERTICES

The George-Liu algorithm [9] returns a pseudoperipheral vertex v with eccentricity equal or greater than the eccentricity of the vertex with minimum degree in $L_{\ell(v)}(v)$. Algorithm 2 shows this pseudoperipheral vertex finder. The George-Liu algorithm [9] begins with an arbitrary vertex v at line 2 in Algorithm 2. Afterward, the algorithm builds the rooted level structure $\mathscr{L}(v)$ at line 3. Then, the George-Liu algorithm [9] builds the rooted level structure $\mathscr{L}(u)$ of a vertex u with minimum degree belonging to $L_{\ell(v)}(v)$ in lines 5 and 6. If $\ell(u) > \ell(v)$ at line 7, then u is attributed to v (i.e., $v \leftarrow u$; see line 8), $\mathscr{L}(v) \leftarrow \mathscr{L}(u)$ is computed at line 9, and the process is repeated; otherwise, the process stops and v is the pseudoperipheral vertex found at line 12 in Algorithm 2.

The Kaveh-Bondarabady algorithm [19] returns a pseudoperipheral vertex v in which $\mathscr{L}(v)$ has a smaller width than the widths of the level structures rooted at vertices with minimum degree in each level of $\mathscr{L}(v)$. Algorithm 3 shows the Kaveh-Bondarabady algorithm [19]. We refer to this algorithm as the KB2 method. The algorithm selects a vertex v with minimum degree of the graph at line 2. Then, the method generates a rooted level structure $\mathscr{L}(v)$ at line 3, computes its width $b(\mathscr{L}(v))$ at line 4, and selects a vertex u with minimum degree (at line 8) from each level $L_i(v)$ of $\mathscr{L}(v)$ (see the loop in lines 7–15 in Algorithm 3). The KB2 algorithm generates a rooted level structure $\mathscr{L}(u)$ from each of such vertices at line 9, computes its width $b(\mathscr{L}(u))$ at line 10, and chooses the one corresponding to the smallest width (in lines 10–14). The algorithm repeats the process as far as reduction in width of the current rooted level structure can be observed (see the loop in lines 5–16 in Algorithm 3). Finally, the Kaveh-Bondarabady algorithm returns a pseudoperipheral vertex s with a small $b(\mathscr{L}(s))$ at line 17.

Algorithm 2 The George-Liu algorithm [9].

Input: graph $G = (V,E)$;

Output: pseudoperipheral vertex $v \in V$;

```
1  begin
2      v ← ArbitraryVertex(V);
       // build a rooted level structure
3      L(v) ← Breadth-First-Search-variant(v);
4      repeat
5          u ← MinimumDegreeVertex(L_ℓ(v)(v));
           // build a rooted level structure
6          L(u) ← Breadth-First-Search-variant(u);
7          if (ℓ(u) > ℓ(v)) then
8              v ← u;
9              L(v) ← L(u);
10         end
11     until (u ≠ v);
12     return v;
13 end
```

In this computational experiment, we implemented and evaluated a slight modification in the Kaveh-Bondarabady algorithm [19] by starting with an arbitrary vertex instead of starting with a vertex with minimum degree at line 2 in Algorithm 3. We refer to this algorithm as the MKB2 method. We studied experimentally these algorithms in conjunction with the Reverse Cuthill-McKee method [11].

We also evaluated the use of an arbitrary vertex instead of selecting a vertex with minimum degree at line 8 in Algorithm 3. Exploratory investigations showed that this modification did not improve the general performance of the RCM ordering. Thus, we discarded this modification.

4 DESCRIPTION OF THE TESTS

Three algorithms for finding pseudoperipheral vertices were implemented and evaluated in this computational experiment (George-Liu [10], KB2 [19], and an alternative algorithm for finding pseudoperipheral vertices described in Section 3) in conjunction with the Reverse Cuthill-McKee method [11]. Thus, these three algorithms (together with the Reverse Cuthill-McKee method) are named RCM-GL, RCM-KB2, and RCM-MKB2, respectively.

We implemented the algorithms in the C++ programming language. We used a GNU (i.e., the g++ version 4.8.4) compiler. To evaluate the bandwidth reductions provided by these algorithms, we used 26 symmetric matrices (with sizes ranging from 10,605 to 607,232) contained in the SuiteSparse matrix collection [6].

Algorithm 3 The Kaveh-Bondarabady algorithm (KB2) [19].

Input: graph $G = (V,E)$;
Output: pseudoperipheral vertex $s \in V$;

```
1  begin
2      v ← VertexMinDegree(V);
           // build the rooted level structure ℒ(v)
3      ℒ(v) ← Breadth-First-Search-variant(v);
4      width ← b(ℒ(v));
5      repeat
6          s ← v;
               // observe each level in the rooted level structure ℒ(v)
7          for (i ← 1 to ℓ(v)) do
8              u ← VertexMinDegree(L_i(v));
                   // build the rooted level structure ℒ(u)
9              ℒ(u) ← Breadth-First-Search-variant(u);
10             if (b(ℒ(u)) < width) then
11                 width ← b(ℒ(u));
12                 v ← u;
13                 ℒ(v) ← ℒ(u);
14             end
15         end
16     until (v = s);
17     return s
18 end
```

The workstation used in the executions of the simulations featured an Intel® Core™ i7-4770 (CPU 3.40 GHz, 8 MB Cache, 8 GB of main memory DDR3 1.333 GHz) (Intel; Santa Clara, CA, United States). This machine used the Ubuntu 14.04.5 64-bit operating system with Linux kernel-version 4.2.0-36-generic.

5 RESULTS AND ANALYSIS

Table 1 shows the characteristics of the matrices [name, size (n, number ($|E|$) of nonzero coefficients), original bandwidth (β_0)] and the average values of bandwidth and time, in seconds, yielded by three pseudoperipheral vertex finders with the Reverse Cuthill-McKee method. Numbers in boldface are the best results.

Table 1 shows that the RCM-MKB2 method performed better on average than the RCM-KB2 method did. The same table and Figure 1 show that the variant of the KB2 algorithm [19] evaluated in this study obtained in general the highest number of best bandwidth results (in 12 matrices) in the dataset composed of 26 symmetric matrices when applied in tandem with the Reverse

Table 1: Results of three pseudoperipheral vertex finders in conjunction with the Reverse Cuthill-McKee method [11] applied to reduce the bandwidth of 26 symmetric matrices.

Matrix	n	$\|E\|$	β_0	GL		MKB2		KB2	
				β	t(s)	β	t(s)	β	t(s)
ted_B	10,605	144,579	48	**42**	3.61	72	6.19	63	3.02
ted_B _unscaled	10,605	144,579	48	**42**	3.59	72	6.17	63	3.05
vibrobox	12,328	301,700	12162	4841	0.13	4604	0.09	**4489**	0.10
minsurfo	40,806	203,622	**202**	**202**	0.05	**202**	356.94	265	0.02
gridgena	48,962	512,084	405	404	0.07	**394**	0.20	403	0.15
qa8fk	66,127	1,660,579	**1048**	1968	0.27	1966	0.19	1966	0.19
qa8fm	66,127	1,660,579	**1048**	1968	0.26	1966	0.19	1966	0.19
thermal1	82,654	574,458	80916	**232**	1.82	80916	3.07	410	3.41
boyd1	93,279	1,211,231	93269	**89508**	0.52	**89508**	0.40	90678	0.43
2cubes _sphere	101,492	1,647,264	100407	4800	0.65	5538	0.48	**4473**	0.31
thermo mech_TC	102,158	711,558	102138	276	0.26	**253**	3565.17	**253**	3390.19
thermo mech_TK	102,158	711,558	102138	276	0.31	**253**	6278.30	**253**	5642.73
G2_circuit	150,102	726,674	93719	**1945**	0.37	1961	0.15	1960	0.43
c-73	169,422	1,279,274	169413	80177	0.49	**61694**	9.94	**81036**	10328.08
c-73b	169,422	1,279,274	169413	80177	0.48	**61694**	9.77	**81036**	10263.89
cont-300	180,895	988,195	157496	604	0.22	**603**	0.26	**603**	0.26
d_pretok	182,730	1,641,672	129917	2699	0.55	2578	0.51	**2577**	0.25
turon_m	189,924	1,690,876	185352	3023	0.80	**3021**	0.33	4338	0.73
thermo mech_dM	204,276	1,423,116	185352	**276**	0.62	603	4085.86	515	7716.47
HTC_336 _4438	226,337	783,496	94690	38689	675.13	**30196**	688.61	30198	982.07
HTC_336 _9129	226,337	762,969	94690	42761	676.43	30199	8973.65	**30198**	996.92
offshore	259,789	4,242,673	237738	**21617**	1.80	25873	1.55	21793	1.54
dielFilterV3clx	420,408	25,309,272	420391	13196	11.24	**12428**	304.49	14705	15.18
boyd2	466,316	1,500,397	373055	**246807**	0.48	262683	37516.37	**246807**	3332.29
gsm_106857	589,446	21,758,924	588744	**17742**	7.91	20991	7.68	21959	4.92
dielFilterV2clx	607,232	25,309,272	498160	**14745**	4.34	**14745**	616365.30	22821	12.58
Number of best results			3	11	13	12	6	8	9

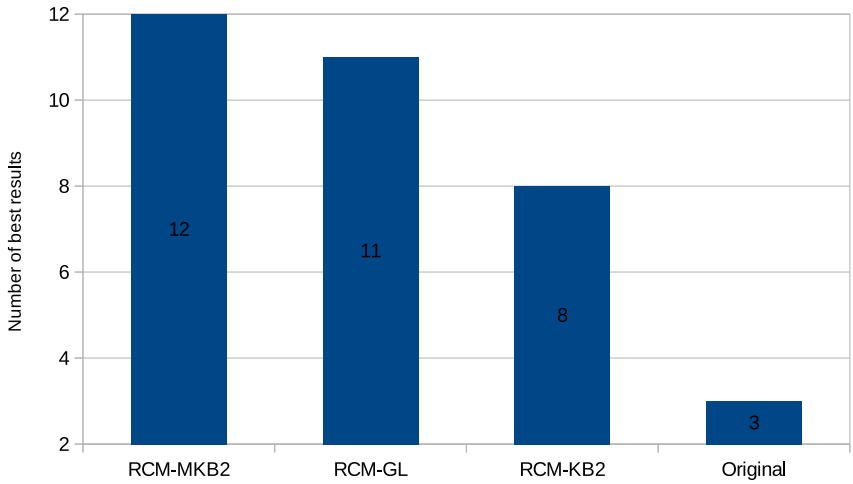

Figure 1: The number of best results yielded by three methods when applied to reduce the bandwidth of 26 symmetric matrices. Original means that the three methods evaluated did not reduce the bandwidth of the original matrix.

Cuthill-McKee method [11]. Original in this figure means that the three methods evaluated did not reduce the bandwidth of the original matrix. Nevertheless, Table 1 also shows that the RCM-GL method [9] yielded almost the same number of best bandwidth results than the RCM-MKB2 method at shorter execution times than did the two other methods evaluated.

Table 1 and Figure 2 show that the RCM-GL and RCM-KB2 methods delivered overall similar running times when applied to 18 matrices. The RCM-KB2 method, however, delivered much longer execution times than the RCM-GL method did when applied to the eight other matrices used in this computational experiment. Furthermore, Table 1 shows that the RCM-KB2 and RCM-MKB2 methods performed erratically in the sense that these methods may deliver poor performance for some instances (e.g., *boyd2*, *thermomech_TK*, *thermomech_dM*). In particular, the execution times of the RCM-MKB2 algorithm were much longer than the execution times of the two other algorithms evaluated when applied to several matrices (e.g., *dielFilterV2clx*, *boyd2*, *HTC_336_9129*). Probably, a bad choice in the initial vertex incurs in many iterations in the repeat-until loop in the MKB2 algorithm.

6 CONCLUSIONS

In this work, we compared the results yielded by the George-Liu [10] and Kaveh-Bondarabady [19] algorithms with a modified Kaveh-Bondarabady algorithm to provide initial vertices to the Reverse Cuthill-McKee method [11] for bandwidth reductions of matrices. The results of the algorithms implemented in this paper achieved the expected bandwidth quality based on the existing literature [9, 16, 19].

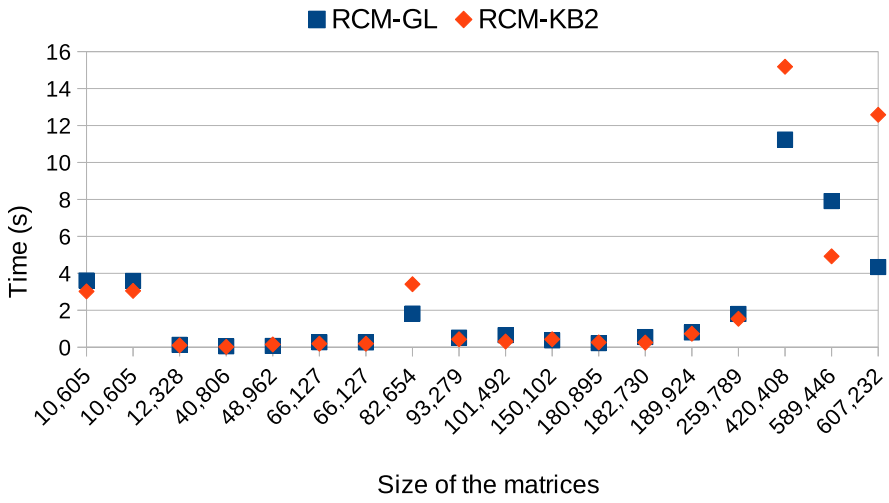

Figure 2: Execution times of the RCM-GL and RCM-KB2 methods when applied to 18 symmetric matrices.

The RCM-GL algorithm [9] yielded, in general, better bandwidth results at shorter execution times than did the two other algorithms evaluated when applied to 26 symmetric matrices (with sizes ranging from 10,605 to 607,232). Thus, based on the results of our experiments, we conclude that although the RCM-MKB2 algorithm is a suitable alternative for reducing the bandwidth of matrices, the RCM-GL method [9] dominated the RCM-MKB2 and RCM-KB2 [19] methods in reducing the bandwidth of sparse matrices. Therefore, our experiments led us to conclude that, among the algorithms evaluated here, the George-Liu algorithm [10] performed best on average. Additionally, the method is the recommended algorithm for finding pseudoperipheral vertices to the Reverse Cuthill-McKee ordering [11].

Heuristics for bandwidth reductions contribute to providing adequate memory location, and hence, improving cache hit rates [5, 16]. We plan to apply the algorithms evaluated in this work to perform parallel implementations of vertex reordering algorithms and, therefore, reduce the computational times of parallel iterative methods for solving linear systems to verify the best algorithm(s) in specific application fields.

RESUMO. A necessidade de se determinar vértices pseudoperiféricos surge de diversas abordagens por teoria dos grafos de ordenação de linhas e colunas de sistemas de equações lineares compostos de matrizes esparsas. Neste trabalho, são mostrados resultados de dois algoritmos para encontrar vértices pseudoperiféricos: George-Liu e Kaveh-Bondarabady. Os resultados desses algoritmos são comparados com os resultados de uma variação do algoritmo de Kaveh-Bondarabady. Por meio de análise experimental, concluiu-se que o

algoritmo de George-Liu retornou melhores resultados que os outros dois algoritmos, principalmente ao se considerar os tempos de execução dos algoritmos.

Palavras-chave: matrizes esparsas, numeração de vértices de grafos, algoritmos em grafos, método Reverse Cuthill-McKee, redução de largura de banda, teoria dos grafos.

REFERENCES

[1] I. Arany. An efficient algorithm for finding peripheral nodes. In "Colloquia Mathematica Societatis János Bolyai (Hungarian Edition), Theory of Algorithms Pécs", volume 44 (1984), pp. 27–35.

[2] I. Arany, W. Smyth & L. Szóda. An improved algorithm for reducing the bandwidth of sparse symmetric matrices. *International Federation for Information Processing (IFIP) Congress*, **44** (1971), 27–35.

[3] M. Benzi, D.B. Szyld & A. Van Duin. Orderings for incomplete factorization preconditioning of nonsymmetric problems. *SIAM Journal on Scientific Computing*, **20**(5) (1999), 1652–1670.

[4] J. Camata, A. Rossa, A. Valli, L. Catabriga, G. Carey & A. Coutinho. Reordering and incomplete preconditioning in serial and parallel adaptive mesh refinement and coarsening flow solutions. *International Journal for Numerical Methods in Fluids*, **69**(4) (2012), 802–823.

[5] G.O. Chagas & S.L. Gonzaga de Oliveira. Metaheuristic-based heuristics for symmetric-matrix bandwidth reduction: a systematic review. *Procedia Computer Science*, **51** (2015), 211–220.

[6] T.A. Davis & Y. Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, **38**(1) (2011), 1.

[7] I.S. Duff & G.A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT Numerical Mathematics*, **16**(3) (1989), 263–270.

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

[9] A. George & L. Joseph. "Computer solution of large sparse positive definite systems". Prentice-Hall, Englewood Cliffs (1981).

[10] A. George & J.W. Liu. An implementation of a pseudoperipheral node finder. *ACM Transactions on Mathematical Software (TOMS)*, **5**(3) (1979), 284–295.

[11] J.A. George. Computer implementation of the finite element method. Technical report, Stanford University CA Dept. of Computer Science (1971).

[12] J.R. Gilbert, C. Moler & R. Schreiber. Sparse matrices in MATLAB: Design and implementation. *SIAM Journal on Matrix Analysis and Applications*, **13**(1) (1992), 333–356.

[13] S.L. Gonzaga de Oliveira & A.A.A.M. Abreu. The use of the reverse Cuthill-McKee method with an alternative pseudo-peripheral vertice finder for profile optimization. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, **6**(1) (2018).

[14] S.L. Gonzaga de Oliveira, A.A.A.M. Abreu, D.T. Robaina & M. Kischinhevsky. An evaluation of four reordering algorithms to reduce the computational cost of the Jacobi-preconditioned conjugate gradient method using high-precision arithmetic. *International Journal of Business Intelligence and Data Mining*, **12**(2) (2017), 190–209.

[15] S.L. Gonzaga de Oliveira, A.A.A.M. Abreu, D.T. Robaina & M. Kischinhevsky. Finding a starting vertex for the reverse Cuthill-McKee method for bandwidth reduction: a comparative analysis using asymmetric matrices. In "International Conference on Computational Science and Its Applications". Springer (2018), pp. 123–137.

[16] S.L. Gonzaga de Oliveira, J.A.B. Bernardes & G.O. Chagas. An evaluation of low-cost heuristics for matrix bandwidth and profile reductions. *Computational and Applied Mathematics*, **37**(2) (2018), 1412–1471.

[17] S.L. Gonzaga de Oliveira, J.A.B. Bernardes & G.O. Chagas. An evaluation of reordering algorithms to reduce the computational cost of the incomplete Cholesky-conjugate gradient method. *Computational & Applied Mathematics*, **37**(3) (2018).

[18] S.L. Gonzaga de Oliveira & G.O. Chagas. "Introdução a Heurísticas para Redução de Largura de Banda de Matrizes". SBMAC (2014).

[19] A. Kaveh & H.R. Bondarabady. Ordering for wavefront optimization. *Computers & Structures*, **78**(1-3) (2000), 227–235.

[20] R. Rivera. Boost C++ Libraries (2004). URL `http://www.boost.org/`. Accessed: 2019-4-2.

[21] W. Smyth. Algorithms for the reduction of matrix bandwidth and profile. *Journal of Computational and Applied Mathematics*, **12** (1985), 551–561.

[22] The MathWorks. Inc. MATLAB (2019). URL `http://www.mathworks.com/products/matlab`.