

A Variant of the George-Liu Algorithm

S. L. Gonzaga de Oliveira[1(✉)], A. A. A. M. Abreu[2], C. Osthoff[3], and L. N. Henderson Guedes de Oliveira[4]

[1] Universidade Federal de Lavras, Lavras, MG, Brazil
sanderson@ufla.br
[2] Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, Canoinhas, SC, Brazil
alexandre.abreu@ifsc.edu.br
[3] Laboratório Nacional de Computação Científica (LNCC), Petropólis, RJ, Brazil
osthoff@lncc.br
[4] Universidade do Estado do Rio de Janeiro, Nova Friburgo, RJ, Brazil
nelio@iprj.uerj.br

Abstract. This paper evaluates a variant of the George-Liu algorithm for finding a pseudoperipheral vertex in a graph aiming at returning a vertex having a larger eccentricity than the original algorithm. The experiments show that the Reverse Cuthill-McKee method with ordering started with vertices given by the new variant of the George-Liu algorithm yields promising results when applied to small symmetric matrices.

Keywords: Pseudoperipheral vertices · Reverse Cuthill-McKee method · Bandwidth reduction · Profile reduction · Reordering algorithms · Graph labeling · Sparse matrices · Orderings

1 Introduction

Heuristics for bandwidth and profile reductions are used to achieve low computational and storage costs for solving sparse linear systems. Thus, there has been given much attention to heuristics for bandwidth and profile reductions (see [1–8] and references therein). The bandwidth and minimization problems are of great interest because they are related to a large number of real-world problems in science and engineering [4,5]. As a consequence, the importance of matrix bandwidth and profile reductions has made these challenging problems subject to numerous research efforts (see [3–6] and references therein).

Let $G = (V, E)$ be a 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 $\beta(G) = \max_{\{v,u\} \in E}[|s(v) - s(u)|]$ where $s(v)$ and $s(u)$ are labels of vertices v and u, respectively. The profile is defined as $profile(G) = \sum_{v \in V} \max_{\{v,u\} \in E}[|s(v) - s(u)|]$. Let A be an $n \times n$ symmetric matrix

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11619, pp. 3–12, 2019.
https://doi.org/10.1007/978-3-030-24289-3_1

associated with a connected undirected graph $G = (V, E)$. Equivalently, the bandwidth of row i of matrix A is given by $\beta_i(A) = i - \min_{1 \leq j < i} [j : a_{ij} \neq 0]$ and the bandwidth of a matrix A is defined as $\beta(A) = \max_{1 \leq i \leq n} [\beta_i(A)]$. The profile of a matrix A is defined as $profile(A) = \sum_{i=1}^{n} \beta_i(A)$.

The Reverse Cuthill-McKee (RCM) method [9] starting with a pseudoperipheral vertex given by the George-Liu (GL) algorithm [10], termed RCM-GL [11], is one of the best-known and widely used heuristics for bandwidth and profile reductions of matrices [1,4–6,12]. Previous publications [4,5] report the RCM-GL method [11] as potentially being one of the most promising low-cost heuristics for bandwidth reductions.

Few algorithms can compete with the RCM-GL method [11] in bandwidth results when considering execution times [4,5]. The reason is that the RCM-GL method [11] yields reasonable bandwidth results at low cost. This may be the reason why this method is available on the MATLAB [11,13,14] and GNU Octave [11,15] mathematical softwares as the function *symrcm*[1], and on Boost C++ Library [16][2]. Thus, the Reverse Cuthill-McKee method [9] is a well-known and successful heuristic for reducing bandwidth and profile of matrices. The method is identified as one of the most important and amply used bandwidth reduction methods [1,4–6,12,13] because the method yields accurate approximations to the solution at low execution times. Although the Reverse Cuthill-McKee method [9] remains in common use, it is well-known that the method is not trouble-free [1]. The choice of the initial pseudoperipheral vertex influences the quality of the results given by this method.

The Reverse Cuthill-McKee method [9] is a kind of heuristic known as level set reorderings, where the vertices in a graph are labeled considering that level sets subdivide the vertices, i.e., each level set contains a group of vertices considering the distance from a given starting vertex. Thus, the quality of the bandwidth results will be strongly influenced by the selection of the starting vertex and by the ordering of the vertices within level sets. In general, the Reverse Cuthill-McKee method yields better results when the width of a level structure rooted at the starting vertex is small, and its eccentricity is very close to the diameter of the graph. Additionally, the computing times of an algorithm for the identification of an appropriate starting vertex for the Reverse Cuthill-McKee method can be even higher than the running times of the Reverse Cuthill-McKee method itself. Thus, the bandwidth results of the Reverse Cuthill-McKee method executed after an algorithm for the selection of a starting vertex must be substantially better than the bandwidth results reached by the Reverse Cuthill-McKee method without the use of such a kind of algorithm.

The purpose of this paper is to evaluate a variant of the George-Liu algorithm. Specifically, this paper applies an alternative algorithm for finding

[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.

pseudoperipheral vertices along with the Reverse Cuthill-McKee method [9] for bandwidth and profile reductions of matrices. We compare its results with the results of the original George-Liu [10] and Reid-Scott [17] algorithms for determining the starting vertex for the Reverse Cuthill-McKee method. We refer to these three algorithms together with the Reverse Cuthill-McKee method as RCM-GL(k), RCM-GL [11], and RCM-RS, respectively.

Section 2 briefly describes the Reverse Cuthill-McKee [9] and George-Liu [10] algorithms. This section also introduces a variant of the George-Liu algorithm. Section 3 describes how we conducted the simulations in this study. Section 4 shows the results. Specifically, we experimentally analyze the effectiveness of the proposed algorithm for finding pseudoperipheral vertices for the Reverse Cuthill-McKee method [9]. Finally, Sect. 5 addresses the conclusions.

2 A Variant of the George-Liu Algorithm for Finding Pseudoperipheral Vertices

The Reverse Cuthill-McKee method [9] is based on graph-theoretical concepts. It labels the vertices of a graph $G(V, E)$ in order of increasing distance from a given pseudoperipheral vertex v. Specifically, the method labels vertices with the same distance from the vertex v in order of increasing degree. Finally, the ordering is reversed. Therefore, the final label of vertex v is $|V|$. Ordering the vertices in such manner partitions them into *level sets* according to the distance from the pseudoperipheral 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 $\mathscr{L}(v) = \{L_0(v), L_1(v), \ldots, L_{\ell(v)}(v)\}$ where $L_0(v) = \{v\}$ and $L_i(v) = Adj(L_{i-1}(v)) - \bigcup_{j=0}^{i-1} L_j(v)$, for $i = 1, 2, 3, \ldots, \ell(v)$, $\ell(v)$ is the *eccentricity* of vertex v, and $Adj(U) = \{w \in V : (u \in U \subseteq V)\ \{u, w\} \in E\}$. In particular, the *width* of a rooted level set is defined as $b(\mathscr{L}(v)) = \max_{0 \leq i \leq \ell(v)} |L_i(v)|$. It is therefore desirable to find v with large $\ell(v)$ and a rooted level set $\mathscr{L}(v)$ with small width.

Algorithm 1 shows a pseudocode of the Reverse Cuthill-McKee method [9]. The method receives a graph $G = (V, E)$ and a pseudoperipheral vertex $v \in V$. 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). The method reverses the final ordering. Therefore, the final label of vertex v is $|V|$ at line 2. 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 10.

Algorithm 2 shows the George-Liu algorithm [10]. The George-Liu 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 algorithm builds the rooted level structure $\mathscr{L}(u)$ of the vertex $u \in L_{\ell(v)}(v)$ with minimum degree (see lines 5 and 6). If $\ell(u) > \ell(v)$ (see line 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 stops and v is the pseudoperipheral vertex found (see line 10 in Algorithm 2).

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
           increasing degree) do
7              i ← i − 1;
8              s(i) ← w;
9          j ← j − 1;
10     return S;
```

Algorithm 1. The Reverse Cuthill-McKee method [9].

Input: graph $G = (V, E)$;
Output: pseudo-peripheral vertex $v \in V$;

```
1  begin
2      v ← ArbitraryVertex(V);
3      L(v) ← BuscaEmLargura(v); // build a rooted level structure
4      repeat
5          u ← MinimumDegreeVertex(L_ℓ(v)(v));
6          L(u) ← BuscaEmLargura(u); // build a rooted level structure
7          if ℓ(u) > ℓ(v) then
8              v ← u; L(v) ← L(u);
9      until (u ≠ v);
10     return v;
```

Algorithm 2. The George-Liu algorithm [10].

Algorithm 3 shows a pseudocode of a variant of the George-Liu algorithm [10]. This variant of the George-Liu [10] algorithm verifies k vertices in $L_{\ell(w)}(w)$ (note that v is attributed to w at line 6 of Algorithm 3) where k is a parameter of the algorithm, instead of verifying only one vertex with minimum degree in $L_{\ell(w)}(w)$, as the George-Liu algorithm [10] performs. If $|L_{\ell(w)}(w)| < k$, then the other $k - |L_{\ell(w)}(w)|$ vertices are verified in the previous levels of $L_{\ell(w)}(w)$, that is, in $L_{\ell(w)-1}(w), L_{\ell(w)-2}(w)$, and so on. In particular, the GL(k) algorithm is reduced to the original George-Liu algorithm [10] for $k = 1$.

Reid and Scott [17] proposed another pseudoperipheral vertex finder based on the George-Liu algorithm [10]. The Reid-Scott algorithm [17] relies on finding a pseudoperipheral vertex with a large eccentricity and small $b(\mathscr{L}(v))$. It sorts vertices in $L_{\ell(v)}(v)$ in order of ascending degree and limits the search to one representative of each degree. Additionally, the Reid-Scott algorithm [17] considers up to five vertices in $L_{\ell(v)}(v)$, omitting any that is a neighbor of a vertex already considered.

Input: connected undirected graph $G = (V, E)$; unsigned integer k;
Output: pseudoperipheral vertex $v \in V$;

```
1  begin
2      if (k > |V| − 1) then k ← |V| − 1;
3      v ← ArbitraryVertex(V); // initial vertex
4      ℒ(v) ← Breadth-First-Search(v); // build the rooted level structure
5      repeat
6          k ← 0; j ← 0; w ← v;
           // k vertices are verified in the last level(s) of ℒ(w)
7          while (j < k) do
               // sort vertices in L_{ℓ(w)−k}(w) in order of ascending degree
8              F ← SortVertices(L_{ℓ(w)−k}(w)); // F is a priority queue
9              while (F ≠ ∅ ∧ j < k) do
10                 u ← Remove(F);
                   // build the rooted level structure
11                 ℒ(u) ← Breadth-First-Search(u);
12                 if (ℓ(u) > ℓ(v)) then
13                     v ← u; ℒ(v) ← ℒ(u);
14                 j ← j + 1;
               // if |L_{ℓ(w)}(w)| < n, then the other n − |L_{ℓ(w)}(w)| vertices
                  are verified in L_{ℓ(w)−1}(w), L_{ℓ(w)−2}(w), ···,
15             k ← k + 1; // until k vertices are observed
16     until (v = w);
17     return (v);
```

Algorithm 3. A variant of the George-Liu algorithm [10].

3 Description of the Tests

The bandwidth results obtained by the Reverse Cuthill-McKee method [9] is proportional to the eccentricity of the starting vertex v and the width $b(\mathscr{L}(v))$ [9,11,18]. Thus, we evaluate the bandwidth and profile results provided by the Reverse Cuthill-McKee method [9] along with three pseudoperipheral vertex finders. To appraise the bandwidth and profile reductions provided by the Reverse Cuthill-McKee method [9] started with a pseudoperipheral vertex given by the George-Liu [10] and Reid-Scott [17] algorithms, and the new variant of the George-Liu algorithm described in Algorithm 3, this paper uses 50 symmetric matrices contained in the SuiteSparse sparse matrix collection [19].

The Reverse Cuthill-McKee method [9] and the algorithms for finding pseudoperipheral vertices were implemented using the C++ programming language. We used the g++ version 4.8.2 compiler.

The workstation used in the execution of the simulations featured an Intel® Core™ i5-3570 (6144 KB Cache, CPU 3.40 GHz, 12 GB of main memory DDR3 1333 MHz) (Intel; Santa Clara, CA, United States). This machine used a Slackware 14.1 64-bit operating system with Linux kernel-version 3.10.17.

4 Results and Analysis

This section presents the bandwidth and profile results obtained by the RCM-GL [11], RCM-RS, and RCM-GL(k) methods applied to 50 symmetric matrices contained in the SuiteSparse sparse matrix collection [19]. The tables below show the characteristics of the matrix (name, size (k), original bandwidth (β_0), and profile (profile$_0$)).

Tables 1 and 2 contain the results of the RCM-GL(k), RCM-RS, and RCM-GL [11] methods applied to reduce the bandwidth and profile of 15 and 35 symmetric matrices, respectively. The tables show the smallest k such that the RCM-GL(k) algorithm returns better results than the George-Liu [10] and Reid-Scott [17] algorithms do. Specifically, Table 1 shows the results of the methods applied to a set composed of 15 symmetric matrices. The table reveals that the three methods find the same number of best bandwidth and profile results when applied to this set composed of 15 symmetric matrices. On the other hand, Table 2 shows that the RCM-GL(k) method delivered the best bandwidth and profile results in the other dataset used in this study. Figure 1 shows the number of best bandwidth and profile results yielded by the three methods evaluated here when applied to 50 symmetric matrices.

Table 1. Results of the Reverse Cuthill-McKee method [9] along with three algorithms for finding pseudoperipheral vertices when applied to reduce bandwidth and profile of a set composed of 15 symmetric matrices.

Matrix	n	Bandwidth				Profile			
		β_0	GL(2)	RS	GL	profile$_0$	GL(2)	RS	GL
ash85	85	39	10	10	10	1153	589	589	589
bcspwr01	39	38	7	7	7	292	122	122	122
bcspwr02	49	34	13	13	13	377	234	234	234
bcspwr03	118	115	21	21	21	1288	804	804	804
bcsstk01	48	35	26	26	26	851	683	683	683
bcsstk04	132	47	54	54	54	3631	3717	3717	3717
bcsstk05	153	28	26	26	26	2449	2313	2313	2313
bcsstk22	138	111	14	14	14	2124	863	863	863
can_144	144	142	18	18	18	7355	1074	1074	1074
can_161	161	79	30	30	30	3378	3079	3079	3079
dwt_234	234	48	19	19	19	1765	1363	1363	1363
lund_A	147	23	23	23	23	2870	2303	2303	2303
lund_B	147	23	23	23	23	2870	2303	2303	2303
nos1	237	4	4	4	4	780	467	467	467
nos4	100	13	12	12	12	766	755	755	755
Number of best results		4	14	14	14	1	14	14	14

Table 2. Results of the Reverse Cuthill-McKee method [9] along with three algorithms for finding pseudoperipheral vertices when applied to reduce bandwidth and profile of a set composed of 35 symmetric matrices.

Matrix	n	Bandwidth				Profile			
		β_0	GL(23)	RS	GL	profile$_0$	GL(77)	RS	GL
494_bus	494	428	62	62	62	40975	10566	10566	10566
662_bus	662	335	85	135	135	45165	20664	32903	32903
685_bus	685	550	72	72	79	28621	17551	17551	17457
ash292	292	24	34	34	34	4224	4659	4659	4659
bcspwr04	274	265	49	65	65	21015	4331	4825	4825
bcspwr05	443	435	59	59	59	36248	8825	8825	8825
bcsstk06	420	47	50	50	50	14691	13241	13241	13241
bcsstk19	817	567	21	21	21	74051	9457	9457	9457
bcsstk20	485	20	19	19	19	4309	4416	4416	4416
bcsstm07	420	47	50	50	50	14691	13310	13310	13310
can_292	292	282	75	75	65	23170	8977	8977	9706
can_445	445	403	92	92	92	22321	23808	23808	23808
can_715	715	611	157	157	157	72423	41231	41293	41293
can_838	838	837	137	137	155	207200	34254	39198	40835
dwt_209	209	184	35	35	35	9503	3914	3914	3914
dwt_221	221	187	16	16	16	9910	2011	2011	2011
dwt_245	245	115	45	45	45	3934	4177	4177	4177
dwt_310	310	28	13	13	13	2696	2695	2695	2695
dwt_361	361	50	25	25	25	5084	5139	5139	5139
dwt_419	419	356	34	34	34	39726	8230	8232	8232
dwt_503	503	452	58	58	58	35914	14816	15544	15544
dwt_592	592	259	42	42	42	28805	10983	10983	10983
dwt_878	878	519	37	37	37	26055	21034	21034	21034
dwt_918	918	839	72	72	72	108355	24347	24347	24347
dwt_992	992	513	63	63	63	262306	36296	36296	36296
gr_30_30	900	31	59	59	59	26970	33872	33872	33872
jagmesh1	936	778	27	27	27	37240	21817	21817	21817
nos2	957	4	4	4	4	3180	1907	1907	1907
nos3	960	43	79	79	79	39101	46168	46168	46168
nos5	468	178	88	88	88	27286	25381	25381	25381
nos6	675	30	16	16	16	16229	9305	9305	9305
nos7	729	81	65	65	65	53144	34110	34110	34110
plat362	362	249	56	56	56	45261	11018	11018	11018
plskz362	362	248	24	24	24	43090	4635	4635	4635
sherman1	1000	100	57	57	57	34740	26109	26109	26109
Number of best results		5	**30**	28	27	6	**29**	23	23

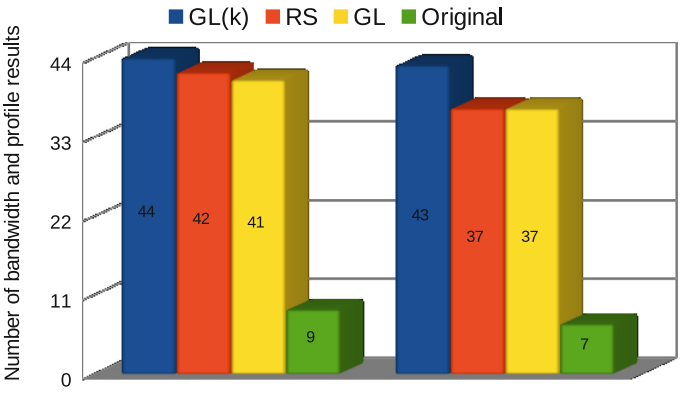


Fig. 1. The number of best bandwidth and profile results obtained using three pseudoperipheral vertex finders along with the Reverse Cuthill-McKee method [9] applied to 50 symmetric matrices when considering the results presented in Tables 1 and 2.

The execution times of the RCM-GL(k) method are proportional to the parameter used. Figure 2 (as line charts for clarity) shows the running times of the three methods when applied to the 50 symmetric matrices. The execution times of the RCM-GL(k) method were competitive with the RCM-GL method [11] when applied to the 50 symmetric matrices used in this study. Additionally, the execution times of the RCM-GL(k) algorithm was lower (or similar) than the processing times or the RCM-RS method.

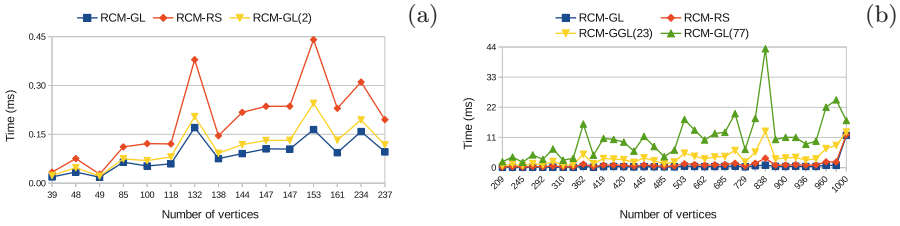


Fig. 2. Execution costs, in milliseconds, of the RCM-GL [11], RCM-RS, and RCM-GL(k) methods applied to sets composed of (a) 15 and (b) 35 symmetric matrices.

5 Conclusions

This paper evaluated a new variant of the George-Liu algorithm [10]. Specifically, this paper compared the results of the new algorithm with the results of the original George-Liu [10] and Reid-Scott [17] algorithms for finding initial vertices for the Reverse Cuthill-McKee method [9] when applied to 50 small symmetric matrices.

The results obtained by the new variant of the George-Liu algorithm applied to a set of standard benchmark matrices taken from the SuiteSparse sparse matrix collection [19] show that the proposed method compared favorably with the two others algorithms for finding pseudoperipheral vertices evaluated here when executed along with the Reverse Cuthill-McKee method [9]. Specifically, the simulations presented in this paper show that the Reverse Cuthill-McKee method [9] with ordering started with a pseudoperipheral vertex given by the new variant of the George-Liu algorithm obtains better results for reducing bandwidth and profile of small symmetric matrices than using the original George-Liu [10] or Reid-Scott [17] algorithms. Parallel approaches of these algorithms are the future steps of this investigation.

References

1. Benzi, M., Szyld, D.B., Van Duin, A.: Orderings for incomplete factorization preconditioning of nonsymmetric problems. SIAM J. Sci. Comput. **20**(5), 1652–1670 (1999)
2. Camata, J.J., Rossa, A.L., Valli, A.M.P., Catabriga, L., Carey, G.F., Coutinho, A.L.G.A.: Reordering and incomplete preconditioning in serial and parallel adaptive mesh refinement and coarsening flow solutions. Int. J. Numer. Meth. Fluids **69**(4), 802–823 (2012)
3. Chagas, G.O., Gonzaga de Oliveira, S.L.: Metaheuristic-based heuristics for symmetric-matrix bandwidth reduction: a systematic review. Procedia Comput. Sci. **51**, 211–220 (2015). https://doi.org/10.1016/j.procs.2015.05.229
4. Gonzaga de Oliveira, S.L., Bernardes, J.A.B., Chagas, G.O.: An evaluation of reordering algorithms to reduce the computational cost of the incomplete Cholesky-conjugate gradient method. Comput. Appl. Math. **37**(3), 2965–3004 (2018)
5. Gonzaga de Oliveira, S.L., Bernardes, J.A.B., Chagas, G.O.: An evaluation of low-cost heuristics for matrix bandwidth and profile reductions. Comput. Appl. Math. **37**, 1412–1471 (2018)
6. Bernardes, J.A.B., Gonzaga de Oliveira, S.L.: A systematic review of heuristics for profile reduction of symmetric matrices. Procedia Comput. Sci. **51**, 221–230 (2015)
7. Kaveh, A., Bijari, S.: Bandwidth, profile and wavefront optimization using PSO, CBO, ECBO and TWO algorithms. Iran. J. Sci. Technol. Trans. Civ. Eng. **41**(1), 1–12 (2017)
8. Kaveh, A.: Bandwidth, profile, and wavefront optimization using CBO, ECBO, and TWO algorithms. Applications of Metaheuristic Optimization Algorithms in Civil Engineering, pp. 235–256. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-48012-1_13
9. George, A.: Computer implementation of the finite element method. Ph.D. thesis, Stanford University, Stanford (1971)
10. George, A., Liu, J.W.H.: An implementation of a pseudoperipheral node finder. ACM Trans. Math. Softw. **5**(3), 284–295 (1979)
11. George, A., Liu, J.W.: Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Englewood Cliffs (1981)
12. Gonzaga de Oliveira, S.L., Abreu, A.A.A.M., Robaina, D.T., Kischnhevsky, M.: An evaluation of four reordering algorithms to reduce the computational cost of the Jacobi-preconditioned conjugate gradient method using high-precision arithmetic. Int. J. Bus. Intell. Data Min. **12**(2), 190–209 (2017)

13. Gilbert, J.R., Moler, C., Schreiber, R.: Sparse matrices in MATLAB: design and implementation. SIAM J. Matrix Anal. **3**(1), 333–356 (1992)
14. The MathWorks, Inc.: MATLAB (1994–2018). http://www.mathworks.com/products/matlab/
15. Eaton, J.W., Bateman, D., Hauberg, S., Wehbring, R.: GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations (2015)
16. Boost: Boost C++ libraries (2017). http://www.boost.org/. Accessed 28 June 2017
17. Reid, J.K., Scott, J.A.: Ordering symmetric sparse matrices for small profile and wavefront. Int. J. Numer. Meth. Eng. **45**(12), 1737–1755 (1999)
18. Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: ACM Proceedings of the 1969 24th International Conference, pp. 157–172. ACM, New York (1969)
19. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Softw. **38**(1), 1–25 (2011)