A Modified Bandwidth Reduction Heuristic Based on the WBRA and George-Liu Algorithm

Sanderson L. Gonzaga de Oliveira[1(✉)], Guilherme O. Chagas[1], Diogo T. Robaina[2], Diego N. Brandão[3], and Mauricio Kischinhevsky[2]

[1] Universidade Federal de Lavras, Lavras, Minas Gerais, Brazil
sanderson@dcc.ufla.br, guilherme.chagas@computacao.ufla.br
[2] Universidade Federal Fluminense, Niterói, Rio de Janeiro, Brazil
{drobaina,kisch}@ic.uff.br
[3] CEFET-RJ, Nova Iguaçu, Rio de Janeiro, Brazil
diego.brandao@eic.cefet-rj.br

Abstract. This paper presents a modified heuristic based on the Wonder Bandwidth Reduction Algorithm with starting vertex given by the George-Liu algorithm. The results are obtained on a dataset of instances taken from the SuiteSparse matrix collection when solving linear systems using the zero-fill incomplete Cholesky-preconditioned conjugate gradient method. The numerical results show that the improved vertex labeling heuristic compares very favorably in terms of efficiency and performance with the well-known GPS algorithm for bandwidth and profile reductions.

1 Introduction

An undirected sparse graph is often used to represent a sparse symmetric matrix A. As a result, such matrices play an important role in graph theory. In order to improve cache hit rates, one can reduce processing times when using the conjugate gradient method by finding an adequate ordering to the vertices of the corresponding graph that represents the matrix A. This reduction in execution times is possible because program code and data have spatial and temporal locality. To provide more specific detail, row and column permutations of A can be obtained by reordering the vertices of the corresponding graph. A heuristic for bandwidth reduction returns an adequate ordering of graph vertices. Consequently, a heuristic for bandwidth reduction provides adequate memory location and cache coherency. Then, the use of vertex reordering algorithms is a powerful technique for reducing execution costs of linear system solvers. Thus, the computational cost of the preconditioned conjugate gradient method (i.e., the linear system solver studied here) can be reduced using appropriately the current architecture of memory hierarchy and paging policies. Hence, much attention has been given recently to the bandwidth and profile reduction problems (see [1,2,9] for discussions and lists of references). This paper considers heuristics for preprocessing

Y. Shi et al. (Eds.): ICCS 2018, LNCS 10862, pp. 416–424, 2018.
https://doi.org/10.1007/978-3-319-93713-7_35

matrices to reduce the running time for solving linear systems on them. Specifically, we consider the ordering of symmetric sparse positive definite matrices for small bandwidth and profile. 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 $\beta(G) = \max_{\{v,u\} \in E} [|s(v) - s(u)|]$, 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 $A = [a_{ij}]$ be an $n \times n$ symmetric matrix associated with a connected undirected graph $G = (V, E)$. Equivalently, the bandwidth of row i of matrix A is given by $\beta_i(A) = i - \min_{1 \leq j \leq i} [j : a_{ij} \neq 0]$, noting that a_{ij} represents the left-most non-null coefficient (i.e., in column j) in row i in matrix A. Thus, 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 bandwidth and profile minimization problems are NP-hard [10,11]. Since these problems have connections with a wide number of important applications in science and engineering, efforts should be continued to develop low-cost heuristics that are capable of producing good-quality approximate solutions. Thus, this paper gives an improved heuristic for bandwidth reduction, and our main objective here is to accelerate a preconditioned conjugate gradient method. A systematic review [9] reported the Wonder Bandwidth Reduction Algorithm (WBRA) [4] as a promising heuristic for bandwidth reduction. This present paper proposes a variant of this algorithm to provide a heuristic for bandwidth reductions with lower execution times and smaller memory requirements than the original algorithm. To be more precise, the purpose of this paper is to propose a modified WBRA with starting vertex given by the George-Liu algorithm [5]. Thereby, this paper consists of improving the Wonder Bandwidth Reduction Algorithm (WBRA) [4] for bandwidth reduction aiming at accelerating the zero-fill incomplete Cholesky-preconditioned conjugate gradient (ICCG) method.

This present work evaluates experimentally the results of the modified WBRA against the GPS algorithm [6] aiming at reducing the execution costs of the ICCG method in sequential runs applied to instances arising from real-world applications. The remainder of this paper is organized as follows. Section 2 presents the improved heuristic proposed in this work. Section 3 shows numerical experiments that compares the modified heuristic for bandwidth reduction with the original WBRA [4] and GPS ordering [6]. Finally, Sect. 4 addresses the conclusions.

2 An Improved Vertex Labeling Heuristic

Heuristics for bandwidth reduction should execute at low processing times because the main objective of bandwidth reduction is to reduce computational costs when solving linear systems [1,2]. The WBRA obtained promising

bandwidth reductions in the experiments presented by Esposito *et al.* [4]. In particular, it achieved better bandwidth results than the GPS algorithm [6]. Nevertheless, in our exploratory investigations, the WBRA reached high computational times and large memory requirements. This occurs because, depending on the instance, many initial vertices may be chosen in its first step and, consequently, various structures are stored and several reordering processes are performed. Sections 2.1 and 2.2 describe the WBRA and the improved WBRA heuristic, respectively.

2.1 The WBRA

The WBRA [4] is based on graph-theoretical concepts. Let $G = (V, E)$ be a connected, simple, and undirected graph, where V and E are sets of vertices and edges, respectively. Given a vertex $v \in V$, the level structure rooted at v, with depth $\ell(v)$ (or the eccentricity of v), is a partitioning $\mathscr{L}(v) = \{L_0(v), L_1(v), \ldots, L_{\ell(v)}(v)\}$, where $L_0(v) = \{v\}$, $L_i(v) = Adj(L_{i-1}(v)) - \bigcup_{j=0}^{i-1} L_j(v)$, for $i = 1, 2, 3, \ldots, \ell(v)$ and $Adj(U) = \{w \in V : (u \in U \subseteq V)\ \{u, w\} \in E\}$. The width of a rooted level structure is defined as $b(\mathscr{L}(v)) = \max_{0 \leq i \leq \ell(v)} [|L_i(v)|]$. The WBRA presents two main steps. In the first step, the WBRA builds level structures rooted at each vertex of the graph, i.e., $|V|$ rooted level structures are built. Then, it reduces the width of each rooted level structure generated using a heuristic named Push_Up [4]. This process is performed level by level of a level structure, considering the bottleneck linear assignment problem. It is performed by moving some vertices from a level with a large number of vertices to adjacent levels of the level structure, satisfying the condition that, for each edge $\{v, u\} \in E$, the vertices v and u belong to the same level of the level structure or belong to adjacent levels. This can convert a rooted level structure in a level structure $\mathscr{K}(\nu, \cdots)$ that is not rooted at a specific vertex. After reducing the width of each level structure, each level structure $\mathscr{K}(\nu, \cdots)$ with width $b(\mathscr{K}(\nu, \cdots))$ smaller than the original bandwidth $\beta(A)$ [i.e., $(\nu \in V)\ b(\mathscr{K}(\nu, \cdots)) < \beta(A)$] is stored in a priority queue organized in ascending order of width $b(\mathscr{K}(\nu, \cdots))$. Hence, in the worst case from the point of view of memory usage, the WBRA takes $O(|V|^2)$ since it can store $|V|$ level structures and a level structure grows in $\Theta(|V|)$ memory.

In its second step, the WBRA labels the vertices of the graph. In the reordering procedure, the WBRA labels the vertices of each level structure stored in the priority queue. The WBRA removes a level structure $\mathscr{K}(\nu, \cdots)$ from the priority queue and sorts the vertices of each level of the level structure in ascending-degree order. The labeling procedure reduces the bandwidths of submatrices generated by labeling the vertices of each level of the level structure. Finally, this algorithm returns the labeling with the smallest bandwidth found. Therefore, the WBRA [4] shows high computational costs because it builds a level structure rooted at each vertex of a graph and these structures are rearranged based on a bottleneck linear assignment. This strategy makes the whole procedure costly in both time and space. Thus, the WBRA for bandwidth reduction of symmetric matrices did not show competitive results in our exploratory investigations when compared

with other heuristics, such as the GPS algorithm [6], when aiming at reducing the computational costs of a linear system solver applied to large instances.

2.2 An Improved Heuristic for Bandwidth Reduction

This paper proposes a modification in the first step of WBRA [4] aiming at obtaining a low-cost heuristic. The improved heuristic proposed here consists of computing only one level structure used by WBRA rooted at a carefully chosen vertex given by the George-Liu algorithm [5] instead of building level structures rooted at each vertex (i.e., $|V|$ rooted level structures) as it is performed in the original WBRA [4]. We will refer this new algorithm as ECMT-GL heuristic. The main difference between the WBRA and ECMT-GL heuristic is in their first steps.

Given a rooted level structure $\mathscr{L}(v)$, the Push_Up heuristic is also applied in the ECMT-GL heuristic to reduce the width $b(\mathscr{L}(v))$, resulting in a level structure $\mathscr{K}(\nu, \cdots)$. Then, only $\mathscr{K}(\nu, \cdots)$ is stored and no priority queue is used. Consequently, the ECMT-GL heuristic grows in $\Theta(|V|)$ memory. Subsequently, similar steps of the WBRA are performed in the ECMT-GL heuristic, with the difference that the labeling procedure is performed using only a single level structure $\mathscr{K}(\nu, \cdots)$, resulting in a low-cost heuristic for bandwidth reduction.

The original WBRA cannot have the same storage requirements than the ECMT-GL heuristic. As described, the original WBRA builds a level structure rooted at each vertex contained in V (then $|V|$ rooted level structures are built) and stores in a priority queue those rooted level structures with width smaller than the original bandwidth $\beta(A)$, which gives the memory requirements of the algorithm. After removing a level structure from the priority queue, the original WBRA labels the vertices and stores only the current level structure that provides the smallest bandwidth found. To have a memory usage of one level structure, the original WBRA would have to label the vertices according to all $|V|$ rooted level structures (instead of labeling the vertices of rooted level structures stored in the priority queue), resulting in an algorithm extremely expensive. Therefore, the variant of the WBRA proposed here improves the original algorithm in terms of processing and storage costs. Esposito *et al.* [4] performed experiments with instances up to 2,000 vertices. Moreover, the authors provided no storage cost analysis of the heuristic. Our exploratory investigations showed that the WBRA presents large memory requirements so that the WBRA is impractical to be applied even to an instance composed of appropriately 20,000 unknowns (what also depends on the number of non-null coefficients of the instance). Thus, our modification in the WBRA increases its efficiency and performance.

3 Results and Analysis

This section presents the results obtained in simulations using the ICCG method, computed after executing heuristics for vertex labeling. Thus, the results of the

improved heuristic proposed in this paper are compared with the results from the use of the GPS algorithm [6].

The WBRA [4], GPS [6], and ECMT-GL heuristics were implemented using the C++ programming language. In particular, the g++ version 4.8.2 compiler was used. Additionally, a data structure based on the Compress Row Storage, Compress Column Storage, and Skyline Storage Scheme data structures was used within the implementation of the ICCG method.

The experiments were performed on an Intel® Core™ i7-2670QM CPU @ 2.20 GHz (6 MB cache, 8 GB of main memory DDR3 1333 MHz) (Intel; Santa Clara, CA, United States) workstation. The Ubuntu 16.04.3 LTS 64-bit operating system was used and the Linux kernel-version 4.4.0-97-generic is installed in this workstation. Three sequential runs, with both a reordering algorithm and with the preconditioned conjugate gradient method, were performed for each instance. A precision of 10^{-16} to the CG method using double-precision floating-point arithmetic was employed in all experiments. Table 1 shows the name and number of non-null matrix coefficients, the dimension n (or the number of unknowns) of the respective coefficient matrix of the linear system (or the number of vertices of the graph associated with the coefficient matrix), the name of the heuristic for bandwidth reduction applied, the results with respect to bandwidth and *profile* reductions, the results of the heuristics in relation to the execution costs [t(s)], in seconds (s), when applied to nine real symmetric positive definite matrices taken from the SuiteSparse matrix (SSM) collection [3]. Moreover, column % in this table shows the percentage (rounded to the nearest integer) of the execution time of the heuristics for bandwidth reduction in relation to the overall execution time of the simulation. In addition, the same table shows the number of iterations and the total running time, in seconds, of the ICCG method applied to the linear systems. The first rows in each instance in Table 1 show "—". This means that no reordering heuristic was used in these simulations. This makes it possible to check whether the use of a heuristic for bandwidth and profile reductions decreases the execution times of the linear system solver. The last column in this table shows the speed-up/down of the ICCG method. Table 1 and Figs. 1(a)–(b) reveal that the execution and memory costs of the ECMT-GL heuristic is significantly lower than the computational cost of the WBRA (that was applied to the *LFAT5000* instance). In particular, the three heuristics evaluated here obtained the same bandwidth and profile results when applied to the *LFAT5000* instance. The executions with the WBRA applied to the *cvxbqp1*, *Pres_Poisson*, and *msc10848* instances were aborted because the high consumption of time and memory. Thus, Table 1 and Figs. 1(a)–(b) show that the new ECMT-GL heuristic outperforms the WBRA in terms of running time and memory consumption.

Even though the ECMT-GL heuristic reduces bandwidth and profile to a considerable extent, Table 1 and Fig. 2 also show that the GPS algorithm [6] obtained in general better bandwidth and profile results than the ECMT-GL heuristic. The ECMT-GL heuristic achieved better bandwidth (profile) results than the GPS algorithm [6] when applied to the *cant* (*cvxbqp1*, *cant*) instance. In particular, the GPS ordering [6] increased the bandwidth and profile of the

Table 1. Results of solutions of nine linear systems (ranging from 79,966 to 4,444,880 non-null coefficients ($|E|$)) contained in the SuiteSparse matrix collection [3] using the ICCG method and vertices labeled by the GPS [6] and ECMT-GL heuristics.

Instance	$\|E\|$	n	Heuristic	β	Profile	t(s)	%	ICCG:iter	ICCG:t(s)	Speedup
LFAT 5000	79966	19994	—	5	84958	—	—	19994	19	—
			ECMT-GL	3	34984	0.1	1	19994	18	1.04
			GPS	3	34984	2.9	14	19994	18	0.90
			WBRA	3	34984	65.7	74	19994	23	0.21
cvxbqp1	349968	50000	—	33333	819815962	—	—	50000	485	—
			ECMT-GL	2240	53110785	6.8	2	50000	280	1.69
			GPS	1690	53398841	12.8	4	50000	332	1.41
thermo mech_TK	711558	102158	—	102138	2667823445	—	—	11121	459	—
			ECMT-GL	364	18788072	2.8	1	9090	307	1.48
			GPS	270	18153270	125.6	31	9725	284	1.12
Pres_Poisson	715804	14822	—	12583	9789525	—	—	349	9	—
			ECMT-GL	614	3166017	0.7	9	233	7	1.09
			GPS	334	2916499	23.0	70	206	10	0.27
msc23052	1142686	23052	—	23046	183660755	—	—	23052	213	—
			ECMT-GL	1992	19046925	7.3	20	1	30	5.75
			GPS	1462	17122364	27.4	54	1	23	4.20
msc10848	1229776	10848	—	10706	50044035	—	—	10848	63	—
			ECMT-GL	1963	11833479	7.8	49	1	8	4.03
			GPS	1694	6826017	14.7	57	1	11	2.45
ct20stif	2600295	52329	—	49323	162549961	—	—	52329	1063	—
			ECMT-GL	4091	107413396	46.0	22	1	159	5.18
			GPS	3329	107013694	121.2	43	1	159	3.79
cant	4007383	62451	—	275	16778583	—	—	24	272	—
			ECMT-GL	275	16778583	0.3	0	1	254	1.07
			GPS	302	16976671	1018.9	79	1	265	0.21
thread	4444880	29736	—	29339	177235110	—	—	29736	835	—
			ECMT-GL	3804	51720915	42.5	28	1	110	5.49
			GPS	3521	45119385	387.3	75	1	132	1.61

cant instance. Table 1 also shows that in general the number of ICCGM iterations was the same when using the ECMT-GL heuristic and GPS algorithm [6]. The number of ICCGM iterations was smaller using the ECMT-GL (GPS [6]) heuristic than using the GPS (ECMT-GL) algorithm when applied to the *thermomech_TK* (*Pres_Poisson*) instance. Nevertheless, the most important issue in this context is to reduce the execution time of the ICCG method. Specifically, if high cache hit rates are achieved, a large number of CGM iterations can be executed faster than a smaller number of iterations with high cache miss rates. In this scenario, Table 1 [see column ICCG:t(s)] shows lower processing times of the ICCG method in simulations in conjunction with the GPS algorithm [6] than in conjunction with the ECMT-GL heuristic only when applied to the *thermomech_TK* and *msc23052* instances. In particular, column t(s) in Table 1 and Fig. 1(a) show that the execution costs of the ECMT-GL heuristic are lower than the GPS algorithm [6]. Figure 1(b) shows that the storage costs of the ECMT-GL heuristic is similar to the GPS algorithm [6]. Moreover, both reordering algorithms shows linear expected memory requirements, but the hidden con-

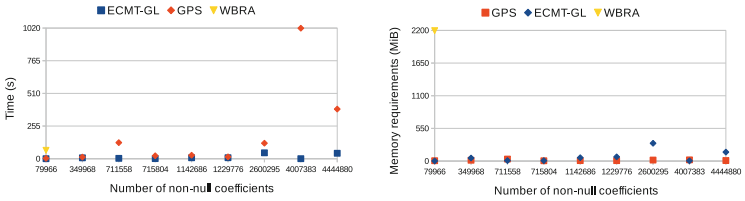

Fig. 1. Running times and memory requirements of heuristics for bandwidth and profile reductions when applied to nine instances contained in the SuiteSparse matrix collection [3].

stant in the GPS algorithm [6] is slightly smaller than the hidden constant in the ECMT-GL heuristic.

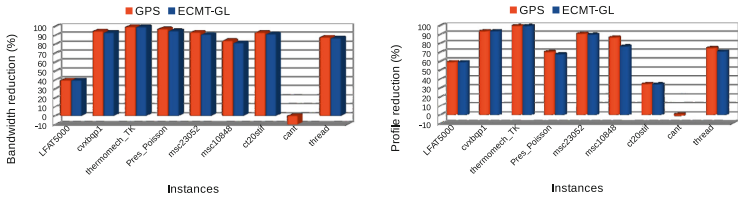

Fig. 2. Bandwidth and profile reductions (in percentage) obtained by the ECMT-GL and GPS [6] heuristics when applied to nine instances contained in the SSM collection [3].

Although the GPS algorithm [6] obtains in general better bandwidth and profile results, low execution times of the ECMT-GL heuristic pay off in reducing the running time of the ICCG method. Specifically, the ECMT-GL heuristic obtained better results (related to speedup of the ICCG method) than the GPS ordering [6] in all experiments, including when using the *thermomech_TK* and *msc23052* instances (at least when considering that only a single linear system is to be solved). As mentioned, the reason for this is that the ECMT-GL heuristic achieved similar performance to the GPS algorithm [6] at lower execution times than the GPS algorithm [6]. Specifically in the simulation using the *thermomech_TK* and *msc23052* instances (situation which could also happen in the cases of solving linear systems composed of multiple right hand side vectors and in transient FEM analysis in which linear systems are solved multiple times), the initial cost for reordering can be amortized in the iterative solution steps. In this scenario, a reordering scheme can be beneficial if it presents a reasonable cost (time and memory usage) and can reduce the runtime of the linear system solver at a higher level (such as in the case of the GPS algorithm [6] when applied to the *thermomech_TK* and *msc23052* instances).

Table 1 and Fig. 3 show the average results from the use of the ICCG method applied to nine instances taken from the SSM collection [3]. The ECMT-GL

heuristic improved significantly the performance of the ICCG method when applied to the *msc23052*, *msc10848*, *ct20stif*, and *thread* instances. Table 1 and Figs. 1(a) and 3 show that on average the new ECMT-GL heuristic outpaced the GPS algorithm in terms of output quality (regarding to the speedup of the ICCG method) and running time in the simulations performed. Column % in Table 1 shows that the execution time of a heuristic for bandwidth reduction can be roughly the same order of magnitude as the ICCG computation (e.g. see results concerning the *msc10848* instance). Setting a lower precision to the CG method can let the reordering time more evident. Therefore, it is useful to parallelize the reordering and this is a next step in this work.

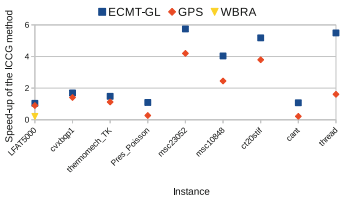

Fig. 3. Speed-ups of the ICCG method resulted when using heuristics for bandwidth and profile reductions applied to nine instances contained in the SSM collection [3].

4 Conclusions

This paper proposes a new approach to the Wonder Bandwidth Reduction Algorithm (WBRA) [4] based on the George-Liu algorithm [5] for matrix bandwidth reduction. This variant of the WBRA was termed ECMT-GL heuristic. Specifically, our approach employs a pseudo-peripheral vertex selected by the George-Liu algorithm [5] in the first step of WBRA. The results of the implementations of the WBRA [4] and GPS algorithm [6] described in this paper confirm their merit in accordance with the findings presented in the current literature, i.e., Table 1 and Fig. 2 show that these algorithms reduce the bandwidth and profile substantially. Specifically, the heuristics for bandwidth reductions evaluated in this computational experiment can enhance locality in accessing data and enable column (profile) information compression. Moreover, numerical experiments showed that the heuristic for bandwidth reduction proposed here provides further worthwhile gains when compared with the original WBRA [4] and GPS algorithm [6]. Nine model problems taken from the SuiteSparse matrix collection [3] were used to examine the efficiency of the proposed algorithm. These model problems are solved with the conjugate gradient method in conjunction with the zero-fill incomplete Cholesky preconditioner.

The performance of the proposed algorithm was compared with the original WBRA [4]. The new heuristic uses less memory and has advantageous performance properties in relation to the original algorithm. Thus, the modification in

the WBRA proposed here reduces the amount of time and memory required by the algorithm to a considerable extent.

The results of the ECMT-GL heuristic were also compared with the results of the well-known GPS algorithm [6]. In experiments using nine instances (with some of them comprised of more than 4,000,000 non-null coefficients), the ECMT-GL heuristic performed best in reducing the computational cost of the ICCG method. We conclude that a high-cost heuristic for bandwidth reduction based on rooted level structures can be improved by starting the algorithm with a single vertex carefully chosen by a pseudoperipheral vertex finder (such as the George-Liu algorithm [5]). Although the original high-cost heuristic will probably yield better bandwidth results for exploring a larger domain space, low execution and storage costs of the new heuristic will pay off in reducing the execution costs of linear system solvers. This makes it possible to improve a number of heuristics developed in this field [1,7–9]. We plan to investigate parallel approaches of these algorithms and compare them along with algebraic multigrid methods in future studies.

References

1. 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)
2. 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)
3. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Software **38**(1), 1–25 (2011)
4. Esposito, A., Catalano, M.S.F., Malucelli, F., Tarricone, L.: A new matrix bandwidth reduction algorithm. Oper. Res. Lett. **23**, 99–107 (1998)
5. George, A., Liu, J.W.H.: An implementation of a pseudoperipheral node finder. ACM Trans. Math. Software **5**(3), 284–295 (1979)
6. Gibbs, N.E., Poole, W.G., Stockmeyer, P.K.: An algorithm for reducing the bandwidth and profile of a sparse matrix. SIAM J. Numer. Anal. **13**(2), 236–250 (1976)
7. 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**(2), 1412–1471 (2018). https://doi.org/10.1007/s40314-016-0394-9
8. 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. (2017). https://doi.org/10.1007/s40314-017-0490-5
9. Gonzaga de Oliveira, S.L., Chagas, G.O.: A systematic review of heuristics for symmetric-matrix bandwidth reduction: methods not based on metaheuristics. In: Proceedings of the Brazilian Symposium on Operations Research (SBPO 2015), Sobrapo, Ipojuca, Brazil (2015)
10. Lin, Y.X., Yuan, J.J.: Profile minimization problem for matrices and graphs. Acta Mathematicae Applicatae Sinica **10**(1), 107–122 (1994)
11. Papadimitriou, C.H.: The NP-completeness of bandwidth minimization problem. Comput. J. **16**, 177–192 (1976)