Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, v. 6, n. 1, 2018.

Trabalho apresentado no XXXVII CNMAC, S.J. dos Campos - SP, 2017.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

The use of the reverse Cuthill-McKee method with an alternative pseudo-peripheral vertice finder for profile optimization

Sanderson L. Gonzaga de Oliveira and Alexandre Abreu[1]
Departamento de Ciência da Computação, UFLA, Lavras, MG

Abstract. The need to determine pseudo-peripheral vertices arises from several methods for ordering sparse matrix equations. This paper evaluates an alternative algorithm for finding such vertices based on the Kaveh-Bondarabady algorithm. Specifically, this paper evaluates a variation of this algorithm against the original algorithm and the George-Liu algorithm. Extensive experiments among these algorithms in conjunction with the reverse Cuthill-McKee method suggest that the modified algorithm is a suitable alternative for reducing profile of symmetric matrices.

Keywords. Sparse matrices, Graph labeling, Graph algorithm, Reverse Cuthill-McKee method, Profile reduction, Graph theory.

1 Introduction

Several problems in modern engineering 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, b is a known vector of length n, and the unknown vector x of length n is sought. Particularly, an efficient solution using a direct method demands 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 the equations to be processed in a proper order to compute the solution efficiently. Thus, for the profile method to present low computational cost it is necessary to order the equations. Specifically in finite element analysis, in the case of one degree-of-freedom per node, performing a vertex reordering is equivalent to reorder the equations. Similarly, performing a vertex reordering is also equivalent to reorder the equations in partial differential equations finite-volume discretizations. Additionally, many solvers of sparse linear systems are computed after a matrix reordering to reduce the fill-in during Gaussian elimination. In addition, the computational cost of iterative solvers for the numerical solution of sparse linear system of equations can be reduced by using a heuristic for matrix profile reductions [2, 10]. To provide more specific detail, the transfer of information to and from memory is related

[1]sanderson@dcc.ufla.br,alexandregrandeabreu@gmail.com

DOI: 10.5540/03.2018.006.01.0441

to the number of non-null coefficients in the lines of the matrix system A. Cache misses are minimized if the non-null coefficients in each line lie in the same level of the memory hierarchy. Thus, cache misses are associated with the profile of A.

Let A be a symmetric adjacency matrix associated with a connected undirected graph $G = (V, E)$, where V and E are sets of vertices and edges, respectively. The profile of a matrix A is defined as $profile(A) = \sum_{i=1}^{n}[i - \min((1 \leq j < i) \;\; a_{ij} \neq 0)]$.

Several algorithms for profile reductions are based on a data structure known as *rooted level structure* (RLS) [10]. 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 $Adj(.)$ returns the adjacent vertices to the vertices of the argument. In particular, $\ell(v) = \max\{(\forall u \in V) \; d(v, u)\}$ 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 an RLS $\mathscr{L}(u)$ is defined as $b(\mathscr{L}(u)) = \max((0 \leq i \leq \ell(u)) \; |L_i(u)|)$.

Results of many graph theoretic-based heuristics for profile reductions depend upon the choice of a starting vertex [10]. A peripheral vertex is a proper starting vertex for heuristics for profile reductions. However, finding peripheral vertices in graphs is computationally expensive, that is, for a graph $G = (V, E)$, one can find a peripheral vertex by executing $|V|$ breadth-first search procedures, obtaining $O(|V|(|V| + |E|))$ in these operations. As a result, many heuristics for profile reductions use *pseudo-peripheral vertices* (PPVs) instead of using peripheral vertices. There exists alternative algorithms for finding peripheral vertices, including Arany's algorithm [1], but these algorithms are still computationally expensive when compared with an algorithm for finding a PPV. Thus, several heuristics for profile reductions require as a first step the determination of a PPV.

This paper evaluates an alternative low-cost algorithm for finding PPVs. Section 2 describes this algorithm. Section 3 describes how the simulations were conducted in this study and shows the results. Finally, Section 4 provides the conclusions.

2 Algorithms for finding pseudo-peripheral vertices

The George-Liu (GL) algorithm [6] for finding pseudo-peripheral vertices is available on the MATLAB software [12], in conjunction with the reverse Cuthill-McKee (RCM) method [5]. The RCM-GL method is $O(|V|+|E|)$ and shows low computational costs [10]. This algorithm was implemented and evaluated in our computational experiment.

Algorithm 1 shows the Kaveh and Bondarabady [11]. We will refer this algorithm as KB2. This algorithm selects a vertex v of minimum degree (in line 2). Then, it generates an RLS $\mathscr{L}(v)$ (in line 3), computes its width $b(\mathscr{L}(v))$ (in line 4), and selects one vertex u of minimum degree (in line 8) from each level $L_i(v)$ of $\mathscr{L}(v)$ (see the for loop in lines 7-16 of Algorithm 1). The KB2 algorithm generates an RLS $\mathscr{L}(u)$ from each of such vertices (in line 9), computes its width $b(\mathscr{L}(u))$ (in line 10), and choose the one corresponding to the smallest width (in lines 11-15). It repeats the process as far as reduction in width of the current RLS can be observed (see the repeat-until loop in lines 7-17 of Algorithm 1). Finally, the Kaveh-Bondarabady algorithm returns a pseudo-peripheral vertex s with a

DOI: 10.5540/03.2018.006.01.0441

small $b(\mathscr{L}(s))$ (in line 18).

```
   Input: graph G = (V, E);
   Output: pseudo-peripheral vertex s ∈ V;
 1 begin
 2     v ← VertexMinDegree(V);
       // build the RLS 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 RLS L(v)
 7         for (i ← 1 to ℓ(v)) do
 8             u ← VertexMinDegree(L_i(v));
               // build the RLS L(u)
 9             L(u) ← Breadth-First-Search-variant(u);
10             w ← b(L(u));
11             if (w < width) then
12                 width ← w;
13                 v ← u;
14                 L(v) ← L(u);
15             end
16         end
17     until (v = s);
18     return s;
19 end
```

Algorithm 1: Kaveh-Bondarabady algorithm (KB2) [11].

We implemented and evaluated in this computational experiment a slightly modification in the Kaveh-Bondarabady algorithm [11] by starting with an arbitrary vertex instead of starting with a vertex with minimum degree (in line 2 of Algorithm 1). We will refer this algorithm as MKB2. We evaluated this simple variation aiming at obtaining satisfactory performance in the "average case". An advantage of this variation over the original algorithm is simplicity and the modified algorithm runs faster than the original algorithm. Moreover, this modified algorithm may be more efficient than the original algorithm because the exact identification of an element may not compensate the computational effort employed. Section 3 shows that the modified algorithm obtains better profile results in symmetric matrices than the original Kaveh-Bondarabady algorithm [11] when providing starting vertices to the RCM method [4].

We evaluated also to choose an arbitrary vertex instead of selecting a vertex of minimum degree (in line 8 of Algorithm 1). This modification did not improve the algorithm so that it was discarded.

DOI: 10.5540/03.2018.006.01.0441

3 Description of the tests, results, and analysis

Three algorithms for finding PPVs were implemented and evaluated in this computational experiment (George-Liu [6], KB2 [11], and an alternative algorithm for finding PPVs described in Section 2) in conjunction with the reverse Cuthill-McKee (RCM) method [4]. Thus, these three algorithms (together with the RCM method) are named RCM-GL, RCM-KB2, and RCM-MKB2, respectively.

Among 74 heuristics for profile reductions found, a systematic review [2] reports the RCM method as one of the most promising heuristics for profile reductions. In addition, this method was applied in our computational experiment because it is possibly one of the most commonly vertex renumbering strategy used [2], and is available in MATLAB [12] and on Boost C++ Library (http://www.boost.org/doc/libs/1_58_0/libs/graph/doc/cuthill_mckee_ordering.html). Moreover, the RCM-GL method dominated several other reordering algorithms (e.g. see [2, 7–9]).

The algorithms were implemented in the C++ programming language. Specifically, the g++ version 4.8.2 compiler was used. To evaluate the profile reductions provided by these algorithms, 50 symmetric instances contained in the Harwell-Boeing sparse matrix collection (*http://math.nist.gov/MatrixMarket/data/Harwell-Boeing* [3]) were used.

The workstation used in the execution of the simulations contained an Intel® Core™ i5-3570 (6144KB Cache, CPU @ 3.40GHz, 12GB of main memory DDR3 1333MHz) (Intel; Santa Clara, CA, United States). The Slackware 14.1.64 64-bit operating system with Linux kernel-version 3.10.17 was used.

Table 1 shows the instance's name and size (n), the value of the initial profile ($profile_0$) of the instance, and the average value of profile obtained by each algorithm in 10 executions carried out in each instance. In this set composed of 50 symmetric instances, one can verify that the RCM-MKB2 algorithm showed the best profile results: this algorithm obtained the largest number of best results (in 34 instances) in this dataset.

Figure 1 shows computational times obtained using these three heuristics for profile reductions when applied to 50 instances contained in the Harwell-Boeing sparse matrix collection. This figure shows that the RCM-MKB2 is faster than the RCM-MKB heuristic and that the RCM-MKB2 shows computational times similar to the RCM-GL method.

4 Conclusions

A modified Kaveh-Bondarabady algorithm (MKB2) for the identification of pseudo-peripheral vertices was evaluated in this work. Results of the MKB2 algorithm were presented in conjunction with the reverse Cuthill-McKee method [4]. The RCM-MKB2 method showed better profile results when applied to 50 symmetric instances contained in the Harwell-Boeing sparse matrix collection than the RCM method with starting vertex given by the George-Liu algorithm [6] and the original Kaveh-Bondarabady algorithm [11].

Heuristics for profile reductions contribute to provide adequate memory location, and hence, improving cache hit rates [2, 10]. We plan also to apply the algorithms evaluated in this work to perform reordering of vertices and reduce the computational cost of iterative methods for solving linear systems to verify the best method(s) in this context.

DOI: 10.5540/03.2018.006.01.0441

Tabela 1: Results of three algorithms for finding pseudo-peripheral vertices in conjunction with the RCM method [4] applied to reduce *profile* of 50 symmetric instances contained in the Harwell-Boeing sparse matrix collection.

Instance	n	$profile_0$	MKB2	GL	KB2
ash85	85	1153	589	589	589
bcspwr01	39	292	122	122	131
bcspwr02	49	377	214	234	235
bcspwr03	118	1288	746	804	759
bcsstk01	48	851	636	683	634
bcsstk04	132	3631	3637	3717	3965
bcsstk05	153	2449	2279	2313	2246
bcsstk22	138	2124	851	863	870
can_144	144	7355	1071	1074	1092
can_161	161	3378	2610	3079	3740
dwt_234	234	1765	1329	1363	1519
lund_A	147	2870	2303	2303	2303
lund_B	147	2870	2299	2303	2299
nos1	237	780	467	467	696
nos4	100	766	816	755	744
494_bus	494	40975	13272	10566	14792
662_bus	662	45165	30032	32903	26043
685_bus	685	28621	17216	17457	22692
ash292	292	4224	3738	4659	4455
bcspwr04	274	21015	4825	4825	3991
bcspwr05	443	36248	8825	8825	12945
bcsstk06	420	14691	13213	13241	13151
bcsstk19	817	74051	9217	9457	9677
bcsstk20	485	4309	4416	4416	4726
bcsstm07	420	14691	13346	13310	13310
can__292	292	23170	9345	9706	9706
can__445	445	22321	21991	23808	24431
can__715	715	72423	37057	41293	48261
can__838	838	207200	35401	40835	41949
dwt__209	209	9503	3664	3914	3442
dwt__221	221	9910	2011	2011	3380
dwt__245	245	3934	5196	4177	3863
dwt__310	310	2696	2779	2695	2695
dwt__361	361	5084	4714	5139	5030
dwt__419	419	39726	8120	8232	12224
dwt__503	503	35914	15544	15544	13948
dwt__592	592	28805	10949	10983	15755
dwt__878	878	26055	19644	21034	20825
dwt__918	918	108355	21176	24347	33700
dwt__992	992	262306	33994	36296	35112
gr_30_30	900	26970	29352	33872	33117
jagmesh1	936	37240	21817	21817	21817
nos2	957	3180	1907	1907	2904
nos3	960	39101	41272	46168	44954
nos5	468	27286	25832	25381	25381
nos6	675	16229	13784	9305	13560
nos7	729	53144	34110	34110	34110
plat362	362	45261	13389	11018	13462
plskz362	362	43090	6736	4635	6483
sherman1	1000	34740	26109	26109	41753
No. of best results	–	–	**34**	18	17

Acknowledgements

This work was undertaken with the support of the Fapemig - Fundação de Amparo à Pesquisa do Estado de Minas Gerais.

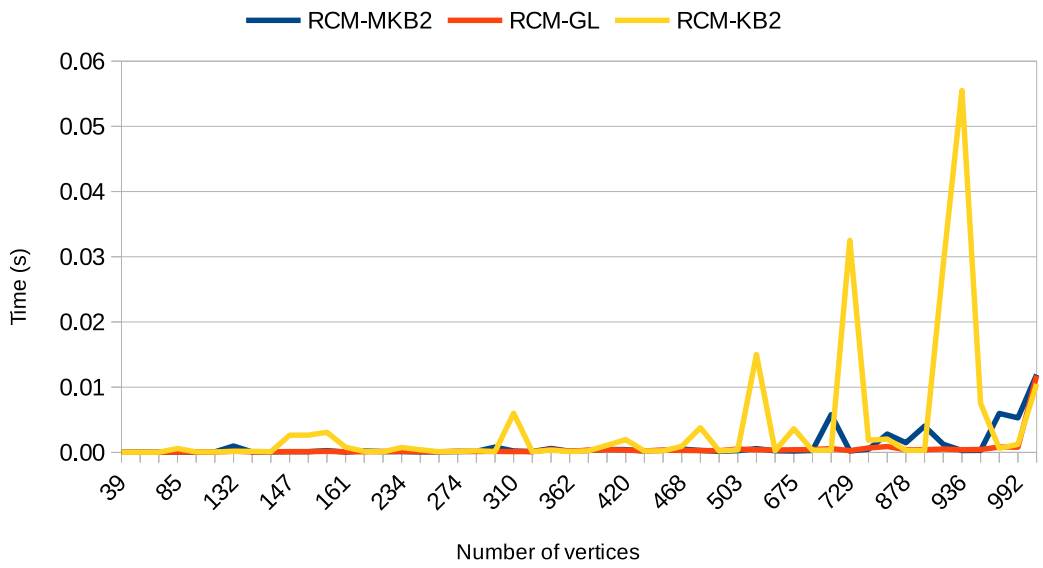

Figura 1: Execution times, in seconds, obtained using three heuristics for profile reductions when applied to 50 instances contained in the Harwell-Boeing sparse matrix collection.

Referências

[1] I. Arany. An efficient algorithm for finding peripheral nodes. In L. Lovász and E. Szemerédi, editors, *Colloquia Mathematica Societatis János Bolyai (Hungarian Edition), Theory of Algorithms Pécs*, volume 44, pages 27–35. North-Holland, Budapest, 1984.

[2] J. A. B. Bernardes and S. L. Gonzaga de Oliveira. A systematic review of heuristics for profile reduction of symmetric matrices. *Procedia Computer Science (Proceedings of the ICCS 2015 - International Conference On Computational Science, Reykjavík, Iceland)*, 51:221–230, 2015.

[3] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15(1):1–14, 1989.

[4] A. George. *Computer implementation of the finite element method.* PhD thesis, Stanford University, Stanford, USA, 1971.

[5] A. George and J. W. Liu. *Computer solution of large sparse positive definite systems.* Prentice-Hall, Englewood Cliffs, 1981.

[6] A. George and J. W. H. Liu. An implementation of a pseudoperipheral node finder. *ACM Transactions on Mathematical Software*, 5(3):284–295, September 1979.

[7] S. L. Gonzaga de Oliveira, A. A. A. M. Abreu, D. T. Robaina, and M. Kischnhevsky. A new heuristic for bandwidth and profile reductions of matrices using a self-organizing

map. In O. Gervasi, B. Murgante, S. Misra, A. M. A. C. V Rocha, C. M. Torre, D. Taniar, B. O. Apduhan, E. Stankova, and S. Wang, editors, *The 16th International Conference on Computational Science and Its Applications (ICCSA), LNCS, Part I, v. 9786*, pages 54–70, Beijing, 2016. Springer.

[8] S. L. Gonzaga de Oliveira, A. A. A. M. Abreu, D. T. Robaina, and M. Kischnhevsky. An evaluation of four reordering algorithms to reduce the computational cost of the jacobi-preconditioned conjugate gradient method using high-precision arithmetic (to appear). *International Journal of Business Intelligence and Data Mining*, 2017.

[9] S. L. Gonzaga de Oliveira, J. A. B. Bernardes, and G. O. Chagas. An evaluation of several heuristics for bandwidth and profile reductions to reduce the computational cost of the preconditioned conjugate gradient method. In *Proceedings of the Brazilian Symposium on Operations Research (SBPO 2016)*, Vitória, Brazil, September 2016. Sobrapo.

[10] S. L. Gonzaga de Oliveira, J. A. B. Bernardes, and G. O. Chagas. An evaluation of low-cost heuristics for matrix bandwidth and profile reductions. *Computational & Applied Mathematics*, 2016, doi=10.1007/s40314-016-0394-9, url=http://link.springer.com/article/10.1007%2Fs40314-016-0394-9.

[11] A. Kaveh and H. A. Rahimi Bondarabady. Ordering for wavefront optimization. *Computer & Structure*, 78:227–235, 2000.

[12] Inc. The MathWorks. MATLAB. http://www.mathworks.com/products/matlab, 1994–2017.

