

A pseudoperipheral vertex finder based on the George–Liu and Kaveh’s B algorithms for small symmetric matrices

Sanderson L. Gonzaga de Oliveira
Universidade Federal de Lavras - UFLA
Lavras - MG
sanderson@dcc.ufla.br

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

ABSTRACT

This work considers the well-known Reverse Cuthill–McKee method for sparse matrix/graph bandwidth reduction. This method needs a starting vertex. There are many algorithms for generating it, and one of the best-known is the George–Liu algorithm. This paper reports a computational experiment that compares eight pseudoperipheral vertex finders along with a modified method that is a hybrid of the George–Liu and Kaveh’s B algorithms. Our simulations then use the resulting vertices as starting points for the Reverse Cuthill–McKee method for bandwidth reduction of sparse matrices. This paper empirically evaluates nine algorithms over a collection of 50 small symmetric matrix instances. This paper shows numerical evidence for the better performance of this variant compared with eight other algorithms for finding a starting vertex for the Reverse Cuthill–McKee method for bandwidth optimization of small matrices with symmetric sparsity patterns at a higher cost than the George–Liu algorithm.

KEYWORDS. Bandwidth reduction, Reverse Cuthill–McKee method, graph labeling.

1. Introduction

Many problems in engineering and science involve 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 x is an unknown vector (which is sought) of length n . Direct or iterative methods are applied to solve sparse linear systems. A low-cost solution using a direct or iterative method demands to order suitably the variables of the sparse linear system. Consequently, the matrix reordering is a preprocessing step for direct and iterative solvers for sparse linear systems to reduce the computational time of the solution. Thus, reducing the bandwidth of sparse matrices can be a relevant task in solving large-scale sparse linear systems [17, 16, 18].

In this context, the cost of the transference of data to and from memory is related to the number of non-null coefficients in the rows of the matrix system A . Cache hits increase if the non-null coefficients in each row lie in the same level of the memory hierarchy. Therefore, cache hits are related to the bandwidth of the matrix system A .

This paper focuses on the problem of efficiently converting a sparse matrix into a band matrix. Such conversion is often the first step of several linear algebraic algorithms that deal with large-scale sparse linear matrices. Specifically, the bandwidth reduction problem is related to agglutinate non-null coefficients around the main diagonal of the matrix A . In particular, we consider a heuristic for reducing the bandwidth of matrices with symmetric sparsity patterns in this computational experiment. The bandwidth minimization problem is NP-hard [25] and, since the 1960s, practitioners have been proposing several heuristics for bandwidth reduction [16, 18, 13, 14].

George and Liu (GL) [10] proposed an algorithm for finding pseudoperipheral vertices for the Reverse Cuthill–McKee (RCM) method [8] and the RCM-GL method [9] still remains in the state of the practice in bandwidth reduction. Thus, the Reverse Cuthill–McKee method [8] with starting vertex given by the George–Liu algorithm [10] is one of the most widely used vertex labeling strategies employed in the literature for reducing the bandwidth size of sparse matrices (e.g. [1, 3, 4, 11, 15, 17, 16, 18, 14, 27, 28, 30]). The RCM-GL method [9] is recognized as one of the most important and amply used bandwidth reduction methods because it shows accurate approximations to the solution at low computational costs. For instance, this method is available on MATLAB [11, 23] and GNU Octave [7] as the function *symrcm* (<https://www.mathworks.com/help/matlab/ref/symrcm.html?requestedDomain=www.mathworks.com>, <https://octave.sourceforge.io/octave/function/symrcm.html>), on Boost C++ Library [2] (http://www.boost.org/doc/libs/1_38_0/libs/graph/doc/cuthill_mckee_ordering.html), and the HSL library [28].

This work proposes a different way of producing the starting vertex for the Reverse Cuthill–McKee method [8] that is a hybrid of the George–Liu and Kaveh’s B [19] algorithms. Thus, this paper is an experimental evaluation based on the Reverse Cuthill–McKee method [8] for matrix bandwidth reductions. Our variant of the George–Liu and Kaveh’s B [19] algorithms is evaluated experimentally against eight other algorithms for finding pseudoperipheral vertices in graphs, including the original George–Liu [8] and Kaveh’s B [19] algorithms, for providing starting vertices for the Reverse Cuthill–McKee method to reduce the bandwidth of symmetric matrices. Thus, this paper evaluates nine algorithms for finding pseudoperipheral vertices for the Reverse Cuthill–McKee method.

The outline of this paper is as follows. Section 2 reviews background information. Section 3 gives a brief description of the Reverse Cuthill–McKee method [8]. Section 4 reviews algorithms for finding pseudoperipheral vertices. Section 5 describes how the experiments were conducted in this study. Section 6 shows the results. Finally, Section 7 provides the conclusions.

2. Definitions and notation

Let $G = (V, E)$ be a connected undirected graph, where V and E are sets of vertices and edges, respectively. The bandwidth of G for a vertex labeling $S = \{s(v_1), s(v_2), \dots, s(v_{|V|})\}$ [i.e., a

bijjective mapping from V to the set $\{1, 2, \dots, |V|\}$ ($s : V \rightarrow \{1, 2, \dots, |V|\}$) 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. Let $A = [a_{ij}]$ be an $n \times n$ symmetric sparse matrix. Equivalently, the bandwidth of row i is defined as $\beta_i(A) = i - \min_{1 \leq j \leq i} [j | a_{ij} \neq 0]$, for $1 \leq i \leq n$ and $a_{ii} \neq 0$, and the overall bandwidth is defined as $\beta(A) = \max_{1 \leq i \leq n} [\beta_i(A)]$.

Several heuristics for bandwidth reductions [16, 18] are based on a data structure known as rooted level structure, where the vertices in a graph representation of a matrix are partitioned into equivalence classes based on their distance from a given root vertex. These classes are called level sets, that is, the vertices are partitioned in relation to the distance from a given starting vertex v . These algorithms are also known as level set reorderings. In particular, $\ell(v) = \max[(u \in V) d(v, u)]$ denotes the eccentricity of the vertex v and the distance $d(v, u)$ is the length of a shortest path connecting vertices v and u . Given a vertex $v \in V$, the level structure $\mathcal{L}(v)$ rooted at vertex v , with depth $\ell(v)$, is the partitioning $\mathcal{L}(v) = \{L_0(v), L_1(v), \dots, L_{\ell(v)}(v)\}$, where $L_0(v) = \{v\}$ and $L_i(v) = \text{Adj}(L_{i-1}(v)) - \bigcup_{j=0}^{i-1} L_j(v)$, for $i = 1, 2, 3, \dots, \ell(v)$, where $\text{Adj}(U) = \{w \in V : (u \in U \subseteq V) (u, w) \in E\}$. The width of a rooted level structure $\mathcal{L}(u)$ is defined as $b(\mathcal{L}(u)) = \max_{0 \leq i \leq \ell(u)} [|L_i(u)|]$.

3. The Reverse Cuthill–McKee method

Algorithm 1 shows a pseudocode of the Reverse Cuthill–McKee method [8], which is a reversed order of the Cuthill–McKee method [5]. The Reverse Cuthill–McKee method [8] uses a level structure rooted at a vertex selected by a pseudoperipheral vertex finder. The Reverse Cuthill–McKee method [8] is a variant of the breadth-first search procedure, where the ordering chosen to reduce the bandwidth of a matrix is obtained by labeling the vertices in the level structure (i.e., in increasing distance order from the starting vertex) in ascending-degree order (see line 6). Then, the algorithm returns the reversed labeling (see line 12 in Algorithm 1).

Algorithm 1: Reverse Cuthill–McKee method [8].

Input: connected graph $G = (V, E)$; starting vertex $v \in V$;
Saída: reordering $S = \{s(1), s(2), \dots, s(|V|)\}$;

```

1 begin
2    $s(1) \leftarrow v$ ;
3    $i \leftarrow 1$ ;
4    $j \leftarrow 1$ ;
5   while ( $i < |V|$ ) do
6     foreach (vertex  $w \in \text{Adj}(G, s(j)) - \{s(1), \dots, s(i)\}$ , in ascending-degree
       order) do
7        $i \leftarrow i + 1$ ;
8        $s(i) \leftarrow w$ ;
9     end foreach
10     $j \leftarrow j + 1$ ;
11  end while
12  return  $S$  in reversed order;
13 end
```

4. A review of algorithms for finding pseudoperipheral vertices

To save space, we give a short description of the algorithms evaluated here. As mentioned, George and Liu [9] proposed an algorithm for the identification of pseudoperipheral vertices for the Reverse Cuthill–McKee method [9]. Algorithm 2 shows the George–Liu algorithm [10].

Algorithm 2: George–Liu algorithm [10].

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

```
1 begin
2    $v \leftarrow \text{ArbitraryVertex}(V)$ ;
   // build a rooted level structure
3    $\mathcal{L}(v) \leftarrow \text{BuscaEmLargura}(v)$ ;
4   repeat
5      $u \leftarrow \text{MinimumDegreeVertex}(L_{\ell(v)}(v))$ ;
     // build a rooted level structure
6      $\mathcal{L}(u) \leftarrow \text{BuscaEmLargura}(u)$ ;
7     if  $\ell(u) > \ell(v)$  then
8        $v \leftarrow u$ ;
9        $\mathcal{L}(v) \leftarrow \mathcal{L}(u)$ ;
10    end if
11  until  $(u \neq v)$ ;
12  return  $v$ ;
13 end
```

Pachl [24] proposed a pseudoperipheral vertex finder based on the George–Liu algorithm [10], which takes $O(|E|\sqrt{|V|})$ time [24]. Algorithm 3 shows Pachl’s algorithm.

Algorithm 3: Pachl’s algorithm.

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

```
1 begin
2    $v_0 \leftarrow \text{ArbitraryVertex}(V)$ ;
   // build a rooted level structure
3    $\mathcal{L}(v_0) \leftarrow \text{BreadthFirstSearch}(v_0)$ ;
   // obtain a vertex from  $L_{\ell(v_0)}(v_0)$ 
4    $v_1 \leftarrow \text{ObtainVertex}(L_{\ell(v_0)}(v_0))$ ;
5    $j \leftarrow 1$ ;
6   repeat
7     // build a rooted level structure
      $\mathcal{L}(v_j) \leftarrow \text{BreadthFirstSearch}(v_j)$ ;
     // obtain a vertex from  $L_{\ell(v_j)}(v_j)$ 
8      $v_{j+1} \leftarrow \text{ObtainVertex}(L_{\ell(v_j)}(v_j))$ ;
9      $j \leftarrow j + 1$ ;
10  until  $(d(v_j, v_{j-1}) = d(v_{j-1}, v_{j-2}) \vee j = |V|)$ ;
11  return  $(v_{j-1})$ ;
12 end
```

Sloan [29] proposed an algorithm based on $\ell(v)$, $b(\mathcal{L}(v))$, and on a shrinking approach recommended by George and Liu that reduces the number of vertices to be searched, following also the idea of the same authors of sorting vertices by their degree. Sloan’s algorithm [29] limits the search to one vertex of each degree. The objective of this algorithm is to return a starting vertex v that presents large $\ell(v)$.

Reid and Scott [26] proposed another pseudoperipheral vertex finder based on the algorithm proposed by Sloan [29]. This algorithm starts from a vertex with a minimum degree. The objective of Sloan's pseudoperipheral vertex finder [29] is to select a pseudoperipheral vertex v with large eccentricity and small $b(\mathcal{L}(v))$. It sorts vertices in $L_{\ell(v)}(v)$ in ascending-degree order. The Reid–Scott algorithm [26] also limits the search in $L_{\ell(v)}(v)$ to one representative of each degree. Additionally, the Reid–Scott algorithm [26] considers up to five vertices in $L_{\ell(v)}(v)$, discarding any that is adjacent to a vertex already checked. Despite the fact that this algorithm is more expensive than the George–Liu algorithm [10], we added it in our experiments.

Kaveh and Bondarabady [21] proposed another algorithm for the identification of pseudoperipheral vertices based on Sloan's algorithm [29]. This computational experiment also evaluates the Kaveh-Bondarabady algorithm and we will refer this algorithm as KB2 for short. Although a recent publication showed that the RCM-GL method [9] dominates the RCM-KB2 method [12], we also added the KB2 algorithm in our experiments.

Kaveh [19] proposed six algorithms for finding pseudoperipheral vertices: the author referred these algorithms as A, B, ..., and F algorithms. Kaveh's B and C algorithms [19] showed the lowest computational costs and the bandwidth results when using Kaveh's B algorithm were better than the results of Kaveh's C algorithm. Thus, our experiments also include Kaveh's B algorithm [19]. Algorithm 4 shows Kaveh's B algorithm. This algorithm is more expensive than the George–Liu algorithm [10] because it verifies every vertex u contained in the last level of the rooted level structure $\mathcal{L}(v)$ of the current vertex v .

Algorithm 4: Kaveh's B algorithm [19].

```

Input: graph  $G = (V, E)$ ;
Output: pseudoperipheral vertex  $v \in V$ ;
1 begin
2    $v \leftarrow \text{ArbitraryVertex}(V)$ ;
   // build the RLS  $\mathcal{L}(v)$ 
3    $\mathcal{L}(v) \leftarrow \text{Breadth-First-Search-variant}(v)$ ;
4   foreach (vertex  $u \in L_{\ell(v)}(v)$ ) do
   // build the RLS  $\mathcal{L}(u)$ 
5      $\mathcal{L}(u) \leftarrow \text{Breadth-First-Search-variant}(u)$ ;
6     if ( $b(\mathcal{L}(u)) < b(\mathcal{L}(v))$ ) then
7        $v \leftarrow u$ ;
8        $\mathcal{L}(v) \leftarrow \mathcal{L}(u)$ ;
9     end if
10  end foreach
11  return  $v$ ;
12 end

```

Kaveh's E algorithm is impractical according to the results presented by Kaveh [19]. Moreover, Kaveh's F algorithm [19] is more expensive than the K91B algorithm [20]. Even though Kaveh's A and D algorithms [19] are more time-consuming than Kaveh's B and C algorithms, we evaluated Kaveh's D algorithm here. Kaveh [20] proposed an algorithm called K91B in our experiments.

Algorithm 5 shows a modified algorithm evaluated in this present computational experiment, named Modified George–Liu and Kaveh's B (MGLKB) algorithm, which is a hybrid algorithm based on the George–Liu algorithm in its first step (see lines 2-10 in Algorithm 5). Subsequently, it executes steps of Kaveh's B algorithm [19] in its second step (see lines 11-17 in Algorithm 5). Although the first step of the MGLKB algorithm follows concepts of the George–Liu

algorithm, it uses no shrinking strategy (in its first loop). The MGLKB algorithm seeks a pseudoperipheral vertex that generates a smaller $b(\mathcal{L}(v))$ in its second loop (i.e., in its second step).

Algorithm 5: MGLKB algorithm.

```

Input: graph  $G = (V, E)$ ;
Output: pseudoperipheral vertex  $v \in V$ ;
1 begin
    // step 1: modified George--Liu algorithm: without
    // shrinking strategy
2  $v \leftarrow \text{ArbitraryVertex}(V)$ ;
    // build the RLS  $\mathcal{L}(v)$ 
3  $\mathcal{L}(v) \leftarrow \text{Breadth-First-Search-variant}(v)$ ;
4 foreach (vertex  $u \in L_{\ell(v)}(v)$ ) do
    // build the RLS  $\mathcal{L}(u)$ 
5  $\mathcal{L}(u) \leftarrow \text{Breadth-First-Search-variant}(u)$ ;
6 if ( $\ell(u) < \ell(v)$ ) then
7      $v \leftarrow u$ ;
8      $\mathcal{L}(v) \leftarrow \mathcal{L}(u)$ ;
9 end if
10 end foreach
    // step 2: steps of Kaveh's B algorithm [19]
11 foreach (vertex  $u \in L_{\ell(v)}(v)$ ) do
    // build the RLS  $\mathcal{L}(u)$ 
12  $\mathcal{L}(u) \leftarrow \text{Breadth-First-Search-variant}(u)$ ;
13 if ( $b(\mathcal{L}(u)) < b(\mathcal{L}(v))$ ) then
14      $v \leftarrow u$ ;
15      $\mathcal{L}(v) \leftarrow \mathcal{L}(u)$ ;
16 end if
17 end foreach
18 return  $v$ ;
19 end

```

5. Description of the tests

Nine algorithms for finding pseudoperipheral vertices were implemented and evaluated in this computational experiment (George–Liu, Pahl’s [24], Sloan’s [29], Kaveh’s B and D [19], K91B [20], Reid-Scott [26], Kaveh–Bondarabady [21], and MGLKB) in conjunction with the Reverse Cuthill–McKee method for bandwidth reduction. Our experiments also include the bandwidth reductions provided by the Reverse Cuthill–McKee method [8] without using an algorithm for the identification of pseudoperipheral vertices. Specifically, the RCM ordering [8] uses the first vertex established in the entry file of the instance as its starting vertex.

This computational experiment uses 50 small symmetric instances contained in the Harwell-Boeing sparse matrix collection [6] to evaluate the bandwidth reductions provided by the algorithms evaluated. This collection is a set of standard test matrices originating from real-world problems arising from a wide variety of scientific and engineering applications.

We implemented the algorithms in the C++ programming language, using the g++ version 4.8.2 compiler. 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). This machine uses a Slackware 14.1 64-bit operating system with Linux kernel-version 3.10.17.

6. Results and analysis

Tables 1–2 show the instance’s name and size (n), the initial bandwidth (β_0) of the instance, and the average bandwidth obtained by each algorithm in 10 serial executions carried out in each matrix applied to a set composed of 50 small symmetric matrices. The symbol “—” indicates that the initial vertex in the entry file is the starting vertex of the Reverse Cuthill–McKee method, that is, there is no search for a pseudoperipheral vertex. Thus, Table 1 shows the results of 10 algorithms applied to reduce the bandwidth of 50 small symmetric instances contained in the Harwell-Boeing sparse matrix collection [6]. The RCM-MGLKB algorithm showed, in general, the best bandwidth results in this dataset: this algorithm obtained the highest number of best bandwidth results in this dataset (in 25 instances).

Table 1: Bandwidth results of several algorithms for finding pseudoperipheral vertices in conjunction with the Reverse Cuthill–McKee method [8] applied to reduce the bandwidth of 15 symmetric instances contained in the Harwell-Boeing sparse matrix collection [6] (continued on Table 2).

instance	n	β_0	MGLKB	Kaveh’s B	KB2	Pachl	Reid–Scott	GL	K91B	Kaveh’s D	—	Sloan
ash85	85	39	12	12	10	10	10	10	14	16	21	10
bcspr01	39	38	5	7	8	7	7	7	7	8	8	8
bcspr02	49	34	13	13	12	13	13	13	13	13	13	13
bcspr03	118	115	18	19	17	21	21	21	21	23	23	23
bcsstk01	48	35	26	26	26	27	26	26	26	26	28	27
bcsstk04	132	47	53	50	54	52	54	54	54	54	54	65
bcsstk05	153	28	24	30	24	30	26	26	26	24	24	24
bcsstk22	138	111	12	12	13	14	14	14	12	14	14	14
can_144	144	142	18	18	18	18	18	18	18	18	18	18
can_161	161	79	18	18	32	18	30	30	30	34	34	30
dwt_234	234	48	17	19	22	19	19	19	19	25	25	25
lund_A	147	23	23	23	23	23	23	23	23	23	23	23
lund_B	147	23	23	23	23	23	23	23	23	23	23	23
nos1	237	4	4	6	6	4	4	4	4	4	4	6
nos4	100	13	12	16	12	12	12	12	12	13	13	15

As expected, Figure 1 shows that the Reverse Cuthill–McKee method was the lowest time-consuming algorithm evaluated. Figures 1–3 show the computational times of 10 methods applied to 50 small symmetric instances contained in the Harwell-Boeing sparse matrix collection [6]. Specifically, Figures 1–3 show that Sloan’s [29], Pachl’s [24], and the George–Liu [10] (Kaveh–Bondarabady [21] and Kaveh’s D [19]) algorithms were the lowest (highest) time-consuming among the algorithms for finding pseudoperipheral vertices evaluated in this computational experiment. Particularly, Figure 3 shows that the behavior of Kaveh’s D [19] algorithm is rather more erratic compared with the other algorithms evaluated here.

The Reid–Scott [26], K91B [20], and Kaveh’s B [19] algorithms were on average half as fast than the George–Liu [10] algorithm. The MGLKB algorithm was on average three times slower than the George–Liu [10] algorithm. These difference in running times can be important when dealing with large-scale instances.

7. Conclusions

This paper evaluated several algorithms for finding pseudoperipheral vertices along with a variation of the well-known George–Liu algorithm. This modified algorithm, named MGLKB algorithm, is also a variant of Kaveh’s B algorithm [19]. The new algorithm is used to provide pseudoperipheral vertices for the Reverse Cuthill–McKee method [8]. We compared its results with the results of

Table 2: Bandwidth results of several algorithms for finding pseudoperipheral vertices in conjunction with the Reverse Cuthill–McKee method [8] applied to reduce the bandwidth of 35 symmetric instances Harwell-Boeing sparse matrix collection [6] (continued from Table 1).

instance	n	β_0	MGLKB	Kaveh's B	KB2	Pachl	Reid–Scott	GL	K91B	Kaveh's D	—	Sloan
494_bus	494	428	67	82	76	62	62	62	62	82	82	84
662_bus	662	335	118	125	86	125	135	135	128	99	102	112
685_bus	685	550	102	75	74	75	79	72	64	93	93	93
ash292	292	24	32	32	30	28	34	34	32	34	34	34
bcsprw04	274	265	49	65	56	65	65	65	65	56	56	56
bcsprw05	443	435	54	59	77	59	59	59	59	74	74	83
bcsstk06	420	47	49	49	48	49	50	50	49	50	50	49
bcsstk19	817	567	20	20	20	21	21	21	21	32	32	22
bcsstk20	485	20	19	19	19	19	19	19	20	19	19	20
bcsstm07	420	47	50	50	50	52	50	50	56	50	50	52
can__292	292	282	65	71	65	75	65	75	76	65	65	64
can__445	445	403	86	85	94	75	92	92	88	111	119	118
can__715	715	611	134	134	129	121	157	157	150	148	202	145
can__838	838	837	167	125	186	153	155	137	172	166	175	184
dwt__209	209	184	33	33	36	37	35	35	33	35	35	42
dwt__221	221	187	17	16	31	16	16	16	16	15	15	30
dwt__245	245	115	55	55	34	43	45	45	53	45	45	40
dwt__310	310	28	14	14	13	13	13	13	14	13	13	13
dwt__361	361	50	15	15	24	25	25	25	15	24	25	25
dwt__419	419	356	34	33	61	34	34	34	35	72	73	69
dwt__503	503	452	57	58	54	57	58	58	62	97	99	64
dwt__592	592	259	41	42	45	41	42	42	42	42	42	42
dwt__878	878	519	34	33	36	37	37	37	38	36	37	37
dwt__918	918	839	50	50	68	63	72	72	50	72	72	72
dwt__992	992	513	55	53	59	63	63	63	65	61	63	63
gr_30_30	900	31	49	48	58	59	59	59	57	58	59	59
jagmesh1	936	778	27	27	27	27	27	27	27	27	27	27
nos2	957	4	4	6	6	4	4	4	4	4	4	6
nos3	960	43	67	65	77	79	79	79	77	79	79	79
nos5	468	178	96	96	88	92	88	88	96	88	88	88
nos6	675	30	16	30	30	16	16	16	29	30	30	30
nos7	729	81	65	65	65	65	65	65	65	65	65	65
plat362	362	249	39	60	52	56	56	56	56	41	41	52
plskz362	362	248	25	37	34	32	24	24	24	25	25	34
sherman1	1000	100	62	57	93	57	57	57	62	57	57	88
Number of best results			25	22	20	19	18	18	16	15	14	10

eight selected algorithms for the identification of pseudoperipheral vertices, in conjunction with the Reverse Cuthill–McKee method [8] to reduce the bandwidth of symmetric matrices.

This paper evaluated 10 algorithms, including the Reverse Cuthill–McKee method [8] without applying an algorithm for finding pseudoperipheral vertices. The MGLKB algorithm obtained the highest number of best bandwidth results (i.e., in 25 instances) in experiments with 50 small symmetric instances contained in the Harwell-Boeing sparse matrix collection [6], in conjunction with the Reverse Cuthill–McKee method for bandwidth reductions. Nevertheless, the MGLKB

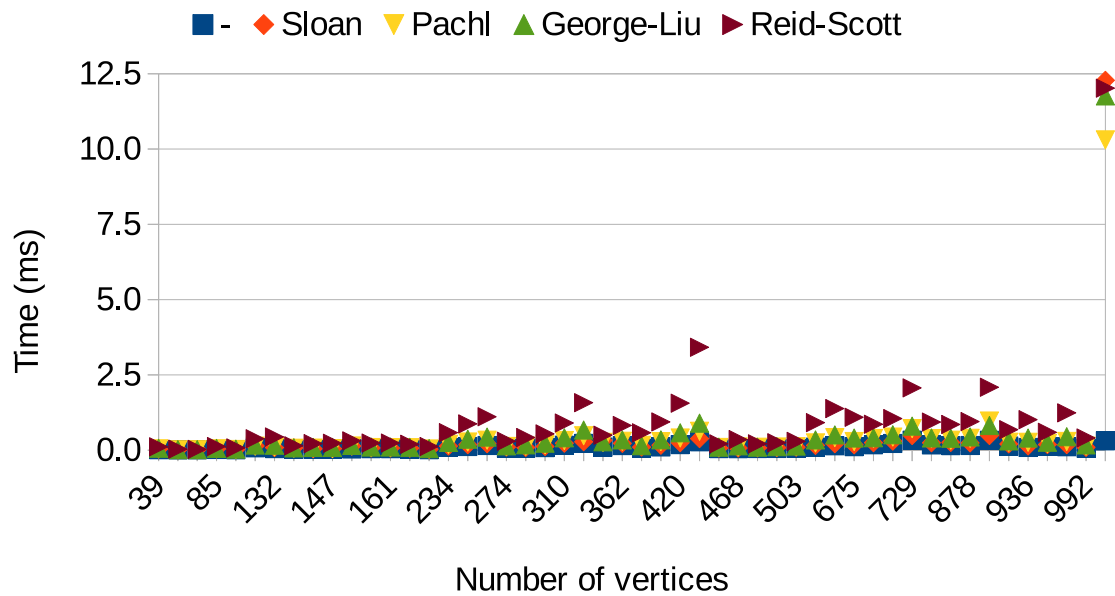


Figure 1: Computational times, in milliseconds, of Sloan’s [29], Pachl’s [24], George–Liu [10], and Reid–Scott [26] algorithms for finding pseudoperipheral vertices in conjunction with the Reverse Cuthill–McKee method [8] applied to 50 symmetric instances contained in the Harwell–Boeing sparse matrix collection [6]. The symbol “-” indicates the execution of the Reverse Cuthill–McKee method [8] without the use of a pseudoperipheral vertex finder.

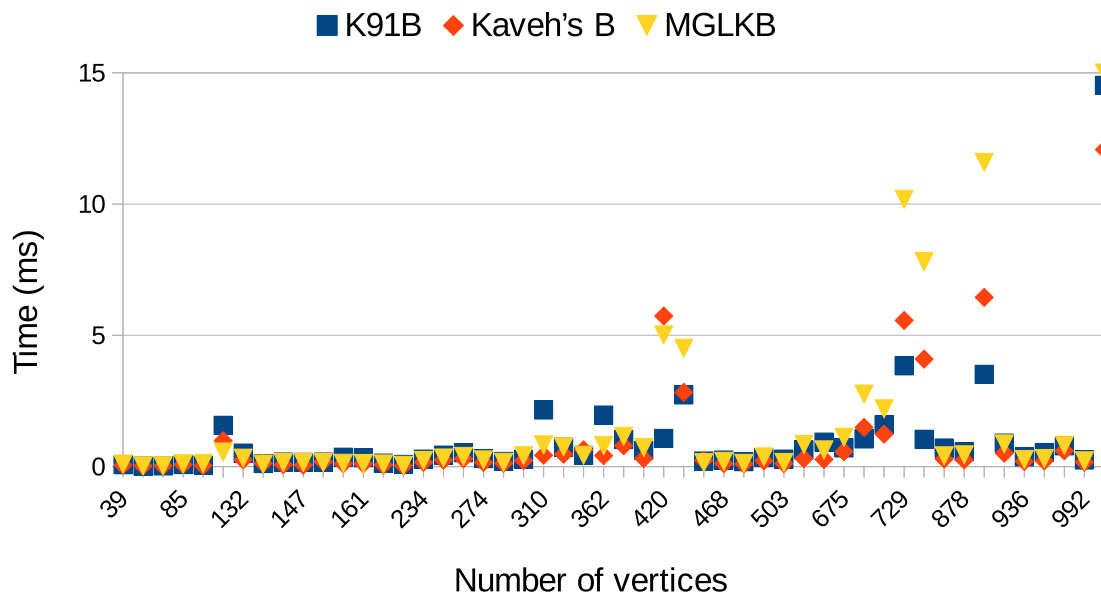


Figure 2: Computational times, in milliseconds the K91B [20], Kaveh’s B [19] and MGLKB algorithms for finding pseudoperipheral vertices in conjunction with the Reverse Cuthill–McKee method [8] applied to 50 symmetric instances contained in the Harwell–Boeing sparse matrix collection [6].

algorithm obtains higher computing times than the George–Liu algorithm [10] (see Figures 1 and 2). Specifically, the George–Liu algorithm [10] reached a similar number of best results than the other

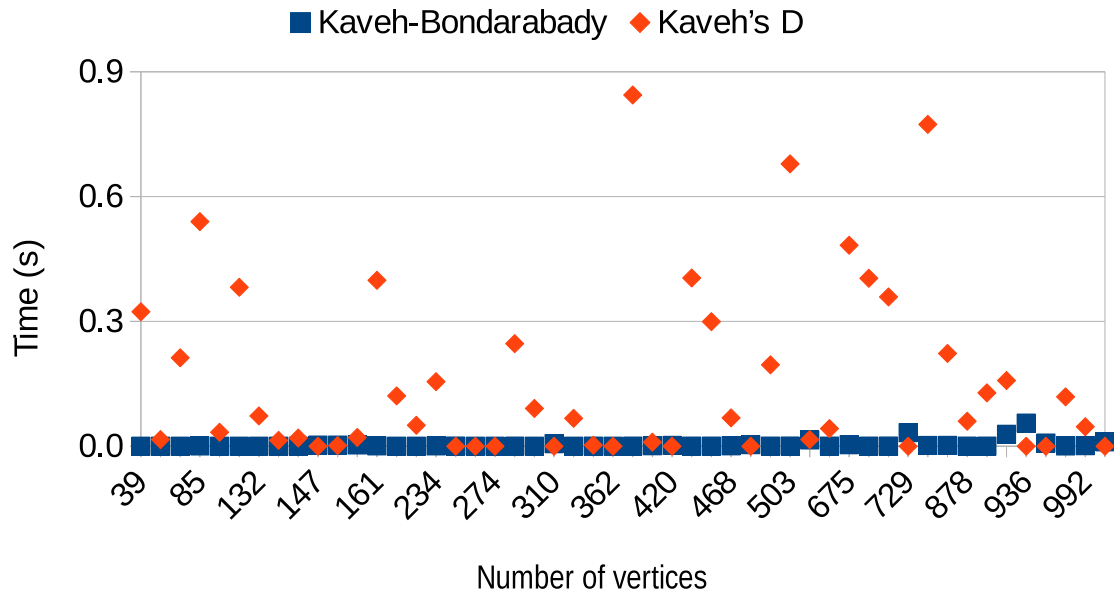


Figure 3: Computational times, in seconds, of the Kaveh–Bondarabady [21] and Kaveh’s D [19] algorithms for finding pseudoperipheral vertices in conjunction with the Reverse Cuthill–McKee method [8] applied to 50 symmetric instances contained in the Harwell–Boeing sparse matrix collection [6].

heuristics (see Table 2) evaluated here at lower processing times. Thus, we recommend the George–Liu algorithm [10] for large-scale matrices. Consistent with the current literature, we conclude that the George–Liu algorithm [10] remains in the state of the practice when applied in conjunction with the Reverse Cuthill–McKee method [8] for large-scale instances.

Heuristics for bandwidth reductions contribute to providing adequate memory location, and consequently, improving cache hit rates. We intend to apply pseudoperipheral vertex finders in conjunction with the Reverse Cuthill–McKee method 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 field. We also plan to evaluate them in conjunction with other methods for bandwidth reductions, including Sloan’s algorithm [29], and Kaveh’s algorithms [19, 20, 21].

Another possible development of this study is to investigate parallel approaches of these algorithms. Specifically, we intend to apply the scheme introduced by Leiserson and Schardl [22].

Acknowledgments

We would like to thank the reviewers for their valuable comments and suggestions.

References

- [1] M. Benzi, D. B. Szyld, and A. Van Duin. Orderings for incomplete factorization preconditioning of nonsymmetric problems. *SIAM Journal on Scientific Computing*, 20(5):1652–1670, 1999.
- [2] Boost. Boost C++ libraries. <http://www.boost.org/>, 2017. Accessed: 2017-06-28.
- [3] J. J. Camata, A. L. Rossa, A. M. P. Valli, L. Catabriga, and G. F. Carey and A. L. G. A. Coutinho. Re-ordering and incomplete preconditioning in serial and parallel adaptive mesh refinement and coarsening flow solutions. *International Journal for Numerical Methods in Fluids*, 69:802–823, 2012.
- [4] G. O. Chagas and S. L. Gonzaga de Oliveira. Metaheuristic-based heuristics for symmetric-matrix bandwidth reduction: a systematic review. *Procedia Computer Science (Proceedings of the*

ICCS - International Conference On Computational Science, Reykjavk, Iceland), 51:211–220, 2015.

- [5] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *ACM Proceedings of the 1969 24th International Conference*, pages 157–172, New York, USA, 1969. ACM.
- [6] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15(1):1–14, 1989.
- [7] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring. GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations, 2015.
- [8] A. George. *Computer implementation of the finite element method*. PhD thesis, Stanford University, Stanford, USA, 1971.
- [9] A. George and J. W. Liu. *Computer solution of large sparse positive definite systems*. Prentice-Hall, Englewood Cliffs, 1981.
- [10] 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.
- [11] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in MATLAB: Design and implementation. *SIAM Journal on Matrix Analysis*, 3(1):333–356, 1992.
- [12] S. L. Gonzaga de Oliveira and A. A. A. M. Abreu. The use of the reverse Cuthill-Mckee method with an alternative pseudo-peripheral vertex finder for profile optimization. In *Proceeding Series of the Brazilian Society of Applied and Computational Mathematics*, v. 6, n. 1 - CNMAC 2017 - XXXVII Congresso Nacional de Matemática Aplicada e Computacional, v. 6, n. 1, pages 010441–1–7, São José dos Campos, 2018. Sociedade Brasileira de Matemática Aplicada e Computacional. DOI: 10.5540/03.2018.006.01.0441.
- [13] S. L. Gonzaga de Oliveira, A. A. A. M. Abreu, D. T. Robaina, and M. Kischnevsky. 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.
- [14] S. L. Gonzaga de Oliveira, A. A. A. M. Abreu, D. T. Robaina, and M. Kischnevsky. 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):190–209, 2017.
- [15] 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 *The XLVIII Brazilian Symposium of Operational Research (SBPO)*, Vitória-ES, Brazil, August 2016. Sociedade Brasileira de Pesquisa Operacional – Sobrapo.
- [16] S. L. Gonzaga de Oliveira, J. A. B. Bernardes, and G. O. Chagas. An evaluation of reordering algorithms to reduce the computational cost of the incomplete Cholesky-conjugate gradient method. *Computational & Applied Mathematics*, 2017, DOI: 10.1007/s40314-017-0490-5.

- [17] 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*, 37(2):1412–1471, 2018. DOI: 10.1007/s40314-016-0394-9.
- [18] S. L. Gonzaga de Oliveira and G. O. Chagas. A systematic review of heuristics for symmetric-matrix bandwidth reduction: methods not based on metaheuristics. In *The XLVII Brazilian Symposium of Operational Research (SBPO)*, Ipojuca-PE, Brazil, August 2015. Sociedade Brasileira de Pesquisa Operacional – Sobrapo.
- [19] A. Kaveh. Ordering for bandwidth reduction. *Computer & Structure*, 24:413–420, 1986.
- [20] A. Kaveh. A connectivity coordinate system for node and element ordering. *Computer & Structure*, 41(6):1217–1223, 1991.
- [21] A. Kaveh and H. A. Rahimi Bondarabady. Ordering for wavefront optimization. *Computer & Structure*, 78:227–235, 2000.
- [22] C. E. Leiserson and Tao B. Schardl. A work-efficient parallel breadth-first search algorithm (or how to cope with the nondeterminism of reducers). In *SPAA '10 – Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*, pages 303–314, New York, NY, 2010. ACM.
- [23] MATLAB. The Mathworks, Inc. <http://www.mathworks.com/products/matlab>, 1994–2017.
- [24] J. K. Pachl. Finding pseudoperipheral nodes in graphs. *Journal of Computer and System Sciences*, 29(1):48–53, 1984.
- [25] C. H. Papadimitriou. The NP-completeness of bandwidth minimization problem. *Computing Journal*, 16:177–192, 1976.
- [26] J. K. Reid and J. A. Scott. Ordering symmetric sparse matrices for small profile and wavefront. *International Journal for Numerical Methods in Engineering*, 45(12):1737–1755, 1999.
- [27] J. A. Ribeiro and S. L. Gonzaga de Oliveira. Uma reviso sistemtica sobre algoritmos paralelos para os problemas de redues de largura de banda e de profile de matrizes. In *The XLIX Brazilian Symposium of Operational Research (SBPO)*, Blumenay-SC, Brazil, August 2017. Sociedade Brasileira de Pesquisa Operacional – Sobrapo.
- [28] Science and Technology Facilities Council. HSL. A collection of Fortran codes for large scale scientific computation. <http://www.hsl.rl.ac.uk/>. Accessed: 2017-06-28.
- [29] S. W. Sloan. A Fortran program for profile and wavefront reduction. *International Journal for Numerical Methods in Engineering*, 28:2651–2679, 1989.
- [30] T. Tsuburaya, Y. Okamoto, and S. Sato. Parallelized ICCG method using block-multicolor orderings in real symmetric linear system derived from voltage-driven FEM in time domain. *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, 34(5):1433–1446, 2015.