COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING, Vol. 10, 913–926 (1994)

A NEW ALGORITHM FOR FINDING A PSEUDOPERIPHERAL VERTEX OR THE ENDPOINTS OF A PSEUDODIAMETER IN A GRAPH

GLAUCIO H. PAULINO
School of Civil and Environmental Engineering, Cornell University, Ithaca, N.Y. 14853, U.S.A.

IVAN F. M. MENEZES
Department of Civil Engineering, PUC-Rio
Rua Marquês de São Vicente, 225, 22453, Rio de Janeiro, R.J., Brazil

MARCELO GATTASS
Department of Computer Science, PUC-Rio
Rua Marquês de São Vicente, 225, 22453, Rio de Janeiro, R.J., Brazil

SUBRATA MUKHERJEE
Department of Theoretical and Applied Mechanics, Cornell University, Ithaca, N.Y. 14853, U.S.A.

SUMMARY

Based on the concept of the Laplacian matrix of a graph, this paper presents the SGPD (spectral graph pseudoperipheral and pseudodiameter) algorithm for finding a pseudoperipheral vertex or the end-points of a pseudodiameter in a graph. This algorithm is compared with the ones by Grimes *et al.* (1990), George and Liu (1979), and Gibbs *et al.* (1976). Numerical results from a collection of benchmark test problems show the effectiveness of the proposed algorithm. Moreover, it is shown that this algorithm can be efficiently used in conjunction with heuristic algorithms for ordering sparse matrix equations. Such heuristic algorithms, of course, must be the ones which use the pseudoperipheral vertex or pseudodiameter concepts

1. INTRODUCTION

After the publication of the landmark paper by Cuthill and McKee (1969),[1] graph theory became a standard approach to reorder sparse matrix equations for reducing bandwidth, profile or wavefront. However, the success of most algorithms depends upon the choice of one or more starting vertices§. Peripheral vertices (i.e. vertices for which the eccentricity is equal to the diameter of the graph) have been shown[2,3] to be good starting vertices for reordering algorithms. Since the location of peripheral vertices in graphs is computationally expensive,[4,5] most reordering algorithms use pseudoperipheral vertices (PVs) instead, i.e. vertices with the highest possible eccentricity. Examples of such algorithms are Reverse Cuthill–McKee (RCM),[3] Gibbs–Poole–Stockmeyer (GPS),[2] Gibbs–King (GK), [6,7] Snay,[8] Sloan,[9] Medeiros

§ Recently, Paulino *et al.*[25,26] have proposed a new class of spectral-based reordering algorithms, which do not depend on the choice of a starting vertex. After having finished the manuscript, the authors became aware of similar independent work (on spectral envelope reduction) by Barnard *et al.*[44] Their report was completed in October 1993 while our manuscript was submitted to the *Int. j. numer. methods eng.* in March 1993.

CCC 0748–8025/94/110913–14

Received August 1993
Revised April 1994

et al.,[10] Fenves and Law,[11] Hoit and Wilson,[12] and Livesley and Sabin.[13] Moreover, several papers have been published about algorithms for finding one or more starting vertices for renumbering the vertices of a graph, e.g. Cheng (1973),[14] Gibbs *et al.* (1976),[2] George and Liu (1979),[15] Pachl (1984),[16] Smyth (1985),[17] Kaveh (1990),[18] and Grimes *et al.*[19]

Other reordering algorithms that use the pseudoperipheral vertex concept are the refined quotient tree, the one-way dissection and the nested dissection in the sparse matrix package SPARSPAK.[3] Note, however, that the use of the pseudoperipheral vertex concept is not restricted to reordering sparse matrix equations. This concept has applications in various fields such as geography,[20] and mapping of finite element graphs onto processor meshes, (see Reference 21, page 1417).

This paper presents the SGPD (spectral graph pseudoperipheral and pseudodiameter) algorithm for finding a pseudoperipheral vertex or the end-points of a pseudodiameter in a graph. A pseudoperipheral vertex is an approximately peripheral vertex, i.e. an heuristic approximation to a peripheral vertex. Similarly, pseudodiametrical vertices are approximately diametrical vertices, i.e. heuristic approximations to the end-points of a diameter. Note that pseudodiametrical vertices are pseudoperipheral ones, but two pseudoperipheral vertices are not necessarily pseudodiametrical ones. This distinction is made here because some reordering algorithms use one pseudoperipheral vertex (e.g. RCM[3]), others use the end-points of a pseudodiameter (e.g. GPS,[2] GK,[6] Sloan,[9] and Medeiros *et al.*[10]), while still others use more than two pseudoperipheral vertices (e.g. Hoit and Wilson,[12] and Snay[8]).

The remainder of this paper is organized as follows. Section 2 provides graph theoretical definitions, notations and a brief discussion about spectral techniques applied to graphs. Section 3 presents the SGPD algorithm. Section 4 outlines the main numerical aspects for the implementation of this algorithm. Section 5 presents some numerical examples using Everstine's[22] collection of benchmark test problems. These examples include eccentricity verification and use of the SGPD in conjunction with some widely used graph reordering algorithms. The coupling of the SGPD with existing reordering algorithms yields new versions of these algorithms. The results obtained show the effectiveness of the proposed SGPD algorithm. Section 6 presents some considerations about computational efficiency. Finally, Section 7 concludes this work.

2. GRAPHS: DEFINITIONS, NOTATIONS AND SPECTRAL TECHNIQUES

The basic graph-theoretical background to this paper can be found in the excellent books by Harary*,[23] and Cvetković *et al.*[24] Further details about spectral techniques applied to graphs, and their association with the finite element method, can be found in References 25 and 26.

Let $G = (V, E)$ be an *undirected* and *connected* graph. $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices with $|V| = n$; $E = \{e_1, e_1, \ldots, e_m\}$ is a set of edges with $|E| = m$, where $|\cdot|$ denotes the cardinality of the set. Edges are unordered pairs of distinct vertices of V. A labelling of G is a function $f: V \rightarrow D$, where D is a collection of domain labels. Here, $D = \{1, 2, \ldots, |V|\}$ is used. The vertices may also be referred by their labels in the labelled graph.

Two vertices v_i and v_i in G are adjacent if $\{v_i, v_j\} \in E$.

If $W \subset V$, the *adjacent* set of W, Adj(W), is

$$\mathrm{Adj}(W) = \{v_i \in (V - W) \mid \{v_i, v_j\} \in E, v_j \in W, \quad i \neq j\}$$

* The more recent book by Buckley and Harary,[45] which is based on the classic book by Harary,[23] is also a good alternative reference in the field of graph theory.

If $W = \{v\}$, where v is a single vertex, the adjacent set of W is denoted by $\mathrm{Adj}(v)$ instead of $\mathrm{Adj}(\{v\})$.

A section-graph $G(W, E(W))$ of $G(V, E)$ is a graph for which $W \subset V$ and

$$E(W) = \{\{v_i, v_j\} \in E \mid v_i \in W, v_j \in W\}$$

A clique is a section graph whose vertices are pair-wise-adjacent.

The degree of the set W is defined as $\deg(W) = |\mathrm{Adj}(W)|$. Again, if W is a single vertex, the degree of W is denoted by $\deg(v)$ instead of $\deg(\{v\})$.

A path in a graph is an ordered set of vertices $(u_1, u_2, \ldots, u_{p+1})$ such that u_k and u_{k+1} are adjacent vertices for $k = 1, 2, \ldots, p$. This path has length p, and it goes from u_1 to u_{p+1}, which are the endpoints of the path.

The distance $d(v_i, v_j)$ between two vertices in G is the length of the shortest path between them, i.e. $d(v_i, v_j) = \min |$ path between v_i and $v_j|$.

The eccentricity $e(v)$ of a given vertex v in G is the largest distance between v and any other vertex of G, i.e.

$$e(v) = \max\{d(v, v_i) \mid v_i \in V\}$$

The diameter $\delta(G)$ is the largest eccentricity of any vertex in the graph, i.e.

$$\delta(G) = \max\{e(v_i) \mid v_i \in V\}$$

A peripheral vertex v is the one for which its eccentricity is equal to the diameter of the graph, i.e. $e(v) = \delta(G)$.

For a given vertex $r \in V$, the rooted level structure[27,28] (this is a crucial concept to many reordering algorithms) is the partitioning

$$L(r) = \{\mathbf{L}_0(r), \mathbf{L}_1(r), \ldots, \mathbf{L}_{e(r)}(r)\}$$

such that:

$$\mathbf{L}_0(r) = \{r\}$$
$$\mathbf{L}_1(r) = \mathrm{Adj}(\mathbf{L}_0(r))$$
$$\mathbf{L}_i(r) = \mathrm{Adj}(\mathbf{L}_{i-1}(r) - \mathbf{L}_{i-2}(r)),\ i = 2, \ldots, e(r)$$

Note that $\cup_{k=0}^{e(r)} \mathbf{L}_k(r) = V$.

The length of $L(r)$ is $e(r)$, and the width of $L(r)$ is

$$w(r) = \max\{|\mathbf{L}_i(r)|,\ 0 \leqslant i \leqslant e(r)\}$$

The association of graphs with matrices is of special importance in this paper. The adjacency (**A**), degree (**D**) and Laplacian (**L**) matrices are defined next.

The adjacency matrix $\mathbf{A}(G) = [a_{ij}]$ of a labelled graph G is defined as:

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

The degree matrix $\mathbf{D}(G) = [d_{ij}]$ is the diagonal matrix of vertex degrees:

$$d_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

The Laplacian matrix $\mathbf{L}(G) = [l_{ij}]$ is defined as:

$$\mathbf{L}(G) = \mathbf{D}(G) - \mathbf{A}(G) \tag{1}$$

or, in component form, $\mathbf{L}(G)$ is given by

$$l_{ij} = \begin{cases} -1 & \text{if } \{v_i, v_j\} \in E \\ \deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

According to Anderson and Morley,[29] the name 'Laplacian matrix' comes from a discrete analogy with the Laplacian operator in numerical analysis (for further explanation, see Paulino *et al.*[25]). The Laplacian matrix is symmetric, singular (each row or column sum up to zero), and positive-semi-definite.[29,30] It is employed here for the study of spectral properties of a graph.

Let the eigenvalues of $\mathbf{L}(G)$ be arranged in ascending order of their values:

$$0 = \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n \leqslant |V|$$

For the first eigenpair, $(\lambda_1, \mathbf{y}_1) = (0, \mathbf{1})$, where $\mathbf{1}$ is a unit vector, and the eigenvector $\mathbf{y}_1$ has been normalized. The special properties of the second eigenpair $(\lambda_2, \mathbf{y}_2)$ of $\mathbf{L}(G)$ have been studied by Fiedler.[30,31] He designates λ_2 as the *algebraic connectivity* of the graph G, which is related to the usual vertex and edge connectivities of G. If the graph has a simple pattern, analytical solutions are available for λ_2.[29,30] The components of $\mathbf{y}_2$ can be assigned to the vertices of G and can be considered as weights for them. Fiedler designates this weighting process as the *characteristic valuation* of G. It is determined uniquely up to a non-zero factor if λ_2 is a simple eigenvalue of $\mathbf{L}(G)$ (i.e. with multiplicity = 1).

3. THE SPECTRAL GRAPH PESUDOPERIPHERAL AND PSEUDODIAMETER (SGPD) ALGORITHM

The automatic algorithm SGPD is presented in Table I.

A similar algorithm for finding a pseudoperipheral vertex in a graph has been proposed by Grimes *et al.*[19] and Kaveh.[18] However, they have used a modified adjacency matrix $\mathbf{B}$ instead of the Laplacian matrix $\mathbf{L}$. The matrix $\mathbf{B}$ is defined as

$$\mathbf{B}(G) = \mathbf{I}(G) + \mathbf{A}(G) \tag{2}$$

where $\mathbf{I}$ is the identity matrix (compare equation (2) with equation (1))†. Note that the eigenvalues of $\mathbf{B}$ are the ones for $\mathbf{A}$ shifted by unity, and the normalized eigenvectors of $\mathbf{A}$ and $\mathbf{B}$ are the same. Grimes *et al.*[19] and Kaveh[18] have used the vertex corresponding to the smallest component in the dominant eigenvector of $\mathbf{B}(G)$ as a pseudoperipheral vertex. Straffin[20] has used the vertex corresponding to the largest component in the dominant eigenvector of $\mathbf{B}(G)$

Table I. Spectral graph pseudoperipheral and pseudodiameter (SGPD) algorithm

1. Find the eigenvector $\mathbf{y}_2$ of the Laplacian matrix $\mathbf{L}(G)$.
2. The vertex corresponding to the smallest (or largest) component in $\mathbf{y}_2$ is a pseudoperipheral vertex. The vertices corresponding to the smallest and largest components in $\mathbf{y}_2$ are the endpoints of a pseudodiameter.

† Booth and Lipton[5] call the above matrix $\mathbf{B}$ the 'augmented adjacency matrix'. In their work, they also justify the use of this matrix instead of the standard adjacency matrix $\mathbf{A}$.

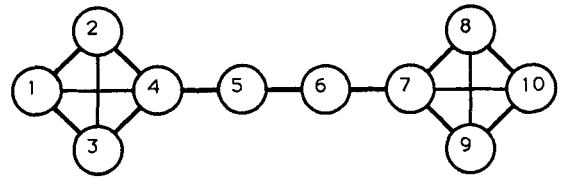

Figure 1. Grimes *et al.*[19] counterexample

to determine the most accessible vertex in a graph network for an interesting application in geography. Note that the concept of *most accessible vertex* is opposed to the concept of *pseudoperipheral vertex.*

Figure 1 shows the counterexample presented by Grimes *et al.*[19] Their algorithm fails for this problem, while the SGPD furnishes the optimal solution. The vertices in the two cliques at the left- (v_1, v_2, v_3) and right- (v_8, v_9, v_{10}) hand sides of Figure 1 are peripheral. The dominant eigenvector $\mathbf{y}_{|V|}$ ($|V| = 10$) of $\mathbf{B}$ corresponding to the largest eigenvalue ($\lambda_{|V|} = 4{\cdot}1284$) is

$$\mathbf{y}^{\mathrm{T}}_{|V|}(\mathbf{B}) = \lfloor 0{\cdot}1073,\ 0{\cdot}1073,\ 0{\cdot}1073,\ 0{\cdot}1211,\ 0{\cdot}0569,\ 0{\cdot}0569,\ 0{\cdot}1211,\ 0{\cdot}1073,\ 0{\cdot}1073,\ 0{\cdot}1073 \rfloor \quad (3)$$

The smallest components of $\mathbf{y}_{|V|}(\mathbf{B})$ correspond to the interior vertices v_5 and v_6, which are inconsistent with the objective of the algorithm of finding peripheral or nearly peripheral vertices. With respect to Figure 1 and the SGPD algorithm of Table I, the eigenvector $\mathbf{y}_2$ of $\mathbf{L}$ corresponding to the algebraic connectivity of the graph (i.e. the second smallest eigenvalue: $\lambda_2 = 0{\cdot}1442$) is

$$\mathbf{y}^{\mathrm{T}}_2(\mathbf{L}) = \lfloor 1{\cdot}0000,\ 1{\cdot}0000,\ 1{\cdot}0000,\ 0{\cdot}8558,\ 0{\cdot}2997,\ -0{\cdot}2997,\ -0{\cdot}8558,\ -1{\cdot}000,\ -1{\cdot}0000,\ -1{\cdot}0000 \rfloor \quad (4)$$

The smallest components of $\mathbf{y}_2(\mathbf{L})$ correspond to the clique (v_8, v_9, v_{10}) at the right-hand side of Figure 1, and the largest components of $\mathbf{y}_2(\mathbf{L})$ correspond to the clique (v_1, v_2, v_3) at the left-hand side of Figure 1. In this case, the SGPD captures the essential structure of the graph and provides a peripheral vertex (e.g. v_8), or the end-points of a diameter (e.g. v_8 and v_1).

For this specific example, it is interesting to relate Figure 1 and the eigenvectors in expressions (3) and (4). Note that the components of $\mathbf{y}_{|V|}(\mathbf{B})$ (equation 3) are symmetric, while the components of $\mathbf{y}_2(\mathbf{L})$ (equation (4)) are skewsymmetric. For the SGPD algorithm (Table I), this last property is essential for obtaining the endpoints of a pseudodiameter (in this case, the actual diameter) of the graph in Figure 1 (it is worth mentioning that this last property is also related to Theorem 2.1 of Reference 32, p. 432).

3. SOME NUMERICAL ASPECTS

The numerical procedure which implements the SGPD algorithm (see Table I) should be able to handle large and generic graphs. Therefore, the main task in the SGPD is the solution of a large eigenproblem. The goal is the determination of the second eigenpair $(\lambda_2, \mathbf{y}_2)$ of the Laplacian matrix $\mathbf{L}$. Here, the eigensolution is accomplished by a special version of the Subspace Iteration method, as reported by Paulino *et al.*[26] However, any other equivalent method can be used for the eigensolution, e.g. the Lanczos methods.[33,34] A brief description of the Subspace Iteration method, as implemented in this work, is presented next.

Since the interest here is in the eigenvector associated to the second eigenvalue of the Laplacian matrix **L**, the dimension suggested for the reduced subspace in the case of a connected graph is $q = 4$.[26] However, if necessary, the dimension of the reduced space (q) can be changed. So, the reduced eigenproblem is solved by QR iterations applied to matrices of order q. To solve the problem of singularity in the computation of the eigenvector corresponding to the null eigenvalue, the Laplacian matrix has been shifted according to

$$\mathbf{L} \leftarrow \mathbf{L} + \alpha \mathbf{I}$$

where α is a shifting constant. Here, $\alpha = 1$ has been adopted, such that all the eigenvalues become positive ($\lambda_j \geqslant 1{\cdot}0$; $1 \leqslant j \leqslant |\mathrm{V}|$). This procedure does not change the normalized eigenvectors of the Laplacian matrix.

The convergence criterion is defined in terms of the relative error between successive eigenvalue approximations:

$$\frac{|\lambda_j^{i+1} - \lambda_j^i|}{\lambda_j^{i+1}} \leqslant \mathrm{TOL}, \qquad 1 \leqslant j \leqslant q \tag{5}$$

where the subscript denotes the jth eigenvalue, the superscripts denote the iteration numbers, and TOL is a specified tolerance for both the Subspace iterations and the QR iterations in the reduced space.[25,26] Each eigenvector approximation is normalized with respect to the absolute value of its largest component. Here, the number of iterations for both the Subspace and the QR methods are unlimited in a numerical sense, i.e. the maximum number of iterations has been chosen to be a very large number.

5. EXAMPLES

Two types of numerical examples are presented next. Firstly, the eccentricity of the vertices obtained by the SGPD algorithm (Table I) is compared with the results of other heuristic algorithms and the actual diameter of the graph. Secondly, the vertices obtained by the SGPD are used as starting vertices of some widely used algorithms for bandwidth, profile and wavefront reduction. For the solution of the eigenproblem in the SGPD algorithm, $\mathrm{TOL} = 10^{-6}$ (Equation (5)) has been adopted.

All the examples that follow come from the collection of benchmark test problems provided by Everstine.[22] The matrices range in order from 59 to 2680. Larger test problems, e.g. with matrices of order around 40,000 and 1,000,000 entries, can be found in the Harwell–Boeing sparse matrix collection.[35] In fact, Everstine's test problems are also included in this collection.

Everstine's[22] collection of examples contains a set of 30 diversified problems representing finite element meshes, which have been widely used to test reordering algorithms.[9,22,26,36] A description of the test problems, and plots of the corresponding meshes, can be found in Everstine's paper.[22] Here, the primary concern is connected graphs. Therefore, the SGPD algorithm will be tested using the test problems for which the associated nodal graph G is connected, i.e. 24 examples of Everstine's collection. The other six examples are associated to non-connected graphs ($\lambda_2 = 0$) and are not treated here. Techniques for treating non-connected graphs, in the context of spectral methods, have been presented by Paulino *et al.*[25,26]

5.1. Eccentricity verification

Table II lists some initial data about Everstine's[22] test problems and the results obtained by George and Liu[15] (here designated G&L), Gibbs *et al.*[2] (here designated GPSD in order to

Table II. Eccentricity comparison among G&L, GPSD and SGPD algorithms

1	2	3	4	5	6	7	8	9	10	11	12	13	14
Graph			G&L			GPSD				SGPD			
$\|V\|$	$\|E\|$	λ_2	$\delta(G)$	PV	$e(PV)$	PV_1	$e(PV_1)$	PV_2	$e(PV_2)$	PV_1	$e(PV_1)$	PV_2	$e(PV_2)$
59	104	0·0666	13	6	13	43	13	5	13	1	13	6	13
66	127	0·0115	32	21	32	1	32	21	32	1	32	21	32
72	75	0·0215	21	72	21	1	21	72	21	57	19	1	21
87	227	0·0969	13	31	13	31	13	59	13	31	13	59	13
162	510	0·0577	22	135	22	158	22	135	22	122	21	158	22
193	1650	0·8147	7	47	7	1	7	186	7	193	7	162	7
209	767	0·1211	12	32	12	118	12	15	12	14	12	15	12
221	704	0·0256	27	154	27	154	27	131	27	155	27	198	27
245	608	0·0354	17	146	17	146	17	2	17	2	17	146	17
307	1108	0·1605	14	5	13	286	13	15	13	126	13	79	13
310	1069	0·0164	39	304	39	1	39	304	39	1	39	310	39
361	1296	0·0358	30	313	30	1	30	313	30	1	30	313	30
419	1572	0·0362	22	312	22	390	22	307	22	386	22	311	22
503	2762	0·1001	23	12	23	12	23	393	23	393	23	12	23
592	2256	0·0201	37	263	37	185	37	263	37	185	37	263	37
758	2618	0·0025	105	233	105	756	105	233	105	16	105	758	105
869	3208	0·0078	57	581	57	1	57	691	57	1	57	581	57
878	3285	0·0147	40	841	40	1	40	849	40	841	40	1	40
918	3233	0·0085	42	834	42	5	42	702	42	704	41	5	42
992	7876	0·0590	30	481	30	1	30	487	30	481	30	16	30
1005	3808	0·0304	34	713	34	713	34	619	34	525	33	697	33
1007	3784	0·0104	47	994	47	26	47	991	47	994	47	1	47
1242	4592	0·0159	31	259	31	1	31	259	31	259	31	8	31
2680	11173	0·0046	75	243	75	243	75	2470	75	2489	73	242	74

differentiate it from the GPS reordering algorithm), and the proposed SGPD (Table I) algorithm. The initial data are $|V|$, $|E|$, algebraic connectivities (λ_2), and the diameters of the associated graphs ($\delta(G)$). The results are the pseudoperipheral vertices (PV_1) and respective eccentricities obtained by G&L, and the pseudodiametrical vertices (PV_1 and PV_2) and respective eccentricities obtained by GPSD and SGPD.

From Table II, one observes that in some cases the pseudodiametrical vertices obtained by the GPSD and SGPD algorithms coincide, e.g. $|V| = 66$, 87, 245, 361, 503 and 592. In many cases, there are common pseudoperipheral vertices for G&L, GPSD and/or SGPD algorithms, e.g. $|V| = 59$, 66, 72, 87, 162, 209, etc.

Moreover, Table II also shows that the eccentricities obtained by the SGPD are comparable to those of G&L and GPSD algorithms. In most cases, the eccentricities obtained by the G&L, GPSD and SGPD algorithms are equal to the diameter of the graph. In 18 occasions, the SGPD gives $e(PV_1) = \delta(G)$; for the other six occasions, $e(PV_1)$ is very close to $\delta(G)$ – the maximum difference between these quantities is 2. In 21 occasions, the SGPD gives $e(PV_2) = \delta(G)$; for the other three occasions, the difference between $\delta(G)$ and $e(PV_2)$ is 1. The pseudodiametrical vertices of the GPSD algorithm satisfy the condition $e(PV_1) = e(PV_2)$. In the case of the SGPD, this condition is satisfied in 20 occasions; for the other four occasions, the difference between $e(PV_2)$ and $e(PV_1)$ is 2 in one occasion, and 1 in the other three occasions.

5.2 *Application to bandwidth, profile and wavefront reduction algorithms*

The definitions used here for matrix bandwidth B, profile P and root mean square (r.m.s.) wavefront $\hat{W}$ are the same as those provided by Everstine[22] or Paulino *et al.*[26] In this section, the vertices obtained by the SGPD algorithm are used as trial starting vertices for the RCM (Table III), GPS (Table IV) and GK (Table V) algorithms.

Table III. Reverse Cuthill–McKee (RCM) algorithm[1,3]

1. Find a pseudoperipheral vertex.
2. Renumber this vertex as 1.
3. For $i = 1, \ldots, |V|$ find all the unnumbered vertices in $\mathrm{Adj}(v_i)$ and label them in increasing order of degree.
4. For $i = 1, \ldots, |V|$ revert the numbering by setting (i) to $(n - i + 1)$.

Table IV. Gibbs–Poole–Stockmeyer (GPS) algorithm[2,37]

1. Find endpoints of a pseudodiameter.
2. Minimize level width.
3. Number the graph in a manner similar to the RCM algorithm.

Table V. Gibbs–King (GK) algorithm[2,6]

1. Find endpoints of a pseudodiameter.
2. Minimize level width.
3. Number the graph level by level in a manner analogous to King's criteria.[7]

The original version of the RCM algorithm uses the G&L[15] pseudoperipheral vertex. The original version of the GPS and GK algorithms uses the GPSD[2] pseudodiametrical vertices. In the examples that follow, the original version of Step 1 of the RCM, GPS and GK algorithms (Tables III, IV and V, respectively) is replaced by the SGPD algorithm. Note that coupling of the SGPD with the RCM, GPS and GK algorithms yields new versions of these reordering algorithms, which are designated here as RCM(SGPD), GPS(SGPD) and GK(SGPD), respectively.

Table VI lists $|V|$, the initial values of B, P and $\hat{W}$, the results for B, P and $\hat{W}$ using the original RCM, and the results for B, P and $\hat{W}$ using the RCM with the SGPD algorithm. These last results are obtained as the ones which give the smallest profiles when PV_1 and PV_2 (see 11th and 13th columns of Table II) are used as starting vertices. The strategy of using two trial starting vertices is efficient because the simple and easily accessed data structure of the RCM (see Table III) makes it an extremely fast reordering algorithm.[3,19,26] For the RCM algorithm, Table IV shows that, on the average, the SGPD is more effective than the G&L algorithm.

Table VII lists $|V|$ and the normalized values of B, P and $\hat{W}$ for the RCM, GPS and GK algorithms. The relative values of B, P and $\hat{W}$ are the ratio of the results obtained with the reordering algorithm using the SGPD starting vertices and the ones obtained with the original reordering algorithm. The relative values of B, P and $\hat{W}$ for the RCM algorithm can be

Table VI. Bandwidth, profile and wavefront reduction using the RCM and the RCM(SGPD) algorithms

1	2	3	4	5	6	7	8	9	10
Problem	Initial			RCM			RCM(SGPD)		
$\|V\|$	B	P	$\hat{W}$	B	P	$\hat{W}$	B	P	$\hat{W}$
59	26	464	8·219	9	315	5·469	9	315	5·469
66	45	640	11·008	4	223	3·435	4	223	3·435
72	13	244	3·460	8	356	5·231	9	382	5·627
87	64	2336	29·378	19	710	8·669	22	699	8·561
162	157	2806	18·955	17	1641	10·532	18	1612	10·378
193	63	7953	43·841	50	5153	28·097	57	4974	26·881
209	185	9712	50·322	34	3804	19·185	34	3812	19·216
221	188	10131	50·393	20	2367	11·349	16	2164	10·198
245	116	4179	18·481	58	5587	25·901	46	4472	19·815
307	64	8132	27·360	45	8651	29·668	42	8115	27·339
310	29	3006	9·852	16	3141	10·363	15	3035	9·958
361	51	5445	15·379	16	5075	14·277	16	5075	14·277
419	357	40145	107·072	34	8609	21·211	39	8474	21·058
503	453	36417	78·603	60	14906	31·759	60	14906	31·759
592	260	29397	55·179	43	11452	20·657	43	11452	20·657
758	201	23871	37·946	29	8718	13·018	29	8581	12·818
869	587	20397	25·019	44	19259	24·169	42	16949	21·650
878	520	26933	31·921	47	22385	26·610	38	22007	25·923
918	840	109273	131·142	58	23096	26·743	62	22984	26·998
992	514	263298	301·994	66	38128	40·125	64	37288	39·088
1005	852	122075	137·660	105	43106	48·902	103	42398	47·066
1007	987	26793	26·925	39	24703	25·427	39	24703	25·427
1242	937	111430	105·201	93	50241	42·425	93	50241	42·425
2680	2500	590543	234·418	70	105798	40·632	70	106261	40·739

Table VII. Relative bandwidth, profile and wavefront reduction using the RCM, GPS and GK algorithms

1	2	3	4	5	6	7	8	9	10
	RCM(SGPD)			GPS(SGPD)			GK(SGPD)		
	RCM			GPS			GK		
Problem |V|	B	P	$\hat{W}$	B	P	$\hat{W}$	B	P	$\hat{W}$
59	1·000	1·000	1·000	0·889	0·909	0·894	0·833	0·917	0·901
66	1·000	1·000	1·000	1·000	1·000	1·000	1·000	1·000	1·000
72	1·125	1·073	1·076	1·286	1·127	1·151	1·000	1·217	1·247
87	1·158	0·984	0·987	0·850	0·952	0·942	0·810	0·829	0·809
162	1·059	0·982	0·985	1·071	0·993	1·002	1·000	1·004	1·012
193	1·140	0·965	0·957	1·302	1·023	1·027	1·196	1·060	1·065
209	1·000	1·002	1·002	0·767	0·772	0·752	0·939	0·767	0·757
221	0·800	0·914	0·899	0·895	0·964	0·959	0·857	0·943	0·938
245	0·793	0·800	0·765	1·000	1·000	1·000	1·000	1·000	1·000
307	0·933	0·938	0·921	1·023	0·966	0·947	1·125	0·946	0·935
310	0·937	0·966	0·961	0·933	0·991	0·990	0·727	0·993	0·993
361	1·000	1·000	1·000	1·000	1·000	1·000	1·000	1·000	1·000
419	1·147	0·984	0·993	1·206	0·981	0·992	1·143	1·002	1·012
503	1·000	1·000	1·000	1·000	1·000	1·000	0·986	0·971	0·966
592	1·000	1·000	1·000	1·000	0·997	0·993	1·000	1·000	1·000
758	1·000	0·984	0·985	1·000	0·999	0·999	1·000	1·000	1·000
869	0·954	0·880	0·896	1·000	1·000	1·000	1·000	1·000	1·000
878	0·808	0·983	0·974	1·000	0·994	0·994	1·000	1·000	1·000
918	1·069	0·995	1·009	1·180	1·027	1·047	1·185	1·075	1·106
992	0·970	0·978	0·974	1·000	1·000	1·000	1·000	1·000	1·000
1005	0·981	0·984	0·962	0·963	1·006	1·001	0·824	0·971	0·946
1007	1·000	1·000	1·000	0·914	0·996	0·993	0·800	1·003	1·002
1242	1·000	1·000	1·000	0·970	0·963	0·961	0·815	0·882	0·875
2680	1·000	1·004	1·003	1·014	1·016	1·017	0·989	1·048	1·053
Average	0·995	0·976	0·973	1·011	0·987	0·986	0·968	0·985	0·984
W/L	8/6	14/3	13/4	8/7	13/5	12/6	10/4	9/7	9/7

obtained directly from Table VI (ratios of 8th and 5th, 9th and 6th, and 10th and 7th columns, respectively).

Note that, for all the columns in Table VII, the number of occasions in which the reordering algorithms with the SGPD are better (ratio $<1{\cdot}0$) for B, P and $\hat{W}$ reduction, is larger than the number of occasions in which the original reordering algorithms are better (ratio $>1{\cdot}0$). These global quantitative results are shown in the last row of Table VII, where W (wins) means the number of occasions that the reordering algorithm with the SGPD (Step 1 in Tables III–V) wins against the original algorithms, and L (losses) means the number of occasions that the reordering algorithms with the SGPD loses against the original algorithms.Obviously, this comparison excludes ties (ratio = 1·0).

From Table VII, the following conclusions can be obtained in a general sense. For the RCM reordering, the SGPD is more effective than the G&L algorithm; for the GPS reordering, the SGPD is slightly better than the GPSD algorithm; for the GK reordering, the SGPD is also slightly better than the GPSD algorithm.

6. SGPD ALGORITHM RUNNING TIME PERFORMANCE

The numerical procedure which implements the SGPD algorithm (see Table I) has been presented in Section 4 and applied to numerical examples in Section 5. We have observed that the proposed eigensolution scheme makes the SGPD algorithm very time-consuming when compared to G&L[3,15] and GPSD[2] algorithms. The following three strategies may significantly reduce the SGPD running time: (1) tuning of convergence parameters; (2) change of the eigensolver; (3) vectorization/parallelization of the SGPD algorithm.

6.1. Tuning of convergence parameters

The goal of the SGPD algorithm (Table I) is to determine the locations of the smallest and largest eigenvector components in $\mathbf{y}_2(\mathbf{L})$ and not their actual values. Therefore, an accurate numerical solution of the eigenproblem may not be necessary. In many cases, a reasonable approximation to $\mathbf{y}_2(\mathbf{L})$ may suffice for the purposes of the SGPD algorithm. Hence, the convergence criterion (equation (5)) may be relaxed in favour of computational efficiency. This means that, in equation (5), the tolerance can be bigger than the default value TOL = 10^{-6} adopted previously.

To make the point stated above, a typical example is presented next. Consider the problem with $|V| = 503$ in Table II. The convergence verification for this problem is illustrated by Table VIII, which shows some preliminary information and the convergence results. The preliminary information includes $|V|$, $|E|$, $\delta(G)$ and the initial values for B, P and $\hat{W}$. The convergence results are: the adopted tolerance (TOL); the CPU time (seconds); λ_2; PV_1, $e(PV_1)$, and the RCM results for B, P and $\hat{W}$ using PV_1 as the starting vertex; and similarly, PV_2, $e(PV_2)$, and the RCM results for B, P and $\hat{W}$ using PV_2 as the starting vertex. The goal here is to assess the quality of the results for eccentricity and for B, P and $\hat{W}$ as the tolerance is increased. Note that, in this case, the pseudodiametrical vertices obtained with TOL = 10^{-6}

Table VIII. Computational efficiency (HP apollo 9000–720)

Preliminary information					
$\|V\|$	$\|E\|$	$\delta(G)$	P	$\hat{W}$	B
503	2762	23	36417	78·603	453

RCM(SGPD)													
TOL	Time (s)	λ_2	# Iter.	PV_1	$e(PV_1)$	B	P	$\hat{W}$	PV_2	$e(PV_2)$	B	P	$\hat{W}$
10^{-6}	96·23	0·1001	108	393	23	67	16667	36·114	12	23	60	14906	31·759
10^{-3}	15·09	0·1082	16	393	23	67	16667	36·114	12	23	60	14906	31·759
10^{-2}	9·04	0·1480	9	393	23	67	16667	36·114	12	23	60	14906	31·759
10^{-1}	5·62	0·2023	5	274	16	65	18370	37·912	81	22	68	17657	38·415

or TOL = 10^{-2} do not change. However, even if they change, they may still be acceptable results. Moreover, from TOL = 10^{-6} to TOL = 10^{-2}, the SGPD running time is reduced by an order of magnitude.

6.2. *Change of the eigensolver*

The dominant computation in the SGPD algorithm (Table I) is the determination of the eigenvector associated to the second eigenvalue of the Laplacian matrix. In this paper, the eigenproblem has been solved by the subspace iteration method as presented by Paulino *et al.*[25] It is worth investigating alternative eigensolvers (together with specific characteristics for improved computational efficiency) to be used by the SGPD algorithm. Promising candidates are Lanczos-type methods.[33,34,38–43]

A comparison between Lanczos and subspace iteration methods has been presented by Nour–Omid *et al.*[33] and Sehmi (Reference 34, p. 68). Simon[38] and Hsieh *et al.*[39] have used the Lanczos method to determine $\mathbf{y}_2(\mathbf{L})$ in recursive domain partitioning algorithms (bisection-type) for parallel finite element analysis. Recently, Barnard and Simon[40] have reported an improved multilevel implementation of this algorithm that is an order of magnitude faster than the one reported in Reference 38. Hendrickson and Leland[41,42] have used additional eigenvectors of the Laplacian matrix, besides $\mathbf{y}_2(\mathbf{L})$, to obtain higher-order partitions (quadrisection-, octasection-type algorithms). They have also developed an efficient multilevel algorithm for partitioning graphs.[43] However, investigation of these algorithms (Lanczos-type),[33,34,38–43] in the context of the present SGPD algorithm (see Table I), is a subject for future research.

6.3. *Vectorization/parallelization*

The numerically intensive part of the SGPD algorithm (Table I) involves standard vector operations using floating-point arithmetic (this is in contrast to other algorithms based on the level structure concept and using integer arithmetic). Therefore, this algorithm is well suited for computers with vector processors. Moreover, the algebraic nature of the algorithm favours its implementation in a parallel computing environment. For example, at Cornell Theory Center, we have available a computing environment that includes vector–scalar super-computing resources and parallel systems such as the IBM ES/9000–900. The investigation of the SGPD as well as other numerical algorithms in advanced computing environments is also a subject for future research.

7. CONCLUSIONS

A new algorithm (SGPD) has been proposed for finding a pseudoperipheral vertex or the end-points of a pseudodiameter in a graph. Based on comparative studies, this algorithm is, in general, more effective than the ones presented by Grimes *et al.*,[19] George and Liu (G&L),[15] and Gibbs *et al.* (GPSD).[2]

ACKNOWLEDGEMENTS

The first author acknowledges the financial support provided by the Brazilian agency CNPq. Ivan F. M. Menezes acknowledges the financial support provided by CAPES, which is another

Brazilian agency for research and development. Most of the computations in the present work have been performed in CADIF (Computer Aided Design Instructional Facility), at Cornell University. The authors are especially grateful to Catherine Mink, CADIF Director, for giving credit to the ideas presented to her and for providing the resources necessary to carry out this research. The excellent computing system support by John M. Wolf, from CADIF, is also acknowledged. Dr Gordon C. Everstine, from David W. Taylor Model Basin (Department of the Navy), has given the authors a computer tape with his collection of benchmark problems for testing nodal reordering algorithms. The first author thanks Dr Shang-Hsien Hsieh for useful discussions about computational graph theory. The authors thank Mr Khalid Mosalam for carefully reading the manuscript, for his constructive criticism and valuable suggestions to this work.

REFERENCES

1. E. Cuthill and J. McKee, 'Reducing the bandwidth of sparse symmetric matrices', *ACM Proceedings of 24th National Conference*, ACM Publication P-69, 1969, pp. 157–172.
2. N. E. Gibbs, W. G. Poole, Jr. and P. K. Stockmeyer, 'An algorithm for reducing the bandwidth and profile of a sparse matrix', *SIAM J. Numer. Anal.* **13**, (2), 236–250 (1976).
3. A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, New Jersey, U.S.A., 1981.
4. W. F. Smyth and W. M. L. Benzi, 'An algorithm for finding the diameter of a graph', in J. L. Rosenfeld (Ed.), *Proceedings of IFIP Congress 74*, North-Holland Publ. Co., 1974, pp. 500–503.
5. K. S. Booth and R. J. Lipton, 'Computing extremal and approximate distances in graphs having unit cost edges', *Acta Inf.*, **15**, 319–328 (1981).
6. N. E. Gibbs, 'Algorithm 509 – A hybrid profile reduction algorithm,' *ACM Trans. Math. Softw.* **2**, (4), 378–387 (1976).
7. I. P. King, 'An automatic reordering scheme for simultaneous equations derived from network systems', *Int. j numer. methods eng.*, **2**, 523–533 (1970).
8. R. A. Snay, 'Reducing the profile of sparse symmetric matrices', *Bull. Géod.*, **50**, (4), 341–352 (1976).
9. S. W. Sloan, 'A FORTRAN program for profile and wavefront reduction', *Inter. j. numer. methods eng.*, **28**, 2651–2679 (1989).
10. S. R. P. Medeiros, P. M. Pimenta and P. Goldenberg, 'An algorithm for profile and wavefront reduction of sparse matrices with a symmetric structure', *Eng. Comput.*, **10**, (3), 257–266 (1993).
11. S. J. Fenves and K. H. Law, 'A two-step approach to finite element ordering', *Int. j. numer. methods eng.*, **19**, 891–911 (1983).
12. M. Hoit and E. L. Wilson, 'An equation numbering algorithm based on a minimum front criteria', *Comput. Struct.*, **16**, (1–4), 225–239 (1983).
13. R. K. Livesley and M. A. Sabin, 'Algorithms for numbering the nodes of finite-element meshes', *Comput. Syst. Eng.*, **2**, (1), 103–114 (1991).
14. K. Y. Cheng, 'Note on minimizing the bandwidth of sparse, symmetric matrices', *Computing*, **11**, 27–30 (1973).
15. A. George and J. W.-H. Liu, 'An implementation of a pseudoperipheral node finder', *ACM Trans. Math. Softw.*, **5**, (3), 284–295 (1979).
16. J. K. Pachl, 'Finding pseudoperipheral nodes in graphs', *J. Comp. Syst. Sci.*, **29**, 48–53 (1984).
17. W. F. Smyth, 'Algorithms for the reduction of matrix bandwidth and profile', *J. Comput. Appl. Math.*, **12 & 13**, 551–561 (1985).
18. A. Kaveh, 'Algebraic graph theory for ordering', *Comput. Struct.*, **37**, (1), 51–54 (1990).
19. R. G. Grimes, D. J. Pierce and H. D. Simon, 'A new algorithm for finding a pseudoperipheral node in a graph', *SIAM J. Matrix Anal. Appl.*, **11**, (2), 323–334 (1990).
20. P. D. Straffin, Jr., 'Linear algebra in geography: eigenvectors of networks', *Math. Mag.*, **53**, (5), 269–276 (1980).
21. P. Sadayappan and F. Ercal, 'Nearest-neighbor mapping of finite element graphs onto processor meshes', *IEEE Trans. Computers*, **C-36**, (12), 1408–1424 (1987).

22. G. C. Everstine, 'A comparison of three resequencing algorithms for the reduction of matrix profile and wavefront', *Int. j. numer. methods eng.*, **14**, 837–853 (1979).
23. F. Harary, *Graph Theory*, Addison Wesley, Reading, Massachusetts, 1969.
24. D. M. Cvetković, M. Doob and H. Sachs, *Spectra of Graphs – Theory and Application*, Academic Press, New York, 1980.
25. G. H. Paulino, I. F. M. Menezes, M. Gattass and S. Mukherjee, 'Node and element resequencing using the Laplacian of a finite element graph – Part I: General concepts and algorithm', *Int. j. numer. methods eng.*, **37**, (9), 1511–1530 (1994).
26. G. H. Paulino, I. F. M. Menezes, M. Gattass and S. Mukherjee, 'Node and element resequencing using the Laplacian of a finite element graph – Part II: Implementation and numerical results', *Int. j. numer. methods engin.*, **37**, (9), 1531–1555 (1994).
27. I. Arany, W. F. Smyth and L. Szóda, 'An improved method for reducing the bandwidth of sparse symmetric matrices', in C. V. Freiman (Ed.), *Proceedings of IFIP Congress 71*, North-Holland, 1972, pp. 1246–1250.
28. W. F. Smyth and I. Arany, 'Another algorithm for reducing bandwidth and profile of a sparse matrix', in S. Winkler (Ed.), *AFIPS Conference Proceedings*, AFIPS Press, Vol. 45, 1976, pp. 987–994.
29. W. N. Anderson, Jr., and T. D. Morley, 'Eigenvalues of the Laplacian of a graph', *Linear and Multilinear Algebra*, **18**, 141–145 (1985); (originally published as University of Maryland Technical Report TR-71-45, October 1971).
30. M. Fiedler, 'Algebraic connectivity of graphs', *Czech. Math. J.*, **23**, (98), 298–305 (1973).
31. M. Fiedler, 'A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory', *Czech. Math. J.*, **25**, (100), 619–633 (1975).
32. A. Pothen, H. D. Simon and K. P. Liou, 'Partitioning sparse matrices with eigenvectors of graphs', *SIAM J. Matrix Anal. Appl.*, **11**, (3), 430–452 (1990).
33. B. Nour-Omid, B. N. Parlett and R. L. Taylor, 'Lanczos versus subspace iteration for solution of eigenvalue problem', *Int. j. numer. methods eng.*, **19**, 859–871 (1983).
34. N. S. Sehmi, *Large Order Structural Eigenanalysis Techniques – Algorithms for Finite Element Systems*, Ellis Horwood, Chichester, U.K., 1989.
35. I. S. Duff, R. G. Grimes and J. G. Lewis, 'Sparse matrix test problems', *ACM Trans. Math. Softw.*, **15**, 1–14 (1989).
36. J. G. Lewis, 'Implementation of the Gibbs–Poole–Stockmeyer and Gibbs–King algorithms', *ACM Trans. Math. Softw.*, **8**, (2), 180–189 (1982).
37. H. L. Crane Jr., N. E. Gibbs, W. G. Poole Jr. and P. K. Stockmeyer, 'Algorithm 508 – Matrix bandwidth and profile reduction', *ACM Trans. Math. Softw.*, **2**, (4), 375–377 (1976).
38. H. D. Simon, 'Partitioning of unstructured problems for parallel processing', *Comput. Syst. Eng.*, **2**, (2/3), 135–148 (1991).
39. S. H. Hsieh, G. H. Paulino and J. F. Abel, 'Recursive spectral algorithms for automatic domain partitioning in parallel finite element analysis', *Comput. Methods Appl. Mech. Eng.*, (in press).
40. S. T. Barnard and H. D. Simon, 'A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems', in R. F. Sincovec *et al.* (Eds), *Parallel Processing for Scientific Computing*, *SIAM*, 1993, pp. 711–718.
41. B. Hendrickson and R. Leland, 'An improved spectral graph partitioning algorithm for mapping parallel computations', *SANDIA Report SAND92–1460*, Category UC–405, Sandia National Laboratories, Albuquerque, NM 87185, U.S A., September 1992.
42. B. Hendrickson and R. Leland, 'The CHACO user's guide – version 1.0', *SANDIA Report SAND93–2339*, Category UC–405, Sandia National Laboratories, Albuquerque, NM 87185, U.S.A., November 1993.
43. B. Hendrickson and R. Leland, 'A multilevel algorithm for partitioning graphs', *SANDIA Report SAND93–1301*, Category UC–405, Sandia National Laboratories, Albuquerque, NM 87185, U.S.A., October 1993 (draft).
44. S. T. Barnard, A. Pothen and H. D. Simon, 'A spectral algorithm for envelope reduction of sparse matrices', Technical Report CS-93-49, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, October 1993 (submitted to *J. Numer. Linear Algebra Appl.*).
45. F. Buckley and F. Harary, *Distance in Graphs*, Addison Wesley, Redwood City, California, 1990.