RAIRO. RECHERCHE OPÉRATIONNELLE

ALAIN BILLIONNET
JEAN-FRANÇOIS BRÊTEAU

A comparison of three algorithms for reducing the profile of a sparse matrix

RAIRO. Recherche opérationnelle, tome 23, n° 3 (1989), p. 289-302

<http://www.numdam.org/item?id=RO_1989__23_3_289_0>

Recherche opérationnelle/Operations Research
(vol. 23, n° 3, 1989, p. 289 à 302)

A COMPARISON OF THREE ALGORITHMS FOR REDUCING THE PROFILE OF A SPARSE MATRIX (*)

by Alain BILLIONNET (1) and Jean-François BRÊTEAU (1)

Abstract. — *First a variant of a well-known algorithm, the Levy algorithm, is described. Then a new algorithm for reducing the profile of a sparse matrix is proposed. These two algorithms and the commonly-used reverse Cuthill-McKee algorithm are tested and compared for their ability to reduce matrix profile.*

Keywords : Sparse matrix; profile reduction; heuristic; graph; interval graph.

Résumé. — *Nous décrivons tout d'abord une variante d'un algorithme classique, l'algorithme de Levy, puis nous proposons un nouvel algorithme pour réduire le profil d'une matrice creuse. Ces deux algorithmes ainsi qu'un troisième, largement utilisé, le « reverse Cuthil-McKee algorithm », sont testés et comparés.*

Mots clés : Matrice creuse; réduction de profil; heuristique; graphe; graphe d'intervalles.

1. INTRODUCTION

Many problems of scientific and engineering interest reduce to the problem of solving a system of linear equations

$$A x = b$$

where A is an n by n, symmetric, positive definite coefficient matrix, b is a vector of length n and x is the solution vector of length n. Applying Cholesky's method to A yields the triangular factorization

$$A = LL^T$$

where L is lower triangular with positive diagonal elements. Let us note that such a factorization always exists when A is symmetric and positive definite.

(*) Received December 1987, revised March 1989.
(1) Institut d'Informatique d'Entreprise, Conservatoire national des Arts et Métiers, 18, allée Jean-Rostand, 91002 Evry.

Recherche opérationnelle/Operations Research, 0399-0559/89/03 289 14/$ 3.40

The system $Ax=b$ becomes $LL^Tx=b$ and by substituting $y=L^Tx$, we can obtain x by solving the triangular systems

$$Ly=b \qquad \text{and} \qquad L^Tx=y.$$

An important fact about applying Cholesky's method to a sparse matrix A is that the matrix usually suffers fill-in. That is L has nonzeros in positions which are zero in the lower triangular part of A. However a judicious reordering of the rows and columns of the coefficient matrix can lead to enormous reductions in fill-in, and hence savings in computer execution time and storage. We consider here one of the simplest methods for solving sparse systems, the envelope or profile method. The objective of this method is to reorder the rows and columns of A so that the nonzeros in the obtained matrix are clustered near the main diagonal since this property is retained in the corresponding Cholesky factor L. We analyse and compare here three algorithms for the reduction of matrix profile: the "Reverse Cuthill McKee algorithm" [CUT-MCK, 69], [GEO, 71], a variant of the King [KIN, 70] and Levy [LEV, 71] algorithms and a new algorithm that we propose here.

2. BASIC CONCEPTS FROM GRAPH THEORY

If V is a finite nonempty set and $E \subseteq \{\{a, b\} : a \neq b \text{ and } a, b \in V\}$ is a collection of unordered pairs of elements of V, then $G=(V, E)$ is a finite undirected graph. Given a n by n symmetric matrix $A=(a_{ij})$ we can define a graph $G=(V, E)$ where V has n vertices $v_1, \ldots, v_n$ and $\{v_i, v_j\} \in E$ if $a_{ij} \neq 0$ and $i \neq j$. The elements of E are called edges. If $\{v_i, v_j\} \in E$ then v_i and v_j are said to be adjacent and we denote by $\Gamma(v_j)$ the set of vertices adjacent to v_j. The degree of a vertex is the number of vertices adjacent to it.

For $A \subset V$ let us define $\Gamma(A)=\bigcup_{v \in A} \Gamma(v)$, $\mathrm{Adj}(A)=\Gamma(A)-A$ and $\widehat{\mathrm{Adj}}(A)=\Gamma(A) \cup A$.

G is connected if there exists a path between all pairs of vertices and $A \subset V$ induces a complete subgraph of $G=(V, E)$ if all pairs of distinct elements of A belong to E.

$G=(V, E)$ with $V=\{v_1, \ldots, v_n\}$ is an interval graph if there exists a set $\{I_1, \ldots, I_n\}$ of intervals of the real line such that for all pairs of distinct vertices v_i, v_j of V, $\{v_i, v_j\} \in E$ if and only if $I_i \cap I_j \neq \varnothing$.

3. THE ENVELOPE METHOD

A well known scheme for exploiting sparsity of A is the so-called envelope or profile method. Let A be an n by n symmetric positive definite matrix, with entries a_{ij}. For the i-th row of A let

$$f_i(A) = \min \{j \mid a_{ij} \neq 0\}.$$

The envelope of A, denoted by $\mathrm{Env}(A)$ is defined by

$$\mathrm{Env}(A) = \{\{i, j\} \mid f_i(A) \leqq j < i\}$$

and the quantity $|\mathrm{Env}(A)|$ is called the profile or envelope size of A and is given by

$$|\mathrm{Env}(A)| = \sum_{i=1}^{n} [i - f_i(A)].$$

Example

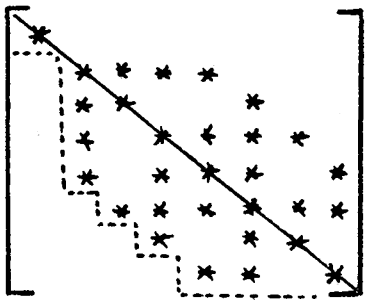

Figure 1. — A matrix whose envelope size is 15 (nonzeros are depicted by ∗)

The problem that we consider here is to find, for a symmetric and positive definite matrix A a reordering of the rows and columns of A which minimizes the envelope size of the obtained matrix. Finding a "good" reordering of A can be regarded as finding a "good" numbering of the vertices of the associated graph $G = (V, E)$.

The problem of minimizing the envelope size of a matrix A is NP-complete. It is proved in [BIL, 86] that this problem is equivalent to that of finding the minimum number of edges which must be added to the associated graph to obtain an interval graph. Finding the optimal envelope size of a matrix A

is undoubtedly a very difficult problem. The three reordering algorithms that we analyse here allow us to only obtain "good" envelope sizes.

4. THREE REORDERING ALGORITHMS

4.1. The reverse Cuthill-McKee algorithm

Perhaps the most widely used profile reduction ordering algorithm is a variant of the Cuthill-McKee ordering [CUT-MCK, 69]: the reverse Cuthill-McKee ordering (RCM) [GEO, 71]. We have considered here the following version of the (RCM) algorithm for the graph associated to A which is supposed to be connected. This version is that which is presented in [GEO-LIU, 81].

step 1: determine a starting vertex r and assign $v_1 \leftarrow r$;

step 2: for $i=1, 2, \ldots, n$ find all the unnumbered vertices adjacent to the vertex v_i and number them in increasing order of degree;

step 3: the reverse Cuthill-McKee ordering is given by $y_1, y_2, \ldots, y_n$ where $y_i = v_{n-i+1}$ for $i=1, 2, \ldots, n$.

We now consider the problem of finding a starting vertex (or a pseudo-peripheral vertex) for the (RCM) algorithm. Given a vertex $v \in V$, the level structure rooted at v is the partitioning $\mathscr{L}(v)$ of V satisfying

$$\mathscr{L}(v) = \{L_0(v), L_1(v), \ldots, L_{l(v)}(v)\}$$

where

$$L_0(v) = \{v\}, \; L_1(v) = \mathrm{Adj}(L_0(v)),$$

and

$$L_i(v) = \mathrm{Adj}(L_{i-1}(v)) - L_{i-2}(v), \qquad i=2, 3, \ldots, l(v).$$

$l(v)$ is called the length of $\mathscr{L}(v)$ and the width $w(v)$ of $\mathscr{L}(v)$ is defined by $w(v) = \max \{|L_i(v)| \mid 0 \leqq i \leqq l(v)\}$.

We can now describe the starting vertex finding algorithm which is essentially a modification of an algorithm due to Gibbs *et al.* [GIB-POO-STO, 76].

step 1: choose an arbitrary vertex r in V;

step 2: construct the level structure rooted at r: $\mathscr{L}(r)$;

step 3: choose a vertex v in $L_{l(r)}(r)$ of minimum degree;

step 4: (*a*) construct the level structure rooted at v; (*b*) if $l(v) > l(r)$, set $r \leftarrow v$ and go to step 3;

step 5: the vertex v is a starting vertex.

4.2. The Levy algorithm [LEV, 71]

R. Levy has proposed an algorithm for reducing the profile of a symmetric matrix. His algorithm for a connected graph can be described as follows

step 1: determine a pseudo-peripheral vertex r and assign $v_1 \leftarrow r$;

step 2: for $i=1, 2, \ldots, n-1$ find an unnumbered vertex y such that $|\mathrm{Adj}(\{v_1, \ldots, v_i, y\})|$ is minimum; number the vertex y as v_{i+1};

step 3: the Levy ordering is given by $v_1, v_2, \ldots, v_n$.

This algorithm reduces the profile by a local minimization of the frontwidth. (For a matrix A, the i-th frontwidth of A is $\omega_i(A) = |\{k \mid k > i$ and $a_{kl} \neq 0$ for some $l \leqq i\}|$.) The only difference between this algorithm and that of King [KIN, 70] is the set in which the vertex y is searched. In the King algorithm y belongs to $\mathrm{Adj}(\{v_1, \ldots, v_i\})$.

4.3. A variant of the Levy algorithm

We propose here a variant of the Levy algorithm (VL). In this method for choosing y we distinguish (step 2 of the Levy algorithm) between the vertices which belong to $\mathrm{Adj}(\{v_1, \ldots, v_i\})$ and the other unnumbered vertices. This algorithm can be described as follows

step 1: determine a pseudo-peripheral vertex r and assign $v_1 \leftarrow r$.

step 2: for $i=1, 2, \ldots, n-1$ find an unnumbered vertex y such that the cardinality of the set $T_i(y) = \widehat{\mathrm{Adj}}(\{v_1, \ldots, v_i, y\})$ is minimum; number the vertex y as v_{i+1}.

step 3: the (*VL*) ordering is given by $v_1, v_2, \ldots, v_n$.

Justification of the (VL) algorithm

Let us consider a n by n matrix A and its associated graph. Minimizing the envelope size of a matrix is equivalent to maximizing the number of zeros outside the envelope. This number of zeros is

$$\sum_{i=2}^{n} (i-1).\left|\widehat{\mathrm{Adj}}(\{v_i\}) - \widehat{\mathrm{Adj}}(\{v_1, \ldots, v_{i-1}\})\right|.$$

So minimizing the envelope size is equivalent to finding an ordering $v_1, \ldots, v_n$ which maximizes the expression $\sum_{i=1}^{n} (i-1)\,|S_{i-1}(v_i)|$ where

$$S_{i-1}(v_i) = \widehat{\mathrm{Adj}}(\{v_i\}) - \widehat{\mathrm{Adj}}(\{v_1, \ldots, v_{i-1}\})$$

for $i=2, \ldots, n$ and $S_0(v_1) = \widehat{\mathrm{Adj}}(\{v_1\})$. Let us notice that for any ordering $v_1, \ldots, v_n$, $\sum_{i=1}^{n} |S_{i-1}(v_i)| = n$. Let $\{v_1, \ldots, v_i\}$ be the set of numbered vertices at a step of the algorithm; we will choose next the vertex y which minimizes the cardinality of $S_i(y)$. So we hope that for the high values of i the cardinality of $S_i(y)$ will be large.

Let us consider the step 2 of the (VL) algorithm.

$$\begin{aligned} |T_i(y)| &= |\widehat{\mathrm{Adj}}(\{v_1, \ldots, v_i, y\})| \\ &= |\widehat{\mathrm{Adj}}(\{v_1, \ldots, v_i\})| + |\widehat{\mathrm{Adj}}(\{y\}) - \widehat{\mathrm{Adj}}(\{v_1, \ldots, v_i\})| \\ &= |\widehat{\mathrm{Adj}}(\{v_1, \ldots, v_i\})| + |S_i(y)|. \end{aligned}$$

Hence the step 2 of the (VL) algorithm is equivalent to finding y which minimizes $|S_i(y)|$.

4.4. A new algorithm for reducing the profile of a sparse matrix

For a matrix A, if $a_{ij} \neq 0$ for all $\{i, j\} \in \mathrm{Env}(A)$ then we say that the envelope of A is full. It has been proved in [TAR, 76] that the envelope of a matrix A is full if and only if the associated graph $G=(V, E)$ is an interval graph.

Let us consider a matrix A with a full envelope and the associated graph $G=(V, E)$.

On the one hand each vertex v_i $(i=1, \ldots, n-1)$ of V clearly satisfies

$$|\Gamma(v_i) \cap \{v_{i+1}, \ldots, v_n\}| = \max\{|\Gamma(v_k) \cap \{v_{i+1}, \ldots, v_n\}| \mid k=1, \ldots, i\}. \quad (1)$$

On the other hand, for each $j \in \{1, \ldots, n\}$

$$\mathscr{C}_i = \{v_i\} \cup [\Gamma(v_i) \cap \{v_{i+1}, \ldots, v_n\}]$$

is a complete subgraph of G [BIL, 86]. (2)

Hence we propose the following algorithm (N) for reducing the envelope size of a matrix whose associated graph is supposed to be connected:

step 1: determine a pseudo-peripheral vertex r and assign $v_n \leftarrow r$;

step 2: for $i=n-1, n-2, \ldots, 1$ find an unnumbered vertex y such that $|\Gamma(y) \cap \{v_{i+1}, \ldots, v_n\}|$ is maximum. Number the vertex y as v_i;

step 3: the ordering is given by $v_1, v_2, \ldots, v_n$.

because if G has a full envelope (is an interval graph) it has been proved [SHI, 84] that all the numberings of the vertices of G given by this algorithm satisfy the property (2).

5. INTERESTING CASES FOR THE (N) ALGORITHM

Example 1: Let us consider the graph of figure 2 in which there exists a vertex of high degree

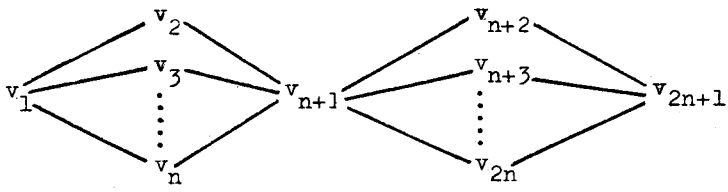

Figure 2

v_1 is a pseudo-peripheral vertex and the (RCM) algorithm leads to the following order of the vertices

$$v_{2n+1}, v_{2n}, \ldots, v_{n+2}, v_{n+1}, v_n, \ldots, v_2, v_1.$$

The corresponding envelope size is $(n-1)(n+2)$.

If we do not take a starting vertex in the set $\{v_1, v_{n+1}, v_{2n+1}\}$ we also obtain an envelope of size $O(n^2)$. For example if we choose v_2 as starting vertex, we obtain the following order of the vertices

$$v_{2n+1}, v_{2n}, \ldots, v_{n+2}, v_n, \ldots, v_3, v_{n+1}, v_1, v_2$$

and an envelope size equal to $n^2/2+5n/2-2$.

The (VL) algorithm with v_{2n+1} as starting vertex gives the following order

$$v_{2n+1}, v_{2n}, \ldots, v_{n+2}, v_n, \ldots, v_3, v_{n+1}, v_2, v_1$$

and the envelope size also is $n^2/2+5n/2-2$.

The (N) algorithm with v_{2n+1} as starting vertex gives the following order

$$v_n,\ v_{n-1},\ \ldots,\ v_4,\ v_1,\ v_3,\ v_2,\ v_{2n},\ v_{2n-1},\ \ldots,\ v_{n+4},\ v_{n+1},\ v_{n+3},\ v_{n+2},\ v_{2n+1}$$

and the envelope size is $4n$.

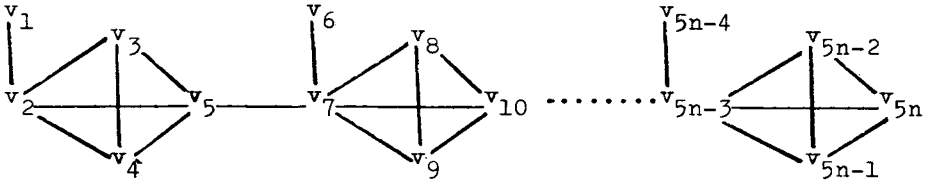

Figure 3

Example 2. — Let us consider the graph of figure 3 in which the vertices of low degree are dispersed. This graph is formed by n subgraphs of five vertices.

The (VL) algorithm numbers first the vertices of degree 1: $v_1,\ v_6,\ \ldots,\ v_{5n-4}$ then the vertices of the complete subgraphs of 4 vertices: $v_2,\ v_3,\ v_4,\ v_5,\ v_7,\ v_8,\ v_9,\ v_{10},\ \ldots,\ v_{5n-3},\ v_{5n-2},\ v_{5n-1},\ v_{5n}$. The corresponding envelope size is $n(5n+9)/2$.

The (N) algorithm with v_1 as starting vertex gives the order

$$v_{5n},\ v_{5n-1},\ v_{5n-2},\ v_{5n-4},\ v_{5n-3},\ v_{5n-5},\ \ldots,\ v_5,\ v_4,\ v_3,\ v_2,\ v_1.$$

The envelope size is in this case: $8n-1$.

Let us remark that in this example the (RCM) algorithm gives the same order as the (N) algorithm.

6. IMPLEMENTATION OF THE THREE ALGORITHMS

The graphs (which are associated with sparse matrices) are represented classically by two one dimensional arrays:

TABADJ: the adjacency lists of each vertex are stored sequentially in this array of length $2m$ (m is the number of edges of the graph);

TABSOM: it is an index array of length n containing pointers to the beginning of each adjacency list in TABADJ.

For the three algorithms (*RCM*), (*VL*) and (*N*) the graphs are supposed to be connected. The starting vertex r is the same in the three cases *i.e.* a pseudo-peripheral vertex which is found by the method of section 4.1. At the end of the algorithm the new numbering of the vertices is stored in the one-dimensional array TABPERM where TABPERM $(i)=k$ means that the original vertex number k is the i-th vertex in the new ordering.

6.1. The (*RCM*) algorithm

The two main stages of this algorithm are

- the search of all the unnumbered neighbours of a vertex v;
- the numbering of these vertices in increasing order of degree.

Therefore we use a one-dimensional boolean array TABNUM which indicates for each vertex if it is numbered or not and another one-dimensional array which gives for each vertex its degree.

Selection of the minimum is used for sorting the unnumbered vertices by their degrees. Then it is not difficult to prove (see for example [GEO-LIU, 81], Chap. 4) that the time complexity of the (*RCM*) algorithm is bounded by $O(d.|E|)$ for a graph $G=(V, E)$ where d is the maximum degree *i.e.* $d=\max_{v \in V} |\Gamma(v)|$.

6.2. The (*VL*) algorithm

The two main stages of this algorithm are

- find an unnumbered vertex v such that $|T_i(v)|$ is minimum (we will call $|T_i(v)|-|\widehat{\mathrm{Adj}}(\{v_1, \ldots, v_i\})|$ the corrected degree of v).
- update the corrected degrees of the vertices.

For the first stage we need an array which indicates for each vertex if it is numbered or not. To find the vertex v whose corrected degree is minimum we use a set of linked lists (in order to avoid a search among all the vertices). The sets of vertices of same corrected degrees are stored in the same list.

For the second stage we need an array which indicates for each vertex if it belongs to the front of the numbering or not. (If $\{v_1, \ldots, v_i\}$ is the set of numbered vertices then the front is $\mathrm{Adj}(\{v_1, \ldots, v_i\})$.) The only vertices which are concerned with the updating of the corrected degree are the vertices of the set $T_i(v)-\mathrm{Adj}(\{v_1, \ldots, v_i\})$ and the unnumbered vertices adjacent to (at least) one vertex of this set.

For this second stage we need also an array which stores the corrected degree of each vertex. When the corrected degrees are updated the corresponding vertices are put into the appropriate list. In order to save computational time they are not suppressed from the other lists. So the same vertex can be stored simultaneously in several lists and for this reason the worst-case time complexity is only bounded by $O(|V|^2)$. Let us note that it is possible by using an ordered queue of doubly linked lists to obtain an $O(|E|)$ implementation (such lists are necessary for deletion purposes). But the computational experiments have shown that, at least for our test-problems, it was not interesting to suppress the vertices from the linked lists.

6.3. The (*N*) algorithm

The two main stages of this algorithm are

– find a vertex v whose number of numbered adjacent vertices $d_N(v)$ is maximum. (We shall call $d_N(v)$ the "restricted degree" of v.)

– update $d_N(y)$ for the unnumbered vertices y adjacent to v.

We use an array which indicates for each vertex if it is numbered or not and another one which gives, for each vertex, its restricted degree.

In order to obtain an efficient algorithm we also use here a set of linked lists. In each list are stored the vertices which have the same restricted degree. The choice of a vertex v automatically suppresses it in the linked list of maximum restricted degree. Then the unnumbered vertices adjacent to x are put in the lists which are associated to their new restricted degrees; these new values are equal to the old ones plus one. As in the previous algorithm it is not interesting to suppress these vertices from the other lists. The worst-case time complexity is bounded by $O(|V|^2)$ but the "experimental time complexity" is $O(|E|)$. Let us also note that, as for the (*VL*) algorithm, it is possible to have an implementation with a worst-case time complexity equal to $O(|E|)$.

7. COMPUTATIONAL RESULTS

The comparison of the three algorithms (*RCM*), (*VL*) and (*N*) takes into account the two following criteria

– the envelope size associated with the new numbering;

– the execution time necessary to obtain the new ordering.

Let us recall that to compare the three algorithms we always use the same starting vertex. Let us note also that these algorithms have been coded in the PASCAL language (optimized version) and that all computer runs were made on a VAX 11-750 (D.E.C.). We present in this section the most significant results that we have obtained.

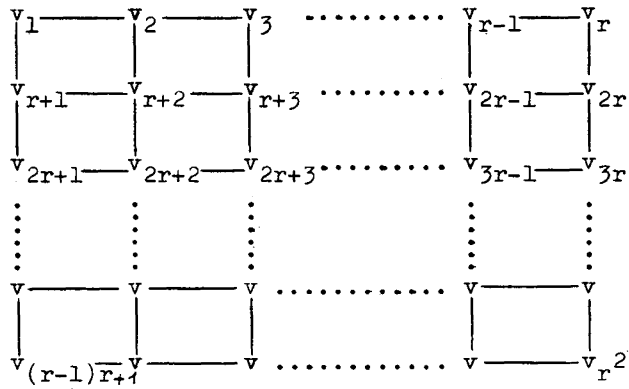

Figure 4

First of all we have considered the graphs of figure 4 with r rows, r columns and $n\ (=r^2)$ vertices.

The numbering "row by row" of the figure 4 leads to an envelope size of $r^3+O\ (r^2)$. The (RCM) and (VL) algorithms lead to a "diagonal" numbering and to an envelope size of $2/3\ r^3+O\ (r^2)$ (cf. *fig.* 5). The (N) algorithm leads to an "alternated" numbering (cf. *fig.* 6); the corresponding envelope size is $4/3\ r^3+O\ (r^2)$.

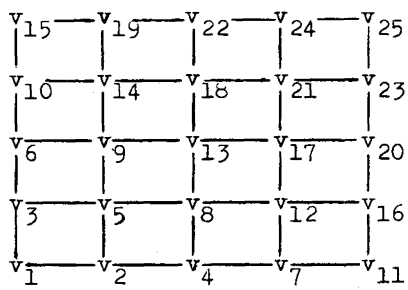

Figure 5 $(r=5)$

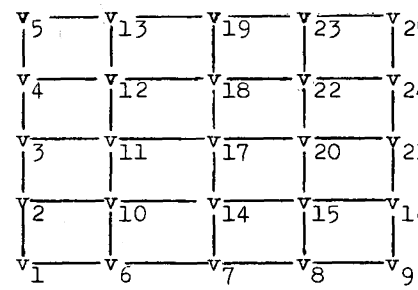

Figure 6 $(r=5)$

We have studied for the three algorithms and for these particular graphs the ratios between the CPU time (measured in seconds and multiplied by 10,000) and the number of edges in the graph. The results of the tests appear in table I.

TABLE I

n	(RCM)	(VL)	(L)
625	0.58	7.83	4.00
1,225	0.50	8.57	4.03
2,500	0.51	8.85	4.14
10,000	0.52	7.82	4.19
19,600	0.78	8.93	4.35

This table shows that the CPU time grows linearly with the number of edges for the three algorithms. The other conclusion that can be drawn from table I is that (RCM) is eight times faster than (N) and that (N) is twice as fast as (VL).

Then we have considered randomly generated graphs. First we consider some graphs of n vertices in which the degree of each vertex is greater than 2 and smaller than 6, the mean degree being 4. The table II allows us to compare the envelope sizes that were obtained by the three algorithms.

TABLE II

n	(RCM)	(VL)	(N)
100	2,249	1,783	1,917
500	52,371	42,879	45,719
1,000	216,111	164,480	183,561
2,000	850,463	674,341	741,833

The (VL) algorithm always leads to the best envelope size. The envelope sizes produced by the (N) algorithm are slightly smaller than the envelope sizes obtained using the (RCM) algorithm. The ratios between the CPU time (multiplied by 10,000) and the number of edges are presented in table III.

The results of these tests are very similar to those of table I.

We have also randomly generated some graphs with a few vertices of large degree; the degree of the others are greater than 1 and smaller than 5. The results appear in table IV.

If a few vertices have a large degree the (RCM) algorithm leads to a very large envelope size comparatively with the envelope sizes that can be obtained

TABLE III

n	(RCM)	(VL)	(N)
100	0.50	8.04	4.02
500	0.60	7.68	4.69
1,000	0.54	8.46	4.03
2,000	0.60	8.07	4.04

TABLE IV

Number of vertices	Number of vertices of large degree	Mean degree	Envelope size produced by		
			(RCM)	(VL)	(N)
100	1	3.50	2,007	704	1,287
100	2	4.30	2,199	689	1,700
100	10	11.76	2,174	1,103	2,279
500	1	3.52	59,536	17,178	33,772
1,000	1	3.56	230,118	68,574	128,635

by other algorithms. The ratios between the CPU time ($\times 10{,}000$) and the number of edges are presented in table V [for the worst cases of (RCM)].

TABLE V

Number of vertices	Number of vertices of large degree	Mean degree	Ratios corresponding to		
			(RCM)	(VL)	(N)
100	1	3.50	1.14	8.00	4.00
500	1	3.52	4.89	7.50	4.09
1,000	1	3.56	9.72	7.98	4.10

For the (VL) and (N) algorithms the results are similar to those of tables I and III. For the (RCM) algorithm, the ratio is greater than those corresponding to (VL) and (N) when there are more than 1,000 vertices in the graph. That can be explained by the fact there is a vertex of large degree: this vertex is considered from the outset of the execution of (RCM) since it is adjacent to almost all the other vertices.

Concluding remarks

On the three kinds of test problems the (*VL*) algorithm always leads to the best envelope size. However this algorithm is generally twice slower than (*N*) and sixteen times slower than (*RCM*) except in the case where there are some vertices of large degree; in this case and for the conditions under which the tests were made, the (*RCM*) algorithm does not seem interesting.

To conclude we can say in short that

— (*RCM*) is a fast algorithm;

— (*VL*) achieves very good reduction of the profile;

— (*N*) yields an interesting compromise between envelope size and CPU time.

REFERENCES

[BER, 70] C. BERGE, *Graphes et Hypergraphes*, Dunod, Paris.

[BIL, 86] A. BILLIONNET, *On Interval Graphs and Matrice Profiles*, R.A.I.R.O. Operations Research, Vol. 20, No. 3, août, pp. 245-256.

[CUT-MCK, 69] E. CUTHILL and J. MCKEE, *Reducing the Bandwidth of Sparse Symmetric Matrices*, Proc. 24th Nat. Conf. Assoc. Comput. Mach., ACM Publ., pp. 157-172.

[EVE, 79] G. C. EVERSTINE, *A Comparison of Three Resequencing Algorithms for Reduction of Matrix Profile and Wavefront*, Int. J. for Num. Meth. in engineering, Vol. 14, pp. 837-853.

[GEO, 71] A. GEORGE, *Computer Implementation of the Finite Element Method*, STAN-CS-71-208, Computer Science Dept., Stanford Univ., Calif.

[GEO-LIU, 81] A. GEORGE and J. W. H. LIU, *Computer Solutions of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 324 p.

[GIB-POO-STO, 76] N. E. GIBBS, W. G. POOLE and P. K. STOCKMEYER, *An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix*, S.I.A.M. J. Numer. Anal., Vol. 13, No. 2, April, pp. 236-250.

[GIB-POO-STO, 76] N. E. GIBBS, W. G. POOLE and P. K. STOCKMEYER, *A Comparison of Several Bandwidth and Profile Reduction Algorithms*, A.C.M. Transactions on Math. Software, Vol. 2, No. 4, December, pp. 322-330.

[GOL, 80] M. C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 284 p.

[KIN, 70] I. P. KING, *An Automatic Reordering Scheme for Simultaneous Equations Derived from Network Systems*. Int. J. Numer. Meth. Engrg., Vol. 2, pp. 523-533.

[LEV, 71] R. LEVY, *Resequencing of the Structural Stiffness Matrix to Improve Computational Efficiency*, J.P.L. Quart. Tech. Rev., Vol. 1, pp. 61-70.

[LEW, 82] J. G. LEWIS, *Implementation of the Gibbs-Poole-Stockmeyer and Gibbs-King Algorithms*, A.C.M. Transactions on Mathematical Software, Vol. 8, No. 2, June, pp. 180-189.

[SHI, 84] D. R. SHIER, *Some Aspects of Perfect Elimination Orderings in Chordal Graphs*, Discrete Applied Mathematics, Vol. 7, pp. 325-331.

[TAR, 76] R. E. TARJAN, *Graph Theory and Gaussian Elimination in Sparse Matrix Computations*, J. R. BUNCH and D. J. ROSE Eds., Academic Press, New York, pp. 3-22.