Another algorithm for reducing bandwidth and profile of a sparse matrix

by W. F. SMYTH
International Labor Office
Geneva, Switzerland

and

ILONA ARANY
Computing Center of the Ministry of Labor
Budapest, Hungary

ABSTRACT

The paper describes a new bandwidth reduction method for sparse matrices which promises to be both fast and effective in comparison with known methods. The algorithm operates on the undirected graph corresponding to the incidence matrix induced by the original sparse matrix, and separates into three distinct phases: (1) determination of a spanning tree of maximum length, (2) modification of the spanning tree into a free level structure of small width, (3) level-by-level numbering of the level structure. The final numbering produced corresponds to a renumbering of the rows and columns of a sparse matrix so as to concentrate non-zero elements of the matrix in a band about the main diagonal.

INTRODUCTION

As electronic computers make possible computations on ever larger data sets, it has come to be realized that most large matrices are, in the nature of things, sparse; more precisely, that a matrix of (large) order \( n \) will generally contain only \( Kn \) non-zero elements, where \( K \) tends to decrease as \( n \) increases (\( K \) is often as little as 2 or 3 and only rarely greater than 20). There has been, accordingly, during the last ten years, a good deal of research into techniques for efficient computer handling of large sparse matrices. These techniques may be separated into two classes:

\[ T_1: \] techniques which deal with the given sparse matrix, more or less directly, in a sparse form (some typical approaches are surveyed in [2] and [3]);

\[ T_2: \] techniques which transform the given sparse matrix into a band form (which may then be processed further by efficient and well-known band matrix algorithms).

Initially "direct" techniques \( T_1 \) attracted more interest, and the utility of "band" techniques \( T_2 \) was occasionally questioned.\(^4\) Research into \( T_2 \) techniques was spurred however by the work of Cuthill-McKee,\(^6\) who described an effective bandwidth reduction algorithm with execution time linear in \( Kn \). Instead of dealing directly with the given matrix, Cuthill-McKee (hereafter called CM) dealt with the numbered graph whose connections correspond to the given matrix's zero/non-zero structure; they renumbered the vertices of this graph, and this renumbering therefore defined the interchanges of rows and columns required to transform the original sparse matrix into a band matrix.

The CM method was later modified in various ways, especially by Rose,\(^6\) who also contributed an important analysis of the application of Gaussian elimination to band matrices, showing in particular the importance of the profile.

The algorithm presented here, called the SA algorithm, is of class \( T_2 \). It results from efforts to improve on previous work; and has been directly stimulated by the approach suggested by Smyth-Benzi (hereafter called SB)\(^1\) as well as by the related algorithm published independently by Gibbs-Poole-Stockmeyer (hereafter called GPS).\(^8\) Like the CM and GPS algorithms, SA renumbers the vertices of a given numbered graph with the objective of minimizing the maximum difference between numbers assigned to connected vertices. Also in common with these algorithms, SA reduces profile as well as bandwidth\(^5\) and may be applied to non-symmetric as well as to symmetric matrices.

TERMINOLOGY

We use the term graph, and the symbol \( G \) or \( G(V, E) \), to denote a finite connected * undirected graph without

\* The algorithm may easily be modified to deal separately with the disjoint components of a single graph.

From the collection of the Computer History Museum (www.computerhistory.org)
loops or multiple edges defined on a vertex set \( V \) of cardinality \( n = |V| > 1 \) and an edge set \( E \) of cardinality \( m = |E| \). For any distinct pair of vertices \( u, v \in V \), we define the usual distance function \( \rho(u,v) \) to be the number of edges on the shortest path from \( u \) to \( v \). For a single vertex \( u \), we adopt the convention \( \rho(u,u) = 0 \); for unconnected vertices \( u,v \), we set \( \rho(u,v) = \infty \); if \( \rho(u,v) = 1 \), then \( u \) and \( v \) are said to be adjacent. The diameter of \( G \) is defined by

\[
diam(G) = \max_{u,v \in V} \rho(u,v).
\]

Since we assume \( G \) is connected, \( diam(G) \leq n-1 \).

Apart from such basic terms, we need for our purposes here to define four main concepts: level structure, free level structure, numbering, and bandwidth.

A level structure \( LS \) of a graph \( G(V,E) \) is an assignment of the vertices of \( V \) into sets, called levels \( L_1, L_2, \ldots, L_\lambda \), such that

1. \( L_1 \) contains at least one vertex;
2. for each \( k = 2, 3, \ldots, \lambda \), \( L_k \) contains every vertex not in a previous level which is adjacent to some vertex of \( L_{k-1} \).

It follows from this definition that if \( G \) is connected, \( LS \) contains all \( n \) vertices of \( V \); that the levels are disjoint; that \( 1 \leq \lambda \leq diam(G) + 1 \); and that \( LS \) is determined uniquely by the choice of vertices in \( L_\lambda \). We may therefore unambiguously denote \( LS \) by \( LS(L_\lambda) \) or, when \( L_\lambda = \{u\} \), by \( LS(u) \). We observe in fact that level \( L_\lambda \) of \( LS(L_\lambda) \) consists of exactly those vertices which occur in the \( k^{th} \) level of every spanning forest (SF) grown from the vertex set \( L_\lambda \) (for a description of this process see Reference 8). \( LS(L_\lambda) \) therefore corresponds (one-many) to the SF \( (L_\lambda) \), and \( LS(u) \) to the spanning trees \( ST(u) \). Whenever edges are not important, then, we may refer to \( LS \) and SF/ST interchangeably.

A free level structure \( FLS \) of a graph \( G(V,E) \) is an arrangement of all \( n \) vertices of \( V \) into \( \lambda \) levels \( L_1, L_2, \ldots, L_\lambda \), such that

1. no level is empty;
2. if \( u \in L_k \) then all vertices adjacent to \( u \) are in either \( L_{k-1}, L_k \) or \( L_{k+1} \).

We note that in this case also \( 1 \leq \lambda \leq diam(G) + 1 \), but that \( FLS(u) \) is no longer uniquely determined (Figure 1). For either \( LS \) or \( FLS \) we speak of the width of level \( k \), \( w(L_k) = |L_k| \), and the width of the structure, \( w(FLS) = \max_{1 \leq k \leq \lambda} w(L_k) \). \( \lambda \) is called the length of the structure.

Following GPS, we now define a numbering \( \alpha = \alpha(V) \) of \( G(V,E) \) to be a one-one map of \( V \) onto the first \( n \) natural numbers \( \{1, 2, \ldots, n\} \). For a given numbering \( \alpha, \) we may define \( \delta_\alpha(G) \), the bandwidth of the graph \( G \) relative to the numbering \( \alpha \), to be

\[
\delta_\alpha = \delta_\alpha(G) = \max_{u,v \in V} |\alpha(u) - \alpha(v)|.
\]

The bandwidth of \( G \) is then

\[
\delta = \delta(G) = \min_{\alpha} |\delta_\alpha(G)|.
\]

**GENERAL DESCRIPTION OF THE SA ALGORITHM**

Our goal is therefore, given \( G(V,E) \), to find a numbering \( \alpha = \alpha(V) \) such that \( \delta_\alpha = \delta(G) \). Our (heuristic) algorithm approaches this problem in three distinct phases:

**I Find LS**

Using the SB method, \( a \) a level structure \( LS = LS(u) \) is found such that \( \lambda = diam(G) + 1 \).

**II Find FLS**

Using a subset \( M \) \((|M| = \min \{|L_\lambda|, |\bar{u}/|\})\) of the vertices of level \( L_\lambda \) in \( LS(u) \), a new level structure, denoted \( LS' = LS'(M) \), is grown. \( LS' \) is also of length \( \lambda \). \( LS \) and \( LS' \) are systematically compared, and an FLS is determined such that \( w = w(FLS) \) is small (if possible, \( w = |\bar{u}/| \)).

**III Number FLS**

FLS is numbered on a level-by-level basis; that is, first, the integers \( \{1, 2, \ldots, \bar{u}\} \) are assigned to the vertices of the first level of FLS \( (w) \) denotes the width of the \( k^{th} \) level of FLS \);
second, the integers \(\{w_i + 1, w_i + 2, \ldots, w_i + w_j\}\) are assigned to the vertices of the second level; and so on, until all vertices have been numbered. The algorithm makes use of knowledge of the edges joining vertices of successive levels in an effort to minimize bandwidth: it searches for a level-by-level numbering \(\alpha\) such that the corresponding bandwidth satisfies

\[\delta_\alpha \leq w + \Delta,\]

where \(\Delta\) successively takes the values 0, 1, \ldots, \(w - 1\). The first numbering found which satisfies this condition is the required numbering \(\alpha\).

The justification of algorithms such as this is partly theoretical, partly experimental, and numerous variations in strategy are possible. In practical terms, we are trying to find a "reasonably good" numbering without needing to investigate all of the \(n!\) different possible numberings; our strategy therefore is always influenced by estimates, often very rough in nature, of the additional benefit to be expected from the additional effort expended. We will find an example of this kind of strategic thinking in Phase II of SA: we do not carry out an exhaustive search to find the FLS of truly minimum width, even though such a search might not on the average be too laborious, simply because (1) it seems that an exhaustive search would not be likely to decrease \(w\); (2) the result might be merely to decrease \(w\) by 1, but not decrease \(\delta_\alpha\), and in addition make numbering more lengthy and difficult. On the other hand, in Phase I, we propose using the SB algorithm\(^8\) instead of the GPS pseudo-diameter algorithm,\(^9\) because the former guarantees finding a longest spanning tree at (apparently) no additional cost. On the theoretical side, the basic justification for level-by-level numbering is the result of Arany-Sződa\(^10\) that corresponding to every numbering \(\alpha\) of \(G\) there exists at least one FLS whose width \(w = \delta_\alpha\), and which may be numbered on a level-by-level basis to yield the numbering \(\alpha\). Since the case \(\delta_\alpha = \delta\) is included in this result, it follows that level-by-level numbering of an FLS is an acceptable approach to bandwidth reduction, in that it does not exclude any minimum case. The result however does not point to any particular FLS-growing or FLS-numbering algorithms.

**PHASE I OF SA: FIND LS**

As noted above, the SB diameter algorithm is proposed here, because it guarantees finding \(\Delta = \text{diam}(G) + 1\) and apparently is comparable to the GPS pseudo-diameter algorithm in execution time. The SB algorithm is described fully in Reference 8 and is not included here\(^*\).

\(^*\) We do however provide a correction to the algorithm SPAN \((h,i)\) given in the Appendix of Reference 8. In step 7 replace the ending "." with "goto11.". Step 8 should read "[Is the vertex connected to jorig in level g+1?] If \(A=1\) and level \(v\), \(\text{elist}(j)\) = g+1, goto10; otherwise goto11.". Interchange steps 9 and 10.

GPS state in Reference 9 that their pseudo-diameter algorithm found the true diameter in all test cases. Figure 2 illustrates a graph of diameter 8 whose GPS pseudo-diameter is 6. The GPS algorithm would require the growth of 6 spanning trees to achieve this result, the SB algorithm 3 spanning trees (starting vertex \(1\)). Removal of edges (4,8) and (5,9) from Figure 2 and insertion of edge (4,5) would permit GPS to find a pseudo-diameter 7 at a cost of growing 4 spanning trees. For a symmetrized version of the Curtis matrix\(^4\) (Figure 3), SB finds diameter 7 at a cost of 6 spanning trees (starting vertex \(5\)), and GPS finds pseudo-diameter 7 from each of six starting vertices of minimum degree at an average cost of 5.5 spanning trees. See Table I.

A more detailed analysis of the application of the GPS pseudo-diameter algorithm to the Curtis matrix

**TABLE I—Comparison of SB and GPS**

<table>
<thead>
<tr>
<th>example</th>
<th>((\text{pseudo-)}) diameter</th>
<th>number of LS grown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2 (mod.)</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>Curtis matrix</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

From the collection of the Computer History Museum (www.computerhistory.org)
Figure 3
shows that 48 of the 54 vertices would, used as starting root vertices, yield maximum pseudo-diameter 7; of these 48 vertices, the 9 extreme vertices yield λ=8 immediately, 22 yield λ=7 but contain an extreme vertex in L₁, and 15 yield λ=6 with an extreme vertex in L₀ (there are two other special cases). These statistics appear to be related to the success of the GPS algorithm, and it would be interesting to know for much larger graphs also (n=500 to 5000) such a high percentage of optimal starting root vertices occurs.

GPS initially searches V for a vertex of lowest degree as starting root vertex, and from the corresponding L₀ selects additional root vertices in increasing order of degree. SB includes no such searches.

PHASE II OF SA: FIND FLS

The algorithm described below is an attempt to deal efficiently with difficult cases which may arise when G is not "well" connected (Figure 4); that is, when a number of vertices swing into different levels in the level structure LS' grown from vertices of L₀ in LS(u).

It seems that the growing of this "reverse" level structure (step (1) of the following algorithm) will normally accomplish what GPS accomplish in phase II of their algorithm; the other steps in the algorithm given below make use of the same information proposed by GPS, but in a more flexible manner.

Given LS(u) of length λ=diam(G)+1, phase II proceeds as follows:

1. Grow "reverse" LS'. For R=|̄n/λ|,* choose a set of vertices M<LS such that |M|=\min \{R, w(L₀)\} and grow LS' (M) with levels Lₖ' of width w'(k), k=1,2, ... , λ (L₀'=M).
2. Calculate initial value of T: T=S\min \{R-\lfloor w(k),0\rfloor\}; if T=0, goto (16).
3. [Two criteria are used to determine whether the movement of a vertex of FLS(M) from one level to another yields a "better" FLS: the value T given here and the value T'=\sum \{R-w'(k)\}\]. The FLS is "better" if the vertex movement increases T (by unity) or decreases T. T is also an "absolute" criterion, since any FLS for which T=0 is accepted; T is not calculated explicitly because it is not "absolute", so that only T' is required (see step (12)). Both T and T' may be used since T decreases iff T does not increase.
4. Calculate level pairs.] As in Reference 9, associate with each vertex v \epsilon V a level pair (g,v), where g is the level of v in LS(M), where h=λ-k+1, and k is the level of v in LS(u).
5. [Separate "movable" vertices into r connected components.] As in Reference 9, separate all vertices such that g=h, into r disjoint connected components C₁,C₂, ... , C_r of cardinality c₁,c₂, ... , c_r, respectively, arranged so that c₁≤c₂≤...≤c_r.
6. [Organize storage for vertices in component Cᵣ.] Associate with each vertex v \epsilon Cᵣ g,j=1,2, ... , cᵣ an identifier (v),g(j),h(j)), where v is a pointer to the level pair of the vertex v=xₗₙₐ₃ and [g(j),h(j)] is the level pair of vₙₐ₃ (that is, g(i)=g(i), h(j)=h(i)); order the vertices so that g(i)≥g(i)≥...≥g(cᵣ).
7. [Initialize vertex movement parameters for Cᵣ.] j=1; \Delta T=, \Delta T'=0; for each k=1,2, ... , λ, w(k)=w'(k).
8. [Move vertex xᵢ,j from level g(j) to g(j)-1.] Perform MOVE-VERTEX(j,i) as follows:
(8.1) g(j)=g(j)-1;
(8.2) w(y)=w(y)+1, w(y+1)=w(y+1)-1;
(8.3) let j successively take the values j+1, ... , cᵣ, 1, ... , j; for the first value j such that g(j)>h(j), set j=0;
(8.4) if there is no such j, set j=0.
9. [Given j, MOVE-VERTEX(j,i) returns the new level y of the moved vertex xᵢ,j, together with the next admissible value of j. The vertex movements take place in a "cascade" from higher-level vertices to lower-level vertices, in accordance with the observation that under these circumstances a vertex xᵢ,j may always be moved to the preceding level, provided that g(j)>h(j) (initially, of course, g(j)>h(j) for every j=1,2, ... , cᵣ). Note that not all possible configurations of Cᵣ are necessarily covered by the movements included here.]
10. [Calculate \Delta T.] \Delta T=\Delta T+[w(y+1)-R-1, 1; otherwise, 0] - (if w(y+1)>R, 1; otherwise 0). The full expression for \Delta T is
T=\min \{R-\lfloor w(y)\rfloor-1,0\} + \min \{R-w(y),0\} - \min \{R-\lfloor w(y+1)\rfloor+1,0\} + \min \{R-w(w(y+1)-1,0)\}.
11. [For increased T, store present arrangement of vertices into levels.] If \Delta T=1, T=T+1, \Delta T=0, and perform STORE-PATTERN; otherwise, goto (12).
12. [STORE-PATTERN does the following: for j=1,2, ... , cᵣ; g(j)=g(j); for k=1,2, ... , λ, w'(k)=w'(k); \Delta T=0. Note that \Delta T' is reset after every increase of T, but that acceptance based on \Delta T' does not reset \Delta T (step (13)).]
13. If \Delta T<0, perform STORE-PATTERN.

* SB propose rather |̄n/λ|.
(14) [More vertex movements possible?] If \( j > 0 \), goto (8).

(15) [More components of "movable" vertices?] \( r \leftarrow r - 1 \), goto (5).

(16) Exit.

The FLS(\( M \)) which best satisfies the \( T \) and \( T' \) criteria has now been determined. Each vertex \( v \) of FLS(\( M \)) is placed into the level specified by \( g(v) \), and the width of each level \( L_k' \) is given by \( w'(k) \).

Figure 4 illustrates the result of the application of the algorithm to an LS with many movable vertices. In this case only three vertices would need to be moved before FLS became acceptable (\( T = 0 \)). In the example of Figure 2, an FLS of width 5 would be accepted (\( T = -3 \)) after movement of two vertices. For the two dimensional grid discussed in detail by GPS, phase II yields an optimal FLS = \( LS(M) \) after step (1), and no vertex movements are required. For the Curtis matrix, on the other hand, consideration of the connected components in decreasing order of size actually inhibits bandwidth reduction: for starting vertex \( u = CD \) and a corresponding \( LS(M) \) of width 11 (\( T = -9 \)), the algorithm requires 56 vertex movements to yield an FLS of width 10 (\( T = -7 \)). Although this result leads to an optimum numbering (\( \delta = 10 \)), much unnecessary work is done. Clearly, in order to evaluate the efficiency and utility of phase II of the SA algorithm, considerable computational experience, especially with large matrices, is desirable.

PHASE III OF SA: NUMBER FLS

We propose here a somewhat more sophisticated numbering algorithm than has previously been employed,

\[ \delta = 2w' - 1 \]

but which retains the property of having execution time approximately linear in \( n \). Indeed, as we shall see, the numbering algorithm has the interesting property of being rather more efficient in the more difficult cases.

Suppose we are given an FLS(\( M \)) of length \( \lambda \) and width \( w' \), with levels \( L_k' \) of width \( w'(k) \), \( k = 1, 2, \ldots, \lambda \). Suppose further that corresponding to each \( V \in V \), we may identify \( g(v) \), the level of \( v \), and \( \{v \mid A(v) \} \), the set of vertices adjacent to \( v \); and that corresponding to each level \( L_k' \) of FLS, we may identify the vertices \( \{x_j \mid L_k' \} \), \( j = 1, 2, \ldots, w'(k) \). Phase III assigns numbers to the vertices on a level-by-level basis, with the objective of arriving at a numbering \( \delta \) of \( G \) which corresponds to a suitably small bandwidth \( \delta \).

In the large, phase III proceeds as follows:

(1) \( \delta \leftarrow 0 \).

(2) \( \delta \leftarrow w' + \Delta \).

(3) For each \( k = 1, 2, \ldots, \lambda \), try to assign numbers to the vertices \( \{x_j \mid L_k' \} \) in such a manner that the number \( n(x) \) of each vertex \( x \) satisfies

\[ |n(x) - n(y)| \leq \delta, \]

for every \( y \in A_v \).

(4) If for some \( k \), it turns out that the vertices cannot be numbered to satisfy this condition, then \( \Delta \leftarrow \Delta + 1 \), goto (2).

We observe that, in principle, this algorithm will terminate sooner or later; indeed, as CM remark, for \( \delta = 2w' - 1 \), any level-by-level numbering of FLS will satisfy the condition given in step (3). We observe further that, in practice, the algorithm will normally terminate at some value \( \delta_0 \) close to \( w' \); in fact, in the great majority of cases, for \( \delta_0 = w' \).

We give now a more detailed description of step (3). Since the same procedure is used for the numbering of each level, we confine ourselves to describing the numbering of the \( k \)th level \( L_k' \). Certain basic values need to be defined and calculated:

\[ W'(k) = \sum_{k=1}^{\lambda} w'(k'), 1 \leq k \leq \lambda; \]

\[ = 0, k = 0. \]

\[ A_{\lambda}(j) = \{v \mid x_j \in L_k \wedge v \in \{x_j \mid A(v) \} \wedge L_{k+1} \}, \]

the set of vertices of \( L_k \) adjacent to the \( j \)th vertex in \( L_k \).

* However an unpublished example due to Arany-Szöda has the following characteristics: \( n = 60, m = 112, w' = 9, \delta(G) = 14 \).
The processing required for each level falls naturally into two stages, which we call INITIALIZE \( L_k' \) and NUMBER \( L_k' \):

**INITIALIZE \( L_k' \)**

1. For each vertex \( X_{kj} \), \( j = 1, 2, \ldots, w'(k) \), calculate
   
   \[ \text{xmin}(j) = \max\{1 + W'(k-1), W'(k) + a_{\alpha} + \text{lown}(j)\} \]
   
   the least number assignable to \( X_{kj} \) which is compatible with \( \alpha \);

2. \[ \text{xrange}(j) = \text{xmax}(j) - \text{xmin}(j) + 1; \]

3. [If \( \text{xrange}(j) \leq 0 \), go to INCREMENT (step (4) in the general outline of phase III given above).]

**NUMBER \( L_k' \)**

1. \( J = 1 \), \( \text{counter} \leftarrow 0 \).
2. If \( \text{nrangeclass}(J) = 0 \), \( J \leftarrow J + 1 \), goto (5).
3. [The numbers to be assigned are chosen in increasing order of \( \text{nrangeclass}(J) \).]
4. \( j' = \text{nrangeclass}(\text{nr}(j')) \), \( \text{nrangeclass}(\text{nr}(j')) \leftarrow j' \).
5. [\( j' \) is a pointer to the next number in \( \text{nrangeclass}(\text{nr}(j')) \).]
6. \( \text{FLS of} \) width 10 determined by applying phase II to Curtis matrix \( L' = \{52, 53, 54\} \)

**CONCLUSIONS**

We have described a bandwidth reduction algorithm which appears to be competitive in effectiveness and efficiency with presently known algorithms. Systematic testing on large matrices encountered in practice is re-
quired. Some of the more important questions remaining to be clarified, either by experiment or by analysis, are as follows:

1. The execution time of SB (phase I of SA) is proportional to $2Hm$. What can be said about the magnitude of $H$?

2. In what (more efficient) way can vertex movements be evaluated during phase II, in order to yield an optimum FLS?

3. Given two FLS of $G$ of widths $w_1$ and $w_2 > w_1$, denote by $\delta_1$ and $\delta_2$ the least bandwidths obtainable by level-by-level numbering of the first and second FLS, respectively. Does it follow that $\delta_2 > \delta_1$?

REFERENCES


