majority of the elements are zero: these arise in many problems such as engineering structural analysis.(10) Storing sparse matrices in the usual way is wasteful; one solution is to store only the non-zero elements, for example by using hashing methods.(6) This leads to special numerical techniques.(10) A preferred solution would be to reduce the matrix by row and column permutations to a band matrix (i.e. a matrix in which all non-zero elements lie in some band close to the diagonal), for which only the band elements need be stored: the advantage is then that standard and efficient numerical techniques are applicable. The reduction to a band matrix is normally carried out by ad hoc methods, by skilled humans prior to computer runs: what is desired is a guaranteeable method for performing this reduction. The branch-and-bound algorithm proved ideal.

For an n by n matrix, there are $(n!)^2$ possible configurations, and exhaustive searches are untenable. The technique used was to partition by row or column selection. Assuming for the moment that the lower bound function g is available (this will be outlined later), then the first partition of the totality of possible configurations is to consider the n possible selections of particular columns of the original matrix for the first position in the reduced matrix. Among these n subsets, the one with smallest lower bound is selected, and the n possible selections of rows for the first position of the reduced matrix considered. Among the resulting 2n-1 subsets that with minimal lower bound is selected for row or column partition as appropriate, and so on, for a minimum of 2n partitions.

The lower bound function computes a best possible bandwidth for a partially determined matrix (with the first m columns fixed and the first m or m-1 rows fixed) as the maximum of the following-
(i) for the fixed portion at top left; count from the diagonal of the fixed portion along the rows and columns to locate the last non-zero.element; within the fixed portion (m by m or m by m-1) the elements are fixed, while beyond the fixed portion it is assumed that the non-zero elements could be positioned immediately following the fixed columns or rows. Repeat this for each row and column within the fixed portion, taking the maximum count found.
(ii) for the unfixed portion at bottom right; count the number of elements in each row and column and assume that these could be placed symmetrically about the diagonal - hence bound to bandwidth here is maximum of [(count + 1)/2] where [I] means the integer part of I.

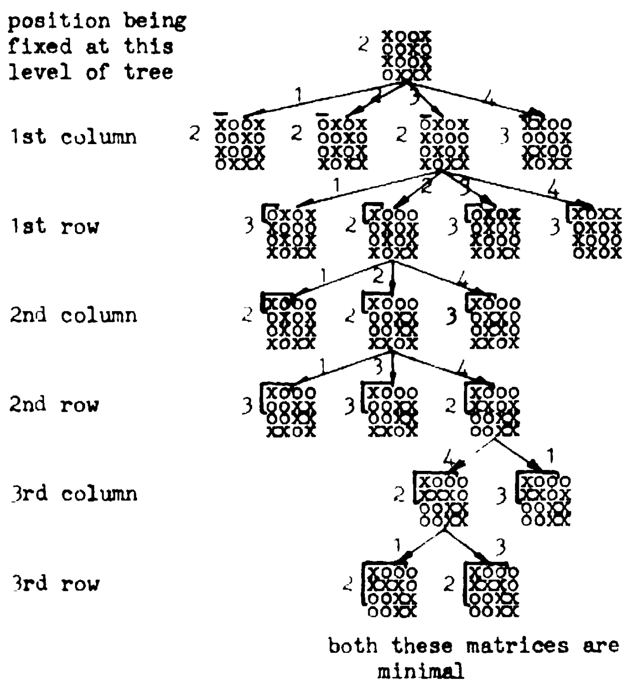
Figure 1 illustrates the method for a 4 by 4 matrix of bandwidth 2. Figure 1(a) shows the original matrix and Figure 1(b) illustrates in tree form the complete search. Each node is labelled with the value of the lower bound function g, and the matrix to data; the elements whose positions are fixed appear in the upper left-hand submatrix marked with thick lines. The search for the next node to develop selects, when more than one node has the minimal value, that node nearest the bottom of the tree. Each level of the tree is labelled to show what development is being made, which row or column is being fixed: each branch of the tree is labelled to indicate which row or column in the original matrix is the one that is being fixed in the position indicate for the whole level.

FIGURE 1. The application of Branch-and-bound to reducing a sparse matrix to its minimal band matrix form.
(a) the original matrix
(b) the search tree: branches are labelled with the row or column of the original being fixed in the position shown at the left; nodes are labelled with the bound, and the fixed portion in the upper left marked off.

1a)

```
X O O X
O O X O
X O O X
O X X X
```

1b)


position being fixed at this level of tree
1st column
1st row
2nd column
2nd row
3rd column
3rd row
both these matrices are minimal