ON PROFILE MINIMIZATION OF TREE STRUCTURES

by

Joseph W.H. Liu

Research Report CS-75-13

Department of Computer Science

University of Waterloo
Waterloo, Ontario, Canada

April 1975

Research was supported in part by an Ontario Graduate Scholarship and in part by Canadian National Research Council Grant A8111.

ON PROFILE MINIMIZATION OF TREE STRUCTURES[†]

by

Joseph W.H. Liu

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

[†] Research supported in part by an Ontario Graduate Scholarship and in part by Canadian National Research Council Grant A8111.

Abstract

A graph-theoretic approach is used to study the envelope method for the direct solution of sparse linear systems. In this context, we introduce the minimum and minimal envelope problems. An $O(N \log_2 N)$ algorithm, which generates minimal envelope orderings for trees of size N, is presented. The corresponding profile or envelope size is shown to be bounded by $2N + N \log_2 N$. Some extensive test results of this algorithm on randomly generated trees are also included.

§1 Introduction

The direct solution of a given N by N symmetric positive definite linear system

$$Ax = b \tag{1.1}$$

involves two major steps: the symmetric factorization of A and the back substitution for x. In the factorization step (or the elimination step), the matrix A is decomposed into the product LDL^T, where L is unit lower triangular and D is a positive diagonal matrix. The solution vector x can then be obtained by performing the forward and backward substitutions:

$$Lz = b, \quad Dy = z, \quad L^Tx = y. \tag{1.2}$$

It is essentially the well-known Cholesky method which has remarkable numerical stability for positive definite systems [28].

When the matrix A is large and sparse, a significant reduction in storage and computation can be obtained if sufficient care is taken to avoid storing and operating on all or some of the zeros. With the assumption that exact numerical cancellation does not occur during the factorization, the matrix A generally "fills-in" so that $L+L^T$ usually has nonzeros in positions which are zero in A. In other words, $L+L^T$ is usually fuller than A.

Consider any permutation matrix P. The solution for (1.1) can be obtained by solving the equivalent system:

$$PAP^T(Px) = Pb. \tag{1.3}$$

Since the permuted system remains sparse, symmetric and positive definite, the above scheme applies equally well to (1.3). Yet, the permuted matrix PAP^T in general fills-in differently, and the amount of fill may be drastically different from that of A. In solving the permuted system, we gain the possible advantage of reduced computation and storage requirements.

In [22], Rose used a graph-theoretic approach to study systematically the effect of the elimination order upon the set of fill (and hence the amount of fill). His analysis established a theoretical foundation on optimal orderings with respect to some criterion functions. In particular, he defined the minimum and minimal triangulation (or fill) problems, the purpose of which is to minimize the number of fill [21,22,23].

In general, for fill-reducing orderings, the correspondingly permuted matrix has its nonzeros spread over the entire matrix. General sparse techniques, which exploit all the zeros in PAP^T or its triangular factor, have to be used. Their implementations usually require special storage methods such as linear lists or bit vectors [12,20,25], and relatively complicated coding.

For purposes of implementation, it is often convenient to exploit only those zeros outside a particular region of the matrix. The envelope method is one such scheme [4,8,10,13,14]. Let M be a symmetric matrix with $M_{ii} \neq 0$, and $f_i(M) = \min\{j | M_{ij} \neq 0\}$. The envelope of M is defined as:

$$\mathrm{Env}(M) = \{(i,j) | f_i(M) \leq j \text{ and } f_j(M) \leq i\}. \tag{1.4}$$

The envelope method exploits zeros outside Env(M). In the context of symmetric factorization, this scheme is attractive because $\mathrm{Env}(A) = \mathrm{Env}(L+L^T)$, so that fills are confined to this envelope region [27]. Furthermore, it requires no more storage than the popular band method [5,6,10] and in most cases significantly less. And yet, it retains the advantages of band schemes: simple data structure [13], efficient data management and convenient coding [4].

To reduce storage and computation in envelope methods, it is desirable to produce orderings PAP^T for which $|Env(PAP^T)|$ is small. The quantity $|Env(PAP^T)|$ is usually referred to as the envelope size or the profile of PAP^T. There are heuristic algorithms that produce profile-reducing orderings: Cuthill-McKee [6], George [7], King [14], Gibbs, Poole and Stockmeyer [11]. A comparative study of these heuristics can be found in [4,12].

In this paper, we study the envelope method from a combinatorial point of view. Parallel to Rose's treatment of the fill problem, we formulate the minimum and minimal envelope problems in §2. Finding minimum or even minimal envelope orderings is in general a difficult task. As a start, we consider minimal envelope orderings on linear systems associated with tree structures.

In §4, we discuss the postordering of rooted trees (Knuth [15]) which turns out to be an effective algorithm for reducing the profiles of trees. The postordering is then modified in §5 to a new algorithm (called MET) that always generates a minimal envelope for trees. It is shown in §6 that this algorithm has a time bound of $O(N \log_2 N)$, where N is the number of nodes in the tree. We also show that the resulting profile is always bounded by $2N + N \log_2 N$. The algorithm is implemented in ALGOL W, and in §7 we include some results of this program on random trees of different sizes.

§2 Preliminaries

In this section we introduce some definitions, notations, and simple results that are useful in subsequent sections.

We begin with some standard graph-theoretic terminology. A graph $G = (X,E)$ consists of a finite set X of nodes together with a set E of unordered pairs of nodes called edges. A graph $G' = (X',E')$ is a subgraph of $G = (X,E)$ if $X' \subset X$ and $E' \subset E$.

Nodes x and y are adjacent if $\{x,y\} \in E$. For a subset $Y \subset X$, the adjacent set of Y is defined as

$$\mathrm{Adj}(Y) = \{x \in X \setminus Y \mid \{y,x\} \in E \text{ for some } y \in Y\}. \qquad (2.1)$$

In case $Y = \{y\}$, we shall use $\mathrm{Adj}(y)$ instead of $\mathrm{Adj}(\{y\})$. A clique is a subgraph whose nodes are pairwise adjacent.

A path of length ℓ is a sequence of ℓ edges $\{x_0,x_1\},\{x_1,x_2\},\ldots,\{x_{\ell-1}, x_\ell\}$, where all the nodes on the path, except possibly x_0 and x_ℓ, are distinct. A cycle is a path of length at least one, which begins and ends at the same node. A graph G is connected if for each pair of distinct nodes, there is a path connecting them. If G is not connected, it consists of two or more maximal connected subgraphs called components.

In this paper, graphs are assumed to be connected, unless otherwise specified. A separator S of a graph is a subset of X whose removal disconnects the graph. A separator is minimal if no subset of it is a separator. If $\{x\}$ is a separator, then x is a cutnode. Clearly, cutnodes are minimal separators.

The distance $d(x,y)$ between the nodes x and y is the length of a shortest path connecting them. Following Berge [2], we define the diameter $\delta(G)$ of a

graph $G = (X,E)$ to be the quantity $\max\{d(x,y) | x,y \in X\}$; and we call x and y peripherial nodes if $d(x,y) = \delta(G)$.

In this paper, we are primarily concerned with a class of connected graphs called trees. A tree is a connected graph with no cycles. It can be easily verified that for a tree $T = (X,E)$, $|X| = |E| + 1$, and every pair of distinct nodes in T is connected by exactly one path (Berge [2]). We denote a rooted tree by (R,T), where R is a distinguished node (the root) in the tree T. If the path from the root R to y passes through x, then x is an ancestor of y and y is a descendant of x. If, in addition, $\{x,y\} \in E$, x is called the father of y and y is a son of x. A node x with all its descendants are called a subtree of the rooted tree. Note that the ancestor-descendant and the father-son relationships are not defined in unrooted trees.

Consider a graph $G = (X,E)$ with $|X| = N$. An ordering or numbering of G is any bijective mapping $\alpha = \{1,2,\ldots,N\} \to X$. We denote the ordered graph and node set by G_α and X_α respectively.

To study the envelope method graph-theoretically, we associate an undirected graph $G(M) = (X(M),E(M))$ with a symmetric matrix M. Here X(M) is the set of nodes corresponding to and labelled as the rows of M, and E(M) is the set of edges, where $\{x_i,x_j\} \in E(M)$ if and only if $M_{ij} \neq 0$ and $i \neq j$. The graph G(M) has an implicit ordering defined by the matrix M. Indeed, there is a one-to-one correspondence between orderings on G(M) and permutations on M. Each α defines a permutation matrix P_α where

$$G(A)_\alpha = G(P_\alpha A P_\alpha^T). \tag{2.2}$$

In the symmetric factorization of the matrix A in (1.1), we can describe the set of fill in terms of the associated graphs $G(A) = (X(A),E(A))$, and $G(L+L^T) = (X(L+L^T),E(L+L^T))$, where $A = LDL^T$, as follows:

$$Fill(A) = E(L+L^T) \setminus E(A). \tag{2.3}$$

Rose called Fill(A) the triangulation set for A since these edges when added to E(A) transforms G(A) into a triangulated graph† $G(L+L^T)$. For details, see Rose [21]. Using the notion of triangulation set, Rose then formulated the minimal and minimum triangulation problems. The former requires the finding of a permutation P such that for any other Q,

$$Fill(QAQ^T) \subset Fill(PAP^T) \Rightarrow Fill(QAQ^T) = Fill(PAP^T). \tag{2.4}$$

The second problem involves the finding of a P with a minimum set of fills, that is, for any Q,

$$|Fill(QAQ^T)| \leq |Fill(PAP^T)| \Rightarrow |Fill(QAQ^T)| = |Fill(PAP^T)|. \tag{2.5}$$

For efficient algorithms to determine minimal fill orderings, see Ohtsuki [17], Rose and Tarjan [24].

To establish the equivalent problems in envelope methods, we introduce the set of potential fill. Let $G_\alpha = (X_\alpha,E)$ be a graph ordered by α. For $i = 1,\ldots,N$, the set $Adj\{\alpha(1),\ldots,\alpha(i)\}$ shall be designated as the i-th front of G_α; its size $|Adj\{\alpha(1),\ldots,\alpha(i)\}|$ is called the i-th frontwidth of G_α. We then define the potential fills of G_α [9] as:

$$Pfill(G_\alpha) = \bigcup_i \{\{\alpha(j),\alpha(i)\} | \alpha(j) \in Adj\{\alpha(1),\ldots,\alpha(i)\}\} \setminus E. \tag{2.6}$$

† A graph is triangulated if every cycle in it has a chord (Berge [3]).

It is important to note that, in terms of the permuted matrix $P_\alpha A P_\alpha^T$, $Pfill(G(A)_\alpha)$ contains locations of zero entries in the envelope of $P_\alpha A P_\alpha^T$. The significance of the front and frontwidth concepts in envelope methods can be justified by the following lemma (George [8]).

Lemma 2.1 Let α be an ordering on $G(A)$. Let $\omega_i = |Adj\{\alpha(1),\ldots,\alpha(i)\}|$. Then

(i) $$|Env(P_\alpha A P_\alpha^T)| = 2\left(\sum_{i=1}^{N} \omega_i\right) + N.$$

(ii) If $\theta(P_\alpha^T A P_\alpha^T)$ is the number of multiplicative operation to factor $P_\alpha A P_\alpha^T$ without exploiting zeros within the envelope,
$$\theta(P_\alpha A P_\alpha^T) = \frac{1}{2} \sum_{i=1}^{N} \omega_i(\omega_i+3).$$

Let α be an ordering on G. Then α is a <u>full envelope ordering</u> if $Pfill(G_\alpha) = \phi$. We call α a <u>minimal envelope ordering</u> if no other ordering β satisfies

$$Pfill(G_\beta) \subsetneq Pfill(G_\alpha).$$

The mapping α is a <u>minimum envelope ordering</u> if

$$|Pfill(G_\alpha)| \leq |Pfill(G_\beta)|$$

for every ordering β.

Clearly, any full envelope ordering gives a minimum envelope; and any minimum envelope ordering is minimal.

The ordering in Figure 2.1 generates a full envelope. In Figure 2.2, the ordering is a minimum envelope ordering with $\{\{5,3\}\}$ as the set of potential fill. Figure 2.3 is an example of a minimal envelope ordering. Here the set

of potential fill is {{6,3},{6,4}}; however, we can find a different ordering with |Pfill| = 1.

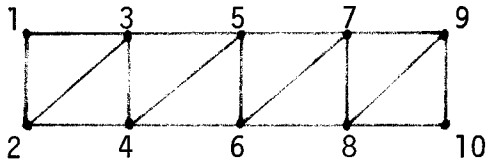


Figure 2.1 A full envelope ordering

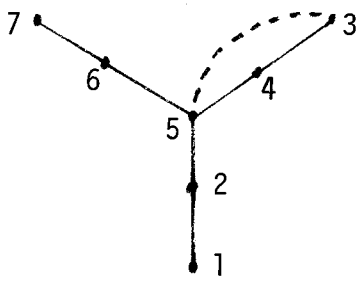


Figure 2.2 A minimum envelope ordering

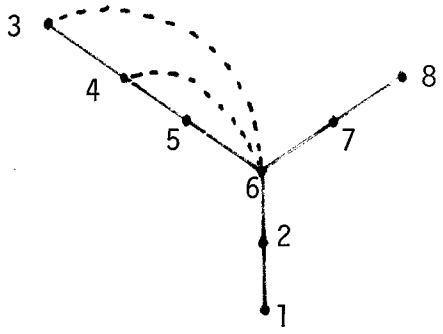


Figure 2.3 A minimal envelope ordering

§3 On full envelope orderings

Except in some rather special cases, a graph does not possess full envelope orderings. We feel that it is still worthwhile to study them for two reasons. Firstly, if α is an ordering on a graph $G = (X,E)$, then α will be a full envelope ordering on the super graph

$$\bar{G} = (X, E \cup Pfill(G_\alpha)).$$

Secondly, they do bear some significance in the study of minimal envelope orderings. If β is a minimal envelope ordering on G, then for any nonempty $H \subset Pfill(G_\beta)$, the super graph

$$\bar{G}_H = (X, E \cup Pfill(G_\beta) \setminus H)$$

does not have any full envelope ordering.

To begin, we give a straightforward characterization of full envelope orderings using the definition (2.6).

Lemma 3.1 The labelling α is a full envelope ordering if and only if $Adj\{\alpha(1),\ldots,\alpha(i)\} \subset Adj(\alpha(i))$, $i = 1,\ldots,N$.

Another characterization in terms of a sequence of cliques is given in the following lemma.

Lemma 3.2 α is a full envelope ordering on G if and only if for every $i = 1,\ldots,N$. $Adj\{\alpha(1),\ldots,\alpha(i)\} \cup \{\alpha(i)\}$ is a clique.

Proof "if part". By lemma 3.1, it is sufficient to show

$$Adj\{\alpha(1),\ldots,\alpha(i)\} \subset Adj(\alpha(i)).$$

Consider any $y \in Adj\{\alpha(1),\ldots,\alpha(i-1)\} \setminus \{\alpha(i)\}$. Since $Adj\{\alpha(1),\ldots,\alpha(i-1)\} \cup \{\alpha(i)\}$ is a clique, y belongs to $Adj(\alpha(i))$.

"only if part". We first show that for $i = 1,\ldots,N$, $\text{Adj}\{\alpha(1),\ldots,\alpha(i)\}$ is a clique. Assume for contradiction that for some i, $\text{Adj}\{\alpha(1),\ldots,\alpha(i)\}$ is not a clique. Then, there exist

$$\alpha(j),\alpha(k) \in \text{Adj}\{\alpha(1),\ldots,\alpha(i)\}$$

such that $\alpha(j)$ and $\alpha(k)$ are not adjacent. For definiteness, let $j > k$. Then

$$\alpha(j) \in \text{Adj}\{\alpha(1),\ldots,\alpha(i),\ldots,\alpha(k)\}$$

and yet $\alpha(j) \notin \text{Adj}(\alpha(k))$.

To complete the proof, we show that $\text{Adj}\{\alpha(k),\ldots,\alpha(i)\} \cup \{\alpha(i)\}$ is a clique. If it is not a clique, since $\text{Adj}\{\alpha(1),\ldots,\alpha(i)\}$ is, we can always find an $x \in \text{Adj}\{\alpha(1),\ldots,\alpha(i)\}$ such that $x \notin \text{Adj}(\alpha(i))$. Contradiction. □

Lemma 3.1 and Lemma 3.2 are characterizations of full envelope orderings on the ordered graph G_α. We can only provide a partial characterization on the unordered graph in terms of separator cliques.

Let S be a minimal separator with m components $C_i = (X_i,E_i)$, $i = 1,\ldots,m$. The following two lemmas are immediate.

Lemma 3.3 For $i = 1,\ldots,m$, $\text{Adj}(Y) \subset S \cup X_i$, $\forall Y \subset X_i$.

Lemma 3.4 For $i = 1,\ldots,m$, $X_i \cap \text{Adj}(x) \neq \phi$, $\forall x \in S$.

A full envelope ordering resequences the nodes in G according to some pattern with respect to the components of minimal separators. It is given in the following lemma.

Lemma 3.5 Let S be a minimal separator of G with m components $C_i = (X_i,E_i)$. If α is a full envelope ordering on G, then there exists a permutation σ on the components so that α orders the nodes in the sequence:

$$X_{\sigma(1)},X_{\sigma(2)},\ldots,X_{\sigma(m-1)}, S \cup X_{\sigma(m)}.$$

Proof Let m = 2. We note that nodes in S cannot be numbered until there is only one component left unnumbered, say $C_{\sigma(2)}$, because of Lemma 3.4.

Let $\alpha(1) \in X_{\sigma(1)}$. If $n_1 = |C_{\sigma(1)}|$, we show that $\alpha(2),\ldots,\alpha(n_1) \in X_{\sigma(1)}$. Assume that $\alpha(1),\ldots,\alpha(i-1) \in X_{\sigma(1)}$ where $i \leq n_1$. By Lemma 3.2, $\alpha(i)$ has to be chosen so that

$$\text{Adj}\{\alpha(1),\ldots,\alpha(i-1)\} \cup \{\alpha(i)\}$$

is a clique. Together with Lemma 3.3,

$$\alpha(i) \in \text{Adj}\{\alpha(1),\ldots,\alpha(i-1)\} \subset S \cup X_{\sigma(1)}.$$

Thus $\alpha(i) \in X_{\sigma(1)}$. Hence α orders the two components in some sequence σ

$$X_{\sigma(1)}, S \cup X_{\sigma(2)}.$$

The proof can then be completed by induction on m, the number of components. □

Theorem 3.1 Let G be a graph that has full envelope orderings. For every minimal separator S of G with components $C_i = (X_i, E_i)$, $i = 1,\ldots,m$, the set $S \cup \{x\}$ is a clique for every $x \in X_i$ for all except possibly two components.

Proof Let α be a full envelope ordering on G. Lemma 3.5 assures that α orders the nodes in some component sequence:

$$C_{\sigma(1)}, C_{\sigma(2)}, \ldots, C_{\sigma(m-1)}, S \cup C_{\sigma(m)}.$$

Consider any $x \in C_{\sigma(2)} \cup \ldots \cup C_{\sigma(m-1)}$. Let $\alpha(k) = x \in C_{\sigma(i)}$.

By Lemma 3.3, $\text{Adj}(C_{\sigma(1)} \cup \ldots \cup C_{\sigma(i-1)}) = S$, so that $S \subset \text{Adj}\{\alpha(1),\ldots,\alpha(k-1)\}$. The result of Lemma 3.2 implies that $S \cup \{x\}$ is a clique. □

We shall find theorem 3.1 useful in §5 when an envelope-minimizing algorithm for trees is analysed. We remark that the above theorem cannot be extended to minimal u-v separators (Rose [22]). An u-v separator is a separator S such that nodes u and v belong to different components with respect to S.

In the example of Figure 3.1, {3,4,7,8} is a minimal 2-5 separator, and the corresponding components are {1,2}, {5}, {6}, {9}. The result of Theorem 3.1 does not hold for {3,4,7,8}.

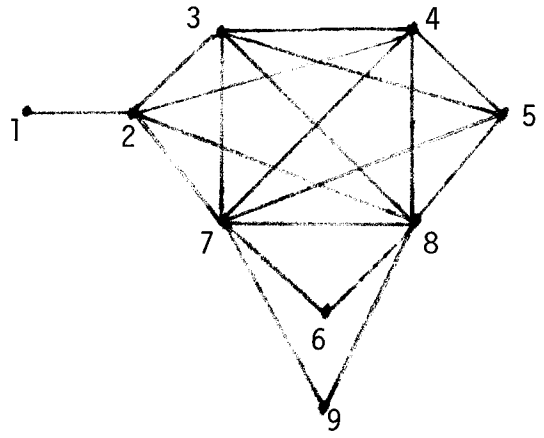


Figure 3.1

In the proof of Theorem 3.1, it is important to realize that the components $C_{\sigma(2)},\ldots,C_{\sigma(m-1)}$, when considered as independent graphs, have full envelope orderings. Thus, the result in Theorem 3.1 is also applicable to these components.

§4 Postorder of trees

Ordering of tree structures associated with symmetric linear systems was first studied by Parter [19]. He showed that any tree can be ordered so that its fill is empty.

However, not all trees have full envelope orderings. As a matter of fact, only those trees with a main stem and branches of length one can be ordered with full envelopes. This can be readily proved by Theorem 3.1. Figure 4.1 is an example of such trees.

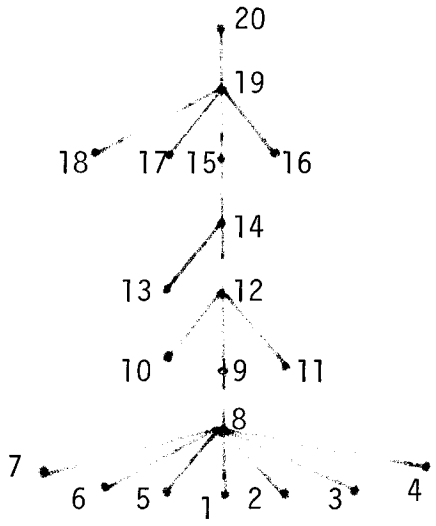


Figure 4.1 A tree with full envelope orderings

A thorough study of tree ordering has been given by Knuth [15,p.315-359], in the context of tree traversals. Ref.[1] also contains a comprehensive treatment of tree traversals. In a complete traversal of a given tree, each node is visited exactly once so that it induces a linear arrangement of the nodes. It is apparent that the postorder traversals of rooted trees is most appropriate for profile minimization.

Let us first review the postorder traversal of a rooted tree. Consider a rooted tree (R,T). Let the rooted subtrees under R be $(s_i(R), T_i(R))$

$i = 1,\ldots,m$. Here $s_1(R),\ldots,s_m(R)$ are the sons of R arranged in some specific order.

The postorder traversal of (R,T) is a systematic visit to the nodes of T using the following algorithm:

Algorithm

```
POSTORDER (R,T):
Comment Let (s_i(R),T_i(R)) i = 1,...,m be the rooted subtrees under
R in (R,T);
For i := 1 until m do
        POSTORDER (s_i(R),T_i(R));
Visit node R;
end;
```

If we number the nodes of the tree in the same order as the sequence of node "visits", we obtain an ordering which generally yields a small envelope. In the example of Figure 4.2 (Knuth [15,p.363]), if we choose node I to be the root and the arrangements of the subtrees as shown, the postordering α on T will then be:

J, N, L, K, E, F, A, B, D, C, G, M, H, I.

Here the set of potential fill is:

$$Pfill(T_\alpha) = \{\{G,A\},\{G,B\},\{G,D\}\};$$

and they are denoted by dotted lines in the figure.

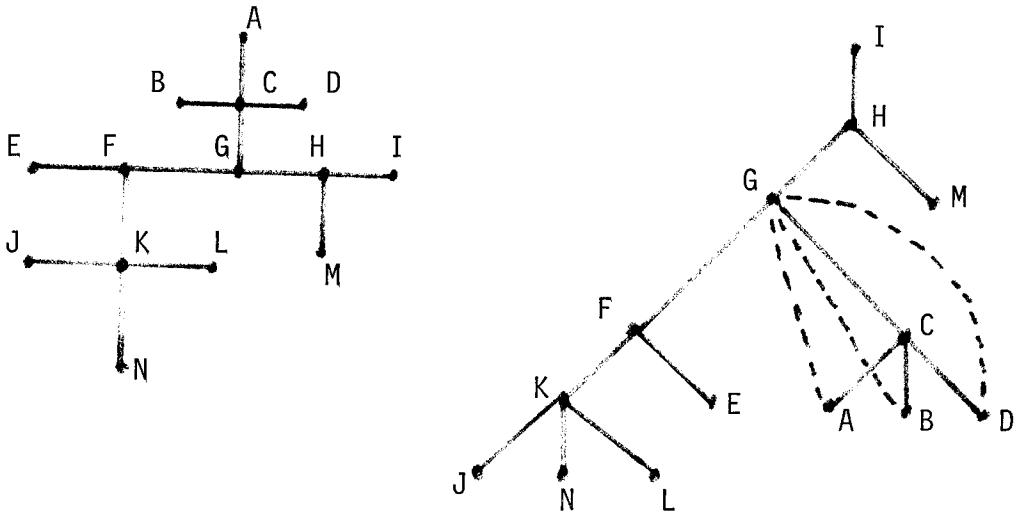


Figure 4.2 A postordering on a rooted tree

Evidently, the envelope size depends on the order of the subtrees $(s_1(R),T_1(R)),\ldots,(s_m(R),T_m(R))$. To see how this arrangement will affect the envelope size, we proceed to characterize $Pfill(T_\alpha)$, where α is the postordering on T with a given subtree arrangement.

Lemma 4.1 Let u,v be nodes in the rooted tree (R,T). Then $\alpha^{-1}(u) < \alpha^{-1}(v)$ if and only if

either $u \in T_k(v)$ for some k

or $u \in T_i(z)$ and $v \in T_j(z)$ where $i < j$ for some node z.

Proof It follows directly from the ordering algorithm. □

Lemma 4.2 Let $\alpha^{-1}(w) < \alpha^{-1}(u) < \alpha^{-1}(v)$. If w is a descendent of v, so is the node u.

Proof Assume for contradiction that u is not a descendant of the node v. By Lemma 4.1, $\alpha^{-1}(u) < \alpha^{-1}(v)$ implies the existence of some node z such that $u \in T_i(z)$ and $v \in T_j(z)$, where $i < j$. Since w is a descendant of v, $w \in T_j(z)$. Yet, this would imply, again by Lemma 4.1, that $\alpha^{-1}(u) < \alpha^{-1}(w)$. □

A node s is said to be a younger son of x if $s = s_i(x)$ for some $i = 2,\ldots,m$.

Theorem 4.1 Let $\alpha^{-1}(u) < \alpha^{-1}(v)$. Then $\{v,u\} \in Pfill(T_\alpha)$ if and only if u is a descendant of v's younger sons.

Proof Let $T_1(v),T_2(v),\ldots,T_m(v)$ be the subtrees under v. We shall prove the equivalent statement:

$$\{v,u\} \in Pfill(T_\alpha) \text{ iff } u \in \bigcup_{i=2}^{m} T_i(v) \setminus Adj(v).$$

Let $u \in \bigcup_{i=2}^{m} T_i(v) \setminus Adj(v)$. Since $Adj(T_1(v)) = \{v\}$ and $\alpha^{-1}(u) < \alpha^{-1}(v)$, $v \in Adj\{\alpha(1),\ldots,u\} \setminus Adj(u)$. So, by Corollary 2.4, $\{v,u\} \in Pfill(T_\alpha)$.

On the other hand, assume that $\{v,u\} \in Pfill(T_\alpha)$. Now that $v \in Adj\{\alpha(1),\ldots,u\} \setminus Adj(u)$, there exists some $k < \alpha^{-1}(u)$ so that $v \in Adj(\alpha(k))$. Thus, $\alpha(k)$ is a son of v. Together with the relation

$$\alpha^{-1}(\alpha(k)) < \alpha^{-1}(u) < \alpha^{-1}(v),$$

we conclude by Lemma 4.2 that $u \in T_i(v)$. Finally, i cannot be one because $\alpha(k)$ is numbered before u and $\alpha(k)$ is a son of v. □

For a rooted tree (R,T) with $T_1(R),\ldots,T_m(R)$ as subtrees under R, it follows from theorem 4.1 that the number of potential fills due to the node R is given by:

$$\sum_{i=2}^{m} |T_i(R)| - m+1.$$

In the context of profile minimization, the best way to arrange the subtrees becomes immediate: always pick the one with the greatest number of nodes as the first subtree. This will ensure a minimal set of potential-fill in using the postorder algorithm.

To make the selection possible, we have to know, for each node x in the rooted tree (R,T), the number D(x) of descendants x has in T. Knuth [15,p.351] introduces a locally-defined function in a tree as a function of the nodes such that the functional value at a node depends only on the node and the functional values of its sons. It is evident that D(x) is a locally-defined function:

$$D(x) = D(s_1) + D(s_2) + \ldots + D(s_m) + m$$

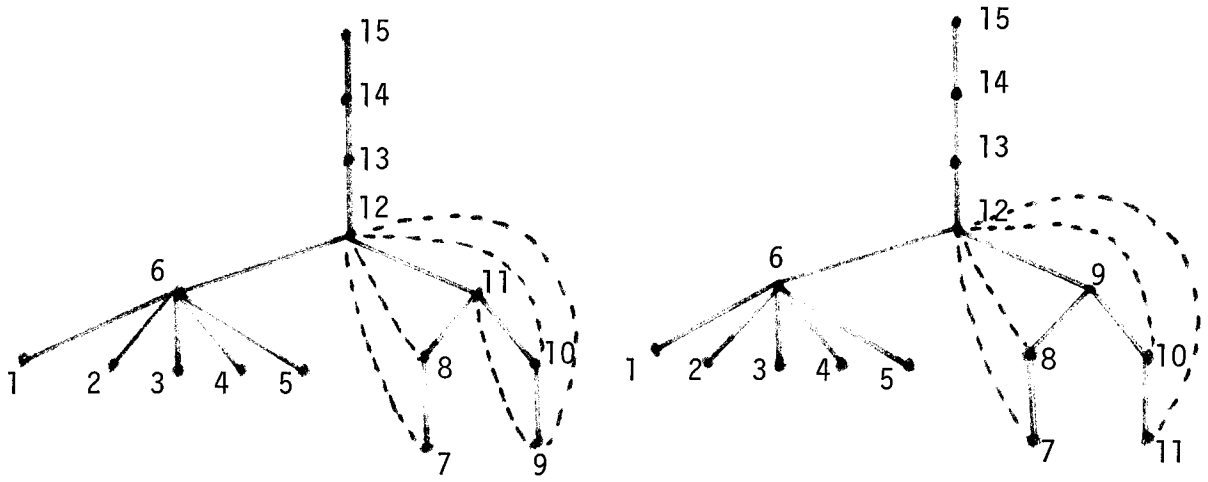
where $s_1, s_2, \ldots, s_m$ are the sons of x in T. Locally-defined functions can be computed very easily, provided that the data representation allows an efficient retrieval of adjacent sets (e.g. adjacency structure [26]). If there are N nodes in the rooted tree, D(x) can be determined in O(N) edge inspections.

The algorithm POSTORDER is only applicable to rooted trees. Thus we are left with the problem of finding an appropriate root for a given tree to start the algorithm. An apparently good one is a peripherial node. This choice is crucial in the new algorithm described in the next section. To determine a peripherial node for trees requires at most O(N) number of edge inspections and we shall discuss this in more detail in §6.

§5 A minimal envelope ordering for trees

The postorder scheme in §4 is a simple, efficient and yet quite effective algorithm for minimizing profiles of tree structures. Unfortunately, it does not guarantee a minimal envelope. In the example of Figure 5.1, α is a postordering. But we can find another ordering β such that

$$Pfill(T_\beta) \subsetneq Pfill(T_\alpha).$$



A postordering α A better ordering β

Figure 5.1 Postordering α is not minimal

In this section, we modify the postorder scheme to a new algorithm that will always generate a minimal envelope ordering. Let us first study the above example more carefully and relate it to Theorem 3.1. The node $\{\alpha(12)\}$ forms a minimal separator in the tree $T = (X,E)$ and also in the extension graph $(X, E \cup Pfill(T_\alpha))$. There are three components with respect to $\{\alpha(12)\}$:

$$C_1 = \{\alpha(1),\alpha(2),\alpha(3),\alpha(4),\alpha(5),\alpha(6)\}$$

$$C_2 = \{\alpha(7),\alpha(8),\alpha(9),\alpha(10),\alpha(11)\}$$

and $$C_3 = \{\alpha(13),\alpha(14),\alpha(15)\}.$$

Theorem 3.1 says that we cannot remove any dotted edges $\{\alpha(12),x\}$ where $x \in C_2$, in the set of potential fills.

But if we consider the component C_2, it is clear that C_2 has a full envelope ordering so that $\{\alpha(11),\alpha(9)\}$ in $Pfill(T_\alpha)$ can be removed. This suggests that in the postorder scheme for (R,T) with subtrees $T_1(R),\ldots,T_m(R)$ under R, we should treat $T_2(R),\ldots T_m(R)$ as general subtrees rather than as rooted subtrees. In the following, we describe our modified algorithm MET (minimal envelope ordering for trees) using two recursive procedures.

Algorithm:

```
MET(T):
  Comment MET generates a minimal envelope ordering for a
     given tree T;
     Find a peripherial node R of T.
  PORDER (R,T);
end;
PORDER (R,T):
  Comment PORDER numbers the tree T rooted at R in a postorder-
     like sequence.  T1(R),...,Tm(R) are the subtrees under the
     root R;
  If m > 0 then
     Begin
        Resequence the subtrees so that |T1(R)| ≥ |Ti(R)|;
        Comment Let s1(R) be the first son of R;
        PORDER(s1(R),T1(R));
        For i := 2 until m do  MET(Ti(R));
     end;
  Number node R;
end;
```

For a given tree, there is a class of orderings that can be produced by the algorithm MET. The better ordering in Figure 5.1 belongs to this class. For convenience, we denote by β an ordering obtained by MET. We prove through a sequence of lemmas that β is a minimal envelope ordering.

Lemma 5.1 If R is a peripherial node in a tree, then $|Adj(R)| = 1$.

Consider the tree T rooted at R. Let x_R be the first descendant of R that has more than one son, and $T_1(x_R),\ldots,T_m(x_R)$ be the subtrees under x_R. By the definition of x_R, we have $m \geq 2$. Here the first subtree is assumed to have the greatest number of nodes, that is, $|T_1(x_R)| \geq |T_i(x_R)|$ for $i = 1,\ldots,m$. Note that $x_R \neq R$ in view of Lemma 5.1.

Lemma 5.2 If $x_R \in Adj(R)$, then

$$|T_i(x_R)| = 1 \quad \text{for } i = 2,\ldots,m.$$

Proof If for some $i \geq 2$, $|T_i(x_R)| \geq 2$, R cannot be a peripherial node. □

Lemma 5.3 Let $\beta^{-1}(u) < \beta^{-1}(x_R)$. $\{x_R,u\} \in Pfill(T_\beta)$ if and only if u is a descendant of x_R's younger sons.

Proof The same as Theorem 4.1. □

Corollary 5.4 x_R is a cutnode of the extension graph $\bar{G} = (X, E \cup Pfill(T_\beta))$ of T.

Proof By Lemma 5.3, the removal of x_R disconnects $\bar{G}$ into (m+1) components:

$$T_1(x_R),\ T_2(x_R),\ldots,\ T_m(x_R),\ \text{Ancestor}\ (x_R),$$

where $Ancestor(x_R)$ is the set of ancestors of x_R. □

Theorem 5.1 β is a minimal envelope ordering.

Proof Let $\bar{G}$ be the extension graph $(X, E \cup Pfill(T_\beta))$ of the tree $T = (X,E)$. If R is the peripherial node used in the algorithm, we consider the removal of some $\{x_R,y\}$ in $Pfill(T_\beta)$. By Lemma 5.3, $y \in T_i(x_R) \setminus Adj(x_R)$ for some $i > 1$. Thus $|T_i(x_R)| > 1$, so that $|T_1(x_R)| > 1$ and by Lemma 5.2, $R \notin Adj(x_R)$.

This means that x_R is a cutnode of $\bar{G}$ with components $T_1(x_R),\ldots,T_m(x_R)$, $Ancestor(x_R)$, where $T_1(x_R) \notin Adj(x_R)$ and $Ancestor\ (x_R) \notin Adj(x_R)$. Hence, by Theorem 3.1, the removal of some $\{x_R,y\}$ in $Pfill(T_\beta)$ from $G(T_\beta)$ gives a graph with no full envelope ordering.

In view of the recursive use of the algorithm on $T_2(x_R),\ldots,T_m(x_R)$ and the remark after Theorem 3.1, the same argument can be used for these subtrees. Finally, with a minor modification on the ancestor set $Ancestor(x_R)$, we can apply the argument to the first subtree $T_1(x_R)$. Thus, we arrive at the conclusion that no edge in $Pfill(T_\beta)$ can be removed showing that β is a minimal envelope ordering. □

We point out here that in general β is not necessarily a <u>minimum</u> envelope ordering. The tree in Figure 5.1 has a better ordering as shown in Figure 5.2.

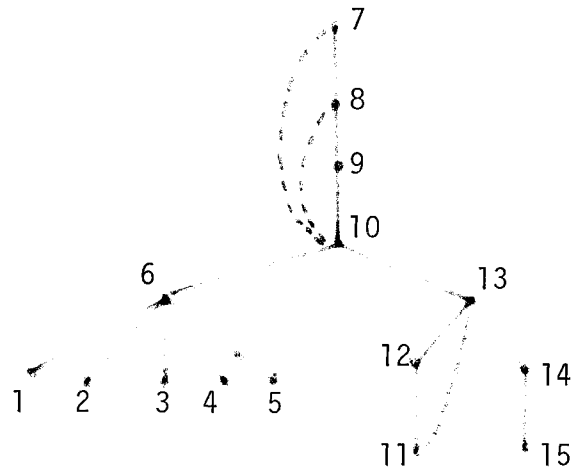


Figure 5.2 A better ordering than β in Fig.5.1

Although β may not be a minimum envelope ordering, the amount of potential fill in β is reasonably small. A bound is established in the following theorem.

Theorem 5.2 $\quad |Pfill(T_\beta)| \leq N \log_2 N.$

Proof Let $p(N)$ be the maximum possible number of potential fills on applying PORDER in the algorithm MET to a rooted tree with N nodes. We shall use induction to show $p(N) \leq N \log_2 N$.

Clearly, $p(2) \leq 2$. Assume the inequality holds for all $k < N$. Let (R,T) be a rooted tree with N nodes which suffers $p(N)$ number of potential fills. Let $T_1(R), T_2(R), \ldots, T_m(R)$ be the subtrees under the root R where $|T_1(R)| \geq |T_i(R)|$. Using Lemma 5.3 and the inductive assumption, we have

$$\begin{aligned} p(N) &\leq \sum_{i=1}^{m} p(|T_i(R)|) + \sum_{i=2}^{m} |T_i(R)| - m+1 \\ &\leq \sum_{i=1}^{m} |T_i(R)| \log_2 |T_i(R)| + \sum_{i=2}^{m} |T_i(R)| \\ &\leq |T_1(R)| \log_2 |T_1(R)| + \sum_{i=2}^{m} |T_i(R)| (\log_2 |T_i(R)| + 1). \end{aligned}$$

But for $i = 2,\ldots,m$, $|T_i(R)| \leq \frac{N-1}{2}$ so that $\log_2 |T_i(R)| \leq \log_2 N - 1$. Thus,

$$\begin{aligned} p(N) &\leq |T_1(R)| \log_2 |T_1(R)| + \sum_{i=2}^{N} |T_i(R)| \log_2 N \\ &\leq N \log_2 N. \end{aligned}$$

The theorem then follows from $|Pfill(T_\beta)| \leq p(N)$. $\square$

§6 Implementation and time complexity of MET

A peripherial node of a tree or subtree is required recursively in the algorithm MET. We first consider how such nodes can be determined efficiently using level structures. In general, the level structure at a node x in a graph is defined as:

$$LS(x) = \{L_0(x), L_1(x), \ldots, L_{\ell(x)}(x)\}$$

where $L_0(x) = \{x\}$

$$L_{i+1}(x) = Adj\left(\bigcup_{k=1}^{i} L_k\right) \qquad i = 0,1,\ldots$$

and $\ell(x) = \max\{d(x,y) \mid y \in X\}$.

In case x is a peripherial node, the length $\ell(x)$ of the level structure becomes the diameter of the graph. The following lemma provides an almost trivial way to determine a peripherial node of a tree.

Lemma 6.1 Let x be a node in a tree T, and its corresponding level structure be

$$LS(x) = \{L_0(x), L_1(x), \ldots, L_{\ell(x)}(x)\}.$$

Then any $y \in L_{\ell(x)}(x)$ is a peripherial node.

Proof The lemma follows from the fact that any two nodes in a tree are connected by exactly one path. □

After finding a peripherial node, say R, we have to compute the sizes of the subtrees, that is, the number of descendants D(x) under each node x in the rooted tree (R,T). D(x) can be determined quite simply by running through the level structure at R

$$LS(R) = \{L_0(R), L_1(R), \ldots, L_\delta(R)\}$$

bottom-up once:

```
For node x, D(x) := 1;
For ℓ := δ step -1 until 1 do
    For y ∈ L_ℓ(R)
        if z ∈ Adj(y) ∩ L_{ℓ-1}(R)
            then D(z) := D(z) + D(y);
```

It is evident that a peripherial node and the corresponding function D(x) can be determined in kN edge inspections. Here k is a constant. We are now ready to determine the asymptotic time complexity of the algorithm MET. Define c(N) to be the maximum possible cost required by MET to find a minimal envelope ordering for a tree of N nodes. We measure the cost by the number of edge inspections.

Theorem 6.1 $c(N) \leq kN \log_2 N$.

Proof We shall use induction to show the inequality. Clearly $c(2) \leq 2k$. Assume the result holds for all $k < N$. Let T be a tree with N nodes that requires c(N) number of edge inspections to find a minimal envelope ordering β.

Let R be a peripherial node found by MET and $T_1(R), T_2(R), \ldots, T_m(R)$ be the subtrees under R. From the algorithm and our induction assumption, we have

$$c(N) \le kN + c(|T_1(R)|) - k|T_1(R)| + \sum_{i=2}^{m} c(|T_i(R)|).$$

$$\le k(N-|T_1(R)|) + \sum_{i=1}^{m} k|T_i(R)|\log_2|T_i(R)|$$

$$= k + k|T_1(R)|\log_2|T_1(R)| + \sum_{i=2}^{m} k|T_i(R)|(\log_2|T_i(R)|+1).$$

Since $|T_i(R)| \le \frac{N-1}{2}$ for $i = 2,\ldots,m$,

$$c(N) \le k + k \log_2 N \sum_{i=1}^{m} |T_i(R)|$$

$$\le k \log_2 N(1+ \sum_{i=1}^{m} |T_i(R)| = kN \log_2 N. \qquad \square$$

§7 Experiments

The algorithm MET was implemented in ALGOL W and run on an IBM 360/75. The program was applied to a sequence of randomly generated trees of different sizes in order to find the experimental running time.

The test trees are generated recursively as follows. Let T_{N-1} be a random tree with N-1 nodes. A node x_i is selected at random from T_{N-1}. We then add the new node x_N and the edge $\{x_i, x_N\}$ to form T_N.

The test results are tabulated in Table 7.1. For each N, the result is the average of twenty random trees. For our implementation, the constant k in Theorem 6.1 equals two. In fact, the plot in Figure 7.1 shows that the cost is proportional to $N \log_2 N$. We also note that the number of potential fills is bounded by $N \log_2 N$ and in most cases much smaller than $N \log_2 N$.

N	N $\log_2$N	|Pfill(T_β)|	Total No. of edge inspections	Time used in seconds	Time/ N log N
100	664.4	76.9	1220.6	0.14	2.107(-4)
200	1528.8	195.8	2621.5	0.28	1.831(-4)
300	2468.6	346.0	4147.1	0.46	1.863(-4)
400	3457.5	504.9	5710.3	0.62	1.793(-4)
500	4482.9	682.1	7336.1	0.85	1.896(-4)
600	5537.3	838.2	8889.1	1.02	1.842(-4)
700	6615.9	1041.9	10627.6	1.22	1.844(-4)
800	7715.1	1203.7	12202.3	1.42	1.840(-4)
900	8832.4	1386.3	13863.7	1.68	1.902(-4)
1000	9965.8	1642.2	15801.2	1.91	1.916(-4)
1100	11113.6	1796.4	17344.7	2.13	1.917(-4)
1200	12274.6	2036.7	19227.5	2.39	1.947(-4)
1300	13447.6	2189.8	20769.1	2.69	2.000(-4)
1400	14631.7	2438.9	22688.5	2.91	1.989(-4)
1500	15826.1	2595.0	24238.3	3.17	2.003(-4)

Table 7.1 Average results of running MET on randomly generated trees

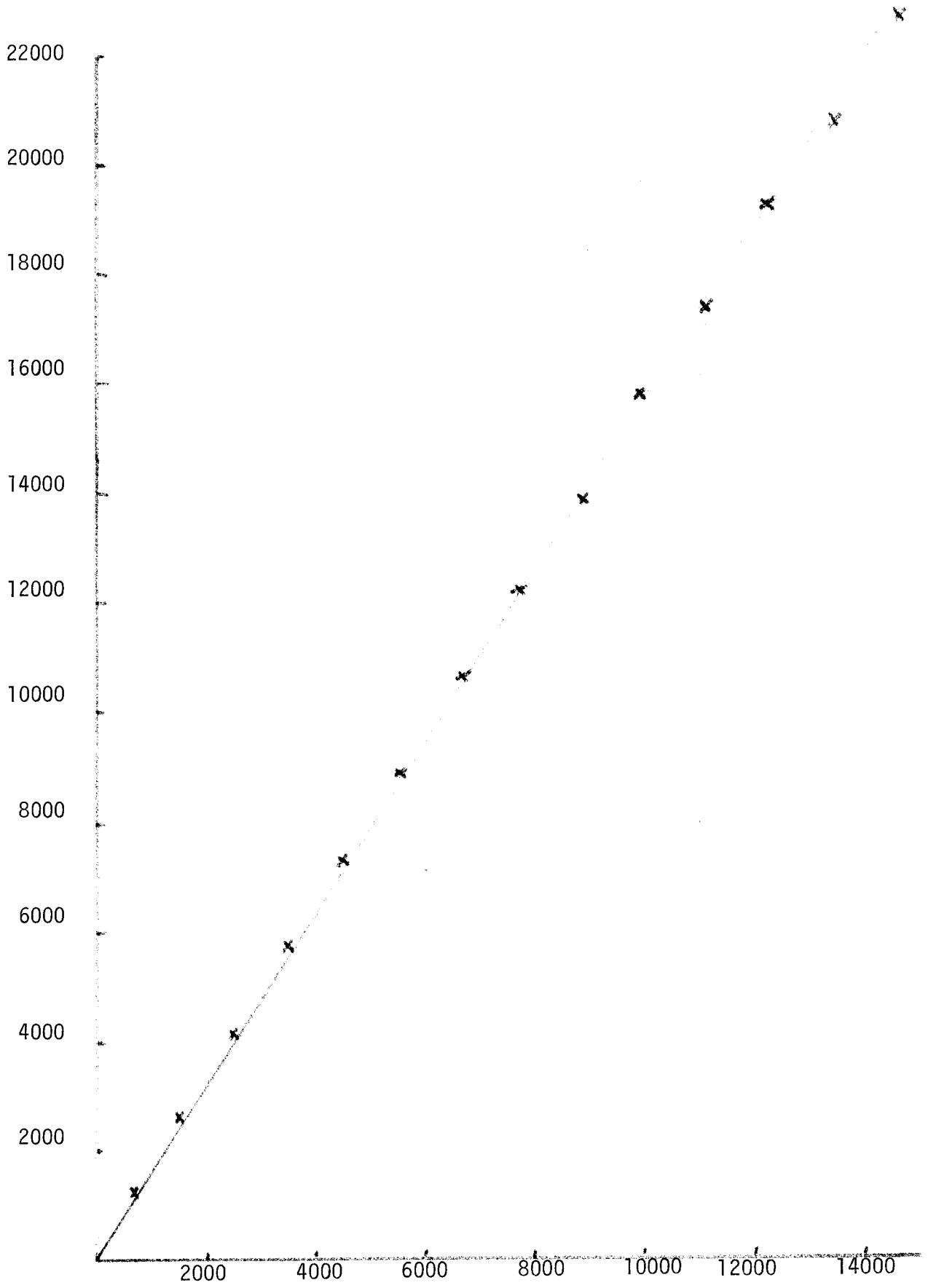


Figure 7.1 Plot of number of edge inspections against N $\log_2 N$

§8 Concluding remarks

We have presented an $O(N \log_2 N)$ algorithm that generates a minimal envelope ordering for trees. The algorithm is a modification of the familiar postorder scheme for rooted trees [15].

In practice, to minimize storage and computation in envelope methods, orderings that produce minimum envelopes are desired. To find them is usually very time-consuming. It is important to point out that minimal orderings can often be far inferior to minimum orderings. However, the minimal orderings produced by our algorithm has reasonably small envelope. Theorem 5.2 says that the number of potential fills is always bounded by $N \log_2 N$ on applying the algorithm to a tree of N nodes.

References

[1] Aho, A.V., Hopcroft, J.E., and Ullman, J.D., The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass. (1974).

[2] Berge, C., The Theory of Graphs and its Application, Wiley, New York (1958).

[3] Berge, C., Some classes of perfect graphs, in Graph Theory and Theoretical Physics, Harary F. (ed.), Academic Press, New York, (1967), 155-166.

[4] Bolstad, J.H., Leaf, G.K., Lindeman, A.J., and Kaper, H.G., An empirical investigation of reordering and data management for finite element systems of equations, ANL-8050 Argonne National Laboratory, Argonne, Illinois (1973).

[5] Cuthill, E., Several strategies for reducing the bandwidth of matrices, in Sparse Matrices and their Application, Rose,D.J., and Willoughby, R.A. (editors), Plenum Press, New York (1972).

[6] Cuthill, E., and McKee, J., Reducing the bandwidth of sparse symmetric matrices, Proc. ACM 23rd National Conference (1969).

[7] George, A., Computer implementation of the finite element method, Stanford Computer Science Dept., Technical Report STAN-CS-71-208, Stanford, California (1971).

[8] George, A., A survey of sparse matrix methods in the direct solution of finite element equations, Proc. Summer Computer Simulation Conference, Montreal, Canada (1973), 15-20.

[9] George, A., and Liu, W.H., A note on fill for sparse matrices, to appear in SIAM J. Numer. Anal.

[10] Gibbs, N.E., Poole, W.G., and Stockmeyer, P.K., An algorithm for reducing the bandwidth and profile of a sparse matrix, to appear in SIAM J. Numer. Anal.

[11] Gibbs, N.E., Poole, W.G., and Stockmeyer, P.K., A comparison of several bandwidth and profile reduction algorithms, ICASE Report (1974).

[12] Gustavson, F.G., Liniger, W., and Willoughby, R., Symbolic generation of an optimal Crout algorithm for sparse systems of linear equations, JACM 17 (1970), 87-109.

[13] Jennings, A., A compact storage scheme for the solution of symmetric simultaneous equations, Comput. J. 9 (1966), 281-285.

[14] King, I.P., An automatic reordering scheme for simultaneous equations derived from network systems, Intern. J. for Numer. Methods in Engineering 2 (1970), 523-533.

[15] Knuth, D.E., Fundamental Algorithms, Addison-Wesley, Reading, Mass. (1968).

[16] Martin, R.S., and Wilkinson, J.H., Symmetric decomposition of positive definite band matrices, Numer. Math. 7 (1965), 355-361.

[17] Ohtsuki, T., A fast algorithm for finding an optimal ordering in the vertex elimination on a graph, submitted to SIAM J. Computing.

[18] Ohtsuki, T., Cheung, L.K., and Fujisawa, T., Minimal triangulation of a graph and optimal pivoting order in a sparse matrix, to appear in J. of Math. Anal. & Appl.

[19] Parter, S.V., The use of linear graphs in Gauss elimination, SIAM Review 3 (1961), 119-130.

[20] Pooch, U.W., and Nieder, A., A survey of indexing techniques for sparse matrices, ACM Computing Review (1973), 109-133.

[21] Rose, D.J., Triangulated graphs and the elimination process, Journ. Math. Anal. and Appl. 32 (1970), 597-609.

[22] Rose, D.J., Symmetric elimination on sparse positive definite systems and the potential flow problem, Ph.D. thesis, Harvard University (1971).

[23] Rose, D.J., A graph-theoretic study of numerical solution of sparse positive definite system of linear equations, in Graph Theory and Computing, Read, R.C. (ed.), Academic Press (1972).

[24] Rose, D.J., and Tarjan, R.E., Algorithmic aspects of vertex elimination on graphs, manuscript.

[25] Rheinboldt, W.C., and Meszteny, C.K., Programs for the solution of large sparse matrix problems based on the arc-graph structure, Technical Report TR-262 (1973), Computer Science Center, University of Maryland.

[26] Tarjan, R.E., Depth first search and linear graph algorithms, SIAM J. Comput. 1 (1972), 146-160.

[27] Tewarson, R.P., Sparse Matrices, Academic Press, New York (1973).

[28] Wilkinson, J.H., The Algebraic Eigenvalue Problem, Clarendon Press, London (1965).