Hierarchies and Planarity Theory

GIUSEPPE DI BATTISTA AND ENRICO NARDELLI

Abstract —**Hierarchies are widely used in many fields of social and mathematical sciences. In diagrammatic representations of hierarchies the minimization of the number of crossings between edges is a well-admitted criterion for improving readability. An efficient algorithm is proposed for testing if a hierarchy is planar (i.e., it can be drawn without edge crossings). Furthermore, a complete combinatorial characterization of the class of planar hierarchies is given.**

I. Introduction

Hierarchies are widely used in structuring and managing complex problems in many fields of social and mathematical sciences [20]. A clear graphic presentation of a hierarchy allows the reader to focus on the information content of the diagram [18], [19].

Several algorithms have been proposed in the literature for achieving readability of diagrams by means of automatic tools (see, for example, [12], [3]). The state of the art in this field is surveyed in [15]. In particular, some algorithms have been presented for automatic drawing of hierarchies [19], [5], [14]. Usually, hierarchies are drawn by assigning the vertices to levels and representing the edges as straight lines (or at least as polygonal lines monotonically increasing in the vertical direction).

A criterion largely used to obtain a good readability is to produce diagrams with a limited number of crossings between edges [2]. From the point of view of combinatorial complexity the problem of minimizing the crossing number for k-level hierarchies has been shown to be NP-complete even if $k = 2$ [9]. The problem remains NP-complete also if the positions of symbols in one of the two levels is fixed [6]. Moreover, the problem of deleting the minimum number of edges to obtain a two-level planar hierarchy is NP-complete [17].

These arguments enforce the heuristic approach used in dealing with the crossing number problem in [19]. Improvements to such an approach have been proposed in [5], [14], [13].

The crossing number problem has been shown to be NP-complete for unlayered graphs [9], but for this kind of graph there exist two well-known algorithms to test if the graph is planar (i.e., it can be drawn without edge crossings): the first is presented in [8] and the second in [11] and [4]. Furthermore, a complete combinatorial characterization of the class of planar graphs was given in [10]. However, as noticed in [5], the planarity-testing algorithms do not yield a representation of the graph according to any hierarchy. See, for instance, the hierarchy in Fig. 1 that is planar in the usual sense but that cannot be drawn without crossings if the vertices are constrained to remain on the levels and the edges are represented with straight lines. An algorithm for testing planarity of two-level hierarchies is given in [6]. However, it is not possible, in general, to reduce the problem of testing the planarity of a k-level hierarchy to the one of testing the planarity of the $(k-1)$ two-level hierarchies that compose it.

In this paper an efficient algorithm is proposed for testing if a hierarchy can be drawn without edge crossings. The basic idea is to virtually generate all possible drawings, then to delete those containing crossings. The data structure that is used to design the algorithm efficiently is the PQ-tree structure, presented in [4]. If the hierarchy admits a cross-free drawing, the algorithm constructs it from the PQ-tree structure.

Manuscript received September 11, 1987; revised July 29, 1988. This work was supported in part by Progetto Finalizzato Trasporti of the Italian National Research Council (CNR), and by the Istituto di Studi sulla Ricerca e Documentazione Scientifica of the Italian National Research Council (CNR). This correspondence was presented in part at the 12th Workshop on Graph-theoretic Concepts in Computer Science, Bernried, 1986.

G. DiBattista is with the Dipartimento di Informatica e Sistemistica, Universita delgi Studi di Roma, Via Buonarroti, 12 00185, Roma, Italia.

E. Nardelli is with the Istituto di Analisi dei sistemi ed Informatica of the Italian National Research Council (CNR), Viale Manzoni, 30 00185, Roma, Italia.

IEEE Log Number 8824021.

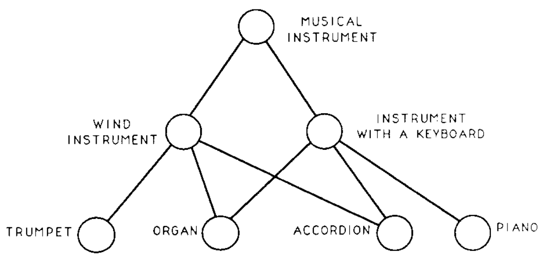


Fig. 1. Nonplanar hierarchy.

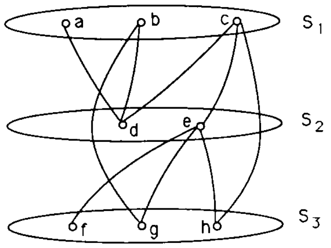


Fig. 2. Hierarchy with three subsets.

Finally, a complete combinatorial characterization of the class of hierarchies that admit a planar representation is given. Such a characterization is formulated in terms of necessary and sufficient conditions for a hierarchy to be cross free.

The rest of the paper is organized as follows. In Section II we introduce basic definitions and some preliminary results. In Section III we present a theoretical approach for testing the planarity of hierarchies. Section IV deals with the planarity testing algorithm and its time complexity. In Section V we give necessary and sufficient conditions for a hierarchy to admit a representation without edge crossings. Finally, in Section VI we outline further research topics.

II. Definitions and Preliminary Results

Let $G(V,E)$ be a graph, and suppose that the set V of its vertices is partitioned into k subsets $(S_1, S_2, \cdots, S_k)$ such that there is no edge of E connecting two vertices in the same subset. We assume that each edge is *directed* from the vertex belonging to the subset with lower index to the vertex of the subset with higher index (we refer to [7] for standard terminology on graphs).

If for each vertex y belonging to subset S_j $(j \neq 1)$ there exists at least one edge (x, y) such that x belongs to a subset S_i $(i < j)$, we say that G is a *hierarchy* [19] (see in Fig. 2 an example of a hierarchy made up of three subsets).

Notice that in Fig. 2 the edges are drawn without arrows because we assume that they are directed from the top to the bottom of the figure. This kind of representation will be used in all the figures of the paper.

A hierarchy is said to be *proper* [19] if every edge connects vertices belonging to consecutive subsets. In the following we refer to a hierarchy by $G(V, E, L)$, where

V	set of vertices,
E	set of directed edges,
L	function $V \rightarrow 1, 2, \cdots, k$ that gives for each vertex the index of the subset to which it belongs.

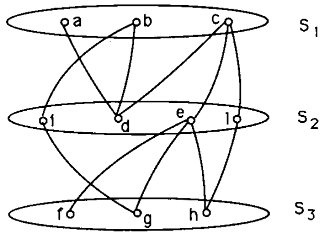


Fig. 3. Proper hierarchy.

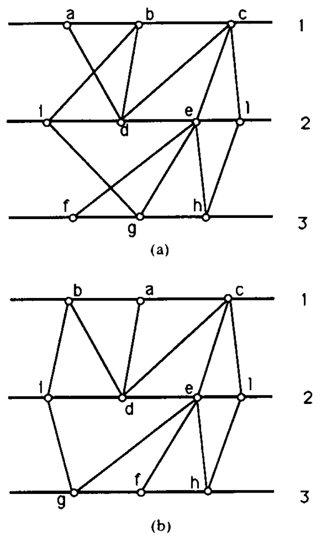


Fig. 4. Two embeddings for the hierarchy of Fig. 3.

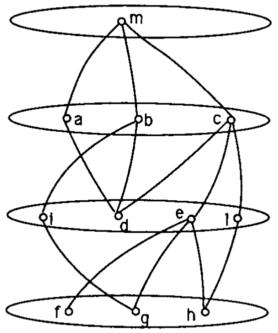


Fig. 5. Hierarchy of Fig. 3 after addition of new subset.

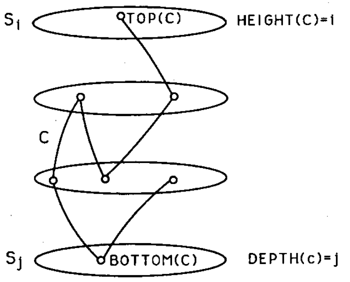


Fig. 6. Example of path.

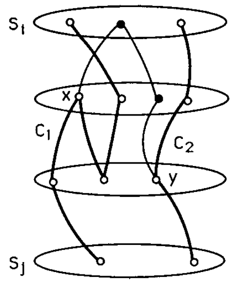


Fig. 7. Example of bridge.

The following properties hold for L:

1) for each $v, w \in V$ such that $L(v) = L(w) = i$, $(v, w) \notin E$;
2) for each v such that $L(v) = i$ and $i \neq 1 \Rightarrow$ there exists $w \in V$ such that $L(w) \leqslant i - 1$ and $(w, v) \in E$.

G is proper if

3) for each $(w, v) \in E \Rightarrow L(v) - L(w) = 1$ (see in Fig. 3 a proper hierarchy).

Every hierarchy can be easily reduced into a proper one by transforming an edge between nonconsecutive subsets into a sequence of edges, each one connecting two consecutive subsets. In the following we deal only with proper hierarchies.

Given k parallel lines on the plane, consecutively numbered, we define a *k-line embedding* of a hierarchy $G(V, E, L)$ to be a drawing such that

- all vertices of subset S_i are drawn on line i;
- edges are drawn as *straight lines*.

The embedding is completely specified by the *ordered sequences* of vertices on lines corresponding to subsets. A *k-line planar embedding* is a k-line embedding without crossings between edges (see in Fig. 4 two embeddings of the same hierarchy, one of which is k-line planar). A hierarchy is said to be k-line planar if it admits at least one k-line planar embedding.

Notice that subset S_1 may have more than one vertex. In this case it is always possible to add a new subset S_0 with exactly one vertex, connected to every vertex of S_1 (see in Fig. 5 the hierarchy of Fig. 3 after the addition of the new subset). Such a transformation does not modify the planarity properties of the given hierarchy. As a consequence, in the following we consider only hierarchies such that $|S_1| = 1$.

Let $G(V, E, L)$ be a hierarchy. We introduce the following definitions.

A *path* is an ordered sequence of vertices $v_1, v_2, \cdots, v_n$, $(n > 1)$ such that for each pair v_i, v_{i+1} $(i = 1, 2, \cdots, n-1)$ either (v_i, v_{i+1}) or (v_{i+1}, v_i) belongs to E.

Let C be a path. BOTTOM(C) is one of the vertices of C such that $L(\text{BOTTOM}(C)) \geqslant L(v)$ for each v belonging to C. TOP(C) is one of the vertices of C such that $L(\text{TOP}(C)) \leqslant L(v)$ for each v belonging to C. Let DEPTH(C) = L(BOTTOM(C)) and HEIGHT(C) = L(TOP(C)) (see Fig. 6).

We denote with LACE(i, j), $(i < j)$, the set of paths C connecting any two vertices x of S_i and y of S_j such that HEIGHT(C) = i and DEPTH(C) = j.

Let C_1 and C_2 be two completely distinct paths belonging to LACE(i, j). We define *bridge* to be a path connecting vertices x and y, with x belonging to C_1 and y belonging to C_2 (see Fig. 7).

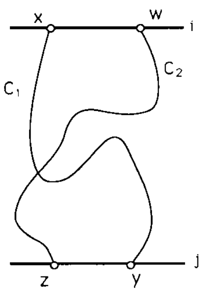


Fig. 8. Example for Lemma 2.

Given a k-line embedding Γ of G and a subset S_i:

- for each vertex x of S_i, ORD(x) indicates the position of x in the ordered sequence associated to S_i in Γ;
- given a subset $R = \{x_1, x_2, \cdots, x_n\}$ $(n > 1)$ of S_i, we say that vertices of R are *consecutively ordered* if ORD(x_{i+1}) = ORD$(x_i) + 1$ $(i = 1, 2, \cdots, n-1)$.

The following lemmas can be easily proved by means of topological arguments.

Lemma 1: Let (x, y) and (w, z) be two edges such that x is distinct from w and y is distinct from z. Suppose that $L(x) = L(w) = i$ and $L(y) = L(z) = i + 1$. If ORD$(x) <$ ORD(w) and ORD$(y) >$ ORD(z), then (x, y) and (w, z) cross each other in Γ.

The previous lemma can be generalized as follows.

Lemma 2: Given four distinct vertices x, y, w, and z, such that $L(x) = L(w) = i$ and $L(y) = L(z) = j > i$, let C_1 be a path between x and y belonging to LACE(i, j), and let C_2 be a path between w and z belonging to LACE(i, j). Suppose that C_1 and C_2 are completely disjoint. If ORD$(x) <$ ORD(w) and ORD$(y) >$ ORD(z), then it exists at least one crossing in Γ (see Fig. 8).

In the following theorem we show three sufficient conditions for G to be not k-line planar.

Theorem 1: Let L_1, L_2, and L_3 be three paths belonging to LACE(i, j). If one of the following conditions hold, then $G(V, E, L)$ is not k-line planar.

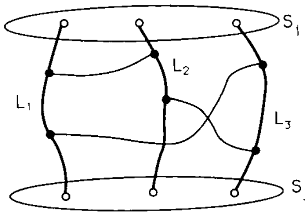
1) L_1, L_2, and L_3 are completely disjoint and pairwise connected by bridges. Bridges share no vertex with L_1, L_2, and L_3, but for the endpoints (see Fig. 9(a)).

2) L_1 and L_2 share an endpoint p and a path C (possibly empty) starting from p, $L_1 \cap L_3 = L_2 \cap L_3 = \varnothing$; there is a bridge b_1 between L_1 and L_3 and a bridge b_2 between L_2 and L_3, $b_1 \cap L_2 = b_2 \cap L_1 = \varnothing$ (see Fig. 9(b));

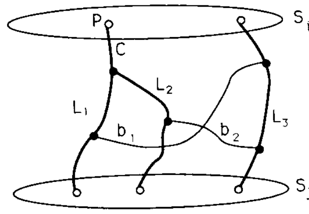
3) L_1 and L_2 share an endpoint p and a path C_1 (possibly empty) starting from p, L_1 and L_3 share an endpoint q $(q \neq p)$ and a path C_2 (possibly empty) starting from q, $C_2 \cap C_1 = \varnothing$; L_2 and L_3 are connected by a bridge b, $b \cap L_1 = \varnothing$ (see Fig. 9(c)).

Proof (by contradiction): Consider any k-line embedding Γ of G. Suppose 1) holds and assume, without loss of generality, that ORD(TOP(L_1)) < ORD(TOP(L_2)) < ORD(TOP(L_3)). If Γ is k-line planar, then ORD(BOTTOM(L_1)) < ORD(BOTTOM(L_2)) < ORD(BOTTOM(L_3)). Using the bridge between L_1 and L_3, it is possible to find a path between TOP(L_1) and BOTTOM(L_3) and a path between TOP(L_2) and BOTTOM(L_2) which satisfy the hypothesis of Lemma 2.

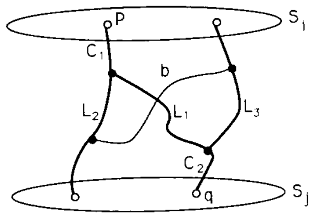
Suppose 2) holds and assume, without loss of generality, that p = TOP(L_1) = TOP(L_2) and ORD(p) < ORD(TOP(L_3)). If Γ is k-line planar, then we can assume that ORD(BOTTOM(L_1)) < ORD(BOTTOM(L_2)) < ORD(BOTTOM(L_3)). Using the bridge between L_1 and L_3, it is possible to find a path between TOP(L_3) and BOTTOM(L_1) and a path between TOP(L_2) and BOTTOM(L_2) which satisfy the hypothesis of Lemma 2.



(a)



(b)



(c)

Fig. 9. Examples for Theorem 1.

Suppose 3) holds and assume, without loss of generality, that

$$p = \text{TOP}(L_1) = \text{TOP}(L_2),$$

$$\text{ORD}(p) < \text{ORD}(\text{TOP}(L_3)),$$

$$q = \text{BOTTOM}(L_1) = \text{BOTTOM}(L_3).$$

If Γ is k-line planar, then ORD(BOTTOM(L_2)) < ORD(q). Using the bridge between L_2 and L_3, it is possible to find a path between TOP(L_1) and BOTTOM(L_1) and a path between TOP(L_3) and BOTTOM(L_2) that satisfy the hypothesis of Lemma 2.

In Section V we shall show that Theorem 1 also gives necessary conditions for a hierarchy to not be k-line planar.

III. Theoretical Bases for Hierarchy Planarity Testing

In this section we state the theoretical bases for the planarity testing algorithm that is shown in Section IV. Let $T(V, E_t, L)$ be any directed spanning tree of a hierarchy $G(V, E, L)$, and let $E_0 = E - E_t$. Suppose that G is k-line planar. The following theorem holds.

Theorem 2: For each directed spanning tree T of G, there exists at least one k-line planar embedding σ of T such that, simply adding to σ the edges in E_0, a k-line planar embedding of G is obtained.

Proof (by construction): Choose any directed spanning tree T of G (see Fig. 10(a)). If G is k-line planar, then it admits a k-line planar embedding Γ. Now, remove from Γ the edges belonging to E_0, and obtain a k-line planar embedding σ for T that satisfies the thesis (see Fig. 10(b)).

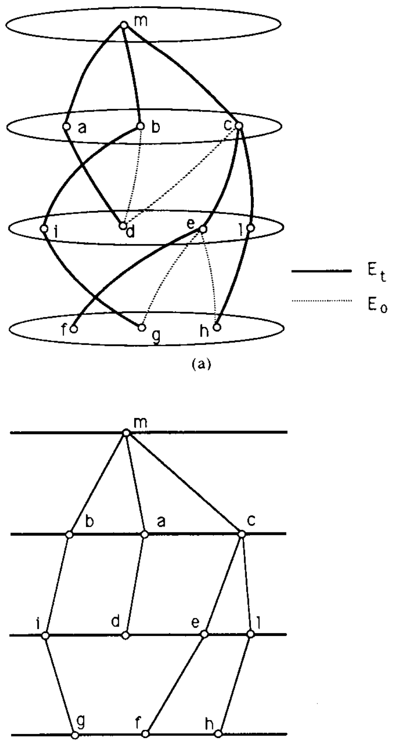


Fig. 10. Directed spanning tree for hierarchy in Fig. 5 and one of its k-line planar embeddings that satisfies Theorem 2.

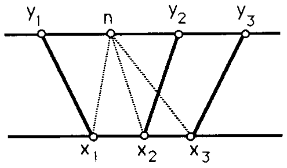


Fig. 11. Example for Fact 1.

Notice that in Fig. 10 edges belonging to E_t are represented with bold lines and edges belonging to E_0 are represented with dotted lines. From now on we shall use this kind of representation. Theorem 2 suggests an approach for k-line planarity testing based on the following algorithm.

Algorithm KLT (G, Σ):

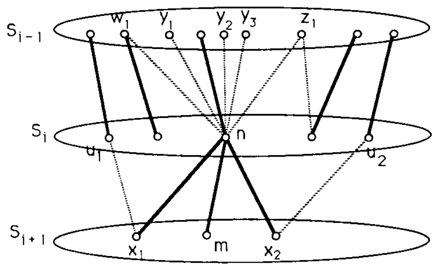
1) Choose any directed spanning tree T of G.
2) Determine the set Σ of all the possible k-line planar embeddings for T.
3) For each edge e belonging to E_0 do

- add the edge e to each embedding in Σ, and
- remove from Σ all the embeddings that are not k-line planar after the addition of e.

For Theorem 2, when Algorithm KLT terminates, if Σ is empty, then the hierarchy G is not k-line planar, else it is k-line planar.

Now we deal with the problem of removing from Σ all the embeddings that are not k-line planar after the addition of a certain edge. We give some preliminary results.

Fact 1: Let T be any directed spanning tree of $G(V, E, L)$. If G is k-line planar, then no vertex has more than two outgoing edges belonging to E_0.



$V^0(n)=\{x_1, x_2\}$
$V'(n)=\{y_1, y_2, y_3\}$
$V''(n)=\{z_1\}$
$V^\wedge(n)=\{w_1\}$
$V^\sim(n, x_1)=\{u_1\}$
$V^\sim(n, x_2)=\{u_2\}$

Fig. 12. Example of introduced notation.

Proof: Suppose n has three outgoing edges in E_0, namely (n, x_1), (n, x_2), and (n, x_3). Since T is a directed spanning tree of G, each one of x_1, x_2, and x_3 has a father in T, respectively y_1, y_2, and y_3. If y_1, y_2, and y_3 are distinct vertices (see Fig. 11), then Theorem 1 (condition 1) holds and G is not k-line planar. If two vertices among y_1, y_2, and y_3 coincide, then Theorem 1 (condition 3) holds and G is not k-line planar.

Fact 2: Let T be any directed spanning tree of $G(V, E, L)$. If G is k-line planar, then no vertex has more than two sons in T with ingoing edges belonging to E_0.

Proof: Suppose n has three sons in T, namely x_1, x_2, and x_3, each one with ingoing edges in E_0. Let (y_1, x_1), (y_2, x_2), and (y_3, x_3) be three edges in E_0. If y_1, y_2, and y_3 are distinct vertices, then Theorem 1 (condition 1) holds and G is not k-line planar. If two vertices among y_1, y_2, and y_3 coincide, then Theorem 1 (condition 3) holds and G is not k-line planar.

Suppose now Γ is any k-line planar embedding of $G(V, E, L)$ obtained by adding edges of E_0 to a k-line planar embedding of a spanning tree T. Let n be a vertex belonging to S_i, FATHER(n) the father of n in T, and SONS(n) the set of sons of n in T.

For the sake of shortness we introduce the following notation (see Fig. 12)

$V^0(n)$	set of vertices of S_{i+1} that belong to SONS(n) and have at least one ingoing edge in E_0; from Fact 2 it follows that $\lvert V^0(n)\rvert \leqslant 2$,
$V'(n)$	set of vertices of S_{i-1} that have exactly one outgoing edge in E_0, which incides on vertex n, and have no outgoing edges in E_t,
$V''(n)$	set of vertices of S_{i-1} that have exactly two outgoing edges in E_0, one of which incides on vertex n, and have no outgoing edges in E_t,
$V^\wedge(n)$	set of vertices of S_{i-1} that have at least one outgoing edge in E_0, which incides on vertex n, and have at least one outgoing edge in E_t.

The following properties (1–5) hold for Γ. They can be trivially proved by contradiction using Lemma 1.

Property 1: Suppose w belongs to SONS(n). For each vertex x such that ORD$(x) <$ ORD(w), then ORD(FATHER$(x)) \leqslant$ ORD(n) and for each vertex y such that ORD$(y) >$ ORD(w), then ORD(FATHER$(y)) \geqslant$ ORD(n). As a consequence, the vertices belonging to SONS(n) are consecutively ordered in Γ on the line corresponding to S_{i+1}.

Property 2: The vertices belonging to $\{$FATHER$(n)\} \cup V'(n)$ are consecutively ordered in Γ on the line corresponding to S_{i-1}, but

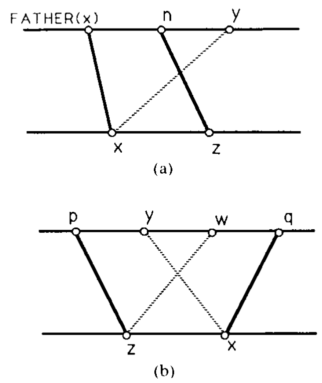


Fig. 13. Two examples for analysis of crossings.

possibly for the vertices of S_{i-1} without outgoing edges in E.

Property 3: Suppose x belongs to $V''(n)$. Then the vertices belonging to $\{x\} \cup \{\text{FATHER}(n)\} \cup V'(n)$ are consecutively ordered in Γ on the line corresponding to S_{i-1}, but possibly for the vertices of S_{i-1} without outgoing edges in E. Note that $|V''(n)| \leqslant 2$.

Property 4: Suppose x belongs to $V^{\wedge}(n)$. Then the vertices belonging to $\{x\} \cup \{\text{FATHER}(n)\} \cup V'(n)$ are consecutively ordered in Γ on the line corresponding to S_{i-1}, but possibly for the vertices of S_{i-1} without outgoing edges in E. Note that $|V^{\wedge}(n)| \leqslant 2$.

Let $V^{-}(n,x)$ be the set of vertices u of S_i such that (u,x) belongs to E_0 and x belongs to $V^0(n)$ (see again Fig. 12).

Property 5: Suppose u belongs to $V^{-}(n,x)$. Let m be any vertex belonging to SONS(n) and distinct from x. Then for all u, either $\text{ORD}(u) < \text{ORD}(n)$ and $\text{ORD}(x) < \text{ORD}(m)$, or $\text{ORD}(u) > \text{ORD}(n)$ and $\text{ORD}(x) > \text{ORD}(m)$.

The previous properties give necessary conditions for a k-line embedding Γ, obtained by adding edges in E_0 to a k-line planar embedding of T, to be planar. It raises naturally the question of whether or not they are also sufficient. In the following the answer will be provided.

Let (x,y) be an edge in E_0. Clearly, x belongs to one of the disjoint sets $V'(y), V''(y), V^{\wedge}(y)$. We say, respectively, that (x,y) is an edge of type 1, 2, or 3.

Suppose now properties 1–5 hold for Γ but it is not k-line planar. Then it has at least one crossing. We distinguish among three different cases

a) an edge of E_t crosses an edge of E_t,
b) an edge of E_t crosses an edge of E_0, and
c) an edge of E_0 crosses an edge of E_0.

Case a) is not possible due to the construction of Γ.

Suppose case b) holds. Let (y,x) be the edge of E_0, and let (n,z) be the edge of E_t. From Property 5 it follows that $n \neq \text{FATHER}(x)$ (see Fig. 13(a)). Furthermore, n has to be between FATHER(x) and y; otherwise, the argument of case a) can be applied. The edge (y,x) may be of type 1, 2, or 3. However, this is not possible due, respectively, to Properties 2, 3, and 4. As a consequence, case b) is not possible.

Finally, suppose case c) holds. Let (y,x) and (w,z) be the two edges of E_0. Assume, without loss of generality, that $\text{ORD}(y) < \text{ORD}(w)$, and let $p = \text{FATHER}(z)$ and $q = \text{FATHER}(x)$ (see Fig. 13(b)). Notice that must be $\text{ORD}(p) \leqslant \text{ORD}(y)$ and $\text{ORD}(q) \geqslant \text{ORD}(w)$; otherwise, the argument of case b) can be applied.

Now three subcases are possible:

c1) $p \neq y$ and $q \neq w$; note that y is between p and w; the edge (y,x) may be of type 1, 2, or 3, but this is not possible due, respectively, to Properties 2, 3, and 4;

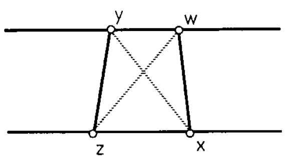


Fig. 14. Embedding that does not satisfy Property 6.

c2) $p \neq y$ and $q = w$; note that y is between p and w, but this is not possible due to Property 4;

c3) $p = y$ and $q = w$; this is the only case in which an embedding Γ that satisfies Properties 1–5 is not k-line planar.

Starting from these arguments, to complete the characterization of a k-line planar embedding Γ we can introduce the following.

Property 6: Suppose w belongs to $V^{\wedge}(z)$. Then there is no edge (y,x) in E_0 such that $y = \text{FATHER}(z)$ and x belongs to SONS(w) (see Fig. 14).

We can say now that Properties 1–6 are necessary and sufficient conditions for a k-line embedding Γ, obtained by adding edges in E_0 to a k-line planar embedding of T, to be planar. A further observation can be made: if we choose the directed spanning tree T using a depth first search (DFS) technique [16], Property 6 is always satisfied.

In fact, if Property 6 does not hold, then there exist the following edges (see again Fig. 14): (y,x) and (w,z) belonging to E_0, and (y,z) and (w,x) belonging to E_t. If T is chosen with a DFS technique, we can suppose without loss of generality that y is visited before w. However, in this case, for the nature of the DFS technique, x is visited starting from y and not starting from w; therefore, (y,x) belongs to E_t.

IV. An Algorithm for Planarity Testing of Hierarchies

The number of k-line planar embeddings of a tree grows rapidly (as a factorial in the worst case) with the number of vertices. As a consequence, the implementation of Algorithm KLT requires a data structure able to deal efficiently with this combinatorial explosion.

Now we show an efficient way to represent all k-line planar embeddings of a tree. We use a data structure already known in literature as PQ-tree [4]. A concise description of the PQ-tree data structure is given in Appendix I.

A PQ-tree allows one to represent all permutations of a set U of objects in which the elements of certain subsets of U occur as consecutive subsequences. As it happens for all hierarchies, an embedding of a directed spanning tree T of a hierarchy is completely specified by the ordered sequences of vertices on each level. Due to Property 1, a k-line planar embedding of a directed spanning tree is completely specified by the ordering of the leaves obtained with a DFS traversal of the tree. Thus all k-line planar embeddings of T can be represented with a PQ-tree, named π, that can be generated by the following straightforward algorithm.

```
Algorithm Create ($T, \pi$):
  BEGIN
    copy $T$ into $\pi$;
  FOR EACH vertex $v$ belonging to $\pi$
    IF $v$ is not a leaf
    THEN
      substitute $v$ with a $P$-node
  END.
```

It is easy to see (using Property 1) that PQ-tree π generated by Algorithm Create (see Fig. 15(a)) represents all the possible k-line planar embeddings of T. Notice that a P-node with just one son is not meaningful and can be replaced by its son (see Fig. 15(b) where such a transformation is applied to the PQ-tree of

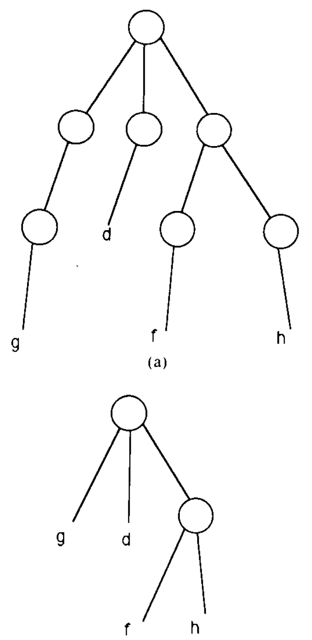


Fig. 15. Application of Algorithm Create to spanning tree of Fig. 10.

Fig. 15(a)). It is not difficult to modify Algorithm Create to produce *PQ*-trees in which all *P*-nodes have at least two sons.

In the detailed description of the planarity testing algorithm, the *PQ*-tree used to represent all k-line embeddings of the directed spanning tree is created incrementally, while the algorithm examines sequentially each subset S_i of G.

A further issue to be considered is the number of edges. A graph has, in general, $O(|V|^2)$ edges, but due to the basic property of planar graphs ($|E| \leqslant 3|V|-6$) we can focus our attention on the case of $O(|V|)$ edges. To have a good time efficiency, the first test in the algorithm gets rid of the hierarchies that do not satisfy this basic property of planar graphs.

We assume to have a procedure Remove(π, A,Empty) that, given the *PQ*-tree π that represents a set Σ of k-line planar embeddings of T and a set A of leaves of T, removes from π all the embeddings such that the leaves belonging to A do not appear as a consecutive subsequence. Empty is a Boolean variable, true if π is empty. Note that Remove is similar to procedure Reduce(T, S) in [4], where $\pi = T$, $A = S$, and Empty is true if T is returned null.

Using the *PQ*-tree data structure and the arguments presented in Section III Algorithm KLT becomes the following.

```
Algorithm KLT'(G, $\pi$):
BEGIN
IF $|E| > 3|V| - 6$
THEN RETURN(not $k$-line planar);
choose any directed spanning tree $T$ of $G$;
$\pi \leftarrow$ the empty $PQ$-tree;
put in $\pi$ the leaf corresponding to the root of $T$;
FOR $i := 1$ TO $k - 1$ DO
  BEGIN
  FOR EACH vertex $m$ belonging to $S_i$ DO
    BEGIN
1)  IF $|\{(m, x) \in E_0\}| > 2$
    THEN RETURN(not $k$-line planar);
    IF SONS($m$) $\neq \varnothing$
    THEN IF |SONS($m$)| > 1
      THEN BEGIN
        replace the leaf of $\pi$ corresponding to $m$ with a
        $P$-node;
        FOR EACH vertex $w$ belonging to SONS($m$) DO
          add to the $P$-node just inserted in $\pi$
          the leaf corresponding to $w$;
        END
      ELSE replace the leaf of $\pi$ corresponding to $m$ with its
      son
    ELSE IF $m$ has no outgoing edges in $E_0$
      THEN Mark($m$);
    END;
  FOR EACH vertex $n$ such that ($n$ belongs to $S_{i+1}$) and ($n$ has
    ingoing edges in $E_0$) DO
    BEGIN
2)  Remove($\pi$, $\{n\} \cup V'(n)$,Empty);
    IF Empty
    THEN RETURN(not $k$-line planar);
3)  FOR EACH vertex $v$ belonging to $V''(n)$ DO
      BEGIN
      Remove($\pi$, $\{n\} \cup V'(n) \cup \{v\}$,Empty);
      IF Empty
      THEN RETURN(not $k$-line planar);
      END;
4)  FOR EACH vertex $v$ belonging to $V \wedge (n)$ DO
      BEGIN
5)    IF $\{(y, z) := y =$ FATHER($n$), $z \in$ SONS($v$)$\} \neq \varnothing$
      THEN RETURN(not $k$-line planar);
6)    Remove($\pi$, $\{n\} \cup V'(n) \cup$ SONS($v$),Empty);
      IF Empty
      THEN RETURN(not $k$-line planar);
      END
    END;
    FOR EACH vertex $m$ such that ($m$ belongs to $S_i$) and
    (SONS($m$) $= \varnothing$) DO
      Mark($m$);
    END;
  RETURN($k$-line planar);
  END.
```

At line 1 the condition of Fact 1 is imposed to improve the time efficiency of the algorithm.

It is straightforward to see that Algorithm KLT′ imposes that Properties 2–6 hold for each embedding contained in π. This is made subset by subset, starting from S_1. At line 2 we make Property 2 satisfied. In the cycle that starts at line 3 we impose that Property 3 holds. In the cycle that starts at line 4 we deal with vertices belonging to $V^{\wedge}(n)$. In particular, at line 5 we check if Property 6 is satisfied (notice that this test can be omitted if the directed spanning tree T has been chosen by means of the DFS technique), and at lien 6 we impose Properties 4 and 5.

Procedure Mark(n) makes the node n negligible in the subsequent manipulations of π carried on by procedure Remove. Procedure Mark marks recursively also *P*-nodes and *Q*-nodes whose sons are all marked. For the purpose of testing k-line planarity such nodes could be removed from π, but they are useful in the construction of the k-line embedding.

The following theorem derives from the foregoing arguments and from the properties stated in Section III.

Theorem 3: Algorithm KLT′ returns "not k-line planar" if and only if the hierarchy G is not k-line planar.

Now we show an example of application of Algorithm KLT′. The hierarchy whose k-line planarity is tested is shown in Fig. 16(a); in Fig. 16(b) a directed spanning tree is chosen. Since the hierarchy is composed by four subsets, the external cycle of the algorithm must be repeated three times. Fig. 16(c) shows π at the end of the first iteration ($i = 1$).

At the beginning of the second iteration ($i = 2$) leaves b, c, and d are replaced by new nodes, as it appears in Fig. 16(d). After the

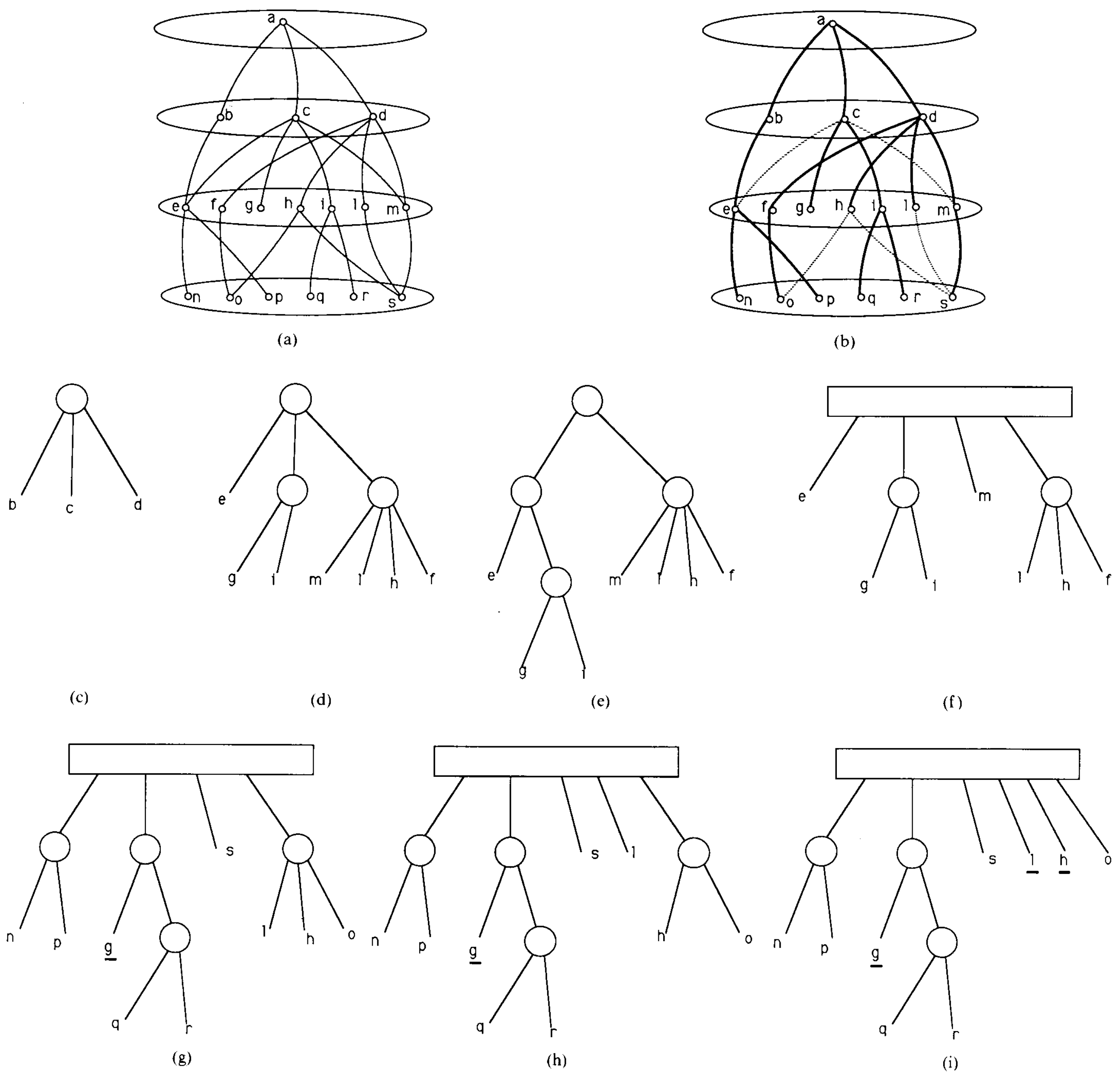


Fig. 16. Application of Algorithm KLT′.

replacement, it starts the cycle that examines vertices belonging to S_{i+1}; due to the fact that c belongs to $V^{\wedge}(e)$ procedure Remove is invoked as follows: Remove(π, $\{e, g, i\}$,Empty). Notice that in this case $V'(e)$ is empty, and SONS(c) = $\{g, i\}$. See in Fig. 16(e) π after the execution of procedure Remove. Since π is not empty, procedure Remove returns with Empty equal to false. Now Remove is applied again, since c belongs to $V^{\wedge}(m)$: Remove(π, $\{m, g, i\}$,Empty). Also in this case Empty is equal to false (see in Fig. 16(f) π after this last Remove).

At the beginning of the third iteration ($i = 3$) leaves e, i, m, and f are replaced, as it appears in Fig. 16(g), and g is marked. Now, since $V'(s) = \{l\}$, Remove (line 2 of the algorithm) is invoked: Remove(π, $\{s, l\}$,Empty). The result is shown in Fig. 16(h). At this point h is found to belong to $V''(s)$; therefore, Remove is applied again: Remove(π, $\{s, l, h\}$,Empty). Nodes l and h are marked. See in Fig. 16(i) the final version of π produced by Algorithm KLT′. As Empty is always returned false, G is k-line planar.

We can consider now the time complexity of the proposed algorithm. The complexity of finding a spanning tree is $O(|E|)$. However, in our case the number of edges is $O(|V|)$. The overall complexity of the cycles that scan subsets and their vertices is $O(|V|)$. In fact, each of the cycles of lines 3 and 4 is iterated at most twice, due to Properties 3 and 4. Moreover, to determine which edges are in the set tested at line 5 has a complexity $O(1)$, due to Fact 1. Procedure Mark has clearly an overall complexity of $O(|V|)$. As a consequence, procedure Remove is critical for the complexity of the whole algorithm.

If the set of objects to manipulate with a PQ-tree has size M, if the constraints on feasible permutations are expressed by means of N sets, and if the sum of the sizes of such N sets is S, then the procedure that satisfies all constraints has complexity

$O(M+N+S)$ [4]. In our case $M=O(|V|)$, $N=O(|E_0|)$. With regard to S, we have that $S=O(|V|)$, using Fact 1. Therefore, the time complexity of the whole Algorithm KLT′ is $O(|V|)$.

Finally, we show how to construct a k-line planar embedding for a k-line planar hierarchy, starting from the PQ-tree π produced by Algorithm KLT′. Assume to have the final version of π. We indicate with f_i a leaf of T. Let $f_1, f_2, \cdots, f_m$, be one among the feasible orderings of the m leaves of T (e.g., obtained with a DFS traversal of π). Suppose to have a stack P_i $(i=1,2,\cdots,k)$ for each subset S_i of G to record the order of the vertices on the corresponding line. Let $s(v)$ be a Boolean function, true if v has been inserted in the stack corresponding to its subset.

The following algorithm TreeEmbed accepts as input an ordering $f_1, f_2, \cdots, f_m$ of the m leaves of T and assigns to each vertex of G a value of function ORD($\cdot$).

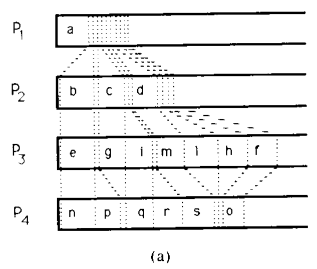
```
Algorithm TreeEmbed($f_1, f_2, \cdots, f_m$):
BEGIN
 FOR EACH subset $S_i$ DO
   $P_i$ := the empty stack;
 FOR EACH vertex $v \in G$ DO
   $s(v)$ := false;
 push the root $r$ of $T$ in $P_1$;
 $s(r)$ := true;
 FOR $j$ := 1 TO $m$ DO
   BEGIN
   push $f_j$ in $P_{L(f_j)}$;
   $s(f_j)$ := true;
   Bubble(FATHER($f_j$)); (*visit recursively $f_j$'s ancestors*)
   END;
 FOR $j$ := 1 TO $k$ DO
   BEGIN
   $t := |P_j|$;
   REPEAT
     pop vertex $v$ from $P_j$;
     ORD($v$) := $t$;
     $t := t-1$;
   UNTIL $t = 0$;
   END;
 END.
```

Procedure Bubble completes the description of Algorithm TreeEmbed.

```
Procedure Bubble($v$);
BEGIN
IF not $s(v)$,
THEN BEGIN
  push vertex $v$ in $P_{L(v)}$;
  $s(v)$ := true;
  Bubble(FATHER($v$));
  END;
END.
```

It is straightforward to see that due to Property 1 the ordering of vertices built by Algorithm TreeEmbed specifies a k-line planar embedding of G. Notice that the time complexity of Algorithm TreeEmbed is $O(|V|)$.

In Fig. 17 we show the application of Algorithm TreeEmbed to the ordering of leaves obtained by a DFS traversal of the PQ-tree of Fig. 16(i). In Fig. 17(a) we show the stacks after the last push and before the first pop. In Fig. 17(b) the k-line planar embedding obtained for the hierarchy of Fig. 16(a) is shown.



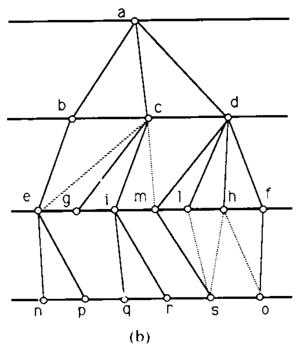


Fig. 17. Application of Algorithm TreeEmbed to hierarchy of Fig. 16(a).

V. A Characterization of Planar Hierarchies

In this section we provide a combinatorial characterization of k-line planar hierarchies. First we deal with the problem of determining which is the maximum number of edges in a k-line planar hierarchy, then we show necessary and sufficient conditions for k-line planarity in terms of forbidden patterns.

The problem of determining the maximum number of edges in a k-line planar hierarchy can be stated as follows. Given a set V of vertices find a function L and a set E of edges such that the hierarchy $G(V,E,L)$ is k-line planar and $|E|$ is maximum.

As noticed earlier we have from the graph theory that $|E| \leqslant 3|V|-6$, but it is interesting to determine a stricter bound for $|E|$. Suppose now to have a directed spanning tree T of G. We denote with E_{0i} the set of edges of E_0 starting from the vertices in S_i.

Lemma 3: For a k-line planar hierarchy $|E_{0i}| \leqslant |S_i|-1$.

Proof: Suppose Γ is a k-line planar embedding of G. Let S_i' be the set of vertices of S_i with exactly one outgoing edge in E_0 and without outgoing edges in E_t. Consider now any two consecutively ordered vertices x and y in $S_i - S_i'$. Let $X = \{x\} \cup$ SONS(x) and $Y = \{y\} \cup$ SONS(y). Due to the properties of a k-line planar embedding there exists at most one edge in E_0 connecting a vertex in X and a vertex in Y. The number of such edges is $|S_i - S_i'|-1$. The lemma follows from the fact that the vertices in S_i' have only one outgoing edge in E_0.

We are able now to count the edges in E_0. Clearly, this is significant only if $k \geqslant 3$, i.e., there exists at least one inner level.

Theorem 4: If $G(V,E,L)$ is a k-line planar hierarchy, then $|E| \leqslant 2|V|-4$.

Proof: Clearly $|E| = |E_t| + |E_0|$, and $|E_t| = |V|-1$. We can say by Lemma 3 that

$$|E_0| = \sum_{i=1}^{k} |E_{0i}| = \sum_{i=2}^{k-1} |E_{0i}| \leqslant \sum_{i=2}^{k-1} (|S_i|-1).$$

However, we have the following upper bounds:

$$\sum_{i=2}^{k-1} |S_i| \leqslant |V|-2, \quad \text{and} \quad -\sum_{i=2}^{k-1} 1 = -(k-2) \leqslant -1,$$

and it follows that

$$|E_0| \leqslant |V|-3.$$

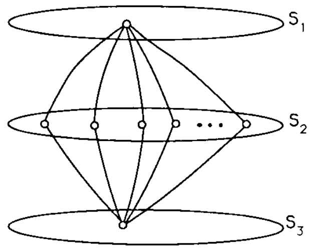


Fig. 18. k-line planar hierarchy with maximum number of edges.

We can now conclude

$$|E|=|E_t|+|E_0|\leqslant|V|-1+|V|-3=2|V|-4.$$

Note that the bound of Theorem 4 is tight. In fact, for each value of $|V|\geqslant 3$ there exists at least one hierarchy with $k\geqslant 3$ such that $|E|=2|V|-4$. Such a hierarchy has three subsets, $|S_1|=1$, $|S_3|=1$, $|S_2|=|V|-2$, and all the vertices belonging to S_2 are connected to the vertex of S_3 (see Fig. 18). This completes the numerical part of the characterization.

In the rest of the section we state that the conditions of Theorem 1 are also necessary for k-line planarity. We give here only a sketch of the proof because it involves many technical details. The complete proof is reported in Appendix II.

Theorem 5: A hierarchy $G(V, E, L)$ is not k-line planar if and only if it satisfies one of the conditions of Theorem 1.

Proof sketch: The "if" part has been proved in Theorem 1. The sketch of the "only if" part follows.

Consider a hierarchy $G(V, E, L)$ that is not k-line planar and apply to it Algorithm KLT. According to Theorem 2, Algorithm KLT stops when the set Σ of all the possible k-line planar embeddings is empty.

Let (r,s) be the edge of E_0 consider during the last step, $L(r)=i$ and $L(s)=i+1$, and let (w,z) be one of the edges crossing (r,s). Let R be an embedding that was contained in Σ at the beginning of the last step. Assume that (w,z) belongs to E_t. Let x be the lowest common ancestor in T of r and w, and y be the lowest common ancestor in T of s and w, and suppose $L(x)>L(y)$.

The situation is depicted in Fig. 19(a). The crossing is due to the fact that w is placed between r and FATHER(s). We can say in this case that w is "constrained" between r and FATHER(s). If there were an embedding R' in which w is not between r and FATHER(s), we could add to R' edge (r,s) without crossings. However, in this case we would have R' in the next step of the algorithm.

The point now is to understand why such an embedding R is constrained. One possible answer is that there exists a link between the paths $(x,\cdots,z)$ and $(y,\cdots,s)$; see Fig. 19(b). It is not difficult to find in the figure the pattern of the condition 3 of Theorem 1, in this way proving the thesis.

VI. Conclusion

A graph-theoretic approach to the planarity problem for hiearchies has been presented. A linear time algorithm for testing the planarity of a hierarchy and a combinatorial characterization of the class of hierarchies that admit a planar representation has been proposed. The algorithm has been implemented on an IBM PC-AT using Pascal language.

Future research will be focused on the following topics.

• A direct extension of the present work is to study how theorems and algorithms may be applied to a more general class of hierarchies, in which sources are not restricted to belong to the first subset.

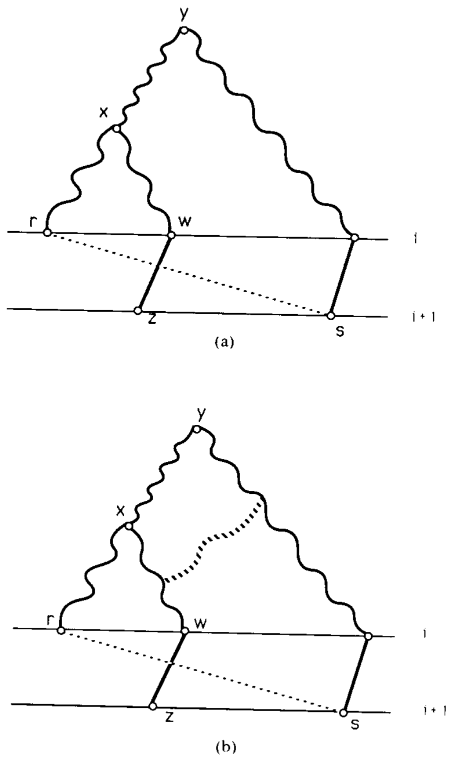


Fig. 19. Example for proof sketch of Theorem 5.

• If a hierarchy is not k-line planar, it is interesting to study heuristic approaches for the problem of minimizing crossings based on the planarity testing algorithm. In fact, Algorithm KLT (introduced in Section III) can be modified in the following way.

```
Algorithm KLT($G,\Sigma$):
1) Choose any directed spanning tree $T$ of $G$.
2) Determine the set $\Sigma$ of all the possible $k$-line planar embeddings for $T$.
3) For each edge $e$ belonging to $E_0$ do:
   if after the addition of $e$ to all the embeddings in $\Sigma$ there is
     at least one embedding that remains planar,
     then
     • add the edge $e$ to each embedding in $\Sigma$, and
     • remove from $\Sigma$ all the embeddings that are not $k$-line
       planar after the addition of $e$,
   else
     • mark $e$.
```

At the end of the algorithm, Σ contains all planar embeddings of G that are planar and maximal (in the sense that no other edges can be added to G without missing planarity). At this point marked edges can be introduced back into such embeddings, using a local optimization strategy similar to the one in [3].

Appendix I

The PQ-tree data structure was introduced in [4] to solve the problem of finding all the feasible permutations of a set U of objects, under the restriction that the objects of certain subsets $N_1,\cdots,N_n$ ($N_i\subset U$, $i=1,2,\cdots,n$) occur as consecutive subsequences. A PQ-tree consists of P-nodes, Q-nodes, and leaves. Given the set $U=\{u_1,\cdots,u_m\}$, the family of PQ-trees over U is defined to be all rooted ordered trees [1] whose leaves are elements of U and whose internal nodes are distinguished as being either P-nodes or Q-nodes [4]. Every P-node has at least

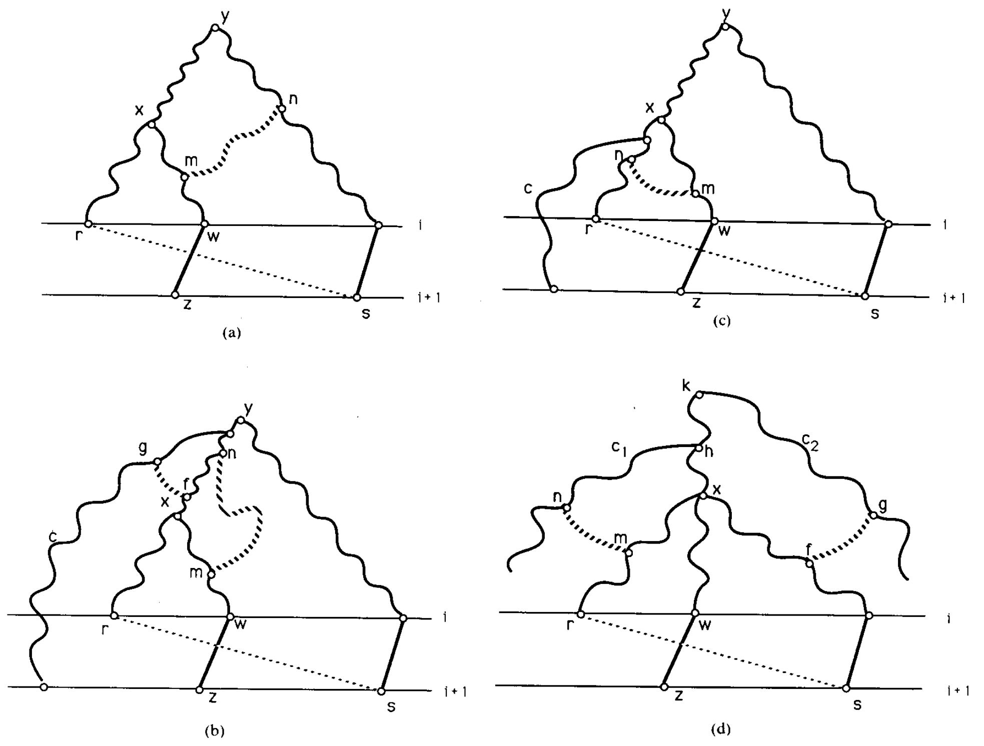


Fig. 20. Examples for proof of Theorem 5. (a) Case a1.1. (b) Case a2.2. (c) Case a3. (d) Case c2.

two sons and every Q-node has at least three sons. Typically, a P-node is drawn as a circle and a Q-node as a rectangle.

A permutation of the objects of U is represented by the ordering of the leaves of a PQ-tree π (i.e., the frontier of π) which is obtained by a traversal of π with a DFS technique. Two PQ-trees of the same family are said to be equivalent if and only if one can be obtained from the other by a finite sequence of equivalence transformations; the only admitted equivalence transformations are to 1) reverse the sons of a Q-node, and 2) permute arbitrarily the sons of a P-node.

Each PQ-tree of an equivalence class has a different frontier and represents a different permutation. Starting from a set U of objects and from a collection of n sets $N_1, \cdots, N_n$ of objects that have to appear consecutively in the permutations of U, in [4] is shown an algorithm that

1) produces the equivalence class of PQ-trees (if one exists) whose frontiers correspond to all the permutations that satisfy the constraints expressed by $N_1, \cdots, N_n$;
2) if no class exists, then stops; and
3) runs in time $O(m+n+S)$, where S is the sum of the sizes of the sets N_i.

In the correspondence we refer to an equivalence class of PQ-trees by means of a PQ-tree which is a member of the class.

Appendix II

Theorem 5: A hierarchy $G(V, E, L)$ is not k-line planar if and only if it satisfies one of the conditions of Theorem 1.

Proof: The "if" part has been proved in Theorem 1. Now we deal with the "only if" part.

Consider a hierarchy $G(V, E, L)$ that is not k-line planar and apply to it Algorithm KLT. Moreover, suppose that the edges of E_0 are ordered by the increasing level of the start vertex, and suppose that they are considered during Algorithm KLT following this ordering.

According to Theorem 2, Algorithm *KLT* stops when the set Σ of all the possible k-line planar embeddings is empty. Let (r, s) be the edge of E_0 considered during the last step, $L(r) = i$ and $L(s) = i+1$, and let (w, z) be one of the edges crossing (r, s).

If m is an ancestor of n in T, we indicate with $t(m, n)$ the path that connects m and n in T. If f and g are two vertices of G, we indicate with $q(f, g)$ a path that connects f and g and contains at least one edge of E_0.

In the following we say that a path c of G is a BARRIER$(i+1)$ if HEIGHT$(c) \leqslant i$, DEPTH$(c) = i+1$, and one of the following statements is true.

a) c is made up entirely of edges belonging to E_t;
b) c is made up of edges belonging to E_t (possibly none) but for the bottommost edge.

We shall also use notations $T(m, n)$, $T(m, n]$, $T[m, n)$, $T[m, n]$ to denote the set of vertices belonging to path $t(m, n)$ according to conventions for open and closed intervals. Namely, a square bracket indicates that the extremal vertex is included in the set and a round bracket indicates that the extremal vertex is not included in the set. In a similar way, given a generic path c in T, we shall indicate with $(c]$ the set of vertices of the path but for TOP(c).

Let R be an embedding that was contained in Σ at the beginning of the last step.

1) Assume that (w, z) belongs to E_t. Let x be the lowest common ancestor in T of r and w, and y be the lowest common ancestor in T of s and w. Three cases are possible.

a) $L(x) > L(y)$
b) $L(x) < L(y)$
c) $x = y$.

G. Ausiello and C. Batini for their continuous encouragement during this research.

References

[1] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.
[2] C. Batini, L. Furlani, and E. Nardelli, "What is a good diagram? A pragmatic approach," in *IEEE Proc. 4th Int. Conf. Entity-Relationship Approach*, Chicago, IL, 1985, pp. 312–319.
[3] C. Batini, E. Nardelli, and R. Tamassia, "A layout algorithm for data-flow diagrams," *IEEE Trans. Software Eng.*, vol. SE-12, no. 4, pp. 538–546, 1986.
[4] K. S. Booth and G. S. Lueker, "Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-tree algorithms," *J. Comput. Syst. Sci.*, vol. 13, pp. 335–379, 1976.
[5] M. Carpano, "Automatic display of hierarchized graphs for comuter aided decision analysis," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-10, no. 11, pp. 705–715, 1980.
[6] P. Eades, B. D. McKay, and N. C. Wormald, "On an edge crossing problem," in *Proc. 9th Australian Computer Science Conf.*, Canberra, 1986, pp. 327–334.
[7] S. Even, *Graph Algorithms*. Potomac, MD: Pittman, Computer Science Press, 1979.
[8] J. Hopcroft and R. Tarjan, "Efficient planarity testing," *J. Ass. Comput. Mach.*, vol. 21, no. 4, pp. 549–568, 1974.
[9] D. S. Johnson, "The *NP*-completeness column: An ongoing guide," *J. Algorithms*, vol. 3, p. 97, 1982.
[10] K. Kuratowski, "Sur le probleme des courbes gauches en topologie," *Fund. Math.*, vol. 15, pp. 217–283, 1930.
[11] A. Lempel, S. Even, and I. Cederbaum, "An algorithm for planarity testing of graphs," in *Proc. Theory of Graphs, Int. Symp.*, Rome, Italy, 1966, pp. 215–232.
[12] E. Reingold and J. Tilford, "Tidier drawing of trees," *IEEE Trans. Software Eng.*, vol. SE-7, no. 2, pp. 223–228, 1981.
[13] L. Rowe, M. Davis, E. Messinger, C. Meyer, C. Spirakis, and A. Tuan, "A browser for directed graphs," Electrical Engineering and Computer Science Dept., Univ. of California, Berkeley, manuscript, 1986.
[14] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for visual understanding of hierarchical systems," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, no. 2, pp. 109–125, 1981.
[15] R. Tamassia, G. Di Battista, and C. Batini, "Automatic graph drawing and readability of diagrams," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-18, no. 1, pp. 61–79, 1988.
[16] R. Tarjan, "Depth first search and linear graph algorithms," *SIAM J. Computing*, vol. 2, pp. 146–160, 1972.
[17] N. Tomii, Y. Kambayashi, and S. Yajima, "On planarization of 2-level graphs," Papers of Tech. Group on Electronic Computers, IECEJ, EC-77-38, 1977.
[18] J. N. Warfield, "On arranging elements of a hierarchy in graphic form," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-3, no. 2, pp. 121–131, 1973.
[19] J. N. Warfield, "Crossing theory and hierarchy mapping," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-7, no. 7, pp. 502–523, 1977.
[20] D. Wilson, "Forms of hierarchy: A selected bibliography," *Gen. Syst.*, vol. 14, pp. 3–15, 1969.

Use of Attributed Grammars for Pattern Recognition of Evoked Potentials

I. BRUHA AND G. P. MADHAVAN, MEMBER, IEEE

Abstract —**A pattern recognition system for classifying brain-stem auditory evoked potential is described. A string of terminal symbols, as a formal representation of the evoked potential waveform, is processed by a regular attributed grammar. Its semantic functions return a list of numeric features that can be processed by a simple statistical classifier. Implementation of attributed grammars is discussed.**

Manuscript received November 3, 1987; revised July 29, 1988. This work was supported in part by NSERC Grant A8034.

The authors are with the Department of Computer Science and Systems and the Department of Medicine, McMaster University, Hamilton, ON, Canada L8N 3Z5.

IEEE Log Number 8824018.

I. Introduction

Classification of brain-stem auditory evoked potential has become a major component of diagnosis in neurology [1]. The quantitative guidelines in this classification procedure are mainly the latencies of evoked potential peaks. In present-day neurological practice, the technician identifies these peaks by visual inspection. The idea of using syntactic recognition for waveform analysis has been described and utilized many times, and it seems to be one of the most powerful approaches to the waveform recognition; see, e.g., [3]–[6]. We describe one of the steps in the process of automating pattern recognition and classification of evoked potential peaks. The entire process consists of four steps (see Fig. 1):

- filter processing;
- formal description of the input waveform (evoked potential) by a string of symbols;
- syntactic analysis;
- statistical classification.

We present a new method for syntactic analysis. The details of the other three steps and their engineering and medical significance were discussed in our earlier paper [2]. Here we focus on the "syntactic analysis" step and present a new grammar that is more rigorous and a parser that is more general and flexible in its implementation. We will be using all terms and definitions in conformity with [3].

II. Syntactic Analysis of Evoked Potentials by Attributed Grammars

The preprocessed evoked potential waveform is segmented in fixed-distance intervals (100 μs), and each interval is described by one of three (terminal) symbols: u for upward slope, d for downward, and f for a flat line.

First of all, we introduce a new grammar $G1$ that is able to identify the start of the relevant waveform, i.e., to skip the artifactual data in the beginning. On the basis of experiments, we have found that the first ten symbols must be eliminated, and then the relevant waveform starts as soon as the substring uu or uf occurs. The grammar $G1$ is therefore defined as

$$G1 = [V_{N1}, V_T, P_1, T0]$$

where V_{N1} is the alphabet of nonterminals, V_T is the alphabet of terminals, P_1 is the set of rewrite rules, and $T0$ is the initial nonterminal:

$$V_{N1} = \{T0, T1, \cdots, T11\}$$
$$V_T = \{u, d, f\}$$

$$\begin{array}{ll} P_1 : T(i-1) \rightarrow x\,Ti & \text{for all } x \in V_T,\ i = 1, \cdots, 10 \\ T10 \rightarrow d\,T10 & T10 \rightarrow f\,T10 \\ T10 \rightarrow u\,T11 & \\ T11 \rightarrow d\,T10 & \\ T11 \rightarrow u & T11 \rightarrow f. \end{array}$$

The language L_1 generated by $G1$ consists of strings

$$x^{10}\,Xuu, \quad x^{10}\,Xuf$$

where $x \in V_T$, and $X \in V_T^*$ does not contain uu or uf as its substring.

The other grammar, $G2$, can identify one hill. A hill is composed of 1) a leading edge that is formed by u followed by a sequence of u's or f's in any combination, and 2) a trailing edge that is formed by d followed by any combination of d's and f's.